

Visualisierung

Vorlesung im Wintersemester 2016/17
3) Interpolation und Filterung
Teil 2

Dozent: Prof. Dr. Boris Neubert

Folien: Prof. Dr. Carsten Dachsbacher

Lehrstuhl für Computergrafik

Karlsruher Institut für Technologie

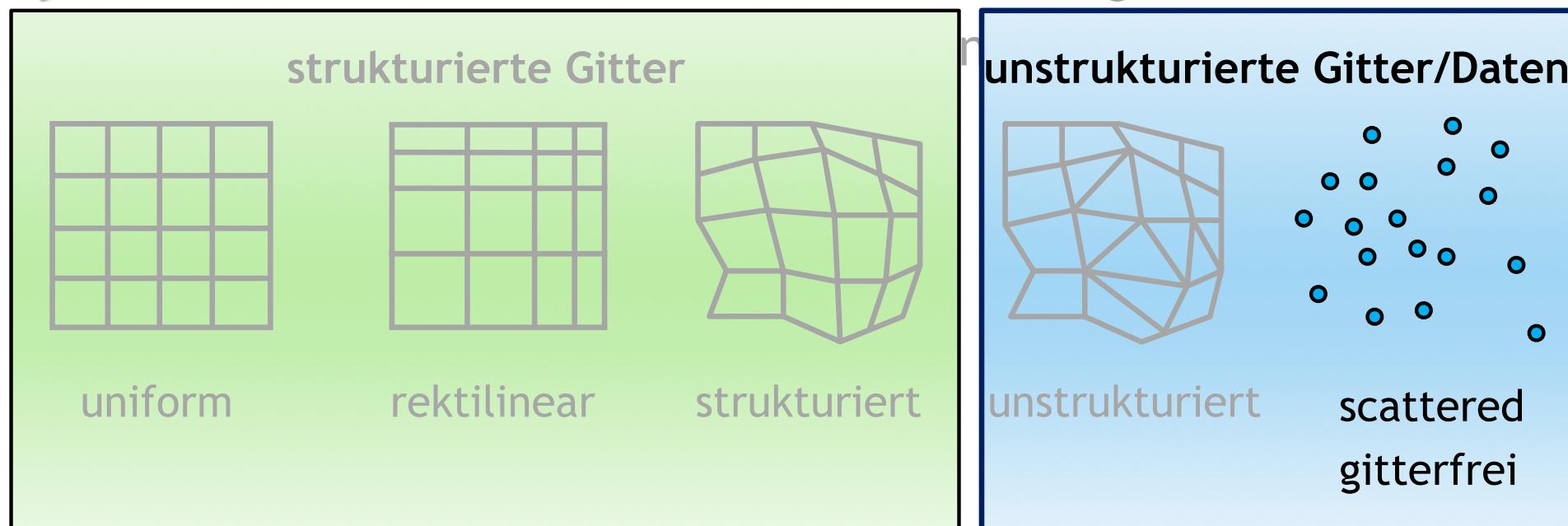


Wiederholung

- lokale / globale Interpolationsmethoden
- Simplex
- Voronoi/ Delaunay
- Triangulationsalgorithmen:
Edge Flip, Plane Sweep, Bowyer Watson
- Interpolation gitterfreier Daten:
Inverser Distance - Shepard / Radial Basis Functions

Gitterfreie Daten und Gittertypen

- gitterfreie Daten: irregulär verteilte Positionen, keine Konnektivität
- Konnektivität herstellen und dann (in Simplizes) interpolieren, oder
- **direkte Interpolation anhand aller Datenpunkte**
- Gitter unterscheiden sich in
- Art und Form der Zellen aus denen sie aufgebaut sind



Radiale Basis Funktionen (RBF)

- ▶ ... sind ein ähnliches Konzept: eine Funktion wird wieder durch eine gewichtete Summe dargestellt, aber: es gibt nur eine Basisfunktion
- ▶ diese Gewichtsfunktion („Kernel“) hängt nur vom Abstand $\|\mathbf{x} - \mathbf{s}_i\|$ ab, Gewichte λ_i müssen bestimmt werden, dann ist der Interpolant:

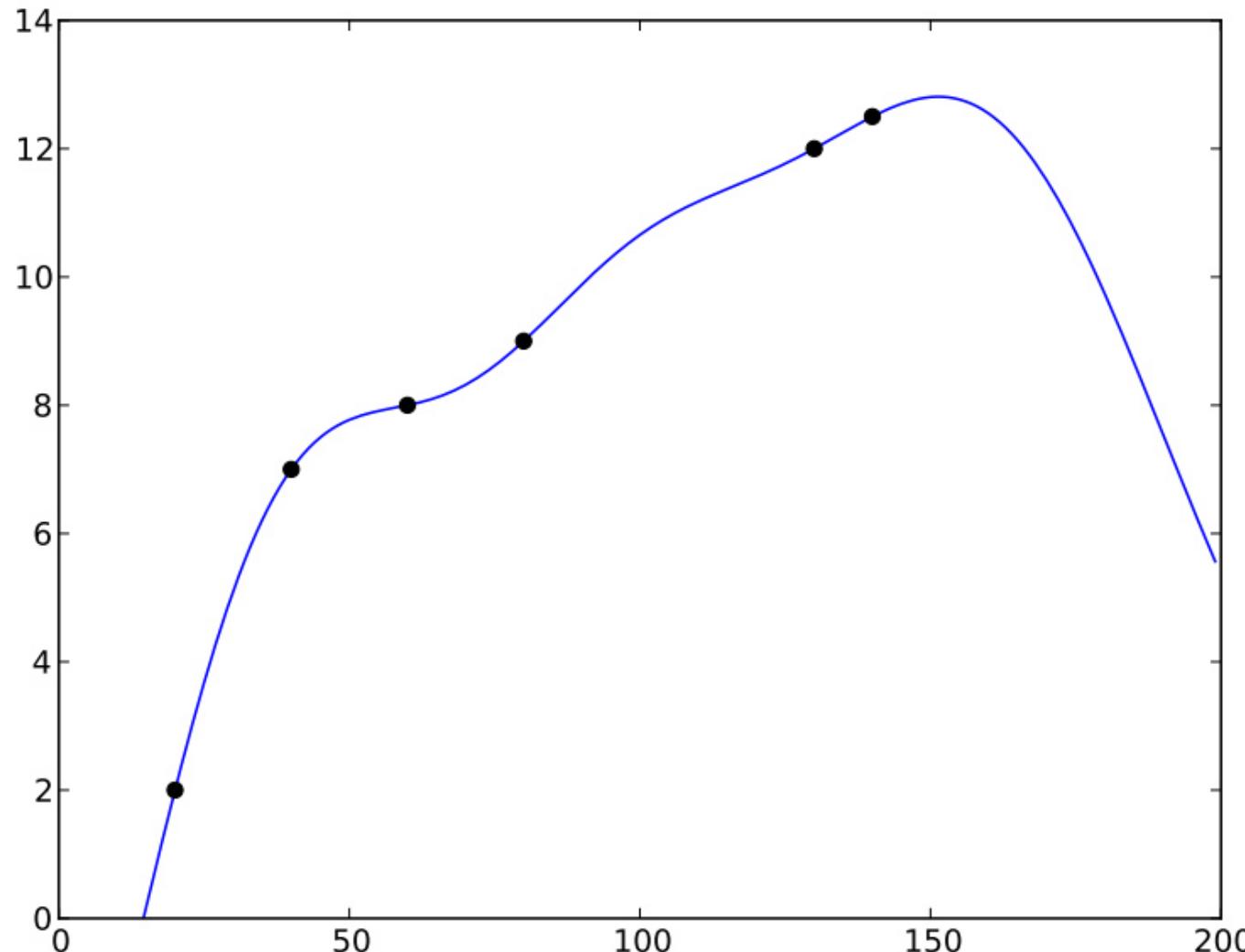
$$v(\mathbf{x}) = \sum_{i=1}^n \lambda_i \phi(\|\mathbf{x} - \mathbf{s}_i\|) + \sum_{m=0}^k c_m p_m(\mathbf{x})$$

- ▶ mit **univariater radialer Basisfunktion $\phi(r)$**
 - ▶ z.B. $\phi(r) = e^{-(r/c)^2}$
 - ▶ **polynomieller Basis p_m** für einen $(k+1)$ -dimensionalen Vektorraum
 - ▶ warum? polynomieller Teil ist **optional**, wird aber oft hinzugefügt, weil lineare Funktionen sonst sehr schlecht interpoliert werden!
 - ▶ außerdem: Extrapolation, da Kernel im Unendlichen verschwindet
 - ▶ daher haben die Polynome einen sehr niedrigen Grad (i.d.R. ≤ 2)

