Familienname:

Vorname:

Matrikelnummer:

Studienkennzahl:

Aufgabe 1 (2 Punkte):
Aufgabe 2 (10 Punkte):
Aufgabe 3 (12 Punkte):
Aufgabe 4 (6 Punkte):

Gesamtpunktzahl:

Note:

Schriftlicher Test 2 (90 Minuten)
VU Einführung ins Programmieren für TM

19. Jänner 2007

Zu gegebenen reellen Stützstellen $x_1 < \cdots < x_n$ und Funktionswerten $y_1, \ldots, y_n \in \mathbb{R}$ garantiert die Lineare Algebra die Existenz eines eindeutigen Interpolationspolynoms $p = \sum_{j=0}^{n-1} a_j x^j$ vom Grad $n-1$ mit

$$p(x_j) = y_j \quad \text{für alle } j = 1, \ldots, n.$$

Für ein gegebenes $t \in \mathbb{R}$ kann dieses Polynom mit Hilfe des **Neville-Verfahrens** ausgewertet werden. Dieses Verfahren basiert nur auf den gegebenen Werten x_j und y_j, d.h., die Koeffizienten a_j werden nicht explizit berechnet: Beim Neville-Verfahren definiert man für $j, m \in \mathbb{N}$ mit $m \geq 2$ und $j + m \leq n + 1$ induktiv die Werte

$$p_{j,1} := y_j \quad \text{und} \quad p_{j,m} := \frac{(t - x_j)p_{j+1,m-1} - (t - x_{j+m-1})p_{j,m-1}}{x_{j+m-1} - x_j}.$$

Es gilt dann $p(t) = p_{1,n}$, wobei der mathematische Beweis für diesen Algorithmus in der Vorlesung zur Numerischen Mathematik folgt. Im Folgenden sollen Sie zunächst eine Matlab-Funktion

$$\texttt{pt} = \texttt{neville}(\texttt{x}, \texttt{y}, \texttt{t})$$

schreiben, die zu gegebenen Vektoren $x, y \in \mathbb{R}^n$ und einem Auswertungspunkt $t \in \mathbb{R}$ den Funktionswert $p(t)$ berechnen. Als Anwendung davon soll die **Richardson-Extrapolation des einseitigen Differenzenquotienten** programmiert werden (siehe unten).

Hinweis. Neben den Formeln (für $j, m \in \mathbb{N}$ mit $m \geq 2$ und $j + m \leq n + 1$)

$$p_{j,1} := y_j \quad \text{und} \quad p_{j,m} := \frac{(t - x_j)p_{j+1,m-1} - (t - x_{j+m-1})p_{j,m-1}}{x_{j+m-1} - x_j}$$

und $p(t) = p_{1,n}$ berücksichtige man vor allem das folgende schematische Vorgehen:

$$\begin{array}{ccccccccccc}
y_1 & = & p_{1,1} & \longrightarrow & p_{1,2} & \longrightarrow & p_{1,3} & \longrightarrow & \cdots & \longrightarrow & p_{1,n} \\
 & & & \nearrow & & \nearrow & & & & \nearrow & \\
y_2 & = & p_{2,1} & \longrightarrow & p_{2,2} & & & & & & \\
 & & & \nearrow & & & & \nearrow & & & \\
y_3 & = & p_{3,1} & \longrightarrow & \vdots & & & & & & \\
\vdots & & \vdots & & \vdots & \nearrow & & & & & \\
y_{n-1} & = & p_{n-1,1} & \longrightarrow & p_{n-1,2} & & & & & & \\
 & & & \nearrow & & & & & & & \\
y_n & = & p_{n,1} & & & & & & & &
\end{array}$$

Aufgabe 1 (2 Punkte). Man kann das Neville-Verfahren auch mit einer rekursiven Funktion programmieren. Warum ist eine Realisierung über geeignete Schleifen effizienter und somit sinnvoller?

