Interpolation

- Gegeben: n Messpunkte $\{x_1, x_2, \ldots, x_n\}$ und dazugehörige Messwerte $\{y_1, y_2, \ldots, y_n\}$
- Gesucht: Geeignete Interpolationsfunktion $f(x)$
- Naheliegende Möglichkeit: Polynominterpolation
- Einfaches Beispiel: Zwei Messwerte $\Rightarrow$ Beschreibung durch eine lineare Funktion $f(x) = a_0 + a_1 x$ (Polynom 1. Grades)
- Bestimmung von a_0 und a_1 trivial

$$f(x_1) = a_0 + a_1 x_1 \stackrel{!}{=} y_1,$$
$$f(x_2) = a_0 + a_1 x_2 \stackrel{!}{=} y_2$$

- Somit

$$a_1 = \frac{y_2 - y_1}{x_2 - x_1}, \quad a_0 = \frac{x_1 y_2 - x_2 y_1}{x_1 - x_2}$$

- Bessere Schreibweise

$$f(x) = \frac{x - x_2}{x_1 - x_2} y_1 + \frac{x - x_1}{x_2 - x_1} y_2$$

- Analog für 3 Messwerte und ein Interpolationspolynom 2. Grades $f(x) = a_0 + a_1 x + a_2 x^2$

$$f(x_1) = a_0 + a_1 x_1 + a_2 x_1^2 \stackrel{!}{=} y_1,$$
$$f(x_2) = a_0 + a_1 x_2 + a_2 x_2^2 \stackrel{!}{=} y_2,$$
$$f(x_2) = a_0 + a_1 x_3 + a_2 x_3^2 \stackrel{!}{=} y_3$$

- Man erhält

$$f(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} y_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} y_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} y_2$$

- Verallgemeinernd ergibt sich für n Stützstellen die Langrangesche Interpolationsformel

$$f(x) = \sum_{i=1}^{n} f(x_i) L_i(x), \quad L_i(x) = \prod_{\substack{k=1 \\ k \neq i}}^{n} \frac{x - x_k}{x_i - x_k}, \quad f(x_i) = y_i$$

- Problematisch bei Lagrange: Neubestimmung der Gewichte beim Hinzufügen neuer Stützstellen
- Newtonscher Algorithmus: Alte Gewichte können beibehalten werden

$$\begin{aligned} p(x) = b_0 &+ b_1(x - x_0) \\ &+ b_2(x - x_0)(x - x_1) \\ &+ b_3(x - x_0)(x - x_1)(x - x_2) \\ &+ \dots \end{aligned}$$

- Die bereits berechneten Gewichte ändern sich nicht, wenn man neue Stützstellen dazu nimmt.

$$\begin{aligned} y_0 &= b_0 + b_1 \underbrace{(x_0 - x_0)}_{=0} + b_2 \underbrace{(x_0 - x_0)}_{=0}(x_0 - x_1) + \dots \\ y_1 &= b_0 + b_1(x_1 - x_0) + (x_1 - x_0) b_2 \underbrace{(x_1 - x_1)}_{=0} + \dots \\ y_2 &= + \dots \end{aligned}$$

- Viele weitere Methoden, z.B. der Algorithmus von Neville-Aitken

- Beschreibung aller Messwerte durch ein einzelnes Polynom nicht immer möglich/sinnvoll
- Lineare Interpolation: Der Wert von x zwischen x_j und x_{j+1} wird durch eine Gerade approximiert
- Beispiel: 3 Stützstellen x_1, x_2, x_3

$$f_{12}(x) = \frac{x - x_2}{x_1 - x_2} y_1 + \frac{x - x_1}{x_2 - x_1} y_2,$$
$$f_{23}(x) = \frac{x - x_3}{x_2 - x_3} y_2 + \frac{x - x_2}{x_3 - x_2} y_3$$

- Verallgemeinernd: Nehme jeweils m benachbarte Stützstellen und beschreibe sie durch ein Polynom $(m-1)$. Grades
- Beispiel: bei 6 Stützstellen gibt es folgende Interpolationsmöglichkeiten
  - 5 lineare Funktionen $f_{12}(x)$, $f_{23}(x)$, $f_{34}(x)$, $f_{45}(x)$, $f_{56}(x)$
  - 4 Polynome 2. Grades $f_{123}(x)$, $f_{234}(x)$, $f_{345}(x)$, $f_{456}(x)$
  - 3 Polynome 3. Grades $f_{1234}(x)$, $f_{2345}(x)$, $f_{3456}(x)$
  - 2 Polynome 4. Grades $f_{12345}(x)$, $f_{23456}(x)$
  - 1 Polynome 5. Grades $f_{123456}(x)$

