1 Newton Polynomial Interpolation

1.1 Collocation

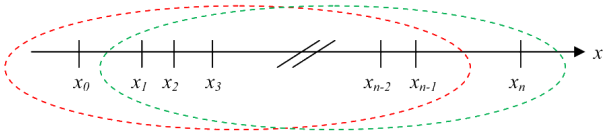
Collocation = All data points are hit by the approximating function (e.g., polynomial). The polynomial

$$y(x) = p(x) = c_0 + c_1x^1 + c_2x^2 + \ldots + c_mx^m \qquad \text{with} \quad y(x_k) = p(x_k) = y_k \quad (k = 0, 1, \ldots, n)$$

results in a linear system of $n + 1$ equations.

1.2 Aitken-Neville Recursion Formula

Overlapping polynomial interpolation functions $f_1 = p_{0,1,\ldots,n-1}$, $f_2 = p_{1,\ldots,n}$ at the data points $x_1, \ldots, x_{n-1}$ can be combined.



$$p(x) = p_{0,1,\ldots,n}(x) = \frac{(x - x_0) \cdot \overbrace{p_{1,2,\ldots,n}}^{\text{green}}(x) - (x - x_n) \cdot \overbrace{p_{0,1,\ldots,n-1}}^{\text{red}}(x)}{(x_n - x_0)}$$

1.3 Newton Polynomials

Properties

+ Embedded (When a data point is added, not everything needs to be recalculated)
+ A single formula
+ Practical: Measurements are uniformly distributed.
− Runge's phenomenon (Oscillations at the edges)
− Computationally expensive

$$\begin{aligned}
y_0 &= a_0 \cdot \pi_0 \\
&\downarrow \\
y_1 &= a_0 \cdot \pi_0 + a_1 \cdot \pi_1 \\
&\downarrow \quad \downarrow \\
y_2 &= a_0 \cdot \pi_0 + a_1 \cdot \pi_1 + a_2 \cdot \pi_2 \\
&\vdots \ \downarrow \quad \searrow \quad \searrow \\
y_n &= a_0 \cdot \pi_0(x_n) + a_1 \cdot \pi_1(x_n) + a_2 \cdot \pi_2(x_n) + \ldots + a_n\pi_n(x_n)
\end{aligned}$$

$$\begin{aligned}
\pi_0 &= 1 \\
\pi_1 &= (x - x_0) \\
\pi_2 &= (x - x_0)(x - x_1) \\
&\vdots \\
\pi_n &= (x - x_0)(x - x_1)\ldots(x - x_{n-1}) \\
\pi_{n+1} &= (x - x_0)(x - x_1)\cdots(x - x_{n-1})(x - x_n)
\end{aligned}$$

Approximation of a "measurement data set" (Messreihe) with polynomials:

$$y(x) \approx p(x) = a_0\pi_0(x) + a_1\pi_1(x) + \ldots + a_n\pi_n(x)$$

1.4 Divided Differences

Note:

$a_{p-1} = y(x_0, x_1, x_2, ..., x_{p-1})$ belongs to $\pi_{p-1} = (x - x_0) \cdot (x - x_1) \cdot ... \cdot (x - x_{p-2})$

More elegant and faster resolution of Newton polynomials.

$$y(x_\mathbf{0}, x_1, \ldots x_\mathbf{k}) = \frac{y(x_\mathbf{1}, x_2, \ldots x_\mathbf{k}) - y(x_\mathbf{0}, x_1, \ldots x_\mathbf{k-1})}{(x_k - x_0)} \quad (k = 0, 1, \ldots, n)$$

$$a(x_\mathbf{0}, \ldots, x_\mathbf{n}) = \frac{a(x_\mathbf{1}, \ldots, x_\mathbf{n}) - a(x_\mathbf{0}, \ldots, x_\mathbf{n-1})}{x_n - x_0}$$

$k = 0$ $\quad y(x_0)$

$k = 1$ $\quad \frac{y(x_1) - y(x_0)}{(x_1 - x_0)}$

$k = 2$ $\quad \frac{y(x_1, x_2) - y(x_0, x_1)}{(x_2 - x_0)}$

$k = 3$ $\quad \frac{y(x_1, x_2, x_3) - y(x_0, x_1, x_2)}{(x_3 - x_0)}$

Symmetry: $y(x_0, x_1) = y(x_1, x_0) \Longrightarrow y(x_0, x_1, ..., x_n)$ is independent of the order of x-values! (The last difference a_n matches, the other differences differ.)

Example:

	x_k	y_k			
x_0	0	(1) a_0			
			(0) a_1		
x_1	1	1		($\frac{1}{2}$) a_2	
			1		($-\frac{1}{12}$) a_3
x_2	2	2		$\frac{1}{6}$	
			$\frac{3}{2}$		
x_3	4	5			

$$y(x) \approx p(x) = a_0 \cdot \pi_0(x) + a_1 \cdot \pi_1(x) + a_2 \cdot \pi_2(x) + a_3\pi_n(x)$$
$$= 1 \cdot \pi_0(x) \ + 0 \cdot \pi_1(x) \ + \frac{1}{2} \cdot \pi_2(x) \ - \frac{1}{12}\pi_n(x)$$

$$y(x) \approx p(x) = 1 + \frac{1}{2}(x - x_0)(x - x_1) - \frac{1}{12}(x - x_0)(x - x_1)(x - x_2)$$
$$= 1 + \frac{1}{2}(x - 0)(x - 1) \quad - \frac{1}{12}(x - 0)(x - 1)(x - 2)$$

1.5 Collocation Error Formula

$$y(x) - p(x) = \frac{y^{(n+1)}(\xi)}{(n+1)!}(x - x_0)(x - x_1)\dots(x - x_{n-1})(x - x_n) = \frac{y^{(n+1)}(\xi)}{(n+1)!}\pi_{n+1}(x) \quad \xi \in (\min(x_i), \max(x_i))$$

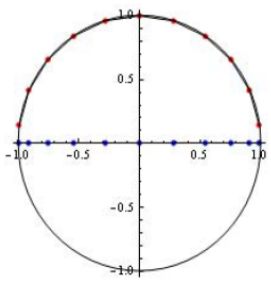
Derivation: $y = p + \frac{y^{(n+1)}(\xi)}{(n+1)!} \cdot \pi_{n+1} = y(x_0, x_1, ..., x_n) + \frac{y^{(n+1)}(\xi)}{(n+1)!} \cdot \pi_{n+1} \Longrightarrow y(x_0, x_1, ..., x_{n+1}) = \frac{y^{(n+1)}(\xi)}{(n+1)!}$
The error term can be interpreted as a derivative. However, it is not clear where the derivative is evaluated in the interval $[x_0, x_n]$, as ξ is not known!

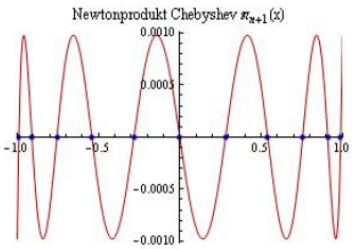
1.6 Runge Phenomenon and Chebyshev Arguments

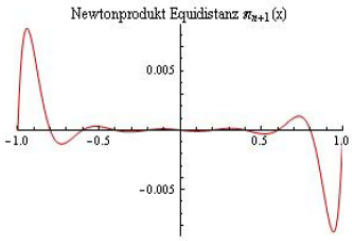
Runge Phenomenon: Potential oscillations at the definition boundaries of polynomial collocations. Caused by high n and high $\frac{y^{(n+1)}(x)}{(n+1)!}$.
Chebyshev Arguments: Adjustment of measurement points through a different distribution (no longer uniformly distributed x_k, but uniformly distributed on the unit circle)

$$x_k = \cos(\frac{2k+1}{2(n+1)}\pi), \quad (k = 0, 1, \dots, n)$$

results in maximum amplitude of $\max_{-1 \le x \le 1} |\pi_{n+1}(x)| = \frac{1}{2^n}$. With an affine transformation from the unit interval $[-1, 1]$ to $[a, b]$: $x \to a + \frac{b-a}{2}(x + 1)$

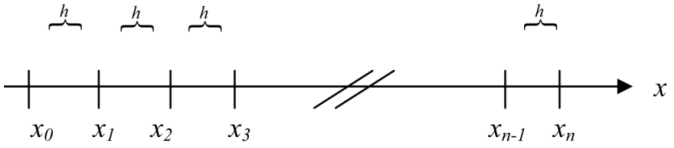






Properties
+ No Runge Phenomenon (oscillations at the edge)
+ A single formula
- "Not embedded" (new measurements mean new polynomial calculation!)
- Practicality: Measurements are often only uniformly possible (limited edge resolution)
- The range of interpolation is fixed (poor extrapolation outside the edge)

1.7 Uniformly Distributed Arguments



$$x_k = x_0 + kh \qquad k = 0, 1, 2, \ldots n$$

$$\boxed{y(x_0, x_1, \ldots, x_k) = \frac{\Delta^k y_0}{h^k k!}}$$

$$p(x) = y_0 + \frac{\Delta^1 y_0}{h^1 1!}\pi_1(x) + \frac{\Delta^2 y_0}{h^2 2!}\pi_2(x) + \ldots + \frac{\Delta^n y_0}{h^n n!}\pi_n(x) = \sum_{k=0}^{n} \frac{\Delta^k y_0}{h^k k!}\pi_k(x)$$

Δ^0	y_0	y_1	y_2	y_3	y_4
Δ^1	$y_1 - y_0$ Δy_0	$y_2 - y_1$ Δy_1	$y_3 - y_2$ Δy_2	$y_4 - y_3$ Δy_3	
Δ^2	$\Delta y_1 - \Delta y_0$ $\Delta^2 y_0$	$\Delta y_2 - \Delta y_1$ $\Delta^2 y_1$	$\Delta y_3 - \Delta y_2$ $\Delta^2 y_2$		

1.8 Taylor Series

$$f(x) = \underbrace{y_0 + \frac{y'(x_0)}{1!}(x - x_0) + \frac{y''(x_0)}{2!}(x - x_0)^2 + \ldots + \frac{y^{(n)}(x_0)}{n!}(x - x_0)^n}_{\text{Taylor Polynom p(x)}} + \underbrace{\frac{y^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}}_{\text{Lagrange Fehlerterm}}$$

2 Hermite Interpolation (Osculation)

Extension of divided differences: Now, derivatives at data points are also possible as conditions.
From the definition of $y(x)$, we have $y(x_0, x_0) = y'(x_0)$. Generalized, this yields

$$y(\underbrace{x_0, \ldots x_n, x_{n+1}}_{(n+2)}) = \frac{y^{(n+1)}(\xi)}{(n+1)!}$$

$$y(\underbrace{x_0, \ldots x_0}_{(n+1)}) = \lim_{\xi \to x_0} \frac{y^{(n)}(\xi)}{(n)!} = \frac{y^{(n)}(x_0)}{n!}$$

Bsp. HS13/14

$$y(x_0, x_1, x_1) = \frac{y(x_1, x_1) - y(x_1, x_0)}{x_1 - x_0} = \frac{y'(x_1) - \frac{y(x_1) - y(x_0)}{x_1 - x_0}}{x_1 - x_0}$$

