Newton Polynomial Interpolation

For some interpolation problem (collocation), a polynomial $p(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_m x^m$ of degree m is required such that the degree m is minimal while meeting collocation condition $y(x_k) = p(x_k)$.

The $n+1$ measurements are given by indexed pairs (x_k, y_k) with $k = 0, 1, 2, \ldots, n$.

Aitken-Neville Recursion Formula

Group together polynomials of a subset of the measurement into a global polynomial, e.g. for a dataset with three measurements:

$$p_{0,1,2}(x) = \frac{(x - x_0)p_{1,2}(x) - (x - x_2)p_{0,1}(x)}{(x_2 - x_0)}$$

Newton Basis Polynomials

Given a set $n+1$ of measurements (resulting *order* $= n$).

The basis polynomials with degree $k = 0, 1, 2, \ldots, n$:

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

can be used to form a collocation polynomial using a linear combination $p(x) = a_0\pi_0(x) + a_1\pi_1(x) + \ldots + a_m\pi_m(x)$. This forms the system of linear equations

$$\begin{aligned}
y_0 &= a_0 \\
y_1 &= a_0 + a_1(x_1 - x_0) \\
y_2 &= a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \\
&\vdots \\
y_n &= a_0 + a_1\pi_1(x_n) + a_2\pi_2(x_n) + \ldots + a_n\pi_n(x_n)
\end{aligned}$$

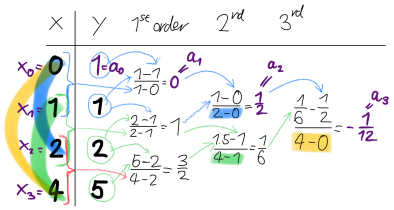
Stability: Since the coefficient a_k is determined by the first k arguments, we can extend the dataset with new data without altering existing coefficients.

Divided Differences

Due to stability, we can write the coefficients as $a_k(x_0, \ldots, x_k)$, only depending on the arguments lower than k. Applying the Aitken-Neville formula, we receive for $k = 0, 1, \ldots, n$:

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

$k=0$	$y(x_0)$
$k=1$	$\frac{y(x_1)-y(x_0)}{(x_1-x_0)}$ $\quad \frac{\Delta y}{\Delta x}$
$k=2$	$\frac{y(x_1,x_2)-y(x_0,x_1)}{(x_2-x_0)}$
$k=3$	$\frac{y(x_1,x_2,x_3)-y(x_0,x_1,x_2)}{(x_3-x_0)}$



from which (in this special example above) we can derive the collocation polynomial

$$y(x) = p(x) = 1 + 0 \cdot \pi_1(x) + 0.5 \cdot \pi_2(x) - \frac{1}{12}\pi_3(x)$$

Error Formula

With the "real", physical (usually unknown) function $y = f(x)$ interpolated by $p(x)$ and some intermediate value ξ:

$$\varepsilon = |y(x) - p(x)| = \left| \frac{f^{(n+1)}(\xi)}{(n+1)!}\pi_{n+1}(x) \right|$$

$[\xi \in (\min(x_k), \max(x_k)), x \in (\min(x_k), \max(x_k))]$

With large values of n, this leads to the *Runge phenomenon* (oscillation of errors towards the boundaries of arguments range), mainly because one can interpret divided differences as derivations.

Chebyshev Arguments for the Runge Phenomenon

Choosing the Chebyshev arguments

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

in the interval $[-1, 1]$ minimises the maximum amplitude to 2^{-n}:

$$\min_{\{x_0, x_1, \ldots, x_n\}} \max_{-1 \le x \le 1} |\pi_{n+1}(x)| \le \frac{1}{2^n}$$

We therefore need the affine formula $x \rightarrowtail a + \frac{b-a}{2}(x+1)$ to transform the interval $[-1, 1]$ into $[a, b]$ in which our measurement x-values reside. Downsides: Expense of increased measurements at edges of the interval. Also, increasing the number of n (meas. minus 1) implies that nearly all arguments and Newton basis polynomials change.

Uniformly distributed arguments

For uniformly distributed arguments, where h is the step size ($x_k = x_0 + kh$), the polynomial p is given by

$$p(x) = \sum_{k=0}^{n} \frac{\Delta^k y_0}{h^k k!}\pi_k(x)$$

with Δ as difference operator $\Delta y_k = y_{k+1} - y_k$ and Δ^m for the composition (e.g. $\Delta^2 = \Delta y_1 - \Delta y_0 = (y_2 - y_1) - (y_1 - y_0)$).

Taylor-Approximation

With a model function $y = f(x)$ with continuous derivatives up to order n on the interval $[x_0, x]$:

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

Hermite Interpolation

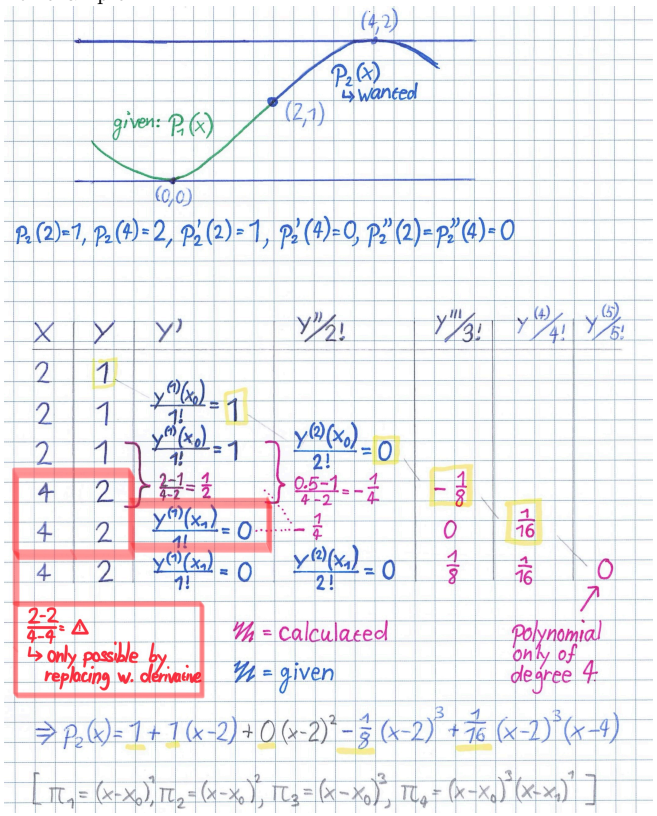
Osculation or Hermite Interpolation is the collocation problem extended by the requirement to match derivatives of model function at some of the arguments.

Solving using Divided Differences

We can use the property that a divided difference with repeated arguments can be interpreted as a derivative:

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

For example:



π_5 would be $\pi_5 = (x-x_0)(x-x_0)(x-x_0)(x-x_1)(x-x_1)$ but the factor is zero, removing the need for it.