- Nachteile der Polynominterpolation
  - Bei hohen Polynomgraden wird die Interpolation schlechter (Runges Phänomen)
  - Die Interpolationsfunktionen sind stetig aber i.d.R. nicht stetig differenzierbar
- Probleme mit den unstetigen Ableitungen lassen sich durch Spline-Interpolation vermeiden
- Idee: Approximiere den Wert von x zwischen x_j und x_{j+1} durch ein Polynom höheren Grades (mind. 3)
- Bei der Bestimmung der Gewichte werden nicht nur benachbarte, sondern *alle* Stützstellen einbezogen.
- Die Ableitungen sind bis zu einer bestimmten Ordnung garantiert stetig
- Häufig benutzt man dabei Polynome 3. Grades, d.h. kubische Splines

- Beispiel: Kubische Splines bei 4 Stützstellen. Interpolationsfunktionen:

$$f_{12}(x) = y_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3,$$
$$f_{23}(x) = y_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3,$$
$$f_{34}(x) = y_3 + b_3(x - x_3) + c_2(x - x_3)^2 + d_2(x - x_3)^3$$

- 9 Unbekannte, man braucht also 9 Gleichungen
- Stetigkeit der Interpolationsfunktion: 3 Gleichungen

$$f_{12}(x_2) = y_2, \quad f_{23}(x_3) = y_3, \quad f_{34}(x_4) = y_4$$

- 1. und 2. Ableitungen an den inneren Stützstellen müssen auch stetig sein: 4 Gleichungen

$$f'_{12}(x_2) = f'_{23}(x_2), \quad f'_{23}(x_3) = f'_{34}(x_3)$$
$$f''_{12}(x_2) = f''_{23}(x_2), \quad f''_{23}(x_3) = f''_{34}(x_3)$$

- Es fehlen noch 2 Bedingungen!

- Mögliche Wahl: Verschwinden der 2. Ableitungen an den äußeren Stützstellen $\Rightarrow$ natürliche Splines

$$f''_{12}(x_1) = 0, \quad f''_{34}(x_4) = 0$$

Nachbesprechung Übungsaufgaben

- Aufgabe 10: Der Einheitskreis passt nicht in ein Einheitsquadrat, dafür aber ein Viertelkreis → `a10-pi-monte-carlo.cc`
- Aufgabe 11: Einfach bereits Gelerntes anwenden → `a11-progwett.cc`
- Aufgabe 12: Münzwerte: $\{1, 3, 8, 20, 50, 100, 250\}$
- Beispiel: Rückgeld beträgt 75 Cent

1. Iteration: $75/250 = 0, \quad 75 \bmod 250 = 75,$
2. Iteration: $75/100 = 0, \quad 75 \bmod 100 = 75,$
3. Iteration: $75/50 = 1, \quad 75 \bmod 50 = 25,$
4. Iteration: $25/20 = 1, \quad 25 \bmod 20 = 5,$
5. Iteration: $5/8 = 0, \quad 5 \bmod 8 = 5,$
6. Iteration: $5/3 = 1, \quad 5 \bmod 3 = 2,$
7. Iteration: $2/1 = 2, \quad 2 \bmod 1 = 0$

- Somit $75 = 50 \times 1 + 20 \times 1 + 3 \times 1 + 1 \times 2$ → `a12-muenzen.cc`

Vorbesprechung Übungsaufgaben

- Aufgabe 14: Beispiel für eine vollkommene Zahl

$$1 + 2 + 3 + 4 + 5 + 6 + 7 = 28$$

- Für jede Zahl n kann man die Teilersumme berechnen, d.h. die Summe aller Zahlen durch die n ohne Rest teilbar ist
- Der Größte Teiler muss kleiner als die Hälfte von n sein.
- Aufgabe 15: Numerische Auslöschung ohne Pivotsuche

$$\left[\begin{array}{cc|c} 1 & a & c \\ 10^n & b & d \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 1 & a & c \\ 0 & b - 10^n a & d - 10^n c \end{array}\right] \approx \left[\begin{array}{cc|c} 1 & a & c \\ 0 & -10^n a & -10^n c \end{array}\right]$$

- Mit Pivotsuche würde man zuerst die beiden Zeilen vertauschen

$$\left[\begin{array}{cc|c} 10^n & b & d \\ 1 & a & c \end{array}\right] \rightarrow \left[\begin{array}{cc|c} 10^n & b & d \\ 0 & a - b/10^n & c - d/10^n \end{array}\right] \approx \left[\begin{array}{cc|c} 10^n & b & d \\ 0 & a & c \end{array}\right]$$

- Einlesen der Matrix, Bildschirmausgabe $\rightarrow$ unkompliziert
- Struktur des Algorithmus bereits vorgegeben (siehe Pseudocode aus der Vorlesung)
  - Äußere Schleife geht die Matrix Zeile für Zeile durch (i)
  - Pivotsuche in der Spalte unter `mat[i][i]`
  - Gibt es (`mat[i][i] == 0`) trotz Pivotsuche?
  - Elimination der Einträge unter `mat[i][i]`: Man addiert ein geeignetes Vielfaches der Zeile i zur Zeile j
- Anschließend folgen das Rückwärtseinsetzen und die Ausgabe des Lösungsvektors
- Bei Unklarheiten: ein 3×3 LGS aufstellen und auf Papier mit Gauß-Elimination lösen!