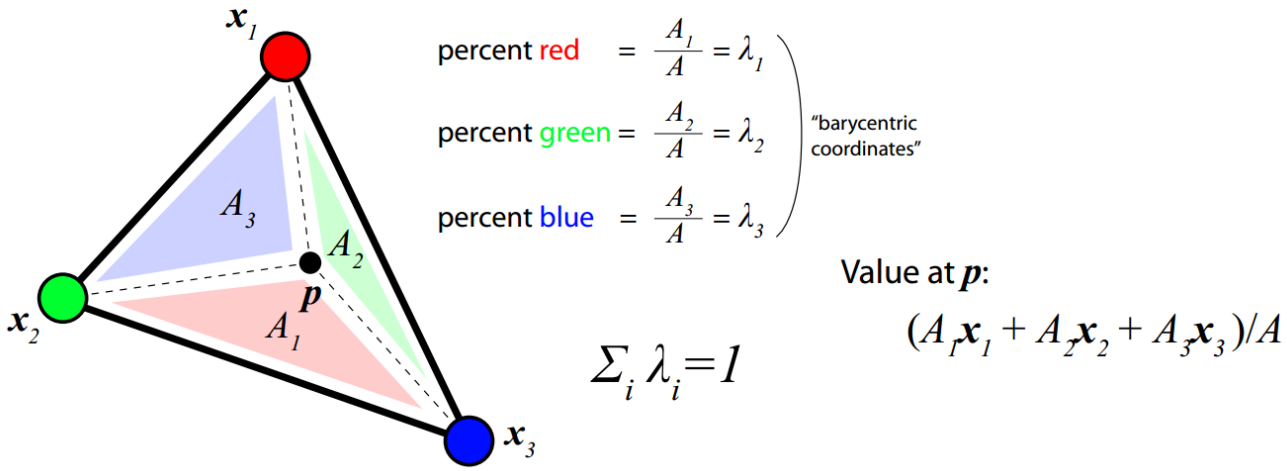
Lineare Interpolation auf Dreiecken

Gegeben seien die Datentupel $(x_i, y_i), (x_j, y_j), (x_k, y_k)$ und sei x ein Punkt innerhalb eines Dreiecks $\mathcal{C}(x_i, x_j, x_k)$. Punkt x lässt sich als Linearkombination

$$x = \lambda_i x_i + \lambda_j x_j + \lambda_k x_k$$

ausdrücken, gleiches gilt für den unbekannten Messwert am Punkt x:

$$f(x) = \lambda_i y_i + \lambda_j y_j + \lambda_k y_k$$



Lineare Interpolation auf Dreiecken

Gegeben seien die Datentupel $(x_i, y_i), (x_j, y_j), (x_k, y_k)$ und sei x ein Punkt innerhalb eines Dreiecks $\mathcal{C}(x_i, x_j, x_k)$. Punkt x lässt sich als Linearkombination

$$x = \lambda_i x_i + \lambda_j x_j + \lambda_k x_k$$

ausdrücken, gleiches gilt für den unbekannten Messwert am Punkt x:

$$f(x) = \lambda_i y_i + \lambda_j y_j + \lambda_k y_k$$

Die Koeffizienten / Gewichte werden als **baryzentrische Koodinaten** von x (in 2D). Seien die Koordinaten von $x = (x^{(1)}, x^{(2)})$, dann können die Koeffizienten wie folgt ausgedrückt werden

$$\lambda_i = \frac{\det(A_i)}{\det(A)}, \qquad \lambda_j = \frac{\det(A_j)}{\det(A)}, \qquad \lambda_k = \frac{\det(A_k)}{\det(A)}$$

mit den Matrizen

$$A = \begin{pmatrix} x_i^{(1)} & x_j^{(1)} & x_k^{(1)} \\ x_i^{(2)} & x_j^{(2)} & x_k^{(2)} \\ 1 & 1 & 1 \end{pmatrix}, A_i = \begin{pmatrix} x^{(1)} & x_j^{(1)} & x_k^{(1)} \\ x^{(2)} & x_j^{(2)} & x_k^{(2)} \\ 1 & 1 & 1 \end{pmatrix}, A_j = \begin{pmatrix} x_i^{(1)} & x^{(1)} & x_k^{(1)} \\ x_i^{(2)} & x^{(2)} & x_k^{(2)} \\ 1 & 1 & 1 \end{pmatrix}, A_k = \begin{pmatrix} x_i^{(1)} & x_j^{(1)} & x^{(1)} \\ x_i^{(2)} & x_j^{(2)} & x^{(2)} \\ 1 & 1 & 1 \end{pmatrix}$$

