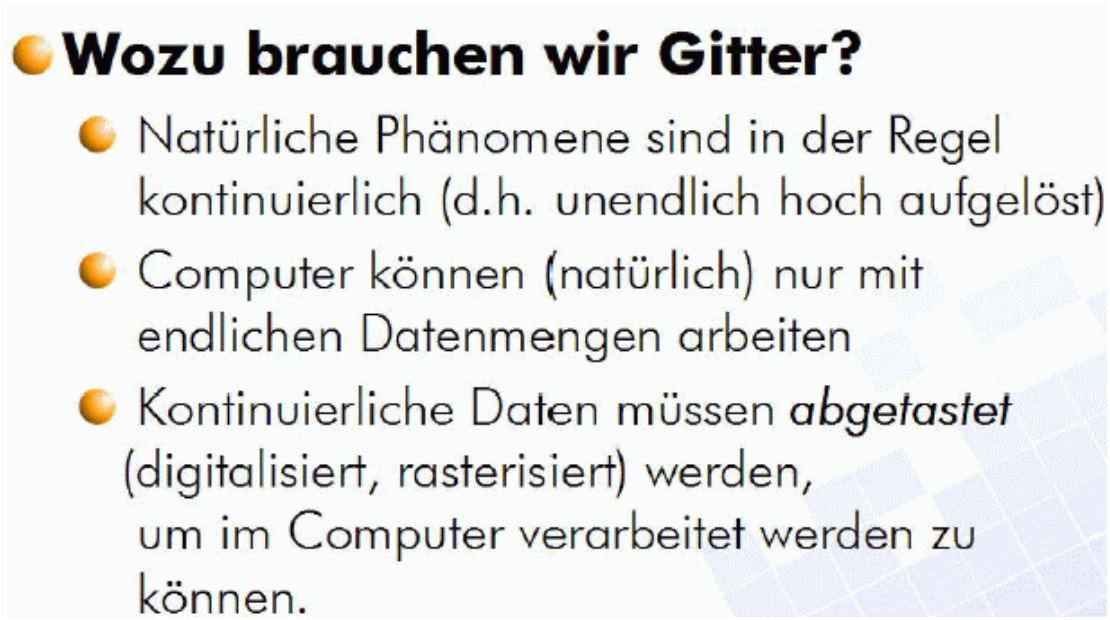
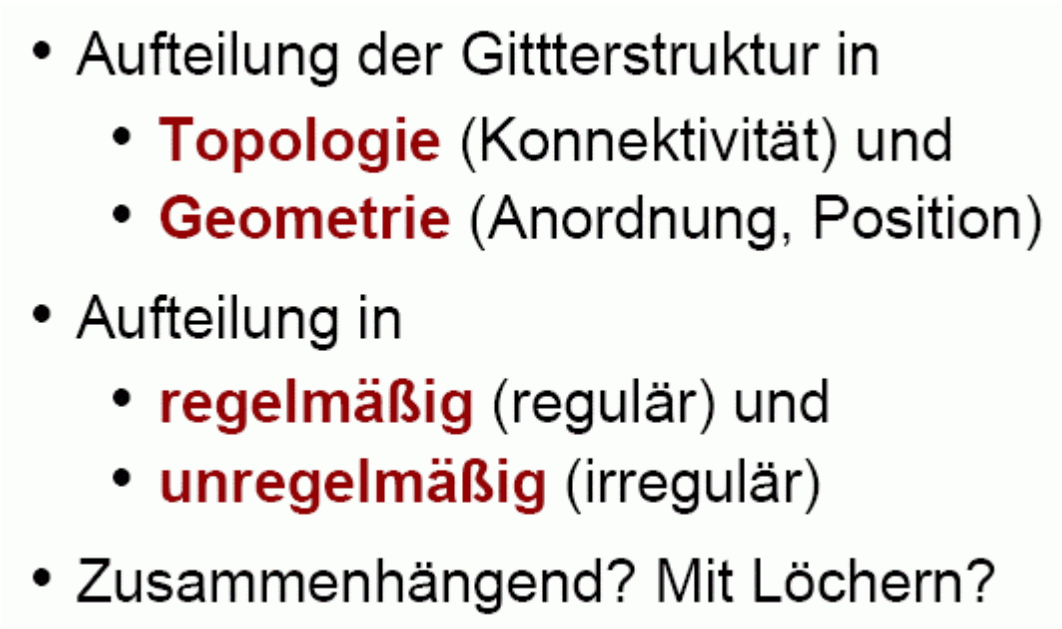
5. Gitter, Gradienten, Interpolation

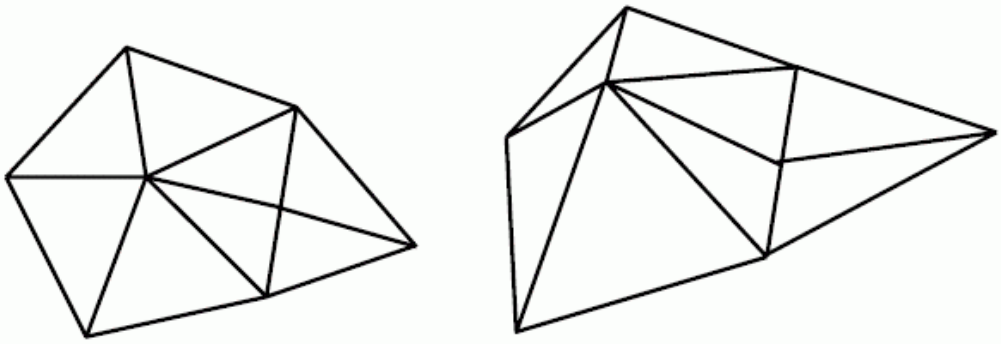
5.1. Gitter



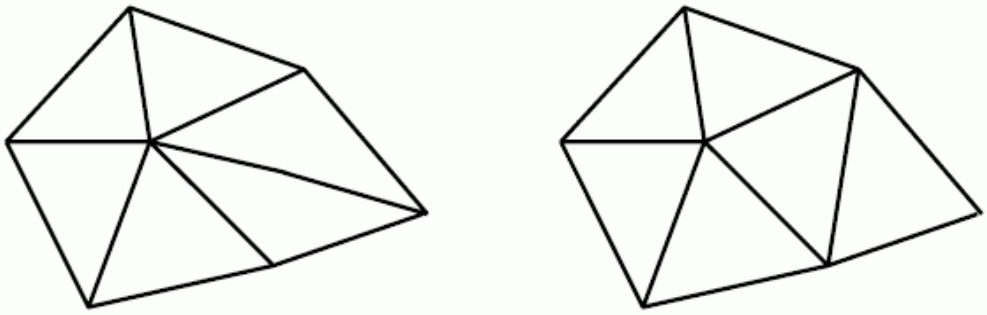
(Rezk-Salama, o.J.)



- Topologie beschreibt **Nachbarschaft** von Zellen
- Berücksichtigt nicht Position der Gitterpunkte, dh. es ist unabhängig von Verzerrungen
- Der **Valenz** eines Gitterpunktes beschreibt Topologie



- Topologisch äquivalent, aber nicht identisch



- Topologisch unterschiedlich, geometrisch aber gleich

Gitterklassifikation:

- Kartesische/uniforme/rechteckige (rectilinear) Gitter: **regelmäßige** Topologie und Geometrie
- Gebogene (curvilinear) Gitter: **regelmäßige** Topologie, **unregelmäßige** Geometrie
- Unregelmäßige (irregular, unstructured) Gitter: **unregelmäßige** Topologie und Geometrie
- Unregelmäßige Topologie, regelmäßige Geometrie werden nicht wirklich verwendet.

Gitterfreie Daten (*set / ordered set*)

Regelmäßige Gitter (*regular grid*): haben achsenparallele Gitterlinien. Der Abstand von benachbarten parallelen Gitterlinien ist konstant.

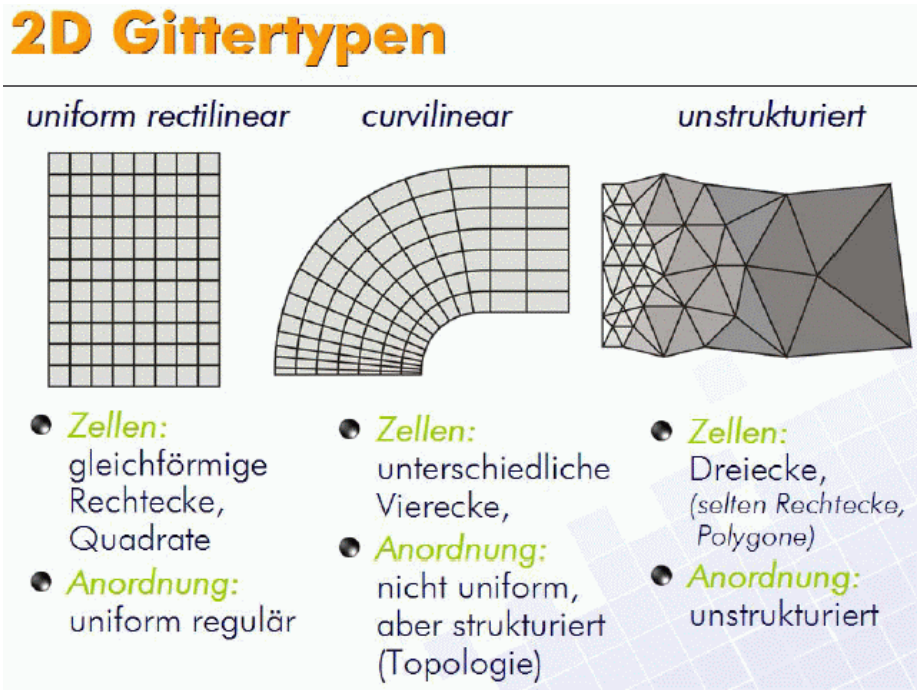
Unregelmäßige oder unstrukturierte Gitter (*unstructured grid*): werden durch beliebige Gitterlinien mit beliebigem Abstand definiert. Das heißt, sowohl die Koordinaten als auch der Verbund der Gitterpunkte muss explizit gegeben sein.

Blockstrukturierte Gitter (*block-grid*): haben achsenparallele Gitterlinien mit unterschiedlichen Abständen.

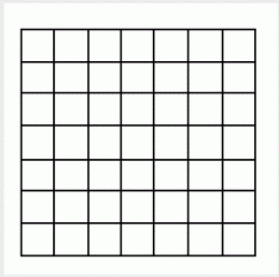
Strukturierte Gitter (*structured grid / curvilinear grid*): haben beliebige Gitterlinien. Die Koordinaten eines Gitterpunktes auf einer Gitterlinie sind analytisch gegeben. Sind die Gitterlinien als Kurven in Parameterform gegeben, sprechen wir auch von "Curvilinearen Gittern".

Hybride Gitter: setzen sich aus unterschiedlichen Gittertypen zusammen.

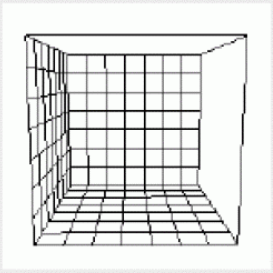
(Bartz 2005)



(Rezk-Salama, o.J.)

