

Numerisches Programmieren, Übungen

Musterlösung 8. Übungsblatt: Stückweise Interpolation

1) Hermite-Interpolation

a) Mit den Interpolationsbedingungen (1)

$$p(x_0) = y_0, \quad p(x_1) = y_1, \quad p'(x_0) = y'_0, \quad p'(x_1) = y'_1$$

sowie dem allgemeinen Ansatz für kubische Polynome

$$p(t) = a_0 + a_1t + a_2t^2 + a_3t^3, \quad t \in [0; 1] = [x_0; x_1]$$

bzw. dessen Ableitung

$$p'(t) = a_1t + 2a_2t + 3a_3t^2, \quad t \in [0; 1] = [x_0; x_1]$$

ergibt sich das folgende lineare Gleichungssystem:

$$\begin{array}{rcll} p(t=0) & = & a_0 & \stackrel{!}{=} y_0 \\ p(t=1) & = & a_0 + a_1 + a_2 + a_3 & \stackrel{!}{=} y_1 \\ p'(t=0) & = & a_1 & \stackrel{!}{=} y'_0 \\ p'(t=1) & = & a_1 + 2a_2 + 3a_3 & \stackrel{!}{=} y'_1 \end{array}$$

Nach etwas Umsortieren erhalten wir in Matrix-Vektor-Notation

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} y_1 \\ y'_1 \\ y'_0 \\ y_0 \end{pmatrix}$$

und Umformen auf Zeilenstufenform liefert

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 1 & y_1 \\ 0 & 1 & 2 & 3 & -y'_1 + 3y_1 \\ 0 & 0 & 1 & 0 & y'_0 \\ 0 & 0 & 0 & 1 & y_0 \end{array} \right).$$

Mit Rückwärtssubstitution ergibt sich somit als Lösung

$$\begin{aligned} a_0 &= y_0 \\ a_1 &= y'_0 \\ a_2 &= -y'_1 + 3y_1 - 2y'_0 - 3y_0 \\ a_3 &= y_1 - (-y'_1 + 3y_1 - 2y'_0 - 3y_0) - y'_0 - y_0 \\ &= 2y_0 + y'_0 - 2y_1 + y'_1 \end{aligned}$$

Damit erhalten wir das kubische Polynom

$$p(t) = y_0 + y'_0 t + (-y'_1 + 3y_1 - 2y'_0 - 3y_0)t^2 + (2y_0 + y'_0 - 2y_1 + y'_1)t^3$$

bzw. sortiert nach y_0 , y_1 , y'_0 und y'_1 :

$$p(t) = y_0 \cdot (1 - 3t^2 + 2t^3) + y_1 \cdot (3t^2 - 2t^3) + y'_0 \cdot (t - 2t^2 + t^3) + y'_1 \cdot (-t^2 + t^3).$$

Mittels Koeffizientenvergleich bzgl. den Variablen y_0 , y_1 , y'_0 und y'_1 erhalten wir schließlich die kubischen Basispolynome H_i , $i = 0, 1, 2, 3$, des Hermite-Ansatzes

$$p(t) = y_0 \cdot H_0(t) + y_1 \cdot H_1(t) + y'_0 \cdot H_2(t) + y'_1 \cdot H_3(t).$$

Sie ergeben sich zu

$$\begin{aligned} H_0(t) &= 1 - 3t^2 + 2t^3 \\ H_1(t) &= 3t^2 - 2t^3 \\ H_2(t) &= t - 2t^2 + t^3 \\ H_3(t) &= -t^2 + t^3. \end{aligned}$$

b) Die 4 Bedingungen für das Teilintervall $[x_i, x_{i+1}]$ lauten

$$\begin{aligned} p(x_i) &= y_i, & p(x_{i+1}) &= y_{i+1} \\ p'(x_i) &= y'_i, & p'(x_{i+1}) &= y'_{i+1}. \end{aligned}$$

