Serie 1

Besprechung: Woche von Montag, 11.3.2010

1.1. Gegeben seien die Werte $(0,0)$, $(1,2)$, $(4,8)$.

a) Geben Sie das Lagrangeinterpolationspolynom an.

b) Geben Sie das Neville-Schema an für die Auswertung des Interpolationspolynoms an der Stelle $x = 2$ an.

c) Für das quadratische Interpolationspolynom $p(x) = a_0 + a_1 x + a_2 x^2$ kann man auch ein lineares Gleichungssystem für die Koeffizienten a_0, a_1, a_2 aufstellen. Geben Sie dieses an und lösen Sie für die a_i

d) Prüfen Sie Ihre Approximation $p(2)$ mittels der `Matlab` oder `numpy`-Funktionen `polyfit`$(\mathbf{x}, \mathbf{f}, n)$ und `polyval`. `polyfit` gibt Ihnen auch die Koeffizienten a_i aus Teilaufgabe c). Vergleichen Sie.

Bemerkung: Eine Verwendung von `polyfit` und `polyval` im Kontext von Polynominterpolation ist eher als eine "quick and dirty"-Methode zu verstehen, denn `polyfit` basiert auf Polynomapproximation in der Monombasis $\{1, x, x^2, \ldots, x^n\}$ und ist deshalb für größere n nicht zu empfehlen.

1.2. Programmieren Sie in `matlab` oder `python` einen Algorithmus, der das Neville-Schema realisiert. Input sollen die Vektoren $\mathbf{x}$, $\mathbf{y}$ der Länge $n+1$. Output soll ein Array (Grösse $(n+1) \times (n+1)$) sein, welches die Spalten des Neville-Schemas enthält.

1.3. Verwenden Sie Ihr Programm aus Aufgabe 1.2, um den Funktionswert $f(0)$ der Funktion

$$f(h) = \frac{\exp(h) - 1}{h}$$

zu bestimmen und den Fehler abzuschätzen. Tatsächlich ist $f(0) = 1$.

Gehen Sie wie folgt vor: 1.) Verwenden Sie die Stützstellen $h_i = 2^{-i}$, $i = 0, \ldots, I$, mit $I = 10$. Die zugehörigen Stützwerte sind $f_i = f(h_i)$). Bestimmen Sie das Neville-Array (Größe $(I+1) \times (I+1)$) mittels Aufgabe 1.2.

2). In jeder festen Spalte $N(:, m)$ des Neville-Arrays sollten die Werte gegen $f(0) = 1$ konvergieren. Um ein Gefühl für die Genauigkeit zu erhalten, plotten Sie für die ersten 3 Spalten des Neville-Arrays N den Betrag des tatsächlichen Fehlers (also: $|N(:, m) - 1|$) gegen die verwendeten Werte $h(:)$ in einem *"doppelt logarithmischen"* Plot. In `matlab`-Notation wäre dies (die Stützstellen $h_0, \ldots, h_I$ sind im array $h(1 : I+1)$ abgelegt):

$$\begin{aligned}\texttt{loglog}(&h(1:I+1), \texttt{abs}(N(1:I+1,1)-1), \ldots\\ &h(1:I+1-1), \texttt{abs}(N(1:I+1-1,2)-1), \ldots\\ &h(1:I+1-2), \texttt{abs}(N(1:I+1-2,3)-1))\end{aligned}$$

Bemerkung: In `python` wird das doppelt logarithmische Plotten mit `matplotlib.pyplot.loglog` realisiert und die Arraygrenzen sind entsprechend zu modifizieren, weil Arrays in `python` bei 0 starten.