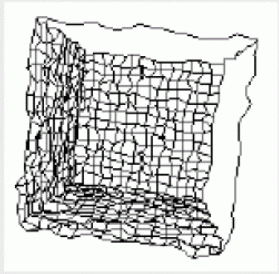


2D-Regular Grid

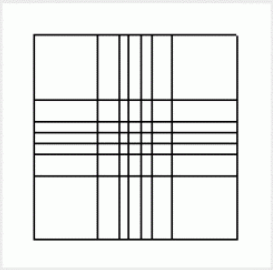


3D-Regular Grid

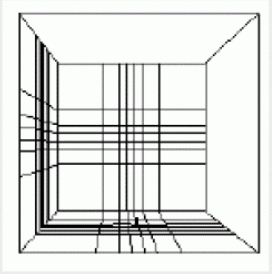
2D-Irregular Grid



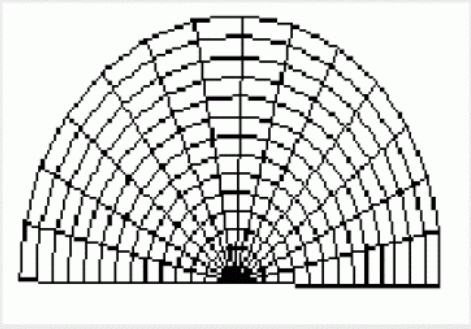
3D-Irregular Grid



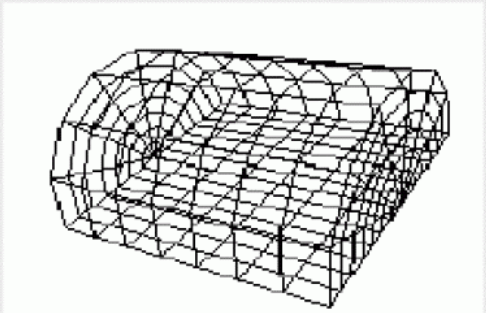
2D-Block-Structured Grid



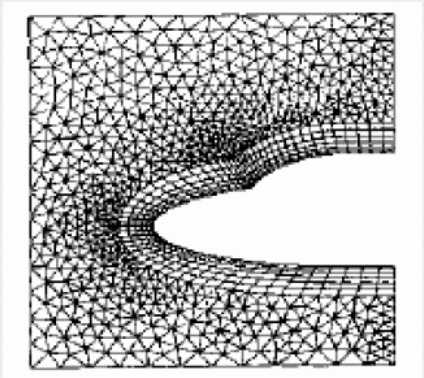
3D-Block-Structured Grid



2D-Structured Grid

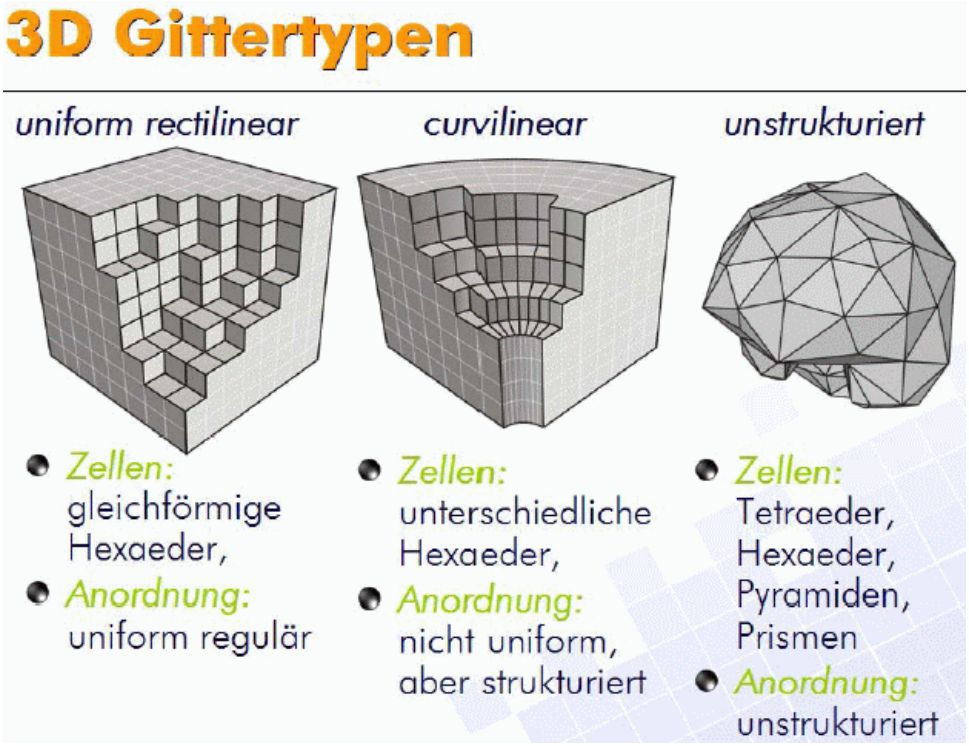


3D-Structured Grid



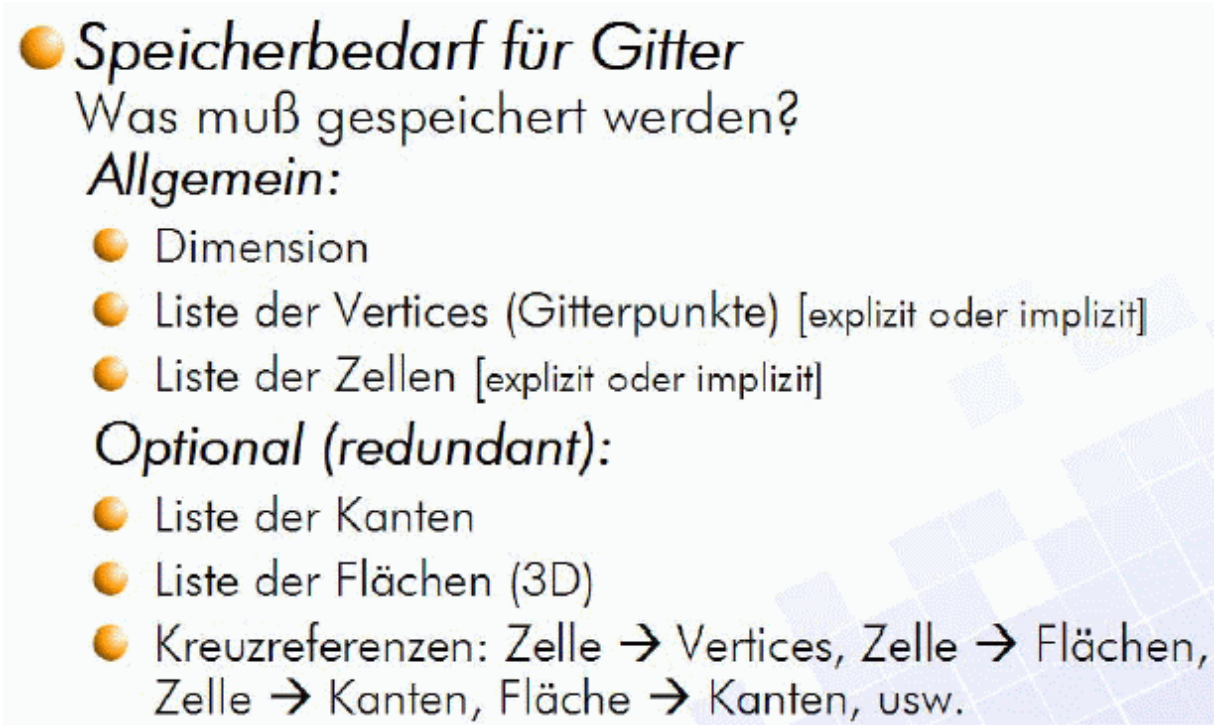
2D-Hybrid Grid

(Bartz 2005)



(Rezk-Salama, o.J.)

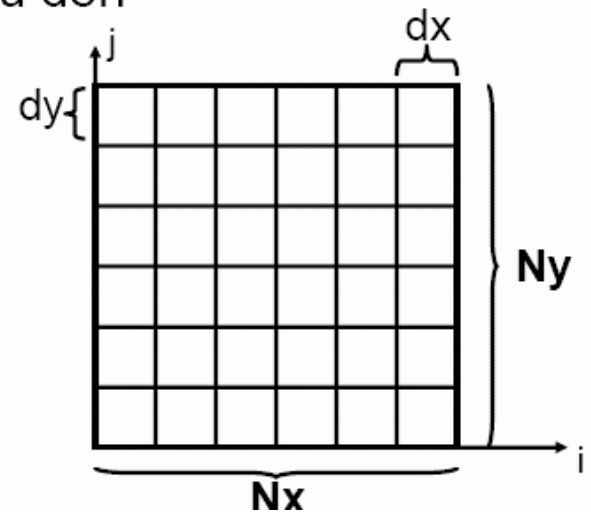
Allgemeine Gitterstrukturen:



(Rezk-Salama, o.J.)

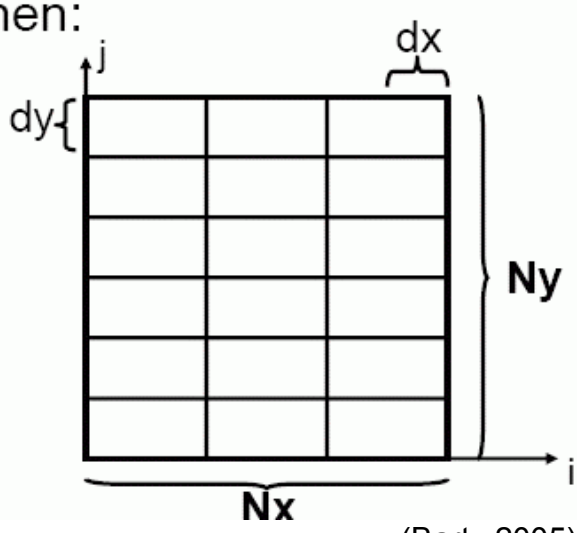
Detaillierte Betrachtung – Kartesische Gitter:

- Auch als **äquidistante** Gitter bekannt
- Gehören als Spezialfall zu den **strukturierten Gittern**
- Konstante Abstände im Gitter in jeder Dimension
- Üblich: dx = dy (= dz)
- Isotroper Datensatz
- Würfelzellen



Detaillierte Betrachtung – Uniforme Gitter:

- Wie kartesische Gitter, aber mit **unterschiedlichen Gitterabständen** in verschiedenen Dimensionen: dx != dy (!= dz)
- Gehört auch zu **strukturierten Gittern**
- Anisotroper Datensatz
- Quaderzellen



(Bartz 2005)

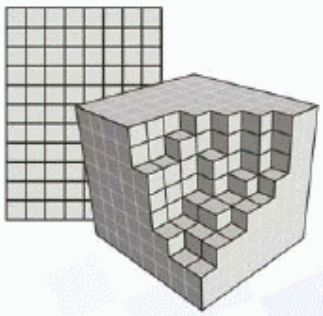
Uniforme Gitter

- *Strukturiertes Gitter*

  Was muß gespeichert werden?

  Zellen und Vertices implizit:

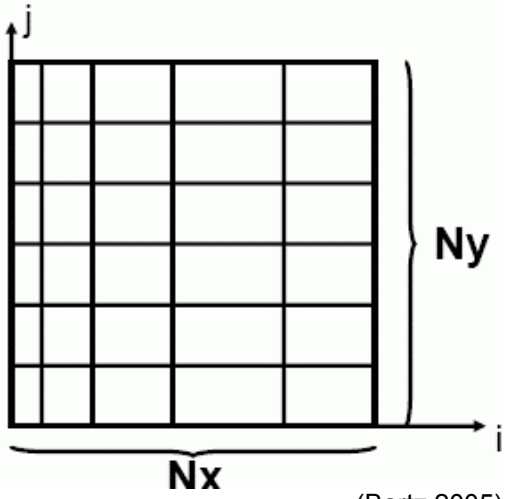
  - Dimension Nx, Ny, Nz:
    Anzahl der Abtastpunkte in x-, y- und z-Richtung
  - Zellgröße/Gitterweite Δx, Δy, Δz:
    Abstand der Abtastpunkte in x-, y- und z-Richtung
  - Datenwerte als Array mit Index n
    $n = Nx \cdot Ny \cdot z + Nx \cdot y + x$



(Rezk-Salama, o.J.)

Detaillierte Betrachtung – Nichtuniforme Gitter:

- Auch als **rectilineare** Gitter bekannt
- Wie uniforme Gitter, aber mit variierenden Gitterabständen (per Funktion angegeben)
- Gehört auch zu **strukturierten Gittern**
- Anisotroper Datensatz
- Quaderzellen unterschiedlicher Größen



(Bartz 2005)

Rectilineare Gitter

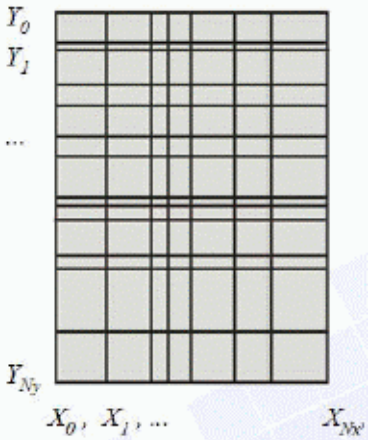
- *Strukturiertes Gitter*

  Was muß gespeichert werden?

  Zellen implizit, Vertices explizit:

  - Dimension Nx, Ny, Nz:
    Anzahl der Abtastpunkte in x-, y- und z-Richtung
  - Koordinaten der Linien/Ebenen:
    - $X_0, X_1, \ldots X_{Nx}$
    - $Y_0, Y_1, \ldots Y_{Ny}$
    - $Z_0, Z_1, \ldots Z_{Nz}$
  - Datenwerte als Array mit Index n

    $n = Nx \cdot Ny \cdot z + Nx \cdot y + x$

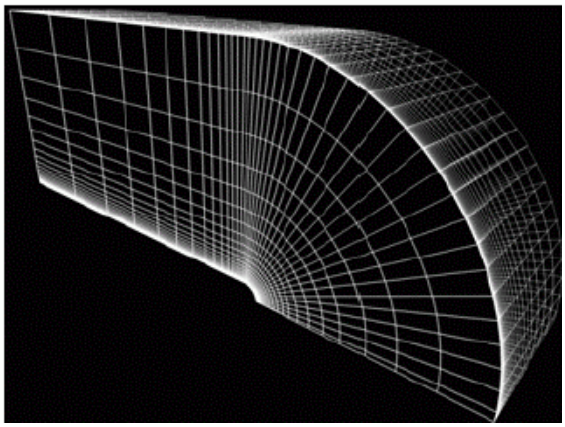


Eigenschaften nichtuniformer Gitter:

- Kartesische und uniforme Gitter als Spezialfall
- Gitter (und Zellen) sind **konvex**
- **Sichtbarkeitssortierung** ist fest (da konvex)
- **Einfache, kompakte** Handhabung durch **implizite** Adressierung
- Lokale Gitterverfeinung **schwierig/nicht möglich**
- Abbildung auf **komplexe Strukturen nicht immer möglich**

Detaillierte Betrachtung – Curvilineare Gitter:

- Auch als **gebogenen** Gitter bekannt
- Wie uniforme Gitter, aber mit variierenden Gitter**geometrie** (per Funktion angegeben)
- Gehört auch zu **strukturierten Gittern**
- Anisotroper Datensatz
- Quaderzellen unterschiedlicher Größen



(Bartz 2005)

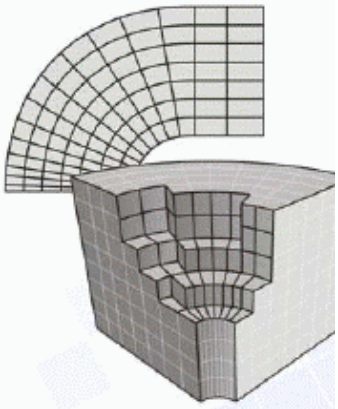
Curvilineare Gitter

- *Strukturiertes Gitter*

  Was muß gespeichert werden?

  Zellen implizit, Vertices explizit:

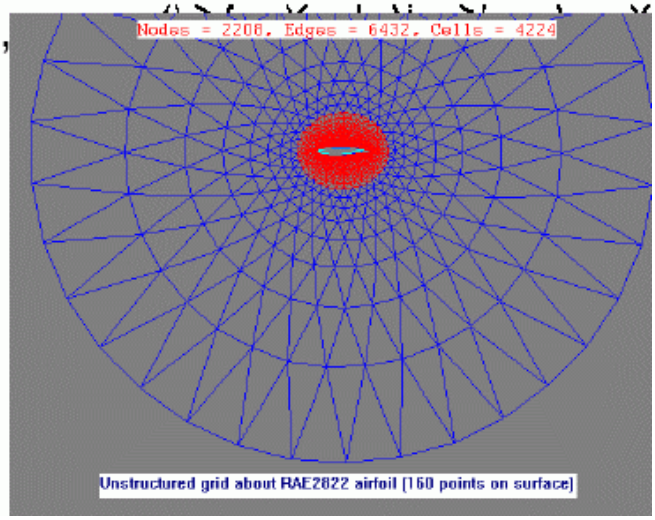
  - Dimension Nx, Ny, Nz:
    Anzahl der Abtastpunkte in x-, y- und z-Richtung
  - Liste der Gitterpunkte P_{ijk}:
    x-, y- und z-Koordinaten der Gitterpunkte
  - Datenwerte als Array mit Index n
    $n = Nx \cdot Ny \cdot z + Nx \cdot y + x$



(Rezk-Salama, o.J.)

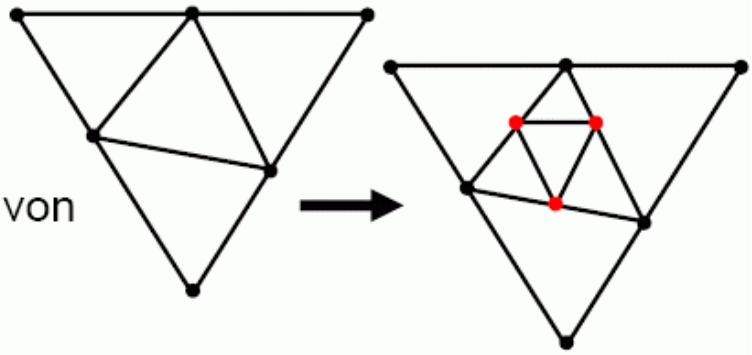
Detaillierte Betrachtung – Unstrukturierte Gitter:

- Auch als **irreguläre** Gitter bekannt
- Variierende Topologie, variierende Gitter-positionen
- Variierende Zelltypen (Tetras, Hexas, Prismas, …)
- Hybrid Gitter ist Spezialform mit verschiedenen Gittertypen



Eigenschaften unstrukturierter Gitter:

- **Lokale Verfeinung** gut möglich
- **Gute Adaption** an unregelmäßige Datenräume (variierende Zelltypen, variierende Zellgrößen)
- **Wohldefinierte Interpolation** innerhalb der Zellen
- Aber **aufwendig** in Speicherung und Auswertung
- Repräsentation durch **explizite** Speicherung von **Nachbarschafts-beziehungen**



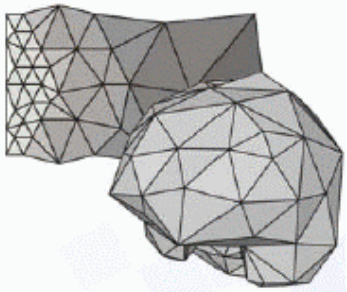
(Bartz 2005)

Unstrukturiertes Gitter

Was muß gespeichert werden?