Um in $[x_i, x_{i+1}]$ die Basispolynome $H_0(t)$ bis $H_3(t)$, $t \in [0; 1]$, nutzen zu können, müssen wir unsere Variable x vorher noch mit der Funktion $t_i(x)$ auf das Intervall $[0; 1]$ transformieren. Das erreichen wir mit

$$t_i(x) := \frac{x - x_i}{x_{i+1} - x_i} = \frac{x - x_i}{h_i} \in [0; 1]$$

Damit ist die meiste Arbeit getan, es muss lediglich noch ein Detail beachtet werden: Wenn wir die transformierte Funktion ableiten, müssen wir die Kettenregel berücksichtigen. Das gibt einen zusätzlichen Faktor h_i , wie man folgendermaßen sieht:

$$\frac{d}{dx} (f(t_i(x))) = \frac{d}{dt} (f(t_i(x))) \cdot \frac{dt}{dx} = \frac{d}{dt} (f(t_i(x))) \cdot \frac{1}{x_{i+1} - x_i} = \frac{d}{dt} (f(t_i(x))) \cdot \frac{1}{h_i}.$$

Um nun das kubische Teilpolynom $p_i(t(x))$ mit Hilfe des Hermite-Ansatzes festzulegen, müssen wir bei den Summanden mit y'_i und y'_{i+1} einen Faktor h_i dazunehmen, weil dort bei der Ableitung ein $1/h_i$ auftauchen würde, das bei der Interpolation nicht auftauchen darf! Somit erhalten wir für ein Teilintervall $[x_i, x_{i+1}]$:

$$p_i(t_i(x)) = y_i \cdot H_0(t_i(x)) + y_{i+1} \cdot H_1(t_i(x)) + h_i \cdot y'_i \cdot H_2(t_i(x)) + h_i \cdot y'_{i+1} \cdot H_3(t_i(x)).$$

In Abb. 1 ist beispielhaft das interpolierende Hermite-Polynom zur Funktion $\sin(x)$ auf dem Intervall $[0; 4\pi]$ dargestellt.

Dieses Bild wurde mit der *matlab*-Routine *visHermiteSin.m* erstellt, das auch auf der Webpage unter der Rubrik *Tutorium* zum Download bereitgestellt ist. Für die Benutzung der Datei unter *octave* muss lediglich der entsprechende plot-Bereich ein- bzw. auskommentiert werden.