Interpolation ohne Gitter

Radiale Basis Funktionen (RBF)

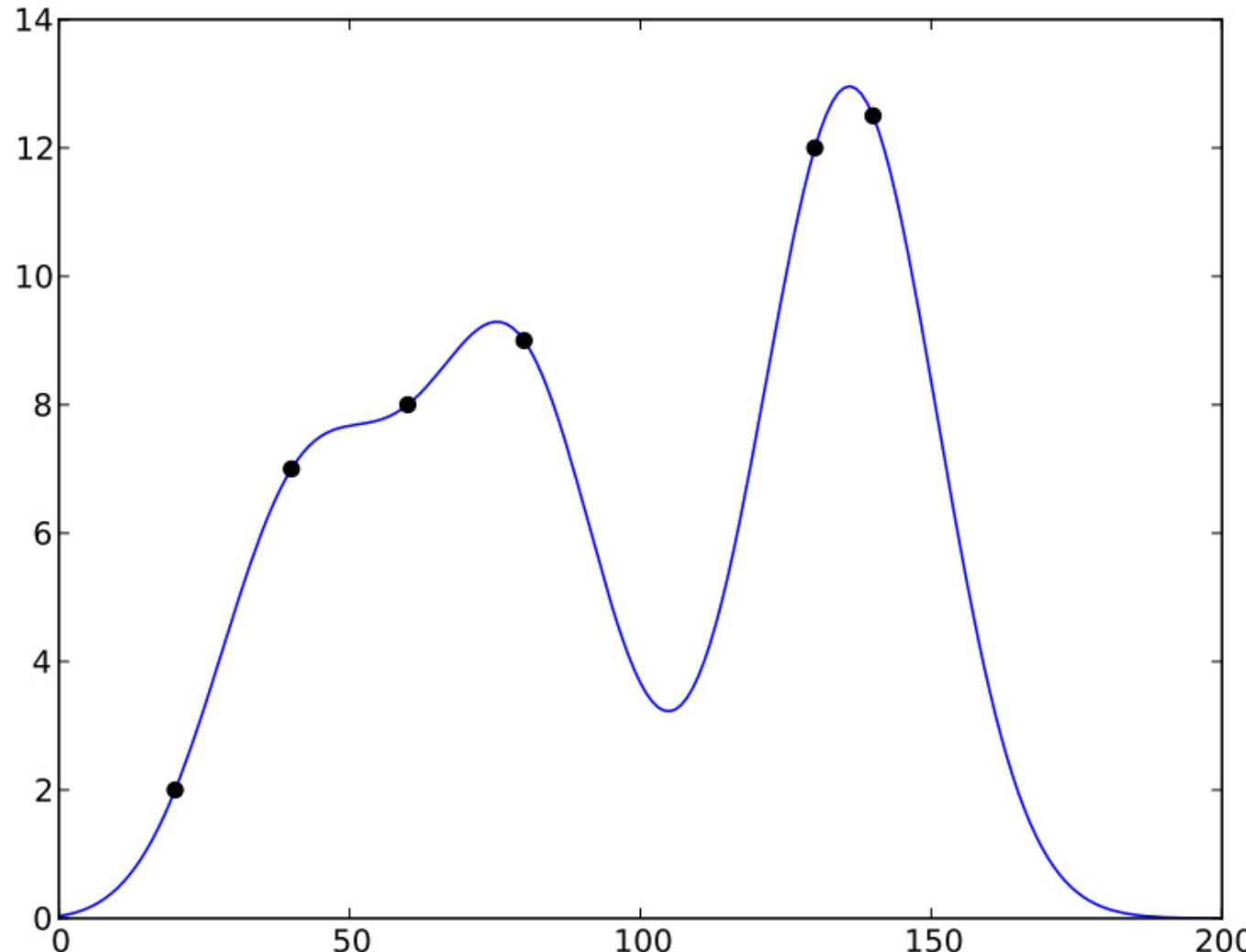
► Beispiel mit $\phi(r) = e^{-(r/c)^2}$



Interpolation ohne Gitter

Radiale Basis Funktionen (RBF)

► Beispiel mit $\phi(r) = e^{-(r/c)^2}$ (kleinerer Kernel als auf vorheriger Folie)



Radiale Basis Funktionen (RBF)

- wir haben ein unterbestimmtes System: wir haben n Gleichungen (die n Datenpunkte sollen interpoliert werden) aber $n + (k + 1)$ Unbekannte
- Gleichungssystem: alle Datenwerte sollen interpoliert werden

$$\sum_{i=1}^n \lambda_i \phi(\|s_j - s_i\|) + \sum_{m=0}^k c_m p_m(s_j) = f_j$$

- wenn der polynomielles Teil verwendet wird benötigen wir zusätzliche Orthogonalitäts-Randbedingungen
- wenn alle Punkte durch ein Polynom darstellbar sind, verschwinden die λ_i (siehe auch SIGGRAPH Course)

$$\sum_{i=1}^n \lambda_i p_m(s_i) = 0 \quad \forall m = 0 \dots k$$

Interpolation ohne Gitter

Radiale Basis Funktionen (RBF)

- wir haben ein unterbestimmtes System: wir haben n Gleichungen (die n Datenpunkte sollen interpoliert werden) aber $n + (k + 1)$ Unbekannte
- wir benötigen zusätzliche Orthogonalitäts-Randbedingungen
 - wenn alle Punkte durch ein Polynom darstellbar sind, verschwinden die λ_i (siehe auch SIGGRAPH Course)

$$\sum_{i=1}^n \lambda_i p_m(x_i) = 0 \quad \forall m = 0 \dots k$$

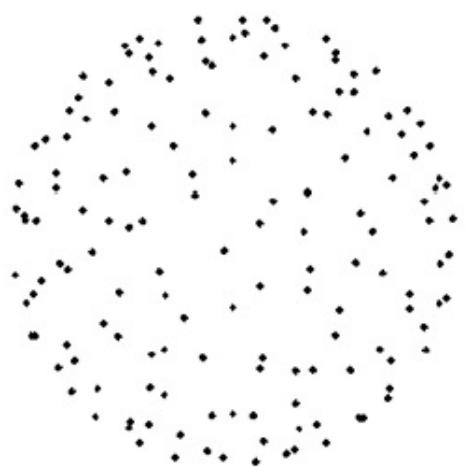
- dadurch erhält man ein lineares Gleichungssystem mit

$$(A \ P^T \ 0) (\lambda \ c) = (f \ 0)$$

Annotations:

- $A_{i,j} = \phi(\|s_j - s_i\|)$ → Koeffizientenvektor für RBF
- Polynomielle Basis → Koeffizienten für die Polynome
- Funktionswerte bei s_i

Hoppe '92 Reconstruction



150 samples



reconstruction
on 50^3 grid

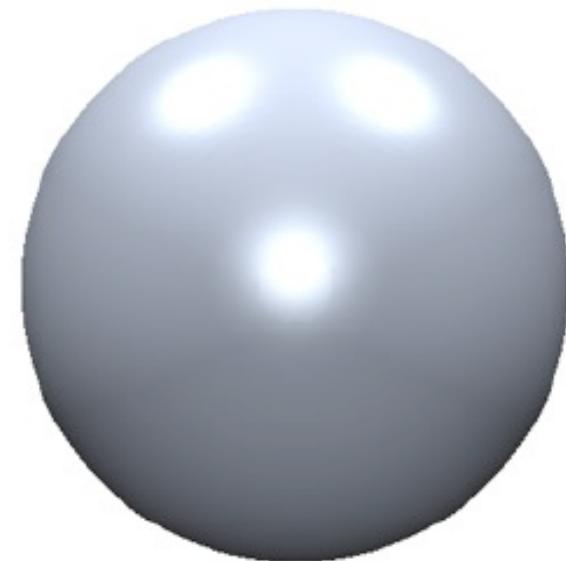
Different Basis Functions



Hoppe '92



Compact RBF
Wendland C^2



Global RBF
Triharmonic

Reconstruction and Representation of 3D Objects with Radial Basis Functions

J. C. Carr^{1,2}

R. K. Beatson²

J. B. Cherrie¹

T. J. Mitchell^{1,2}

W. R. Fright¹

B. C. McCallum¹

T. R. Evans¹

¹Applied Research Associates NZ Ltd *

²University of Canterbury†



Figure 1: (a) Fitting a Radial Basis Function (RBF) to a 438,000 point-cloud. (b) Automatic mesh repair using the biharmonic RBF.

