Prof. Dr. Tino Weinkauf

Introduction to Visualization and Computer Graphics

Grids and Interpolation

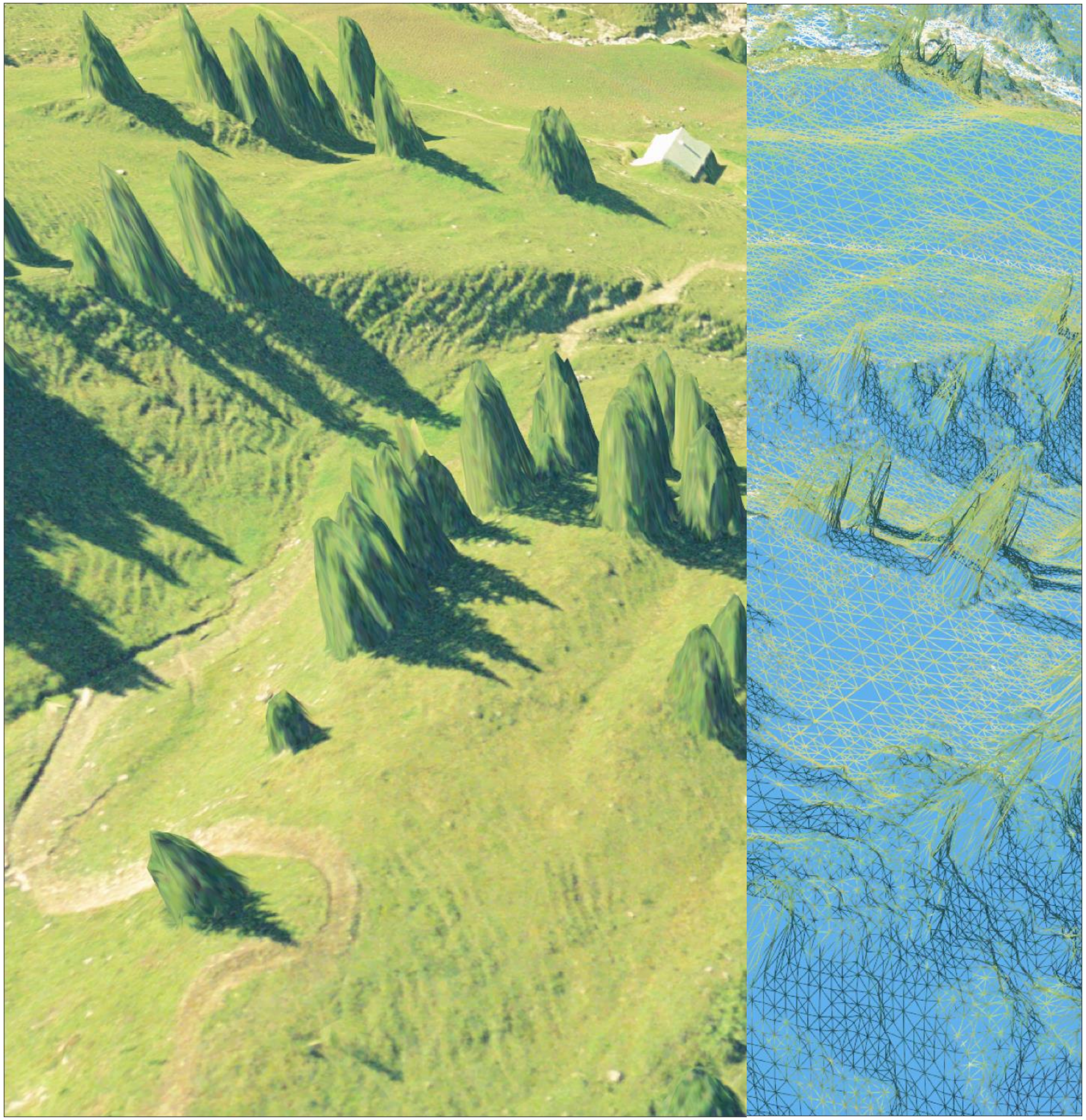
- No lecture next Tuesday!

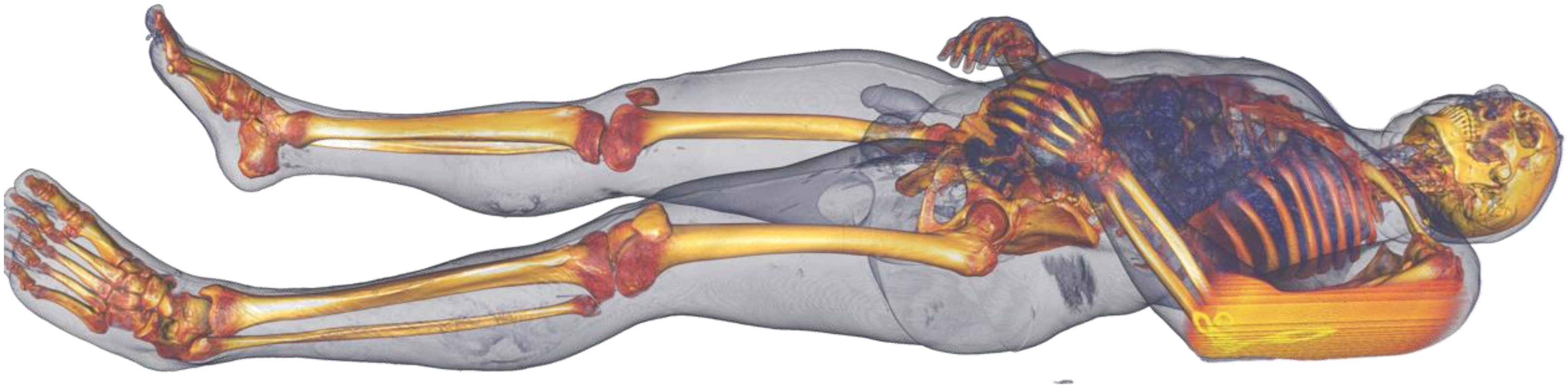
Introduction to Visualization and Computer Graphics
DH2320, Fall 2015

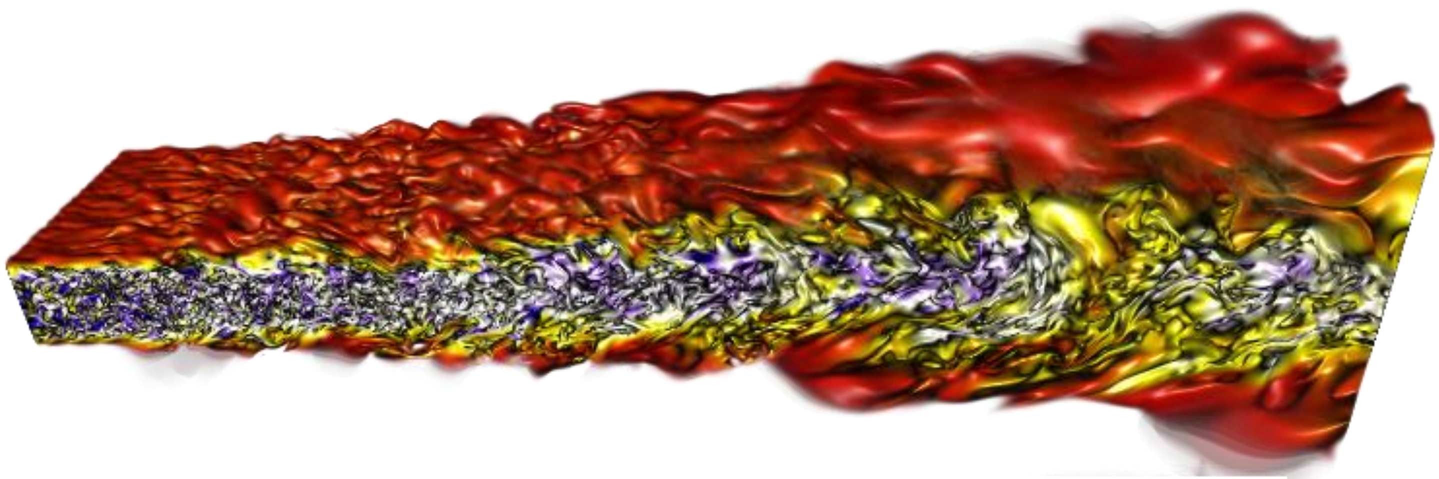
Prof. Dr. Tino Weinkauf

Grids and Interpolation

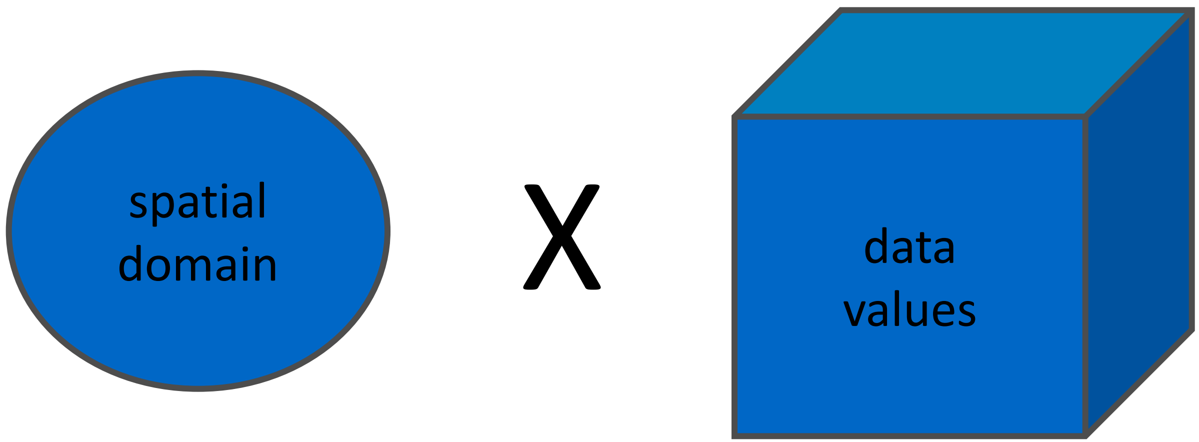
Structured Grids
Unstructured Grids







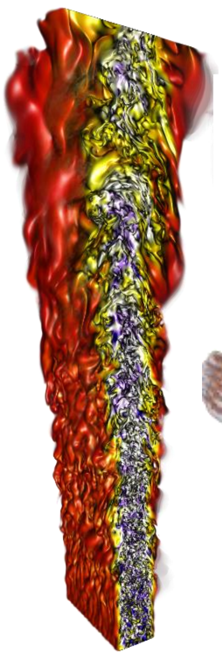
R^n R^m

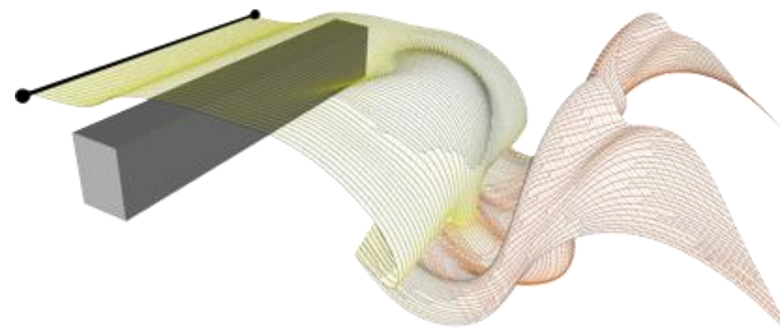


independent variables *dependent variables*

- In most cases, the visualization data represent a **continuous real object**, e.g., an oscillating membrane, a velocity field around a body, an organ, human tissue, etc.
  - This object lives in an n-dimensional space - the **domain**
- Usually, the data is only given at a finite set of locations, or **samples**, in space and/or time
  - Remember imaging processes like numerical simulation and CT-scanning, note similarity to pixel images
- We call this a **discrete structure**, or a **discrete representation** of a continuous object

