Numerik 2 Hausaufgabe 03 - Projekt 1 Interpolation

Verfasser: Wahid Far, Okhtay
Matrikelnr.: 870485

Index

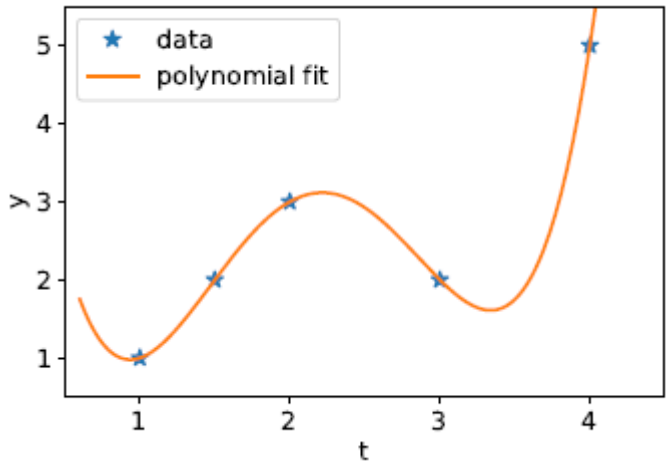
1)Interpolationsproblem

Der Begriff Interpolation setzt sich aus Folgendem zusammen:
i)inter == (lat.) zwischen
ii)polation == (lat.) glätten,polieren

Man stellt sich also die Frage, ob es eine Funktion bzw. Graf gibt, der die vorgegebenen Punkte (Stützstellen) interpoliert. Das heißt, die gesuchte Funktion soll die Stützstellen durchlaufen und die Punkte dazwischen sozusagen zu einer Funktion glätten. Sie ist also unendlich oft differenzierbar und besitzt keine Ecken.
Es muss also folgendes gelten:

$$p(x_i) = f(x_i) = y \text{ mit } i \in \mathbb{N} \text{ und } x \in \mathbb{R}$$

Das heißt, $p(x_i)$ ist jene Funktion, die die vorgegebenen Punkte $f(x_i)$ durchlaufen soll.



(Quelle: Folie 2 des Skriptes „Interpolation und Approximation“ von Prof. Frank Haußer)

Anwendung findet die Interpolation beim Vergleich von Messwerten und deren Fehlertoleranz. Zudem wird sie beim Approximieren von komplexeren Funktionen verwendet, wobei eine Interpolation zwischen den gegebenen Punkten durch eine einfachere Funktion stattfindet.

2)Polynominterpolation

Eine Polynominterpolation beantwortet das Interpolationsproblem mithilfe einer Polynomfunktion. Hierzu ist das Verständnis der Begriffe eines reellen Polynoms und Stützstellen erforderlich.

Definition- reelles Polynom:

Ein (reellwertiges) Polynom ist ein Term der Form

$$p(x) = \sum_{i=0}^{n} a_i x^k = a_0 x^0 + a_1 x^1 + \cdots + a_n x^n$$

mit $a_i \in \mathbb{R}$ und $k \in \mathbb{N}$, wobei a_i als reelle Koeffizienten und $a_i x^k$ als Monome von p bezeichnet werden.

Definition – Stützstelle/Stützpunkte:

Sei die Stützstellenmenge $S \subseteq \mathbb{R}$. Eine Polynomfunktion $f : S \rightarrow \mathbb{R}$ interpoliert die Stützstellen wenn gilt :

$f(x_k) = y_k$ mit $k \in \{0,\ldots,n\}$

Ein interpolierendes Polynom wird auch als Interpolationspolynom der Stützstellen bezeichnet.