Abstract

We use polyharmonic Radial Basis Functions (RBFs) to reconstruct smooth, manifold surfaces from point-cloud data and to repair incomplete meshes. An object's surface is defined implicitly as the

1 Introduction

Interpolating incomplete meshes (hole-filling) and reconstructing surfaces from point-clouds derived from 3D range scanners are ubiquitous problems in computer graphics and Computer Aided

Scattered Data Interpolation



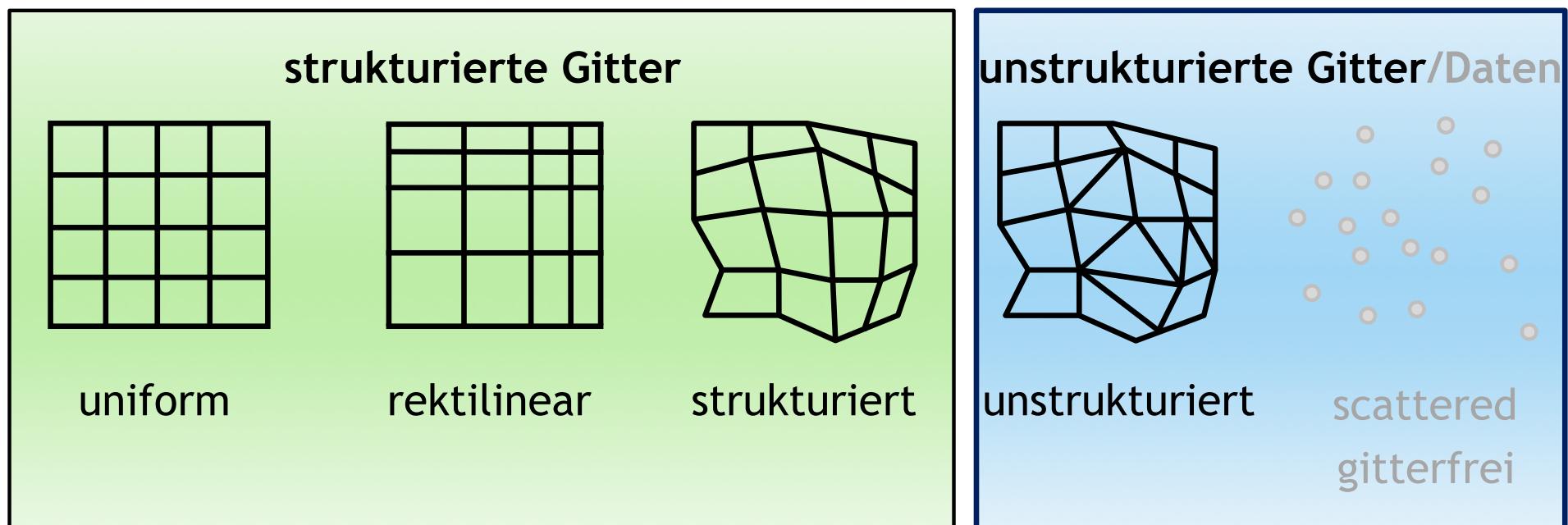
Vergleich der Methoden

- ▶ Nearest Neighbor (Voronoi-Zellen)
 - ▶ schnell, wenn Datenstruktur zur Suche vorhanden ist, aber ...
 - ▶ unstetiger Interpolant (einer ursprünglich vielleicht stetigen Funktion!)
- ▶ Lokale Methoden: Triangulation und anschließende lineare Interpolation
 - ▶ setzt (idealerweise) eine Delaunay Triangulation voraus
- ▶ Shepard Interpolation (Inverse Distance Weighting)
 - ▶ funktioniert gut für die Interpolation „einiger“ Datenpunkte, z.B. innerhalb einer Zelle in einem partitionierten Raum
 - ▶ beliebige Stützstellen, Übertragung auf 3D und n D ist einfach möglich
- ▶ Radiale Basisfunktionen
 - ▶ Interpolant mit hoher Qualität
 - ▶ zeitaufwändig für große Daten ($N > 200$)
 - ▶ numerisch instabil für große N ($N > 1000$)
 - ▶ beliebige Stützstellen, Übertragung auf 3D und n D ist einfach möglich

Datenstrukturen

Gitterfreie Daten und Gittertypen

- ▶ gitterfreie Daten: irregulär verteilte Positionen, keine Konnektivität
- ▶ Konnektivität herstellen und dann (in Simplizes) interpolieren, oder
- ▶ direkte Interpolation anhand aller Datenpunkte
- ▶ Gitter unterscheiden sich in
 - ▶ Art und Form der Zellen aus denen sie aufgebaut sind
 - ▶ Form in der topologische Information gegeben ist



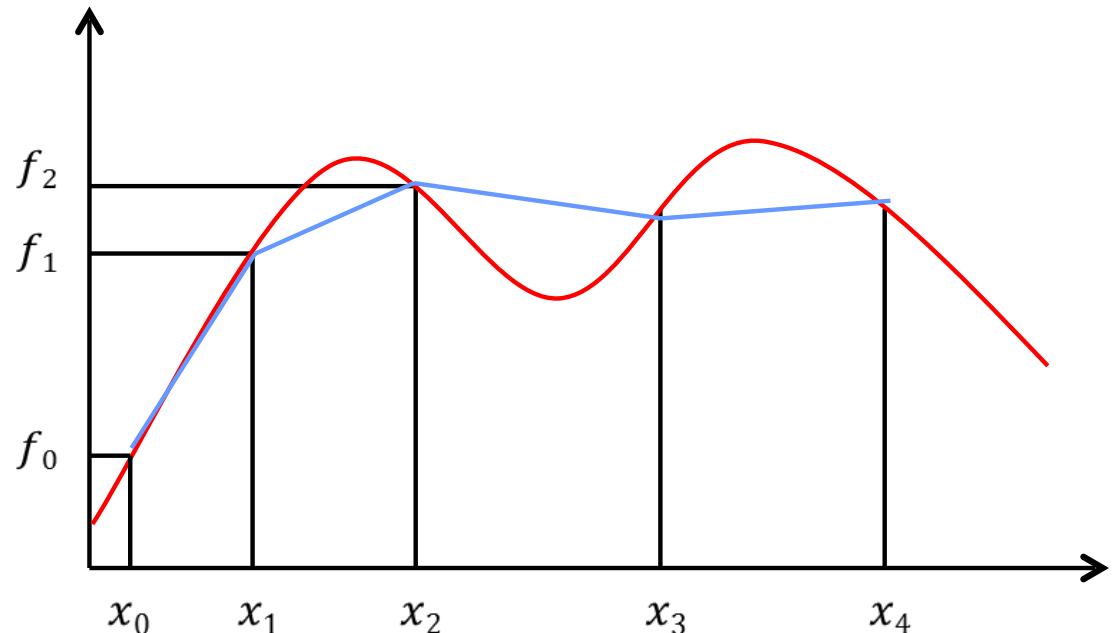
Datenstrukturen

Daten auf Gittern

- Gitter sind wichtig: viele Daten sind als Gitter gegeben, z.B. aus Simulationen, oder durch Abbildung auf Gitter durch Resampling
- Interpolation ist auch wichtig für das **Resampling** selbst
 - z.B. Resampling eines Tetraedernetzes auf ein kartesisches Gitter
- Interpolation ist wichtig für **hochqualitatives Rendering**
- außerdem interessieren uns noch **Ableitungen und Gradienten**, die bei der Visualisierung wichtig sind
- wir betrachten ...
 - zunächst: welche Interpolation kennen wir in 1D?
 - anschließend: wie erweitern wir sie auf höhere Dimensionen?