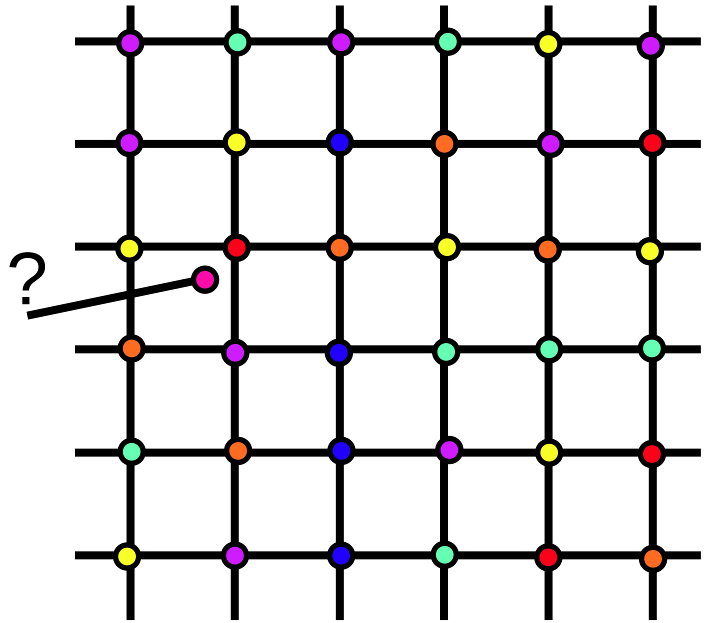
- Discrete representations
  - We usually deal with the reconstruction of a continuous real object from a given discrete representation



- Discrete structures consist of point samples
- Often, we build **grids/meshes** that connect neighboring samples

- Discrete representations
  - We usually deal with the reconstruction of a continuous real object from a given discrete representation



- Discrete structures consist of point samples
- Often, we build **grids/meshes** that connect neighboring samples

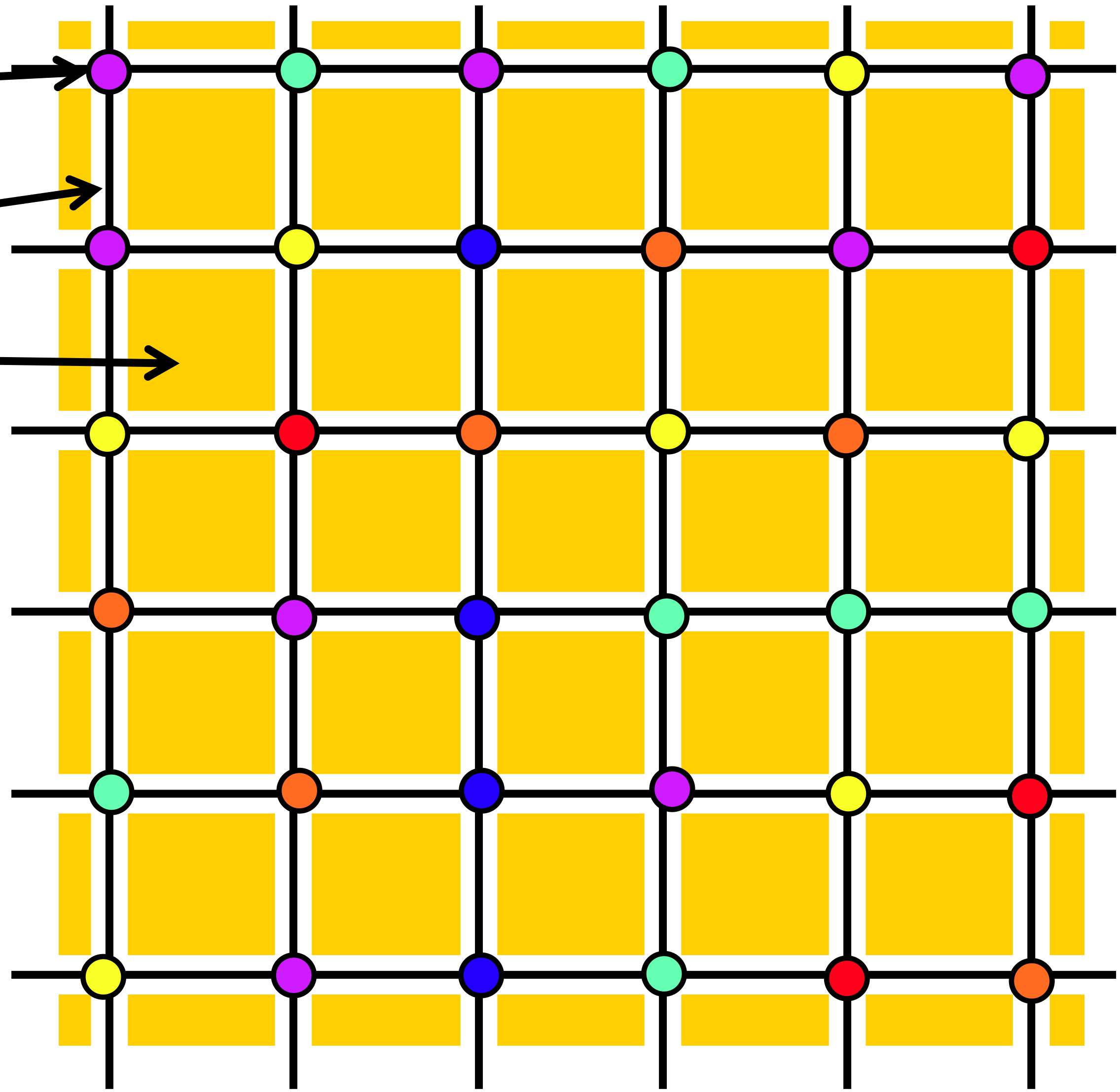
- Grid terminology

0D: grid vertex
(grid point)

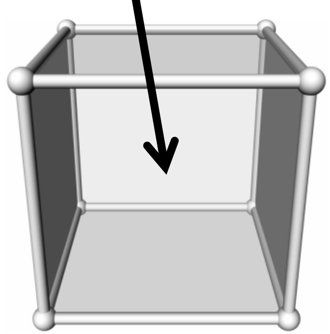
1D: grid line

2D: grid face

3D: grid voxel



grid vertices
grid lines
grid faces



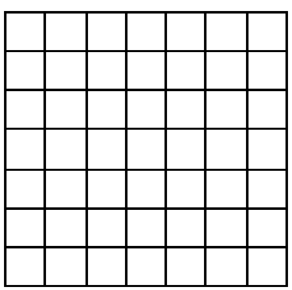
grid cell: largest-dimensional element in a grid
2D: grid face
3D: grid voxel

Data Connectivity

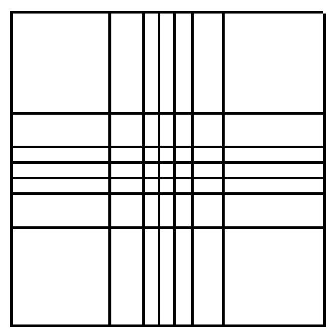
- There are different types of grids:
- **Structured grids**
  connectivity is implicitly given.
  - **Block-structured grids**
    combination of several structured grids
- **Unstructured grids**
  connectivity is explicitly given.
- **Hybrid grids**
  combination of different grid types

Structured grids

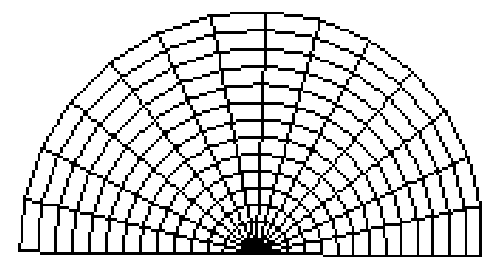
- “Structured” refers to the implicitly given connectivity between the grid vertices
- We distinguish different types of structured grids regarding the implicitly or explicitly given coordinate positions of the grid vertices



uniform grid
implicitly given coordinates



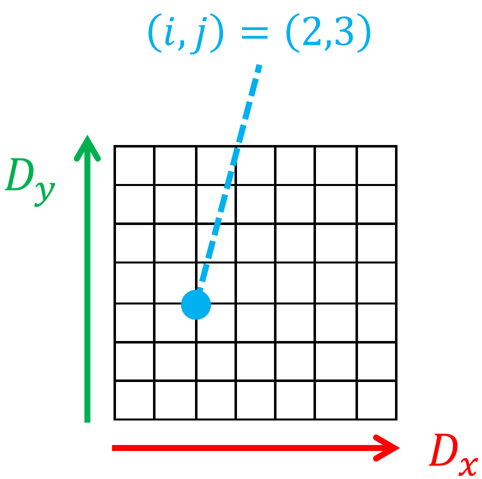
rectilinear grid
semi-implicitly given coordinates



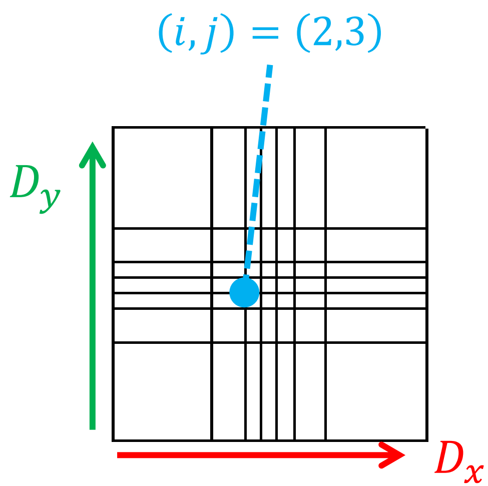
curvilinear grid
explicitly given coordinates

Structured grids

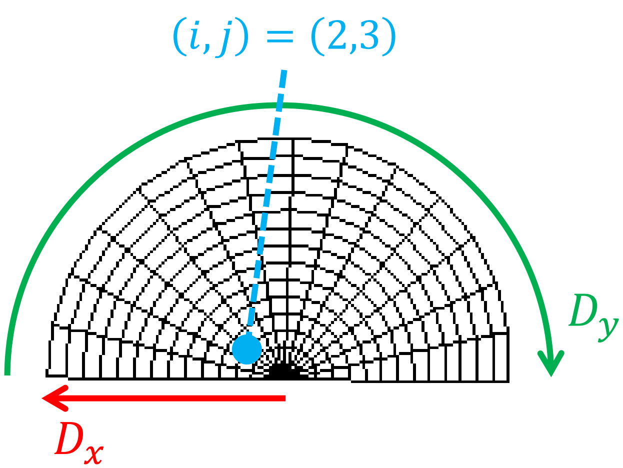
- Number of grid vertices: D_x, D_y, D_z
- We can address every grid vertex with an index tuple (i, j, k)
  - $0 \leq i < D_x$ $\quad$ $0 \leq j < D_y$ $\quad$ $0 \leq k < D_z$



uniform grid
implicitly given coordinates



rectilinear grid
semi-implicitly given coordinates



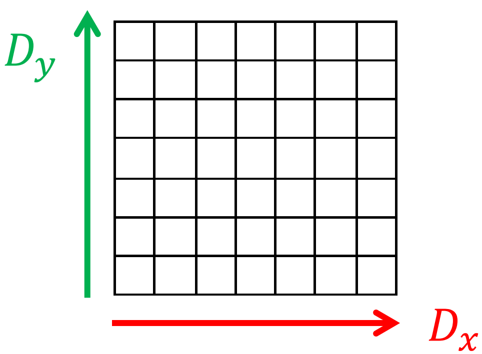
curvilinear grid
explicitly given coordinates

Structured grids

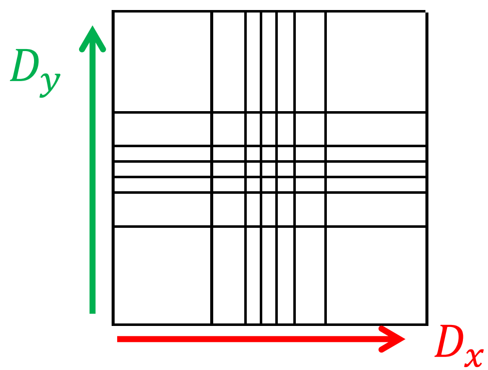
- Number of grid vertices: D_x, D_y, D_z

- We can address every **grid cell** with an index tuple (i, j, k)
  - $0 \leq i < D_x - 1$ $\quad$ $0 \leq j < D_y - 1$ $\quad$ $0 \leq k < D_z - 1$