Interpolation, Beispiel

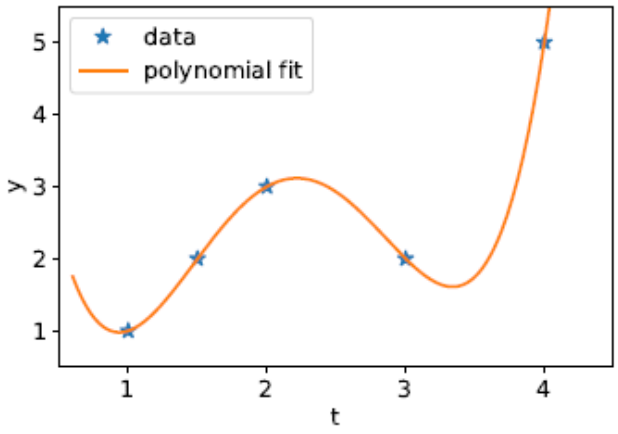
Gegeben:

- 5 Datenpunkte (x_i, y_i), $i = 0,\ldots 4$: **Stützpunkte**
- x_i: **Stützstellen, Knoten**, paarweise verschieden
- y_i: **Stützwerte**

Gesucht:

- Interpolierende Funktion g:

$$g(x_i) = y_i, \quad i = 0,\ldots 4 \quad \textbf{Interpolationseigenschaft}$$



(Quelle: Folie 2 des Skriptes „Interpolation und Approximation“ von Prof. Frank Haußer)

Wir betrachten hier das Verfahren der Polynominterpolation mit der Newton-Basis.

Definition- Polynominterpolation nach Newton:

Gegeben seien die Stützstellen $x_1, \ldots, x_n$ und

$p(x) = \sum_{k=0}^{n} b_i * N_i(x)$ mit k,i,n $\in \mathbb{N}$

$b_i = f[\, x_0, x_1, \ldots, x_i \,]$ == konstante Koeffizienten, dividierte Differenzen

$N_i(x) = \prod_{j=0}^{i-1}(x - x_j)$ == Basisfunktion nach Newton

Um die dividierten Differenzen praktisch zu berechnen, nutzen wir folgendes Schema:

Beispiel 1.11. Berechnung der dividierten Differenzen, $n = 3$

x_i	$f[x_i]$	$f[x_{i-1}, x_i]$	$f[x_{i-2}, x_{i-1}, x_i]$	$f[x_{i-3}, x_{i-2}, x_{i-1}, x_i]$
x_0	y_0			
x_1	y_1	$\frac{f[x_1]-f[x_0]}{x_1-x_0}$		
x_2	y_2	$\frac{f[x_2]-f[x_1]}{x_2-x_1}$	$\frac{f[x_1,x_2]-f[x_1,x_0]}{x_2-x_0}$	
x_3	y_3	$\frac{f[x_3]-f[x_2]}{x_3-x_2}$	$\frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$	$\frac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0}$

(Quelle: Seite 7 vom „Numerik 2 Skript“ von Prof. Frank Haußer)

Analog gilt dieses Schema auch für n

Bsp noch mit echten Zahlen vlcht einbauen

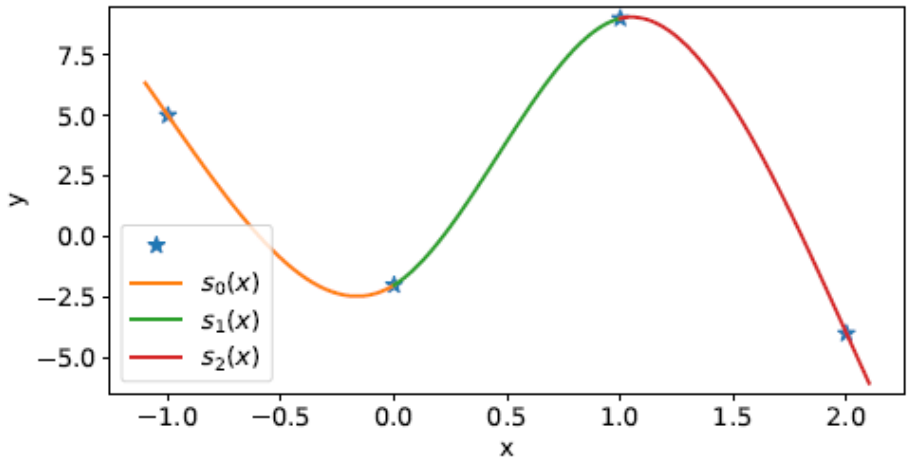
Es ist hierbei anzumerken, dass jede Polynominterpolation des gleichen Grades n eindeutig ist. Das heißt, eine Polynominterpolation mit den identischen Stützstellen würde nach dem Lagrange und Newton Verfahren die gleiche Funktion erzeugen. Somit existiert nur eine interpolierende Funktion des gleichen Grades n mit den gleichen Stützstellen.