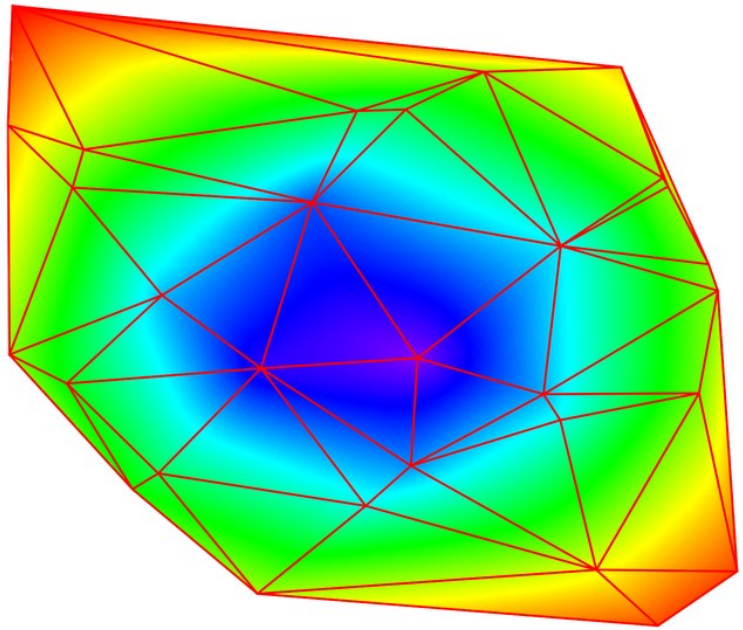
Lineare Interpolation auf Dreiecken

Gegeben seien die Datentupel $(x_i, y_i), (x_j, y_j), (x_k, y_k)$ und sei x ein Punkt innerhalb eines Dreiecks $\mathcal{C}(x_i, x_j, x_k)$. Punkt x lässt sich als Linearkombination

$$x = \lambda_i x_i + \lambda_j x_j + \lambda_k x_k$$

ausdrücken, gleiches gilt für den unbekannten Messwert am Punkt x:

$$f(x) = \lambda_i y_i + \lambda_j y_j + \lambda_k y_k$$



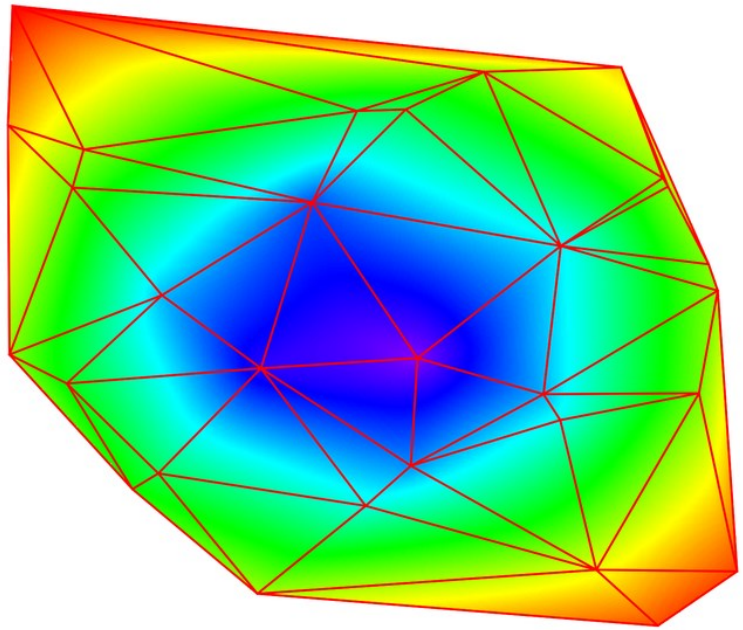
Lineare Interpolation auf Dreiecken

Gegeben seien die Datentupel $(x_i, y_i), (x_j, y_j), (x_k, y_k)$ und sei x ein Punkt innerhalb eines Dreiecks $\mathcal{C}(x_i, x_j, x_k)$. Punkt x lässt sich als Linearkombination

$$x = \lambda_i x_i + \lambda_j x_j + \lambda_k x_k$$

ausdrücken, gleiches gilt für den unbekannten Messwert am Punkt x:

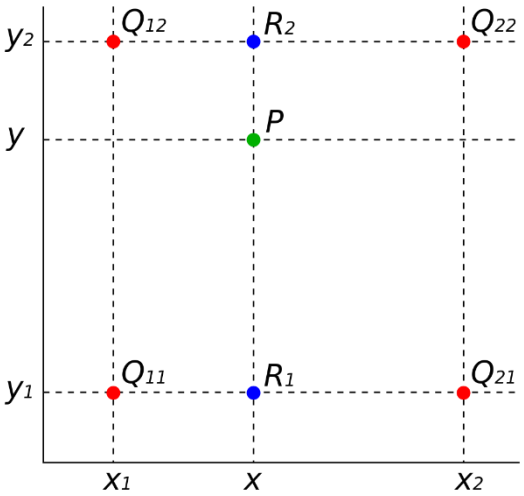
$$f(x) = \lambda_i y_i + \lambda_j y_j + \lambda_k y_k$$



Eigenschaften:

- $f(x)$ ist kontinuierlich
- Auf Dreieckskanten nicht stetig differenzierbar
- lokaler Interpolator der nur $(x_i y_i), (x_j, y_j), (x_k, y_k)$ berücksichtigt, wenn $x \in \mathcal{C}(x_i, x_j, x_k)$
- Extrapolation ist konzeptuell **NICHT** möglich

Bilineare Interpolation auf einem Punktgitter



Seien die Messwerte $y_{i,j}$ auf einem Punktgitter $Q_{i,j} \in \mathbb{R}^2$ gegeben. Der unbekannte Wert an einem Punkt $P = (x^{(1)}, y^{(2)})$ innerhalb eines Gitterelements $Q_{i,j}, Q_{i+1,j}, Q_{i,j+1}, Q_{i+1,j+1}$ kann wie folgt vorhergesagt werden:

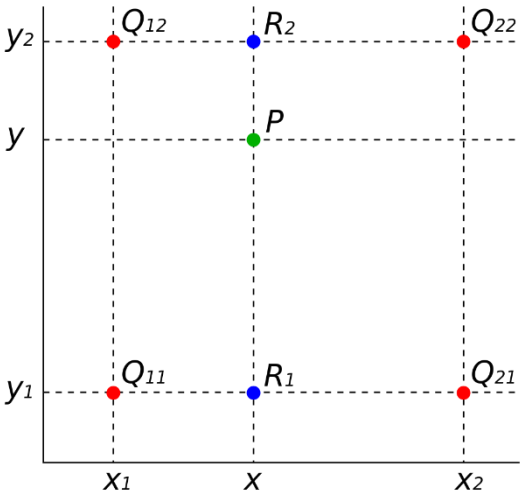
$$f\left(x^{(1)}, x^{(2)}\right) = \frac{x_{i,j+1}^{(2)} - x^{(2)}}{x_{i,j+1}^{(2)} - x_{i,j}^{(2)}} f_{i,j} + \frac{x^{(2)} - x_{i,j}^{(2)}}{x_{i,j+1}^{(2)} - x_{i,j}^{(2)}} f_{i,j+1}$$

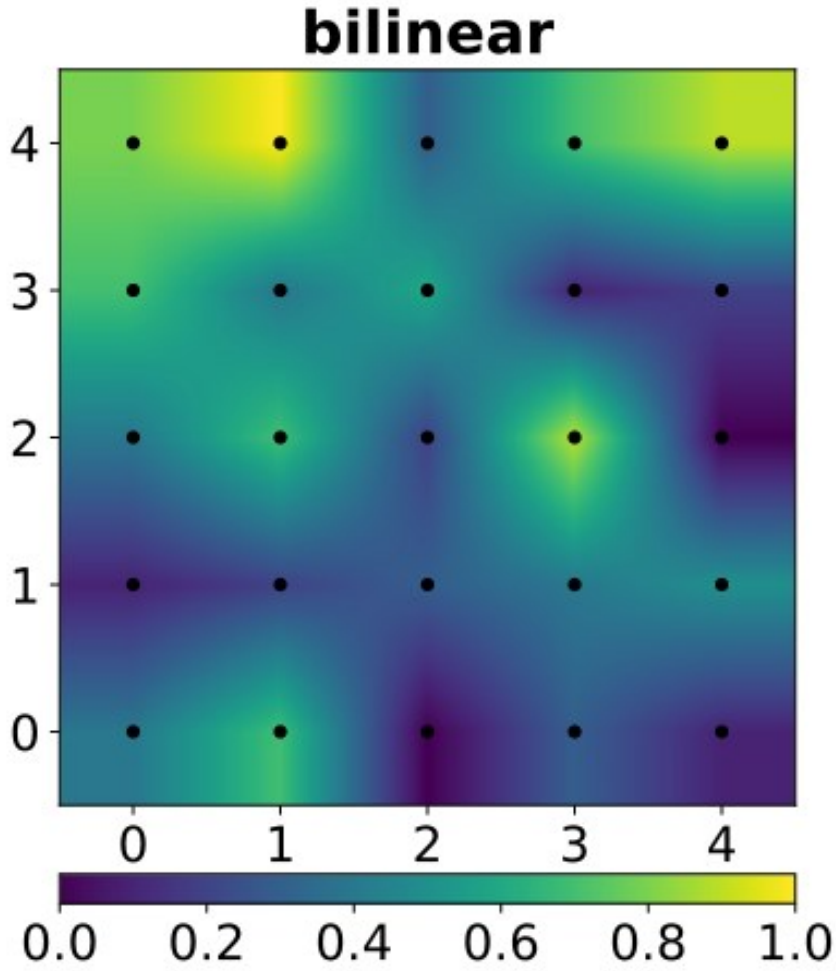
mit

$$f_{i,j} = \frac{x_{i+1,j}^{(1)} - x^{(1)}}{x_{i+1,j}^{(1)} - x_{i,j}^{(1)}} y_{i,j} + \frac{x^{(1)} - x_{i,j}^{(1)}}{x_{i+1,j}^{(1)} - x_{i,j}^{(1)}} y_{i+1,j}$$

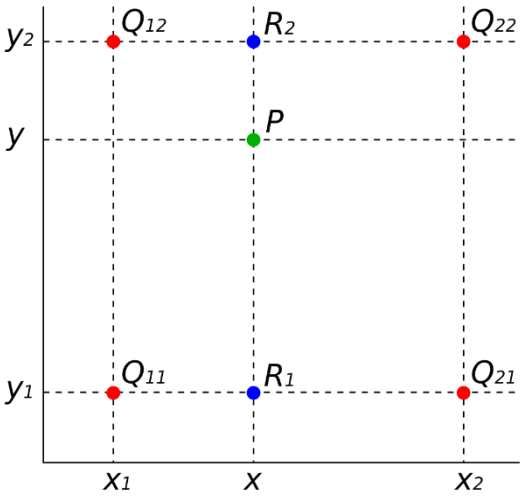
$$f_{i,j+1} = \frac{x_{i+1,j}^{(1)} - x^{(1)}}{x_{i+1,j}^{(1)} - x_{i,j}^{(1)}} y_{i,j+1} + \frac{x^{(1)} - x_{i,j}^{(1)}}{x_{i+1,j}^{(1)} - x_{i,j}^{(1)}} y_{i+1,j+1}$$

