1 Numerische Quadratur

1. Interpolationspolynom

Lagrange Polynome

$$p_n(x) = \sum_{j=0}^{n} y_j L_j^n(x), \quad L_j^n(x) = \prod_{\substack{i=0 \\ i\neq j}}^{n} \frac{x-x_i}{x_j-x_i}, \quad j = 0,1,\dots,n$$

hat Grad n $\quad [L_0^0 = 1]$

3. Numerische Quadratur

$$\int_a^b f(x)\,dx \approx \int_a^b p_n[f \mid x_0 \dots x_n](x)\,dx = \sum_{j=0}^{n} f(x_j) \int_a^b L_j^n(x)\,dx$$

$$\Rightarrow \int_a^b f(x)\,dx \approx \sum_{j=0}^{n} f(x_j) w_j \quad \text{wobei} \quad w_j = \int_a^b L_j^n(x)\,dx$$

unabhängig von f

→ wichtige Beispiele:

i. Mittelpunktsregel (MR) $(n=0)$

- Knoten: $x_0 = \frac{a+b}{2}$
- Quadratur: $Q_0 = \frac{b-a}{2} f\left(\frac{a+b}{2}\right)$

ii. Trapezregel (TR) $(n=1)$

- Knoten: $x_0 = a, \; x_1 = b$
- Quadratur: $\frac{b-a}{2}\left(f(a) + f(b)\right)$

iii. Simpsonregel (SR) $(n=2)$

- Knoten: $x_0 = a, \; x_1 = \frac{b+a}{2}, \; x_2 = b$
- Quadratur: $\frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$

2. Interpolationsfehler

$$\rightarrow e(x) = f(x) - p_n[f \mid x_0 \dots x_n](x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{j=0}^{n} (x - x_j)$$

$$\rightarrow \text{Maximaler Fehler: } \max |e(x)| = \frac{|f^{(n+1)}(\xi)|}{(n+1)!} (b-a)^{n+1}$$

4. Quadraturfehler

→ Genauigkeitsgrad q: $Q[x^k] = I[x^k]$ für $k = 0,1,\dots,q$ und $Q[x^{k+1}] \neq I[x^{k+1}]$

→ Ordnung s: $s = q+1$

für NC QR gilt:
- n gerade $\Rightarrow q = n+1, \; s = n+2$
- n ungerade $\Rightarrow q = n, \; s = n+1$

$$\rightarrow \text{Fehler: } E[f] \leq \frac{\|f^{(q+1)}\|_\infty}{(q+1)!} (b-a)^{q+2} = \frac{\|f^{(s)}\|_\infty}{s!} (b-a)^{s+1}$$

5. Summierte Quadraturregel

→ Intervall $I = [a,b]$ wird in N Teile zerteilt. $x_j = a + hj, \; j = 0,1,\dots,N, \; h = \frac{b-a}{N}$

→ Fehler: $E^N[f] \leq \frac{\|f^{(s)}\|_\infty}{s!} h^s (b-a) = \mathcal{O}(h^s)$ → s bestimmen: log-log-plot

↳ f muss genügend glatt sein!

- Summierte Mittelpunktsregel: $Q_0^N[f] = \sum_{j=1}^{N} h f\left(\frac{x_{j-1}+x_j}{2}\right)$
- Summierte Trapezregel: $Q_1^N[f] = \frac{h}{2}\left(f(a) + 2\sum_{j=0}^{N-1} f(x_j) + f(b)\right)$
- Summierte Simpsonregel: $Q_2^N[f] = \frac{h}{6}\left[f(a) + 2\sum_{j=0}^{N-1} f(x_j) + 4\sum_{j=0}^{N} f\left(\frac{x_{j-1}+x_j}{2}\right) + f(b)\right]$

7. Adaptive-Quadratur

→ Verwende feinere und grobere Teil-Intervale

→ Fehlerschätzer: Vergleiche zwei Quadraturregeln

$$\left.\begin{aligned} E^1[f] &\approx \frac{2^s}{2^s-1}\,|Q^1[f]-Q^2[f]| \\ E^2[f] &\approx \frac{1}{2^s-1}\,|Q^1[f]-Q^2[f]| \end{aligned}\right\} s \text{ von } QR^1$$