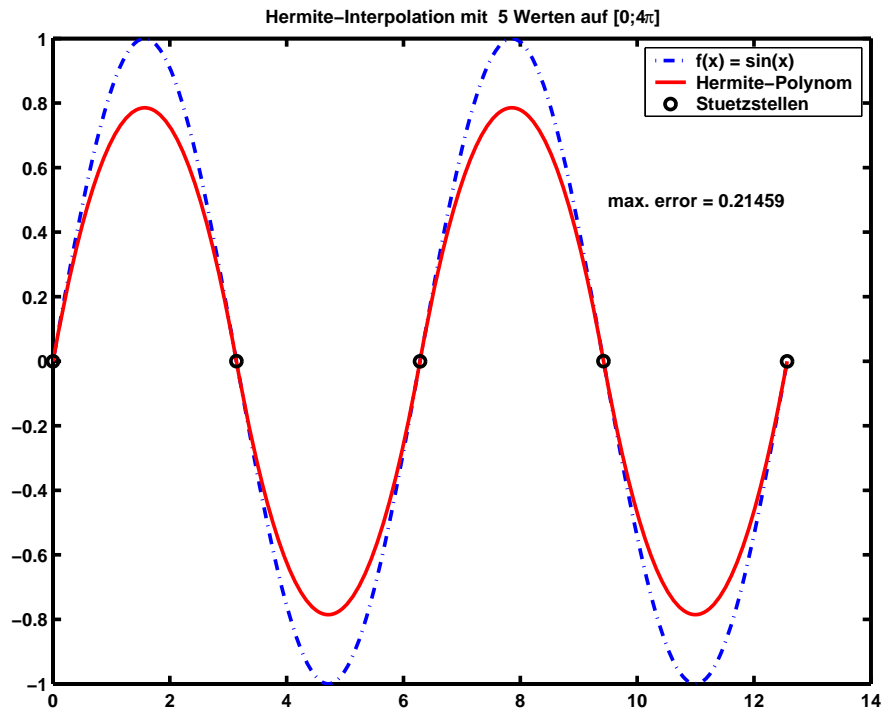


Abbildung 1: Visualisierung eines Hermite-Polynoms als Interpoland von $\sin(x)$ auf $[0; 4\pi]$.

2) Interpolation mit kubischen Splines

a) Zunächst bestimmen wir explizit eine Spline-Funktion. Als Werte sind hierfür gegeben:

i	0	1	2	3
x_i	-1	0	1	2
y_i	2	0	2	3
y'_i	9	?	?	0

Die Spline-Funktion $s(x)$ ist stückweise definiert über

$$s(x) = p_i(t_i(x)) \quad \text{für } x \in [x_i, x_{i+1}]$$

mit der Transformation auf das Einheitsintervall

$$[x_i, x_{i+1}] \rightarrow [0, 1], \quad x \mapsto t_i(x) = \frac{x - x_i}{h_i}$$

und den kubischen Funktionen

$$p_i(t_i(x)) = y_i \cdot H_0(t_i(x)) + y_{i+1} \cdot H_1(t_i(x)) + h \cdot y'_i \cdot H_2(t_i(x)) + h \cdot y'_{i+1} \cdot H_3(t_i(x)).$$

Es sind also noch die fehlenden Ableitungen y'_i zu bestimmen, bevor wir die Funktion vollständig aufstellen können.

Da die Stützstellen äquidistant sind, können wir das lineare Gleichungssystem verwenden, um y'_1 und y'_2 zu bestimmen. Speziell für diesen Fall lautet es

$$\begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix} = \frac{3}{h} \begin{pmatrix} y_2 - y_0 - \frac{h}{3}y'_0 \\ y_3 - y_1 - \frac{h}{3}y'_3 \end{pmatrix}.$$

Setzen wir $h = 1$ sowie die y -Werte ein, so erhalten wir

$$\begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix} = 3 \begin{pmatrix} 2 - 2 - \frac{1}{3} \cdot 9 \\ 3 - 0 - \frac{1}{3} \cdot 0 \end{pmatrix} = 9 \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

und als Lösung ergibt sich schließlich

$$\begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix} = \begin{pmatrix} -3 \\ 3 \end{pmatrix}.$$

Wir erhalten damit als Spline-Funktion

$$s(x) = \begin{cases} p_0(t_0(x)) & \text{für } x \in [-1, 0[\text{ mit } t_0(x) = x + 1 \\ p_1(t_1(x)) & \text{für } x \in [0, 1[\text{ mit } t_1(x) = x \\ p_2(t_2(x)) & \text{für } x \in [1, 2] \text{ mit } t_2(x) = x - 1 \end{cases}$$

mit

$$\begin{aligned} p_0(t) &= 2 \cdot H_0(t) && + 9 \cdot H_2(t) && - 3 \cdot H_3(t) \\ p_1(t) &= && 2 \cdot H_1(t) && - 3 \cdot H_2(t) && + 3 \cdot H_3(t) \\ p_2(t) &= 2 \cdot H_0(t) && + 3 \cdot H_1(t) && + 3 \cdot H_2(t). \end{aligned}$$

- b) Ziel: Darstellung von $s(x)|_{[x_i; x_{i+1}]}$ mittels stückweiser kubischer Polynome $p_i(x)$ (analog zur Aufgabe 1)).