Osculation Error Formula

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

$$x, \xi \in (\min x_i, \max x_i)_{i=0,1,\dots,n}$$

with d denoting the total number of conditions and d_i denoting the number of conditions for the argument x_i $(i = 0, \dots, n)$. We have $d = d_0 + d_1 + \dots + d_n$.
In the above example, this would be

$$\frac{y^{(6)}(\xi)}{6!}(x-2)^3(x-4)^3$$

($d = 6$ constraints → at most degree of 5)

Multivariate Polynomial Interpolation

Bilinear Interpolation

E.g. we want to interpolate color between four pixels. We get the four basis functions $\{\pi_0(x), \pi_1(x)\} \times \{\pi_0(y), \pi_1(y)\}$ represented by a 2-fold tensor product:

$$\overbrace{\begin{bmatrix} 1\cdot 1 & (x-x_0)\cdot 1 \\ 1\cdot(y-y_0) & (x-x_0)(y-y_0) \end{bmatrix}}^{\pi_0(x) \quad \pi_1(x)} \left.\begin{matrix}\\\\\end{matrix}\right\} \begin{matrix}\pi_0(y)\\ \pi_1(y)\end{matrix}$$

and the polynomial

$$\begin{aligned} p(x,y) = {} & a_{0,0}\pi_0(x)\pi_0(y) + a_{1,0}\pi_1(x)\pi_0(y) \\ & + a_{0,1}\pi_0(x)\pi_1(y) + a_{1,1}\pi_1(x)\pi_1(y) \end{aligned}$$

Then, the computation proceeds using divided differences in all steps:

1. Univariate interpolation for x when holding $y = y_0$, ignoring a y parameter

   $$p(x, y_0) = p(x_0, y_0)\pi_0(x) + \frac{p(x_n, y_0) - p(x_0, y_0)}{x_n - x_0}\pi_1(x)$$

   using the measurements at x_0 and x_n. We therefore receive a two-dimensional, univariate polynom
2. Univariate interpolation for x when holding $y = y_n$:

   $$p(x, x_n) = p(x_0, y_n)\pi_0(x) + \frac{p(x_n, y_n) - p(x_0, y_n)}{x_n - x_0}\pi_1(x)$$

   giving a second polynomial "at the other edge"
3. Univariate interpolation for y:

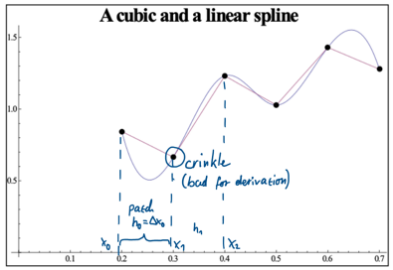
   $$p(x, y) = p(x, y_0)\pi_0(y) + \frac{p(x, y_1) - p(x, y_0)}{y_1 - y_0}\pi_1(y)$$

   where we can substitute $p(x, y_1)$ and $p(x, y_0)$ with the above calculations

---

Spline Interpolation

Given a set of measurement points with arguments $x_0, x_1, \dots, x_n$. (n = # meas - 1)



The splines have to be a continuous curve (C^0) and often fulfill a smoothness condition (C^1 = cont. derivative, C^2 = cont. second derivative, ...). Typical situation is that we seek patching polynomials of degree d which must be a C^{d-1} curve. It is assumed that $n > d-1$ and gives a total of $n(d-1) + 2n - (d-1) = n(d+1) - (d-1)$ conditions.

Cubic Splines

For degree $d = 3$ and $i = 0, 1, \dots, n-1$ we define a cubic polynomial

belonging to patch $[x_i, x_{i+1}]$ ($h = x_{i+1} - x_i = \Delta x_i$):

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

To determine the coefficients, the following equations can be used:

For a_i:

$$a_i = y_i \ (i = 0, \dots, n-1) \tag{1}$$

For b_i:

$$b_{i-1} = \frac{a_i - a_{i-1}}{h_{i-1}} - \frac{2c_{i-1} + c_i}{3}h_{i-1} \ (i = 1, \dots, n-1) \tag{2}$$

while the last one is

$$b_{n-1} = \frac{y_n - a_{n-1}}{h_{n-1}} - c_{n-1}h_{n-1} - d_{n-1}h_{n-1}^2 \tag{3}$$

For c_i coefficients: ($(n-2) \times n$) linear system (underdetermined ⇒ many solutions)

$$\begin{bmatrix} h_0 & 2(h_0+h_1) & h_1 & 0 & \dots & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \dots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix}$$

$$= \begin{bmatrix} \vdots \\ 3\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right) \\ \vdots \end{bmatrix}_{i=1,\dots,n-2}$$

For d_i:

$$d_{i-1} = \frac{c_i - c_{i-1}}{3h_{i-1}} \tag{4}$$

Natural Cubic Splines

Fulfils the boundary condition $S''(x_0) = S''(x_n) = 0$ and minimises the mean total curvature $\int_{x_0}^{x_n} |f''(x)|^2 \, dx$. The tri-diagonal system of equations takes the regular form ($n-1$ equations for $n-1$ coeff.):

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & 0 & \dots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \\ 0 & 0 & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{bmatrix}$$

$$= \begin{bmatrix} \vdots \\ 3\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right) \\ \vdots \end{bmatrix}_{i=1,\dots,n-1}$$

The d-coefficients take the form as described in (4) with

$$d_{n-1} = -\frac{c_{n-1}}{3h_{n-1}} \tag{5}$$

and the b-coefficients as described in (2) and (3).

"Clamped"/Complete Cubic Splines

These splines satisfy a first order derivative condition on the boundary arguments:

$$S'(x_0) = y'_0 \text{ and } S'(x_n) = y'_n$$

The linear system for the c-coefficients:

$$\begin{pmatrix} 2h_0 & h_0 & & & & \\ h_0 & 2(h_0+h_1) & h_1 & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & \\ & & h_2 & 2(h_2+h_3) & h_3 & \\ & & & \ddots & \ddots & \ddots \\ & & & h_{n-3} & 2(h_{n-3}+h_{n-2}) & h_{n-2} \\ & & & & 2h_{n-2} & 4h_{n-2}+3h_{n-1} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} 3\left(\frac{a_1-a_0}{h_0} - y'_0\right) \\ \vdots \\ 3\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_i-y_{i-1}}{h_{i-1}}\right) \\ \vdots \\ 9\frac{y_n - a_{n-1}}{h_{n-1}} - 6\frac{a_{n-1}-a_{n-2}}{h_{n-2}} - 3y'_n \end{pmatrix}_{i=1,\dots,n-2}$$

From (4) we get $d_0, \dots, d_{n-2}$, from (2) we get $b_0, \dots, b_{n-2}$ and b_{n-1} from (3) and finally d_{n-1} from (5).

Error Formulas

$$|y(x) - S(x)| \le \max \frac{|y^{(4)}(x)|}{4!}\frac{5H^4}{16} = \max|y^{(4)}(x)|\frac{5}{384}H^4$$

$$|y'(x) - S'(x)| \le \max \frac{|y^{(4)}(x)|}{4!}H^3 = \max|y^{(4)}(x)|\frac{1}{24}H^3$$

$$|y''(x) - S''(x)| \le \max|y^{(4)}(x)|\frac{3}{8}H^2$$

$$x \in [x_0, x_n], H = \max_{i=0,\dots,n-1} h_i$$

Bernstein-Bézier Splines

The basis functions for Bézier splines are the Bernstein polynomials:

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

by an affine transformation into the interval $[a, b]$ we get the transformed Bernstein polynomials:

$$B_{i,n}(u, a, b) = B_{i,n}\left(\frac{u-a}{b-a}\right) = \frac{1}{(b-a)^n}\binom{n}{i}(b-u)^{n-i}(u-a)^i$$

The polynomials have the following properties:

1. They form a linear basis for the polynomials of order n
2. $B_{i,n}(t)$ has exactly one maximum at $t = i \div n$
3. $B_{i,n}(t)$ has a zero at 0 of order i and a zero at 1 of order $n - i$
4. Symmetry: $B_{i,n}(t) = B_{n-i,n}(1-t)$
5. They are a partition of unity: $\sum_{i=0}^{n} B_{i,n}(t) = 1$

Bezier curves Using a set of $n+1$ d-dimensional control points (vectors) $\vec{P_0}, \vec{P_1}, \ldots, \vec{P_n}$ $(n \geq 2)$ in $\mathbb{R}^d$:

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

giving a spline of degree n.

