Einführung in die Numerik
6. Programmierübung

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Mathematik **WS 2015/16**
Prof. Dr. Herbert Egger, Dr. Mirjam Walloth, Thomas Kugler **13.1.2015**

Programmierübung

Aufgabe P1
evalPP.m: Wir betrachten stückweise Polynome $s \in \Pi_k(T^n)$, wobei $T^n = \{x_0 < \ldots < x_n\}$ ein Gitter des Intervals $[x_0, x_n]$ sei. Das stückweise Polynom s ist gegeben durch

$$s_i(x) = a_{i,0} + a_{i,1}(x - x_{i-1}) + \ldots a_{i,k}(x - x_{i-1})^k, \qquad i = 1, \ldots, n.$$

Das stückweise Polynom ist also durch die Stützstellen und Koeffizienten

$$s.x = (x_0, \ldots, x_n); \qquad s.a = \begin{pmatrix} a_{1k} & a_{1,k-1} & \cdots & a_{1,0} \\ \vdots & \vdots & & \vdots \\ a_{n,k} & a_{n,k-1} & \cdots & a_{n,0} \end{pmatrix}$$

eindeutig festgelegt. Schreiben Sie eine Funktion

```
function y=evalPP(s,x),
```

welche das stückweise Polynom an den Stellen des Vektors x auswertet.
Hinweis: Führen Sie die Kommandos

```
s.a=1
s.b=2
s.c=3
```

aus und betrachten Sie die Resultate. Sie haben soeben den Datentyp `struct` verwendet. Diesen werden Sie auf diesem Blatt mehrfach benötigen. Hilfe wie immer mit `help struct`.
differentiatePP.m Schreiben Sie eine Funktion

```
function sd=differentiatePP(s)
```

welche ein stückweises Polynom mit Stützstellen `s.x` und Koeffizienten `s.a` elementweise differenziert. Dabei soll `sd.x` wieder die Stützstellen und `sd.a` die Koeffizienten des Resultats beinhalten.
testp1.m:

(a) Bestimmen Sie die stückweisen Polynome zu den Eingaben
  i. `s0.x=[0:2]; s0.a=[0; 1];`
  ii. `s1.x=[0:2]; s1.a=[0,1; 1,-1];`

(b) Werten Sie diese stückweisen Polynome an den Stellen `x=[0:0.01:2]` aus (Horner Schema) und überprüfen Sie hiermit die Korrektheit der Auswerteroutine.

(c) Benutzen Sie zur Kontrolle auch die folgenden MATLAB Routinen:

```
pp=mkpp(s.x,s.a);
y=ppval(pp,x);
```

Hilfe zu den Befehlen erhalten Sie z.B. mit `help mkpp` oder `doc mkpp`. Weitere Befehle kommen noch weiter unten.

(d) Berechnen Sie `differentiatePP(s1)` und vergleichen Sie die Ergebnisse mit der tatsächlichen Ableitung.

Aufgabe P2

piecewiseConstantApproximation.m: Schreiben Sie eine Routine

```
function s0=piecewiseConstantApproximation(xi,f)
```

welche die stückweise konstante Approximation zur Funktion f am Gitter mit den Stützstellen x_i berechnet. Dabei soll `s0.x` die Stützstellen und `s0.a` die Koeffizienten des stückweisen Polynoms beinhalten. Zur Berechnung der Integrale $\int_{x_{i-1}}^{x_i} f(x)dx$ können Sie die MATLAB Funktion

```
y=integral(f,a,b)
```

mit $a = x_{i-1}$ und $b = x_i$ verwenden. Hilfe dazu erhalten Sie mit `help integral` oder `doc integral`.

testp2.m:

(a) Bestimmen Sie die stückweise konstanten Bestapproximationen für $f(x) = \sin(\pi x)$ auf äquidistanten Gittern T^n von $[0,1]$ mit $n = 2^m$, $m = 0,\ldots,5$.

(b) Werten Sie die Funktionen f und s_0 an einem feinen Gitter $x_l = lh'$ mit $h' = 1/N$ und $N = 1000$ aus und plotten Sie die Ergebnisse.

(c) Bestimmen Sie jeweils die Gitterweite h sowie Approximationen für die Fehler $\|f - s_0\|_{L^2(0,1)}$ und $\|f - s_0\|_{L^\infty(0,1)}$ mittels

$$e_2 = \Big(\frac{1}{N}\sum_{l=1}^{N} |f(z_l) - s_0(z_l)|^2\Big)^{1/2}_{L^2(0,1)} \qquad \text{und} \qquad e_\infty = \max_{l=1,\ldots,N} |f(z_l) - s_0(z_l)|.$$

(d) Stellen Sie die Schrittweite h und die oben berechneten Fehler in einer Tabelle dar. Bestimmen Sie auch die estimated order of convergence und vergleichen Sie mit den theoretischen Aussagen.

Aufgabe P3

piecewiseLinearInterpolation.m: Schreiben Sie eine Routine

```
function s1=piecewiseLinearInterpolation(xi,f)
```

welche die stückweise lineare Spline-Interpolierende zur Funktion f am Gitter mit den Stützstellen x_i berechnet. Dabei soll `s1.x` die Stützstellen und `s1.a` die Koeffizienten des stückweisen Polynoms beinhalten.

testp3.m:

(a) Bestimmen Sie die stückweise lineare Interpolierende zu $f(x) = \sin(\pi x)$ auf äquidistanten Gittern T^n von $[0,1]$ mit $n = 2^m$, $m = 0,\ldots,5$.

(b) Die MATLAB Funktion

```
pp=interp1(xi,f(xi),'linear','pp');
[x,a]=unmkpp(pp);
```

erzeugt dieselben Stützstellen und Koeffizienten. Überprüfen Sie damit Ihr Ergebnis.