3) Splineinterpolation

Bei einer hohen Anzahl von Stützstellen entsteht eine hohe Oszillation des Interpolationspolynoms. Das heißt, je höher der Grad ist desto stärker ist die Oszillation.
Dadurch entstehen an den Übergängen des Polynoms „Ecken“, die durch die Festlegung von natürlichen Randbedingungen und durch die Nutzung eines

kubischen Polynoms (dritten Grades) zwischen zwei Datenpunkten (Teilintervall) geglättet werden.

Stückweise kubische Interpolation



- Jedes lokale Polynom s_j: 4 freie Parameter
- Zwei Interpolationsbedingungen
- Fordere zusätzlich: Stetigkeit der 1. und der 2. Ableitung

$$s'(x_j) = s'_j(x_j) = s'_{j-1}(x_j)$$
$$s''(x_j) = s''_j(x_j) = s''_{j-1}(x_j)$$

(Quelle: Seite 7 des Skriptes „Spline Interpolation“ von Prof. Frank Haußer)

Wir betrachten also hier die kubische Splineinterpolation mit natürlichen Randbedingungen.

Vorgehen - kubische Splineinterpolation mit natürlichen Randbedingungen:

Sei S(x) ein kubischer Spline mit natürlichen Randbedingungen und j ∈ {0,…, n-1} dann gilt als Ansatz:

$$m_j = s''(x_j)$$

$$s_j''(x) = m_j \frac{(x_{j+1}-x)}{h} + m_{j+1} \frac{(x-x_j)}{h}$$

$$h_j = x_{j+1} - x_j$$

a_j, b_j , c_j, d_j == zu berechnenden Parameter

Somit können wir folgendes LGS lösen

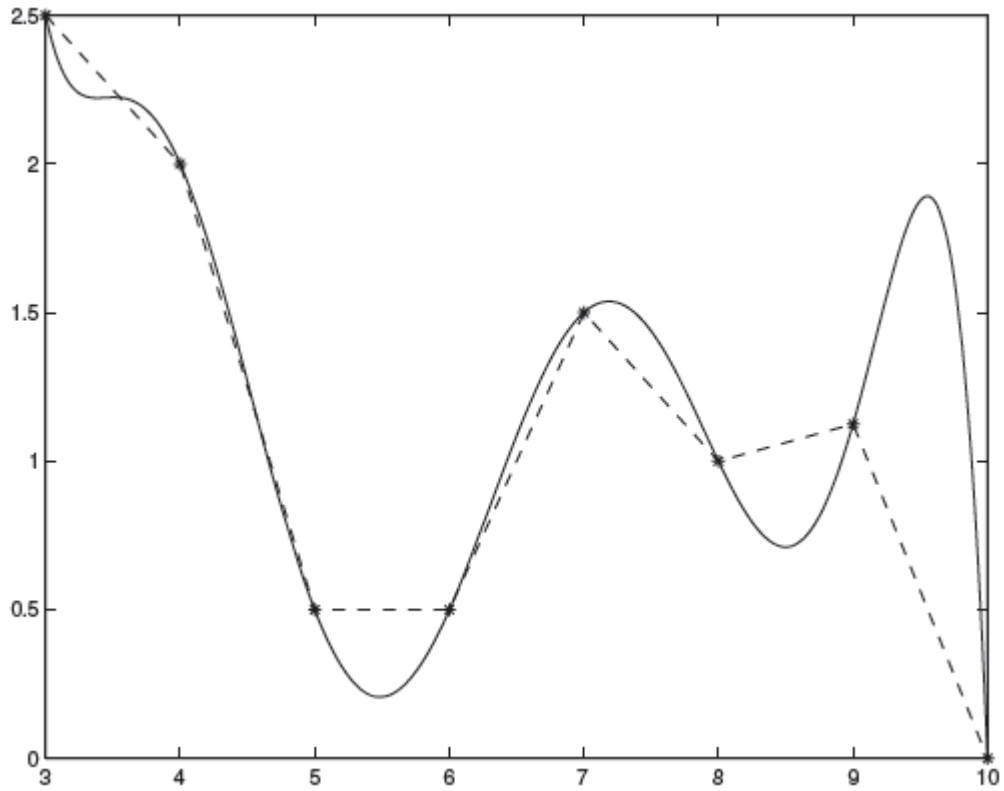
$$\begin{pmatrix} 4 & 1 & & \\ 1 & 4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 4 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_{n-1} \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} y_0 - 2y_1 + y_2 \\ y_1 - 2y_2 + y_3 \\ \vdots \\ y_{n-2} - 2y_{n-1} + y_n \end{pmatrix}$$