Interpolationsbedingung:

$$s(x_i) = y_i, \quad i = 0, \dots, n \quad (n + 1 \text{ Bedingungen})$$

\mathcal{C}^2 -Stetigkeit: entscheidend nur die „Klebestellen“ x_1, \dots, x_{n-1} , dazwischen kann man jedes stückweise kubische Polynom immer problemlos stetig ableiten:

$$s'(x_{i+1}) = y'_{i+1}, \quad i = 0, \dots, n - 2 \quad (\mathcal{C}^1\text{-Stetigkeit, notwendig für } \mathcal{C}^2) \quad (1)$$

$$s''(x_{i+1}) = y''_{i+1}, \quad i = 0, \dots, n - 2 \quad (n - 1 \text{ Bedingungen}). \quad (2)$$

- c) Wir werden sofort in Teilaufgabe d) sehen, warum wir die zweiten Ableitungen der Hermite-Basispolynome brauchen. Hier berechnen wir sie schnell:

$$\begin{aligned} H_0''(t) &= -6 + 12t = \begin{cases} -6, & \text{für } t = 0 \\ 6, & \text{für } t = 1 \end{cases} \\ H_1''(t) &= 6 - 12t = \begin{cases} 6, & \text{für } t = 0 \\ -6, & \text{für } t = 1 \end{cases} \\ H_2''(t) &= -4 + 6t = \begin{cases} -4, & \text{für } t = 0 \\ 2, & \text{für } t = 1 \end{cases} \\ H_3''(t) &= -2 + 6t = \begin{cases} -2, & \text{für } t = 0 \\ 4, & \text{für } t = 1 \end{cases} . \end{aligned}$$

- d) Wie in Aufgabe 1) können wir für die stückweisen kubischen Polynome $p_i(x)$ den Hermite-Ansatz machen, auch wenn wir die ersten Ableitungen y'_i ($i = 1, \dots, n - 1$) aus (1) noch nicht kennen. Wieder benötigen wir die Transformationsfunktion $t_i(x) = (x - x_i)/(x_{i+1} - x_i) = (x - x_i)/h$, die mit unseren äquidistanten Stützstellen immer dasselbe h benutzt, um unseren x -Wert ins Intervall $[0; 1]$ zu bekommen:

$$p_i(t_i(x)) = y_i \cdot H_0(t_i(x)) + y_{i+1} \cdot H_1(t_i(x)) + h \cdot y'_i \cdot H_2(t_i(x)) + h \cdot y'_{i+1} \cdot H_3(t_i(x)) \quad (3)$$

Die Bedingungen (2) erlauben uns nun, die y'_i ($i = 1, \dots, n - 1$) zu bestimmen, indem wir einfach die zweite Ableitung der Funktionen $p_i(t(x) = 1)$ und $p_{i+1}(t(x) = 0)$ an den „Klebestellen“ x_{i+1} ($i = 0, \dots, n - 2$) gleichsetzen:

$$\begin{aligned} p_i''(t = 1) &= s''(x_{i+1}) = p_{i+1}''(t = 0), \quad i = 0, 1, \dots, n - 2 \\ &\Leftrightarrow \\ &\frac{1}{h^2} [y_i \cdot H_0''(1) + y_{i+1} \cdot H_1''(1) + h \cdot y'_i \cdot H_2''(1) + h \cdot y'_{i+1} \cdot H_3''(1)] = \\ &= \frac{1}{h^2} [y_{i+1} \cdot H_0''(0) + y_{i+2} \cdot H_1''(0) + h \cdot y'_{i+1} \cdot H_2''(0) + h \cdot y'_{i+2} \cdot H_3''(0)] \end{aligned}$$

Der Faktor $1/h^2$ kommt vom zweimaligen Ableiten der Basispolynome (Kettenregel mit $t_i(x)$, vgl. Aufg. 1)).