For Hermite interpolation, the same Newton tables are used, but with repetitions (see the example). The values marked in red are calculated as usual.
It is important that no gaps occur in the calculation of the divided differences (derivatives must be continuously available, e.g., y', y'', y''' good; y', y''' bad)! Otherwise, the system of equations is unsolvable. In case of indeterminate results, a variable can be introduced, which can be determined at the end.

x	y		
$x_0 = 2$	$y(x_0) = 1$		
		$\frac{y^{(1)}(x_0)}{1!} = 1$	
$x_0 = 2$	$y(x_0) = 1$		$\frac{y^{(2)}(x_0)}{2!} = 0$
		$\frac{y^{(1)}(x_0)}{1!} = 1$	
$x_0 = 2$	$y(x_0) = 1$		
$x_1 = 4$	$y(x_1) = 2$		
		$\frac{y^{(1)}(x_1)}{1!} = 0$	
$x_1 = 4$	$y(x_1) = 2$		$\frac{y^{(2)}(x_1)}{2!} = 0$
		$\frac{y^{(1)}(x_1)}{1!} = 0$	
$x_1 = 4$	$y(x_1) = 2$		

x	y				
2	(1) a_0				
		(1) a_1			
2	1		(0) a_2		
		1		(−1/8) a_3	
2	1		−1/4		(1/16) a_4
	1/2		0		(0) a_5
4	2		−1/4		1/16
		0		1/8	
4	2		0		
		0			
4	2				

Modified Newton polynomials are used. In this case:

$\pi_0 = 1$	$\pi_4 = (x-x_0)(x-x_0)(x-x_0)(x-x_1)$
$\pi_1 = (x-x_0)$	$\pi_5 = (x-x_0)(x-x_0)(x-x_0)(x-x_1)(x-x_1)$
$\pi_2 = (x-x_0)(x-x_0)$	$\pi_6 = (x-x_0)(x-x_0)(x-x_0)(x-x_1)(x-x_1)(x-x_1)$
$\pi_3 = (x-x_0)(x-x_0)(x-x_0)$	

$$
\begin{aligned}
p_2(x) &= a_0 \cdot \pi_0 + a_1 \cdot \pi_1 + a_2 \cdot \pi_2 + a_3 \cdot \pi_3 + a_4 \cdot \pi_4 + a_5 \cdot \pi_5 \\
&= a_0 \cdot 1 + a_1 \cdot (x-x_0) + a_2 \cdot (x-x_0)^2 + a_3 \cdot (x-x_0)^3 + a_4 \cdot (x-x_0)^3(x-x_1) + a_5 \cdot (x-x_0)^3(x-x_1)^2 \\
&= 1 + (x-2) - \frac{1}{8}(x-2)^3 + \frac{1}{16}(x-2)^3(x-4)
\end{aligned}
$$

2.3 Error Formula

$$y(x) - p(x) = \frac{y^{(d)}(\xi)}{d!}(x-x_0)^{d_0}(x-x_1)^{d_1}\cdots(x-x_n)^{d_n}$$

$$x, \xi \in (\min x_i, \max x_i), \quad i = 0, 1, \ldots, n$$

d is the total number of conditions, and d_i is the number of conditions per support point x_i. ($d = \sum_{i=0}^{n} d_i$)

2.4 Missing Derivatives

In case of missing derivative specifications, variables are introduced, and the tableau is computed with these. In the end, the missing variables are determined from the back.

Example:

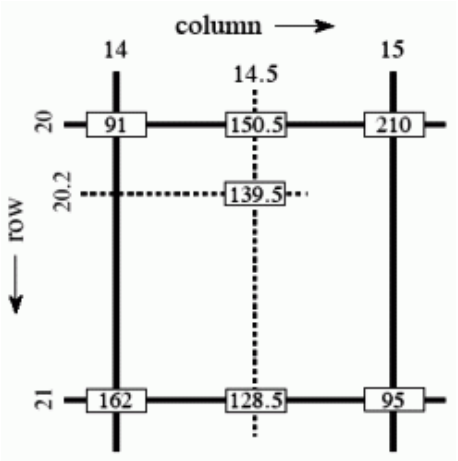
Desired: 2nd order polynomial passing through points $y(2) = 1$ and $y(4) = 1$ with the derivative $y'(4) = 1$.

x	y			
2	$\textcircled{1}\, a_0$			
		$\textcircled{\beta}\, a_1$		
2	1		$\textcircled{-\frac{\beta}{2}}\, a_2$	
		0		$\textcircled{\frac{1+\beta}{4}}\, a_3 \overset{!}{=} 0$
4	1		$\frac{1}{2}$	
		1		
4	1			

- Because the polynomial must have order $d = 2$, we have $\frac{1+\beta}{4} = 0$. From this, we get $\beta = -1$.
- The solution polynomial is:

$$
\begin{aligned}
p(x) &= a_0 + a_1(x-x_0) + a_2(x-x_0)^2 + a_3(x-x_0)^2(x-x_1) \\
&= 1 + \beta(x-x_0) - \frac{\beta}{2}(x-x_0)^2 + \frac{1+\beta}{4}(x-x_0)^2(x-x_1)\Big|_{\beta=-1} \\
&= 1 - (x-2) + \frac{1}{2}(x-2)^2
\end{aligned}
$$

3 Multi-variate Polynomial Interpolation



- The basis of polynomials can be calculated through the *2-fold tensor product* $\Rightarrow$ Bilinear Interpolation
  $\{1, x, x^2\} \otimes \{1, y, y^2\} = \{1, x, y, x^2, y^2, xy, x^2y, xy^2, x^2y^2\}$;
  This way, the number of equations n can be calculated:
  $\mathbb{R}^n \otimes \mathbb{R}^k = \mathbb{R}^{nk}$
- Interpolate the desired order (linear, quadratic, ...) in the X-direction at position y_0: $p(x, y_0)$
- Interpolate the desired order (linear, quadratic, ...) in the X-direction at position y_1: $p(x, y_1)$
- Compute the tableau for divided differences (linear case here):

y	z	
y_0	$p(x, y_0) = a_{y0}$	
y_1	$p(x, y_1)$	$\frac{p(x,y_1)-p(x,y_0)}{y_1-y_0} = a_{y1}$

- Formulate the polynomial:
  $p(x, y) = a_{y0}\pi_0(y) + a_{y1}\pi_1(y) + a_{y2}\pi_2(y) + \ldots$
- Expand the polynomial:
  $p(x, y) = a_{0,0}\pi_0(x)\pi_0(y) + a_{1,0}\pi_1(x)\pi_0(y) + a_{0,1}\pi_0(x)\pi_1(y) + a_{1,1}\pi_1(x)\pi_1(y) + \ldots$

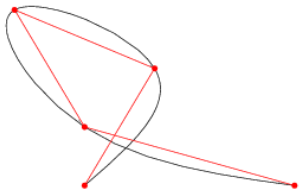
3.1 Example:

Given points $(14, 20), (14, 21), (15, 20)(15, 21)$ with values $\{91/255, 162/255, 210/255, 95/255\}$. Linear interpolation is sought. This results in the following polynomial bases: $(\{1, x\} \times \{1, y\} = \{1, x, y, xy\})$

1. Interpolation for $y = y_0 = 20$: $p(x, y_0) = p(x_0, y_0)\pi_0(x) + \frac{p(x_1,y_0)-p(x_0,y_0)}{x_1-x_0}\pi_1(x) = \frac{1}{255}\left(91 \cdot 1 + \frac{210-91}{15-14} \cdot (x - 14)\right)$
2. The same for $y = y_1 = 21$: $p(x, y_1) = \frac{1}{255}\left(162 \cdot 1 + \frac{95-162}{15-14} \cdot (x - 14)\right)$
3. Then the final interpolation: $p(x, y) = p(x, y_0)\pi_0(y) + \frac{p(x,y_1)-p(x,y_0)}{y_1-y_0}\pi_1(y)$
   $= \frac{1}{255}\left((91 \cdot 1 + \frac{210-91}{15-14} \cdot 1) \cdot 1 + \frac{162 \cdot 1 \frac{95-162}{15-14} \cdot (x-14) - 91 \cdot 1 + \frac{210-91}{15-14} \cdot (x-14)}{21-20} \cdot (y - 20)\right) = -215.98 + 15.0549x + 10.4902y - 0.7294xy$

bi-cubic und multivariate

4 Spline Interpolation



The idea of spline interpolation is not to interpolate the data with a high-degree polynomial but with several low-degree polynomials. This helps to avoid the tendency of high-degree polynomials to oscillate. At the "breakpoints,"the transitions from one patch to the next, the derivatives of the neighboring patches must match up to a specified order of derivation.

Properties

+ No Runge phenomenon (oscillations at the boundaries)
+ Polynomials have low degree
+ Computation effort lower than Newton [Spline: $O(n)$ (due to tridiagonal band matrix),Newton: $O(n^2)$]
– "Not embedded"(new measurements mean new polynomial calculation!)
– Polynomials must be composed for evaluation (post-processing effort is large)

4.1 One-Dimensional Splines

4.1.1 Principle

For each *patch* (out of a total of n), a polynomial of degree d (usually cubic, $d = 3$) is computed. Conditions C^k ($k = 1, \ldots, d-1$) can be defined at the transitions at x_i.

Degrees of Freedom n patches with 1 polynomial each with $(d+1)$ coefficients each give a total of $n(d+1)$ degrees of freedom.

Conditions n patches with start and end $= 2n$; $d-1$ derivatives per inner point $(n-1)(d-1)$ result in a total of $(n(d+1)-(d-1))$ conditions.

Additional Conditions When comparing degrees of freedom and conditions, $d-1$ conditions need to be chosen. For the *natural spline,* the second derivatives are set to 0; for *clamped splines,* the first derivatives are specified.