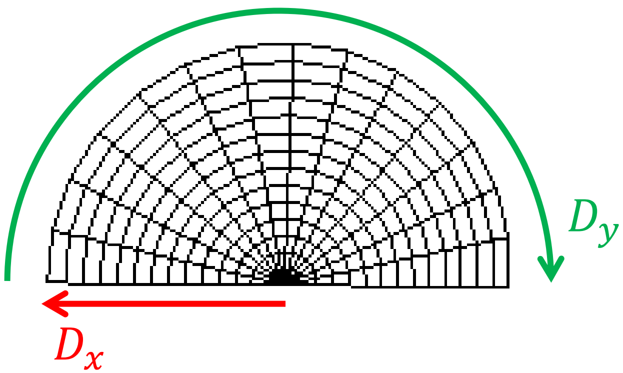
- ➔ Number of cells: $(D_x - 1) \times (D_y - 1) \times (D_z - 1)$



uniform grid
implicitly given coordinates



rectilinear grid
semi-implicitly given coordinates



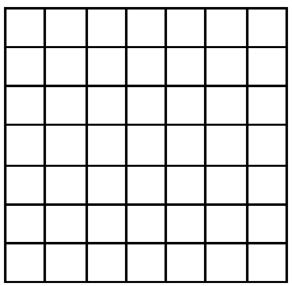
curvilinear grid
explicitly given coordinates

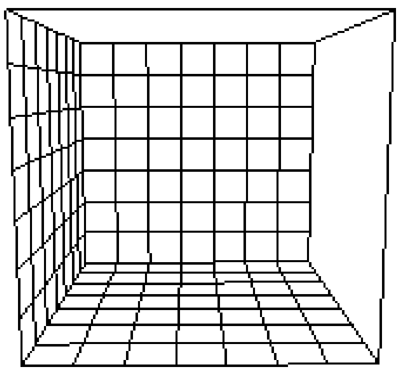
very important

- **Regular or uniform grids**

- Cells are rectangles or rectangular cuboids of the same size

- All grid lines are parallel to the axes





- To define a uniform grid, we need the following:
  - Bounding box: $(x_{min}, y_{min}, z_{min})$ - $(x_{max}, y_{max}, z_{max})$
  - Number of grid vertices in each dimension: D_x, D_y, D_z
  - ➔ Cell size: d_x, d_y, d_z

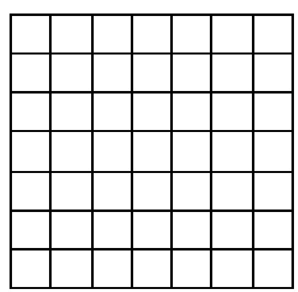
- **Regular or uniform grids**

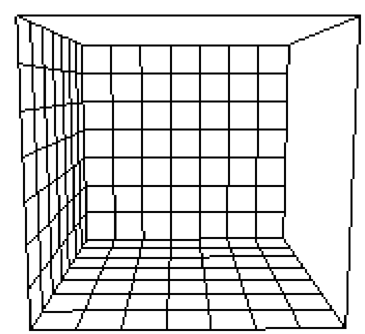
- Well suited for image data (medical applications)

- Coordinate ➔ cell is very simple and cheap
  - Global search is good enough; local search not required

- Coordinate of a grid vertex:

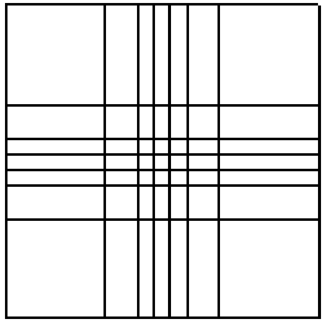
$$(i \cdot d_x, j \cdot d_y, k \cdot d_z)$$



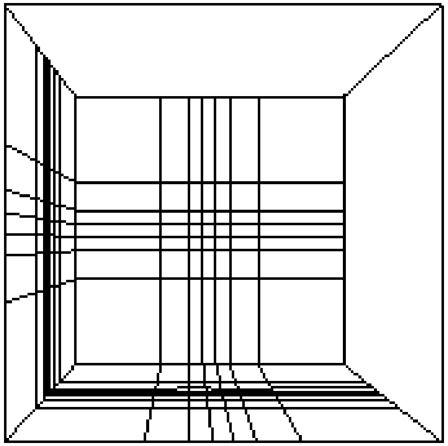


- **Cartesian grid**
- Special case of a uniform grid: $d_x = d_y = d_z$
- Consists of squares (2D), cubes (3D)

- **Rectilinear grids**



- Cells are rectangles of *different* sizes
- All grid lines are parallel to the axes
- Vertex locations are inferred from positions of grid lines for each dimension:
  - XLoc = {0.0, 1.5, 2.0, 5.0, …}
  - YLoc = {-1.0, 0.3, 1.0, 2.0, …}
  - ZLoc = {3.0, 3.5, 3.6, 4.1, …}



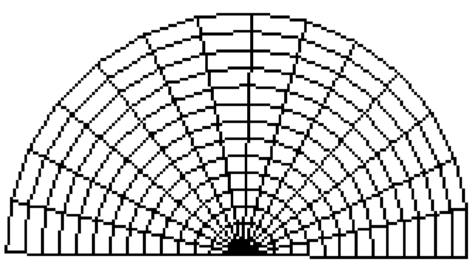
- Coordinate ➔ cell still quite simple

- **Curvilinear grids**

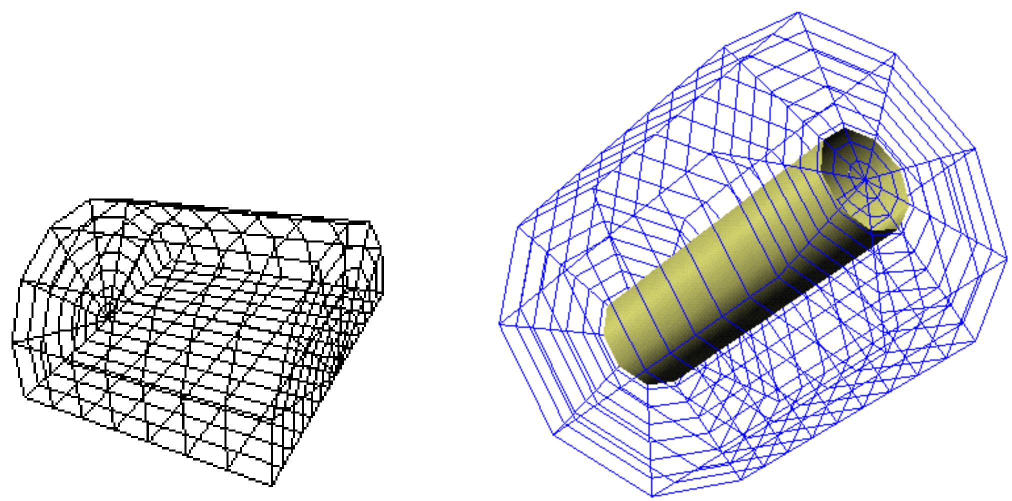
- Vertex locations are explicitly given
  - XYZLoc = {(0.0, -1.0, 3.0), (1.5, 0.3, 3.5), (2.0, 1.0, 3.6), …}

- Cells are quadrilaterals or cuboids

- Grid lines are not (necessarily) parallel to the axes



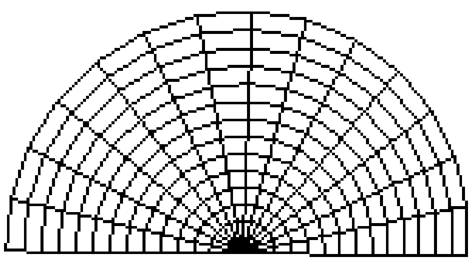
2D curvilinear grid



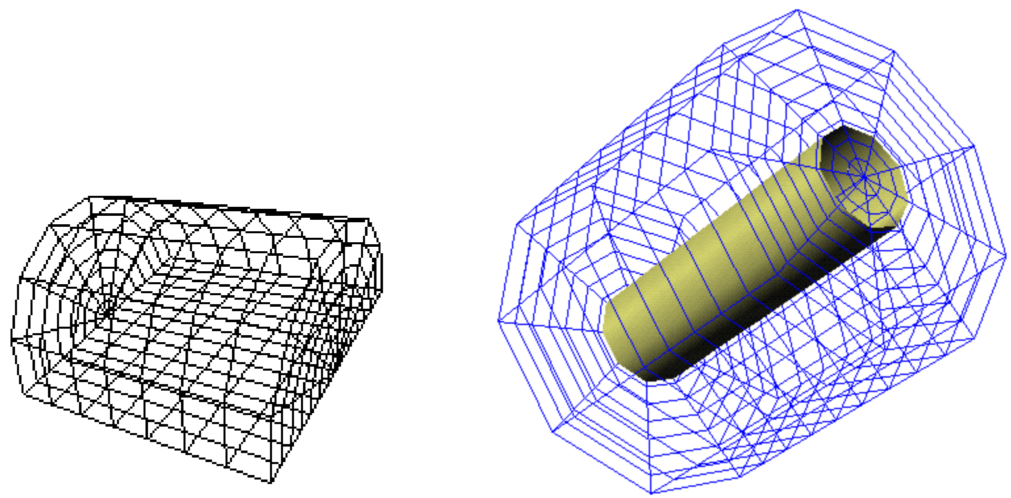
3D curvilinear grids

- **Curvilinear grids**

- Coordinate ➔ cell:
  - **Local search** within last cell or its immediate neighbors
  - **Global search** via quadtree/octree