Jetzt sortieren wir das noch kurz nach Summanden mit y' auf der linken und y auf der rechten Seite:

$$\begin{aligned} \frac{h}{h^2} [y'_i \cdot H''_2(1) + y'_{i+1} \cdot (H''_3(1) - H''_2(0)) - y'_{i+2} \cdot H''_3(0)] &= \\ = \frac{1}{h^2} [-y_i \cdot H''_0(1) + y_{i+1} \cdot (-H''_1(1) + H''_0(0)) + y_{i+2} \cdot H''_1(0)]. \end{aligned}$$

Wenn wir jetzt die Werte für H''_i an der Stelle 0 bzw. 1 aus Teilaufgabe c) einsetzen, erhalten wir für $i = 0, \dots, n-2$:

$$\begin{aligned} \frac{1}{h} [y'_i \cdot 2 + y'_{i+1} \cdot (4 - (-4)) - y'_{i+2} \cdot (-2)] &= \\ = \frac{1}{h^2} [-y_i \cdot 6 + y_{i+1} \cdot (-(-6) + (-6)) + y_{i+2} \cdot 6] \\ \Leftrightarrow \\ [y'_i + 4y'_{i+1} + y'_{i+2}] &= \frac{3}{h} [y_{i+2} - y_i]. \end{aligned}$$

Das ist ein lineares Gleichungssystem der Dimension $n-1$ (analog zu Ergebnis aus Vorlesung, dort heißen die H_i nur α_{i+1}):

$$\begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & & 1 & 4 \end{pmatrix} \begin{pmatrix} y'_1 \\ y'_2 \\ \vdots \\ y'_{n-2} \\ y'_{n-1} \end{pmatrix} = \frac{3}{h} \begin{pmatrix} y_2 - y_0 \\ y_3 - y_1 \\ \vdots \\ y_{n-1} - y_{n-3} \\ y_n - y_{n-2} \end{pmatrix} - \begin{pmatrix} y'_0 \\ 0 \\ \vdots \\ 0 \\ y'_n \end{pmatrix}.$$

Dabei müssen die Randbedingungen y'_0 und y'_n gesetzt werden. Eine Möglichkeit ist z.B. $y'_0 = 0 = y'_n$ oder auch die echte Ableitung der zu interpolierenden Funktion bei x_0, x_n .

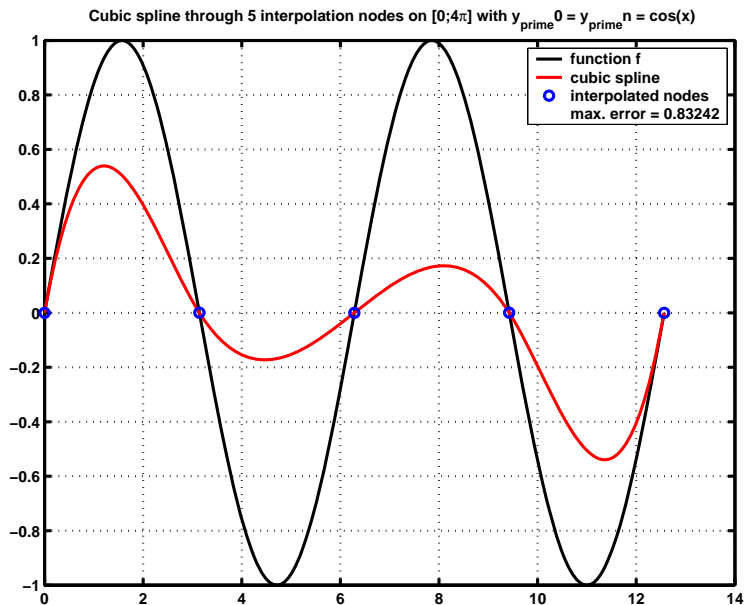


Abbildung 2: Visualisierung einer kubischen Spline-Funktion als Interpoland von $\sin(x)$ auf $[0; 4\pi]$ für 5 Punkte.