(Quelle: Seite 21 Skript „Numerik 2“ von Prof. Frank Haußer)

Das heißt bei folgendem Datensatz würde folgende LGS entstehen:

i	0	1	2	3	4	5	6	7
x_i	3	4	5	6	7	8	9	10
$f(x_i)$	2.5	2.0	0.5	0.5	1.5	1.0	1.125	0.0

$$\begin{pmatrix} 4&1&0&0&0&0 \\ 1&4&1&0&0&0 \\ 0&1&4&1&0&0 \\ 0&0&1&4&1&0 \\ 0&0&0&1&4&1 \\ 0&0&0&0&1&4 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \end{pmatrix} = 6 \begin{pmatrix} -1 \\ 1.5 \\ 1 \\ -1.5 \\ 0.625 \\ -1.25 \end{pmatrix}$$



(Quelle: „Numerik für Ingenieure“ von Dahmen)

Wir können sehen wie durch die kubischen Polynome zwischen den Datenpunkten die Ecken verschwinden.

4)Bezug zur Code-Implementierung

Zu Aufgabe 1)

Für die natürlichen Splines s stellen wir für die Bestimmung der zweiten Ableitungen mit den Knoten xi folgendes LGS auf:

$$\begin{pmatrix} 4 & 1 & & \\ 1 & 4 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 4 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_{n-1} \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} y_0 - 2y_1 + y_2 \\ y_1 - 2y_2 + y_3 \\ \vdots \\ y_{n-2} - 2y_{n-1} + y_n \end{pmatrix}$$

(Quelle: Seite 21 Skript „Numerik 2“ von Prof. Frank Haußer)

Hierbei gilt m_0 = m_n = 0 und das LGS hat die Dimension n-1 für die Unbekannten m_1 bis m_n-1

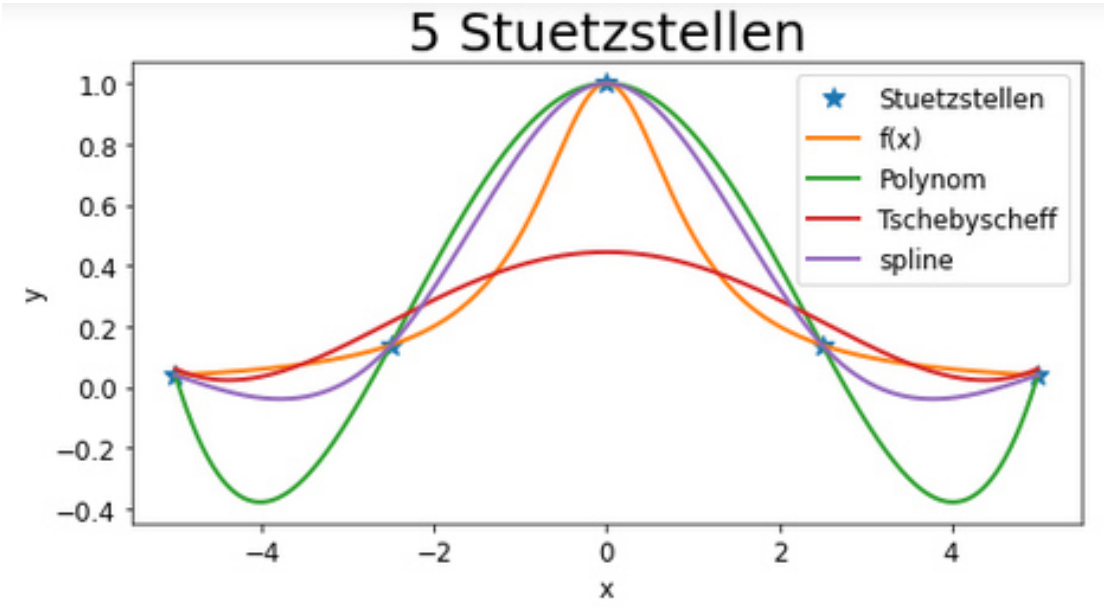
Code dazu:

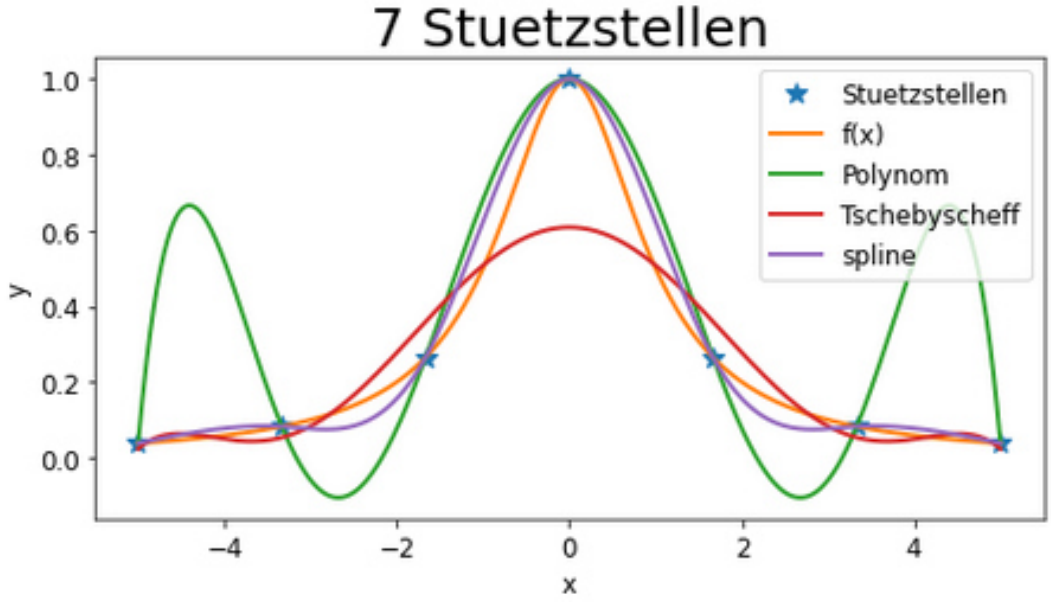
```
def curv(yi, h):
    '''Berechnung der zweiten Ableitungen des natuerlichen Splines s an den Knoten xi'''
    mi = np.zeros(xi.shape) # zweiten Ableitungen
    b = np.zeros(xi.shape)  # rechte Seite des LGS
    # Berechnung der rechten Seite des LGS
    b[1:-1] = yi[:-2] - 2*yi[1:-1] + yi[2:]
    b = b * 6 / h**2

    N = len(b) - 2 # n-2 unbekannte
    A = np.zeros((N, N), dtype=float) # linke Seite LGS
    # Erstellung der Tridiogonal-Matrix mit Werten [1,4,1]
    for i in range(0,N-1):
        A[i][i] = 4
        A[i][i+1] = 1
        A[i+1][i] = 1
    A[-1][-1] = 4
    # Berechnung der mi durch loesen der LGS
    mi[1:-1] = np.linalg.solve(A, b[1:-1])
    return mi
```

