Fachbereich Mathematik & Informatik
Freie Universität Berlin
Prof. Dr. Carsten Gräser, Tobias Kies

9. Übung zur Vorlesung

Numerik I

SoSe 2015

http://numerik.mi.fu-berlin.de/wiki/SS_2015/NumerikI.php

Abgabe: Fr., 26.06.2015, 12:00 Uhr

Allgemeine Hinweise

Die Punkte unterteilen sich in Theoriepunkte (TP) und Programmierpunkte (PP). Bitte beachten Sie die auf der Vorlesungshomepage angegebenen Hinweise zur Bearbeitung und Abgabe der Übungszettel, insbesondere der Programmieraufgaben.

1. Aufgabe (4 TP)
Sei $I = \{(i,j) \in \{0,\dots,3\}^2 \mid j \leq i\}$. Weiter sei für alle $(i,j) \in I$

$$z_{ij} := \frac{1}{4}\begin{pmatrix} i \\ j \end{pmatrix} \in \mathbb{R}^2.$$

Zeigen Sie, dass für alle $f \in \mathbb{R}^I$ genau ein $p \in \mathcal{P}_3 = \operatorname{span}\{x^i y^j \mid i,j \in \{0,\dots,3\},\ i+j \leq 3\}$ existiert mit

$$\forall_{k \in I}\ p(z_k) = f_k.$$

2. Aufgabe (4 PP)

a) Implementieren Sie ein Verfahren zur summierten Newton-Cotes-Quadratur, indem Sie eine Funktion `y = newton_cotes_quad( f_handle, a, b, n, m )` schreiben, der ein Funktionen-Handle `f_handle`, Intervallgrenzen `a`, `b`, der zu verwendende Polynomgrad `n` (mit $n \in \{1,\dots,6\}$) und die Anzahl der Teilintervalle `m` übergeben werden. Geben Sie in `y` das Ergebnis der Quadratur zurück. Testen Sie Ihr Verfahren anhand geeigneter Monome x^i auf dem Intervall $[0,1]$ auf Richtigkeit.

b) Testen Sie Ihre Implementation an der Funktion $f(x) = \frac{2\ln(\frac{x}{2}+1)}{x^2+4}$ auf dem Intervall $[0,2]$ für alle $n \in \{1,\dots,6\}$ und verschiedene Werte von m. Plotten sie zu jedem `n` jeweils die Quadratur-Fehler bezüglich `m`. Sie dürfen dabei ohne Beweis verwenden, dass $\int_0^2 f(x)\,\mathrm{d}x = \frac{\ln 2}{8}\pi$.

3. Aufgabe (8 PP)

a) Implementieren Sie eine Funktion `taylor_coeff = spline_coeff(x,f)`, wobei `x` einen Vektor von Stützstellen und `f` einen Vektor von Funktionswerten und Randwerten bezeichnet. Geben Sie in `taylor_coeff` die Taylor-Koeffizienten der zugehörigen vollständigen kubischen Spline-Interpolierenden auf den durch `x` beschriebenen Intervallen zurück.

b) Schreiben Sie eine Funktion `y = spline_eval(x,taylor_coeff,t)` mit den Variablen `x`, `taylor_coeff` wie oben und `t` als Vektor von Stellen, an denen die Interpolierende ausgewertet wird. Speichern Sie die Ergebnisse in `y`.

c) Testen Sie Ihre Implementation an den Funktionen

$$f_1 \colon [0, 2\pi] \longrightarrow \mathbb{R} \qquad f_2 \colon [-5, 5] \longrightarrow \mathbb{R}$$
$$x \longmapsto \sin(x) \qquad x \longmapsto \frac{1}{1+x^2}$$

mit 10 äquidistanten Stützstellen und plotten Sie ihre Ergebnisse.

d) Plotten Sie anschließend für alle $n \in \{5, 10, 20, 30, 40, 50\}$ (als Anzahl der Stützstellen) die approximierten Fehler $|||f_i - I_n f_i|||_{[a_i,b_i]}$ in einer geeigneten logarithmischen Skalierung gegen n. Dabei ist die Approximation $|||\cdot|||_{[a,b]}$ der $\infty-$Norm auf $C([a, b])$ gegeben durch

$$|||g|||_{[a,b]} = \max_{x \in \widetilde{[a,b]}} |g(x)|\,, \qquad \widetilde{[a, b]} = \{a + k\tfrac{b-a}{1000} \mid k = 0, \ldots, 1000\}.$$

Was beobachten Sie? Wie verhalten sich die Ergebnisse im Vergleich zu anderen Verfahren zur Interpolation?