→ Pseudo MATLAB

```
function Q = adapt_quad(f, a, b, tol)
  Q_1 = quad1(f, a, b)
  Q_2 = quad2(f, a, b)

  E = 1/(2^s - 1) |Q_1 - Q_2|

  if E < tol
     Q = Q_2
  else
     Q_1 = adapt_quad(f, a, (a+b)/2, tol/2)
     Q_2 = adapt_quad(f, (a+b)/2, b, tol/2)

     Q = Q_1 + Q_2
  end
end
```

8. Zweidimensionale Quadratur

$$I[f] = \int_a^b \int_c^d f(x,y)\,dx\,dy = \sum_{i=0}^{n}\sum_{j=0}^{m} \omega_i \omega_j f(x_i, y_j)$$

$$\tilde{x}_0 = \frac{b-a}{2}x_0 + \frac{a+b}{2} =$$

$$\tilde{x}_1 = \frac{b-a}{2}x_1 + \frac{a+b}{2} =$$

$$\tilde{\omega}_0 = \frac{b-a}{2}\omega_0 = \frac{b-a}{2},$$

$$\tilde{\omega}_1 = \frac{b-a}{2}\omega_1 = \frac{b-a}{2}.$$

② Einschrittverfahren

1. Grundbegriffe

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

→ Allgemeines AWP

$$\left.\begin{aligned} \dot{y}_1(t) &= f_1(t, y_1(t), \dots, y_n(t)) \\ &\vdots \\ \dot{y}_n(t) &= f_n(t, y_1(t), \dots, y_n(t)) \end{aligned}\right\} \quad y(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{bmatrix}$$

→ Reduktion von Systemen

- gegeben $y^{(n)} = f(t, y(t), \dot{y}(t), \dots, y^{(n-1)}(t))$
- $z_0(t) = y(t)$, $z_1(t) = \dot{y}(t) = \dot{z}_0(t), \dots, y^{(n-1)}(t) = f(t, z_0, z_1, \dots)$

  ↳ definiere

$$\dot{\vec{z}}(t) = \vec{g}(t, \vec{z}(t)) \quad \text{wobei} \quad \vec{z}(t) = \begin{bmatrix} z_0(t) \\ z_1(t) \\ \vdots \\ z_{n-1}(t) \end{bmatrix}, \quad \vec{g}(t, \vec{z}(t)) = \begin{bmatrix} z_1(t) \\ z_2(t) \\ \vdots \\ f(t, z_0, z_1 \dots) \end{bmatrix}$$

→ Automatisieren

- gegeben $\dot{\vec{y}}(t) = f(t, \vec{y}(t))$

$$\vec{z} = \begin{bmatrix} \vec{y} \\ t \end{bmatrix} = \begin{bmatrix} \vec{z}_1 \\ z_{n+1} \end{bmatrix} \to g(\vec{z}(t)) = \begin{bmatrix} \vec{f}(t, \vec{y}(t)) \\ 1 \end{bmatrix} = \begin{bmatrix} \vec{f}(z_{n+1}, \vec{z}_1) \\ 1 \end{bmatrix}$$

↳ damit $\dot{\vec{z}}(t) = g(\vec{z}(t))$

2. Existenz/Eindeutigkeit

→ Lipschitz-Stetigkeit

$\exists \lambda > 0$ s.d. $\forall x, \hat{x} \in \mathbb{R}^n, t \in \mathbb{R} \to \| \vec{f}(t,x) - f(t,\hat{x}) \| \leq \lambda \| x - \hat{x} \|$

→ Stetig-differenzierbare Funktionen sind Lipschitz

→ Funktionen mit beschränkte Ableitung sind Lipschitz

→ Picard-Lindelöf: Sei $\vec{f}$ stetig in $(t, \vec{y})$ und Lipschitz in $\vec{y}$ auf $t \in [t_0, t_0 + \delta]$, $\delta > 0$. So hat $\dot{\vec{y}}(t) = \vec{f}(t, \vec{y}(t))$ eine eindeutige Lösung für zumindest eine kurze Zeit $t \in [t_0, t_0 + \varepsilon]$, $\varepsilon > 0$.

