Woche 2

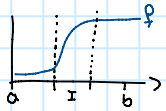
Freitag, 3. März 2023 09:51

Quadratur continued

- Wieso weitersuchen?
  - Bisher geht QF nicht auf Funktion ein → adaptive QF
  - QF scheinen noch nicht ausgereizt → Höhere Ordnung

Adaptive Quadratur:

Äquidistante Stützstellen können lokale Strukturen der Funktion nicht ausnutzen

Betrachte folgende Funktion f: 

f ist sehr "glatt", außer in I

Beispielsweise können wir die (zusammengesetzte) Trapezregel mit Partition $M = \{a = x_0 < \dots < x_m = b\}$ anwenden. Beachte, dass der lokale Fehler gegeben ist, als:

$$E_{lokal} = \left| \int_{x_k}^{x_{k+1}} f(x)dx - \frac{f(x_k)-f(x_{k+1})}{2}(x_{k+1}-x_k) \right| \leq (x_{k+1}-x_k)^3 \underbrace{\|f''\|_{L^\infty([x_k,x_{k+1}])}}_{= \max_{x\in[x_k,x_{k+1}]} |f''(x)|}$$

Beobachtung:
- kleiner Fehler bei I benötigt sehr kleine Unterteilung $[x_k, x_{k+1}]$
- äquidistante Stützstellen sind unpraktisch

Lösung: Passe Schrittweite an → Wie? Verwende QF als Fehlerschätzer

Fehlerschätzer: Man nimmt zwei QF Q^{p_1} und Q^{p_2} verschiedener Ordnungen $p_1 > p_2$ und verwendet als Mass für die lokale Komplexität:

$$\Delta = |Q^{p_1}(f;a,b) - Q^{p_2}(f;a,b)| \approx E_{lokal}$$

(oft verwendet man $Q^{p_1} = SR$, $Q^{p_2} = TR$). Falls der lokale Fehler zu gross ist, halbiert man das Intervall $[x_j, x_{j+1}]$. Dies wiederholt man so lange, bis Δ kleiner als eine gewisse Toleranz ist.

Quadratur mit erhöhter Ordnung:

Eine QF mit n Knoten hat 2n Freiheitsgrade, weil man die n Gewichte w_i auch frei wählen kann. In der Theorie, sollte bei perfekter Wahl der Knoten & Gewichte eine QF der Ordnung 2n erreichen kann.

Wichtig: Bei n Knoten ist die max. Ordnung 2n

Welche Verfahren gibt es? Wir betrachten als Referenzintervall $[a,b] = [-1,1]$

Verfahren: (n Knoten)

1) Gauss-Legendre: Frei wählbare Gewichte $\{w_i\}_{i=1}^n$ und frei wählbare Knoten

$\{x_i\}_{i=1}^{n} \subseteq [-1,1]$. Ordnung: $2n$

2) Gauss-Lobatto: Intervallenden $x_1 = -1$, $x_n = 1$ sind als Knoten festgelegt. Also nur $n-2$ Knoten frei wählbar (& ihre Gewichte). Ordnung: $2n-2$

3) Radau-Quadratur: Intervallanfang $x_1 = -1$ fix. $n-1$ Knoten frei wählbar. Ordnung: $2n-1$

Wie bestimmt man die Knoten und Gewichte? Man löst ein LGS für die zu erzielende Ordnung

Beispiel: $n = 3$, Lobatto

Wollen Ordnung $2n-2 = 4$

bekannte Knoten: $x_1 = -1$, $x_3 = 1$

Finde w_1, w_2, w_3 und x_4 $\Rightarrow$ 4 Gleichungen

Trick: Lobatto-QF ist symmetrisch

Symmetrisch: $w_i = w_{n+1-i}$, $x_i = -x_{n+1-i}$

Also: $x_2 = -x_2 \Rightarrow x_2 = 0$ } nur noch 2 Glg!

$w_1 = w_3$

$$I_n = \int_a^b \underbrace{x^n}_{=f(x_n)}\,dx = \frac{b^{n+1}-a^{n+1}}{n+1}$$

$$\begin{pmatrix} I_0 \\ \vdots \\ I_{2n-1} \end{pmatrix} = \begin{pmatrix} x_1^0 & \cdots & x_n^0 \\ x_1^1 & & \vdots \\ \vdots & & \vdots \\ x_1^{2n-1} & \cdots & x_n^{2n-1} \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$$

$$I_0 = \int_{-1}^{1} x^0\,dx \overset{!}{=} \sum_{i=1}^{3} w_i \underbrace{f(x_i)}_{=1} = w_1 + w_2 + w_3 = 2w_1 + w_2$$

I_1 fällt weg, da QF symmetrisch

$$I_2 = \int_{-1}^{1} x^2\,dx \overset{!}{=} \sum_{i=1}^{3} w_i \underbrace{f(x_i)}_{=x_i^2} \overset{x_2=0}{=} w_1 x_1^2 + w_3 x_3^2 \overset{\substack{w_1=w_3 \\ x_1=-x_3}}{=} 2w_1 \underbrace{x_1^2}_{=1}$$

Also mit $I_0 = 2$ und $I_2 = \int_{-1}^{1} x^2\,dx = \frac{2}{3}$ erhalten wir:

$$\begin{cases} 2 = 2w_1 + w_2 \\ \frac{2}{3} = 2w_1 \end{cases} \Rightarrow \begin{matrix} w_1 = w_3 = \frac{1}{3} \\ w_2 = \frac{4}{3} \end{matrix}$$ Simpson-Regel

Frage: Wie kommen wir von $[-1,1]$ auf $[a,b]$? Via Substitution!

$$\int_a^b f(x)\,dx = \int_{-1}^{1} f(x(u))\frac{dx}{du}\,du = \frac{b-a}{2}\int_{-1}^{1} f(x(u))\,du = \frac{b-a}{2}\sum_i w_i f(u_i) = \sum_i \underbrace{\left(\frac{b-a}{2}w_i\right)}_{w_i^*} f(\underbrace{x(u_i)}_{x_i^*})$$

$$x(u) = \frac{b-a}{2}u + \frac{b+a}{2}, \quad \frac{dx}{du} = \frac{b-a}{2}$$

Python:

- np.dot(a,b) ≡ Standard Skalarprodukt
- np.diff (x) ≡ gibt Array der Abstände von x_{j+1}, x_j zurück

  $diff([x_0, \dots, x_N]) = [x_1 - x_0, x_2 - x_1, \dots, x_N - x_{N-1}]$

Listen sind wie Arrays, aber ohne Funktionalität

list1 = [ ] leere liste

list1.append(item) ein Item wird am Ende der Liste hinzugefügt

list1[:] Item zugriff