Tipps zu Serie 6

Aufgabe 1

Hier geht es um den in der Stunde bereits erwähnte Aitken-Neville Algorithmus zur Auswertung eines Lagrange'schen Interpolationspolynoms.

a) Schaut auch dazu erst einmal die Notitzen aus der Übung, beziehungsweise die Referenz im Skript an.

b) Hier sollt ihr eine Matlab Funktion schreiben, welche als Input zu interpolierende Datenpunkte (t_i, y_i), $i = 0, 1, \ldots, n$ und Auswertungsstellen $(x_1, \ldots, x_m)$ nimmt. Der einzige Unterschied zum normalen Aitken-Neville Algorithmus, der den Funktionswert des Interpolationspolynoms an den Auswertungsstellen $(x_1, \ldots, x_m)$ zurückgibt, ist nun, dass ihr die Ableitung des Interpolytionspolynoms an den entsprechenden Auswertungsstellen $(x_1, \ldots, x_m)$ zurückgeben sollt. Berechnet dafür einfach die Ableitungen der im Skript definierten Rekursionsformel. Für den Rekursionsbeginn $p_{i,i}(t)$ sähe dies zum Beispiel so aus:

$$p_{i,i}(t) = y_i$$
$$p'_{i,i}(t) = \frac{\mathrm{d}}{\mathrm{d}t} p_{i,i}(t) = 0$$

Berechnet nun also für beliebige $0 \leq k \leq l \leq n$ die Ableitung von

$$p_{k,l}(t) = \frac{(t - t_k)p_{k+1,l}(t) - (t - t_l)p_{k,l-1}(t)}{t_l - t_k}$$
$$p'_{k,l}(t) = \frac{\mathrm{d}}{\mathrm{d}t} p_{k,l}(t) = \ldots$$

Benutzt dafür die Produktregel der Differentialrechnung. Ihr werden natürlich wieder eine Rekursionsformel erhalten, welche dann analog zum normalen Algorithmus in Matlab implementiert werden kann.

c) Hier ein kleines Beispiel, wie die Matlab Funktion `polyfit()` funktioniert:

```
% Beispiel fuer polyfit()

% Interpolyationspunkte (x_0,y_0),...,(x_3,y_3)
% Dementsprechend benoetigen wir ein Interpolationspolynom
% von Grad 3
% x-Werte
x = [-1,0,3,5];
% y-Werte
y = [4,-1,2,3];

%Berechnung der Koeffizienten der Monombasis
% a = polyfit(x-Werte, y-Werte, Grad des Interpolationspolynoms)
a = polyfit(x,y,3);
```

```
14
15 % Auswertung des Interpolationspolynoms auf dem Intervall [-2,6]
16 % Wir verwenden 1000 aequidistante Auswertungpunkte xx
17 xx = linspace(-2,6,1000);
18 % Zugehoerige y_werte des Polynoms mit Koeffizienten a an xx
19 yy = polyval(a,xx);
20
21 % Plot
22 hold on;
23 plot(x,y,'go')
24 plot(xx,yy)
25 legend('Interpolationspunkte','Interpolationspolynom')
```

Berechnet analog zum Beispiel die Monomkoeffizienten `a`. Überlegt euch dann, wie sich die Koeffizienten bei der Ableitung des gesamten Polynoms verändern, und wertet schließlich das Polynom mit den neuen Koeffizienten analog zum Beispiel an den entsprechenden Stellen aus.

Aufgabe 3

Hier geht es um die Auswertung eines Interpolations mittels des Horner-Schemas. In der Übungsstunde war meine Formulierung leider nicht ganz konsistent mit dem Skript: Ich hatte dort das Horner-Schema als Algorithmus zur Auswertung eines Interpolationspolynoms in der Newtonbasis vorgestellt. Im Skript wird dies jedoch als *modifiziertes Horner-Schema* definiert.

Das **Horner-Schema**, welches auch in dieser Aufgabe benötigt wird, ist im Skript in [Rem. 3.2.5] definiert und dient zur Auswertung eines Polynoms in der Monombasis mit Koeffizienten $c_0, \dots, c_{m-1}$. Es entspricht somit der Matlab-Funktion `polyval()` aus Aufgabe 1.

a) Genau diese sollt ihr nun *effizient* in C++ implementieren, jedoch soll zusätzlich auch noch die Ableitung des Polynoms an der Auswertungsstelle x zurückgegeben werden. Orientiert euch dabei an der *rekursiven* Formulierung aus Matlab Code 3.2.7 im Skript und Aufgabe 1.

b) Nun sollt ihr wieder eine naive Version des voherigen Algorithmus implementieren. Sprich, ihr rechnet den Wert des Polynoms an der Auswertungsstelle x einfach als Summe entsprechend der Definition der Monombasis aus:

$$p(x) = c_o x^{m-1} + c_1 x^{m-2} + \cdots + c_{m-2} x^1 + c_{m-1} x^0$$

Dabei soll äquivalent zu Matlab c_0 der Koeffizient des Führenden Monoms (Monom mit dem höchsten Exponenten) sein.

c) Berechnet nun die theoretischen Komplexitäten beider Verfahren. Beachtet dabei die Anzahl Additionen, Multiplikationen und Funktionen, die ihr zur Auswertung des Polynoms verwendet.

d) Testet nun die Laufzeiten eurer beiden Funktionen und vergleicht das Ergebnis mit dem vorher berechneten theoretischen Wert.