2. und 3. Labor
Interpolation

1. Berechnung des Interpolationspolynoms I: Newton-Darstellung.
Man schreibe ein Programm, welches anhand der Newton-Darstellung[1] das Lagrange Interpolationspolynom berechnet.
Wende dann das Programm auf $f : [-1, 1] \to \mathbb{R}$, $f(x) = e^x$, mit $11, 21, 31$ äquidistante Stützstellen (d.h. $n = 10, 20, 30$). Man erzeuge 3 Bilder dazu.
(Hinweis: Dividierte Differenzen müssen im Voraus berechnet werden, s. [Trîmbiţaş].)

2. Runges Phänomen I: was schief gehen kann.
Wir untersuchen Runges Funktion $f : [-1, 1] \to \mathbb{R}$, $f(x) = \dfrac{1}{1 + 25x^2}$.
Man wende das Newton-Programm an, um das entsprechende Interpolationspolynom für $11, 21, 31$ äquidistante Stützstellen (wieder $n = 10, 20, 30$) zu berechnen. Man erzeuge 3 Bilder dazu.

3. Runges Phänomen I: Tschebyscheff[2] Knoten (eine Lösung).
Äquidistante Knoten habe sich als schlecht erwiesen, deshalb wollen wir jetz sogenannte Tschebyscheff Interpolationsknoten[3] verwenden. Diese werden, für $i = 0, \dots n$, durch folgende Beziehungen definiert

$$\begin{aligned} x_i &= \cos \frac{(2i+1)\pi}{2n+1}, \quad &\text{(Tscheb. Knoten 1. Art)} \\ x_i &= \cos \frac{i\pi}{n}, &\text{(Tscheb. Knoten 2. Art)} \end{aligned}$$

Man wende das Newton-Programm an, um das entsprechende Interpolationspolynom für $11, 21, 31$ Tschbyscheff Stützstellen 1. und 2. Art (wieder $n = 10, 20, 30$) zu berechnen. Man erzeuge 6 Bilder dazu.

4. Numerische Instabilität.
Tschbyscheff Stützstellen sind nur teilweise eine Lösung, denn das Newton-Verfahren ist bei $n \gg 1$ numerisch instabil.
Man wende das Newton-Programm an, um das entsprechende Interpolationspolynom für 69 bzw. 71 Tschbyscheff Stützstellen 1. Art zu berechnen. Man erzeuge 2 Bilder dazu.

---

[1] [Trîmbiţaş] ist eine gute Referenz sowohl für Matlab/Ocrave als auch für Python Nutzer.

[2] Cebyshev (Engl.) oder Cebîşev (Rum.).

[3] Tschebyscheff Knoten sind Nullstellen entsprechender Tschebyscheff-Polynome. Z. B. die Tschebyscheff-Polynome erster Art werden durch

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), k \geq 2$$

rekursiv berechnet.

5. Berechnung des Interpolationspolynoms II: Die baryzentrische Darstellung.
Man schreibe ein Programm, welches anhand der baryzentrischen Darstellung[4]

$$L(x) = \begin{cases} \dfrac{\sum_{i=0}^{n} \frac{\omega_i f(x_i)}{x - x_i}}{\sum_{i=0}^{n} \frac{\omega_i f(x_i)}{x - x_i}} & \text{if } x \neq x_i \\ f(x_i) & \text{otherwise} \end{cases}$$

mit $\omega_i = \prod_{k \neq i}^{n} \frac{1}{x_i - x_k}$ das Lagrange Interpolationspolynom berechnet.
Man wende das baryzentrische Programm an, um das entsprechende Interpolationspolynom für 69 bzw. 71 Tschbyscheff Stützstellen 1. Art zu berechnen. Man erzeuge 2 Bilder dazu. Was lässt sich zur numerischen (In)stabilität festestellen?

6. Lineare Splines.
Man schreibe ein Programm, welches eine gegebene Funktion mit stückweise linearen Splines interpoliert. Wende dieses Progamm an um Runges Funktion an $11, 21, 31$ äquidistante Stützstellen (d.h. $n = 10, 20, 30$) zu interpolieren (3 Bilder).
(Hinweis: die Dachfunktion nicht vergessen!)

7. Natürliche kubische Splines.
Man schreibe ein Programm, welches eine gegebene Funktion mit natürliche kubische Splines interpoliert. Wende dieses Progamm an, um Runges Funktion an $11, 21, 31$ äquidistante Stützstellen (d.h. $n = 10, 20, 30$) zu interpolieren (3 Bilder).
(Hinweis: s. [Trîmbiţaş] oder [Beu].)

[4] s. [Trefethen] oder [Helzel].