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

7. Übung zur Vorlesung

Numerik I

SoSe 2015

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

Abgabe: Fr., 12.06.2015, 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)

a) Bestimmen Sie das Hermite-Interpolationspolynom $p \in P_4$, welches die Bedingungen

$$p(0) = 0, \quad p(1) = 1, \quad p'(1) = 4, \quad p''(1) = 12, \quad p'''(1) = 24$$

erfüllt.

b) Sei $f = cos$. Bestimmen Sie das Hermite-Interpolationspolynom $p \in P_5$ mit

$$\begin{aligned} p(-\pi) &= f(-\pi), & p'(-\pi) &= f'(-\pi), & p''(-\pi) &= f''(-\pi) \\ p(\pi) &= f(\pi), & p'(\pi) &= f'(\pi), & p''(\pi) &= f''(\pi). \end{aligned}$$

Skizzieren Sie anschließend die Graphen von f und p.

2. Aufgabe (4 TP)

Zeigen Sie für nicht notwendigerweise paarweise verschiedene Knoten $x_0, \dots x_n$, dass die dividierte Differenz $f[x_0, \dots, x_n]$ unabhängig von der Reihenfolge der Knoten ist, d.h.

$$f[x_0, \dots, x_n] = f[x_{i_0}, \dots, x_{i_n}]$$

gilt für jede Permutation $i_0, \dots, i_n$ der Indizes $(0, \dots, n)$.

3. Aufgabe (8 PP)

a) Programmieren Sie in `MATLAB` eine Funktion `newton_coeff = ddiff(f_info, x)`, die die Koeffizienten `newton_coeff` eines Hermite-Interpolations-Problems bezüglich der Newton-Basis berechnet. Dabei bezeichne `x` einen Vektor nicht notwendigerweise paarweise verschiedener Stützstellen und `f_info` ein Cell-Array, das Funktionen-Handles auf die zu interpolierende Funktion f und alle gegebenenfalls benötigten Ableitungen von f enthält.

Schreiben Sie anschließend auch eine Funktion `horner(newton_coeff, x, t)`, die das Interpolationspolynom an einer beliebigen Stelle `t` (oder einem Vektor von Stellen `t`) auswertet.

b) Erweitern Sie nun Ihre Funktion `ddiff` um die Möglichkeit, effizient neue Knoten hinzufügen zu können, ohne bereits berechnete dividierte Differenzen erneut auswerten zu müssen.

Genauer, schreiben Sie Ihre Funktion um zu

```
[newton_coeff, extension_info] =
ddiff( f_info, x, newton_coeff_old, extension_info_old ),
```

wobei `newton_coeff_old`, `extension_info_old` optionale Parameter seien. Implementieren Sie die Funktion so, dass Sie nach einem Aufruf der Form

```
[newton_coeff_old, extension_info_old] = ddiff( f_info, x_old )
```

die Koeffizienten für eine Erweiterung der Stützstellen entsprechend dem Schema `x = [x_old, x_neu]` durch

```
newton_coeff = ddiff( f_info, x, newton_coeff_old, extension_info_old )
```

mit einem Aufwand von $\mathcal{O}(\text{size}(\texttt{x\_old}) \cdot \text{size}(\texttt{x\_neu}))$ berechnen können.

c) Testen Sie Ihr Programm nun an den Funktionen

$$f_1(x) = e^{-x}, \quad f_2(x) = \arctan(x)$$

und berechnen Sie die Koeffizienten für die jeweiligen Interpolationsprobleme mit Stützstellen $x^{(1)} = (-5, 0, 0, 0, 5)$. Berechnen Sie anschließend auch die Koeffizienten zu den Stützstellen $x^{(2)} = (-5, 0, 0, 0, 5, -3, -3, 3, 3)$ unter Verwendung der bereits aus der Rechnung zu $x^{(1)}$ bekannten Daten.

Plotten Sie Ihre Ergebnisse und vergleichen Sie diese mit denen bei Wahl von äquidistanten Gittern mit 5 bzw. 9 einfachen Stützstellen.