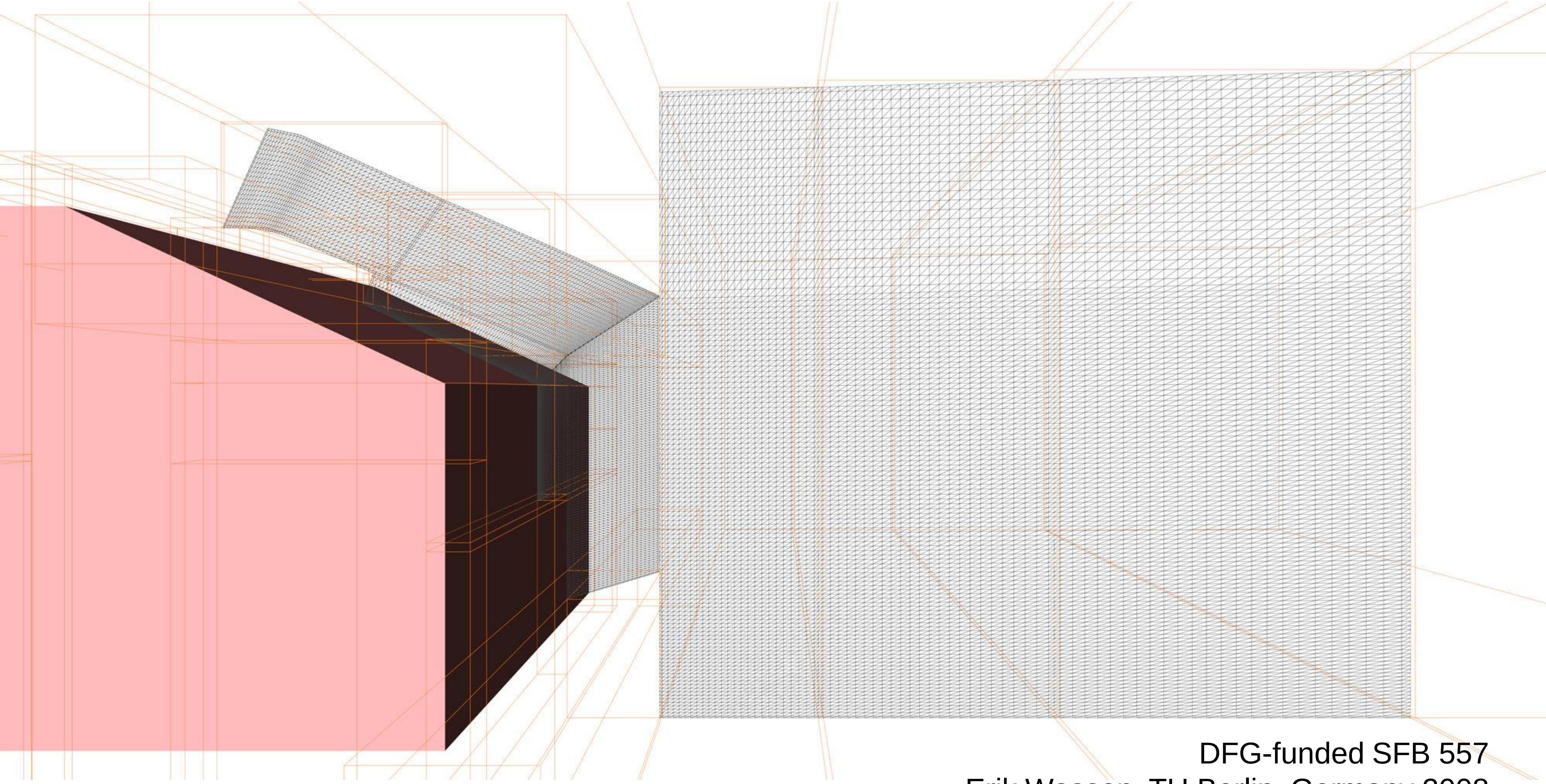
2D curvilinear grid



3D curvilinear grids

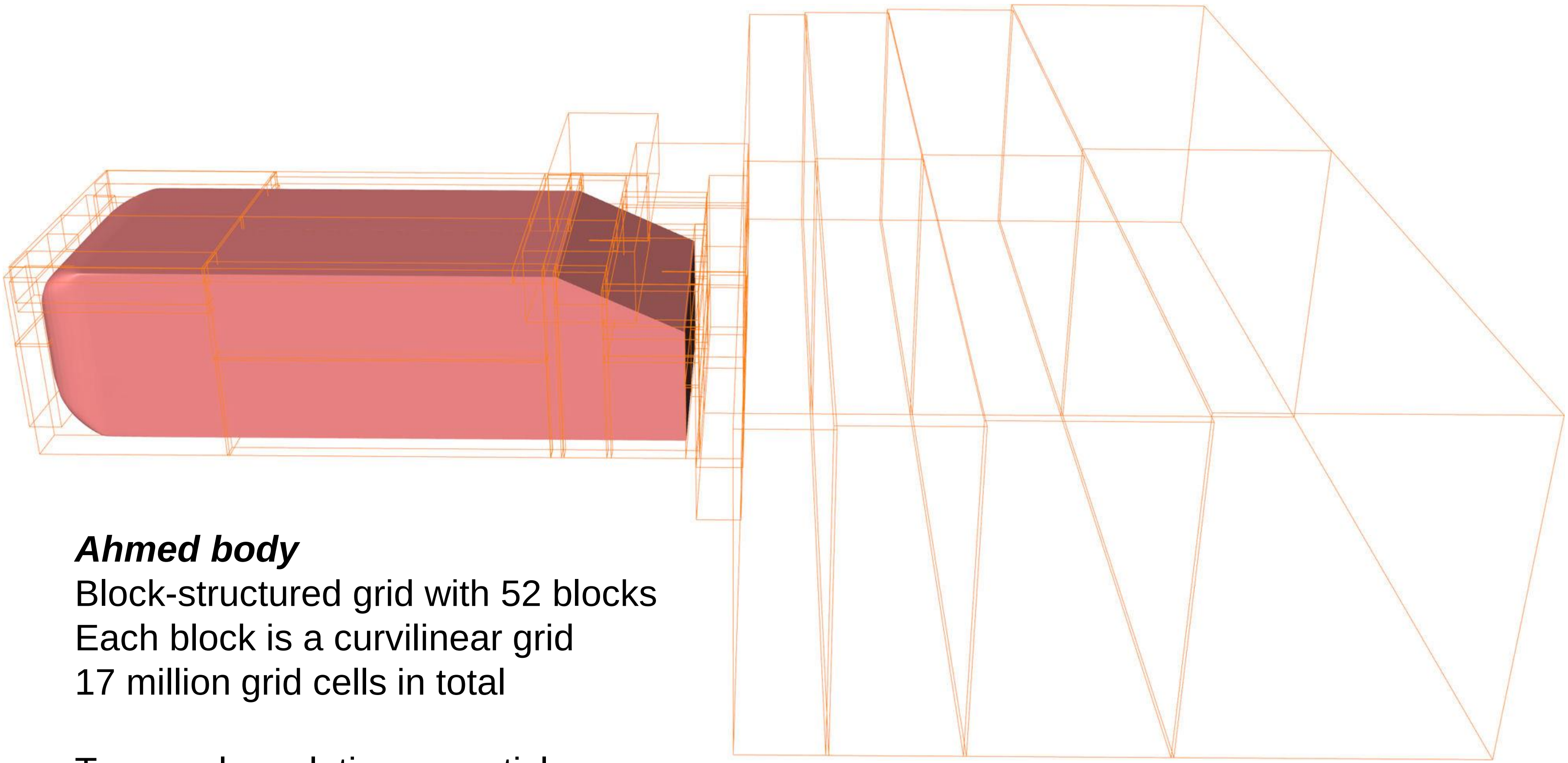
- **Block-structured grids**

- combination of several structured grids



DFG-funded SFB 557
Erik Wassen, TU Berlin, Germany 2008

- Demands on data storage, an example:

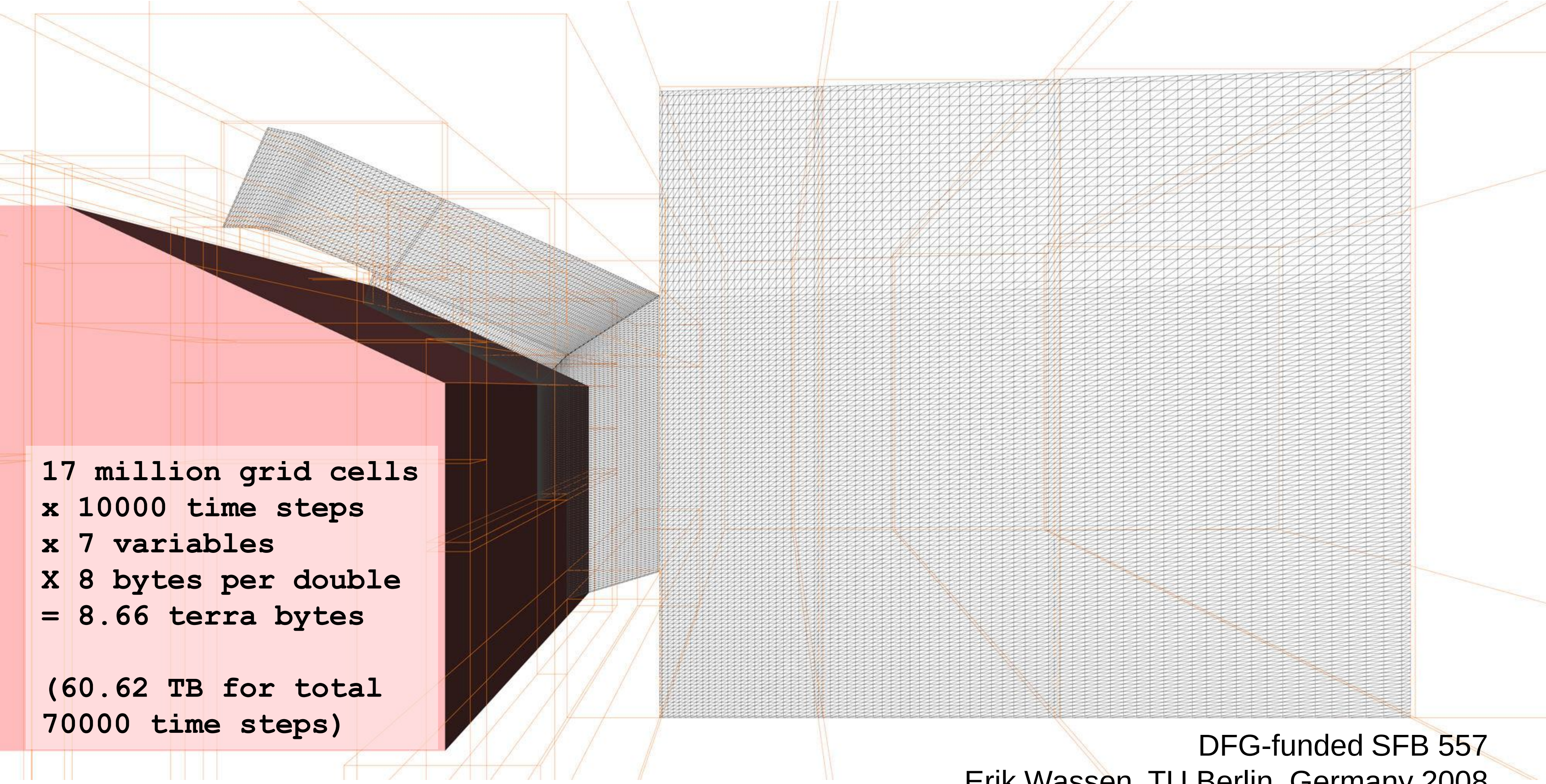


Ahmed body
Block-structured grid with 52 blocks
Each block is a curvilinear grid
17 million grid cells in total

Temporal resolution: a particle needs 10000 time steps from front to back of the Ahmed body

DFG-funded SFB 557
Erik Wassen, TU Berlin, Germany 2008

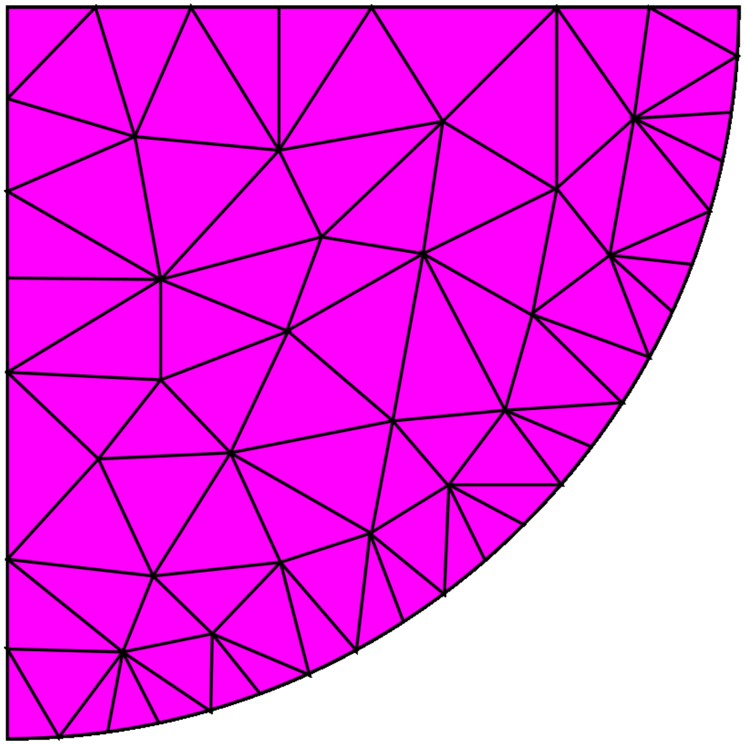
- Demands on data storage, an example:



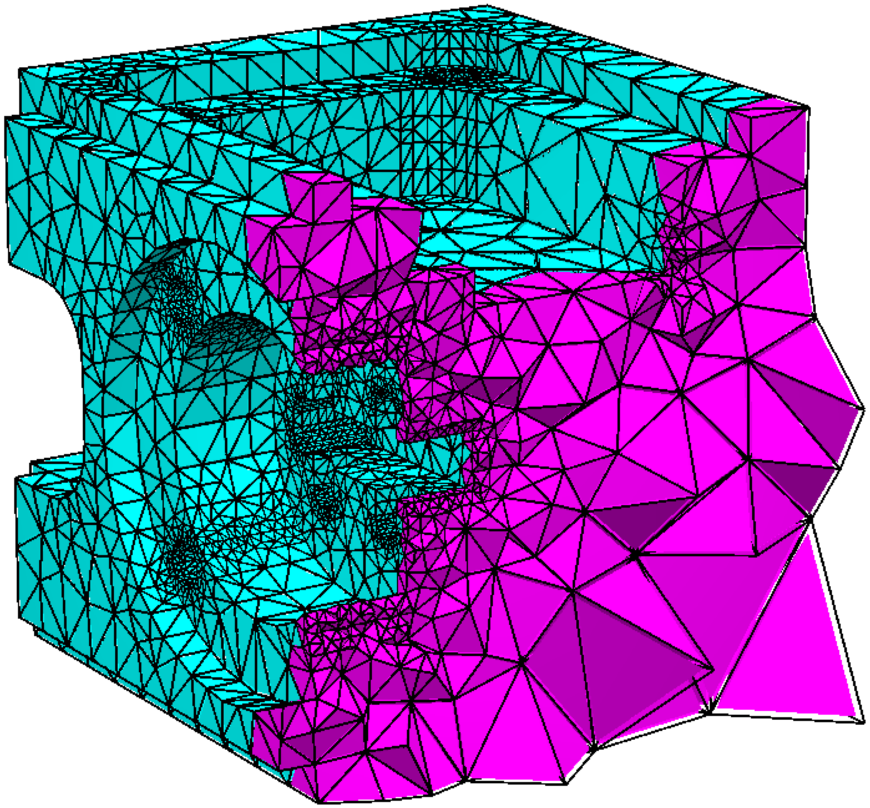
DFG-funded SFB 557
Erik Wassen, TU Berlin, Germany 2008

➔ Do not save every time step, not every variable, and not every block.