4.1.2 General Procedure (Cubic Splines)

The final interpolation has the following form (description of **one** patch):

$$\boxed{S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3}$$

$$S_i' = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$$
$$S_i'' = 2c_i + 6d_i(x - x_i)$$

Also needed (patch width) $h_i = x_{i+1} - x_i = \Delta x_i$

1. $a_i = y_i \qquad (i = 0, \ldots n-1)$
2. For cubic splines, there are $d - 1 = 3 - 1 = 2$ additional conditions.

$$\begin{pmatrix} h_0 & 2(h_0+h_1) & h_1 & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & \\ & & h_2 & 2(h_2+h_3) & h_3 & \\ & & & \ddots & \ddots & \ddots \\ & & & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} \\ \\ \\ 3\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right) \\ \\ \\ \end{pmatrix}_{i=1,\ldots,n-2}$$

- **Natural Spline** (Boundary curvature): $y_0'' = 0 = y_n''$ (minimizes the energy norm! ($\min \int |f''(x)|^2 \mathrm{d}x$))
  (a) $c_0 = c_n = 0$
  (b) Solve the system of equations for c_i:

$$\begin{pmatrix} 2(h_0+h_1) & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & h_2 & 2(h_2+h_3) & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \\ & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} 3\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right) \end{pmatrix}_{i=1,\dots,n-1}$$

- **Clamped Spline:**
  (a) $b_0 = y_0'; b_n = y_n'$
  (b) Solve the system of equations for c_i:

$$\begin{pmatrix} 2h_0 & h_0 & & & & & \\ h_0 & 2(h_0+h_1) & h_1 & & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & & \\ & & h_2 & 2(h_2+h_3) & h_3 & & \\ & & \ddots & \ddots & \ddots & & \\ & & & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} & \\ & & & & 2h_{n-2} & 4h_{n-2}+3h_{n-1} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} 3\left(\frac{a_1-a_0}{h_0} - y_0'\right) \\ \vdots \\ \left(3\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i-y_{i-1}}{h_{i-1}}\right)\right)_{i=1,\dots,n-2} \\ \vdots \\ 9\frac{y_n - a_{n-1}}{h_{n-1}} - 6\frac{a_{n-1}-a_{n-2}}{h_{n-2}} - 3y_n' \end{pmatrix}$$

3. $b_{i-1} = \frac{a_i - a_{i-1}}{h_{i-1}} - \frac{2c_{i-1}+c_i}{3}h_{i-1} \qquad (i = 1, \dots, n-1)$
4. $b_{n-1} = \frac{y_n - a_{n-1}}{h_{n-1}} - c_{n-1}h_{n-1} - d_{n-1}h_{n-1}^2 = \frac{y_n - y_{n-1}}{h_{n-1}} - \frac{2}{3}c_{n-1}h_{n-1}$
5. $d_{i-1} = \frac{c_i - c_{i-1}}{3h_{i-1}} \qquad (i = 1, \dots n-1)$
6. Clamped Spline: $d_{n-1} = \frac{y_n' - b_{n-1} - 2c_{n-1}h_{n-1}}{3h_{n-1}^2}$ Natural Spline: $d_{n-1} = -\frac{c_{n-1}}{3h_{n-1}}$

Further Methods
In *Hermite cubic splines*, C^1 functions are formed from patched cubic polynomials with prescribed first derivatives.
Periodic splines are C^2 functions formed from patched cubic polynomials with periodicity ($S'(x_0) = S'(x_n)$ and $S''(x_0) = S''(x_n)$).

4.1.3 Error Estimation

For cubic splines with C^2, the following error estimate applies ($H = \max h_i$ $(i = 0, \dots n-1)$):
$|y(x) - S(x)| \leq \max|y^{(4)}(x)| \frac{5}{384}H^4 \qquad |y'(x) - S'(x)| \leq \max|y^{(4)}(x)| \frac{1}{24}H^3 \qquad |y''(x) - S''(x)| \leq \max|y^{(4)}(x)| \frac{3}{8}H^2$

4.2 Bernstein-Bézier Splines (B-B Splines)

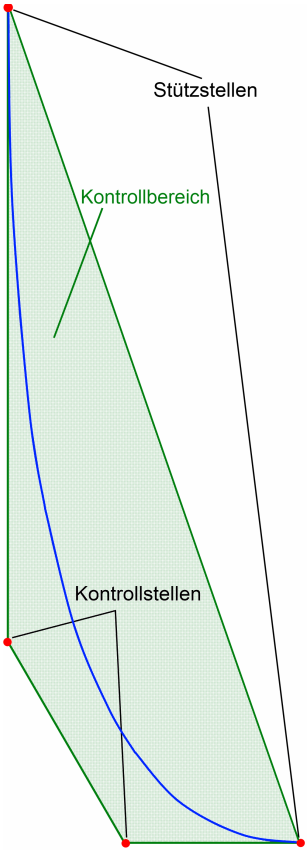
4.2.1 Bernstein Polynomials

The Bernstein polynomials are defined in the interval $t \in [0,1]$ as

$$B_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i \qquad t \in [0,1]\ i = 0,1,\ldots,n$$

with i: index and n: order of the spline.
Through an affine transformation into the interval $t \in [a,b]$, we have

$$B_{i,n}(u,a,b) = \frac{1}{(b-a)^n}\binom{n}{i}(b-u)^{n-i}(u-a)^i \qquad u \in [a,b]\ i = 0,1,\ldots,n$$



Properties Bernstein polynomials...

- form a linear basis for polynomials of order n (they can be used to build any polynomial of order n)
- have exactly one maximum at $t = \frac{i}{n}$
- have a root at 0 (order i) and at 1 (order $n-1$)
- are symmetric: $B_{i,n}(t) = B_{n-i,n}(1-t)$
- are bounded between $t \in [0,1]$ and are restricted to $[0,1]$
- sum up to: $\sum_{i=0}^{n} B_{i,n}(t) = 1$

$\frac{d}{dt}B_{i,n}(t) = n(B_{i-1,n-1}(t) - B_{i,n-1}(t)) = -n\Delta B_{i-1,n-1}(t)$
$\frac{d^2}{dt^2}B_{i,n}(t) = n(n-1)(B_{i-2,n-2}(t) - 2B_{i-1,n-2}(t) + B_{i,n-2}(t)) = n(n-1)\Delta^2 B_{i-2,n-2}(t)$
$\frac{d^k}{dt^k}B_{i,n}(t) = (-1)^k n(n-1)\ldots(n-k+1)\Delta^k B_{i-k,n-k}(t)$

Pascal's Triangle

$n=0$:				1				
$n=1$:			1		1			
$n=2$:		1		2		1		
$n=3$:	1		3		3		1	
$n=4$:	1		4		6		4		1

Bernstein Polynomials $0 \le t \le 1$

$B_{0,0}(t) = 1$
$B_{0,1}(t) = 1-t \qquad B_{1,1}(t) = t$
$B_{0,2}(t) = (1-t)^2 \qquad B_{1,2}(t) = 2t(1-t) \qquad B_{2,2}(t) = t^2$
$B_{0,3}(t) = (1-t)^3 \qquad B_{1,3}(t) = 3t(1-t)^2 \qquad B_{2,3}(t) = 3t^2(1-t) \qquad B_{3,3}(t) = t^3$
$B_{0,4}(t) = (1-t)^4 \qquad B_{1,4}(t) = 4t(1-t)^3 \qquad B_{2,4}(t) = 6t^2(1-t)^2 \qquad B_{3,4}(t) = 4t^3(1-t) \qquad B_{4,4}(t) = t^4$

4.2.2 Simple Bézier Curves

A Bézier curve is defined by control points ($\vec{P}_0, \vec{P}_1, \ldots \vec{P}_n$ ($n \ge 2$) in R^d) and the Bernstein polynomials:

$$\vec{r}(t) = \sum_{i=0}^{n} \vec{P}_i B_{i,n}(t) \quad t \in [0,1]$$

Properties

- Bézier curves always lie within the convex hull of the control points.
- $\vec{r}(0) = \vec{P}_0 \qquad \vec{r}(1) = \vec{P}_n$
  $\vec{r}'(0) = n(\vec{P}_1 - \vec{P}_0) \qquad \vec{r}'(1) = n(\vec{P}_n - \vec{P}_{n-1})$
  $\vec{r}''(0) = n(n-1)(\vec{P}_2 - 2\vec{P}_1 + \vec{P}_0) \qquad \vec{r}''(1) = n(n-1)(\vec{P}_n - 2\vec{P}_{n-1} + \vec{P}_{n-2})$
- If C^k continuity is demanded at a point, k equations or k control points per point are necessary.

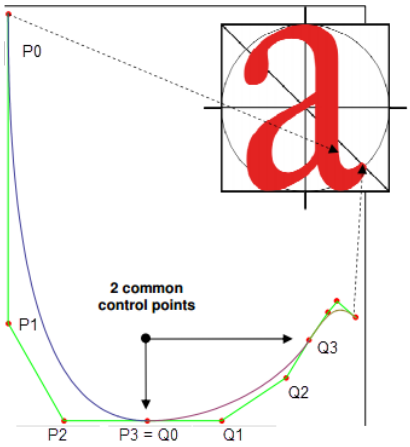
4.2.3 Casteljau Recurrence

The Casteljau recurrence is a similar idea to Neville-Aitken. It allows a point on a Bézier curve to be calculated as a linear combination of two points from Bézier curves of lower order.

$$\vec{r}_{\vec{P}_0,\vec{P}_1,\ldots,\vec{P}_n}(t) = (1-t)\cdot\vec{r}_{\vec{P}_0,\vec{P}_1,\ldots,\vec{P}_{n-1}}(t) + t\cdot\vec{r}_{\vec{P}_1,\vec{P}_2,\ldots,\vec{P}_n}(t) \qquad t\in[0,1]$$

4.2.4 Composite Bézier Curves

The composite (simple) Bézier curves should satisfy the known conditions (C^k). These are the equations to the point Q_0 with control points $Q_{1,\ldots,m-1}$ after Q_0, as well as the control points $P_{1,\ldots,n-1}$ before Q_0.



Here, $m = n = 3$

$$\begin{aligned}
C^0: &\quad \boxed{\vec{P}_n = \vec{Q}_0}\\
C^1: &\quad \vec{r}'_P(1) = \boxed{n(\vec{P}_n - \vec{P}_{n-1}) = m(\vec{Q}_1 - \vec{Q}_0)} &= \vec{r}'_Q(0)\\
C^2: &\quad \vec{r}''_P(1) = \boxed{n(n-1)(\vec{P}_n - 2\vec{P}_{n-1} + \vec{P}_{n-2}) = m(m-1)(\vec{Q}_2 - 2\vec{Q}_1 + \vec{Q}_0)} &= \vec{r}''_Q(0)\\
C^k: &\quad \vec{r}^{(k)}_P(1) = \boxed{n(n-1)\ldots(n-k+1)(\Delta^k\vec{P}_{n-k}) = m(m-1)\ldots(m-k+1)(\Delta^k\vec{Q}_0)} &= \vec{r}^{(k)}_Q(0)
\end{aligned}$$

where m is the degree of $\vec{r}_Q$, and n is the degree of $\vec{r}_P$.

4.2.5 Composite Bézier Curves with $P_{i,j}$

When $\vec{r_j}(t) = \sum_{i=0}^{n}\vec{P}_{i,j}B_{i,n}(t) \quad t\in[0,1]$ is defined with the fixed points Q_j (degree: n), the following formulas are obtained:

$$\begin{aligned}
C^0: &\quad \boxed{\vec{P}_{n,j-1} = \vec{P_{0,j}} = \vec{Q}_j}\\
C^1: &\quad \vec{r}'_{j-1}(1) = \boxed{\vec{Q}_j - \vec{P}_{n-1,j-1} = \vec{P}_{1,j} - \vec{Q}_j} &= \vec{r}'_j(0)\\
C^2: &\quad \vec{r}''_{j-1}(1) = \boxed{\vec{Q}_j - 2\vec{P}_{n-1,j-1} + \vec{P}_{n-2,j-1} = \vec{P}_{2,j} - 2\vec{P}_{1,j} + \vec{Q}_j} &= \vec{r}''_j(0)\\
C^k: &\quad \vec{r}^{(k)}_{j-1}(1) = \boxed{\Delta^k\vec{P}_{n-k,j-1} = \Delta^k\vec{P}_{0,j}} &= \vec{r}^{(k)}_j(0)
\end{aligned}$$

4.2.6 Bézier Surfaces as Tensor Splines

Bézier surfaces are formed in the same way as Bézier curves, using a tensor product of Bernstein polynomials. A surface in the interval $[0,1]\times[0,1]$ is formed by

$$\vec{z}(s,t) = \sum_{i=0}^{n}\sum_{j=0}^{m}\vec{P}_{ij}B_{in}(s)B_{jm}(t) \qquad s,t\in[0,1]$$

Details and Formulas see Skript p. 19 – 23.

