

Numerisches Programmieren

4. Programmieraufgabe: Polynominterpolation

Das Schema von Aitken und Neville

Das Schema von Aitken und Neville ist eine effiziente Methode, um einzelne Werte eines Interpolationspolynoms zu berechnen ohne das Polynom explizit bestimmen zu müssen. Zugrunde liegt die Rekursionsgleichung

$$p_{i,\dots,i+l}(x) = \frac{(x - x_i)p_{i+1,\dots,i+l}(x) - (x - x_{i+l})p_{i,\dots,i+l-1}(x)}{x_{i+l} - x_i} \quad (1)$$

wobei $p_{i,\dots,i+l}$ das Interpolationspolynom zu den Stützstellen x_i, \dots, x_{i+l} bezeichnet.

Die Polynome $p_{j,j}$, die nur in einer einzigen Stützstelle x_j interpolieren sind konstant und es gilt $p_{j,j}(x) = f_j (\forall x)$. Davon ausgehend lassen sich die $p_{i,i+l}$ in folgender Reihenfolge berechnen:

$$\begin{array}{ccccccc}
 f_0 = p_{0,0}(x) & \rightarrow & p_{0,1}(x) & \rightarrow & p_{0,2}(x) & \rightarrow & \dots & \rightarrow & p_{0,n}(x) \\
 & & \nearrow & & \nearrow & & \nearrow & & \nearrow \\
 f_1 = p_{1,1}(x) & \rightarrow & p_{1,2}(x) & \rightarrow & p_{1,3}(x) & \rightarrow & \dots & & \\
 & & \nearrow & & \nearrow & & & & \\
 f_2 = p_{2,2}(x) & \rightarrow & p_{2,3}(x) & & \dots & & & & \\
 & & \nearrow & & \nearrow & & & & \\
 & & \vdots & & \dots & & & & \\
 f_n = p_{n,n}(x) & & & & & & & &
 \end{array} \quad (2)$$

2D-Interpolation

Analog zum Vorgehen in 1D lassen sich auch zweidimensionale Interpolationspolynome konstruieren. Die Aufgabenstellung ist hierbei zu vorgegebenen Stützstellen (x_i, y_j) und zugehörigen Werten f_{ij} $i, j = 1 \dots n$ (s. Abb. 1) ein Polynom p vom Grad n in x und y zu finden, sodass $p(x_i, y_j) = f_{ij} \forall i, j$ (Abb. 4).

Die Auswertung des 2D-Interpolationspolynoms p an einer Stelle (x^*, y^*) lässt sich durch „Festhalten“ einer Koordinate auf den eindimensionalen Fall zurückführen. Wir halten zunächst die x -Koordinate fest und definieren die Hilfspolynome

$$\tilde{p}_i(y) := p(x_i, y)$$

Die \tilde{p}_i sind (eindimensionale) Polynome n -ten Grades in y und somit durch die Interpolationsbedingung $\tilde{p}_i(y_j) = p(x_i, y_j) = f_{ij} \forall j$ eindeutig festgelegt (Abb. 2). Insbesondere lässt sich somit durch 1D-Interpolation $p(x_i, y^*) = \tilde{p}_i(y^*)$ bestimmen. Als nächstes halten wir nun die x -Koordinate fest und definieren

$$\hat{p}(x) := p(x, y^*)$$

\hat{p} ist ein Polynom n -ten Grades in x und durch die Interpolationsbedingung $\hat{p}(x_i) = p(x_i, y^*) = \tilde{p}_i(y^*) \forall i$ festgelegt (Abb. 3). Auswerten von \hat{p} an der Stelle x^* liefert also schließlich das gesuchte $p(x^*, y^*) = \hat{p}(x^*)$.