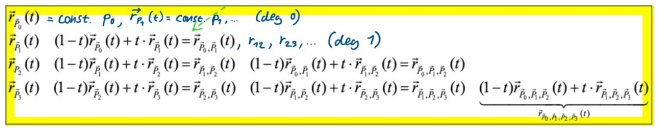
The Bézier curve lies in the convex hull of its points: $\{x \in R^d \mid x = \sum_{i=0}^{n} \lambda_i \vec{P_i} \wedge \sum_{i=0}^{n} \lambda_i = 1 \wedge 0 \leq \lambda_i \leq 1\}$ and has the following properites:

1. The curve starts at point $\vec{P_0}$ and ends at point $\vec{P_n}$
2. $\vec{r'}(0) = n(\vec{P_1} - \vec{P_0}), \quad \vec{r'}(1) = n(\vec{P_n} - \vec{P_{n-1}})$
3. $\vec{r''}(0) = n(n-1)(\vec{P_2} - 2\vec{P_1} + \vec{P_0})$,
   $\vec{r''}(1) = n(n-1)(\vec{P_n} - 2\vec{P_{n-1}} + \vec{P_{n-2}})$ meaning that if we want to stop with no velocity, the last two points have to be the same
4. $\frac{d}{dt^k}\vec{r}(0) = n(n-1)\cdots(n-k+1)\Delta^k \vec{P_0}$,
   $\frac{d}{dt^k}\vec{r}(1) = n(n-1)\cdots(n-k+1)\Delta^k \vec{P_{n-k}}$

Casteljau Recursive Formula

We can construct a Bézier curve similar to Aitken-Neville as a convex linear combination of points of lower order Bézier curves:

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



Composite Bézier Curves

Smush together multiple Bézier curves using common control points and smoothness conditions.

C^1 smoothness condition: Given the curve $\vec{r}_p(t)$ using control points $\vec{P}_0, \ldots, \vec{P}_n$ and the curve $\vec{r}_q(t)$ using control points $\vec{Q}_0, \ldots, \vec{Q}_m$ and $\vec{P}_n = \vec{Q}_0$, then the composite curve is continuously differentiable at the common control point iff

$$\vec{r'}_p(1) = n(\vec{P}_n - \vec{P}_{n-1}) = m(\vec{Q}_1 - \vec{Q}_0) = \vec{r'}_q(0) \ .$$

C^2 smoothness condition: *Additionally to the conditions above*, for C^2 smoothness the two curves are only twice continuously differentiable at the common point iff

$$\begin{aligned}\vec{r''}_p(1) &= n(n-1)\left(\vec{P}_n - 2\vec{P}_{n-1} + \vec{P}_{n-2}\right) \\ &= m(m-1)\left(\vec{Q}_2 - 2\vec{Q}_1 + \vec{Q}_0\right) = \vec{r''}_q(0)\end{aligned}$$

C^k smoothness condition: j times continuously differentiable iff:

$$\begin{aligned}\vec{r}_p^{(j)}(1) &= n(n-1)\cdots(n-j+1)(\Delta^j \vec{P}_{n-j}) \\ &= m(m-1)\cdots(m-j+1)(\Delta^j \vec{Q}_0) = \vec{r}_q^{(j)}(0)\end{aligned}$$

Least-Squares Approximation

We aim to collocate all measurement points with a linear combination

$$\sum_{j=0}^{m} a_j g_j \quad (a_j \in \mathbb{R})$$

of $m-1$ basis functions $\{g_j\}_{j=0,\ldots,m}$ (m = degree (x^m))

- Error = Residuum, errors = residuals. $e_i = r_i = y_i - \sum_{j=0}^{m} a_j g_j(x_i)$
- $\min_i \sum_i |e_i|$: **L^1 norm**
- $\min_i \sum_i e_i^2$: **L^2 norm**
- normalisation of data: $\frac{x-\mu}{\sigma}$ (μ=mean, σ=deviation)

Some basis functions are:

- Trigonometric:

  $\left\{e^{ikx} \mid k \in Z\right\}$ or $\{\cos(kx), \sin(kx) \mid k \in Z\}$ $\quad [x \in (0, 2\pi)]$

- Polynomials:

  $\{x^j \mid j \in \mathbb{N}_0\}$ (standard monomials)

  $\left\{\left(\frac{x-\mu}{\sigma}\right)^j \mid j \in \mathbb{N}_0\right\}$ (normalised standard monomials)

  $\{\pi_j(x) \mid j \in \mathbb{N}_0\}$ (Newton polynomials)

  $\{T_j(x) \mid j \in \mathbb{N}_0\}$ (Chebyshev, $x \in (-1, 1)$)

  $\left\{L_j(x) = \frac{e^x}{j!}\frac{d^j}{dx^j}\left(e^{-x}x^j\right) \mid j \in \mathbb{N}_0\right\}$ (Laguerre, $x \in (0, \infty)$)

  $\left\{H_j(x) = (-1)^j e^{\frac{x^2}{2}}\frac{d^j}{dx^j}\left(e^{-\frac{x^2}{2}}\right) \mid j \in \mathbb{N}_0\right\}$ (Hermite, $x \in (0, \infty)$)

- Exponentials

  $\left\{e^{\alpha k x} \mid k \in \mathbb{Z}\right\}$

Design Matrix

Given a *measurement* with $N+1$ points $\{(x_i, y_i)\}_{i=0,\ldots,N}$ (with x a vector in the multivariate case) and a set of *basis functions* $\{g_j\}_{j=0,\ldots,m}$, the design matrix G results from the linear system $y_i = \sum_{j=0}^{m} a_j g_j(x_i) \quad (i = 0, \ldots, N)$ with $N+1$ equations and $m+1$ unknowns:

$$\underbrace{\begin{bmatrix} g_0(x_0) & \ldots & g_m(x_0) \\ \vdots & \ddots & \vdots \\ g_0(x_N) & \ldots & g_m(x_N) \end{bmatrix}}_{\text{Design matrix } G} \cdot \begin{bmatrix} a_0 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_N \end{bmatrix}$$

that in general is overdetermined ($m < N$) thus the squared sum of residuals $S = \sum_{i=0}^{N}\left(y_i - \sum_{j=0}^{m} a_j g_j(x_i)\right)^2$ has to be minimised [↓].

Normal Equation Theorem: The squared sum of residuals S is minimal iff:

$$G^T \cdot G \cdot a| = G^T \cdot y| \quad (a| \text{ and } y| \text{ are the column vectors})$$

with $G^T \cdot G$ being a normal, symmetric and relatively small $(m+1) \times (m+1)$ matrix resulting in $f(x) = \sum a_j g_j(x)$ as best L^2 fit for data.

Singular Value Decomposition

Any real matrix G of dimensions $((N+1) \times (m+1))$ can be decomposed as the product $G = UDV^T$ where U is an orthogonal (meaning $UU^t = U^tU = I \Rightarrow U^{-1} = U^T$) (large) matrix with $\dim U = (N+1) \times (N+1)$, D is a diagonal matrix with $\dim D = (N+1) \times (m+1)$ and V is orthogonal (small) with $\dim V = (m+1) \times (m+1)$.

