Universität Heidelberg
Interdisziplinäres Zentrum für Wissenschaftliches Rechnen

Dr. Andreas Potschka

Übungsblatt 3

Einführung in die Numerik, Sommersemester 2017

1. **Matrixpolynomauswertung** (5 Punkte)
   Sei $A \in \mathbb{R}^{n\times n}$ beliebig gegeben. Man gebe einen Algorithmus an zur Auswertung des Matrixpolynoms
   $$p(A) = \sum_{i=0}^{m} a_i A^i$$
   mit Koeffizienten $a_i \in \mathbb{R}$, der möglichst wenig Speicherplatz und arithmetische Operationen (1 a.Op. = 1 Mult. + 1 Add.) benötigt.

2. **Fehlerdarstellung für Nullstellen von Polynomen** (2 Punkte)
   Es seien die Nullstellen eines Polynoms $p(x) = \sum_{i=0}^{m} a_i x^i$ zu bestimmen. Man zeige, dass für eine Näherung $\tilde{z}$ zu einer einfachen Nullstelle $z \neq 0$ in erster Näherung die folgende Abschätzung gilt:
   $$\left|\frac{\tilde{z}-z}{z}\right| \dot{\leq} \left|\frac{p(\tilde{z})}{p'(z)z}\right|.$$
   Dies motiviert die Genauigkeitskontrolle bei der Berechnung von Nullstellen von Polynomen in der Programmieraufgabe unten.
   Hinweis: Die Aufgabe ist leichter als sie aussieht (Taylor-Entwicklung).

3. **Diskrete „Approximation“ von Ableitungen** (4 Punkte)
   Die Funktion $f(x) = x+1$ stelle eine physikalische Größe dar, von der Werte $\tilde{f}(x_i) \approx f(x_i)$ an äquidistant verteilten Punkten
   $$x_i = ih, \quad 0 \leq i \leq n := 10^3, \quad h = 10^{-3},$$
   mit einem maximalen relativen Fehler von $0,1\%$ gemessen werden. Man zeige, dass bei der Approximation der Ableitungswerte $f'(x_i)$ mit dem zentralen Differenzenquotienten
   $$f'(x_i) \approx \frac{\tilde{f}(x_{i+1}) - \tilde{f}(x_{i-1})}{2h}, \quad i = 1,\ldots,n-1$$
   aus diesen Werten ein relativer Fehler von 100% auftreten kann. Dies zeigt die Fragwürdigkeit der Approximation von Ableitungen durch Differenzenquotienten.
   Hinweis: Man konstruiere spezielle Störungen.

4. **Eigenschaften der Lagrangeschen Polynombasis** (5 Punkte)
   Gegeben seien $n+1$ paarweise verschiedene Punkte $x_i \in \mathbb{R}^1, i = 0,1,\ldots,n,$ und die zugehörigen $n+1$ sog. Lagrangeschen Polynome
   $$L_i^{(n)}(x) = \prod_{j=0, j\neq i}^{n} \frac{x-x_j}{x_i-x_j}, \quad i = 0,\ldots,n.$$
   Man zeige, dass die Polynome $\{L_i^{(n)}, i = 0,\ldots,n\}$, eine Basis des Polynomraums P_n (Vektorraum aller Polynome vom Grad kleiner oder gleich n) bilden, und dass die folgenden

Beziehungen gelten:

$$i) \quad \sum_{i=0}^{n} L_i^{(n)}(x) = 1, \quad x \in \mathbb{R}^1, \qquad\qquad ii) \quad \sum_{i=0}^{n} x_i^k L_i^{(n)}(0) = 0, \quad k = 1, \dots, n,$$

$$iii) \quad \sum_{i=0}^{n} x_i^{n+1} L_i^{(n)}(0) = (-1)^n \prod_{i=0}^{n} x_i.$$

Hinweis: Man verwende die Eindeutigkeit des Lagrangeschen Interpolationspolynoms. Bei *iii)* verwende man (im Vorgriff auf die Vorlesung kommenden Dienstag) die Darstellung des Fehlers

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

für den Lagrangeschen Interpolanten p zu den Stützpunkten $(x_i, f(x_i))$ einer $n+1$ mal stetig differenzierbaren Funktion für einen Zwischenpunkt ξ_x aus der konvexen Hülle der Punkte $x_0, \dots, x_n$ und x.

PA. **Lösung quadratischer Gleichungen** (10 Punkte)

Man erstelle eine Python-Funktion zur Berechnung aller reellen Lösungen der quadratischen Gleichung

$$p(x) = ax^2 + bx + c = 0,$$

zu gegebenen $a, b, c \in \mathbb{R}$. Es sollen alle möglichen Fälle der Degenerierung (z. B.: $a = 0$) berücksichtigt und der Einfluß des Rundungsfehlers minimiert werden. Die Lösungen sollen als Liste zurückgegeben werden, die entweder leer sein kann (`[]`), eine (`[y]`), zwei (`[y1, y2]`) oder, falls unendlich viele Lösungen existieren, drei Lösungen (`[-1.0, 0.0, 1.0]`) enthält. Man erprobe das Programm anhand der folgenden Fälle:

a :	0	0	0	0	2	2	2	4	1	1	1	-1	-1	-4	-1	-4	-1	-1	$2,5 \cdot 10^9$
b :	0	0	1	2	0	0	0	2	2	2	2	0	0	0	2	8	2	2	-10^5
c :	0	1	0	1	0	1	-4	0	-3	1	2	0	-1	2	0	12	-1	-5	1

Die berechneten Lösungen sollen akzeptiert werden, wenn das heuristische Kriterium

$$|p(\tilde{z})| \leq \begin{cases} \mathsf{eps}\, |p'(\tilde{z})\tilde{z}|, & \text{wenn } \tilde{z} \text{ eine einfache Nullstelle ist,} \\ \mathsf{eps}, & \text{sonst,} \end{cases}$$

erfüllt ist (siehe Aufgabe 2).

Hinweise: Folgende Python-Konstrukte könnten hilfreich sein: `np.finfo(float).eps` aus NumPy (ansonsten müssen zur Lösung dieser Aufgabe keine Funktionen aus NumPy benutzt werden); überprüfen ob eine Liste `a_list` leer ist mittels `if not a_list:`; die Funktionen `enumerate` und `zip`. Eine mögliche Referenz: `http://www.saltycrane.com/blog/2008/04/how-to-use-pythons-enumerate-and-zip-to/` (das geht auch mit mehr als zwei Listen)

Abgabe bis Donnerstag, 11.05.2017, 14:15 Uhr.

Webseite:
`http://typo.iwr.uni-heidelberg.de/groups/mobocon/teaching/numerik-0-ss17`