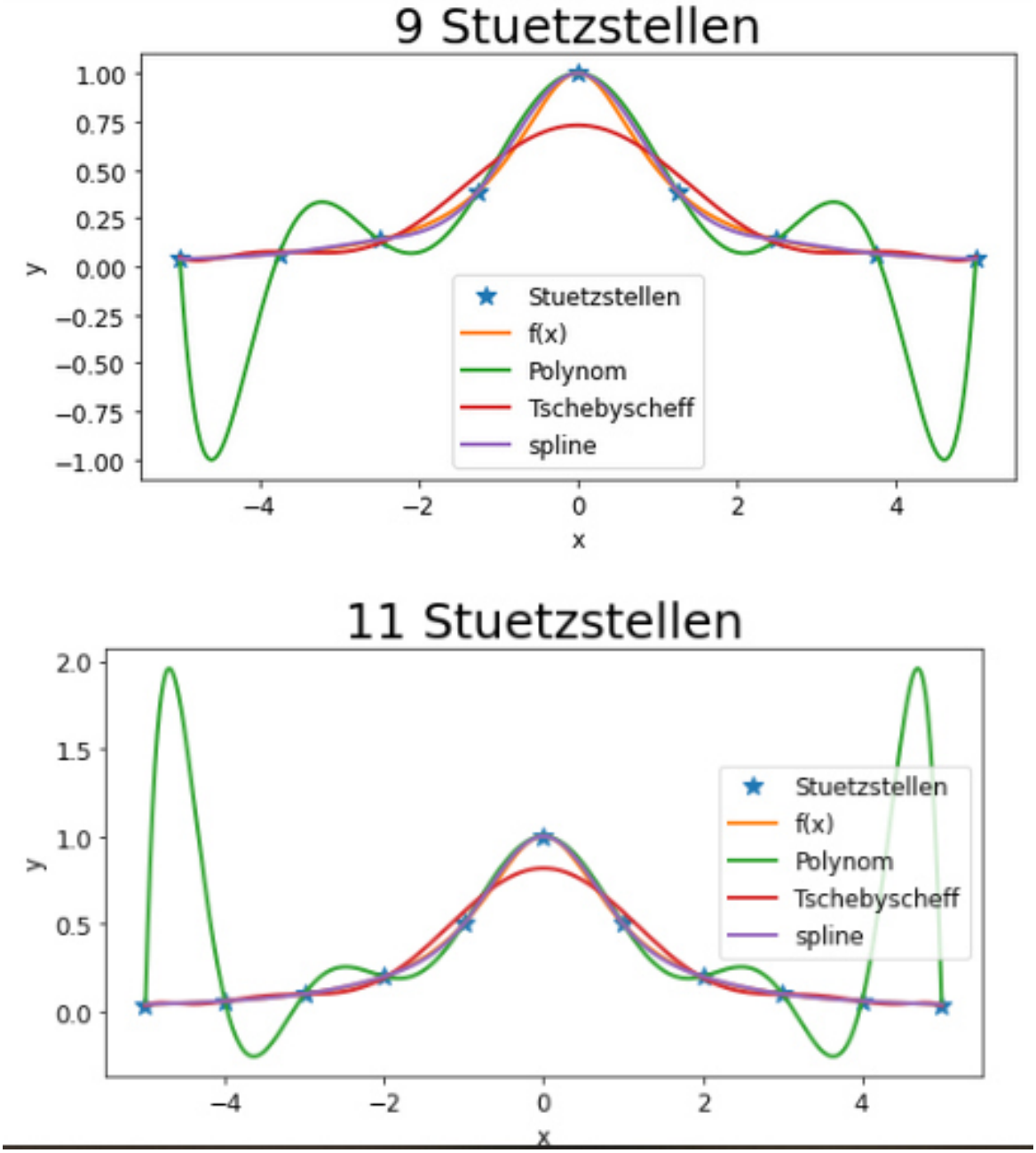
Zu Aufgabe 2)

Wir stellen fest, dass durch die höhere Anzahl der Stützstellen die Polynominterpolation stärker oszilliert. Das heißt, eine größere Anzahl von Stützstellen verbessert nicht die Polynominterpolation.