Using this decomposition, the minimisation of the square sum of residuals can be found in a numerically stable way: With G decomposed to $G = UDV^T$, the system translates to

$$\hat{r}| = \hat{y}| - D\hat{a}| = \vec{0} \quad [\hat{r} = U^T r|, \ \hat{y}| = U^T y| \text{ and } \hat{a} = V^T a|]$$

and is solved for zero since $\hat{S}$ is minimal iff $\hat{r}_i = 0 \quad (i = 0, \ldots, m)$. Due to the orthogonality of U, the square sum $\hat{S}$ of residuals $\hat{r}$ is equal to the square sum S of the original residuals $r|$.

In the final step, we receive $a| = V \cdot \hat{a}|$.

Also, the singular values of D (the values on the diagonal) are the *roots of the eigenvalues of* G^TG.

Uniform Arguments and Orth. Polynomials

If the set of basis functions $\{g_j\}_{j=0,\dots,m}$ is *orthogonal* with respect to the inner product $\langle g_j, g_k\rangle := \sum_{i=0}^{N} g_j(x_i)g_k(x_i)$, G^TG is diagonal and the solution is trivial:

$$a_j = \frac{\langle y|, g_j|\rangle}{\langle g_j|, g_j|\rangle} \quad (j = 0, 1, \dots, m)$$

Uniform Arguments If we have a set of uniformly distributed arguments of a measurement $\{x_0 + t \cdot h\}_{t=0,\dots,N}$, then is the set of polynomials

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

with integer normalisation $t = (x - x_0) \div h$ orthogonal with respect to the inner product $\langle g, \tilde{g}\rangle$. Here,
$N^{(i)} = N(N-1)(N-2)\cdots(N-i+1)$ and
$t^{(i)} = t(t-1)(t-2)\cdots(t-i+1)$ are defined as falling factorials.
$\Rightarrow$ We therefore get a design matrix with "x"-values $p_{0,N}(t), p_{1,N}(t), \dots$ (e.g. $1;\ 1-\frac{t}{2};\ 1-\frac{3t}{2}+\frac{1}{2}(-1+t)t$, ...) and "y"-values being the $t = 0, \dots, N$ integer values.

Chebyshev Orthogonal Polynomials

Analogously as in polynomial collocation, these polynomials help avoid Runge-phenomenon.
The T-polynomials are defined on the interval $[-1, 1]$:

$$T_n(x) = \cos(n \cdot \arccos(x)) \quad (n = 0, 1, \dots)$$

The degree n is the amount of divisions (intersections at $y = 0$). The first few Chebyshev polynomials are:

$$\overbrace{1}^{0};\ \overbrace{x}^{1};\ \overbrace{2x^2-1}^{2};\ \overbrace{4x^3-3x}^{3};\ \overbrace{8x^4-8x^2+1}^{4};\ \overbrace{16x^5-20x^3+5x}^{5}$$
$$\underbrace{32x^6-48x^4+18x^2-1}_{6};\ \underbrace{64x^7-112x^5+56x^3-7x}_{7};\ \dots$$

They fulfil the second order recursion:
$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad (n \geq 2)$. We can *normalise the polynomials*: $\frac{1}{2^n}T_{n+1}(x)$, which is called a min-max polynomial which we can also write in the Newton polynomial form $(x - x_0)\cdots(x - x_n) = \pi_{n+1}(x)$ and has $\max_{-1\leq x\leq 1} |\pi_{n+1}(x)| = \frac{1}{2^n}$.

Discrete Orthogonality

The Chebyshev polynomials are orthogonal in the following sense:

$$\langle T_j, T_k\rangle := \sum_{i=0}^{N} T_j(x_i)T_k(x_i) = \begin{cases} 0 & j \neq k \\ (N+1)/2 & j = k \neq 0 \\ N+1 & j = k = 0 \end{cases}$$
$$(j, k = 0, \dots, N)$$

$$x_i = \cos\left(\frac{2i+1}{2(N+1)}\pi\right) \quad (i = 0, 1, ..., N)$$

which results in a diagonal G^TG matrix:

$$\begin{bmatrix} N+1 & 0 & 0 & \dots & 0 \\ 0 & \frac{N+1}{2} & 0 & \dots & 0 \\ 0 & 0 & \frac{N+1}{2} & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \dots & \frac{N+1}{2} \end{bmatrix}_{(m+1)\times(m+1)}$$

We can use an affine transformation $x \mapsto a + \frac{b-a}{2}(x+1)$ to map the Chebyshev arguments on an interval $[a, b]$ and it's inverse $-1 + 2\frac{x-a}{b-a}$ mapping $[a, b]$ to $[-1, 1]$, resulting in the basis

$$\{1, T_1(-1+2\frac{x-a}{b-a}), T_2(-1+2\frac{x-a}{b-a}), T_3(-1+2\frac{x-a}{b-a}), \dots\}$$

and the Chebyshev arguments

$$x_i = a + \frac{b-a}{2}\left(\cos\left(\frac{2i+1}{2(N+1)}\pi\right) + 1\right) \quad (i = 0, 1, ..., N)$$

Continuous Orthogonality With a (theoretical) infinite amount of measurements $N \to \infty$, the Chebyshev polynomials fulfil the continuous orthogonal relation:

$$\langle T_j, T_k\rangle_{cont} := \int_{-1}^{1} T_j(x)T_k(x)\frac{dx}{\sqrt{1-x^2}} = \begin{cases} 0 & j \neq k \\ \pi/2 & j = k \neq 0 \\ \pi & j = k = 0 \end{cases}$$

This means, if $y(x)$ is a function on $(-1, 1)$ which is absolutely square-integrable with respect to the weight function $w(x) := \frac{1}{\sqrt{1-x^2}} \quad (x \in (-1, 1))$ in the sense that $\int_{-1}^{1} |y(x)|^2 \frac{dx}{\sqrt{1-x^2}} < \infty$ then the continuous square-sum of residuals

$$S := \int_{-1}^{1} \left(y(x) - \sum_{j=0}^{m} a_j T_j(x)\right)^2 \frac{dx}{\sqrt{1-x^2}}$$

is minimal iif

$$a_j = \begin{cases} \frac{2}{\pi}\int_{-1}^{1} y(x)T_j(x)\frac{dx}{\sqrt{1-x^2}} & j > 0 \\ \frac{1}{\pi}\int_{-1}^{1} y(x)\frac{dx}{\sqrt{1-x^2}} & j = 0 \end{cases}$$

The *error* for continuous least-square polynomials is approximated by the first non-zero coefficient when continuing the series beyond m:

$$|e| = |a_{m+n}P_{m+n}| \quad (n \in \mathbb{N}, a_{m+n} \neq 0)$$

Multivariate Linear Least-Squares Approx.

X-Arguments now are vectors: $\vec{x}_0, \vec{x}_1, \dots, \vec{x}_N$ of dimension d. Usually, tensor products of univariate basis functions are used:

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

defining a basis of $m_1 \times m_2 \times \cdots \times m_d$ functions in d variables $x^{(1)}, x^{(2)}, ..., x^{(d)}$.