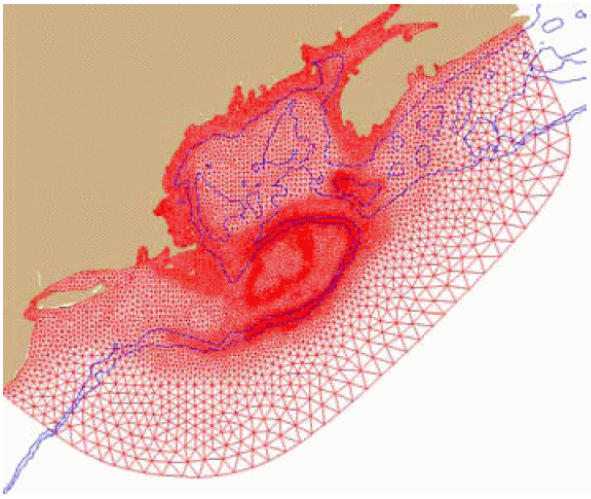
Zellen und Vertices explizit:

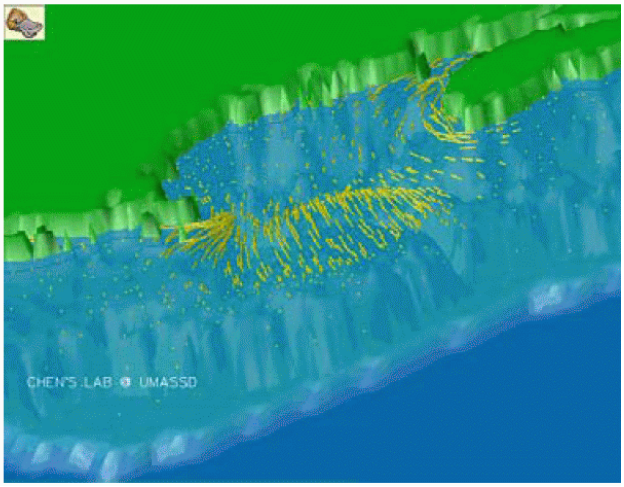
- Anzahl der Gitterpunkte N
- Anzahl der Zellen Z
- Liste der Gitterpunkte P_{ijk}:
  x-, y- und z-Koordinaten der Gitterpunkte
- Liste der Zellen:
  z.B. Tetraeder: 4 Indices der Gitterpunkte



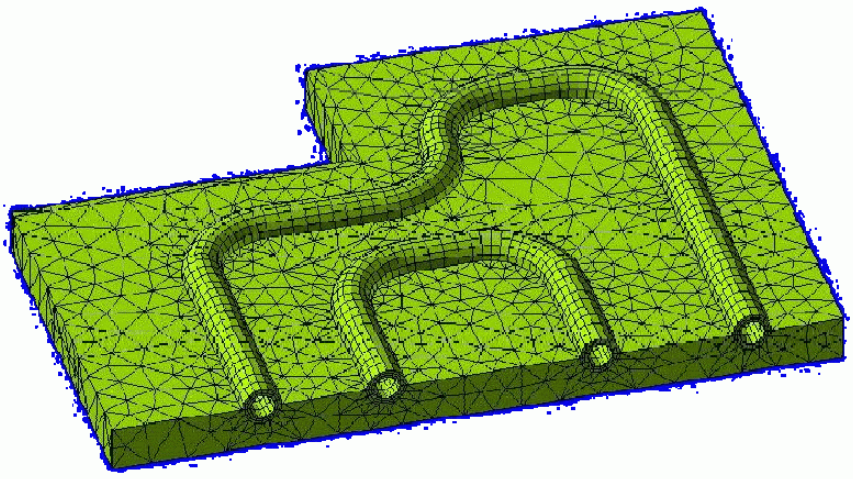
(Rezk-Salama, o.J.)

Beispiele unstrukturierter Gitter:





Detaillierte Betrachtung – Hybrid-Gitter:

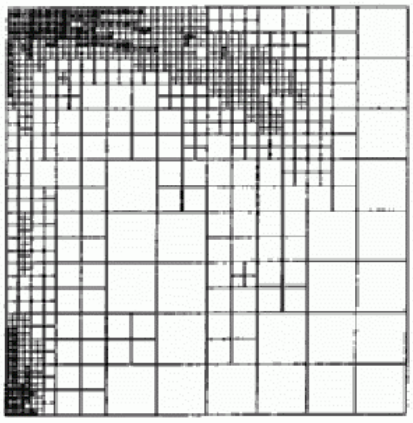


(Bartz 2005)

Sonderfall "Multi-Gitter":

Lokale Gitterverfeinerung

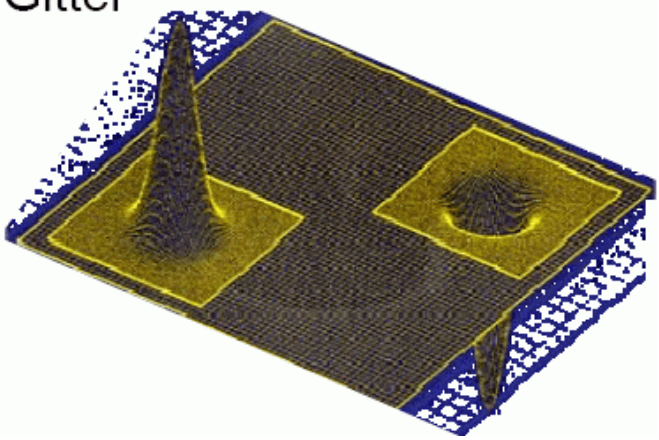
- Mesh subdivision and / or Multiblock



(Woessner, o.J.)

Detaillierte Betrachtung – Multi-Gitter:

- Kombination von Gittern **verschiedener Auflösung** und ggf. verschiedener Gittertypen
- Oft auch **überlagerte** Gitter
- Besser **Auflösung/ Darstellung**, wo sie benötigt wird
- **Anschlussregionen** können knifflig sein



(Bartz 2005)

Zusammenfassung Gittertypen

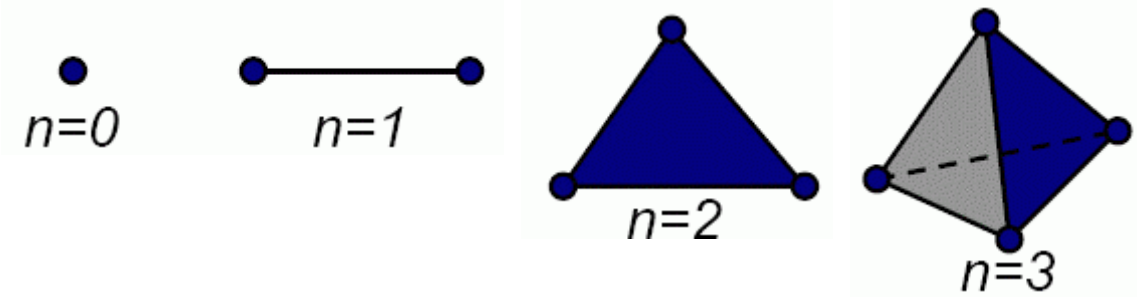
- *Uniform rectilinear:*
  Vorteile: einfach, effizienter Speicherzugriff, Zellsuche ist trivial,
  Nachteile: überall gleiche Auflösung, keine lokale Verfeinerung möglich
- *Curvilinear:*
  Vorteile: flexibler als rectilineare Gitter
  Nachteile: Gitterstruktur muß zusätzlich gespeichert werden, Algorithmus zur Zellsuche erforderlich
- *Unstrukturiert:*
  Vorteile: sehr hohe Flexibilität, lokale Verfeinerung möglich
  Nachteile: Position jedes Datenpunktes und Zellen müssen explizit gespeichert werden, Zellsuche erforderlich

(Rezk-Salama, o.J.)

Allgemeine Gitter setzen sich oft aus **Simplizes** zusammen:

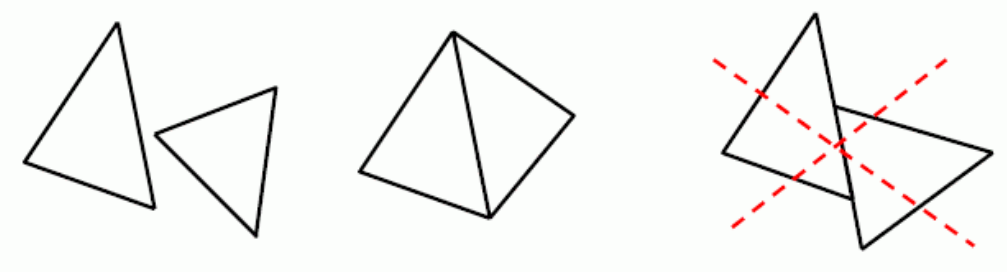
- Ein Simplex im R^n besteht aus einer **Konvexen Hülle** aus n+1 affin unabhängigen Punkten
- R^0: Punkte, R^1: Linien, R^2: Dreiecke, R^3: Tetraeder, …
- Partitionen durch Simplizes werden allgemein als Triangulierungen bezeichnet (auch bei n>2)

(Bartz 2005)



Begriff des "simplizialen Komplexes":
Menge von Simplices mit bestimmten Eigenschaften

- Ein simplizialer Komplex C besteht aus Simplices, die entweder eine Seite (im R^2 eine Kante oder Ecke) oder nichts gemeinsam haben,
- jede Seite (Kante) eines Simplex aus C gehört auch zu C.

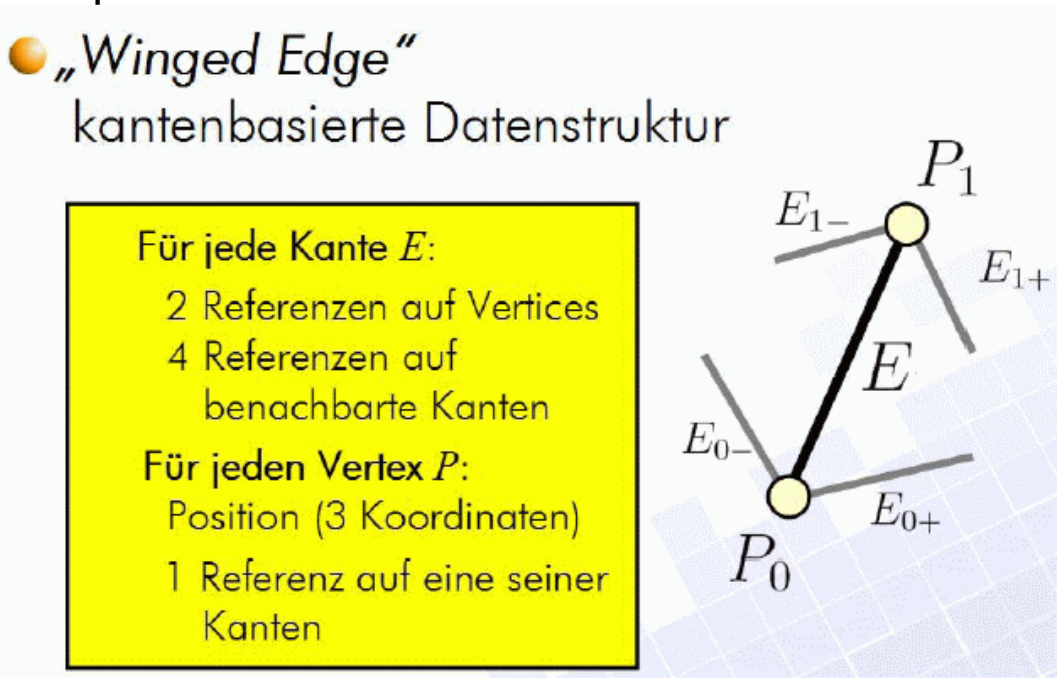


Datenstrukturen zum Abspeichern unstrukturierter Gitter:

Speicherung von Nachbarschaftsbeziehungen
- gemeinsame Seite / Polygon
- gemeinsame Kante
- gemeinsame Ecke

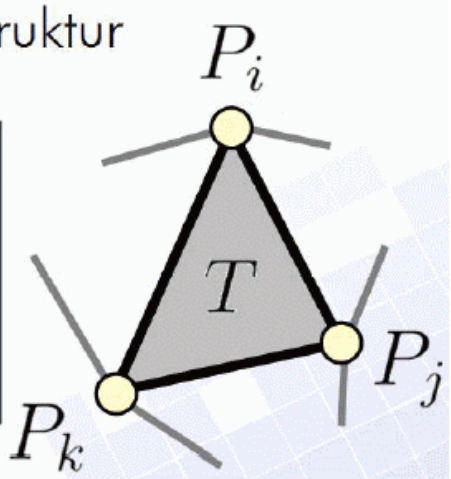
mehrere Varianten möglich

Beispiel: Dreiecksnetz



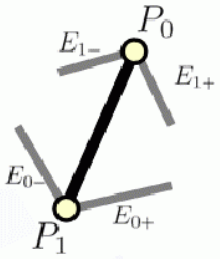
- *„Shared Vertex“*
  flächenbasierte Datenstruktur

Für jeden Vertex P:
Position (3 Koordinaten)
Für jedes Dreieck T:
3 Referenzen auf Vertices

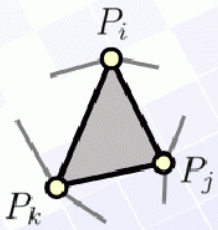


Welche Datenstruktur ist effizienter?

Faustregel: Dreiecksnetz mit n Vertices, hat ca. 3n Kanten und 2n Flächen



- *„Winged Edge“*
  benötigt ca. (3+1)n + 6·3n = **22n** Werte
- *„Shared Vertex“*
  benötigt ca. 3n + 3·2n = **9n** Werte



ABER: Winged Edge ermöglicht schnellere Algorithmen, da die Traversierung (z.B. Aufzählung aller Kanten zu einem geg. Vertex, aller Vertices zu einem geg. Dreieck, etc.) in ***konstanter*** Zeit möglich ist!

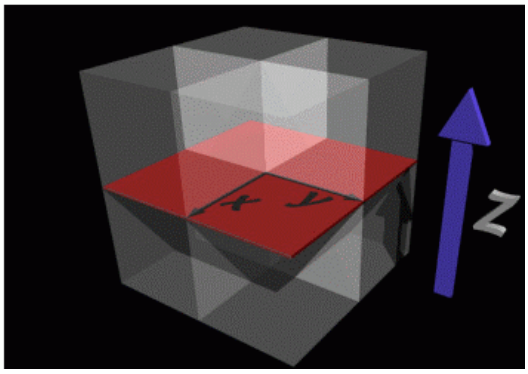
(Rezk-Salama, o.J.)