3. Runge-Kutta-Verfahren

$$h = \frac{\overset{\text{Endzeit}}{T} - t_0}{N}$$

$$\to \dot{y}(t) = f(t, y(t)) \to y(t) = y_0 + \int_{t_0}^{t} f(\tau, y(\tau))\, d\tau$$
$$= y_0 + h \int_0^1 f(t_0 + h\tau, y(t_0 + h\tau))\, d\tau$$

→ Dieses Integral können wir mittels Quadratur lösen

- Mittelpunktsregel: $y(t_1) \approx y(t_0) + h f(t_0 + \frac{h}{2}, y(t_0 + \frac{h}{2}))$

Problem: Implizites Aufrufen von $y(t_1)$, was wir suchen.

↳ Lösung: verwende Euler-Verfahren um eine Approximation zu erhalten.

(Wir verwenden eine schlechte Approximation – Euler – um eine bessere zu erhalten)

$$\Rightarrow y(t_1) \approx y(t_0) + h f(t_0 + \tfrac{h}{2}, y_0 + \tfrac{h}{2} f(t_0, y_0))$$

$$\left.\begin{aligned} k_1 &= f(t_j, y_j) \\ k_2 &= f(t_j + \tfrac{h}{2}, y_j + \tfrac{h}{2} k_1) \end{aligned}\right\} \quad y_{j+1} = y_j + h k_2 \quad \text{für } j = 0, 1, \dots, N-1$$

Verbesserte Polygonzug-Methode von Euler **oder** Verfahrung von Runge 2. Ordn.

- Trapezregel: $y(t_1) \approx y(t_0) + \frac{h}{2}[f(t_0, y(t_0)) + f(t_1, y(t_1))]$

$$y(t_1) \approx y(t_0) + \frac{h}{2}[f(t_0, y(t_0)) + t_0 + h f(t_0, y(t_0))]$$ Euler

$$\left.\begin{aligned} k_1 &= f(t_j, y_j) \\ k_2 &= f(t_j + h, y_j + h k_1) \end{aligned}\right\} \quad y_{j+1} = y_j + \frac{h}{2}(k_1 + k_2)$$

↳ Verfahren von Heun

→ **Runge Kutta ESV:** Ein s-stufiger RK-ESV ist definiert durch

$$\boxed{y_{j+1} = y_j + h\sum_{i=1}^{s} b_i k_i} \quad \text{wobei} \quad \boxed{k_i = f\Big(t_j + c_i h,\ y_j + h\sum_{l=1}^{s} a_{il} k_l\Big)}$$

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & & & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \equiv \begin{array}{c|c} \vec{c} & A \\ \hline & b^T \end{array}$$

- s .... Anzahl Stufen
- c_i .... Knoten
- b_i .... Gewichte
- a_{il} ...... RK-Matrix/Koeffizienten

↳ Butcher-Tableau (BT)

- Explizite RK-ESV → BT ist eine untere Dreiecks-Matrix mit Nullen auf der Diagonale

4. Fehler für ESV

→ $y_{j+1} = y_j + h\varnothing(t_j, y_j, h) \to \varnothing$ = Verfahrens-/Inkrementsfunktion

i. RK ⇒ $\varnothing = \sum_{i}^{s} b_i k_i$

ii. Explizit ⇒ einfach, $\varnothing$ zu berechnen
Implizit ⇒ Man muss ein LGS lösen, um $\varnothing$ zu berechnen

→ **Definitionen:**

- $t_j = t_0 + h\cdot j$, $j = 1,2,\dots,N$ und $h = \frac{T-t_0}{N}$
- Exakte Lösung bei $t_j \to y(t_j)$
- Approximierte Lösung bei $t_j \to y(t_j) \approx y_j$

→ **Globale Diskretisierungsfehler (GDF):** $E_j = y(t_j) - y_j$

→ **Konvergenzordnung (KO):** $p \mid E = \max_{j\in[0,N]} |y(t_j) - y_j| = \mathcal{O}(h^p)$

wie stark der maximale GDF wachst für verschiedene h

- Euler $\mathcal{O}(h^1)$
- Heun $\mathcal{O}(h^2)$
- verb. Euler $\mathcal{O}(h^2)$
- RK 4 $\mathcal{O}(h^4)$