- **Unstructured grids**



2D unstructured grid
consisting of triangles



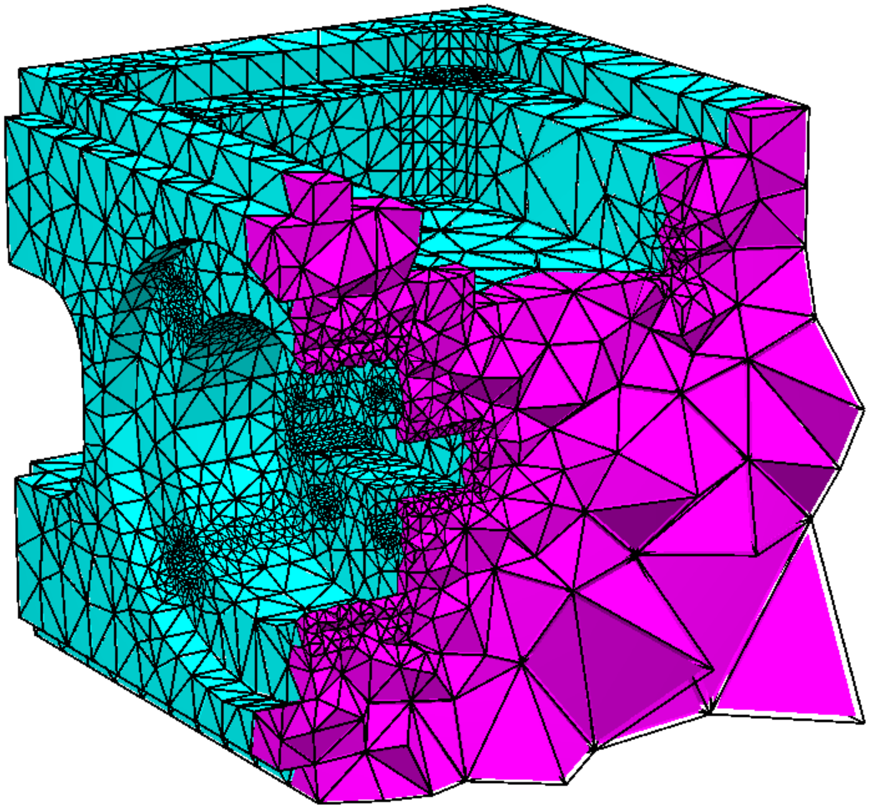
3D unstructured grid
consisting of tetrahedra
(from TetGen user manual)

- **Unstructured grids**

- Vertex locations and connectivity explicitly given

- Linear interpolation within a triangle/tetrahedron using barycentric coordinates

- Coordinate ➔ triangle/tetra:
  - **Local search** within last triangle/tetra or its immediate neighbors
  - **Global search** via quadtree/octree



3D unstructured grid consisting of tetrahedra (from TetGen user manual)

How to store unstructured grids? Different requirements:

- Efficient storage
  - bytes per face / bytes per vertex
- Efficient access
  - of face / vertex properties (e.g., position)
- Efficient traversal
  - e.g., neighboring face, 1-ring of a vertex,...
- Requirements are competing

Face set

- Store faces
  - 3 positions
  - no connectivity ("match positions")
- Example: STL
  - very simple structure (too simple, unpractical!)
  - easily portable

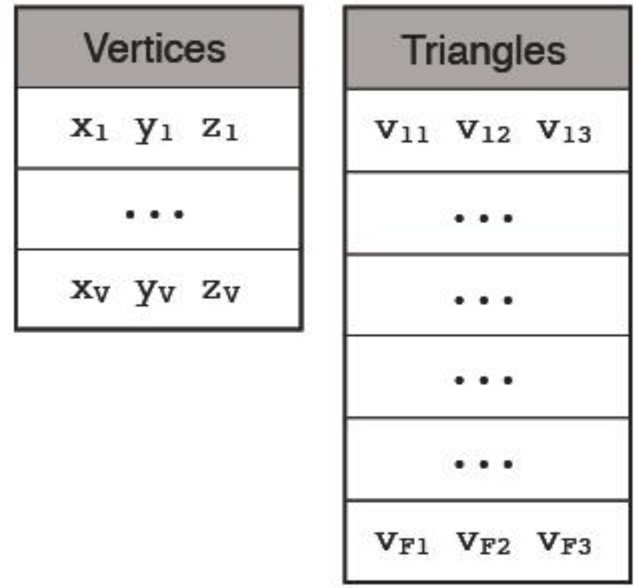
Triangles		
x_{11} y_{11} z_{11}	x_{12} y_{12} z_{12}	x_{13} y_{13} z_{13}
x_{21} y_{21} z_{21}	x_{22} y_{22} z_{22}	x_{23} y_{23} z_{23}
...	...	...
x_{F1} y_{F1} z_{F1}	x_{F2} y_{F2} z_{F2}	x_{F3} y_{F3} z_{F3}

36 B/f = 72 B/v

no connectivity!

Shared vertex

- vertex table stores positions
- triangle table stores indices into vertices
- No explicit connectivity
- Examples: OFF, OBJ, PLY
  - Quite simple and efficient
  - Enables efficient operations on *static* meshes

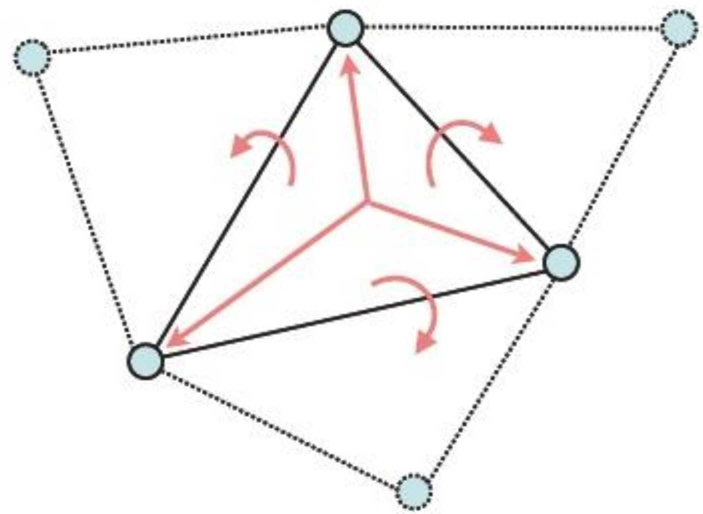


12 B/v + 12 B/f = 36 B/v

no neighborhood info

Face-based connectivity

- vertices store
  - position
  - face reference
- faces store
  - 3 vertex references
  - references to 3 neighboring faces

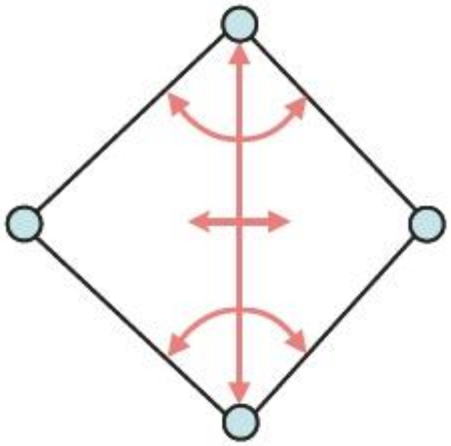


64 B/v

no edges!

Edge-based connectivity

- vertex stores
  - position
  - reference to 1 edge
- edge stores references to
  - 2 vertices
  - 2 faces
  - 4 edges
- face stores
  - reference to 1 edge

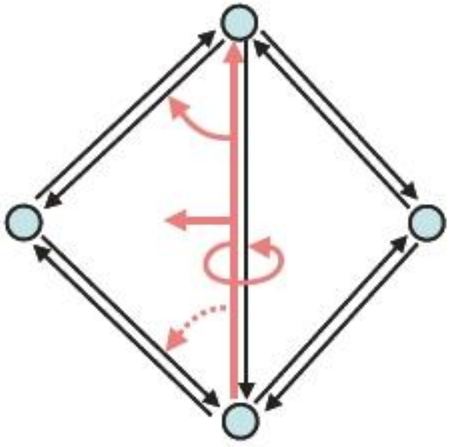


120 B/v

edge orientation?

Half-edge based connectivity

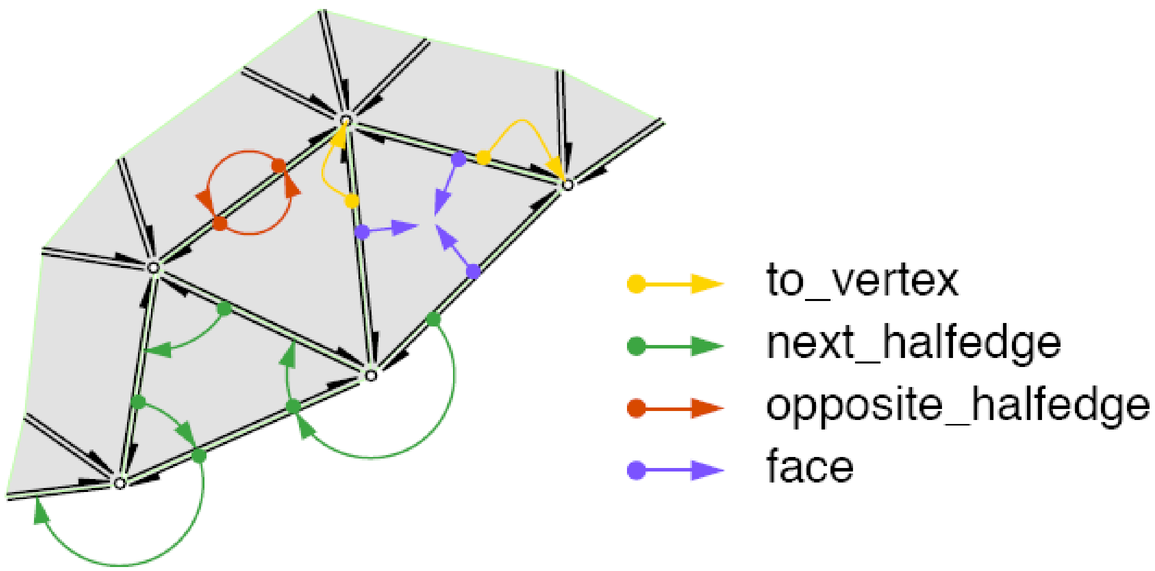
- vertex stores
  - position
  - reference to 1 half-edge
- half-edge stored references to
  - 1 vertex
  - 1 face
  - 1, 2, or 3 half-edges
- face stores
  - reference to 1 half-edge