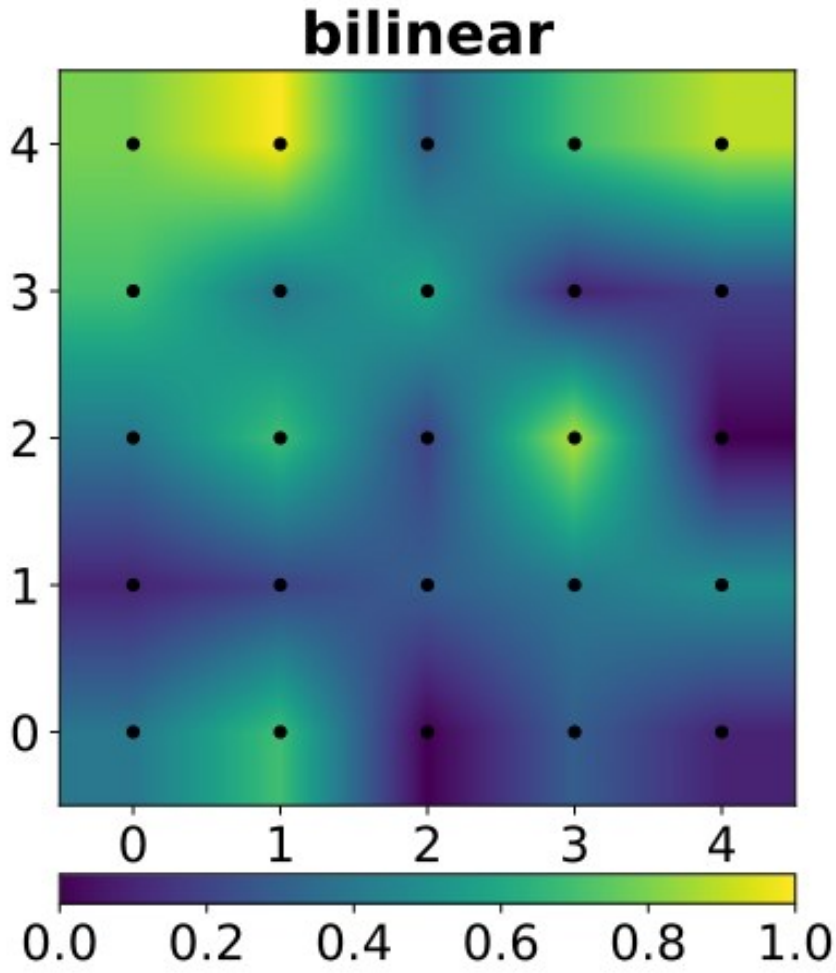
Bilinear Interpolation





Bilinear Interpolation

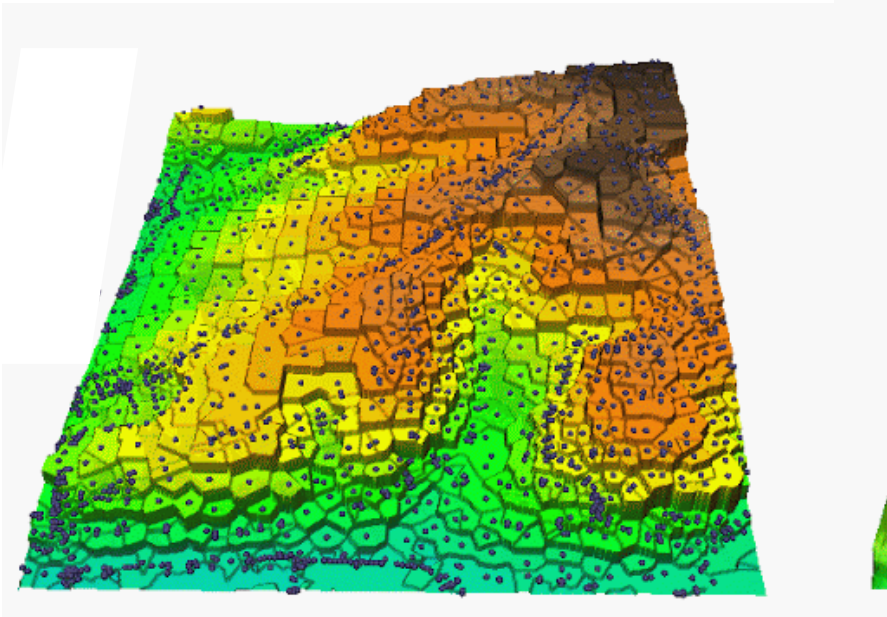


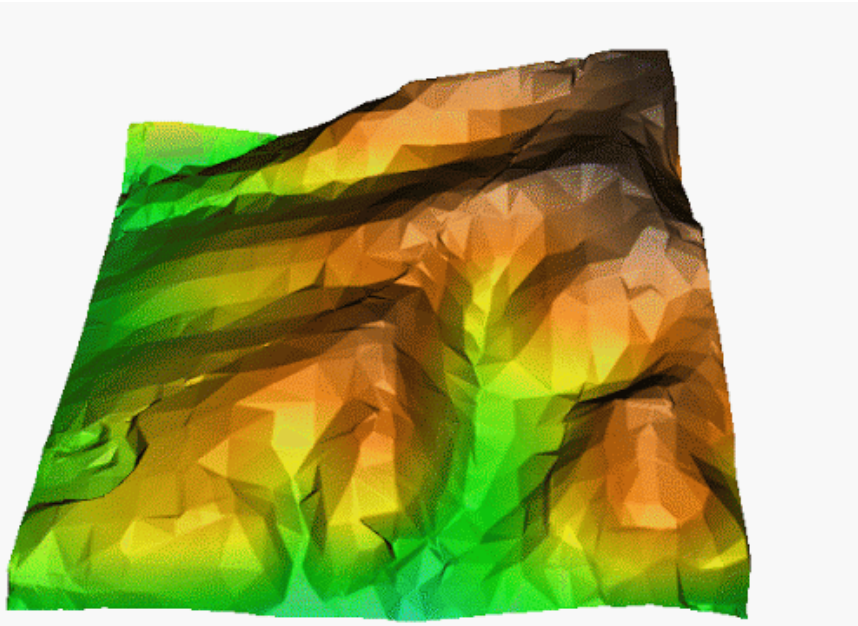


Eigenschaften:

- $f(x)$ ist kontinuierlich
- auf Gitterkanten nicht differenzierbar
- Lokaler Interpolator, der sich nur auf
  $(x_{i,j}y_{i,j}), (x_{i+1,j}, y_{i+1,j}),$
  $(x_{i,j+1}, y_{i,j+1}), (x_{i+1,j+1}, y_{i+1,j+1})$
  bezieht, wenn x innerhalb dieses Gitterelements ist
- Extrapolation ist konzeptionell nicht möglich

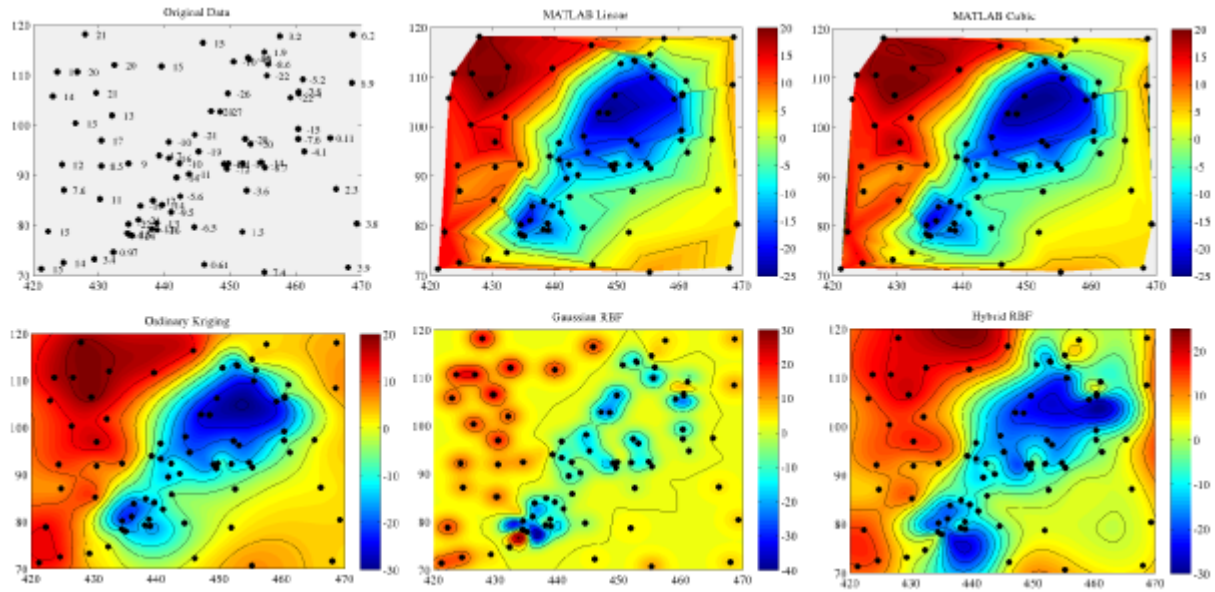
Nearest Neighbour vs Lin. Interp. auf Dreiecken





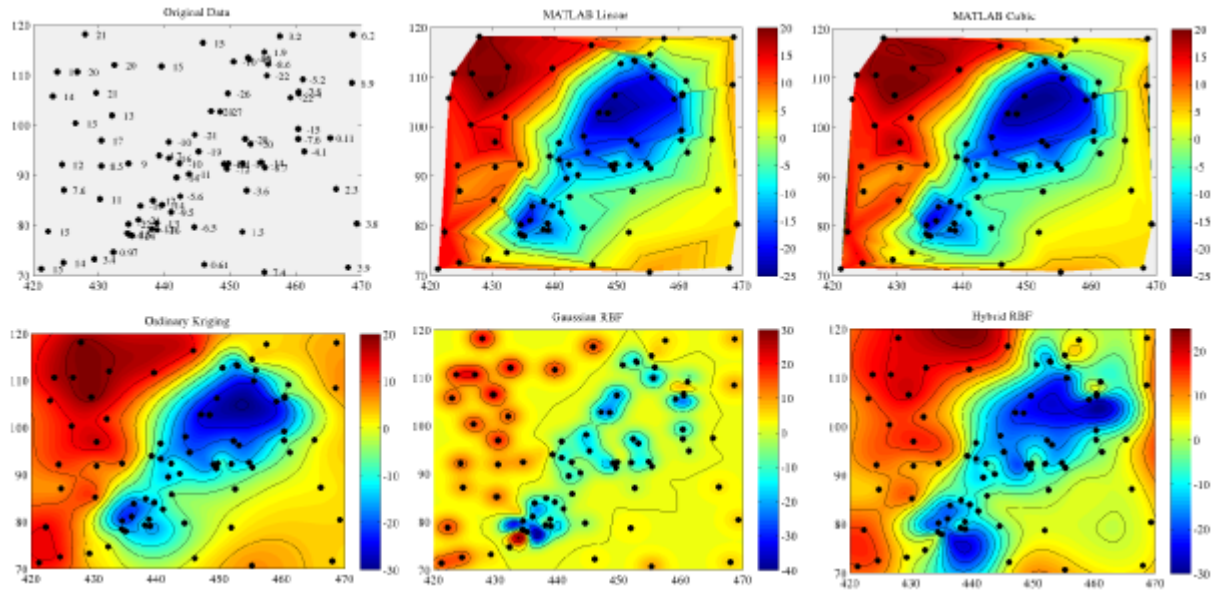
Gitter-freie Interpolation

- Inverse distance weighting (IDW)
- Polynom Interpolation
- Splines und Radial Basis Functions