4.2.7 Comparison of Bézier and Newton Interpolation with Equally Distributed Arguments on the X-Axis

From the Bézier perspective:

+ Small error, no oscillation, no Runge phenomenon (with a low degree, e.g., 3)
+ $O(n)$ (linear computation effort)
- Not embedded (new data requires recalculation of the interpolation)
- Post-processing (graphics, integration, derivatives) is more complex, as each curve segment must be considered individually.

5 Least-Squares Approximation

5.1 Linear Least-Squares

Goal: Approximation of data points by minimizing a function that indicates the deviation in y (residues). In this method, the data points are not precisely matched, and the degree m of the approximation polynomial is usually significantly smaller than the number of given data points $N+1$.

From a set of basis functions $\{g_0, g_1, \ldots, g_m\}$ with $m \ll N$ and the measurements $(x_0, y_0), (x_1, y_1), \ldots, (x_N, y_N)$, the overdetermined $(m < N)$ system of equations is set up and solved for the unknowns a_i.

$$\overbrace{\begin{pmatrix} g_0(x_0) & g_1(x_0) & \ldots & g_m(x_0) \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ g_0(x_N) & g_1(x_N) & \ldots & g_m(x_N) \end{pmatrix}}^{\text{Design Matrix } G} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ \vdots \\ y_N \end{pmatrix} \quad \Leftrightarrow \quad G \cdot a = y$$

Error function to be minimized:

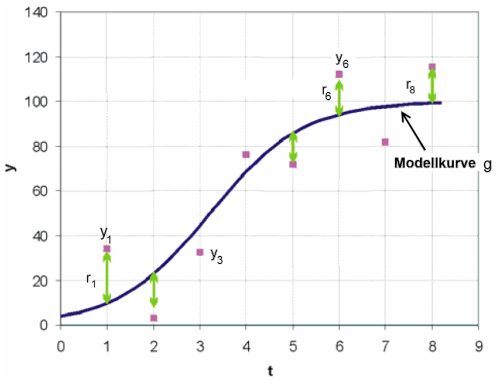
$$\underbrace{S}_{Error} = \sum_{i=0}^{N} \Big(y_i - \underbrace{\sum_{j=0}^{m} a_j g_j}_{Model} \Big)^2 = \sum_{i=0}^{N} r_i^2$$

Model function:

$$y = \sum_{j=0}^{m} a_j g_j \overset{Poly!}{=} \sum_{j=0}^{m} a_j x^j$$

Deviations (Residues):

$$r_i = y_i - \sum_{j=0}^{m} a_j g_j$$

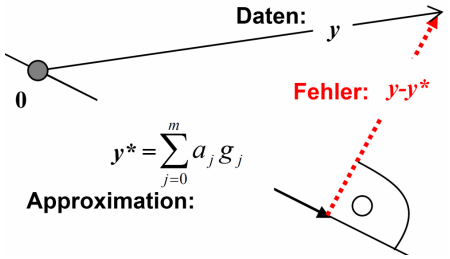


5.1.1 Normal Equation

Minimizing the quadratic error with "normal matrix" G (design matrix):

$$\underbrace{G^T G}_{NormalMatrix} \cdot a = G^T y \quad \Rightarrow \quad a = (G^T G)^{-1} G^T y$$

$$\underbrace{\underbrace{G^T}_{(m+1)\times(N+1)} \cdot \underbrace{G}_{(N+1)\times(m+1)}}_{(m+1)\times(m+1)} \cdot \underbrace{a}_{(m+1)\times 1} = \underbrace{\underbrace{G^T}_{(m+1)\times(N+1)} \cdot \underbrace{y}_{(N+1)\times 1}}_{(m+1)\times 1}$$



The symmetric $(m+1)\times(m+1)$ matrix $G^T \cdot G$ is positive definite if g_i are linearly independent. The new system of equations is regular and has a unique solution for the coefficients a_j.

$$\overbrace{\begin{pmatrix} \langle g_0|, g_0| \rangle & \langle g_1|, g_0| \rangle & \ldots & \langle g_m|, g_0| \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle g_0|, g_m| \rangle & \langle g_1|, g_m| \rangle & \ldots & \langle g_m|, g_m| \rangle \end{pmatrix}}^{G^T \cdot G} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} \langle y|, g_0| \rangle \\ \vdots \\ \vdots \\ \langle y|, g_m| \rangle \end{pmatrix}$$

5.1.2 QR Decomposition

A square $n \times n$ matrix Q is orthogonal if $Q^T \cdot Q = Q \cdot Q^T = I$ or $Q^T = Q^{-1}$. Any matrix G with dimensions $(N+1)\times(m+1)$ where $N \geq m$ and rank $m+1$ can be represented as the product of an orthogonal $(N+1)\times(N+1)$ matrix Q and an upper triangular matrix R with dimensions $(N+1)\times(m+1)$.

The equation system $Ga| = y|$ then becomes $Ra| = Q^T y|$, which is easier to solve[Citation needed].

5.1.3 Singular Value Decomposition (SVD)

Any matrix $\boldsymbol{G}$ with dimensions $(N+1)\times(m+1)$ can be decomposed as follows:

$$\boldsymbol{G} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T = \boldsymbol{U}\cdot\begin{bmatrix} d_{00} & 0 & \dots & 0 \\ 0 & d_{11} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & \dots & 0 & d_{mm} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix}\cdot\boldsymbol{V}^T$$

$\boldsymbol{D}$ $((N+1)\times(m+1))$ is the central diagonal matrix with singular values, $\boldsymbol{U}$ $((N+1)\times(N+1))$ and $\boldsymbol{V}$ $((m+1)\times(m+1))$ are orthogonal and square matrices (meaning, among other things, that $\boldsymbol{U}^T = \boldsymbol{U}^{-1}$).

The decomposition itself is not further explained here and can be looked up elsewhere. SVD can be used to solve equation systems (as here), compute eigenvectors and eigenvalues, and find inverses.
From $Ga| = y|$ we get $UDV^Ta| = y|$. This can be solved to

$$a| = VD^{-1}U^T\cdot y|$$

5.1.4 Monomials

With $g = a_0\underbrace{1}_{g_0} + a_1\underbrace{x}_{g_1} + a_2\underbrace{x^2}_{g_2} + \dots + a_m\underbrace{x^m}_{g_2}$ the Design Matrix becomes:

$$G \underbrace{=}_{\text{When Monomials}} \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^m \\ 1 & x_1 & x_1^2 & \dots & x_1^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^m \end{bmatrix}$$

5.1.5 Uniform Arguments (Orthogonality Properties)

With uniform elements $x_i - x_j = (j-i)h$ for all $i,j \to \{x_0 ... x_N\} = \{x_0 + t\cdot h\}_{t=0\dots N}$ and **orthogonal polynomials**, GG^T can be diagonalized, making the equations computationally simpler to solve.

$$p_{k,N}(t) = \sum_{i=0}^{k}(-1)^i\binom{k}{i}\binom{k+i}{i}\frac{t^{(i)}}{N^{(i)}} = 1 + \sum_{i=1}^{k}(-1)^i\binom{k}{i}\binom{k+i}{i}\frac{t(t-1)(t-2)\dots(t-i+1)}{N(N-1)(N-2)\dots(N-i+1)} \qquad (k=1,\dots,N)$$

with $t = \frac{x-x_0}{h}$ $\qquad \binom{k}{i} = \frac{k!}{i!(k-i)!} = nCr(k,i)$
$p_{k,N}$ can now be substituted as g_k into the design matrix, and the product G^TG becomes a $(m+1)\times(m+1)$ diagonal matrix. Finally, a can be calculated using the known formula $G^TGa = G^Ty$.
Example:

$$\boldsymbol{x} = \begin{bmatrix}3 & 4 & 5 & 6 & 7\end{bmatrix} \qquad \Rightarrow \qquad \boldsymbol{t} = \boldsymbol{x} - 3 = \begin{bmatrix}0 & 1 & 2 & 3 & 4\end{bmatrix}$$

$$\Rightarrow P_0(x) = 1 \qquad P_1(x) = 1 - \frac{x-3}{2} \qquad P_2(x) = 1 - \frac{2\cdot 3(x-3)}{4} + \frac{1\cdot 3(x-3)(x-3-1)}{4\cdot 3} = 1 - \frac{3x-9}{2} + \frac{1}{2}(x-4)(x-3)???$$

$$\boldsymbol{G} = \overset{P_0}{\begin{bmatrix}1\end{bmatrix}}\ \overset{P_1}{\begin{bmatrix}1\\1/2\\0\\-1/2\\-1\end{bmatrix}}\ \overset{P_2}{\begin{bmatrix}1\\-1/2\\-1\\-1/2\\1\end{bmatrix}} \qquad \Rightarrow \qquad \boldsymbol{G}^T\boldsymbol{G} = \begin{bmatrix}\|P_0\|^2 & 0 & 0\\ 0 & \|P_1\|^2 & 0\\ 0 & 0 & \|P_2\|^2\end{bmatrix} = \begin{bmatrix}5 & 0 & 0\\ 0 & \frac{5}{2} & 0\\ 0 & 0 & \frac{7}{2}\end{bmatrix}$$

5.1.6 Chebyshev Orthogonal Polynomials

Idea: Approximation of a continuous polynomial using Chebyshev polynomials.
Definition
Chebyshev polynomials are defined as $\boxed{T_n(x) = \cos(n \arccos(x))}$ with $(n = 0, 1, \ldots)$ and $(-1 \leq x \leq 1)$. This polynomial results in a clustering of data points in the boundary regions. The first few polynomials T_n and their expressions in terms of x:

$T_0 = 1$	$x^0 = 1 = T_0$
$T_1 = x$	$x^1 = x = T_1$
$T_2 = 2x^2 - 1$	$x^2 = \frac{1}{2}T_2 + \frac{1}{2}T_0$
$T_3 = 4x^3 - 3x$	$x^3 = \frac{1}{4}T_3 + \frac{3}{4}T_1$
$T_4 = 8x^4 - 8x^2 + 1$	$x^4 = \frac{1}{8}T_4 + \frac{1}{2}T_2 + \frac{3}{8}T_0$
$T_5 = 16x^5 - 20x^3 + 5x$	$x^5 = \frac{1}{16}T_5 + \frac{5}{16}T_3 + \frac{5}{8}T_1$

Additional Chebyshev polynomials can be computed using the recursion formula $T_{n+1}(x) = 2xT_n(x) - T_{n-1}$ $(n \geq 2)$ with initial conditions $T_1(x) = x,\ T_0(x) = 1$.