Koordinatensysteme:
- verschiedene Systeme in Gebrauch
- kartesisches System am häufigsten
- wichtige Alternativen: Zylinder-, Kugelkoordinaten

Kartesisches Koordinatensystem:

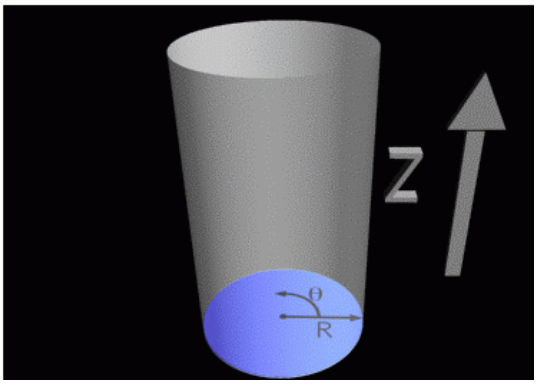
- Setzt sich im R^n aus n Einheitsvektoren als Basis zusammen:
  mit n=3

$$\vec{e}_1, \vec{e}_2, \vec{e}_3$$



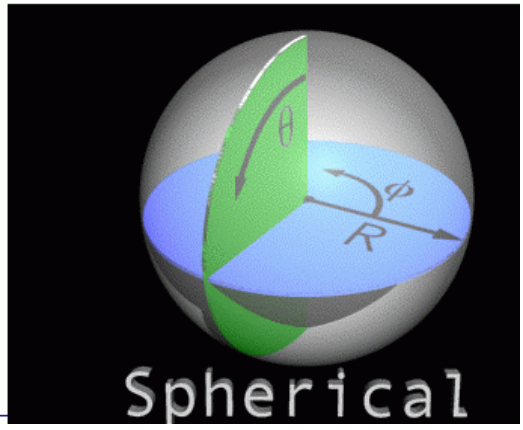
Zylinder-Koordinatensystem:

- Basis im R^3 setzt sich aus Winkel Θ, Höhe z bzw. h und Radius R zusammen
- x = R cos (Θ)
- y = R sin (Θ)
- z = z
- $\Theta = \arctan(\frac{y}{x})$
- $R = \sqrt{(x^2 + y^2)}$



Kugel-Koordinatensystem:

- Basis im R^3 setzt sich aus den Winkeln Θ, Φ und dem Radius R zusammen
- x = R sin(Θ) cos(Φ)
- y = R sin(Θ) sin(Φ)
- z = R cos(Θ)
- $\Theta = \arccos(\frac{z}{R})$
- $\Phi = \arccos(\frac{y}{x})$
- $R = \sqrt{(x^2 + y^2 + z^2)}$



(Bartz 2005)

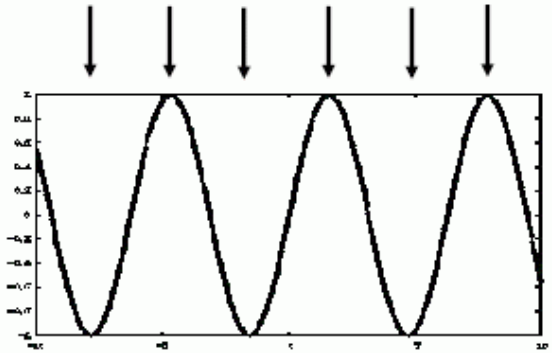
5.2 Abtastung

Beim Übergang von kontinuierlichen Daten zu einem diskreten Gitter findet Abtastung statt (Diskretisierung).

Signaltheorie: Diskretisierung unterliegt dem *Abtasttheorem*
- dessen Nicht-Beachtung führt zu Aliasing (artifiziellen Mustern), z.B. Treppenstufen, Schein-Schwingungen, Moiré-Muster.

Abtasttheorem (Nyquist, Shannon):

Für eine hinreichende Rekonstruktion eines Signals, muss man es **mindestens doppelt so schnell** abtasten (**Nyquist - Rate**)



1x +1x => 2x

→ Abtastrate mindestens doppelt so hoch wählen wie die Abmessung des kleinsten interessierenden Details (besser dreimal so hoch)

(siehe auch Vorlesung "Bildanalyse und Bildverstehen"!)

für manche Datensätze von Interesse:
Übergang in den Frequenzraum
d.h. Darstellung des Datensatzes mit Hilfe periodischer Funktionen unterschiedlicher Frequenz

- Fourier-Analyse (Fourier-Transformation) wird zur Abbildung zwischen **Ortsraum** (Spatial Domain) und **Frequenzraum** (Frequency Domain) verwendet
- Statt Ortsraum f(x) wird of Zeitraum f(t) verwendet
- Ortsraum f(x) $\Leftrightarrow$ Frequenzraum F(ν)

Forwärts-FT

$$f(x) = \int_{-\infty}^{\infty} F(\nu) e^{-2\pi i \nu x} d\nu \quad \Leftrightarrow \quad F(\nu) = \int_{-\infty}^{\infty} f(x) e^{2\pi i \nu x} dx$$

inverse FT

Fourier-Darstellung:

$$f(x) = a_0 + \sum_{\nu>1} \left(a_\nu \cos(2\pi\nu x) + b_\nu \sin(2\pi\nu x)\right)$$

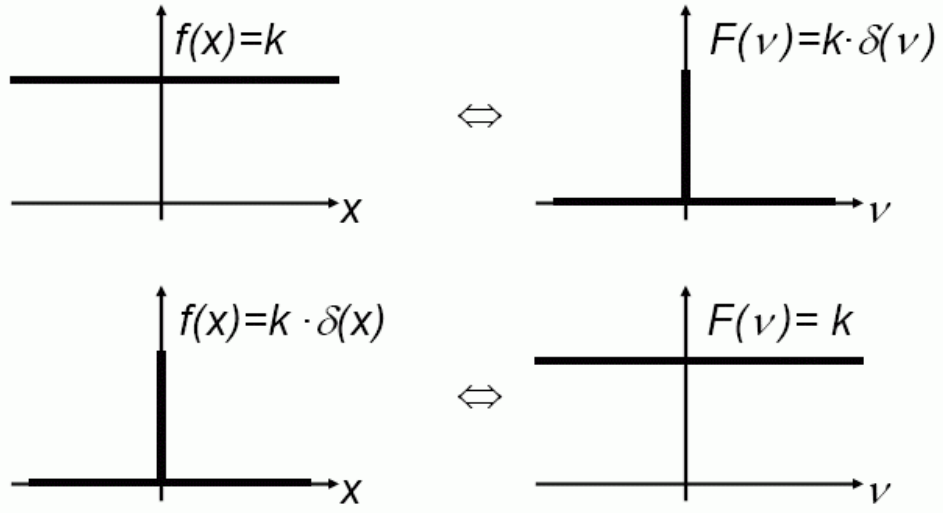
Berechnung der Koeffizienten mit Hilfe von

$$a_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) dx$$

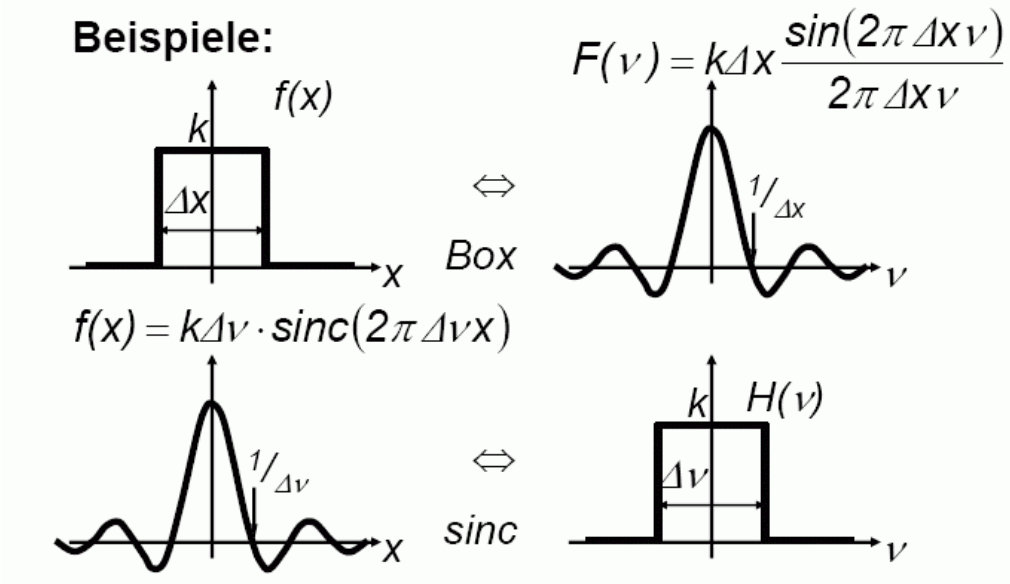
$$a_\nu = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(2\pi\nu x) dx$$

$$b_\nu = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(2\pi\nu x) dx$$

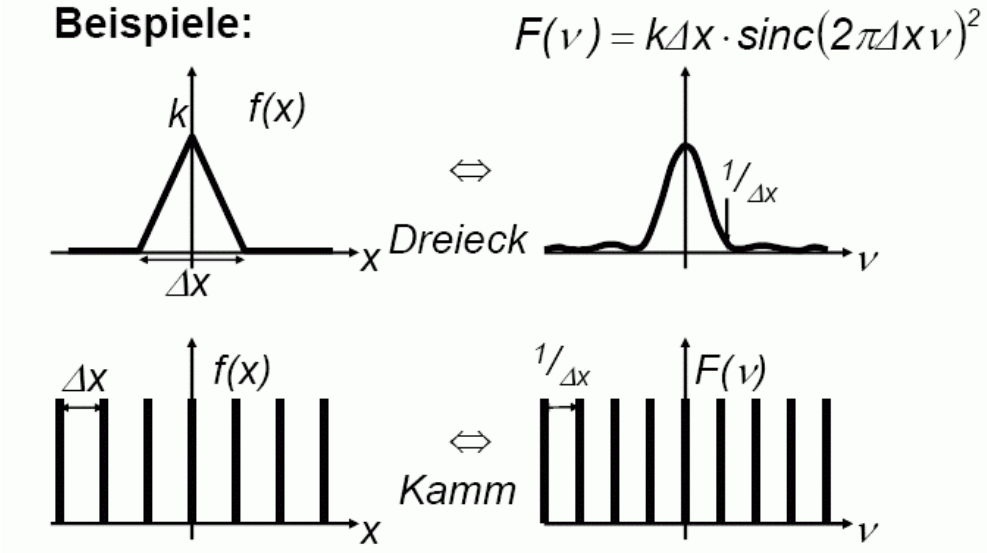
Beispiele:



Beispiele:



Beispiele:



(Bartz 2005)

5.3 Gradienten

In orts- oder zeitbezogenen Datensätzen interessiert man sich oft für die *Veränderung* der Daten (örtlich / zeitlich).
→ Betrachtung von Ableitungen / Gradienten.

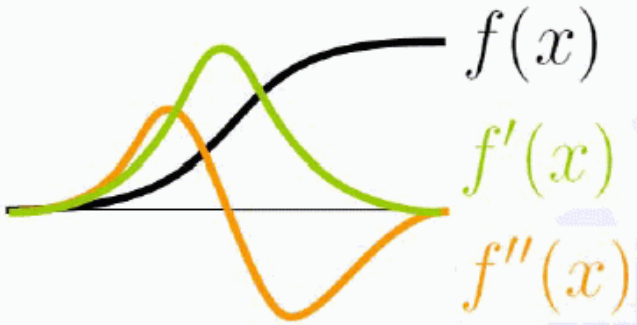
- Ableitungen im 1D: $f'(x) = \frac{df(x)}{dx}$

  Eigenschaften der Ableitung:

  $f'(x)$ beschreibt die Steigung der Kurve $f(x)$

  Lokale Extrema: $f(x) = \max \quad \Rightarrow \quad f'(x) = 0$

- Kantendetektion:



$|f'(x)| = \max$

$f''(x) = 0$

$\Rightarrow$ Kante in $f(x)$

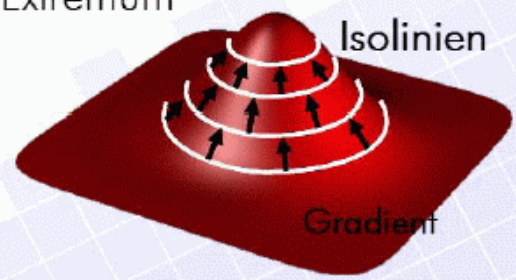
- 1.Ableitung im 2D:

  Der Gradient $\text{grad} f(x,y) = \begin{pmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{pmatrix}$

Eigenschaften des Gradientenvektors:

- zeigt in die Richtung des größten Anstiegs
- steht senkrecht auf den Isolinien
- ist gleich Null bei lokalem Extremum

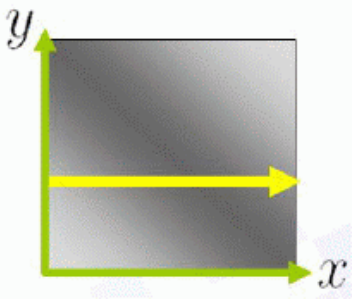
Beispiel:
2D Höhenfeld



Richtungsableitung

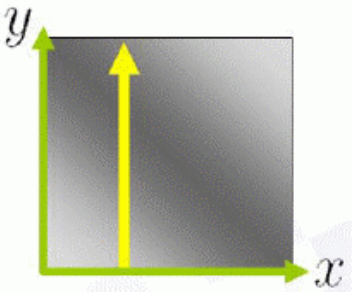
Partielle Ableitung im 2D:

- in x-Richtung: $\frac{\partial f(x,y)}{\partial x}$



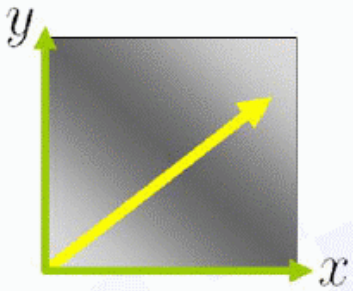
Partielle Ableitung im 2D:

- in x-Richtung: $\frac{\partial f(x,y)}{\partial x}$
- in y-Richtung: $\frac{\partial f(x,y)}{\partial y}$



Partielle Ableitung im 2D:

- in x-Richtung: $\frac{\partial f(x,y)}{\partial x}$
- in y-Richtung: $\frac{\partial f(x,y)}{\partial y}$
- in beliebige Richtung $\vec{w} = \begin{pmatrix} w_x \\ w_y \end{pmatrix}$



$$\begin{aligned} \frac{\partial f(x,y)}{\partial \vec{w}} &= w_x \frac{\partial f(x,y)}{\partial x} + w_y \frac{\partial f(x,y)}{\partial y} \\ &= \langle \vec{w} \circ \mathrm{grad} f(x,y) \rangle \end{aligned}$$

Ableitungen im Dreidimensionalen:

- Gradientenvektor:

$$\operatorname{grad} f(x,y,z) = \begin{pmatrix} \frac{\partial f(x,y,z)}{\partial x} \\ \frac{\partial f(x,y,z)}{\partial y} \\ \frac{\partial f(x,y,z)}{\partial z} \end{pmatrix}$$

- Richtungsableitung in Richtung $\vec{w} = \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix}$:

$$\begin{aligned} \frac{\partial f(x,y)}{\partial \vec{w}} &= w_x \frac{\partial f(x,y,z)}{\partial x} + w_y \frac{\partial f(x,y,z)}{\partial y} + w_z \frac{\partial f(x,y,z)}{\partial z} \\ &= \langle \vec{w} \circ \operatorname{grad} f(x,y,z) \rangle \end{aligned}$$

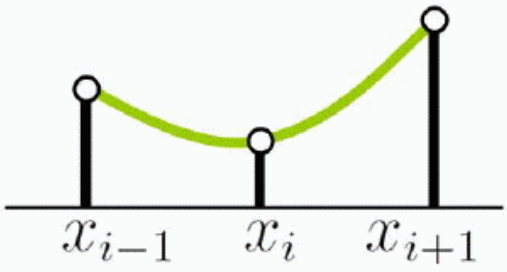
Differenzieren auf Gittern

Wie kann ich Ableitungen berechnen auf diskreten Daten?