→ **Lokale Diskretisierungsfehler (LDF)**

$$e_j := y^*(t_j) - \Big[\underbrace{y^*(t_{j-1}) + h\varnothing(t_{j-1}, y^*(t_{j-1}), h)}\Big] \quad * \text{ exakte Lösung}$$

1 Schritt mit ESV wobei die exakte Lösung verwendet wird
„Wie weit entfernt von der exakten Lösung ein ESV-Schritt sein kann"

→ **Konsistenzordnung:**

$$\tau_j = \underbrace{\frac{y(t_j) - y(t_{j-1})}{h}}_{h\to 0 \Rightarrow \dot{y}(t_j)} - \underbrace{\varnothing(t_{j-1}, y(t_{j-1}), h)}_{h\to 0 \Rightarrow f(t_j, y(t_j))} \to 0 \text{ für } h \to 0$$

- Konsistenzordnung p des Verfahrens: $\max_{j\in[0,N]} |\tau_j| = \mathcal{O}(h^p)$

„Verfahren ist konsistent" ⇒ $p \geq 1$

→ **Fehlerschätzer**

$$E \leq \Big(\underbrace{|y(t_0) - y_0|}_{\substack{\text{AW-Fehler} \\ \text{(vernachlässigen)}}} + \sum_{j=1}^{N} |e_j|\Big) \underbrace{e^{L(T-t_0)}}_{\substack{\text{werden wir auch} \\ \text{vernachlässigen}}} \longrightarrow \sum_{j=1}^{N} |e_j| = \underset{\text{lokal}}{N\,tol} < \underset{\text{global}}{Tol}$$

- $|e_j| \lesssim atol$ (absolute Toleranz)
- $|e_j| \lesssim |y_{j-1}|\, rtol$ (relative Toleranz)

$\Big\}\ |e_j| \lesssim atol + |y_{j-1}|\, rtol$

- Schrittweitenhalbierung:

i. $h \to y_{j+1} = y_j + h\varnothing(t_j, y_j, h)$

ii. $\frac{h}{2} \to \hat{y}_{j+1/2} = y_j + \frac{h}{2}\varnothing(t_j, y_j, h) \Rightarrow \hat{y}_{j+1} = \hat{y}_{j+1/2} + \frac{h}{2}\varnothing(t_j + h/2, \hat{y}_{j+1/2}, h)$

$$\Bigg\}\ \boxed{|\hat{e}_{j+1}| = \frac{|e_{j+1}|}{2^p}}$$

$$\boxed{\begin{aligned} |e_{j+1}| &\approx \frac{2^p}{2^p - 1} |\hat{y}_{j+1} - y_{j+1}| =: \varepsilon_{j+1} \\ |\hat{e}_{j+1}| &\approx \frac{1}{2^p - 1} |\hat{y}_{j+1} - y_{j+1}| =: \hat{\varepsilon}_{j+1} \end{aligned}}$$

→ a posteriori Fehlerschätzer

→ Adaptive Schrittweitensteuerung

- wir haben: $\varepsilon_{j+1} \approx C h^{p+1} \longrightarrow C = \frac{\varepsilon_{j+1}}{h^{p+1}}$
- wir wollen: $tol_{j+1} \approx C H^{p+1} \longrightarrow H \approx h \cdot \left(\frac{tol_{j+1}}{\varepsilon_{j+1}}\right)^{\frac{1}{p+1}}$

Schrittweiten-Vorschlag um $\varepsilon_{j+1} \approx tol_{j+1}$

$$\boxed{H = h \cdot \min\left(facmax,\ \max\left(facmin,\ fac\left(\frac{tol}{\varepsilon}\right)^{1/p+1}\right)\right)}$$

$(\approx 0.8 - 0.9)$

begrenze Vergrösserung (Faktor 2 bis 5)

begrenze Verkleinerung (Faktor ½ bis 0.1)

Sicherheitsfaktor