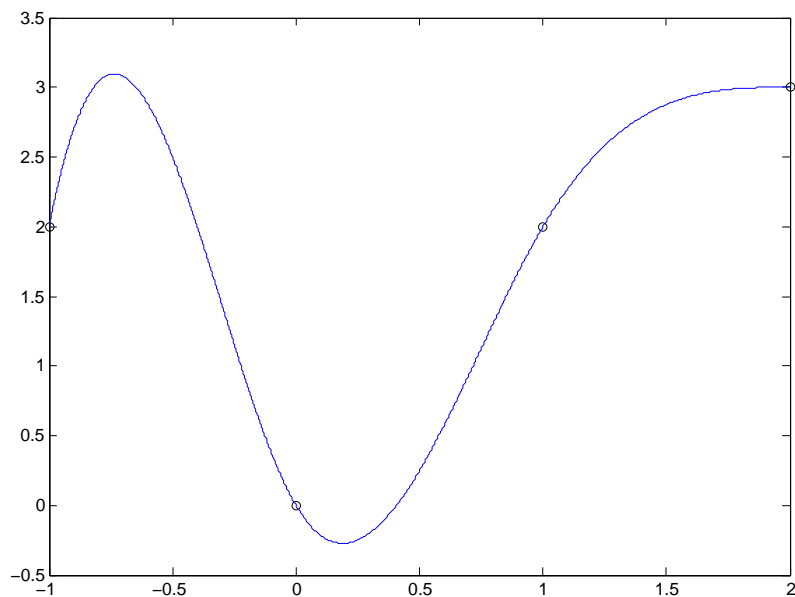


Abbildung 3: Visualisierung der berechneten Spline-Funktion $s(x)$

In Abb. 2 ist die kubische Splinefunktion zur Funktion $\sin(x)$ auf dem Intervall $[0; 4\pi]$ dargestellt. Man erkennt deutlich den Unterschied zur Hermite-Interpolation (selbe Situation). Allerdings hat das Hermite-Polynom mit Ableitungen an den Zwischenknoten mehr Informationen von der zu interpolierenden Funktion erhalten.

Splines sind per Konstruktion Funktionen, die möglichst glatt durch die vorgegebenen Interpolationspunkte laufen, da sie die Krümmung der Gesamtfunktion minimieren. Starke Oszillationen wie beispielsweise bei der Polynominterpolation hohen Grades können so nicht auftreten.

Abbildung 2 wurde mit der *matlab*-Routine `spline_easyN.m` erstellt, das auch auf der Webpage unter der Rubrik *Tutorium* zum Download bereitgestellt ist. Für die Benutzung der Datei unter *octave* muss lediglich der entsprechende plot-Bereich ein- bzw. auskommentiert werden.

Zu Splines gäbe es noch viel zu sagen. Es sind eine Reihe von Verallgemeinerungen möglich (nicht äquidistant, Kontrollpunkte abseits der Kurve,...). Details hierzu werden üblicherweise in Vorlesungen zu CAD diskutiert. Eine schöne Eigenschaft der Splines soll nicht unerwähnt bleiben: Eine Veränderung in einem Kontrollpunkt wirkt sich immer nur lokal auf einen kleinen Bereich der Kurve aus, der Großteil der Kurve bleibt unverändert.

Wiederholung: Interpolation

a) Dreiecksschema:

x_i	$i \setminus k$	0	1	2	3
0	0	$y_0 = 0$	\rightarrow 1	\rightarrow -1	
			\nearrow	\nearrow	
1	1	$y_1 = 1$	\rightarrow -1		
			\nearrow		
2	2	$y_2 = 0$			

Die Berechnungen im Einzelnen:

$$\begin{aligned}
 [x_0]f &= \sin(0) = 0 & [x_0, x_1]f &= \frac{[x_1]f - [x_0]f}{x_1 - x_0} = \frac{1 - 0}{1 - 0} = 1 \\
 [x_1]f &= \sin\left(\frac{\pi}{2}\right) = 1 & [x_1, x_2]f &= \frac{[x_2]f - [x_1]f}{x_2 - x_1} = \frac{0 - 1}{2 - 1} = -1 \\
 [x_2]f &= \sin(\pi) = 0 & [x_0, x_1, x_2]f &= \frac{[x_1, x_2]f - [x_0, x_1]f}{x_2 - x_0} = \frac{-1 - 1}{2 - 0} = -1
 \end{aligned}$$