- Möglichkeit 1: Approximiere die diskreten Daten (lokal) durch eine differenzierbare Funktion und differenziere diese.
- Möglichkeit 2: Ersetze das Differential durch finite Differenzen

$$f'(x) = \frac{df(x)}{dx} \mapsto \frac{\Delta f(x)}{\Delta x}$$

Möglichkeit 1: Curve Fitting



Bestimme die Ableitung $f'(x_i)$ an der Stelle x_i

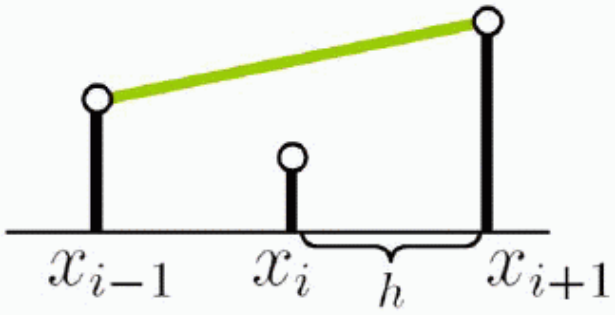
1. Finde ein Polynom 2.Grades,

$$f(x) = ax^2 + bx + c$$

das die diskreten Werte lokal approximiert.

2. Differenziere dieses Polygon an der Stelle x_i

Möglichkeit 2: Finite Differenzen



Bestimme die Ableitung $f'(x_i)$ an der Stelle x_i

		Fehlerordnung
Vorwärtsdifferenzen:	$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{h}$	$o(h)$
Rückwärtsdifferenzen:	$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{h}$	$o(h)$
Zentrale Differenzen:	$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$	$o(h^2)$

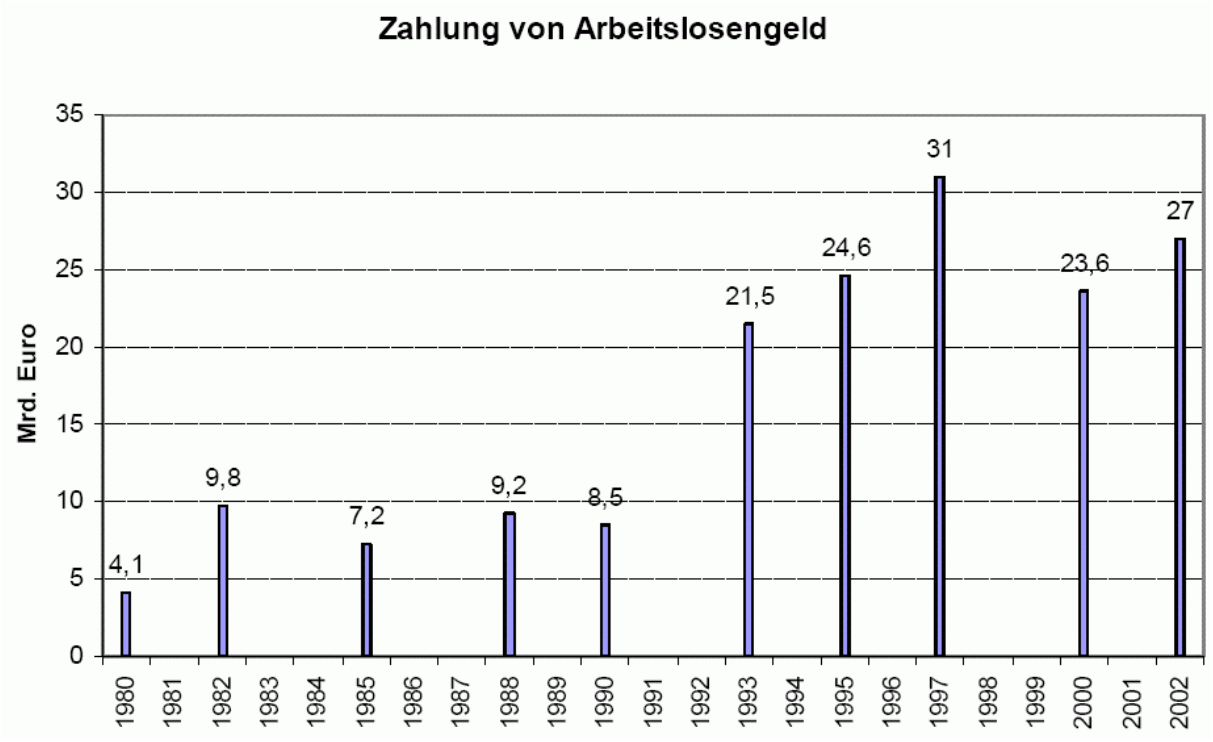
(Rezk-Salama, o.J.)

5.4 Interpolation

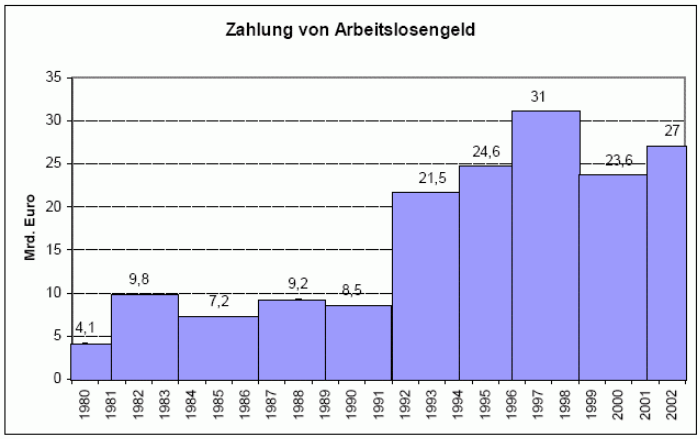
> In nichts zeigt sich der Mangel an mathematischer Bildung mehr, als in einer übertrieben genauen Rechnung
>
> (Carl Friedrich Gauss, dt. Mathematiker, 1777-1855)

Interpolation:
stetige Fortsetzung eines Datensatzes an Punkten, wo keine Daten erhoben wurden (wo aber in einer Umgebung Daten erhoben wurden).

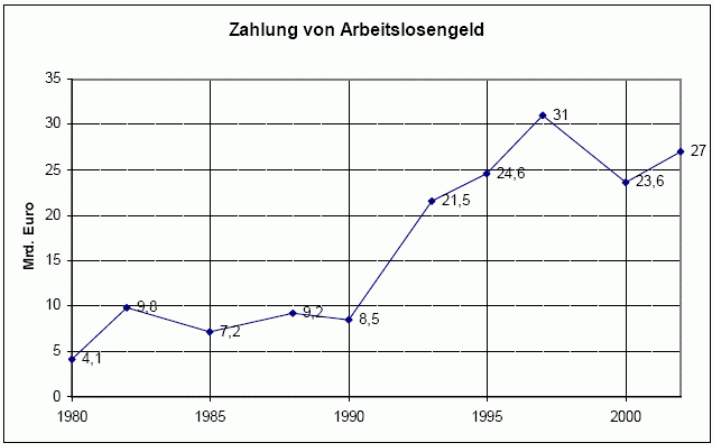
Beispiel: Datensatz durch Balken visualisiert, keine Interpolation:



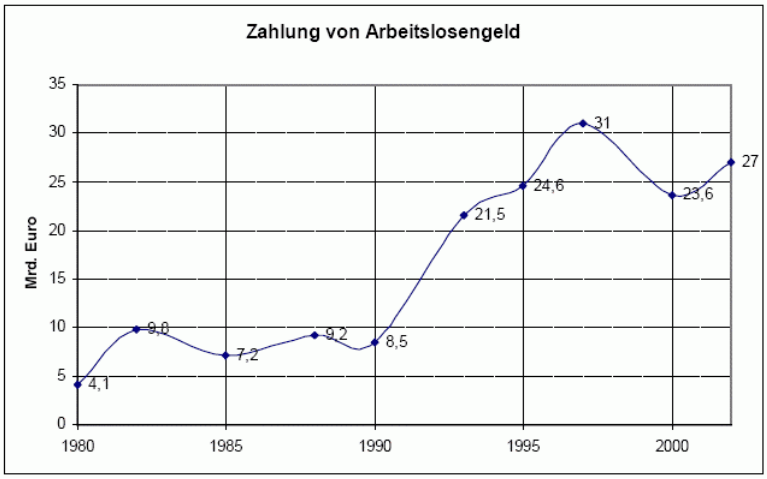
Datensatz: Balken, nächste Nachbarregion (konstante Interpolation, nearest neighbor)



Datensatz: linear interpoliert

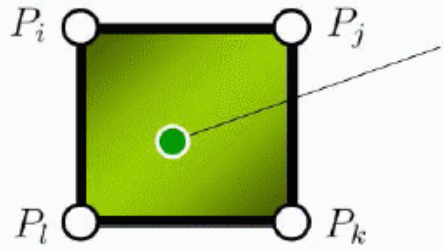


Datensatz: quadratisch interpoliert



(Bartz 2005)

- An den Datenpunkten P_i sind Datenwerte gegeben.



Wie bestimme ich einen Datenwert für einen Punkt, der innerhalb einer Zelle liegt?

Interpolation

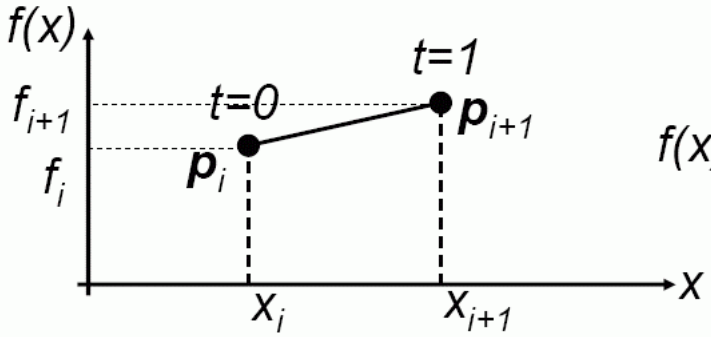
d.h. approximative Rekonstruktion des Signals

(Rezk-Salama, o.J.)

Lineare Interpolation:

1-dimensionaler Fall

Geg.: Datenpaare $\mathbf{p}_i = (x_i, f_i)$, mit i=0..n-1, $x_i < x_j$ für i<j



$$f(x) = \frac{x_{i+1}-x}{x_{i+1}-x_i} f_i + \frac{x-x_i}{x_{i+1}-x_i} f_{i+1}$$

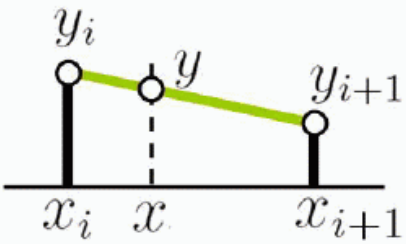
Idee: Parametrisierung des Liniensegments mit Hilfe eines Parameters t:

$$L(\mathbf{p}_i, t, \mathbf{p}_{i+1}) = (1-t)\mathbf{p}_i + t\mathbf{p}_{i+1}, \quad t \in [0,1]$$

$$t(x) = \frac{x - x_i}{x_{i+1} - x_i}, \quad x \in [x_i, x_{i+1}]$$

(Bartz 2005)

Lineare Interpolation (1D)



Lineares *Blenden*,

$$f(x) = \alpha\, y_{i+1} + (1-\alpha) y_i$$

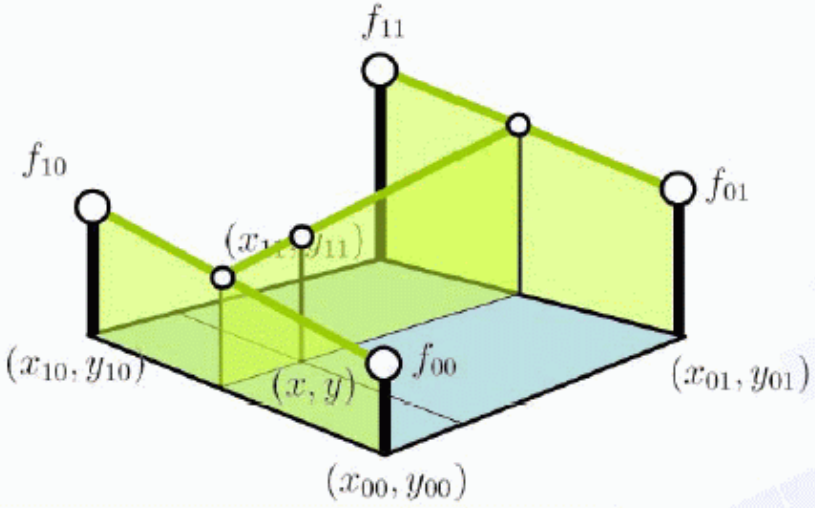
mit

$$\alpha = \frac{x - x_i}{x_{i+1} - x_i}$$

$$x = x_i \Rightarrow \quad \alpha = 0 \quad \Rightarrow f(x) = y_i$$

$$x = x_{i+1} \Rightarrow \quad \alpha = 1 \quad \Rightarrow f(x) = y_{i+1}$$

Bilineare Interpolation (2D)



$$f_0 = \alpha\, f_{10} + (1-\alpha) f_{00}$$

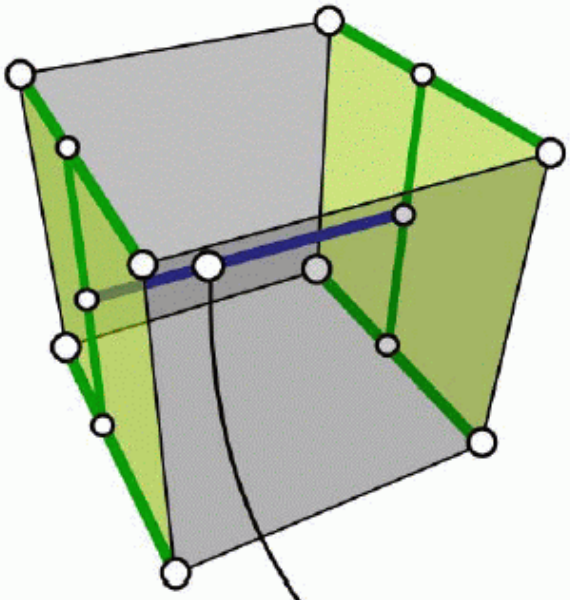
$$f_1 = \alpha\, f_{11} + (1-\alpha) f_{01}$$

$$f(x,y) = \beta\, f_1 + (1-\beta) f_0$$

$$\alpha = \frac{x - x_{00}}{x_{10} - x_{00}}$$

$$\beta = \frac{y - y_{00}}{y_{01} - y_{00}}$$

Trilineare Interpolation (3D)



4 lineare Interpolationen

$$
\begin{aligned}
f_{00} &= \alpha\, f_{001} + (1-\alpha) f_{000} \\
f_{01} &= \alpha\, f_{011} + (1-\alpha) f_{010} \\
f_{10} &= \alpha\, f_{101} + (1-\alpha) f_{100} \\
f_{11} &= \alpha\, f_{111} + (1-\alpha) f_{110}
\end{aligned}
$$

2 bilineare Interpolationen

$$
\begin{aligned}
f_{0} &= \beta\, f_{01} + (1-\beta) f_{00} \\
f_{1} &= \beta\, f_{11} + (1-\beta) f_{10}
\end{aligned}
$$

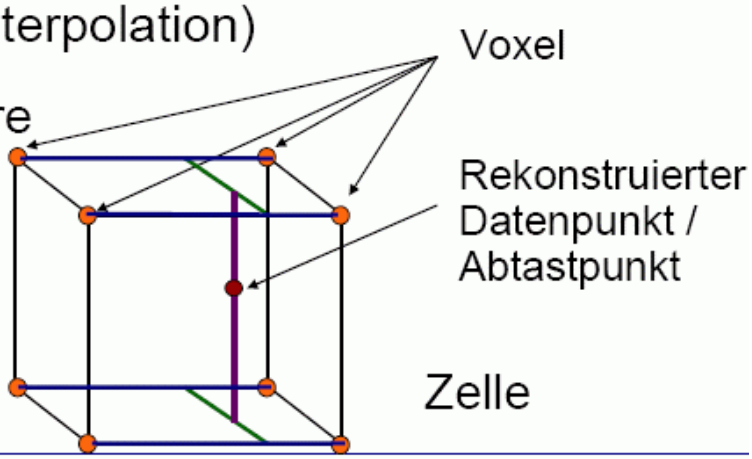
1 trilineare Interpolation

$$
f = \gamma\, f_{1} + (1-\gamma) f_{0}
$$

(Rezk-Salama, o.J.)