Properties:

- The maximum amplitude of the Chebyshev polynomial is $\frac{1}{2^n}$ (max amplitude of the error term).
- Amplitude: $T_n(x) \in [-1, +1]$
- Roots: $T_n(x) = 0 \Leftrightarrow x = \cos(\frac{2i+1}{2n}\pi) \qquad i = 0, 1, \ldots, n-1$ (These roots are the Chebyshev nodes)
- $T_n(x) = \pm 1 \Leftrightarrow x = \cos(\frac{i\pi}{n})$ $(i = 0, 1, \ldots, n)$

Recipe
Chebyshev polynomials can only be used if x **with Chebyshev distances** $x_i = \cos(\frac{2i+1}{2n}\pi)$ $(i = 0, 1, \ldots, n-1)$ are used and not obtained with equidistant distances.
Goal: Approximate $y(t) = p_N(t)$ (polynomial, defined in $[a, b]$) with Chebyshev polynomials of degree m.

1. Normalize $y(t)$ to the standard interval $[-1, 1]$ (affine transformation) $t = a + \frac{b-a}{2}(x + 1)$.
2. Compose $y(x)$ from T_n
3. Truncate $y(x)$ to degree m: $y_m(x)$
4. Back-transform: $x = 2\frac{t-a}{b-a} - 1$
5. Substitute T_n into y_m (see table above)
6. Error estimate for truncate method:
   $\max_t |y(t) - y_m(t)|$ (Truncated part)

Example
Wanted: Approximation $y(t) = t^3$ with degree $m = 2$ for interval $(a, b) = (0, 1)$

1. Transformation with $t = \frac{x+1}{2}$:
   $y(x) = \left(\frac{x+1}{2}\right)^3 = \frac{1}{8}(x^3 + 3x^2 + 3x + 1)$
2. Expand with table:
   $y(x) = \frac{1}{8}\left(\frac{T_3(x)+3T_1(x)}{4} + 3\frac{T_2(x)+T_0}{2} + 3T_1(x) + T_0(x)\right)$
   $= \frac{1}{32}T_3(x) + \frac{3}{16}T_2(x) + \frac{15}{32}T_1(x) + \frac{5}{16}T_0(x)$
3. Truncate to degree $m = 2$:
   $y(x) \approx \frac{3}{16}T_2(x) + \frac{15}{32}T_1(x) + \frac{5}{16}T_0(x)$
4. Back-transform with $x = 2t - 1$:
   $y(t) \approx \frac{3}{16}T_2(2t-1) + \frac{15}{32}T_1(2t-1) + \frac{5}{16}T_0(2t-1)$
5. Replace $T_n(2t - 1)$:
   $y(t) \approx \frac{3}{16}(2(2t-1)^2 - 1) + \frac{15}{32}(2t-1) + \frac{5}{16}$
6. Error estimate:
   $\max_t \left|\frac{1}{32}T_3(2t-1)\right|$

The **Design Matrix** looks like this:

$$G \underbrace{=}_{\text{When Chebyshev}} \begin{bmatrix} T_0(x_0)=1 & T_1(x_0)=x_0 & \dots & T_m(x_0) \\ T_0(x_1)=1 & T_1(x_1)=x_1 & \dots & T_m(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ T_0(x_N)=1 & T_1(x_N)=x_N & \dots & T_m(x_N) \end{bmatrix}_{N\times m}$$

With this, G^TG becomes:

$$G^TG \underbrace{=}_{\text{When Chebyshev}} \begin{bmatrix} N+1 & 0 & \dots & 0 \\ 0 & \frac{N+1}{2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{N+1}{2} \end{bmatrix}_{m\times m}$$

and $(G^TG)^{-1}G^T$ becomes:

$$(G^TG)^{-1}G^T \underbrace{=}_{\text{When Chebyshev}} \begin{bmatrix} \frac{1}{N+1}T_0(x_0)=\frac{1}{N+1} & \frac{1}{N+1}T_0(x_1)=\frac{1}{N+1} & \dots & \frac{1}{N+1}T_0(x_N)=\frac{1}{N+1} \\ \frac{2}{N+1}T_1(x_0)=\frac{2}{N+1}x_0 & \frac{2}{N+1}T_1(x_1)=\frac{2}{N+1}x_1 & \dots & \frac{2}{N+1}T_1(x_N)=\frac{2}{N+1}x_N \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N+1}T_m(x_0) & \frac{2}{N+1}T_m(x_1) & \dots & \frac{2}{N+1}T_m(x_N) \end{bmatrix}_{m\times N}$$

5.1.7 Discrete Least-Squares Chebyshev Approximation

Follows from $a=(G^TG)^{-1}G^Ty$ for $x_i=\cos(\frac{2i+1}{2(N+1)}\pi)$ $(i=0,1,\dots,N)$:

$$p(x)=\sum_{j=0}^{m} a_jT_j(x) \quad \text{where} \quad a_j=\begin{cases} \frac{1}{N+1}\sum_{i=0}^{N} y(x_i) & j=0 \\ \frac{2}{N+1}\sum_{i=0}^{N} T_j(x_i)y(x_i) & j>0 \end{cases}$$

5.1.8 Continuous Least-Squares

For continuous functions that are **not polynomials** (!), the continuous version can also be used.
S: error function to be minimized, w: weight with $(-1\le x\le 1)$

$$S=\int_{-1}^{1}(y(x)-p(x))^2\cdot w(x)\cdot \mathrm{d}x$$

Continuous Least-Squares Chebyshev Approximation
The weight is particularly placed on the boundary (as is customary with Chebyshev). $(w(x)=\frac{1}{\sqrt{1-x^2}})$

$$p(x)=\sum_{j=0}^{m} a_jT_j(x) \quad \text{where} \quad a_j=\begin{cases} \frac{1}{\pi}\int_{-1}^{1}\frac{y(x)}{\sqrt{1-x^2}}dx & j=0 \\ \frac{2}{\pi}\int_{-1}^{1}\frac{y(x)T_j(x)}{\sqrt{1-x^2}}dx & j>0 \end{cases}$$

Continuous Least-Squares Legendre Approximation:
Here, the weight $w(t)=1$ is used.

Legendre Polynomials:

$$P_n(x)=\frac{1}{2^n n!}\cdot\frac{\mathrm{d}^n}{\mathrm{d}x^n}(x^2-1)^n \qquad P_0(x)=1 \qquad P_1(x)=x \qquad P_2(x)=\frac{1}{2}(3x^2-1)$$

$$P_3(x)=\frac{1}{2}(5x^3-3x) \qquad P_4(x)=\frac{1}{8}(35x^4-30x^2+3) \qquad P_5(x)=\frac{1}{8}(63x^5-70x^3+15x)$$

$$1=P_0(x) \quad x=P_1(x) \quad x^2=\frac{2P_2(x)+P_0(x)}{3} \quad x^3=\frac{2P_3(x)+3P_1(x)}{5} \quad x^4=\frac{8P_4(x)+20P_2(x)+7P_0(x)}{35}$$

Coefficients:

$$p(x)=\sum_{j=0}^{m} a_jP_j(x) \quad \text{wobei} \quad a_j=\frac{2j+1}{2}\cdot\int_{-1}^{1} y(x)P_j(x)\cdot \mathrm{d}x \qquad (j=0,1,...,m)$$

5.2 Multi-Variate Linear Least Squares Approximation

Multi-variate least square problems are solved exactly the same way as uni-variate least square problems (using the same methods)! The variable x is no longer a number but a vector with d dimensions!
Basis Generation: The bases are computed using the tensor product of uni-variate bases (multiply all with all in d-variables).

$$\sum_{j_1=0}^{m_1}\sum_{j_2=0}^{m_2}...\sum_{j_d=0}^{m_d} a_{j_1,j_2,...,j_d}\cdot g_{j_1}(x^{(1)})g_{j_2}(x^{(2)})\cdot\cdot...\cdot g_{j_d}(x^{(d)})$$

Examples

Basis 3. Grad:	$\{1, x, y,\ x^2, 2xy, x^2,\ x^3, 3x^2y, 3xy^2, y^3\}$
Basis 4. Grad:	$\{1, x, y,\ x^2, 2xy, x^2,\ x^3, 3x^2y, 3xy^2, y^3,\ x^4, 4x^3y, 6x^2y^2, 4xy^2, y^4\}$

5.2.1 Matrix Condition Number

Given a linear system of equations in the form $Ax = b$ with the solution $x = A^{-1}b$. The condition number $\kappa(A)$ measures the maximum relative error of the solution x with respect to the relative error of the right-hand side b:

$$\kappa(A) = \begin{cases} \max\left(\frac{||\Delta x||/||x||}{||\Delta b||/||b||} \quad \text{if } b \neq 0, \Delta b \neq 0, Ax = b\right) & \text{if the matrix } A \text{ is regular} \\ \infty & \text{otherwise} \end{cases}$$

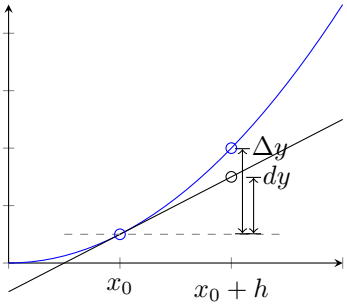
where $||\dots||$ denotes a vector norm, usually the Euclidean norm $||\dots||_2$ or the maximum norm $||\dots||_\infty$.
If $\kappa(A)$ is relatively small, the system/matrix is called "well-conditioned," otherwise, it is "ill-conditioned."
For a regular matrix A and the Euclidean norm $||\dots||_2$:

- $\kappa(A) = \frac{|\sigma_{max}|}{|\sigma_{min}|}$, $|\sigma_{max}|$ = absolute maximum singular value, and $|\sigma_{min}|$ = absolute minimum singular value of A
- If A is also a normal matrix: $\kappa(A) = \frac{|\lambda_{max}|}{|\lambda_{min}|}$, $|\lambda_{max}|$ = absolute maximum eigenvalue, and $|\lambda_{min}|$ = absolute minimum eigenv

6 Error Propagation

6.1 Differential

The differential df denotes the linear part of the increment of a variable in a function.



$$\Delta f = f(x_0 + h) - f(x_0) \approx df = f'(x_o)dx = f'(x_0)h = f'(x_0)\Delta x$$

Δf is the *absolute* error of f at x_0. The *relative* error is $\frac{\Delta f}{f} \approx \frac{df(x_0)}{f(x_0)}$
Any n times differentiable function can be approximated with a **Taylor** series:

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \ldots + \frac{h^n}{n!}f^{(n)} + R_n(x_0, h)$$

Where $R_n(x_0, h)$ denotes the remainder term and converges to 0 with a speed of $o(h^n)$ ("fast") as $n \to \infty$. There are, for example, the

$$\text{Lagrange Form: } R_n(x_0, h) = \frac{h^{n+1}}{(n+1)!}f^{(n+1)}(\underbrace{x_0 + \vartheta h}_{\xi}) \qquad (\xi \in (x_0, x_0 + h), \vartheta \in (0, 1)) \qquad \text{or the}$$

$$\text{Cauchy Form: } R_n(x_0, h) = \frac{h^n(1-\vartheta)^n}{(n+1)!}f^{(n+1)}(\underbrace{x_0 + \vartheta h}_{\xi}) \qquad (\xi \in (x_0, x_0 + h), \vartheta \in (0, 1))$$