(c) Werten Sie die Funktionen f und s_1 auf einem feiner Gitter $x_l = lh'$ mit $h' = 1/N$ und $N = 1000$ aus und plotten Sie die Ergebnisse.

(d) Bestimmen Sie jeweils die Gitterweite h sowie Approximationen für die Fehler $\|f - s_1\|_{L^2(0,1)}$ und $\|f - s_1\|_{L^\infty(0,1)}$ mittels

$$e_2 = \Big(\frac{1}{N}\sum_{l=1}^{N} |f(z_l) - s_0(z_l)|^2\Big)^{1/2}_{L^2(0,1)} \qquad \text{und} \qquad e_\infty = \max_{l=1,\ldots,N} |f(z_l) - s_0(z_l)|.$$

(e) Berechnen Sie die Ableitung $f'(x)$ sowie die Ableitung $s_1'(x)$ und bestimmen Sie wie davor Näherungswerte $\|f' - s_1'\|_{L^2(0,1)}$ und $\|f' - s_1'\|_{L^\infty(0,1)}$. Stellen Sie die Ergebnisse wieder in einer Tabelle dar, bestimmen Sie die estimated order of convergence, und vergleichen Sie mit den theoretischen Resultaten.

Aufgabe P4

piecewiseCubicInterpolation.m: Schreiben Sie eine Routine

```
function s3=piecewiseCubicInterpolation(xi,f)
```

welche die kubische Spline-Interpolierende s zur Funktion f und Stützstellen x_i berechnet. Als Zusatzbedingungen verwenden wir die *not-a-knot* Bedingungen

$$s_1'''(x_1) = s_2'''(x_1) \qquad \text{und} \qquad s_{n-1}'''(x_{n-1}) = s_n'''(x_{n-1}).$$

Das Gleichungssystem für die Momente $m_i = s''(x_i)$ lautet dann

$$\begin{pmatrix} h_2 & h_1 - h_2 & -h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-1} & 2(h_{n-1}+h_n) & h_n \\ & & -h_n & h_n - h_{n-1} & h_{n-1} \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ \vdots \\ m_{n-1} \\ m_n \end{pmatrix} = \begin{pmatrix} 0 \\ g_1 \\ \vdots \\ g_{n-1} \\ 0 \end{pmatrix}$$

mit $g_i = \frac{6}{h_{i+1}}(y_{i+1} - y_i) - \frac{6}{h_i}(y_i - y_{i-1})$. Das System stimmt also bis auf die erste und letzte Zeile mit dem für den natürlichen interpolierenden Spline überein. Die Koeffizienten a_i, b_i, c_i und d_i der stückweisen Polynomdarstellung können dann so wie im Skriptum berechnet werden.

testp4.m:

(a) Bestimmen Sie die stückweise lineare Interpolierende zu $f(x) = \sin(\pi x)$ auf äquidistanten Gittern T^n von $[0,1]$ mit $n = 2^m$, $m = 1, \ldots, 5$.

(b) Die MATLAB Routine

```
pp=interp1(xi,f(xi),'cubic','pp');
[x,a]=unmkpp(pp);
```

sollte dasselbe Ergebnis liefern.

(c) Werten Sie die Funktionen f und s_1 auf einem feinen Gitter $x_l = lh'$ mit $h' = 1/N$ und $N = 10000$ aus und plotten Sie die Ergebnisse.

(d) Bestimmen Sie jeweils die Gitterweite h sowie Approximationen für die Fehler $\|f - s_1\|_{L^2(0,1)}$ und $\|f - s_1\|_{L^\infty(0,1)}$ mittels

$$e_2 = \Big(\frac{1}{N}\sum_{l=1}^{N} |f(z_l) - s_0(z_l)|^2\Big)^{1/2}_{L^2(0,1)} \qquad \text{und} \qquad e_\infty = \max_{l=1,\ldots,N} |f(z_l) - s_0(z_l)|.$$

(e) Berechnen Sie die Ableitung $f^{(m)}(x)$ sowie die Ableitung $s_3^{(m)}(x)$ für $m = 1, 2, 3$ und plotten Sie diese jeweils auf dem Gitter x_l.

(f) Bestimmen Sie wie zuvor Näherungswerte für $\|f^{(m)} - s_3^{(m)}\|_{L^2(0,1)}$ und $\|f^{(m)} - s_3^{(m)}\|_{L^\infty(0,1)}$. Stellen Sie die Ergebnisse wieder in Tabellen dar, bestimmen Sie die estimated order of convergence, und vergleichen Sie mit den theoretischen Resultaten.

Organisatorische Hinweise:

a) Legen Sie ein Verzeichnis `pp6` an und speichern Sie alle Funktionen und Skripte für diese Übung darin ab. Der Inhalt des Verzeichnisses sollte hier also sein:

```
testp1.m evalPP.m differentiatePP.m
testp2.m piecewiseConstantApproximation.m
testp3.m piecewiseLinearInterpolation.m
testp4.m piecewiseCubicInterpolation.m
```

b) Zippen Sie das Verzeichnis; in Linux: `zip -r pp6.zip pp6` von außerhalb des Verzeichnisses. In Windows kann WinZip oder ein ähnliches Tool verwendet werden.

c) Überprüfen Sie, dass alle Skripte und Funktionen lauffähig sind!

d) Kommentieren Sie die Funktionen geeignet. Insbesondere sollten Eingabe und Ausgabe Parameter beschrieben werden. `help <funktionsname>` sollte Information über die Routinen liefern.

e) Antworten auf die Fragen sollten in den `testp*.m` files als Kommentare hinterlegt werden.