Als Interpolant ergibt sich damit:

$$\begin{aligned}
 p(x) &= [x_0]f + [x_0, x_1]f \cdot (x - x_0) + [x_0, x_1, x_2]f \cdot (x - x_0) \cdot (x - x_1) \\
 &= \boxed{0} + \boxed{1}(x - 0) + (\boxed{-1})(x - 0)(x - 1) \\
 &= 0 + 1 \cdot (x - 0) - 1 \cdot (x - 0)(x - 1) \\
 &= x - x(x - 1) = x(2 - x) = 2x - x^2
 \end{aligned}$$

b) Erweiterung des Dreiecksschemas:

x_i	$i \setminus k$	0	1	2	3
0	0	$y_0 = 0$	\rightarrow 1	\rightarrow -1	\rightarrow $-\frac{3}{20}$
			\nearrow	\nearrow	\nearrow
1	1	$y_1 = 1$	\rightarrow -1	\rightarrow $-\frac{21}{20}$	
			\nearrow	\nearrow	
2	2	$y_2 = 0$	\rightarrow $-\frac{3}{10}$		
			\nearrow		
$\frac{1}{3}$	3	$y_3 = \frac{1}{2}$			

Die Berechnungen im Einzelnen:

$$\begin{aligned}
 [x_3]f &= \sin\left(\frac{\pi}{6}\right) = \frac{1}{2} \\
 [x_2, x_3]f &= \frac{[x_3]f - [x_2]f}{x_3 - x_2} = \frac{\frac{1}{2} - 0}{\frac{1}{3} - 2} = -\frac{3}{10} \\
 [x_1, x_2, x_3]f &= \frac{[x_2, x_3]f - [x_1, x_2]f}{x_3 - x_1} = \frac{-\frac{3}{10} + 1}{\frac{1}{3} - 1} = -\frac{21}{20} \\
 [x_0, x_1, x_2, x_3]f &= \frac{[x_1, x_2, x_3]f - [x_0, x_1, x_2]f}{x_3 - x_0} = \frac{-\frac{21}{20} + 1}{\frac{1}{3} - 0} = \boxed{-\frac{3}{20}}
 \end{aligned}$$

Als neuer Interpolant ergibt sich damit:

$$p(x) = x(2-x) - \frac{3}{20} \cdot x(x-1)(x-2)$$

c)

$$\begin{aligned} p_3(u) &= p_1(u) \cdot \frac{u-u_4}{u_0-u_4} + p_2(u) \cdot \frac{u-u_0}{u_4-u_0} \\ &= p_1(u) + \frac{u-u_0}{u_4-u_0} \cdot (p_2(u) - p_1(u)) \end{aligned}$$

d) Spline-Interpolation

- Interpolation mit lokalen Polynomen niedrigen Grades, dadurch keine starken Oszillation an den Rändern auf Grund eines hohen Polynomgrads
- geringere Komplexität der Berechnungen ($\mathcal{O}(n)$ statt $\mathcal{O}(n^2)$), wobei in diesem Fall $n = 100$ gilt)

e) Der erste Fehler befindet sich in Zeile 07: Dort muss es in der letzten Klammer heißen:

```
/(xs[i+k]-xs[i]);
```

Der zweite Fehler steckt in Zeile 09: Die Rückgabe muss im ersten Element des f-Feldes erfolgen:

```
09 return f[0];
```

f) Die Methode `foo()` berechnet die zweidimensionale Polynominterpolation nach Aitken-Neville und testet, ob die Stützwerte passende Dimensionen haben (in x- und y-Richtung gleich viele).