With the Taylor series, the absolute error becomes

$$\Delta f = \frac{1}{1!}df(x_0) + \frac{1}{2!}d^2f(x_0) + \ldots + \frac{1}{n!}d^nf(x_0) + R_n(x_0, h)$$

6.2 Multivariate Differentials, Taylor

In the multidimensional case, the differential looks like this:

$$\Delta f \approx df(\vec{x}) = \sum_{i=1}^{n} \frac{\partial f(\vec{x})}{\partial x_i}h_i = h_1\frac{\partial f(\vec{x})}{\partial x_1} + \ldots + h_n\frac{\partial f(\vec{x})}{\partial x_n} \qquad \text{with} \quad \vec{h} = [h_1, \ldots, h_d]^T,\ \vec{x} = [x_1, \ldots x_d]^T$$

The vector field of partial derivatives is called the gradient field of f and is denoted by $\text{grad}(f)$ or $\vec{\nabla}f(\vec{x})$. Thus, $\Delta f \approx \vec{\nabla}f(\vec{x}) \cdot \vec{h}$.
In the multivariate case, the Taylor series for multi-indices $\alpha = \{\alpha_1, \ldots, \alpha_n\}$ and $|\alpha| = \alpha_1 + \ldots + \alpha_n$ is given by

$$f(\vec{x}_0 + h) = f(\vec{x}_0) + \frac{1}{1!}\vec{\nabla}f(\vec{x}_0) \cdot \vec{h} + \sum_{|\alpha|=2}^{N} \frac{1}{\alpha!}\frac{\partial^{|\alpha|}f(\vec{x}_0)}{\partial x^\alpha} \cdot \vec{h}^\alpha + \sum_{|\alpha|=N+1} R_\alpha(\vec{x}_0, \vec{h}) \cdot \vec{h}^\alpha$$

or simplified for 2nd order

$$f(\vec{x}_0 + \vec{h}) = f(\vec{x}_0) + \vec{h}\nabla f(\vec{x}_0) + \underbrace{\left(\frac{h_1^2}{2!}\frac{\partial^2 f}{\partial x_1^2} + \frac{h_2^2}{2!}\frac{\partial^2 f}{\partial^2 x_2^2} + \frac{h_1 h_2}{1!1!}\frac{\partial^2 f}{\partial x_1 \partial x_2}\right)}_{\text{for 2nd order}} +$$

$$R_{(3,0)}(\vec{x}_0, \vec{h}) \cdot \vec{h}_1^3 + R_{(2,1)}(\vec{x}_0, \vec{h}) \cdot \vec{h}_1^2\vec{h}_2 + R_{(1,2)}(\vec{x}_0, \vec{h}) \cdot \vec{h}_1\vec{h}_2^2 + R_{(0,3)}(\vec{x}_0, \vec{h}) \cdot \vec{h}_2^3$$

6.3 Jacobian Matrix

The Jacobian matrix maps all first derivatives of a function. If the Jacobian matrix is square ($m = n$), its determinant $\det(J)$ can be interpreted as the transformed volume:

$$\underbrace{dy_1 dy_2 \ldots dy_n}_{\text{„transformiertes Volumenelement“}} = \det(J_f(\vec{x}))\ \underbrace{dx_1 dx_2 \ldots dx_n}_{\text{„Volumenelement“}}$$

$$J_f(\vec{x}) := \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\vec{x}) & \frac{\partial f_1}{\partial x_2}(\vec{x}) & \ldots & \frac{\partial f_1}{\partial x_n}(\vec{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\vec{x}) & \frac{\partial f_m}{\partial x_2}(\vec{x}) & \ldots & \frac{\partial f_m}{\partial x_n}(\vec{x}) \end{bmatrix}$$

The Jacobian determinant can also be a measure of error propagation. The calculation is done directly using the determinant rule, or

$$\frac{D(f_1, f_2, \ldots, f_n)}{D(x_1, x_2, \ldots, x_n)} = \det(J_f(\vec{x})) \quad \text{or} \quad \det(J_f) = \sqrt{\det\left(\underbrace{J_f^T(\vec{x})J_f(\vec{x})}_{\text{diagonal matrix}}\right)}.$$

Also, for the transformation T and its inverse T^{-1}: $\det(J_T) = \frac{1}{\det J_{T^{-1}}}$

7 Numerical Ordinary Differential Equations

7.1 Definitions

Ordinary Differential Equation (ODE):

$$\boxed{y'(x) = f(x, y(x)) \qquad \text{with} \qquad y(x_0) = y_0}$$

$f(x, y)$ is also referred to as the right-hand side (r.h.s.) function.

7.2 Explicit Methods

Forward time stepping is used without the need for solvers, based on the Taylor approximation. However, this approach becomes computationally intensive quickly, requiring higher derivatives in multiple dimensions.

$$\boxed{y(x_0 + h) = y(x_0) + \frac{h}{1!}y'(x_0) + \frac{h^2}{2!}y''(x_0) + \ldots + \frac{h^p}{p!}y^{(p)}(x_0) + \underbrace{\frac{h^{p+1}}{(p+1)!}y^{(p+1)}(\xi)}_{\text{remainder term}}}$$

Further derivatives are derived using the chain rule:

$$\boxed{y''(x) = \frac{\partial f(x,y)}{\partial x}1 + \frac{\partial f(x,y)}{\partial y}f(x,y) = \frac{\partial y'(x)}{\partial x}1 + \frac{\partial y'(x)}{\partial y}y'(x)} \quad \boxed{y'''(x) = \frac{\partial y''(x)}{\partial x}1 + \frac{\partial y''(x)}{\partial y}y'(x)} \quad \ldots$$

Derivatives up to the third degree are listed here (immense computational effort required):

$$\begin{aligned} y(x+h) =& y(x) + \frac{f(x,y)}{1!}h + \\ & \frac{1}{2!}\left(\frac{\partial f(x,y)}{\partial x}1 + \frac{\partial f(x,y)}{\partial y}f(x,y)\right)h^2 + \\ & \frac{1}{3!}\left(\frac{\partial^2 f(x,y)}{\partial x^2}1 + 2\frac{\partial^2 f(x,y)}{\partial x \partial y}f(x,y) + \frac{\partial^2 f(x,y)}{\partial y^2}f(x,y)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2 f(x,y) + \frac{\partial f(x,y)}{\partial x}\frac{\partial f(x,y)}{\partial y}\right)h^3 + \ldots + \\ & \frac{1}{4!}y^{(4)}(x)h^4 + \ldots + \frac{1}{p!}y^{(p)}(x)h^p + \underbrace{\frac{1}{(p+1)!}y^{(p+1)}(\xi)h^{p+1}}_{\text{remainder term}} \end{aligned}$$

$$\text{Global error} = \underbrace{\max_{0 \le i \le k}}\, |y_i - y(x_i)|$$

$$\text{Local error (slope)} = \tau_h(x_n) = \frac{y(x_n + h) - y(x_n)}{h} - \left(\frac{y'(x_n)}{1!} + \frac{y''(x_n)}{2!}h^1 + \cdots + \frac{y^{(p)}(x_n)}{p!}h^{p-1}\right)$$

$$h\tau_h(x_n) = y(x_n + h) - y(x_n) - \left(\frac{y'(x_n)}{1!}h^1 + \frac{y''(x_n)}{2!}h^2 + \cdots + \frac{y^{(p)}(x_n)}{p!}h^p\right)$$

Assuming that the initial value of the current step is exactly correct, i.e., $y_n = y(x_n)$:

$$h\tau_h(x_n) = y(x_n + h) - \left(y(x_n) + \frac{y'(x_n)}{1!}h^1 + \frac{y''(x_n)}{2!}h^2 + \cdots + \frac{y^{(p)}(x_n)}{p!}h^p\right) = y(x_n + h) - y_{n+1}$$

7.2.1 Explicit Euler Method

Special case of explicit methods:
Taylor with order $p = 1$
Relatively inaccurate approximation since the derivative remains unchanged in each step.

Global error: $\max_{0 \le i \le k} |y_i - y(x_i)|$ with the number of steps k
Local error: $y(x_n + h) - y_{n+1}$ when the initial value of the current step is exactly correct

$$\begin{aligned} y_0 &= y(x_0) \\ y(x_0 + h) \approx y_1 &= y_0 + f(x_0, y_0)h \approx y(x_0) + y'(x_0)h \\ y(x_1 + h) \approx y_2 &= y_1 + f(x_1, y_1)h \approx y(x_1) + y'(x_1)h \\ &\vdots \\ y(x_{n-1} + h) \approx y_n &= y_{n-1} + f(x_{n-1}, y_{n-1})h \approx y(x_{n-1}) + y'(x_{n-1})h \end{aligned}$$

7.2.2 Explicit Taylor Methods of Higher Order

For $p > 1$, a better approximation is achieved, but the computational effort increases significantly. See Section 7.2 for calculation.

$$y_0 = y(x_0)$$
$$y_{k+1} = y_k + \frac{y'_k}{1!}h_k + \frac{y''_k}{2!}h_k^2 + \ldots + \frac{y_k^{(p)}}{p!}h_k^p$$

7.2.3 Explicit Runge-Kutta (RK) Methods Skript S. 8

s recursive stages k are defined, which, when summed, yield the Taylor approximation. Here, $s = 4$.

$$k_1 = hf(x, y)$$
$$k_2 = hf(x + mh, y + mk_1)$$
$$k_3 = hf(x + nh, y + nk_2)$$
$$k_4 = hf(x + ph, y + pk_3)$$

$$y(x + h) \approx y(x) + ak_1 + bk_2 + ck_3 + dk_4$$

In contrast to the script (p. 10), the h values are not included in the k definitions but appear in the final formula. With the correct boundary conditions (p. 11), this formula corresponds to the Taylor polynomial up to order 4.
By equating with Taylor, explicit RK methods define an overdetermined system of equations with 8 equations and 7 unknowns m, n, p and a, b, c, d.

There are different **solutions**, also listed here for order $s = 2$ and $s = 4$:

Solution Name	# Stages (s)	Solutions
Heun	2	$a = b = \frac{1}{2}$, $m = 1$
Explicit Midpoint	2	$a = 0$, $b = 1$, $m =$
Classical Runge-Kutta	4	$m = n = \frac{1}{2}$, $p = 1$

$$\text{Local error (slope)} = \tau_h(x_n) = \frac{y(x_n + h) - y(x_n)}{h} - \left(\frac{\tilde{k}_1 + 2\tilde{k}_2 + 2\tilde{k}_3 + \tilde{k}_4}{6h} \right)$$

The tilde superscript $\tilde{k}$ indicates that the stage values k must be evaluated using exact values for x, y.