96 to 144 B/v

no case distinctions during traversal

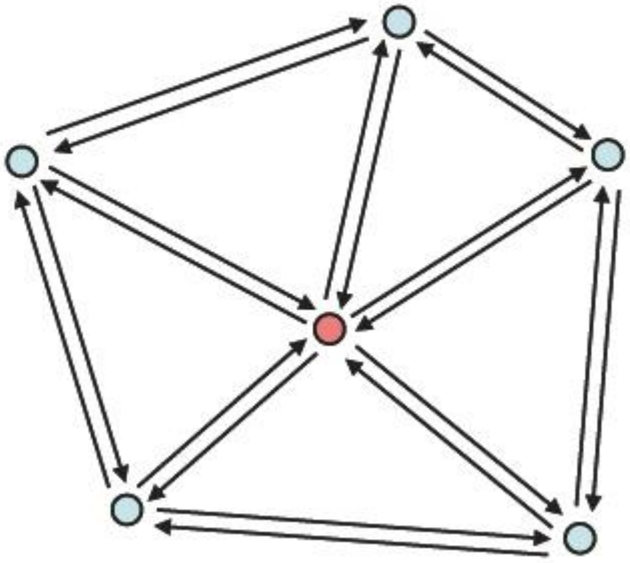
- **Half-edge based connectivity**



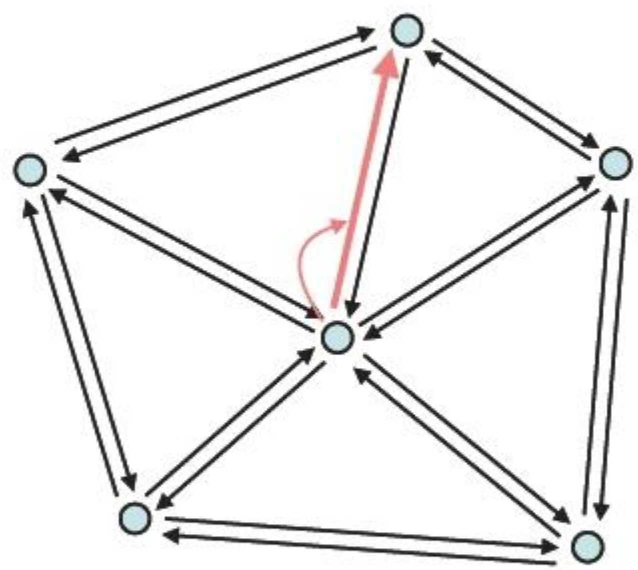
Half-edge based connectivity: Traversal

- Building blocks
  - Vertex to (outgoing) halfedge
  - half-edge to next (previous) halfedge
  - half-edge to neighboring half-edge
  - half-edge to face
  - half-edge to start (end) vertex
- Example: Traverse around vertex (1-ring)
  - enumerate vertices/faces/half-edges

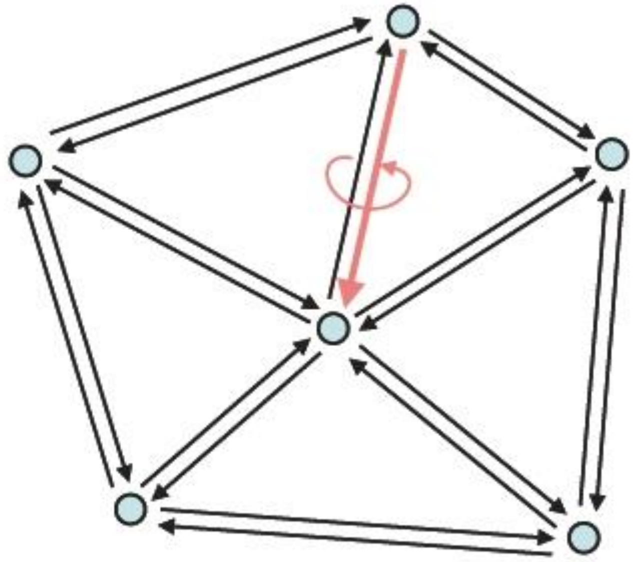
- Start at vertex



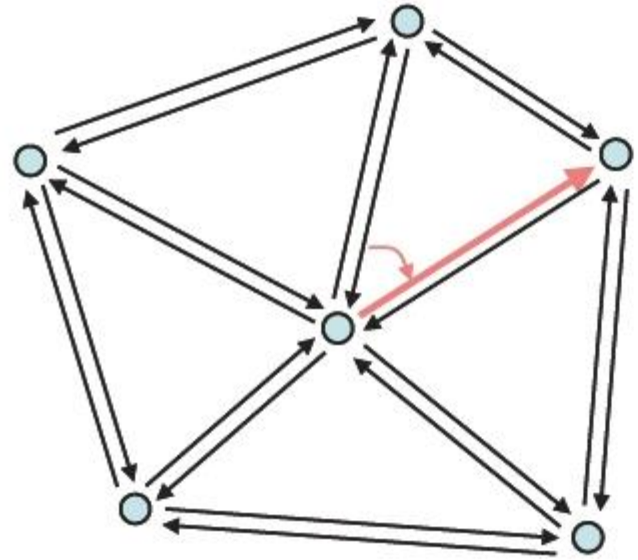
- Start at vertex
- Outgoing halfedge



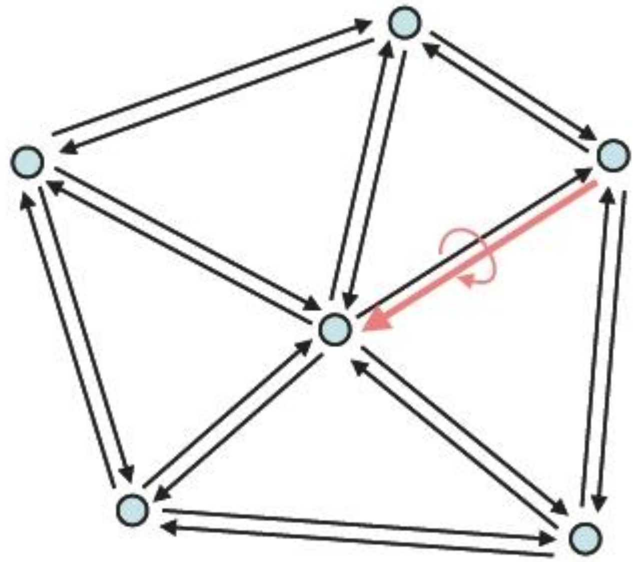
- Start at vertex
- Outgoing halfedge
- Opposite halfedge



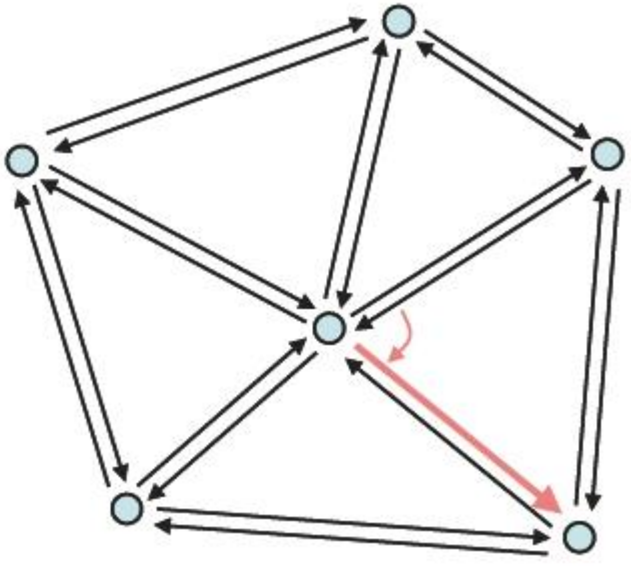
- Start at vertex
- Outgoing halfedge
- Opposite halfedge
- Next half-egde



- Start at vertex
- Outgoing halfedge
- Opposite halfedge
- Next half-egde
- Opposite ...

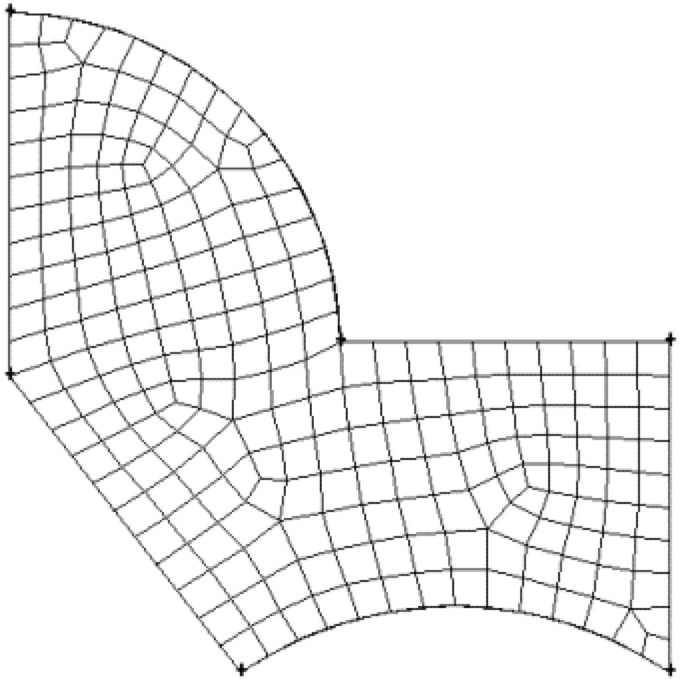


- Start at vertex
- Outgoing halfedge
- Opposite halfedge
- Next half-egde
- Opposite ...
- Next ...
- ...



- CGAL
    - www.cgal.org
    - Computational geometry
    - Free for non-commercial use
- Open Mesh
    - www.openmesh.org
    - Mesh processing
    - Free, LGPL license
- gmu (gmu-lite)
    - proprietary, directed edges

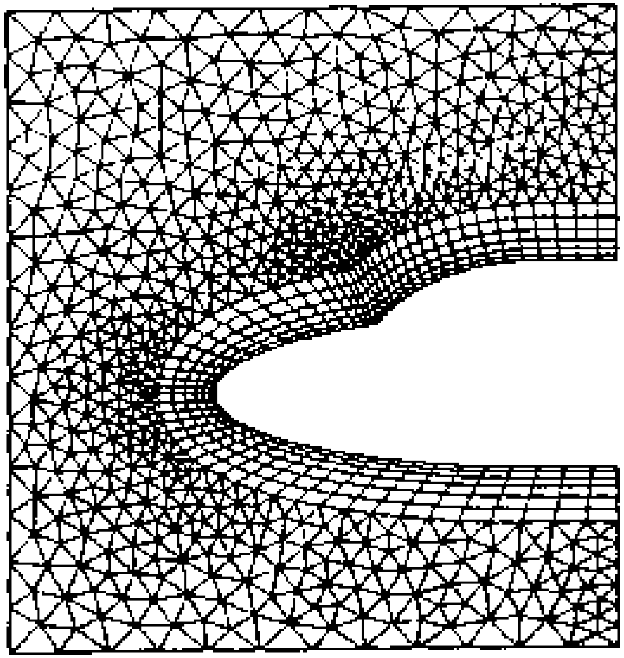
- **Unstructured grids**



2D unstructured grid
consisting of quads

Source: https://www.sharcnet.ca/Software/Gambit/html/modeling_guide/mg0303.htm

- **Hybrid grids**
- combination of different grid types



2D hybrid grid

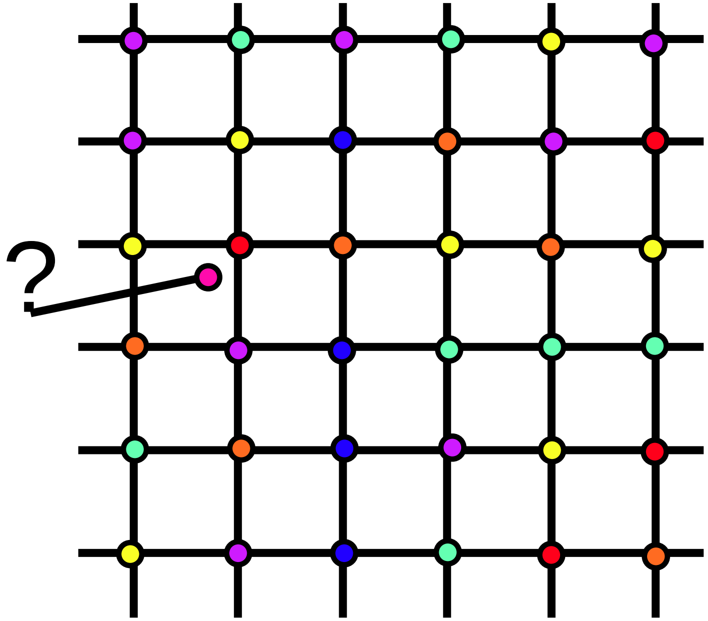
Introduction to Visualization and Computer Graphics
DH2320, Fall 2015