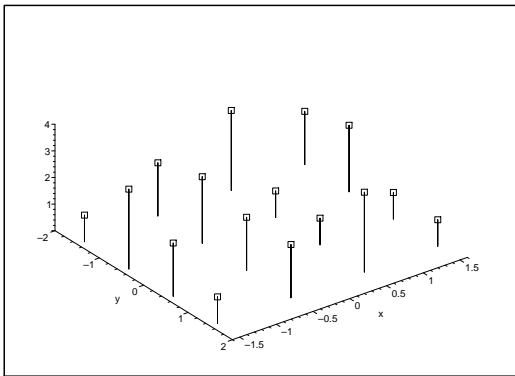


Abb. 1: Stützpunkte

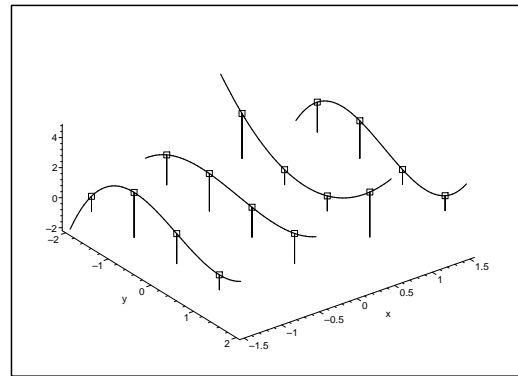


Abb. 2: Die Polynome \tilde{p}_i

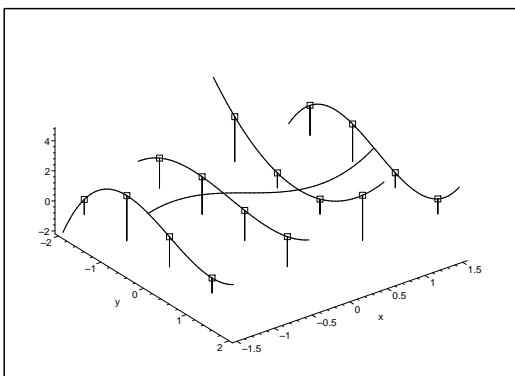


Abb. 3: \tilde{p}_i und \hat{p} für $y^* = 0$

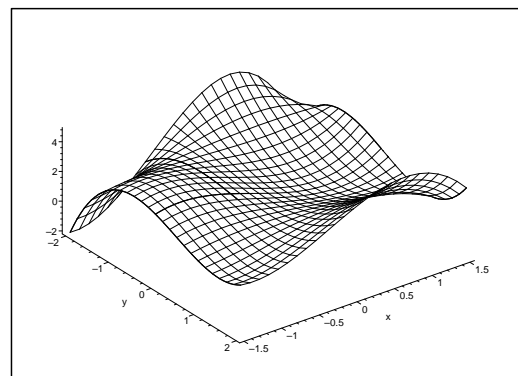


Abb. 4: vollst. Interpolationspolynom

Bildinterpolation

Wir können ein Graustufenbild als eine Funktion $f : [a, b] \times [c, d] \rightarrow [0, 1]$ auffassen, wobei $f(x, y)$ den Helligkeitswert an der Stelle (x, y) angibt ($0 \hat{=}$ schwarz, $1 \hat{=}$ weiß). Typischerweise steht uns das Bild jedoch nur in diskretisierter Form zur Verfügung, d.h. es wurde bereits in $n \times m$ (gleich große, einfarbige) Rechtecke bzw. Pixel unterteilt und wir kennen die Farbwerte der Pixel. Wir gehen im Folgenden davon aus, dass die Farbe eines Pixels dem Funktionswert in seinem Mittelpunkt entspricht. Außerdem nehmen wir an, dass die gegebenen Pixel Seitenlänge 1 und ihre Mittelpunkte ganzzahlige Koordinaten beginnend bei $(1, 1)$ haben. Somit ist $f : [0.5, n + 0.5] \times [0.5, m + 0.5] \rightarrow [0, 1]$ und wir kennen $f(i, j) \forall i = 1 \dots n, j = 1 \dots m$. Abb. 5 zeigt die Pixel und ihre Mittelpunkte.

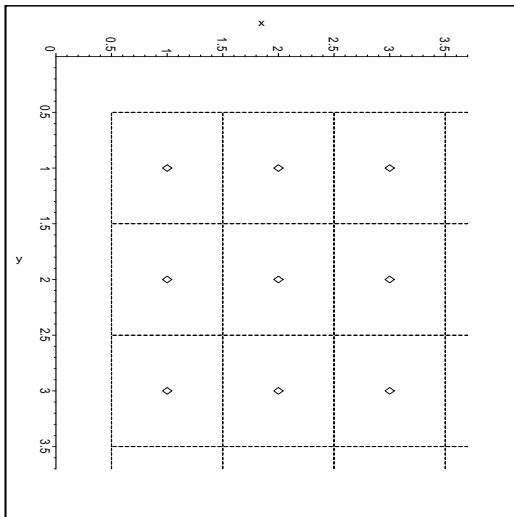


Abb. 5: altes Pixelraster

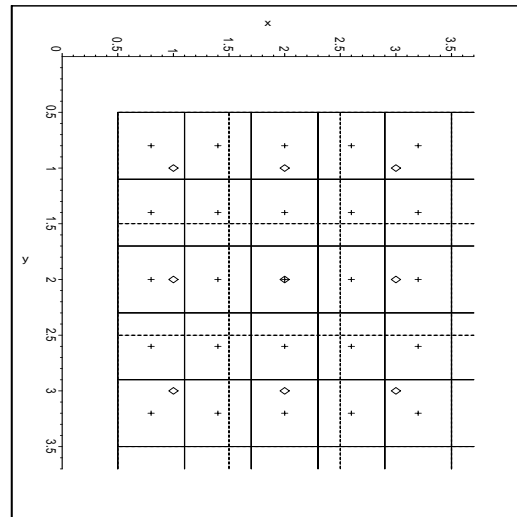


Abb. 6: altes und neues Pixelraster

Wollen wir das Bild nun mit einer höheren Auflösung von $\tilde{n} \times \tilde{m}$ Pixeln darstellen, so unterteilen wir $[0.5, n + 0.5] \times [0.5, m + 0.5]$ in ein neues Raster aus $\tilde{n} \times \tilde{m}$ Rechtecken (s. Abb. 6). Die Farben der neuen Pixel sind die Werte von f in den Mittelpunkten der neuen Pixel. Um sie zu erhalten müssen wir f mit Hilfe der Werte aus dem alten Raster interpolieren.