Univariate Interpolation

- ▶ univariate Interpolation bedeutet nichts anderes als
 - ▶ geg. eine Funktion $f: \mathbb{R} \rightarrow \mathbb{R}$ von der wir nur einige diskrete Funktionswerte $f(x_i)$, mit $x_{i+1} > x_i$ („1D-Gitter“), kennen
 - ▶ ges. eine Funktion ϕ die sich möglichst wenig von f unterscheidet und leicht auszuwerten ist
 - ▶ Funktionen $g: \mathbb{R} \rightarrow \mathbb{R}^m$ analog (jede der m -Dimensionen separat), wir betrachten hier der Einfachheit halber nur skalare Funktionen
- ▶ mögliche Interpolationen
 - ▶ nearest neighbor (0-order)
 - ▶ linear (first-order)
 - ▶ glatt (higher-order)



Univariate Interpolation

Stückweise-lineare Interpolation

- ▶ einfachste Interpolation
 - ◀ oft verwendet, da schnell zu berechnen
 - ◀ C^0 -stetig an den Segmentgrenzen
- ▶ Datenpunkte $(x_0, f_0), \dots, (x_n, f_n)$
- ▶ einen Punkt x mit $x_i \leq x \leq x_{i+1}$ können wir durch eine lokale Koordinate u beschreiben:

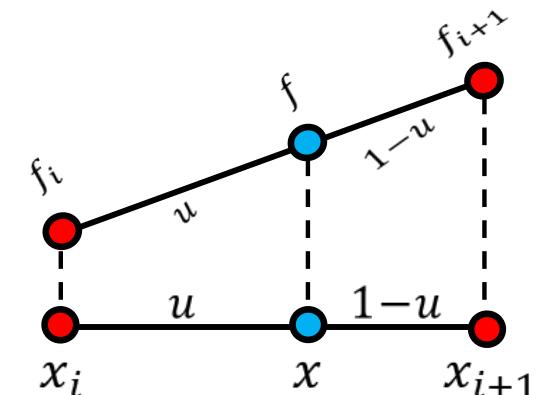
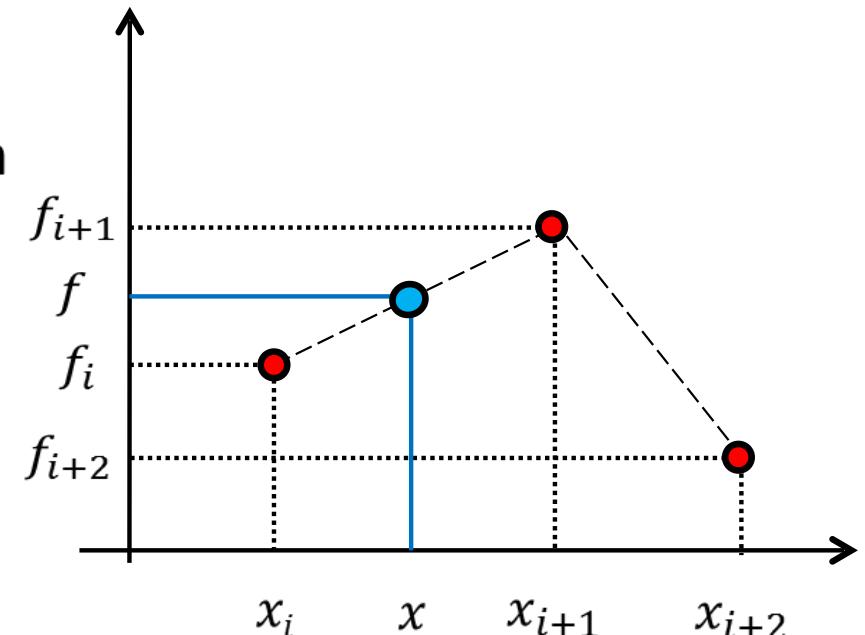
$$u = \frac{x - x_i}{x_{i+1} - x_i} \in [0; 1]$$

◀ d.h.

$$\begin{aligned} x &= x_i + u(x_{i+1} - x_i) \\ &= (1 - u)x_i + ux_{i+1} \end{aligned}$$

und

$$f(x) = (1 - u)f_i + uf_{i+1}$$



Univariate Interpolation



Univariate Interpolation

Taylor Interpolation (Higher Order Interpolation)

- ▶ Idee: verwende die Monom-Basis und berechne ein Polynom, das die Datenpunkte interpoliert (das Polynom ist eindeutig)
- ▶ $(m + 1)$ Datenpunkte \rightarrow Basis-Funktionen $m_i = x^i$ mit $i \in \mathbb{N}_0$ mit $0 \leq i \leq m$
 - ▶ $(m + 1)$ -dimensionaler Vektorraum $P^m = \{1, x, x^2, \dots, x^m\}$

Univariate Interpolation

Taylor Interpolation (Higher Order Interpolation)

- ▶ Idee: verwende die Monom-Basis und berechne ein Polynom, das die Datenpunkte interpoliert (das Polynom ist eindeutig)
- ▶ $(m + 1)$ Datenpunkte \rightarrow Basis-Funktionen $m_i = x^i$ mit $i \in \mathbb{N}_0$ mit $0 \leq i \leq m$
 - ▶ $(m + 1)$ -dimensionaler Vektorraum $P^m = \{1, x, x^2, \dots, x^m\}$
- ▶ mit den Koeffizienten c_i erhalten wir die Darstellung $\phi(x) = \sum_i c_i x^i$ mit der Interpolation der Datenpunkte: $\phi(x_j) = f(x_j) = f_j \quad \forall j = 0 \dots m$
- ▶ Berechnung der Koeffizienten mit der sog. Vandermonde-Matrix $V_{ji} = x_j^i$

$$\mathbf{V} \cdot \mathbf{c} = \mathbf{f}$$

- ▶ Probleme bei der Taylor/Polynom-Interpolation
 - ▶ aufwändige Berechnung: z.B. $O(m^3)$ mit Gaußschen Eliminationsverfahren
 - ▶ Vandermonde-Matrix für hohe Dimensionen „fast singulär“
(d.h. schlecht invertierbar, daher numerische Probleme)
 - ▶ die Anzahl der Punkte bestimmt die Dimension des Vektorraums

Allgemeiner Ansatz: Polynominterpolation

- Idee: wähle eine andere Polynombasis mit dem Ziel
 - den Aufwand bei der Berechnung der Koeffizienten zu verringern
 - stabilere Berechnung der Koeffizienten
- geg. n Punkte $X = \{x_i\} \subseteq \Omega \subseteq \mathbb{R}$ mit dazugehörigen Funktionswerten f_i
- n -dimensionaler Funktionenraum $\Omega_n(\Omega)$ mit Basis $\{b_{i=1..n}\}$
- Koeffizienten c_i mit $\phi(x) = \sum_i c_i b_i(x)$
- Interpolation der Datenpunkte $\phi(x_j) = f(x_j) = f_j \quad \forall j = 0 \dots n - 1$
- Berechnung der Koeffizienten durch Lösung das lin. Gleichungssystems

$$\mathbf{M} \cdot \mathbf{c} = \mathbf{f}$$

- mit $M_{ji} = b_i(x_j)$

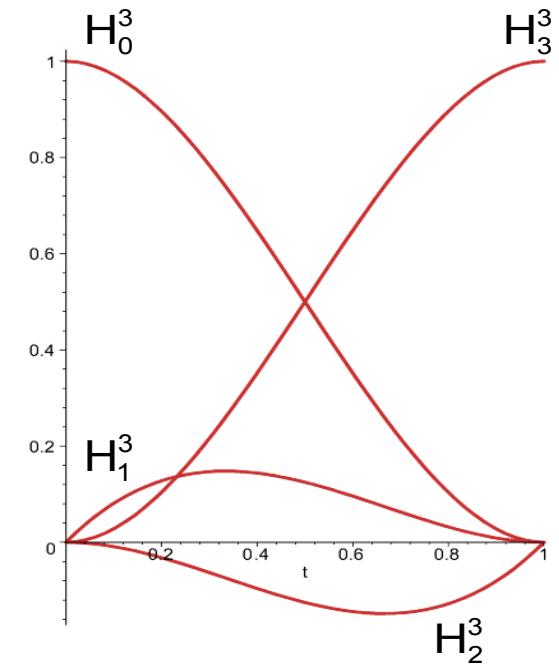
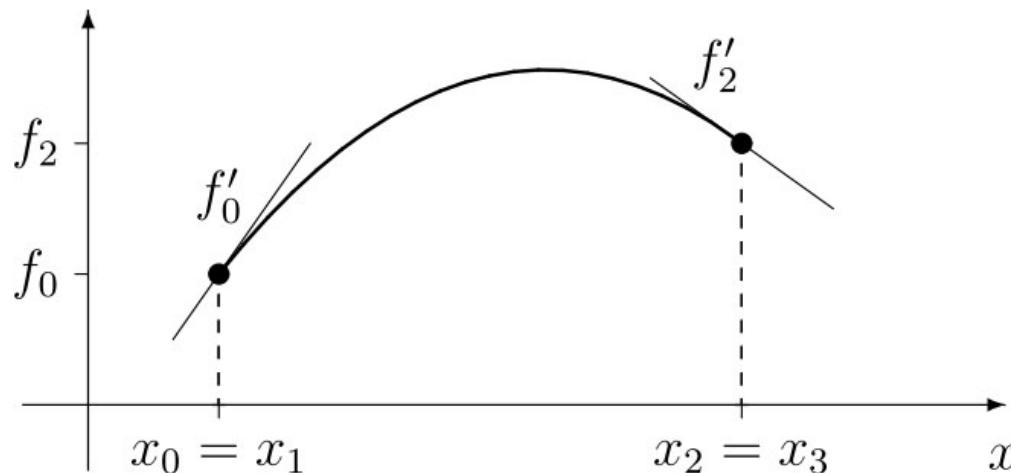
Beispiele für Basisfunktionen