Gitter-freie Interpolation

- Inverse distance weighting (IDW)
- ~~Polynom Interpolation~~ Polynom Approximation
- Splines und Radial Basis Functions



Inverse Distance Weighting (IDW)

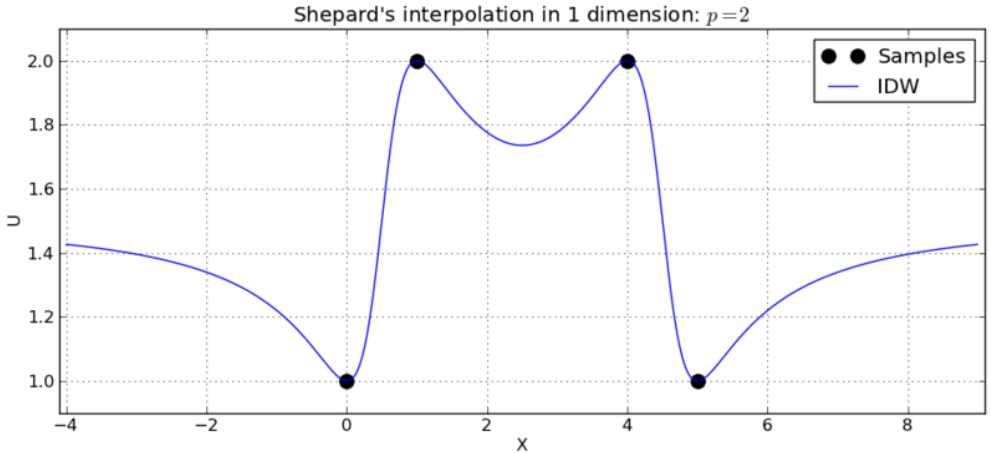
Annahme: Der Wert an einem beliebigen Punkt hängt nur vom Abstand zu den Datenpunkten ab.

Gegeben seien Tupel $(x_i, y_i); i = 1, \dots, N; x_i \in \mathbb{R}^n$. Der unbekannte Wert $f(x)$ an einem beliebigen Punkt $x \in \mathbb{R}^n$ lässt sich vorhersagen mit:

$$f(x) = \frac{\sum_{i=1}^{N} \lambda_i(x) y_i}{\sum_{i=1}^{N} \lambda_i(x)}$$

Mit

$$\lambda_i(x) = \frac{1}{|x - x_i|^p}.$$



Inverse Distance Weighting (IDW)

Annahme: Der Wert an einem beliebigen Punkt hängt nur vom Abstand zu den Datenpunkten ab.

Gegeben seien Tupel $(x_i, y_i); i = 1, \dots, N; x_i \in \mathbb{R}^n$. Der unbekannte Wert $f(x)$ an einem beliebigen Punkt $x \in \mathbb{R}^n$ lässt sich vorhersagen mit:

$$f(x) = \frac{\sum_{i=1}^{N} \lambda_i(x) y_i}{\sum_{i=1}^{N} \lambda_i(x)}$$

Mit

$$\lambda_i(x) = \frac{1}{|x - x_i|^p}.$$

Oder alternativ:

$$f(x) = \sum_{i=1}^{N} \lambda_i(x) y_i \text{ mit } \lambda_i(x) = \frac{\frac{1}{|x-x_i|^p}}{\sum_{j=1}^{N} \frac{1}{|x-x_j|^p}} = \frac{1}{1 + \sum_{j=1, j\neq i}^{N} \frac{|x-x_i|^p}{|x-x_j|^p}}$$

zum Vermeiden des globalen Vorfaktoren …

Inverse Distance Weighting (IDW)

$$f(x) = \sum_{i=1}^{N} \lambda_i(x) y_i \text{ mit } \lambda_i(x) = \frac{\frac{1}{|x-x_i|^p}}{\sum_{j=1}^{N} \frac{1}{|x-x_j|^p}} = \frac{1}{1+\sum_{j=1, j\neq i}^{N} \frac{|x-x_i|^p}{|x-x_j|^p}}$$