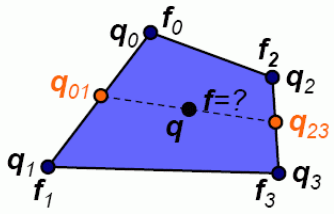
Trilineare Volumeninterpolation:

- Vier **lineare Interpolationen** auf den Kanten
- Darauf zwei lineare Interpolationen (zwei **bilineare** Interpolation)
- Darauf eine lineare Interpolation

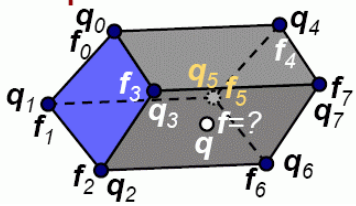


(Bartz 2005)

Vierecke:

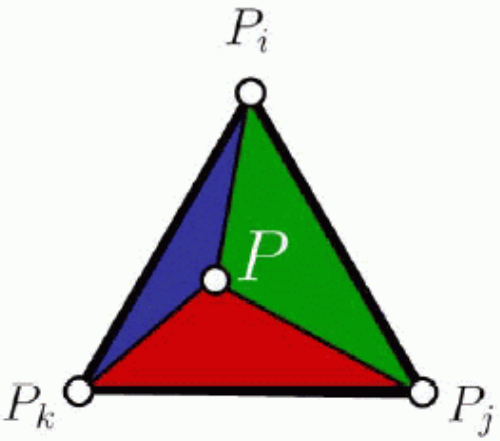


Hexaeder / Trilineare Interpolation:



(Bartz 2005)

Dreieckige Zellen (2D)



Der Punkt P kann ausgedrückt werden als

$$P = a_i P_i + a_j P_j + a_k P_k$$

mit

$$a_i + a_j + a_k = 1$$

$$A = Area(\triangle P_i P_j P_k)$$

$$a_i = \frac{1}{A} Area(\triangle P P_j P_k)$$

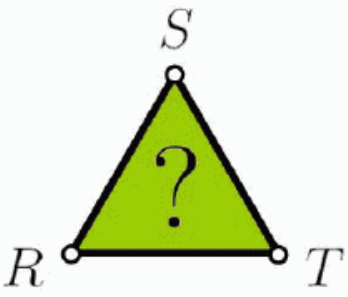
$$a_j = \frac{1}{A} Area(\triangle P_i P P_k)$$

$$a_k = \frac{1}{A} Area(\triangle P_i P_j P)$$

Schwerpunktskoordinaten (Baryzentrische Koordinaten)

$$a_i = a_j = a_k = \frac{1}{3}$$

$\Rightarrow P$ ist Schwerpunkt

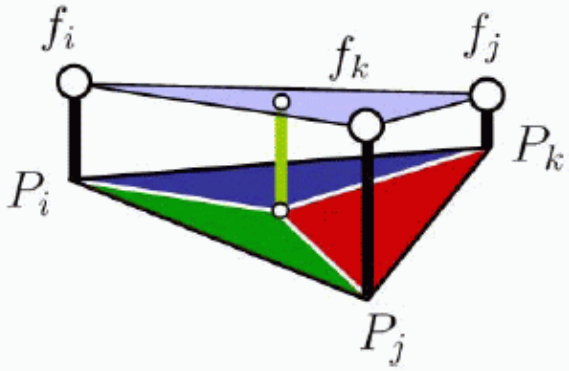


- Wie bestimme ich den Flächeninhalt eines Dreiecks $Area(\triangle RST)$?

$$R = \begin{pmatrix} R_x \\ R_y \end{pmatrix} \quad S = \begin{pmatrix} S_x \\ S_y \end{pmatrix} \quad T = \begin{pmatrix} T_x \\ T_y \end{pmatrix}$$

$$\begin{vmatrix} 1 & 1 & 1 \\ R_x & S_x & T_x \\ R_y & S_y & T_y \end{vmatrix} = \pm 2 \cdot Area(\triangle RST)$$

Lineare Interpolation im Dreieck



Bestimme die baryzentrischen Koordinaten des Punkts P

$$P = a_i P_i + a_j P_j + a_k P_k$$

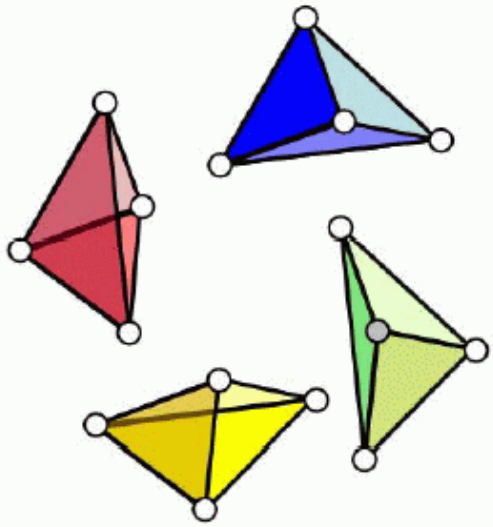
Bestimme den Funktionswert mit den gleichen Gewichten

$$f = a_i f_i + a_j f_j + a_k f_k$$

Anmerkung: Auch Punkte P außerhalb des Dreiecks lassen sich in baryzentrischen Koordinaten ausdrücken. Die Gewichte sind dann nicht mehr alle positiv. Es gilt aber nach wie vor $a_i + a_j + a_k = 1$

Tetraeder

Baryzentrische Koordinaten im Tetraeder

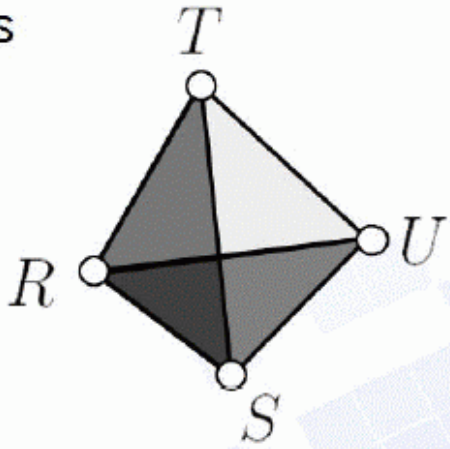


1. Verbinde den inneren Punkt mit allen Randpunkten.

Es entstehen 4 Tetraeder als Unterteilung

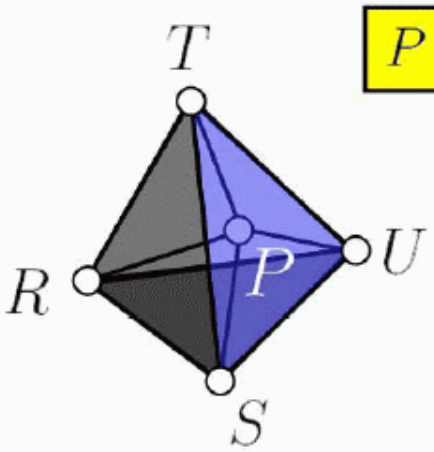
2. Das Volumen eines Teil-Tetraeders ist das baryzentrische Gewicht für den gegenüberliegenden Punkt

- Volumen eines Tetraeders $Vol\big(Tet(R,S,T,U)\big)$ mit Vertices R, S, T und U:



$$\begin{vmatrix} 1 & 1 & 1 & 1 \\ R_x & S_x & T_x & U_x \\ R_y & S_y & T_y & U_y \\ R_z & S_z & T_z & U_z \end{vmatrix} = \pm 6 \cdot Vol\big(Tet(R,S,T,U)\big)$$

Baryzentrische Koordinaten im Tetraeder



$$P = a_R R + a_S S + a_T T + a_U U$$

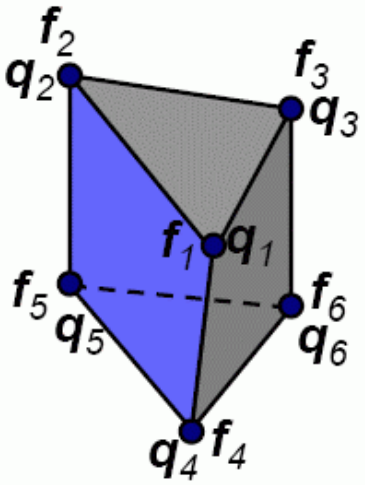
$$a_U = \frac{\begin{vmatrix} 1 & 1 & 1 & 1 \\ R_x & S_x & T_x & P_x \\ R_y & S_y & T_y & P_y \\ R_z & S_z & T_z & P_z \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 & 1 \\ R_x & S_x & T_x & U_x \\ R_y & S_y & T_y & U_y \\ R_z & S_z & T_z & U_z \end{vmatrix}}$$

Interpolation:

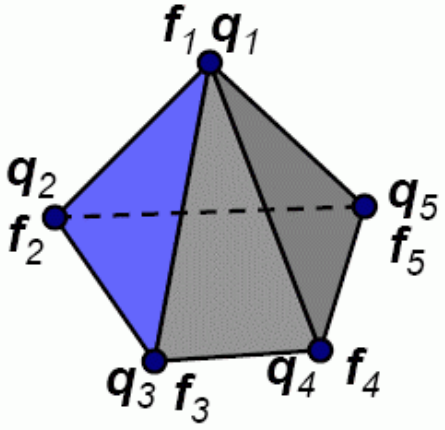
$$f = a_R f_R + a_S f_S + a_T f_T + a_U f_U$$

(Rezk-Salama, o.J.)

Interpolation in Prisma und Pyramide:



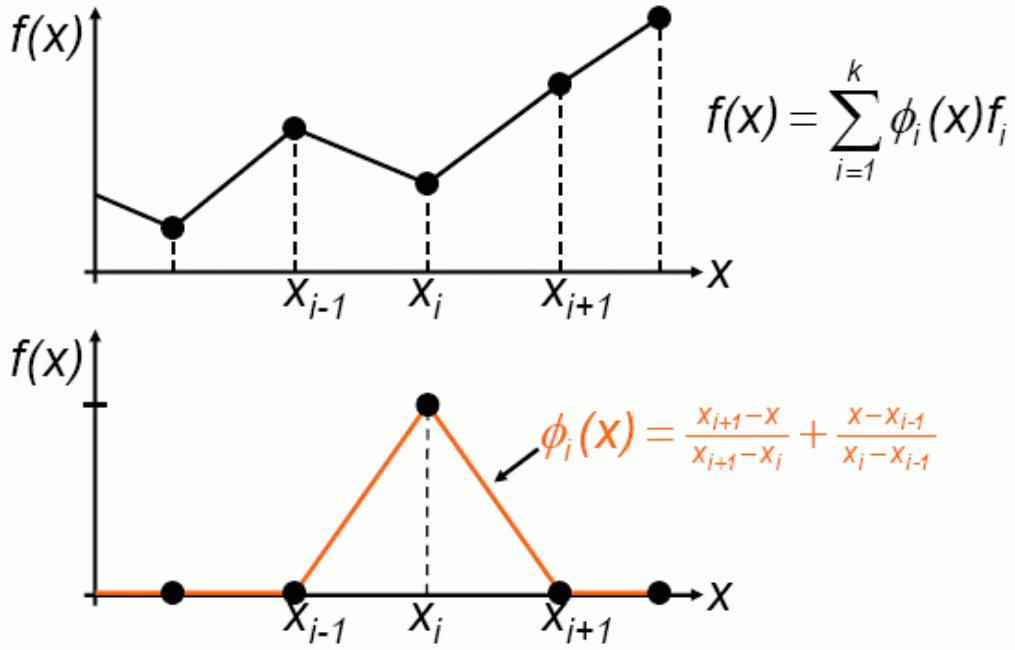
zweimal baryzentrisch
gefolgt von linear



bilinear auf Grundfläche
gefolgt von linear

Interpolation mit Basisfunktionen
(Gewichtungsfunktionen)

Beispiel: Lineare Interpolation



für die Gewichtungsfunktionen soll gelten...

Gewünschte Eigenschaften:

Glattheit

Lokalität

Affine Invarianz: $\Sigma_i \phi_i(x) = 1$

Unabhängigkeit: $\phi_i(x_j) = \delta_{ij}$

Normiertheit: $\int \phi_i(x)dx = 1$

Orthogonalität: $\int \phi_i(x)\phi_j(x)dx = \delta_{ij}$

Interpolationsproblem mit Basisfunktionen:

gegeben seien k Datenpunkte (q_i, f_i) $(i = 1, ..., k)$
und k Basisfunktionen ϕ_i

gesucht: k Koeffizienten c_i, die mittels der Funktion

$$f(q) = \Sigma\, c_i * \phi_i(q)$$

das Interpolationsproblem lösen,
d.h. f verläuft durch die Datenpunkte:

$$f(q_i) = f_i \quad \text{für alle } i = 1, ..., k.$$

Wenn die ϕ_i unabhängig sind, so gilt $c_i = f_i$.
Sonst muss ein lineares Gleichungssystem mit k Gleichungen und k Unbekannten gelöst werden:
Definiere die Matrix M mit den Einträgen $\phi_i(q_j)$, dann lautet das LGS: M***c** = **f**.