- andere Basisfunktionen resultieren in anderen Interpolationsschemata
 - ▶ Taylor Basis $b_i(x) = x^i$
 - ▶ Lagrange Basis $b_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$
 - ▶ $b_i(x_j) = \delta_{ij} \Rightarrow c_i = f_i$ (**M** ist Einheitsmatrix, meist für Beweise)
 - ▶ Newton Basis $b_i(x) = \prod_{j=1}^i (x - x_j)$
 - ▶ effiziente Berechnung der Koeffizienten (**M** ist Dreiecksmatrix)
 - ▶ Bernstein Basis (Bézier) $b_i^k(x) = \binom{k}{i} (1 - x)^{k-i} x^i$
 - ▶ Bézier-Splines
 - ▶ B-Splines $b_i^k(x) = \frac{x - t_i}{t_{i+k} - t_i} b_i^{k-1}(x) + \frac{x - t_{i+1}}{t_{i+k+1} - t_{i+1}} b_{i+1}^{k-1}(x)$
- Problem mit Polynominterpolation: hoher Grad führt zu „Überschwingern“ → stückweise Interpolation mit niedrigem Grad

Univariate Interpolation

Hermite Splines (stückweise kubische Interpolation)

- ▶ manchmal sind von einer Funktion $f: \mathbb{R} \rightarrow \mathbb{R}^d$ neben den Funktionswerten $f(x_i)$ auch die Ableitungen $f'(x_i)$ bekannt (oder können leicht berechnet werden)
- ▶ dann bietet sich eine Interpolation mit (kubischen) Hermite Polynomen H an
- ▶ Koeffizienten beschreiben die Endpunkte und Ableitungen



$$H_0^3(t) = (1-t)^2(1+2t)$$

$$H_1^3(t) = t(1-t)^2$$

$$H_2^3(t) = -t^2(1-t)$$

$$H_3^3(t) = (3-2t)t^2$$

Univariate Interpolation

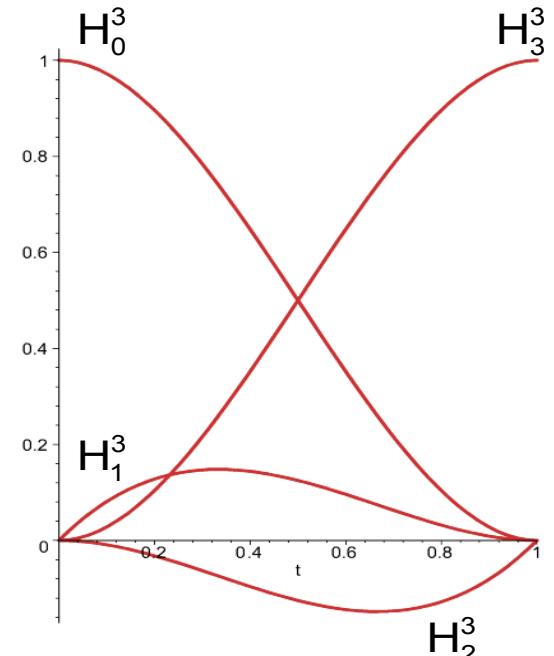
Hermite Splines (stückweise kubische Interpolation)

► Hermite-Interpolation

- ▶ $f(x) = H_0^3(t) \cdot f(x_i) + H_1^3(t) \cdot f'(x_i) + H_2^3(t) \cdot f'(x_{i+1}) + H_3^3(t) \cdot f(x_{i+1})$
- ▶ mit $t = \frac{x-x_i}{x_{i+1}-x_i} \in [0; 1]$ für $x_i \leq x \leq x_{i+1}$

- ▶ sind Ableitungen nicht bekannt, dann kann man sie aus den Datenpunkten berechnen
- ▶ z.B. für $f: \mathbb{R} \rightarrow \mathbb{R}$ mit

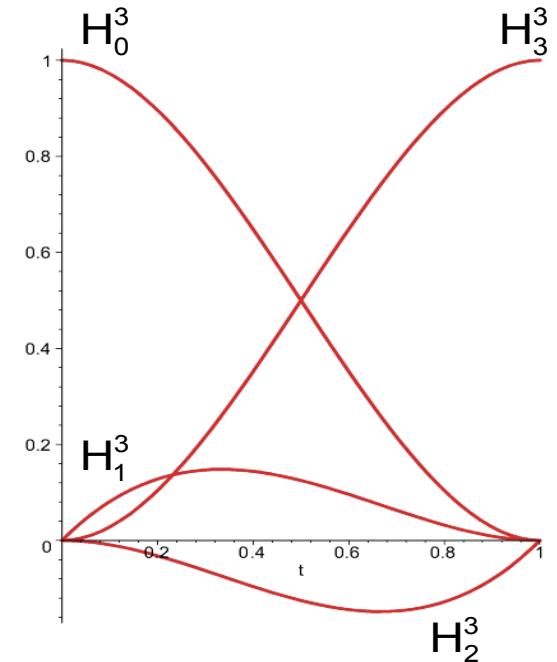
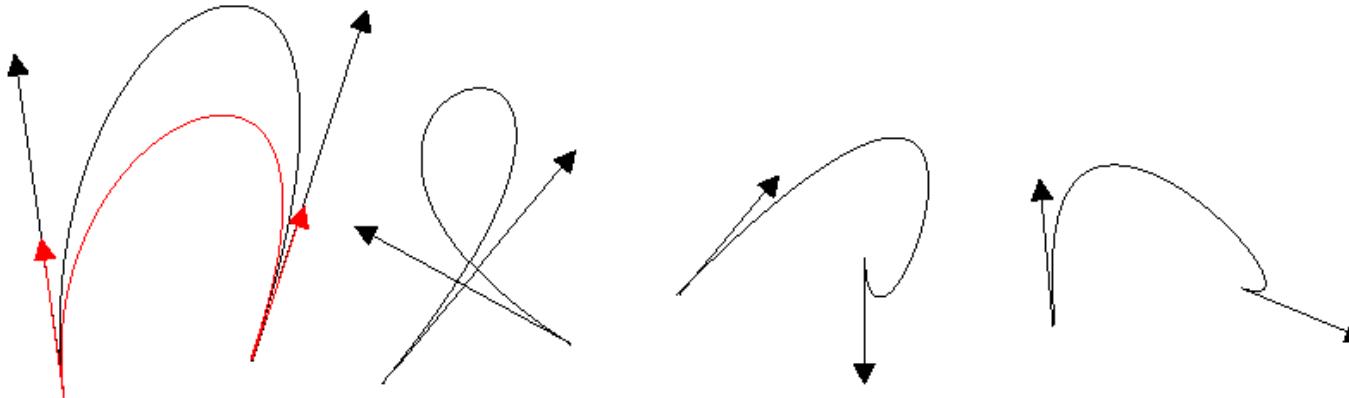
$$f'(x_i) = (1 - c) \frac{f(x_{i+1}) - f(x_{i-1})}{a}$$
- ▶ für $a = 2$ und $c \neq 0$ spricht man von Cardinal Splines
- ▶ für $a = 2$ und $c = 0$ von Catmull-Rom Splines
- ▶ $c \in [-1; 1]$ ist Spannung der Kurve (kleineres c , weichere Kurve)
- ▶ lokale Eigenschaft: im Intervall $x_i < x < x_{i+1}$ hängt die Funktion $f(x)$ nur von den Datenpunkten x_{i-1}, x_i, x_{i+1} und x_{i+2} ab