Prof. Dr. Tino Weinkauf

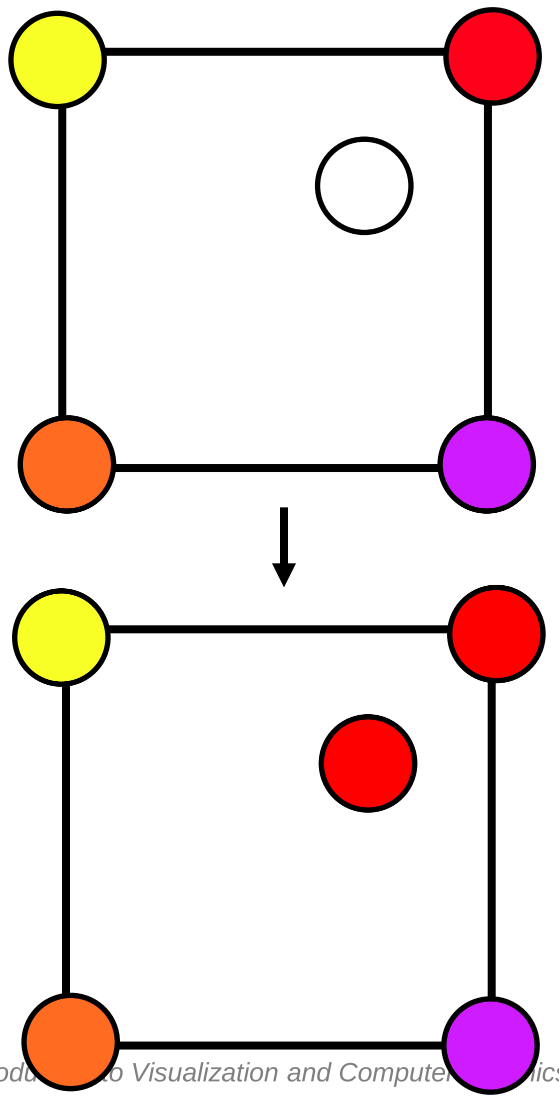
Grids and Interpolation

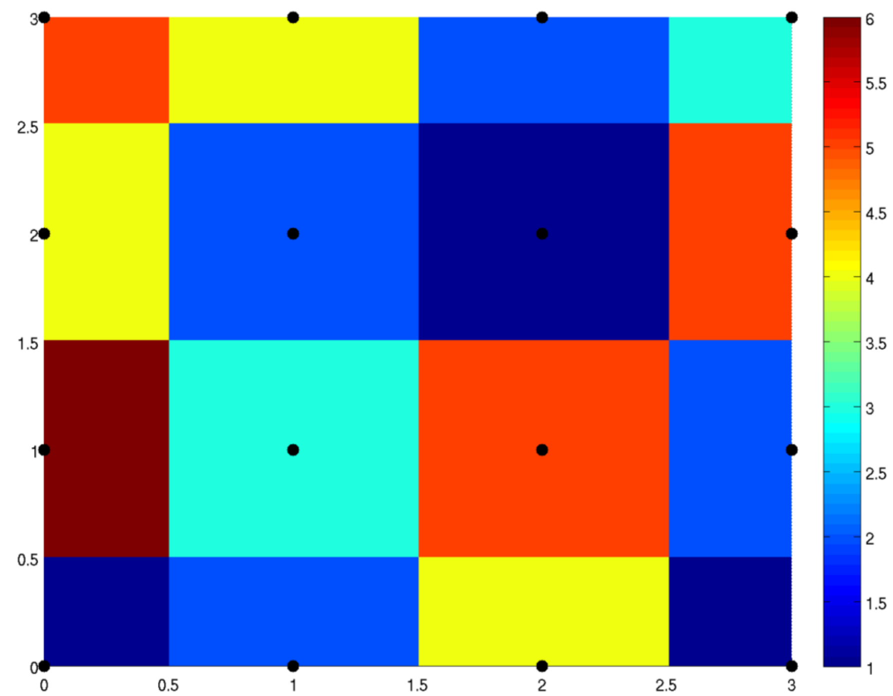
Linear, Bilinear, Trilinear Interpolation in Structured Grids
Gradients
Linear Interpolation in Unstructured Grids

- A grid consists of a finite number of **samples**
  - The continuous signal is known only at a few points (**data points**)
  - In general, data is needed in between these points
- By **interpolation** we obtain a representation that matches the function at the data points
  - **Reconstruction** at any other point possible

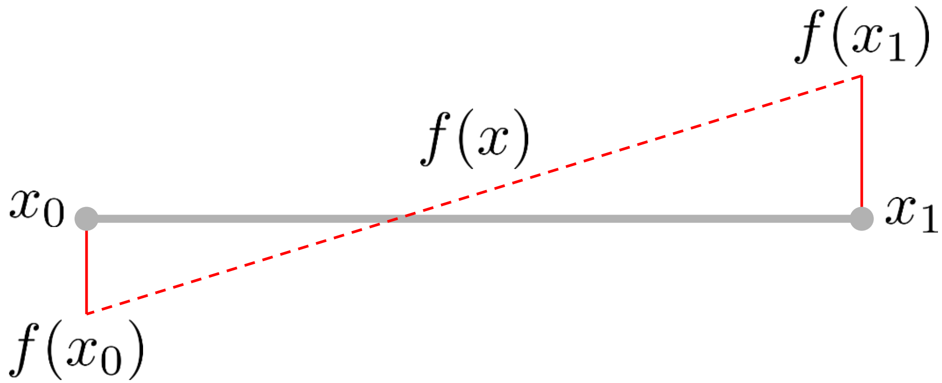


- Simplest approach: **Nearest-Neighbor Interpolation**
  - Assign the value of the nearest grid point to the sample.





- **Linear Interpolation** (in 1D domain)
  - Domain points x, scalar function $f(x)$



General:

$$f(x) = \frac{x_1 - x}{x_1 - x_0} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1) \qquad x \in [x_0, x_1]$$

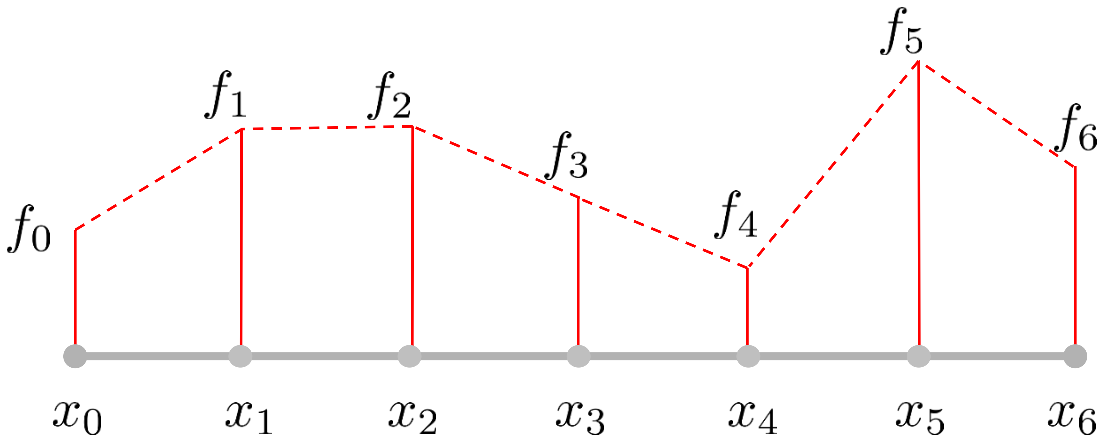
Special Case:

$$f(x) = (1 - x)\ f(0) + x\, f(1) \qquad\qquad x \in [0, 1]$$

$$= \begin{bmatrix} (1 - x) & x \end{bmatrix} \begin{pmatrix} f(0) \\ f(1) \end{pmatrix} = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{pmatrix} f(0) \\ f(1) \end{pmatrix}$$

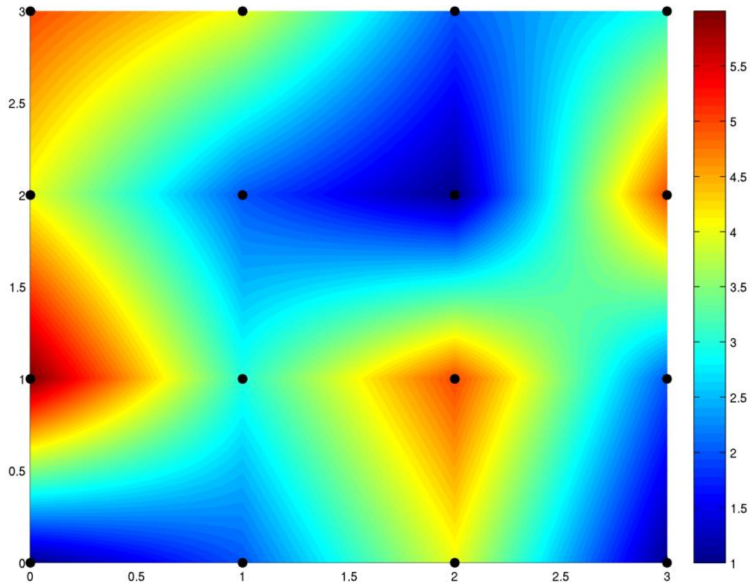
Basis Coefficients

- **Linear Interpolation** (in 1D domain)
  - Sample values $f_i := f(x_i)$

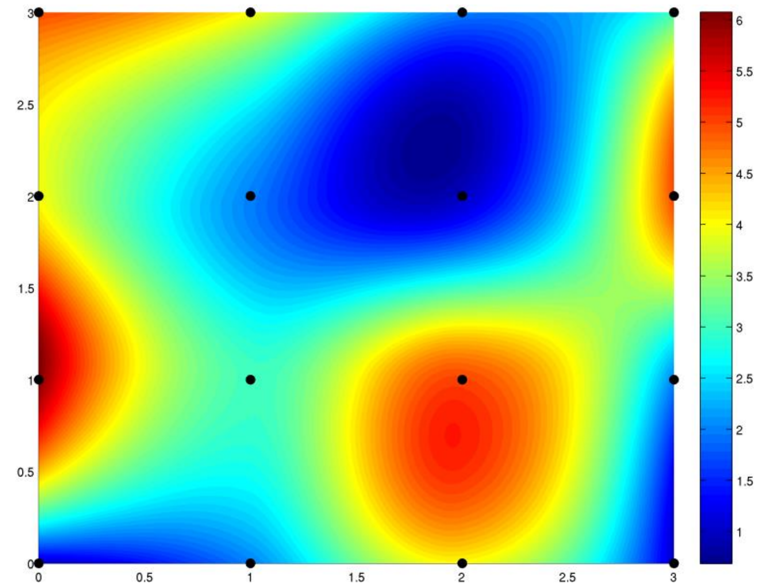


- C^0 Continuity (discontinuous first derivative)
  - Use higher order interpolation for smoother transition, e.g., **cubic** interpolation

- **Interpolation in 2D, 3D, 4D, …**



Bi-Linear



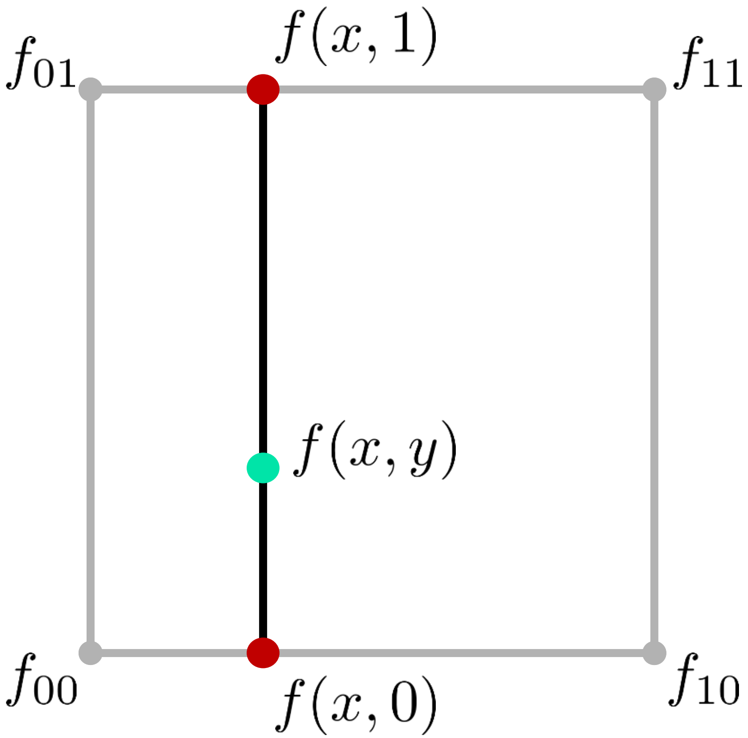
Bi-Cubic

- **Tensor Product Interpolation**
  - Perform linear / cubic … interpolation in each x,y,z … direction **separately**

very important

- Bilinear Interpolation

2D, “bi-linear”