Basisfunktionen:

Polynombasen

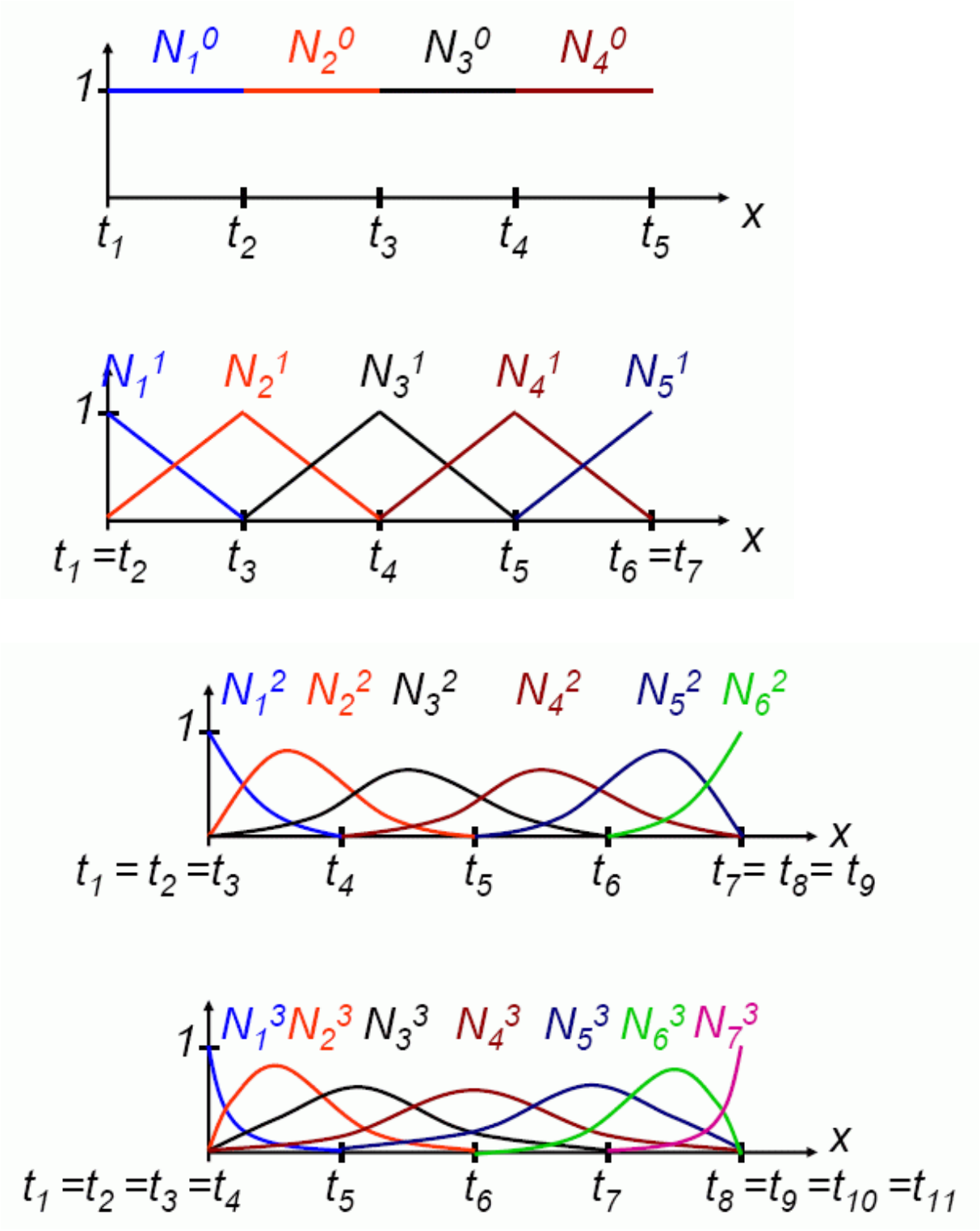
- Taylorbasis $\phi_i = x^i$
- Lagrangebasis $\phi_i = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$
- Newtonbasis $\phi_i = \prod_{j=1}^{k} (x - x_j)$
- Bernsteinbasis $\phi_i^k(t) = \binom{k}{i} (1-t)^{k-i} t^i \qquad [(1-t)+t]^k$

Spline-Basisfunktionen

- B-Spline-Basisfunktionen

(rekursiv definiert, siehe Vorlesung "Grundzüge der Computergrafik")

B-Spline-Basisfunktionen:



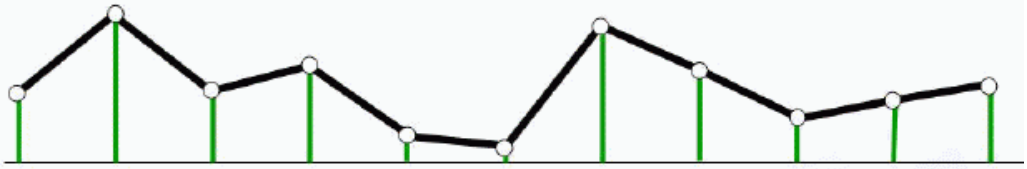
(Bartz 2005)

durch deren Verwendung glattere Interpolation!

anderer Ansatz zur glatten Interpolation:

Glatte Interpolation

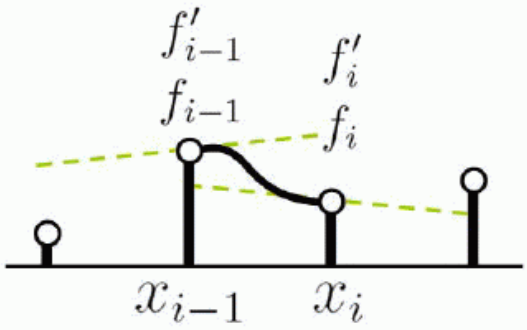
- (Stückweise) Lineare Interpolation ist C_0-stetig



- Wie bekomme ich eine „glatte“ Interpolation (C_1-stetig)?

- Lösung: z.B. stückweise kubische Interpolation

Catmull-Rom-Interpolation



1. Bestimme die Ableitungen f_i durch zentrale Differenzen

2. Bestimme eine (lokale) kubische Funktion, die sowohl die Funktionswerte f_i als auch die Ableitungen f'_i interpoliert

→ **Hermite-Interpolation**

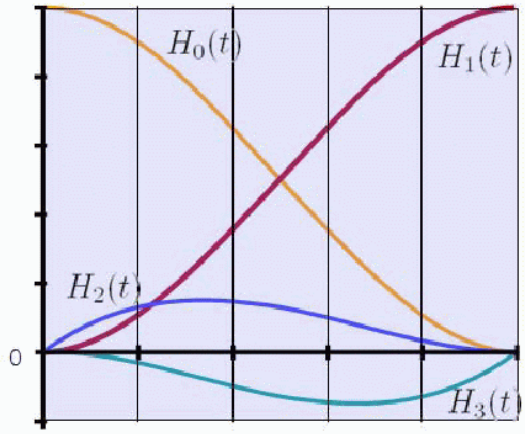
Hermite-Interpolation:

Gegeben: f_0, f_1, f'_0 und f'_1.

Gesucht: Polynom 3.Grades $f(t)$, sodaß gilt

$f(0) = f_0$	$f(1) = f_1$	$\frac{\partial}{\partial t} f(0) = f'_0$	$\frac{\partial}{\partial t} f(1) = f'_1$

		$f(0)$	$f(1)$	$f'(0)$	$f'(1)$
$H_0(t)$	$= (1-t)^2(2t+1)$	1	0	0	0
$H_1(t)$	$= t(1-t)^2$	0	1	0	0
$H_2(t)$	$= -t^2(1-t)$	0	0	1	0
$H_3(t)$	$= t^2(3-2t)$	0	0	0	1



Das gesuchte Polynom zur lokalen Interpolation bei gegebenem f_0, f_1, f'_0 und f'_1 ist:

$$f(t) = f_0\,H_0(t) + f_1\,H_1(t) + f'_0\,H_2(t) + f'_1\,H_3(t)$$

(Rezk-Salama, o.J.)

Scattered Data - Interpolation

"Scattered Data":
Daten von Beobachtungspunkten, die nicht in einem Verbund (regelmäßigem Gitter) liegen
(Geografie, Molekularbiologie, Mechanik, Ökologie, Meteorologie...)

3 Ansätze

Scattered Data Interpolation:

- Definiere radiale Basisfunktionen und löse Interpolationsproblem
  ➔ Shepard Interpolation
- Erzeuge möglichst optimales Gitter und Interpoliere zellbasiert
  ➔ Delaunay Triangulierung
- Erzeuge Gitter und verwende dieses zur Interpolation
  ➔ Natural Neighbor Interpolation

Shepard-Interpolation:

Idee: definiere für jeden Beobachtungspunkt eine Basisfunktion $\phi_i(\mathbf{q})$.

$$\phi_i(\mathbf{q}) = \frac{\lambda_i(\mathbf{q})}{\sum_j \lambda_j(\mathbf{q})} \qquad \lambda_i(\mathbf{q}) = \frac{1}{\|\mathbf{q} - \mathbf{q}_i\|^p}$$

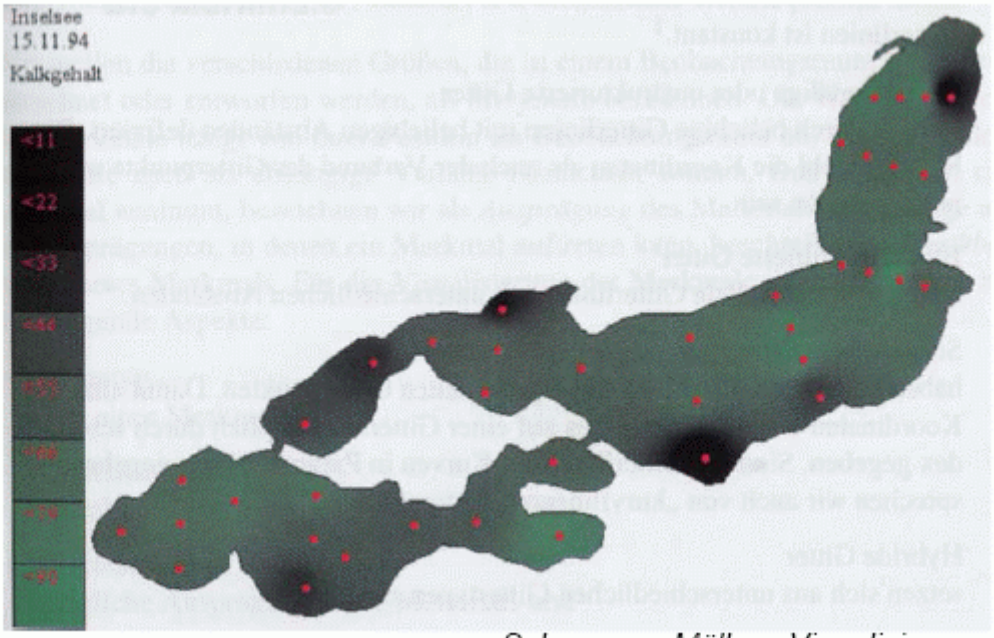
Interpolationsproblem von $\mathbf{f}(\mathbf{q}) = \Sigma_i \phi_i(\mathbf{q})\mathbf{f}_i$ ist mit der Definition $\mathbf{f}(\mathbf{q}_i) := \mathbf{f}_i = \lim_{\mathbf{q}\to\mathbf{q}_i} \mathbf{f}(\mathbf{q})$ automatisch gelöst

Eigenschaften der Shepard-Interpolation:

- $\mathbf{f}(\mathbf{q})$ ist Affinkombination der ϕ_i: $\mathbf{f} = A_k(\phi_i, \mathbf{f}_i)$
  - ➔ $\mathbf{f}(\mathbf{q})$ liegt in der konvexen Hülle der Datenpunkte
- für p>1 ist $\mathbf{f}(\mathbf{q}) \in \mathbf{C}^\infty$ und $\nabla_{\mathbf{q}} \phi_i(\mathbf{q}_i) = 0$
  - ➔ Datenpunkte sind Sattelpunkte
- globale Interpolation
  - ➔ jedes $\mathbf{f}(\mathbf{q})$ wird von allen Datenpunkten beeinflusst
- nur konstante Präzision, d.h. eine konstante Funktion wird exakt rekonstruiert

(Bartz 2005)

Scattered-Data: Shepard Interpolation



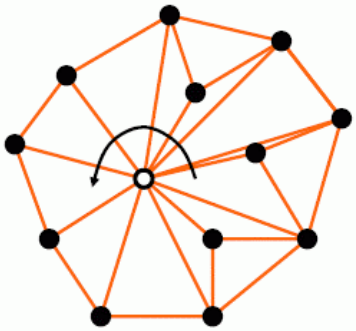
aus Schumann, Müller - Visualisierung

2. Ansatz:

Erzeugung eines Gitters aus Simplices zwischen den Datenpunkten, interpoliere dann in diesen Simplices linear.

Erzeugung eines Gitters: Beispiel 2D

Existiert immer eine Triangulierung?



– ja.

Welche Triangulierung ist geeignet?

- minimiere den maximalen Winkel
- maximiere den minimalen Winkel
- minimiere maximalen Umkreisradius
- maximiere minimalen Inkreisradius
- minimiere längste Kante
- maximiere kürzeste Kante

Delaunay-Triangu-lierung

dual

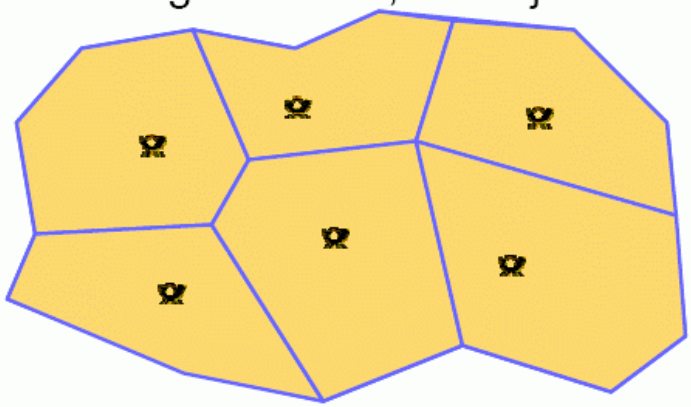
Voronoi Diagramm

zum Begriff des Voronoi-Diagramms
(vgl. auch Vorlesung "Algorithmische Geometrie"):

Postproblem:

Seien n Poststellen in einer Stadt gegeben.

Welches sind die Einzugsbereiche, wenn jeder zur nächstgelegenen Poststelle geht?