Univariate Interpolation

Hermite Splines im \mathbb{R}^2

- ▶ Koeffizienten beschreiben die Endpunkte und Ableitungen
- ▶ Beispiel in 2D: Koeffizienten sind hier Endpunkte und Tangentenvektoren (eines Segments/Parameterintervalls)



$$H_0^3(t) = (1-t)^2(1+2t)$$

$$H_1^3(t) = t(1-t)^2$$

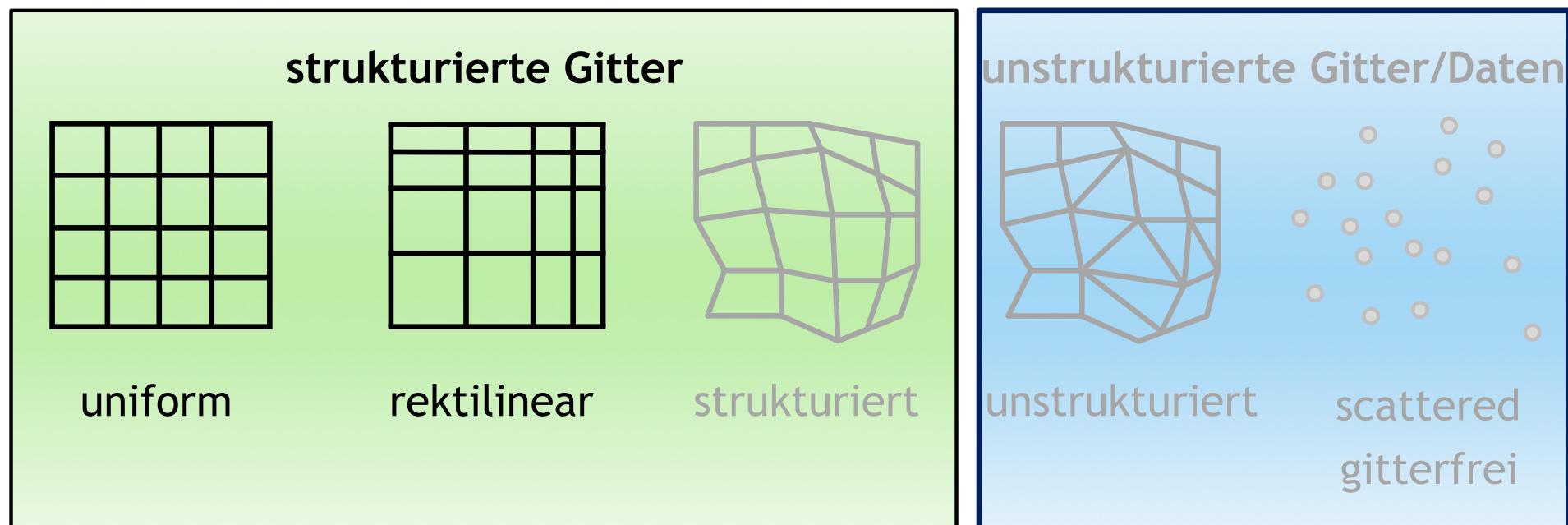
$$H_2^3(t) = -t^2(1-t)$$

$$H_3^3(t) = (3-2t)t^2$$

Datenstrukturen

Gitterfreie Daten und Gittertypen

- gitterfreie Daten: irregulär verteilte Positionen, keine Konnektivität
- Konnektivität herstellen und dann (in Simplizes) interpolieren, oder
- direkte Interpolation anhand aller Datenpunkte
- Gitter unterscheiden sich in
 - Art und Form der Zellen aus denen sie aufgebaut sind
 - Form in der topologische Information gegeben ist



Interpolation auf (regulären) Gittern

Tensorprodukt Ansatz: von univariat (1D) auf höhere Dimensionen

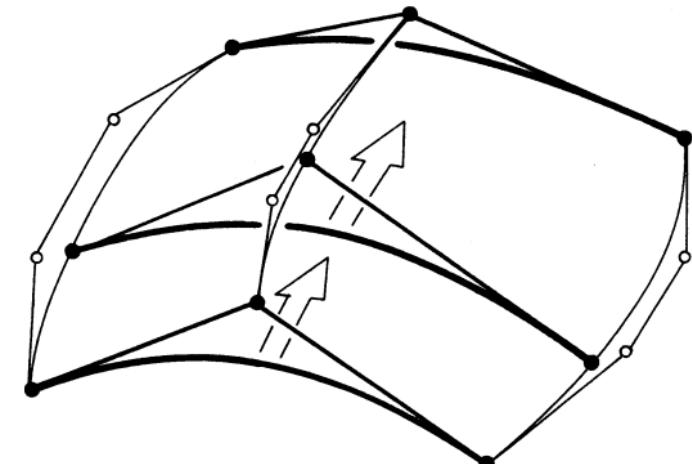
- ▶ Voraussetzung für Tensorprodukt: Stützstellen bilden rechteckiges Array
- ▶ meistens kombiniert man mehrere 1D Interpolationen
 - ▶ z.B. bilineare Interpolation in 2D (bzw. trilineare in 3D) auf der Basis linearer Interpolation in 1D

Interpolation auf (regulären) Gittern

Tensorprodukt Ansatz: von univariat (1D) auf höhere Dimensionen

- ▶ Voraussetzung für Tensorprodukt: Stützstellen bilden rechteckiges Array
- ▶ meistens kombiniert man mehrere 1D Interpolationen
 - ▶ z.B. bilineare Interpolation in 2D (bzw. trilineare in 3D) auf der Basis linearer Interpolation in 1D
- ▶ Beispiel für den 2D-Fall (allgemein)
 - ▶ geg. $n \cdot m$ Werte f_{jl} mit $j = 1..n$ und $l = 1..m$ an den Punkten $X \times Y = (x_1, \dots, x_n) \times (y_1, \dots, y_m)$
 - ▶ n univariate Basisfunktionen $\xi_j(x)$ auf X
 - ▶ m univariate Basisfunktionen $\psi_l(y)$ auf Y
 - ▶ $\rightarrow n \cdot m$ Basisfunktionen auf $X \times Y$:
$$\phi_{ij}(x, y) = \xi_i(x) \cdot \psi_j(y)$$
 - ▶ Tensorprodukt

$$f(x, y) = \sum_{i=1, j=1}^{n, m} \phi_{ij}(x, y) c_{ij}$$



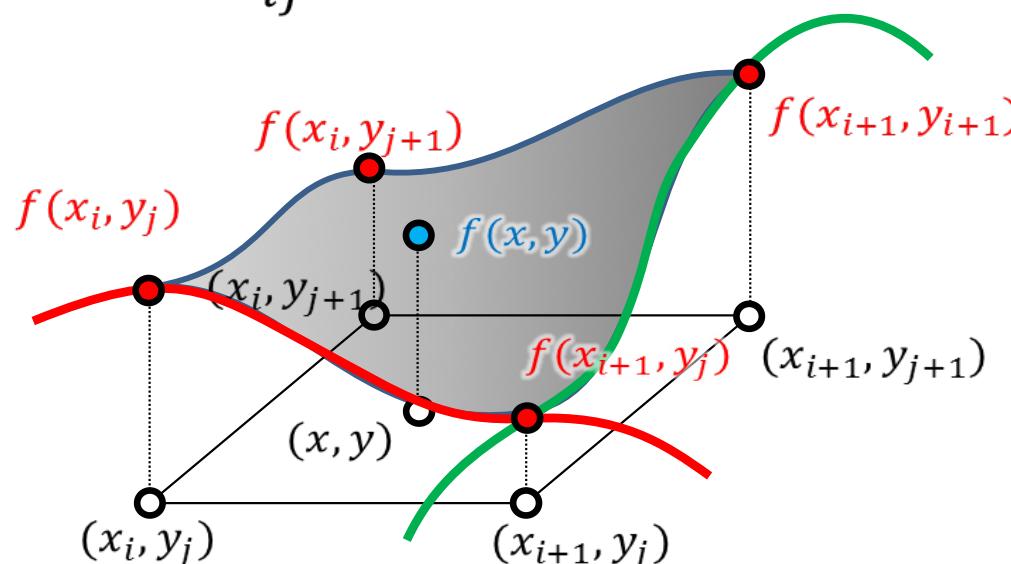
- ▶ Erweiterung auf höhere Dimensionen als 2D funktioniert analog