Example: We have a 2d input $(x, y) = \vec{x}_i$ and a 1d output z_i. The residuals we want to minimise therefore are $r_i = z_i - f(\vec{x}_i)$ with f being the approximated surface. We chose the basis functions $\{1, x\} \times \{1, y, y^2\}$ giving 6 bivariate basis functions: $\{1, 1y, 1y^2, x \cdot 1, xy, xy^2\}$ with 6 unknown weight variables $a_{00}, a_{01}, a_{02}, a_{10}, a_{11}, a_{12}$ and the design matrix G:

G	$g_0 = 1$	$g_1 = y$	...	$g_m = xy^2$
$\vec{x}_0$	1	$g_1(\vec{x}_0)$	...	$g_m(\vec{x}_0)$
$\vec{x}_1$	1	$g_1(\vec{x}_1)$	...	$g_m(\vec{x}_1)$
			$\vdots$	
$\vec{x}_N$	1	$g_1(\vec{x}_N)$	...	$g_m(\vec{x}_N)$

Continuous Legendre Polynomials

Analogous to Chebyshev polynomials as orthogonal basis functions, we can use Legendre polynomials for a continuous least-squares approximation with the weighting function
$w(x) = 1 \quad (-1 \leq x \leq 1)$
The orthogonal Legendre P-polynomials (with respect to the integral inner product) are defined by Rodriguez' formula:

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}\left(x^2-1\right)^n$$

yielding the polynomials

$$P_0(x) = 1 \quad P_1(x) = x \quad P_2(x) = \frac{1}{2}\left(3x^2-1\right)$$
$$P_4(x) = \frac{1}{8}\left(35x^4-30x^2+3\right) \quad P_5(x) = \frac{1}{8}\left(63x^5-70x^3+15x\right)$$
$$P_6(x) = \frac{1}{16}\left(231x^6-315x^4+105x^2-5\right) \quad \dots$$

The polynomials have the following properties:

- $\max_{-1\leq x\leq 1} |P_n(x)| = 1 \quad (n \in \mathbb{N}_0)$
- $\min_{-1\leq x\leq 1} P_n(x) = -1 \quad (n \in \mathbb{N})$
- $P_n(-x) = (-1)^n P_n(x)$ (symmetry or anti-symmetry)
- $P_n(1) = 1, P_n(-1) = (-1)^n$

the continuous square-sum of residuals
$S := \int_{-1}^{1}\left(y(x) - \sum_{j=0}^{m} a_j P_j(x)\right)^2 dx$ is minimal iff:

$$a_j = \frac{2j+1}{2}\int_{-1}^{1} y(x)P_j(x)\,dx \quad (j = 0, \dots, m)$$

yielding the continuous least-squares Legendre approximation polynomial $\sum_{j=0}^{m} a_j P_j(x)$ of degree m.

Differentials & Taylor-Formula

The differential of a function at a specific input value x_0 is $df(x_0) = f'(x_0)\,dx$ and has the rules

1. *Linearity:* $d(\alpha u \pm \beta v) = \alpha d(u) \pm \beta d(v) \quad (\alpha, \beta \in \mathbb{R})$
2. *Product:* $d(uv) = ud(v) + vd(u)$
3. *Quotient:* $d(u \div v) = (vd(u) - ud(v)) \div v^2$
4. *Chain rule:* $d(v \circ u) = (v' \circ u)d(u)$
5. *Inversion:* $d(u^{-1}(y)) = \frac{1}{u'(x)}dy \quad (x = u^{-1}(y))$

Univariate Taylor Formula

If $y = f(x)$ is defined on the interval $I = [x0, x_0 + h = x]$

$$f(x_0 + h) = f(x_0) + \frac{1}{1!}f^{(1)}(x_0)h + \frac{1}{2!}f^{(2)}(x_0)h^2 + \cdots + \frac{1}{n!}f^{(n)}(x_0)h^n + R_n(x_0, h)$$

With the remainder error term $R_n(x_0, h)$:

- *Lagrange form:*

$$\frac{h^{n+1}}{(n+1)!}f^{(n+1)}(\underbrace{x_0 + \vartheta \cdot h}_{\xi}) \quad (\xi \in (x_0, x_0 + h),\ \vartheta \in (0, 1))$$

- *Cauchy form:*

$$\frac{h^n(1-\vartheta)^n}{n!}f^{(n+1)}(\underbrace{x_0 + \vartheta \cdot h}_{\xi}) \ (\xi \in (x_0, x_0 + h), \vartheta \in (0, 1))$$

Multivariate Taylor Formula

$$f(\vec{x}_0 + \vec{h}) = \overbrace{f(\vec{x}_0) + \frac{1}{1!}\vec{\nabla}f(\vec{x}_0) \cdot \vec{h}}^{\text{Linear approximation}} + \underbrace{\sum_{|\alpha|=2}^{N} \frac{1}{\alpha!}\frac{\partial^{|\alpha|}f(\vec{x}_0)}{\partial x^\alpha}\vec{h}^\alpha}_{\text{Higher order deriv.}} + \underbrace{\sum_{|\alpha|=N+1} R_\alpha(\vec{x}_0, \vec{h})\vec{h}^\alpha}_{\text{Reminder term}}$$

where the reminder terms are absolutely bounded by $\max_{\vec{x}\in S}\left|\frac{1}{\alpha!}\frac{\partial^\alpha f(\vec{x})}{\partial x^\alpha}\right|$ with $|\alpha| = N + 1$ and $S = \vec{x}_0 + ([-h_1, h_1] \times [-h_2, h_2] \times \cdots \times [-h_n, h_n])$ is an n-dimensional "rectangle" at center $\vec{x}_0$.

Jacobian

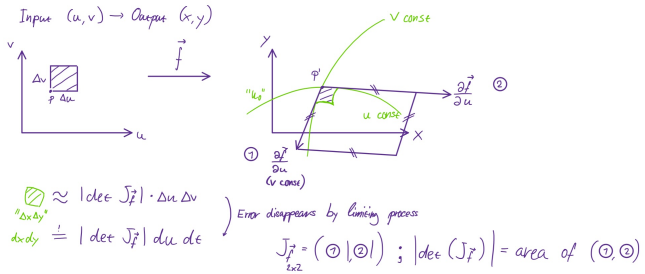
For each vector-valued function $\vec{f} : \mathbb{R}^n \mapsto \mathbb{R}^m : \vec{y} = \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x}))$ we can find a Jacobian matrix with the rows being the gradients $\vec{\nabla}f_j$:

$$J_f(\vec{x}) = \begin{bmatrix} \frac{\partial f_1(\vec{x})}{\partial x_1} & \frac{\partial f_1(\vec{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\vec{x})}{\partial x_n} \\ \frac{\partial f_2(\vec{x})}{\partial x_1} & \frac{\partial f_2(\vec{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\vec{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m(\vec{x})}{\partial x_1} & \frac{\partial f_m(\vec{x})}{\partial x_2} & \cdots & \frac{\partial f_m(\vec{x})}{\partial x_n} \end{bmatrix}_{m\times n}$$

in the case of the Jacobian matrix being a square matrix ($m = n$), it has a determinant $\det(J_f(\vec{x}))$. The determinant of a linear transformation A (square matrix) allows the computation of "transformed volumes": $\text{vol}(A(C)) = |\det(A)|\text{vol}(C)$, meaning for $y_j = f_j(x_1, \ldots, x_n) \quad (j = 1, ..., m = n)$:

$$\underbrace{dy_1 dy_2 \cdots dy_n}_{\text{transformed volume element}} = \det(J_f(\vec{x}))\underbrace{dx_1 dx_2 \cdots dx_n}_{\text{volume element}}$$

(requiring that J_f is regular/invertible, i.e. $\det_{J_f} \neq 0$)



We therefore can do $\iint f(g(u,v), h(u,v))\det_J \, dudv$ when calculating a volume in x, y using functions $x = g(u, v)$ and $y = h(u, v)$ and the respective determinant of the transformation jacobian.