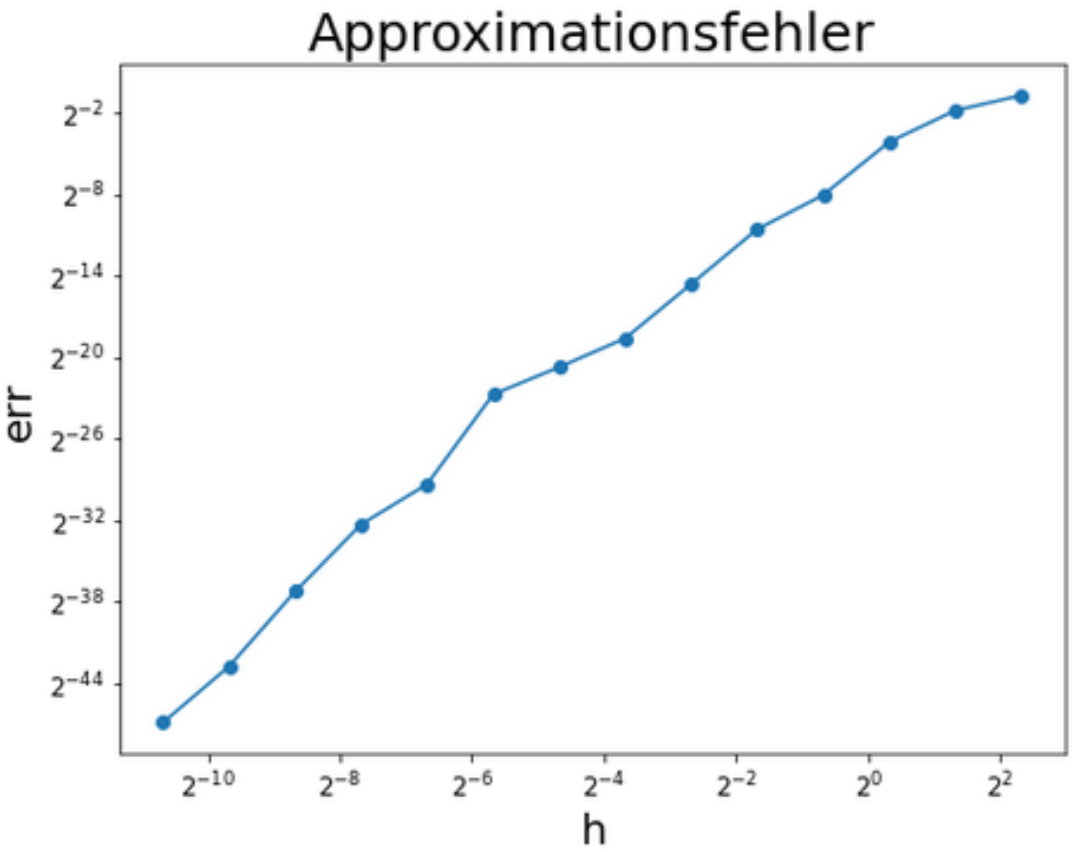
Bei der Spline-Interpolation kann man bei kleineren Schrittwerten (also einer höheren Stützstellenanzahl) eine Verbesserung der Interpolation erkennen.


5 Stuetzstellen
Stuetzstellen
f(x)
Polynom
Tschebyscheff
spline
y
x
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−4
−2
0
2
4


7 Stuetzstellen
Stuetzstellen
f(x)
Polynom
Tschebyscheff
spline
y
x
1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4



Zudem gilt: Je kleiner die Schrittweite h wird desto kleiner ist der Approximationsfehler der L-Unendlich Norm.


Approximationsfehler
err
h
2^{-2}
2^{-8}
2^{-14}
2^{-20}
2^{-26}
2^{-32}
2^{-38}
2^{-44}
2^{-10}
2^{-8}
2^{-6}
2^{-4}
2^{-2}
2^{0}
2^{2}