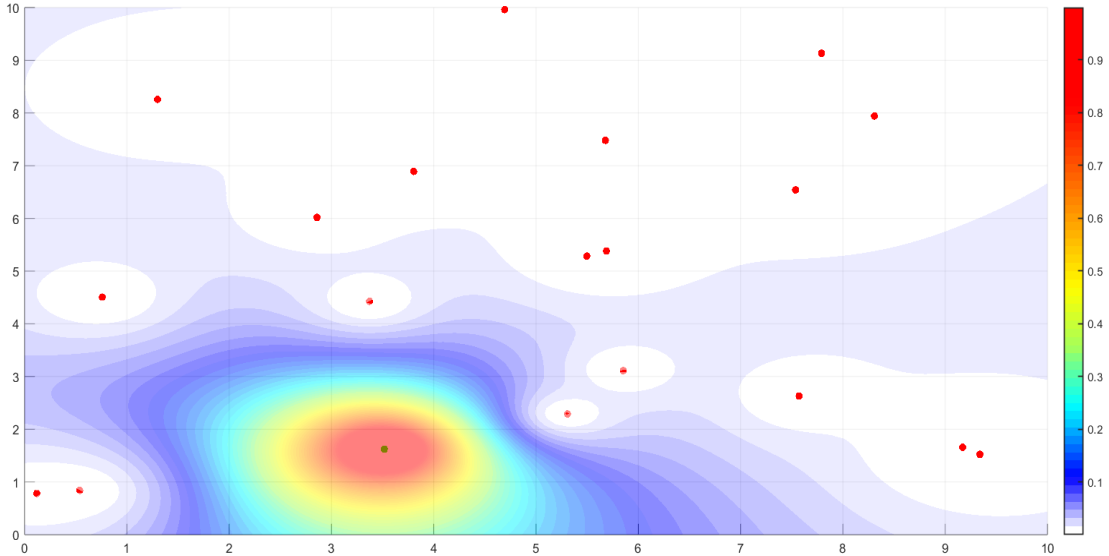


Illustration of weight function λ_1 for various $x \in [0,1]^2$ with parameter $k = 2$.

Inverse Distance Weighting (IDW)

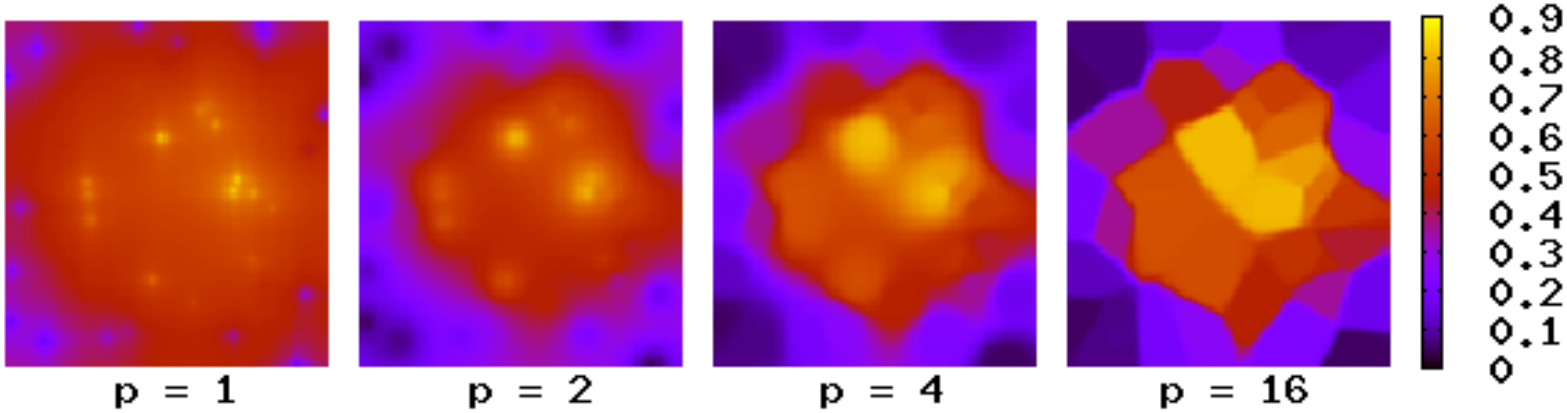
Gegeben seien Tupel $(x_i, y_i); i = 1, \dots, N; x_i \in \mathbb{R}^n$. Der unbekannte Wert $f(x)$ an einem beliebigen Punkt $x \in \mathbb{R}^n$ lässt sich vorhersagen mit:

$$f(x) = \frac{\sum_{i=1}^{N} \lambda_i(x) y_i}{\sum_{i=1}^{N} \lambda_i(x)}$$

Mit

$$\lambda_i(x) = \frac{1}{|x - x_i|^{\boldsymbol{p}}}.$$

Einfluss des Exponenten p als Lokalisierungsparameter:



Inverse Distance Weighting (IDW)

Gegeben seien Tupel $(x_i, y_i); i = 1, \dots, N; x_i \in \mathbb{R}^n$. Der unbekannte Wert $f(x)$ an einem beliebigen Punkt $x \in \mathbb{R}^n$ lässt sich vorhersagen mit:

$$f(x) = \frac{\sum_{i=1}^{N} \lambda_i(x) y_i}{\sum_{i=1}^{N} \lambda_i(x)}$$

Mit

$$\lambda_i(x) = \frac{1}{|x - x_i|^{\boldsymbol{p}}}.$$

Eigenschaften:

- $f(x)$ ist kontinuierlich;
- Globaler Interpolator, welcher alle Daten berücksichtigt;
- Gewichte hängen rein Punktabstand zw. Interpolations- und Datenpunkten ab;

- Extrapolation ist konzeptionell möglich, aber für

$$|x - x_i| \to \infty \; \forall i \Rightarrow f(x) \to \frac{\sum_{i=1}^{N} y_i}{N}$$

Polynom Interpolation

Annahme: Der unbekannte Wert an einem beliebigen Punkt lässt sich über eine Polynomfunktion $f(x) = \sum_{i=0}^{m} \lambda_i x^i$ vorhersagen.