For Example: We want to compute the following integral on $\mathbb{R}^2$:

$$\iint_{\mathbb{R}^2} e^{-\frac{x^2+y^2}{2}} \, dxdy$$

by transformation to polar coordinates (r, φ). Hence, we have the transformation T: $(r, \varphi) \mapsto_T (x, y)$ with

$$T(r, \varphi) = \begin{bmatrix} r\cos(\varphi) \\ r\sin(\varphi) \end{bmatrix} = \begin{bmatrix} x(r, \varphi) \\ y(r, \varphi) \end{bmatrix}$$

and the Jacobian matrix:

$$J_T = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \varphi} \\ \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \varphi} \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & -r\sin(\varphi) \\ \sin(\varphi) & r\cos(\varphi) \end{bmatrix}$$

with the determinant $\det(J_T) = r\cos^2(\varphi) + r\sin^2(\varphi) = r$. The transformed integral therefore looks like this:

$$\iint_{\mathbb{R}^2} e^{-\frac{x^2+y^2}{2}} \, dxdy \Rightarrow \int_0^{2\pi}\int_0^{\infty} e^{-\frac{r^2(\sin^2(\varphi)+\cos^2(\varphi))}{2}} \det J_T \, drd\varphi$$

$$= \int_0^{2\pi}\left(\int_0^{\infty} e^{-\frac{r^2}{2}} r \, dr\right)d\varphi = \int_0^{2\pi}\left[-e^{-\frac{r^2}{2}}\right]_0^{\infty} d\varphi$$

$$= \int_0^{2\pi} 1 \, d\varphi = [\varphi]_0^{2\pi} = 2\pi$$

Numerical ODEs

Equations of the form $y'(x) = f(x, y(x))$.

Explicit Euler Method

Applying Taylor expansion with order $p = 1$:

$$y(x + h) = y(x) + y'(x)h + \frac{y^{(2)}(\xi)}{(2)!}h^2$$

$$y(x + h) = y(x) + f(x, y(x))h + \frac{1}{2!}\left(\frac{\partial f(\xi, y(\xi))}{\partial x}1 + \frac{\partial f(\xi, y(\xi))}{\partial y}f(\xi, y(\xi))\right)h^2$$

resulting in the following scheme:

$$y_0 = y(x_0)$$

$$y(x_0 + h) \approx y_1 = y_0 + f(x_0, y_0)h = y(x_0) + y'(x_0)h$$

$$\vdots$$

$$y_{k+1} = y_k + f(x_k, y_k)h \quad (k = 0, \ldots, n)$$

with h as step size (not necessarily constant).

Global Error

The global error is the maximum difference to the exact solution

$$\max_{0\le i\le k}|y_i - y(x_i)|$$

Convergence: Global error $\to 0$ with $\max_i h_i \to 0$

Local Error

The local/step error is the error after one step from a correct starting point, excluding past errors made:

Slope error : $\underbrace{\tau_h(x_n) := \frac{y(x_n + h) - y(x_n)}{h}}_{\text{True slope}} - \underbrace{(y'(x_n) = f)}_{\text{approx. slope}}$

Output local error : $h\tau_h(x_n) := \underbrace{y(x_n + h)}_{\text{True output}} - \underbrace{(y(x_n) + f \cdot h)}_{\text{Euler output}}$

For higher order slopes, the approximate slope is

$$\frac{y'(x_n)}{1!} + \frac{y''(x_n)}{2!}h^1 + \ldots + \frac{y^{(p)}(x_n)}{p!}h^{p-1}$$

Consistency: $\tau_h \to 0$ for $\max_i h_i \to 0$

Higher-Order Local Taylor Methods

The explicit Euler method used order $p = 1$. We can expand to a scheme of order p:

$$y_0 = y(x_0)$$

$$y_{k+1} = y_k + \frac{y'_k}{1!}h_k + \frac{y''_k}{2!}h_k^2 + \frac{y'''_k}{3!}h_k^3 + \cdots + \frac{y_k^{(p)}}{p!}h_k^p$$

The local Taylor method is convergent with order p:

$$\underbrace{\max_{0\le i\le k}|y_i - y(x_i)|}_{\text{global error}} = \mathcal{O}(h^p) \quad (h \to 0)$$

(simplified, with some constants $= \max_{0\le i\le k} h_i$).
However, they come with rather high costs arising from computing higher-order derivatives.

Runge-Kutta Methods

Designed to mimic the Taylor method, but without computing derivatives. Instead, they use function evaluations at intermediate points.

Explicit Midpoint/Heun Method of Order 2

For the two-stage Runge-Kutta method, we have

$$k_1 = hf(x, y)$$
$$k_2 = hf(x + mh, y + mk_1)$$
$$y(x + h) = y(x) + ak_1 + bk_2$$

using the "traditional notation" with h being a part of the function evaluation.
With constants $a = \frac{1}{2}, b = \frac{1}{2}$ and $m = 1$, we have Heun's method:

$$y_0 = y(x_0)$$
$$y_{i+1} = y_i + \frac{h_k}{2}\left(f(x_k, y_k) + f(x_k + h_k, y_k + hf(x_k, y_k))\right)$$

Explicit "Classical" Runge-Kutta Method of Order 4

Assuming that r.h.s. function f has continuous partial derivatives up to order 4, we have

$$k_1 = hf(x, y)$$
$$k_2 = hf(x + \frac{1}{2}h, y + \frac{1}{2}k_1)$$
$$k_3 = hf(x + \frac{1}{2}h, y + \frac{1}{2}k_2)$$
$$k_4 = hf(x + h, y + k_3)$$
$$y(x + h) = y(x) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

General Framework (Butcher Tableau)

A step from x_n to $x_{n+1} = x_n + h_n \quad (n \in \mathbb{N})$ where h_n is the step size (variable or fixed) is given by

$$\left.\begin{aligned} k_1 &= f(x_n + c_1 h_n, y_n + h_n \textstyle\sum_{j=1}^{s} a_{1,j} k_j) \\ k_2 &= f(x_n + c_2 h_n, y_n + h_n \textstyle\sum_{j=1}^{s} a_{2,j} k_j) \\ &\vdots \\ k_s &= f(x_n + c_s h_n, y_n + h_n \textstyle\sum_{j=1}^{s} a_{s,j} k_j) \end{aligned}\right\} \begin{array}{l} s \text{ stages,} \\ h \text{ not part of it} \\ \text{(more general)} \end{array}$$

and the next step is

$$y_{i+1} = y_n + h_n \sum_{j=1}^{s} b_j k_j$$

the coefficients $a_{i,j}, b_i, c_i$ are given in the Butcher tableau:

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

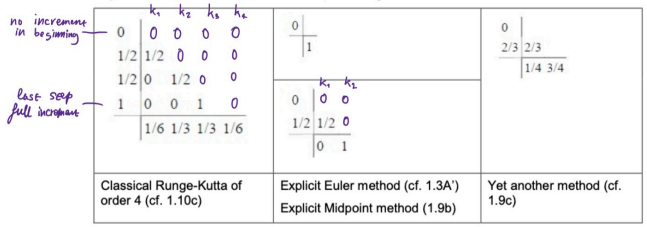
for **explicit** methods, the tableau is *lower triangular* ($a_{ij} = 0 \ (j \geq i)$):

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

and has to satisfy the row-sum property $c_i = \sum_{j=1}^{s} a_{i,j} = \sum_{j=1}^{i-1} a_{i,j} \ (i = 2, \ldots, s)$ (for "simulation" of Taylor method).
The **local error** of a step is

$$\tau_h(x_i) := \frac{y(x_i + h) - y(x_i)}{h} - \left(\sum_{j=1}^{s} b_j k_j\right)$$

some common butcher tableaus for explicit methods are:



FSAL property (First Same As Last) means that $a_{s,j} = b_j \ (j = 1, \ldots, s-1)$ and $b_s = 0$ and $\sum_{j=1}^{s-1} b_j = 1$. This allows reusing k_s as k_1 for the next step.