Aufgabe 2 (10 Punkte). Man schreibe eine MATLAB-Funktion `pt = neville(x,y,t)`, wobei, um Fehler zu vermeiden, eine Hilfsmatrix $P = (p_{j,m})_{j,m=1}^n \in \mathbb{R}^{n \times n}$ verwendet werden soll (vgl. Aufgabe 4).

Richardson-Extrapolation. Um die Ableitung einer Funktion $f(x)$ im Punkt x zu approximieren, kann man für $h > 0$ den einseitigen Differenzenquotienten

$$\Phi(h) = \frac{f(x+h) - f(x)}{h}$$

verwenden. Es gilt dann im Limes $\Phi(0) = f'(x)$. Für eine fest gewählte Schrittweite $h_0 > 0$ betrachten wir die Folgen $x_n = 2^{-(n-1)}h_0$ und $y_n = \Phi(x_n)$. Für jedes $n \in \mathbb{N}$ kann man zu $x_1, \ldots, x_n$ und $y_1, \ldots, y_n$ das zugehörige Interpolationspolynom $p_n(t)$ vom Grad $n-1$ mit dem Neville-Verfahren bei $t = 0$ auswerten. Man erhält so eine Folge von Approximationen

$$\phi_n := p_n(0) \approx \Phi(0) = f'(x).$$

Aufgabe 3 (12 Punkte). Man schreibe eine MATLAB-Funktion

$$\texttt{phi} = \texttt{richardson}(\texttt{f}, \texttt{x}, \texttt{h0}, \texttt{eps})$$

die eine Approximation $\texttt{phi} = \phi_n \approx f'(x)$ zurückliefert, die mittels Richardson-Extrapolation gewonnen wurde. Dabei soll $n \in \mathbb{N}$ minimal sein mit der Eigenschaft

$$|\phi_n - \phi_{n-1}| \leq \texttt{eps} \cdot \max\{|\phi_{n-1}|, |\phi_n|\}.$$

Verwenden Sie die Funktion aus Aufgabe 2 für das Neville-Verfahren.

Hinweis. Neben den Formeln (für $j, m \in \mathbb{N}$ mit $m \geq 2$ und $j + m \leq n + 1$)

$$p_{j,1} := y_j \quad \text{und} \quad p_{j,m} := \frac{(t - x_j)p_{j+1,m-1} - (t - x_{j+m-1})p_{j,m-1}}{x_{j+m-1} - x_j}$$

und $p(t) = p_{1,n}$ berücksichtige man vor allem das folgende schematische Vorgehen:

$$\begin{array}{ccccccccccc}
y_1 & = & p_{1,1} & \longrightarrow & p_{1,2} & \longrightarrow & p_{1,3} & \longrightarrow & \cdots & \longrightarrow & p_{1,n} \\
 & & & \nearrow & & \nearrow & & & & \nearrow & \\
y_2 & = & p_{2,1} & \longrightarrow & p_{2,2} & & & & & & \\
 & & & \nearrow & & & & \nearrow & & & \\
y_3 & = & p_{3,1} & \longrightarrow & \vdots & & & & & & \\
\vdots & & \vdots & & \vdots & \nearrow & & & & & \\
y_{n-1} & = & p_{n-1,1} & \longrightarrow & p_{n-1,2} & & & & & & \\
 & & & \nearrow & & & & & & & \\
y_n & = & p_{n,1} & & & & & & & &
\end{array}$$

Aufgabe 4 (6 Punkte). Man kann das Neville-Verfahren so programmieren, dass zur Speicherung *keine* Matrix $P = (p_{j,m})_{j,m=1}^n$ aufgebaut wird, sondern die gegebenen y_j-Werte geeignet überschrieben werden. Es wird also kein weiterer Zusatzspeicher benötigt. Man schreibe eine MATLAB-Funktion `pt = neville(x,y,t)`, die einen Algorithmus realisiert, der ohne zusätzlichen Speicherbedarf auskommt.