Interpolation auf (regulären) Gittern

- ▶ Tensorprodukt mit $n \cdot m$ Basisfunktionen $\phi_{ij}(x, y) = \xi_i(x) \cdot \psi_j(y)$

$$f(x, y) = \sum_{i=1, j=1}^{n, m} \phi_{ij}(x, y) c_{ij}$$

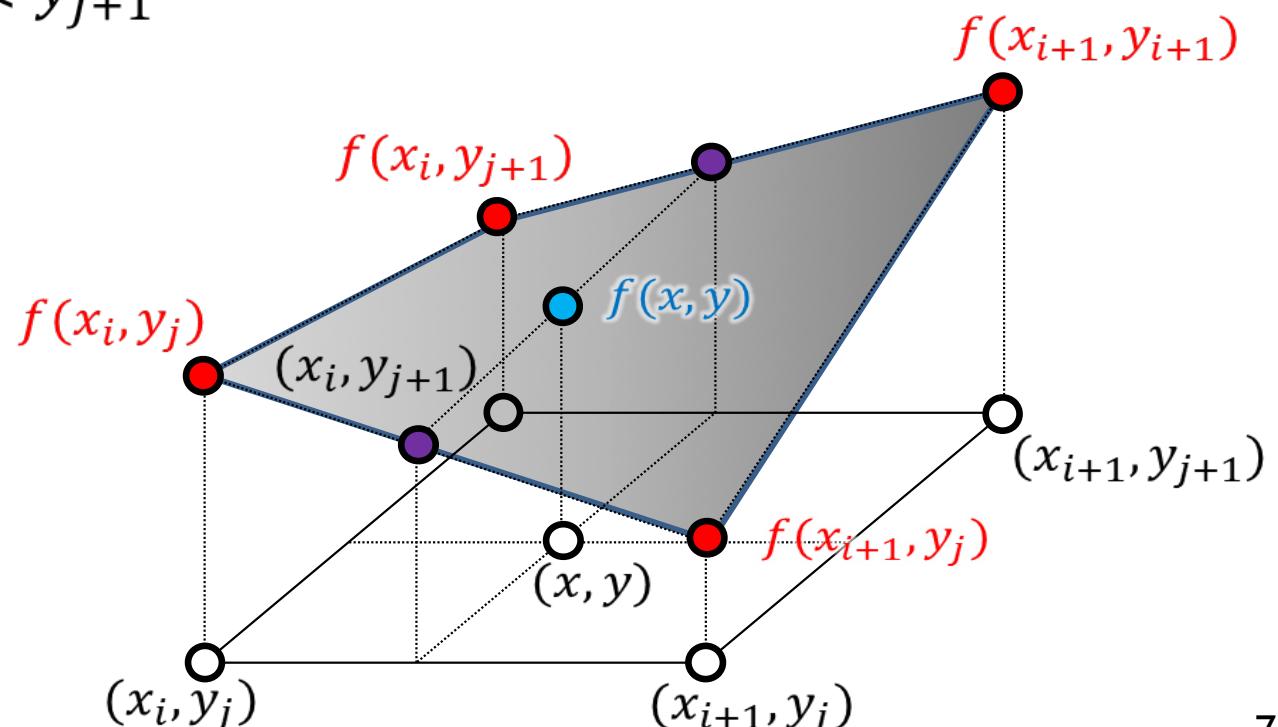
- ▶ je nach Art der Basisfunktionen muss u.U. ein Gleichungssystem gelöst werden, um die Koeffizienten c_{ij} zu bestimmen
 - ▶ bei bi-/trilinearer Interpolation entfällt dieser Schritt
 - ▶ sind $\xi_i(x)$ und $\psi_j(y)$ bsp. radiale Basisfunktionen ist leicht ersichtlich, dass die Koeffizienten c_{ij} neu bestimmt werden müssen



Interpolation auf (regulären) Gittern

Beispiel: Bilineare Interpolation auf einem Rechteck

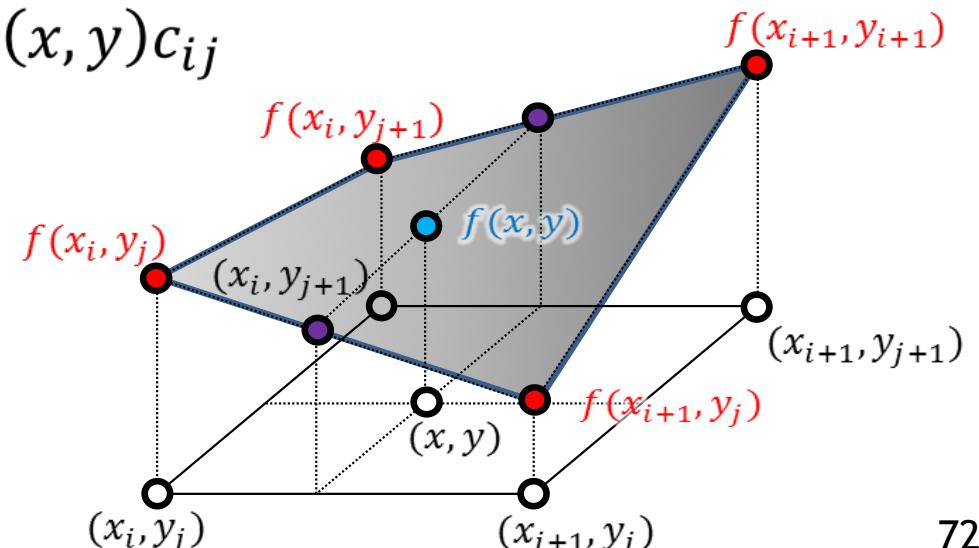
- ▶ Tensorprodukt zweier linearer Interpolationen
 - ▶ dadurch: lokale Interpolation innerhalb einer Zelle
- ▶ geg. 4 Datenpunkte $(x_i, y_j), \dots, (x_{i+1}, y_{j+1})$ und 4 Datenwerte $f_{i,j} = f(x_i, y_i), \dots$
- ▶ bilineare Interpolation der Punkte (x, y) mit
$$x_i \leq x < x_{i+1} \wedge y_j \leq y < y_{j+1}$$



Interpolation auf (regulären) Gittern

Beispiel: Bilineare Interpolation auf einem Rechteck

- geg. Werte f_{ij} mit $i = 1..2$ und $j = 1..2$
an den Punkten $X \times Y = (x_1, x_2) \times (y_1, y_2)$
- je zwei univariate Basisfunktionen $\xi_i(x)$ auf X und $\psi_j(y)$ auf Y
 - $\xi_1(x) = (1 - \alpha) \quad \xi_2(x) = \frac{x - x_1}{x_2 - x_1} =: \alpha$
 - $\psi_1(y) = (1 - \beta) \quad \psi_2(y) = \frac{y - y_1}{y_2 - y_1} =: \beta$
- $\rightarrow 2 \cdot 2$ Basisfunktionen auf $X \times Y$: $\phi_{ij}(x, y) = \xi_i(x) \cdot \psi_j(y)$
 - $\phi_{11}(x, y) = (1 - \alpha)(1 - \beta), \dots$
- Tensorprodukt $f(x, y) = \sum_{i=1, j=1}^{n, m} \phi_{ij}(x, y) c_{ij}$
- die Lösung der Gleichungssystems
für c_{ij} ist uns schon bekannt:
es sind die Werte an den
4 Eckpunkten der Zelle