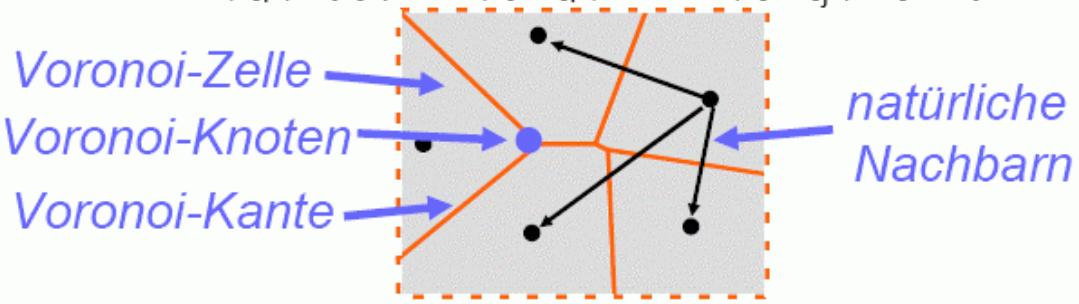


Definition Voronoi-Diagramm:

geg: eine Menge von Punkten $Q=\{\mathbf{q}_1,\ldots,\mathbf{q}_k\}$ aus $\mathbf{R}^n$ und eine Distanzfunktion dist($\mathbf{q}$,$\mathbf{r}$)

Def.: Das Voronoi-Diagramm Vor(Q) enthält für jeden Punkt $\mathbf{q}_i$ eine Zelle $V(\mathbf{q}_i)$ mit

$$V(\mathbf{q}_i)=\{\mathbf{q} \mid dist(\mathbf{q}, \mathbf{q}_i) < dist(\mathbf{q}, \mathbf{q}_j)\ \forall j \neq i\}$$

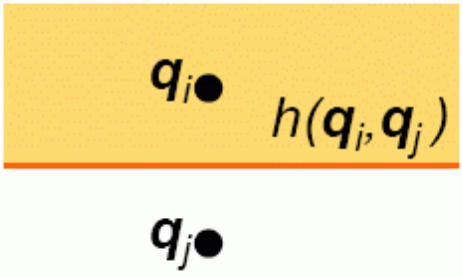


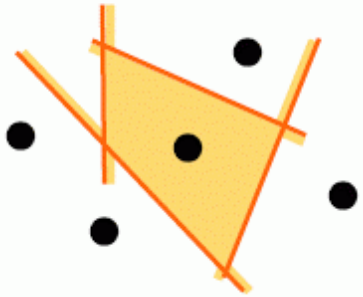
Voronoi-Zellen:

Sei $h(\mathbf{q}_i,\mathbf{q}_j)$ der Halbraum definiert durch die Mittelsenkrechte zwischen $\mathbf{q}_i$ und $\mathbf{q}_j$, indem $\mathbf{q}_i$ liegt, dann gilt:

$$V(\mathbf{q}_i) = \cap_{j\neq i} h(\mathbf{q}_i,\mathbf{q}_j)$$

➔ die Voronoi-Zellen sind konvex





Erzeugung des Voronoi-Diagramms –

Algorithmen:

Sweep-Line-Algorithmus

- sortiere Ereignisse in eine Hauptrichtung
- bearbeite Ereignis und erzeuge neue Kanten

Einfügealgorithmus

- ordne Punkte in zufälliger Reihenfolge an
- füge Punkt für Punkt in Voronoi-Diagramm ein

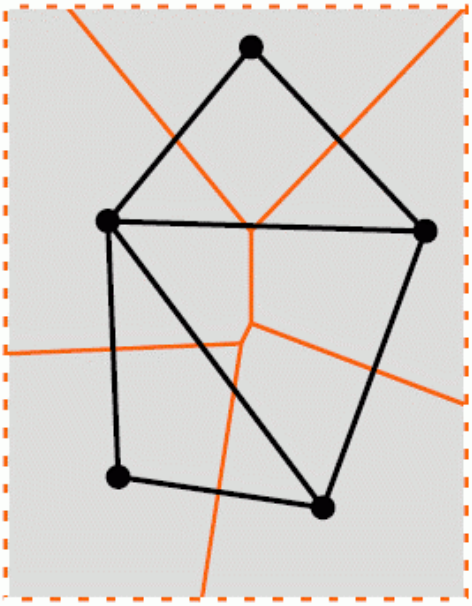
Voronoi in höheren Dimensionen:

- Größe: $\Theta(k^{\lceil n/2 \rceil})$
- Laufzeit: $O(k\cdot \log k + k^{\lceil n/2 \rceil})$

(siehe "Algorithmische Geometrie")

Definition Delaunay Graph:

Der Duale Graph des Voronoi-Diagramms Vor(Q) wird Delaunay-Graph Del(Q) genannt. Die Punkte Q sind die Knoten. Zwei Knoten $\mathbf{q}_i$ und $\mathbf{q}_j$ sind genau dann verbunden, wenn die Voronoi-Zellen $V(\mathbf{q}_i)$ und $V(\mathbf{q}_j)$ eine gemeinsame Kante haben.



Eigenschaften des Delaunay-Graphen:

- die Eckpunkte jeder Delaunay-Zelle liegen auf einem Kreis
- insbes. sind die Zellen konvex
- Wenn keine 4 Eckpunkte auf einem Kreis liegen ("allgemeine Position"), ist der Delaunay-Graph eine Triangulierung.

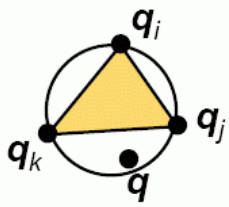
Jede Triangulierung des Delaunay-Graphen wird als *Delaunay-Triangulierung* bezeichnet.

Algorithmen zur Konstruktion einer Delaunay-Triangulierung zu einer Punktmenge:
- Flipping-Algorithmus
- Sweep-Line-Verfahren
- Einfügealgorithmus (inkrementell)
- divide and conquer
- und andere

(siehe "Algorithmische Geometrie")

Umkreis-Test:

Im Umkreis eines Delaunay-Dreiecks liegt kein 4. Punkt



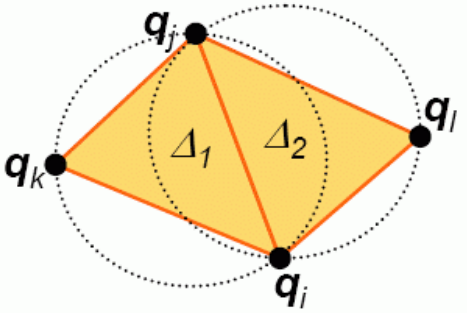
Geg.: Dreieck ($\mathbf{q}_i$,$\mathbf{q}_j$,$\mathbf{q}_k$) und $\mathbf{q}$

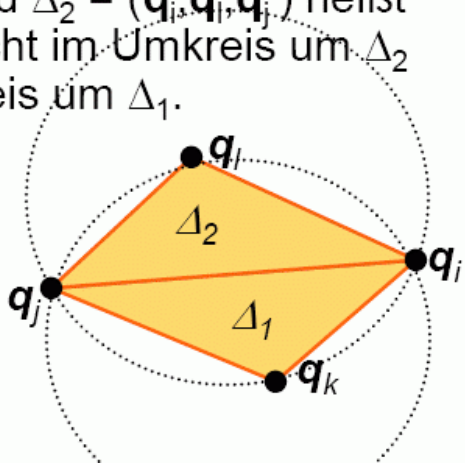
Frage: Liegt $\mathbf{q}$ im Umkreis?

$$\Leftrightarrow \det \begin{pmatrix} q_{i,x} & q_{i,y} & \|\boldsymbol{q}_i\|^2 & 1 \\ q_{j,x} & q_{j,y} & \|\boldsymbol{q}_j\|^2 & 1 \\ q_{k,x} & q_{k,y} & \|\boldsymbol{q}_k\|^2 & 1 \\ q_x & q_y & \|\boldsymbol{q}\|^2 & 1 \end{pmatrix} > 0$$

Flipping Algorithmus:

Definition: Eine Kante ($\mathbf{q}_i$,$\mathbf{q}_j$) mit benachbarten Dreiecken Δ_1 = ($\mathbf{q}_i$,$\mathbf{q}_j$,$\mathbf{q}_k$) und Δ_2 = ($\mathbf{q}_i$,$\mathbf{q}_l$,$\mathbf{q}_j$) heißt lokal Delaunay, wenn $\mathbf{q}_k$ nicht im Umkreis um Δ_2 liegt, bzw. $\mathbf{q}_l$ nicht im Umkreis um Δ_1.





Flipping Algorithmus:

Theorem: Sind alle Kanten einer die konvexe Hülle abdeckende Triangulierung lokal Delaunay, so ist die Triangulierung Delaunay.

Algorithmus:

- starte mit beliebiger Triangulierung
- staple alle nicht lokalen Delaunay-Kanten auf Stapel
- solange Stapel nicht leer
  - flippe oberste Kante von Stapel
  - untersuche vier benachbarte Kanten und eliminiere sie vom bzw. staple sie auf den Stapel

(Bartz 2005)

3. Ansatz:
Natürliche-Nachbar-Interpolation

Idee:

- Der Punkt *q*, an dem interpoliert werden soll, wird als neuer Punkt ins Voronoi-Diagramm der Beobachtungspunkte eingefügt
- es werden zwei Voronoi-Diagramme betrachtet: mit und ohne Punkt *q*
- die natürlichen Nachbarn von q im neuen Voronoi-Diagramm sind die Stellen, von wo die Werte für die Interpolation benutzt werden

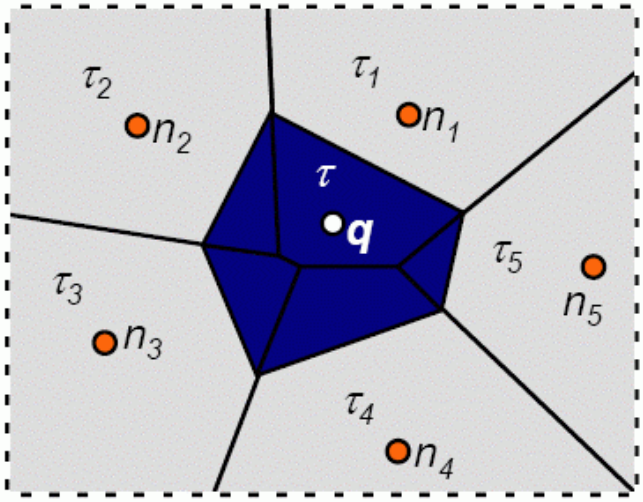
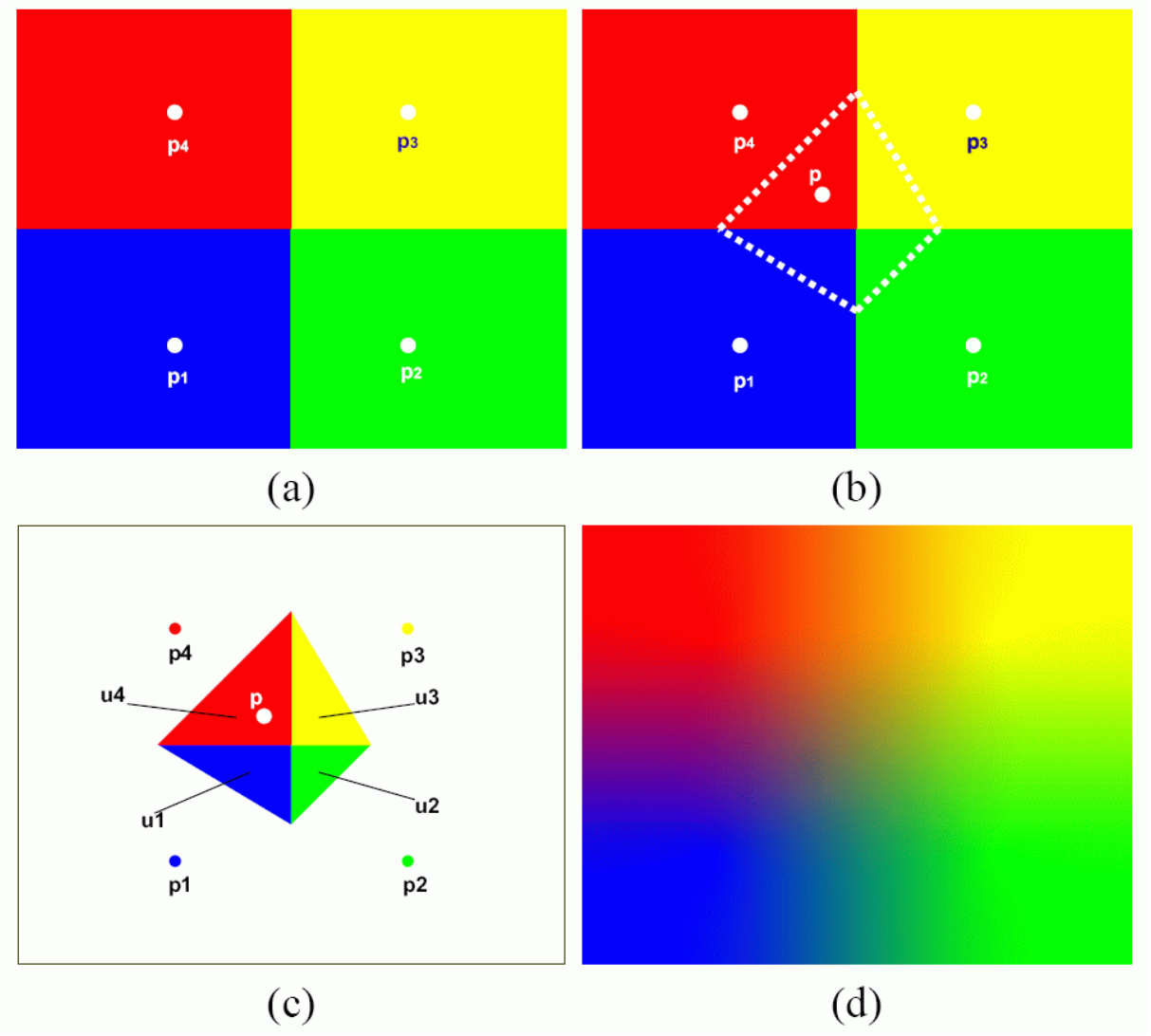


Illustration zur **Natürlichen Nachbarinterpolation**

(Bartz 2005)

als Gewichte werden die Flächenanteile verwendet, die die Voronoi-Zellen des alten Diagramms aus der Zelle von *q* im neuen Diagramm ausschneiden



(a) Punkte im alten Voronoi-Diagramm ohne *p* (der Punkt, wo interpoliert werden soll, heißt hier *p*)

(b) eingezeichnete Zelle von *p* im neuen Diagramm

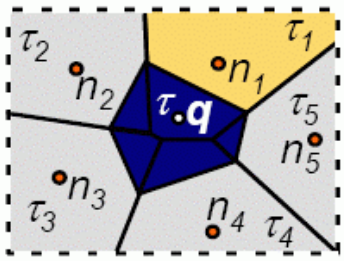
(c) Flächen u1, ..., u4, die die Gewichte liefern, mit denen die Beobachtungswerte an den Stellen p1, ..., p4 in die Berechnung für *p* eingehen

(d) interpoliertes Ergebnisbild (hier Farbe als zu interpolierende Größe)

(aus Park et al., o.J.)

Teilprobleme:

- Punktinklusionsproblem:
  In welcher Zelle liegt **q**?
- Berechne Zelle um **q**
- **Berechne Schnitt von neuer Zelle mit alten Zellen**



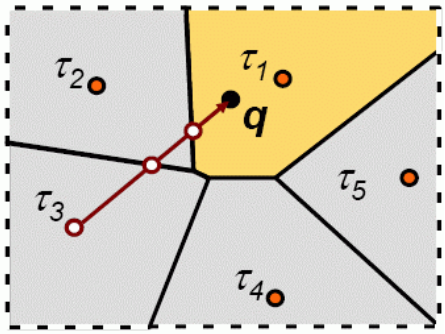
Punktinklusionsproblem:

geg: Tesselierung des Raumes $\mathbf{R}^n$ in eben begrenzte konvexe Zellen τ_i mit Begrenzungsebenen $\{\varphi_{i,j}\}$ und einen Punkt $\mathbf{q} \in \mathbf{R}^n$

ges: Zelle τ_i, die **q** enthält

Lösung:

- wähle zufälliges τ_i
- solange **q** nicht in τ_i
  - wähle beliebigen Punkt **r** aus τ_i
  - finde Begrenzung $\varphi_{i,j}$ von τ_i, die (**qr**) schneidet
  - betrachte τ_j als τ_i



(aus Bartz 2005)

weitere Algorithmen zur Punktlokalisation in der Vorlesung "Algorithmische Geometrie"