Adaptive Explicit Runge-Kutta Methods

Use estimates of the local error to adjust the step size h_i. To get an estimate of the local error, we use two different methods of different orders $p\ \hat{p}$, typically with $\hat{p} = p \mp 1$ and noted as $p(\hat{p})$. They share the *same coefficients matrix* and are called **embedded methods**.
The combined butcher tableau is:

$$\begin{array}{c|ccccc} c_1 = 0 & a_{1,1} = 0 & a_{1,2} = 0 & \cdots & a_{1,s-1} = 0 & a_{1,s} = 0 \\ c_2 & a_{2,1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ c_{s-1} & a_{s-1,1} & a_{s-1,2} & \cdots & 0 & 0 \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \\ & \hat{b}_1 & \hat{b}_2 & \cdots & \hat{b}_{s-1} & \hat{b}_s \end{array}$$

and thus receiving two solutions

$$y_{i+1} = y_i + h_i \sum_{j=1}^{s} b_j k_j$$

$$\hat{y}_{i+1} = y_i + h_i \sum_{j=1}^{s} \hat{b}_j k_j$$

and an estimate for the local error (double bars indicating norm function when applying to systems of differential equations):

$$||e_i|| = \left|\left| h_i \sum_{j=1}^{s} (b_j - \hat{b}_j) k_j \right|\right|$$

We can then implement the following algorithm:

1. *Choose an appropriate method* (e.g. the Heun-Euler method with $p = 2$ and $\hat{p} = 1$, writing HE2(1) where Heun is the primary method and Euler is embedded in Heun).
2. Choose initial step size h_0 and an absolute tolerance ε_a (= accuracy) as well as a relative tolerance ε_r (= precision). Given accuracy goal ag and precision goal pg, the tolerances can be computed as $\varepsilon_a = 10^{-ag}$ and $\varepsilon_r = 10^{-pg}$. The errors form the tolerance parameter/error norm $\varepsilon = \varepsilon_a + \varepsilon_r |y_n|$
3. Then start repeating with $i = 0$:
4. (a) Approximate local error $||e_i||$ and compare it to the error norm ε.

   (b) If $||e_i|| \leq \varepsilon \Rightarrow$ error below tolerance, the step is *accepted*. We therefore calculate y_{i+1} according to the primary method, set $x_{i+1} = x_i + h_i$ and increase i. We also increase the step size:

   $$h_{\text{next}} = h_{i+1} = h_i \left(\frac{||e_i||}{\varepsilon}\right)^{-\frac{1}{\varrho}} \geq h_i$$

   with $\varrho = \min\{p, \hat{p}\} + 1$ (e.g. 2 for HE2(1))

   (c) If $||e_i|| > \varepsilon$, the step is *rejected*. We therefore repeat without increasing i and reduce the step size:

   $$h_{\text{next}} = h_i \left(\frac{||e_i||}{\varepsilon}\right)^{-\frac{1}{\varrho}} < h_i$$

   with $\varrho = \min\{p, \hat{p}\} + 1$ (e.g. 2 for HE2(1)) and h_{next} being the step size for the next iteration.

   (d) Repeat until desired interval is reached.

Heun-Euler 2(1) Method

Has the combined butcher tableau

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

With a single step for Heun:

$$y_{k+1} \approx y_k + \frac{1}{2}h_k \left(\underbrace{f(x_k, y_k)}_{\propto k_1} + \underbrace{f(x_k + h_k, y_k + h_k f(x_k, y_k))}_{\propto k_2}\right)$$

and Euler:

$$\hat{y}_{k+1} = y_k + f(x_k, y_k) h_k$$

We also have a simple local error estimate in the form of $e_k = y_{k+1} - \hat{y}_{k+1}$ which is for HE2(1):

$$e_k = \frac{1}{2}h_k(k_2 - k_1)$$

Stability of Explicit Runge-Kutta Methods

Stability refers to the evolution of the global error over time when continuing using the same step size h.
We can benchmark the stability of a method by applying it to the Dahlquist test equation:

$y' = Ay \quad y(0) = 1$

with $A \in \mathbb{C}$ or A a constant matrix in case that y is a vector. The exact solution is $y(t) = e^{At}$, or:

$$y = e^{\Re(A)x} \left(\underbrace{\cos(\omega x) + i \sin(\omega x)}_{\text{oscillating } = e^{i\omega x}}\right) \quad \text{with } \omega = \Im(A)$$

Example: Absolute stability analysis for Heun's method

The Heun method is a Runge-Kutta method of order 2 with 2 stages:

$$y_0 = y(x_0)$$
$$y_{i+1} = y_i + \frac{1}{2}h\big(f(x_i, y_i) + f(x_i + h, y_i + hf(x_i, y_i))\big)$$

applied to Dahlquist test equation $y' = Ay$ ($f(x, y) = Ay$), the scheme becomes:

$$\begin{aligned} y_0 &= y(0) = 1 \\ y_{i+1} &= y_i + \frac{1}{2}h\big(Ay_i + A(y_i + hAy_i)\big) \\ &= \underbrace{\left(1 + hA + \frac{1}{2}(hA)^2\right)}_{\text{Function } F(hA)=F(z)} y_i \\ &= F(z)y_i \quad \text{with } z := hA \text{ and } F(z) := 1 + z + \frac{1}{2}z^2 \end{aligned}$$

The following stability cases can be distinguished:

1. **Stable** if $\Re(A) < 0$ and thus the solution's amplitude is exponentially decreasing. This implies that $|F(z)| < 1$ for all $z = hA$.
2. **Unstable** if $\Re(A) > 0$ and thus the solution's amplitude is exponentially increasing. This implies that $|F(z)| > 1$ for all $z = hA$.
3. **Neutral** if $\Re(A) = 0$ and thus the solution's amplitude is constant. This implies that $|F(z)| = 1$ for all $z = hA$ but usually is algebraically hard to achieve.

in the case for Heun's method, we have a stability region of $hA \in (-2, 0)$ for $hA \in \mathbb{R}$.

Stiffness

Stiffness has a negative impact on the stability of a numerical scheme and enforces small step-sizes leading to high computational costs.

To get a stiffness analysis in a simple, one-dimensional, explicit case, we substitute $A = \frac{\partial f}{\partial y}$ for the Dahlquist test equation $y' = Ay$. If we have a bad step size h for a stiff problem then the numerical solution will be outside of the stability region and thus unstable.