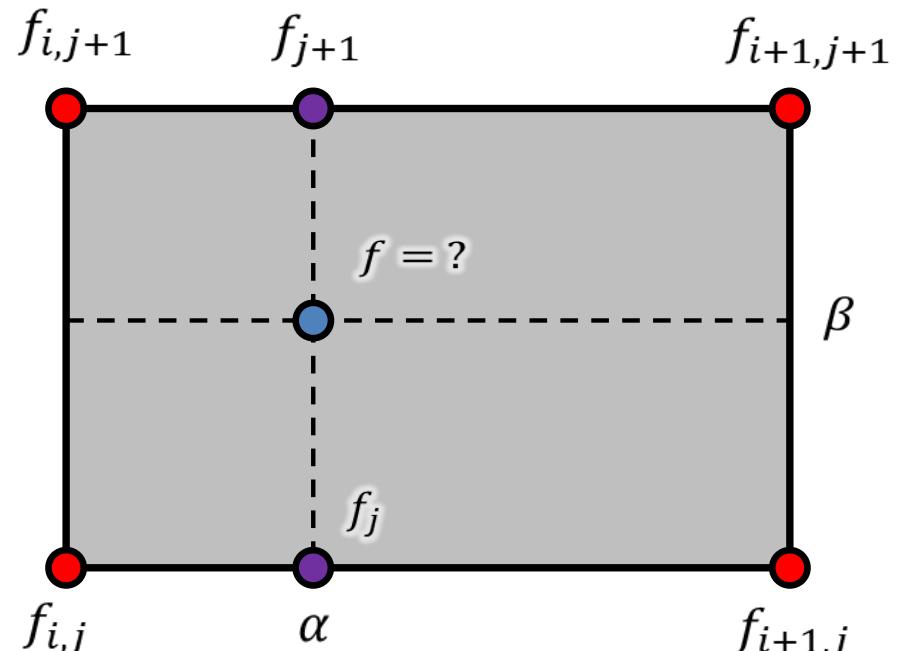
Interpolation auf (regulären) Gittern

Beispiel: Bilineare Interpolation auf einem Rechteck

- $f(x, y) = (1 - \beta)[(1 - \alpha)f_{i,j} + \alpha f_{i+1,j}] + \beta[(1 - \alpha)f_{i,j+1} + \alpha f_{i+1,j+1}]$
= $(1 - \beta)f_j + \beta f_{j+1}$
- mit $f_j = (1 - \alpha)f_{i,j} + \alpha f_{i+1,j}$
 $f_{j+1} = (1 - \alpha)f_{i,j+1} + \alpha f_{i+1,j+1}$
- und den lokalen Koordinaten

$$\alpha = \frac{x - x_i}{x_{i+1} - x_i}, \quad \beta = \frac{y - y_i}{y_{i+1} - y_i}$$

und $\alpha, \beta \in [0; 1]$



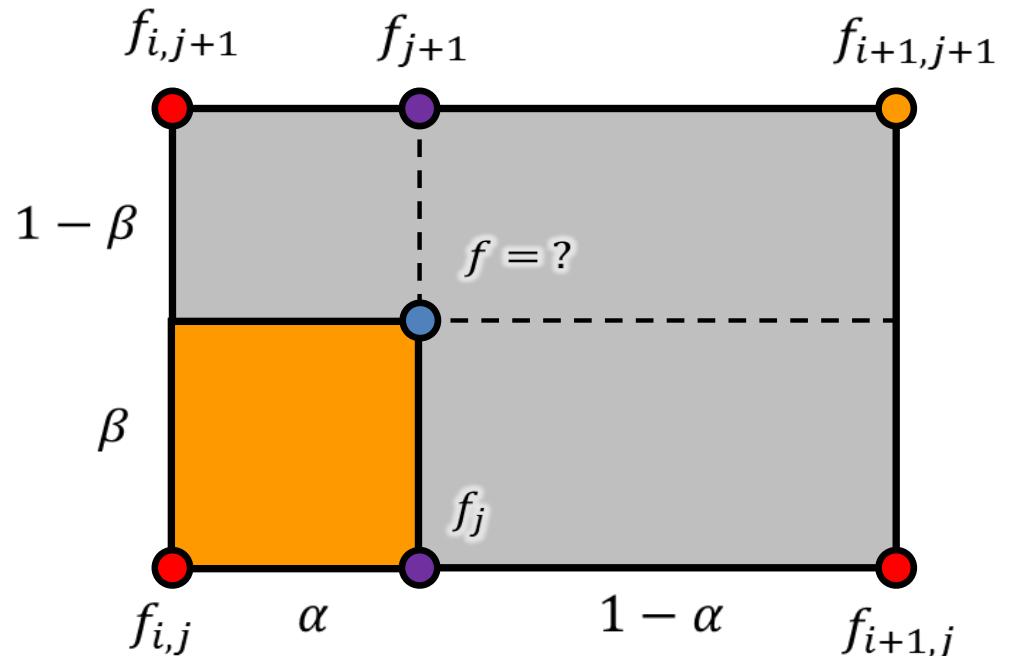
Interpolation auf (regulären) Gittern

Beispiel: Bilineare Interpolation auf einem Rechteck

- Sortierung der Terme nach den Datenwerten

$$f(x, y) = (1 - \alpha)(1 - \beta)f_{i,j} + \alpha(1 - \beta)f_{i+1,j} + \\ (1 - \alpha)\beta f_{i,j+1} + \color{orange}\alpha\beta f_{i+1,j+1}$$

- Gewichtung eines Datenwerts mit der gegenüberliegenden Fläche geteilt durch die Gesamtfläche
- bilineare Interpolation ist nicht linear, sondern quadratisch
(abgesehen von Interpolation entlang von Geraden parallel zu den Kanten)



Trilineare Interpolation auf einem uniformen 3D-Gitter

- ▶ einfache Erweiterung der bilinearen Interpolation
 - ▶ mit drei lokalen Koordinaten α, β, γ
 - ▶ die Lösung des Gleichungssystems für die c_{ijk} sind bekannt
 - ▶ ebenfalls keine lineare Interpolation
- ▶ effiziente Auswertung:

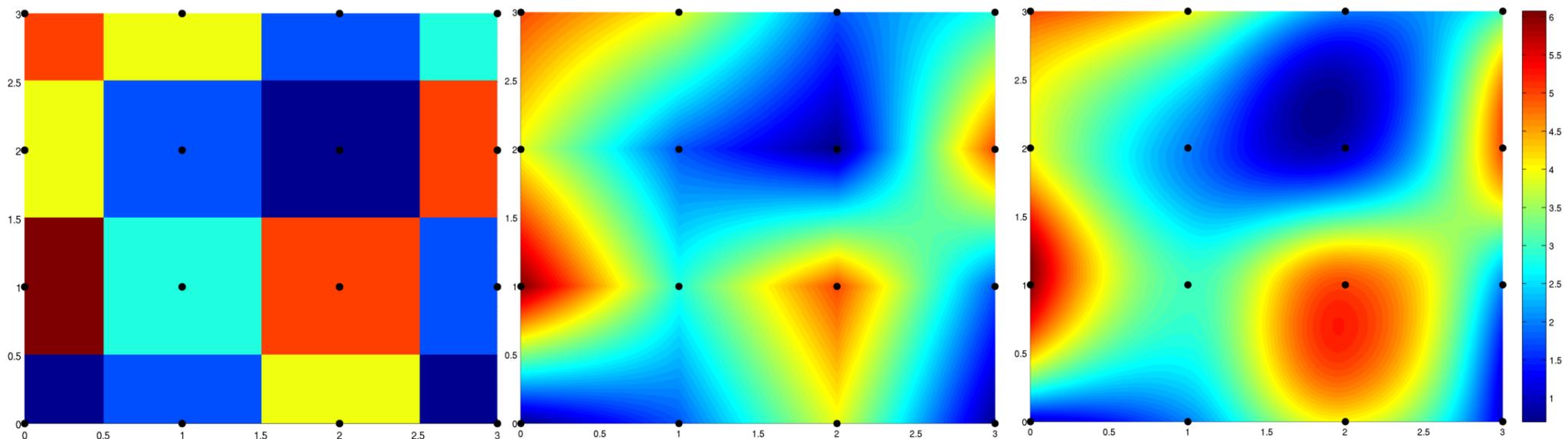
$$f(\alpha, \beta, \gamma) = a + \alpha(b + \beta(e + h\gamma)) + \beta(c + f\gamma) + \gamma(d + g\alpha)$$

mit den Koeffizienten (=Datenwerten an den Ecken) a, b, c, d, e, f, g, h

- ▶ für Interpolation mit höherer Stetigkeit verwendet man häufig
 - ▶ stückweise kubische Interpolation in 1D (bi-/trikubisch in 2D/3D)
 - ▶ oft basierend auf den Hermite Polynomen
 - ▶ Datenpunkte benötigt für die Interpolation
 - ▶ linear: 2, bilinear 4, trilinear 8
 - ▶ kubisch: 4, bikubisch: 16, trikubisch: 64

Interpolation auf (regulären) Gittern

Beispiel: Nearest Neighbor, bilineare und bikubische Interpolation



Datenstrukturen

Gitterfreie Daten und Gittertypen

- gitterfreie Daten: irregulär verteilte Positionen, keine Konnektivität
- Konnektivität herstellen und dann (in Simplizes) interpolieren, oder
- direkte Interpolation anhand aller Datenpunkte
- Gitter unterscheiden sich in
- Art und Form der Zellen aus denen sie aufgebaut sind