7.2.4 General Framework for Runge-Kutta Methods (Butcher Tableaus) Skript S. 13

$$k_1 = f(x + c_1 h, y + h\underbrace{\sum_{j=1}^{s} a_{1,j} k_j}_{g_1})$$
$$k_2 = f(x + c_2 h, y + h\underbrace{\sum_{j=1}^{s} a_{2,j} k_j}_{g_2})$$
$$k_3 = f(x + c_3 h, y + h\underbrace{\sum_{j=1}^{s} a_{3,j} k_j}_{g_3})$$
$$\vdots$$
$$k_s = f(x + c_s h, y + h\underbrace{\sum_{j=1}^{s} a_{s,j} k_j}_{g_s})$$

} s stages

$$y(x + h) \approx y(x) + h\left(\sum_{j=1}^{s} b_j k_j\right)$$

$$\begin{array}{c|cccc} 0 & 0 & 0 & \cdots & 0 & 0 \\ c_2 & a_{2,1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

$\boldsymbol{a}$ = Splitting in Y
$\boldsymbol{b}^T$ = Solutions given by the method
$\boldsymbol{c}$ = Splitting in X

FSAL (First Same As Last) Special Form:

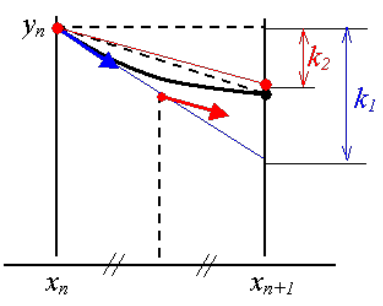
$$\begin{array}{c|cccc} 0 & 0 & 0 & \cdots & 0 & 0 \\ c_2 & a_{2,1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{s-1} & a_{s-1,1} & a_{s-1,2} & \cdots & 0 & 0 \\ 1 & b_1 & b_2 & \cdots & b_{s-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & 0 \end{array}$$

Conditions (For Explicit Method):

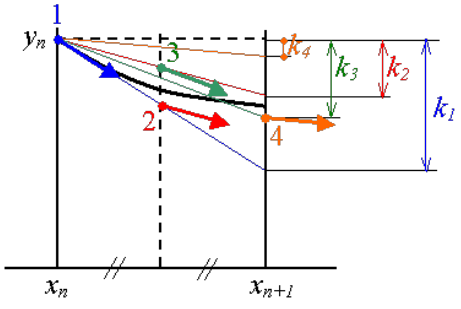
- $c_i = \sum_{j=1}^{s} a_{ij} = \sum_{j=1}^{i-1} a_{ij} \; (i = 2, \ldots, s)$
- $\sum_{j=1}^{s} b_j = 1$
- $c_1 = 0$
- $a_{1j} = 0 \quad 1 \leq j \leq s$
- $a_{ij} = 0 \quad j \geq i$

$\begin{array}{c\|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$	$\begin{array}{c\|cc} 0 & & \\ 1 & & \\ \hline & & \end{array}$ $\begin{array}{c\|cc} 0 & & \\ 1/2 & 1/2 & \\ \hline & 0 & 1 \end{array}$	$\begin{array}{c\|cc} 0 & & \\ 2/3 & 2/3 & \\ \hline & 1/4 & 3/4 \end{array}$
Classical Runge-Kutta of order 4 (cf. 1.10c)	Explicit Euler method (cf. 1.3A') Explicit Midpoint method (1.9b)	Yet another method (cf. 1.9c)

Heun Method (RK-2): $b_1 = b_2 = \frac{1}{2}; c_2 = \frac{1}{2}; a_{2,1} = \frac{1}{2}$



Runge-Kutta 2ème ordre :
$k_2 = h\, f\,(x_n + h/2,\ y_n + k_1/2)$



Runge-Kutta 4ème ordre

7.2.5 Adaptive Explicit Methods Skript S. 17

Idea: Automatic adaptation of the step size h. This requires defining the maximum error of the approximation. The *Accuracy Goal* (ag) specifies the minimum matching number of decimal places, while the *Precision Goal* (pg) represents the number of significant figures in the result. The tolerance parameter is:

$$\varepsilon = \varepsilon_a + |y|\varepsilon_r = 10^{-ag} + |y|10^{-pg} \geq |e|.$$

Embedded Pairs of Explicit Runge-Kutta Methods

0	0	0	$\cdots$	0	0
c_2	$a_{2,1}$	0	$\cdots$	0	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	0	$\vdots$
c_{s-1}	$a_{s-1,1}$	$a_{s-1,2}$	$\cdots$	0	0
c_s	$a_{s,1}$	$a_{s,2}$	$\cdots$	$a_{s,s-1}$	0
	b_1	b_2	$\cdots$	b_{s-1}	b_s
	$\hat{b}_1$	$\hat{b}_2$	$\cdots$	$\hat{b}_{s-1}$	$\hat{b}_s$

The method introduces a second approximation, an *embedded order* $\hat{p}$, responsible for error calculation. The *primary order* p is required to calculate the step size. Usually, $\hat{p} = p - 1$.
The Butcher tableau is expanded by a row of b numbers.

Local error:

$$e_n = y(x+h) - \hat{y}(x+h) = h_n \sum_{j=1}^{s} (b_j - \hat{b}_j) k_j \Rightarrow \|e_n\| = \left\| h_n \sum_{j=1}^{s} (b_j - \hat{b}_j) k_j \right\|$$

If at the current step $\boxed{\frac{\|e_n\|}{\varepsilon} > 1}$, then the estimation of h_n was too optimistic, and the step must be repeated with a smaller step size (**Reject**).
Otherwise $\boxed{\frac{\|e_n\|}{\varepsilon} \leq 1}$, continue with the step size update (**Proceed**):

$$h_{n+1} = h_n \left(\frac{\varepsilon}{\|e_n\|} \right)^{\frac{1}{\tilde{p}}} = h_n \left(\frac{\|e_n\|}{\varepsilon} \right)^{-\frac{1}{\tilde{p}}}$$

$\varepsilon = \varepsilon_a + \varepsilon_r |y_n|$ with $\tilde{p} = \min(p, \hat{p}) + 1$ (Order of the primary method)

Example

x	$\boldsymbol{y}$	h_n	$\left(\frac{\|e_n\|}{\varepsilon}\right)^{-\frac{1}{\tilde{p}}}$	h_{n+1}	Status
10	$[1, -1]^T$	1	0.4	0.4	Reject
10	$[1, -1]^T$	0.4	1.5	0.6	Proceed
10.4	$[0.420958, -1.40852]^T$	0.6	0.5	0.3	Reject

7.2.6 Stability of Explicit Methods Skript S. 27

The global relative error must not diverge; it must be bounded. Since the analysis with the ODE under investigation is challenging due to the lack of knowledge about the solution, the general ODE is considered, the Dahlquist model:

$$y' = Ay \quad y(0) = 1 \qquad \text{with } A = \Re\{A\} + j\Im\{A\} \in \mathbb{C}$$

Its solution is $y = e^{\Re\{A\}x} (\cos(\Im\{A\}x) + j \sin(\Im\{A\}x))$, i.e., a vibration with exponential amplitude $e^{\Re\{A\}x}$ and frequency $\Im\{A\}$.
With the method under investigation, a *linear stability function* $F(z)$ can be calculated.
Example The examination is shown using the Heun method (RK-2):

$$k_1 = Ay_k$$
$$k_2 = A(y_k + 1hk_1) = A(y_k + Ahy_k) \qquad \Longrightarrow$$
$$y_0 = y(x_0) = y(0) = 1$$
$$y_{k+1} = y_k \underbrace{\left(1 + hA + (hA)^2 \frac{1}{2}\right)}_{F(hA)=F(z), \quad z=hA \in \mathbb{C}}$$

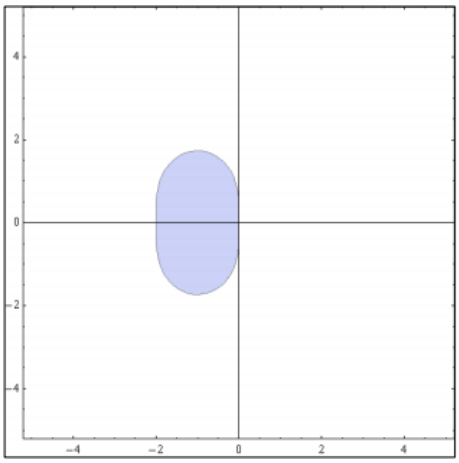
The linear stability function can now be substituted as follows:

$$y_{k+1} = \Big(|F(z)| \exp\left(j \arg(F(z))\right) \Big) y_k$$

Now three cases can be distinguished:

1. $\Re\{A\} < 0$: The amplitude of the ODE decreases exponentially. The stability condition requires that the approximated values y_k $(k = 0, 1, \ldots)$ also decrease exponentially. Therefore, and from $y_{k+1} \propto |F(z)|$, it follows that $|F(z)| < 1$.
2. $\Re\{A\} = 0$: The amplitude of the ODE is constantly 1. The stability condition requires that the approximated values y_k $(k = 0, 1, \ldots)$ also remain constant. Therefore, and from $y_{k+1} \propto |F(z)|$, it follows that $|F(z)| = 1$.
3. $\Re\{A\} > 0$: The amplitude of the ODE increases exponentially. The stability condition requires that the approximated values y_k $(k = 0, 1, \ldots)$ also increase exponentially. Therefore, and from $y_{k+1} \propto |F(z)|$, it follows that $|F(z)| > 1$.

Cases 2 and 3 do not pose problems (conditions are always fulfilled); only the first case is critical.



Case 1 of the stability condition ($\Re\{A\} < 0$):
$1 > |F(z)| = |1 + hA + \frac{1}{2}(hA)^2| \Rightarrow -2 < h\Re\{A\} < 0$.
Cases 2 and 3 are not considered in the figure.

Recursive Formula S29 Recursive formulas exist for the stability polynomial $F(z) = 1 + b_1 k_1(z) + \ldots + b_s k_s(z)$:
$k_1(z) = z, \qquad k_{j+1}(z) = z(1 + a_{j+1,1} k_1(z) + a_{j+1,2} k_2(z) + \ldots + a_{j+1,j} k_j(z))$

Weakness of Explicit Methods: Stability!

7.2.7 Stiffness Skript S. 33

Definition: Stiffness = very challenging (according to Wikipedia). Rough idea: Explicit methods are forced to unacceptably small step sizes (computational complexity). This happens despite the actual ßmoothßurfaces of the functions, which should intuitively not be very difficult to calculate.

If stability violations (stiffness) are detected, a switch to implicit methods (black-box solvers) should be made, and then, with non-stiffness detection, a switch back to explicit methods should be made.

Stiffness depends on:

1. Specific ODE
2. Specific step size h
3. Specific method (e.g., RK-4)

Stiffness Detection Requirement: $c_{s-1} = c_s = 1$. Stiffness can be detected by testing $\left| h \frac{\partial f(x,y)}{\partial y} \right|$ against the absolute limit of the stability region. Stiffness is present if $|h\tilde{\lambda}|$ is outside or on the boundary of the stability region.
Example of explicit Runge-Kutta:

$$k_{s-1} = f(x + c_{s-1}h, \underbrace{y + h\sum_{j=1}^{s} a_{s-1,j} k_j}_{g_{s-1}}) \qquad k_s = f(x + c_s h, \underbrace{y + h\sum_{j=1}^{s} a_{s,j} k_j}_{g_s}) \qquad \Rightarrow \qquad \tilde{\lambda} = \frac{\|k_s - k_{s-1}\|}{\|g_s - g_{s-1}\|}$$

$\tilde{\lambda}$ is an estimate for $f_y = \frac{\partial}{\partial y} f(x, y)$, e.g., $y' = x^4 - 25y^4 = f(x, y)$, and takes on the role of $\Re\{A\}$ in stability analysis.