Stiffness detection for explicit Runge-Kutta methods can be done by using the approximation

$$\tilde{\lambda} = \frac{||k_s - k_{s-1}||}{||g_s - g_{s-1}||}$$

as an estimate for the partial derivative $\frac{\partial f}{\partial y}$ at position $x + h$ provided that $c_{s-1} = c_s = 1$ for an explicit Runge-Kutta method with at least two k values:

$$k_{s-1} = f(x + c_{s-1}h, \underbrace{y + h\sum_{j=1}^{s} a_{s-1,j}k_j}_{g_{s-1}})$$

$$k_s = f(x + c_s h, \underbrace{y + h\sum_{j=1}^{s} a_{s,j}k_j}_{g_s})$$

The value $\tilde{\lambda}$ takes the role of $|\Re(A)|$ in the stability analysis. By testing $|h\tilde{\lambda}|$ agains the absolute boundary values of the stability region, we can detect stiffness.

Systems of ODEs

$\vec{y}'(x) = \vec{f}(x, \vec{y}(x))$ with $\vec{y}(x_0) = \vec{y}_0$ and m dimensions

We can transform a higher-order ODE by substituting $z_1 = z', z_2 = z'', \ldots, z_{n+1} = h^{(n+1)}$ into a system of first-order ODEs: $z' = z_1, z_1' = z_2, \ldots$

Basic Math

Roots

$$\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{a \cdot b}$$
$$\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$$
$$(\sqrt[n]{a})^m = \sqrt[n]{a^m}$$
$$\sqrt[m]{\sqrt[n]{a}} = \sqrt[m \cdot n]{a}$$

Logarithm

$$\log_n(a \cdot b) = \log_n(a) + \log_N(b)$$
$$\log_n(a \div b) = \log_n(a) - \log_N(b)$$
$$\log_n(a^b) = b \cdot \log_n(a)$$

Trigonometry

$$\tan\theta = \frac{\sin\theta}{\cos\theta}$$
$$\sin -\theta = -\sin\theta \ (\cos \text{ same})$$
$$\sin 2\theta = 2\sin\theta\cos\theta$$
$$\cos 2\theta = 2\cos^2\theta - \sin^2\theta = 2\cos^2\theta - 1 = 1 - 2\sin^2\theta$$
$$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$$
$$\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta$$

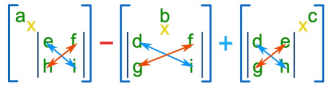
Binomial Coefficient

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Faculties

$$0! = 1, 1! = 1, 2! = 2, 3! = 6, 4! = 24, 5! = 120, 6! = 720$$

Determinant



$$\det\begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$

$$\det\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = a(ei - fh) - b(di - fg) + c(dh - eg)$$

■ Definition 3.11

Es sei $A = [a_{ij}]$ eine n x n - Matrix mit n > 2.

Dann bezeichnet

A_{ij} für i, j = 1, 2, ..., n

diejenige (n – 1) x (n – 1) - Untermatrix von A, welche durch Weglassen der i-ten Zeile und der j-ten Spalte von A entsteht.

$$A_{ij} = \begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ \vdots & & \vdots & & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & & \vdots & & \vdots \\ a_{n1} & \cdots & a_{nj} & \cdots & a_{nn} \end{bmatrix}$$

Weiter heissen die Zahlen

$c_{ij} = (-1)^{i+j} \cdot \det(A_{ij})$

Cofaktoren der Matrix A.

■ Satz 3.11 / Definition 3.12

Es sei $A = [a_{ij}]$ eine n x n - Matrix mit n > 2 und c_{ij} ihre Cofaktoren.

Dann haben die Ausdrücke

$a_{i1} \cdot c_{i1} + a_{i2} \cdot c_{i2} + \ldots + a_{in} \cdot c_{in}$ für i = 1, 2, ..., n

und die Ausdrücke

$a_{1j} \cdot c_{1j} + a_{2j} \cdot c_{2j} + \ldots + a_{nj} \cdot c_{nj}$ für j = 1, 2, ..., n

alle den gleichen Wert. Dieser heisst **Determinante** von A und wird mit **det(A)** bezeichnet.

Properties

- $\det\left(A^{-1}\right) = \frac{1}{\det(A)}$
- $\det(A^T) = \det(A)$
- $\det(I) = 1$
- $\det(cA) = c^n \det(A)$ (for an $n \times n$ matrix)
- $\det(AB) = \det(A)\det(B)$
- $\det(A) = \prod_{i=1}^{n} \lambda_i$

Derivatives

$f(x)$	$\frac{df}{dx}$
$\sinh(x)$	$\cosh(x)$
$\cosh(x)$	$\sinh(x)$
$\operatorname{arcsinh}(x)$	$1 \div \sqrt{x^2+1}$
$\operatorname{arccosh}(x)$	$1 \div \sqrt{x^2-1}$ $(1 < x)$
$\tan(x)$	$\cos^{-2}(x)$
$\log(x)$	x^{-1}

Integrals

$$\int x^n \, dx = \tfrac{1}{n+1}x^{n+1} + C$$
$$\int \tfrac{1}{x} \, dx = \ln|x| + C$$
$$\int \tfrac{1}{ax+b} \, dx = \tfrac{1}{a}\ln|ax+b| + C$$
$$\int \tfrac{1}{(x+a)^2} \, dx = -\tfrac{1}{x+a} + C$$
$$\int \tfrac{1}{1+x^2} = \tan^{-1}x + C$$
$$\int \ln ax \, dx = x\ln ax - x + C$$
$$\int e^{ax} \, dx = \tfrac{1}{a}e^{ax} + C$$
$$\int \sin(ax) \, dx = -\tfrac{1}{a}\cos(ax) + C$$
$$\int \sin^2(ax) \, dx = \tfrac{x}{2} - \tfrac{\sin(2ax)}{4a} + C$$
$$\int x\cos x \, dx = \cos x + x\sin x + C$$
$$\int \sinh(ax) \, dx = a^{-1}\cosh ax + C$$
$$\int \cosh(ax) \, dx = a^{-1}\sinh ax + C$$

Integration Techniques

Integration by Parts

$$\int_a^b u(x)v'(x) \, dx = [u(x)v(x)]_a^b - \int_a^b u'(x)v(x) \, dx$$

Or, with $u = u(x)$, $du = u'(x) \, dx$, $v = v(x)$ and $dv = v'(x) \, dx$:

$$\int u \, dv = uv - \int v \, du$$

Substitution

$$\int_a^b f(g(x)) \cdot g'(x) \, dx = \int_{g(a)}^{g(b)} f(u) \, du$$

Leibniz Integral Rule

$$\frac{d}{dx}\left(\int_{a(x)}^{b(x)} f(x,t) \, dt\right) =$$
$$f(x,b(x)) \cdot \frac{d}{dx}b(x) - f(x,a(x)) \cdot \frac{d}{dx}a(x) + \int_{a(x)}^{b(x)} \frac{\partial}{\partial x}f(x,t) \, dt$$

Special case where $a(x) = a = \text{const.}$ and $b(x) = b = \text{const.}$:

$$\frac{d}{dx}\left(\int_a^b f(x,t) \, dt\right) = \int_a^b \frac{\partial}{\partial x}f(x,t) \, dt$$

Converting second-order ODEs

For a second-order equation $y'' = f(x, y, y')$, we can introduce $p = y'$, transforming the equation to $p' = f(x, y, p)$ and thus receiving a system of ODEs with initial conditions $p(x_0) = y_1, y(x_0) = y_0$.