Gegeben seien Tupel $(x_i, y_i); i = 1, \dots, N; x_i \in \mathbb{R}^n$. Sei $f(x) = \sum_{i=0}^{m} \lambda_i x^i$ ein Polynom vom Grad $m = N - 1$, dann kann man die Koeffizienten $\lambda_0, \dots, \lambda_{N-1} \in \mathbb{R}$ so wählen, dass

$$f(x_i) = y_i.$$

Um die Koeffizienten $\lambda_0, \dots, \lambda_m$ zu finden, muss folgendes Gleichungssystem gelöst werden:

$$\mathbf{M}\boldsymbol{\lambda} = \mathbf{y}$$

mit $\boldsymbol{\lambda} = (\lambda_0, \dots, \lambda_m)^{\mathrm{T}}$, $\mathbf{y} = (y_1, \dots, y_N)$ und

$$\mathbf{M} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_m^2 & \cdots & x_N^m \end{pmatrix}$$

Polynom Interpolation

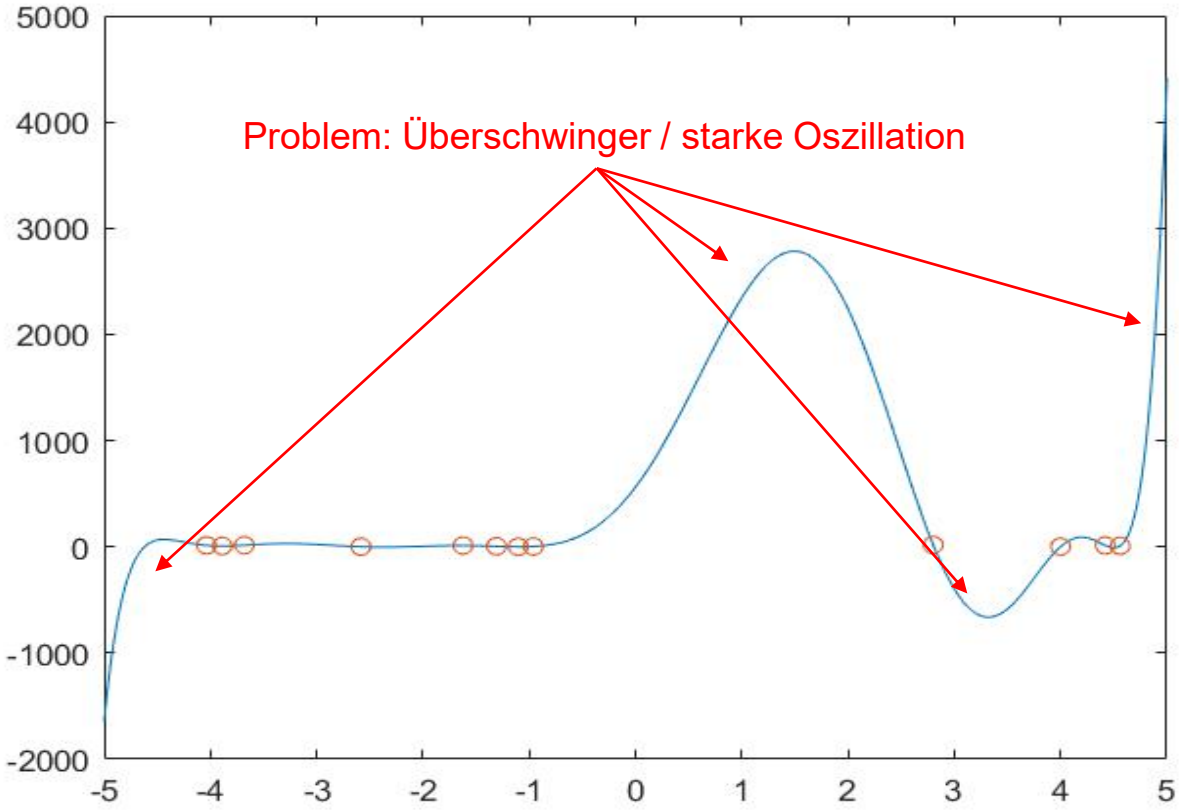
Annah
Polyn

Gege
Grad

Um d

en:

mit λ



Interpolationspolynom für $N = 12$ Datenpunkte und Grad $m = N - 1 = 11$.

Polynom Approximation

Annahme: Der unbekannte Wert an einem beliebigen Punkt lässt sich über eine Polynomfunktion $f(x) = \sum_{i=0}^{m} \lambda_i x^i$ vorhersagen.

Gegeben seien Tupel $(x_i, y_i); i = 1, \dots, N; x_i \in \mathbb{R}^n$ und sei $f(x) = \sum_{i=0}^{m} \lambda_i x^i$ ein Polynom vom Grad m. Die Oszillation des Polynoms außerhalb der Daten kann dadurch reduziert werden, wenn der Grad $m \ll N - 1$ gewählt wird. Die Koeffizienten können nur noch so gewählt werden, dass sie

$$\min_{\lambda_0,\dots,\lambda_m} \sum_{i=1}^{N} |f(x_i) - y_i|^2$$

minimieren. Dieses Verfahren wird als **Approximation** (nicht-exakte Interpolation) bezeichnet, weil im Allgemeinen $f(x_i) = y_i \; \forall i$ **NICHT** erreicht werden kann.
Es gilt nur noch

$$f(x_i) \approx y_i \; \forall i.$$

Eigenschaften:

- $f(x)$ ist ein Polynom vom Grad m (d.h. kontinuierlich und m-mal stetig differenzierbar)
- globaler Interpolator der **alle** (x_i, y_i) mit $1 \leq i \leq N$ berücksichtigt
- für $m \ll N - 1$: Approximations-Polynom $f(x_i) \approx y_i$ mit $1 \leq i \leq N$
- Extrapolation ist konzeptionell möglich