7.3 Systems of ODEs and Higher-Order ODEs Skript S. 37

Systems of ODEs are written in vector notation and calculated as usual. That's all.

8 Differentiation Table

	$F(x)$	$F'(x)$
Addition	$f(x) \pm g(x)$	$f'(x) \pm g'(x)$
Linearity	$af(x)$	$af'(x)$
Product Rule	$f(x)g(x)$	$f'(x)g(x) + f(x)g'(x)$
Quotient Rule	$\frac{f(x)}{g(x)}$	$\frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2}$
Chain Rule	$f(g(x))$	$f'(g(x)) \cdot g'(x)$
	$f^{-1}(x)$	$\frac{1}{f'(f^{-1}(x))}$
Basic functions	x^n for any real n	nx^{n-1}
	e^x	e^x
	a^x $(a > 0)$	$(\ln a)a^x$
	$\ln x$	$\frac{1}{x}$
Trig functions	$\sin x$	$\cos x$
	$\cos x$	$-\sin x$
	$\tan x$	$\frac{1}{\cos^2 x} = 1 + \tan^2 x$
	$\arctan x = \tan^{-1} x$	$\frac{1}{1+x^2}$
	$\arcsin x = \sin^{-1} x$	$\frac{1}{\sqrt{1-x^2}}$
Hyperbolic Trig	$\sinh x$	$\cosh x$
	$\cosh x$	$\sinh x$
	$\tanh x$	$\frac{1}{\cosh^2 x}$
	$\sinh^{-1} x$	$\frac{1}{\sqrt{1+x^2}}$
	$\tanh^{-1} x$	$\frac{1}{1-x^2}$

9 Overview of polynomials

Name	Formula	Example	Reference
Newton	$\pi_n(x) = \prod_{i=1}^{n}(x - x_{i-1})$	$\pi_2 = (x - x_0)(x - x_1)$	1.3, S. 1
Bernstein	$B_{i,n}(x) = \binom{n}{i}(1-x)^{n-i}x^i$	$B_{1,2} = 2x(1-x)$	4.2.1, S. 8
Monomials	x^n	x^2	5.1.4, S. 11
Orthogonals	$p_{k,N}(t) = \sum_{i=0}^{k}(-1)^i\binom{k}{i}\binom{k+i}{i}\frac{t^{(i)}}{N^{(i)}}$	$P_1(x) = 1 - \frac{x-3}{2}$	5.1.5, S. 11
Chebyshev	$T_n(x) = \cos(n \arccos(x))$	$T_2 = 2x^2 - 1$	5.1.6, S. 12

10 Integration Table

Linearity	$\int af(x) + bg(x)\,dx = a\int f(x)\,dx + b\int g(x)\,dx$
Substitution	$\int f(w(x))w'(x)\,dx = \int f(w)\,dw$
Integration by parts	$\int u(x)v'(x)\,dx = u(x)v(x) - \int u'(x)v(x)\,dx$

Basic Functions

$$\int x^n\,dx = \frac{x^{n+1}}{n+1} + C \qquad \int \frac{1}{x}\,dx = \ln|x| + C$$

$$\int e^{ax}\,dx = \frac{1}{a}e^x + C \qquad \int a^x\,dx = \frac{a^x}{\ln a} + C$$

Trigonometric functions

$$\int \sin x\,dx = -\cos x + C \qquad \int \cos x\,dx = \sin x + C$$

$$\int \frac{1}{\cos^2 x}\,dx = \tan x + C \qquad \int \tan x\,dx = -\ln|\cos x| + C$$

$$\int \cot x\,dx = \ln|\sin x| + C$$

Hyperbolic Trig functions

$$\int \sinh x\,dx = \cosh x + C \qquad \int \cosh x\,dx = \sinh x + C$$

$$\int \tanh x\,dx = \ln(\cosh x) + C \qquad \int \coth x\,dx = \ln|\sinh x| + C$$

Functions with $a^2 \pm x^2$

$$\int \frac{dx}{\sqrt{a^2 - x^2}} = \sin^{-1}\left(\frac{x}{a}\right) + C \qquad \int \frac{dx}{a^2 + x^2} = \frac{1}{a}\tan^{-1}\left(\frac{x}{a}\right) + C$$

$$\int \frac{dx}{a^2 - x^2} = \frac{1}{2a}\ln\left|\frac{x+a}{x-a}\right| + C$$

$$\int \frac{dx}{\sqrt{x^2 - a^2}} = \cosh^{-1}\left(\frac{x}{a}\right) + C \qquad \int \frac{dx}{\sqrt{x^2 + a^2}} = \sinh^{-1}\left(\frac{x}{a}\right) + C$$

Inverse Functions

$$\int \ln x\,dx = x\ln x - x + C \qquad \int \arcsin x\,dx = x\arcsin x + \sqrt{1 - x^2} + C$$

$$\int \arctan x = x\arctan x - \frac{1}{2}\ln(1 + x^2) + C$$

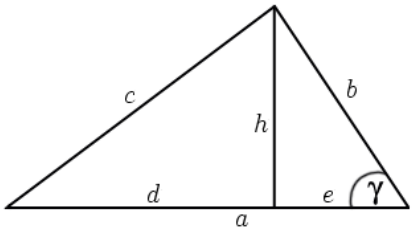
11 Idiotenseite

11.1 Dreiecksformeln

Cosinussatz $c^2 = a^2 + b^2 - 2 \cdot a \cdot b \cdot \cos\gamma$

Sinussatz $\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma} = 2r = \frac{u}{\pi}$

Pythagoras $\sin^2(b) + \cos^2(b) = 1 \qquad \tan(b) = \frac{\sin(b)}{\cos(b)}$



$\sin\beta = \frac{b}{a} = \frac{\text{Gegenkathete}}{\text{Hypotenuse}}$
$\cos\beta = \frac{c}{a} = \frac{\text{Ankathete}}{\text{Hypotenuse}}$

$\tan\beta = \frac{c}{b} = \frac{\text{Gegenkathete}}{\text{Ankathete}}$
$\cot\beta = \frac{c}{b} = \frac{\text{Ankathete}}{\text{Gegenkathete}}$

11.2 Funktionswerte für Winkelargumente

deg	rad	sin	cos	tan
0°	0	0	1	0
30°	$\frac{\pi}{6}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{3}$
45°	$\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1
60°	$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$

deg	rad	sin	cos
90°	$\frac{\pi}{2}$	1	0
120°	$\frac{2\pi}{3}$	$\frac{\sqrt{3}}{2}$	$-\frac{1}{2}$
135°	$\frac{3\pi}{4}$	$\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$
150°	$\frac{5\pi}{6}$	$\frac{1}{2}$	$-\frac{\sqrt{3}}{2}$

deg	rad	sin	cos
180°	π	0	-1
210°	$\frac{7\pi}{6}$	$-\frac{1}{2}$	$-\frac{\sqrt{3}}{2}$
225°	$\frac{5\pi}{4}$	$-\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{2}}{2}$
240°	$\frac{4\pi}{3}$	$-\frac{\sqrt{3}}{2}$	$-\frac{1}{2}$

deg	rad	sin	cos
270°	$\frac{3\pi}{2}$	-1	0
300°	$\frac{5\pi}{3}$	$-\frac{\sqrt{3}}{2}$	$\frac{1}{2}$
315°	$\frac{7\pi}{4}$	$-\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$
330°	$\frac{11\pi}{6}$	$-\frac{1}{2}$	$\frac{\sqrt{3}}{2}$

11.3 Periodizität

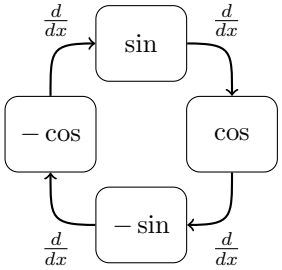
$\cos(a + k \cdot 2\pi) = \cos(a) \qquad \sin(a + k \cdot 2\pi) = \sin(a) \qquad (k \in \mathbb{Z})$

11.4 Quadrantenbeziehungen

$\sin(-a) = -\sin(a)$
$\sin(\pi - a) = \sin(a)$
$\sin(\pi + a) = -\sin(a)$
$\sin\left(\frac{\pi}{2} - a\right) = \sin\left(\frac{\pi}{2} + a\right) = \cos(a)$

$\cos(-a) = \cos(a)$
$\cos(\pi - a) = -\cos(a)$
$\cos(\pi + a) = -\cos(a)$
$\cos\left(\frac{\pi}{2} - a\right) = -\cos\left(\frac{\pi}{2} + a\right) = \sin(a)$

11.5 Ableitungen



11.6 Additionstheoreme

$\sin(a \pm b) = \sin(a) \cdot \cos(b) \pm \cos(a) \cdot \sin(b)$
$\cos(a \pm b) = \cos(a) \cdot \cos(b) \mp \sin(a) \cdot \sin(b)$
$\tan(a \pm b) = \frac{\tan(a) \pm \tan(b)}{1 \mp \tan(a) \cdot \tan(b)}$

11.7 Doppel- und Halbwinkel

$\sin(2a) = 2\sin(a)\cos(a)$
$\cos(2a) = \cos^2(a) - \sin^2(a) = 2\cos^2(a) - 1 = 1 - 2\sin^2(a)$
$\cos^2\left(\frac{a}{2}\right) = \frac{1+\cos(a)}{2} \qquad \sin^2\left(\frac{a}{2}\right) = \frac{1-\cos(a)}{2}$

11.8 Produkte

$\sin(a)\sin(b) = \frac{1}{2}(\cos(a-b) - \cos(a+b))$
$\cos(a)\cos(b) = \frac{1}{2}(\cos(a-b) + \cos(a+b))$
$\sin(a)\cos(b) = \frac{1}{2}(\sin(a-b) + \sin(a+b))$

11.9 Euler-Formeln

$\sin(x) = \frac{1}{2j}\left(e^{jx} - e^{-jx}\right) \qquad \cos(x) = \frac{1}{2}\left(e^{jx} + e^{-jx}\right)$
$e^{x+jy} = e^x \cdot e^{jy} = e^x \cdot (\cos(y) + j\sin(y))$
$e^{j\pi} = e^{-j\pi} = -1$

11.10 Summe und Differenz

$\sin(a) + \sin(b) = 2 \cdot \sin\left(\frac{a+b}{2}\right) \cdot \cos\left(\frac{a-b}{2}\right)$
$\sin(a) - \sin(b) = 2 \cdot \sin\left(\frac{a-b}{2}\right) \cdot \cos\left(\frac{a+b}{2}\right)$
$\cos(a) + \cos(b) = 2 \cdot \cos\left(\frac{a+b}{2}\right) \cdot \cos\left(\frac{a-b}{2}\right)$
$\cos(a) - \cos(b) = -2 \cdot \sin\left(\frac{a+b}{2}\right) \cdot \sin\left(\frac{a-b}{2}\right)$
$\tan(a) \pm \tan(b) = \frac{\sin(a \pm b)}{\cos(a)\cos(b)}$