Bikubische Interpolation

Als Stützstellen werden diejenigen Pixel des Originalbildes gewählt, die dem Pixel des neuen Bildes am nächsten liegen (s. Abb. 7). Um auf jeder Seite des gesuchten Pixels gleich viele Stützstellen zu haben wählt man deren Zahl meist gerade. In

dieser Programmieraufgabe implementieren wir die *bikubische* Interpolation mit 4×4 Stützstellen.

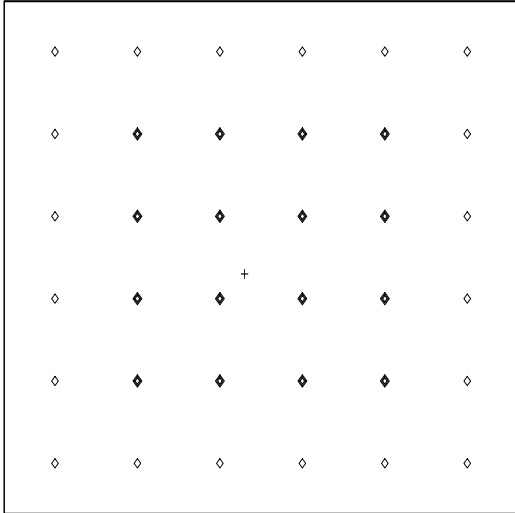


Abb. 7: Bsp. für Stützstellen bei bikubischer Interpolation

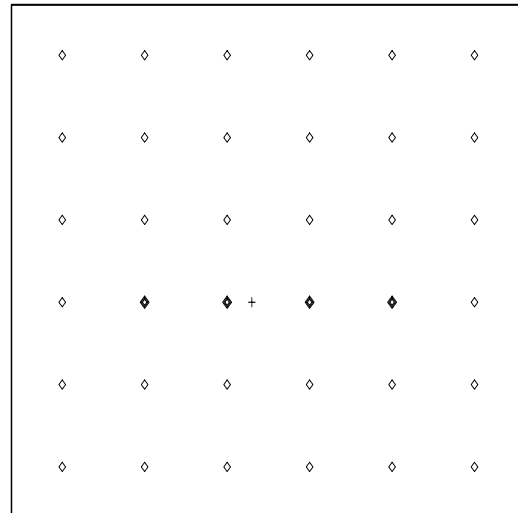


Abb. 8: Hier genügt 1D-Interpolation

Sonderfälle

Manchmal fallen die zu interpolierenden Punkte mit den Pixeln des alten Rasters zusammen, wodurch sich die Interpolation erübrigt. Falls nur eine Koordinate mit dem alten Raster zusammenfällt, sind die Stützwerte von \hat{p} sofort bekannt und es genügt 1D-Interpolation anzuwenden (s. Abb. 8).

Aufgaben

Im Folgenden werden die zu implementierenden Methoden aufgelistet. Details finden Sie jeweils in den Kommentaren zu den einzelnen Methoden. Überlegen Sie sich zu allen Methoden Testbeispiele und testen Sie unbedingt jede einzelne Methode direkt nach der Erstellung. Sie können zusätzlich das zur Verfügung gestellte Testprogramm verwenden. Es ersetzt aber nicht das Verwenden eigener Testbeispiele.

- Klasse *Aitken*: Methoden *aitken* und *aitken2d*
- Klasse *Interpolation*: Methoden *transformGrid* und *bicubic*
Für die Methode *transformGrid* müssen Sie zunächst die Funktion

$$g : \{1 \dots \tilde{n}\} \times \{1 \dots \tilde{m}\} \rightarrow [0.5, n + 0.5] \times [0.5, m + 0.5]$$

bestimmen, die zu einem Pixel in Koordinaten des neuen Rasters die Position in den alten Koordinaten berechnet (“+”-Zeichen in Abb. 6).

Formalien und Hinweise

- Das Programmgerüst erhalten Sie auf den Webseiten zur Vorlesung.
- Ergänzen Sie das Programmgerüst bitte **nur an den dafür vorgegebenen Stellen!** Falls Sie die Struktur der Programme an anderer Stelle verändern, können wir sie evtl. nicht mehr testen.
- Beseitigen Sie vor Abgabe Ihres Programms alle Ausgaben an die Konsole und reichen sie bis zum **05. Juli 2010, 12:00 Uhr** über das Web-Portal (siehe Homepage) ein.
- Wir empfehlen, die Programme unter Linux (Rechnerhalle) zu testen.