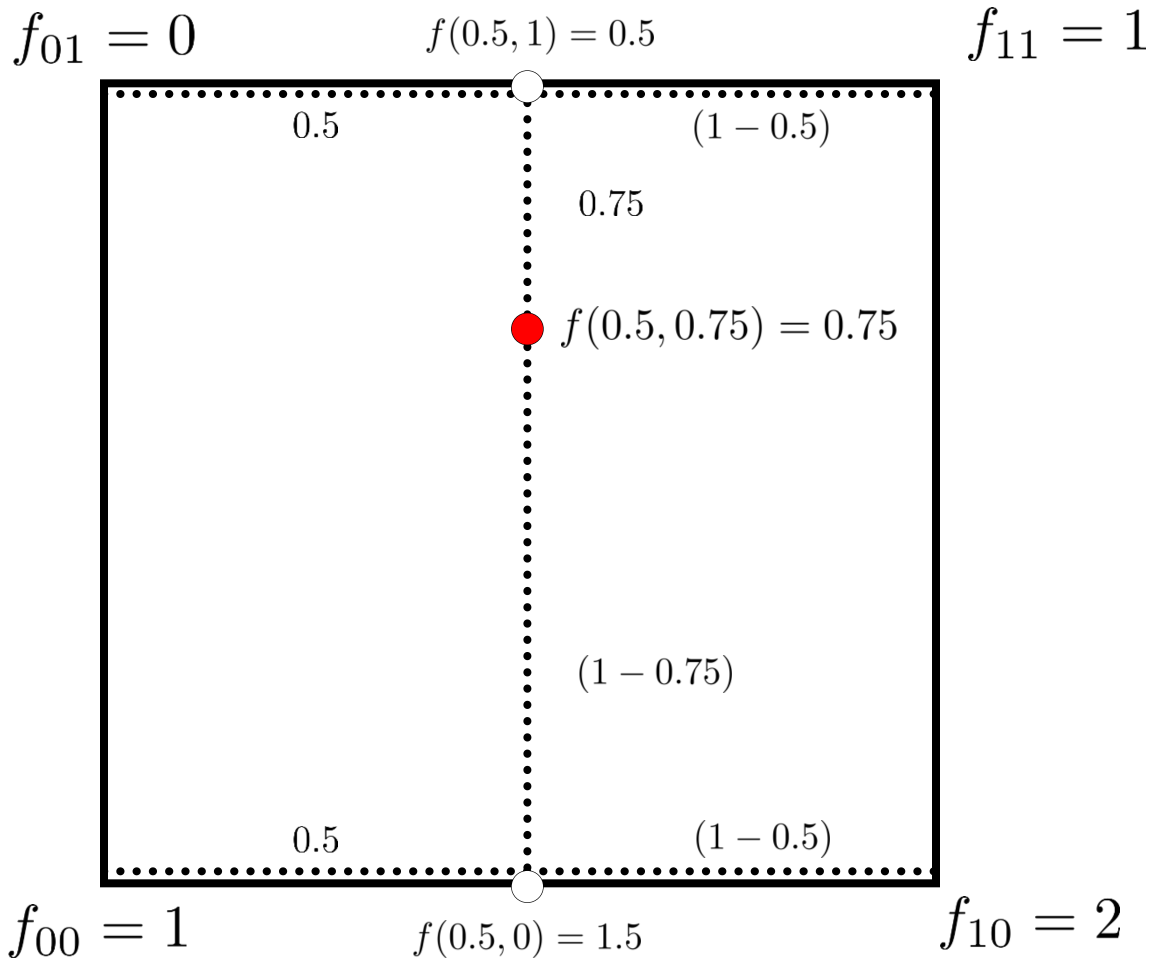


$$\begin{aligned} f(x,y) = &(1-x)\,(1-y)\,f_{00} + x\,(1-y)\,f_{10} + \\ &(1-x)\,y\,f_{01} + x\,y\,f_{11} \\ = &(1-y)\,((1-x)\,f_{00} + x\,f_{10}) + \\ &y\,((1-x)\,f_{01} + x\,f_{11}) \end{aligned}$$

“interpolate twice in x direction
and then once in y direction”

- **Example: Bi-linear interpolation in a 2D cell**
  - Repeated linear interpolation

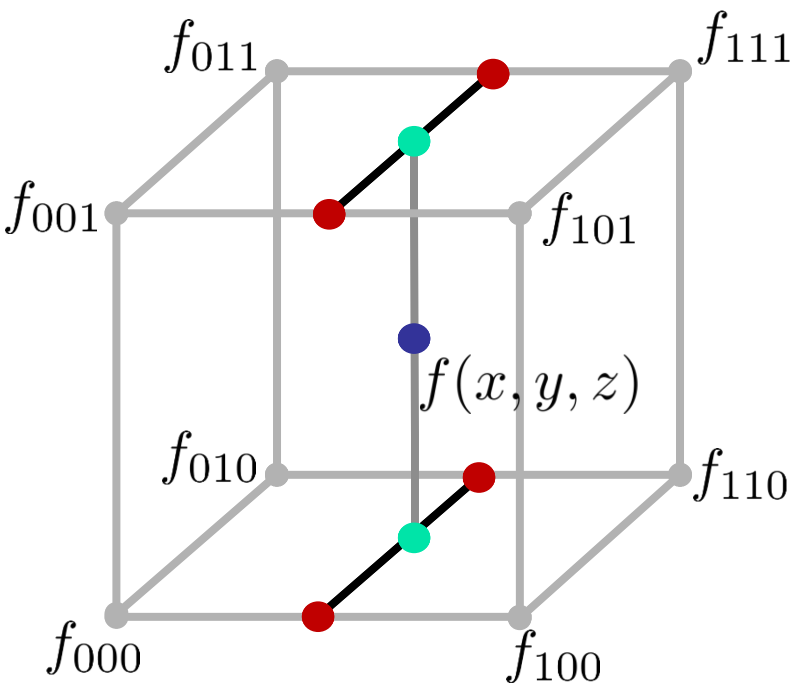
$$f(0.5, 0.75) = ?$$



- Trilinear Interpolation

3D, “tri-linear”

$$f(x,y,z) = \sum_{k=0}^{p} \sum_{j=0}^{m} \sum_{i=0}^{n} b_i(x) b_j(y) b_k(z) \, f_{ijk}$$



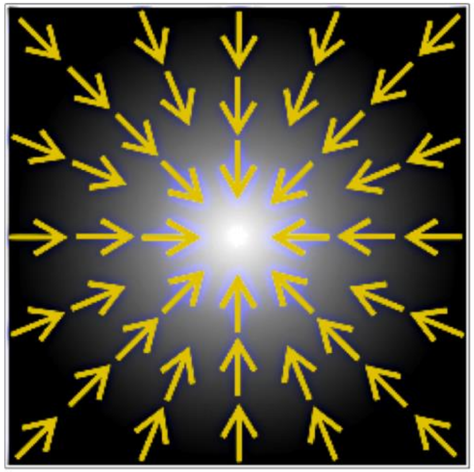
“interpolate four times in x direction, twice in y direction, and once in z direction”

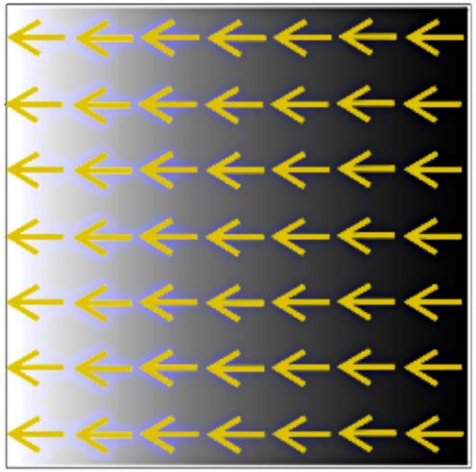
- **Function Derivative Estimation**
  - Called **Gradients** for multidimensional functions
  - Have a lot of important applications (e.g., normal for volume rendering, critical point classification for vector field topology …)

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

"vector of partial derivatives"

  - Describes direction of steepest ascend





- Two ways to estimate gradients:
  - Direct derivation of interpolation formula
  - Finite differences schemes

- **Field Function Derivatives**, Bi-Linear

$$f(x,y) = \begin{bmatrix} (1-x) & x \end{bmatrix} \begin{bmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{bmatrix} \begin{bmatrix} (1-y) \\ y \end{bmatrix} \longrightarrow$$

derive this interpolation formula

$$\frac{\partial f(x,y)}{\partial x} = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{bmatrix} \begin{bmatrix} (1-y) \\ y \end{bmatrix}$$

$$= (f_{10} - f_{00})(1-y) + (f_{11} - f_{01})\,y$$

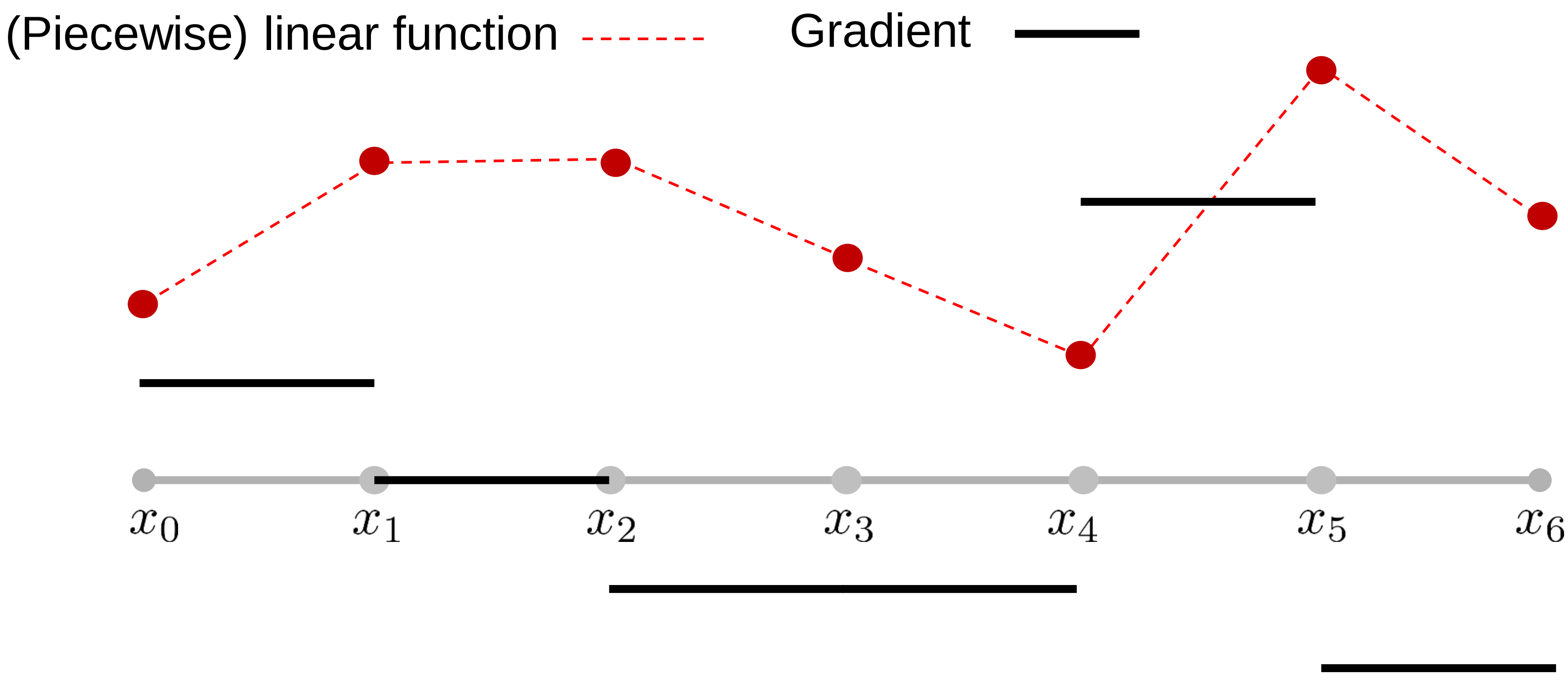
“constant in x direction“

$$\frac{\partial f(x,y)}{\partial y} = \begin{bmatrix} (1-x) & x \end{bmatrix} \begin{bmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$= (f_{01} - f_{00})(1-x) + (f_{11} - f_{10})\,x$$

“constant in y direction“

- Problem of exact linear function differentiation: discontinuous gradients



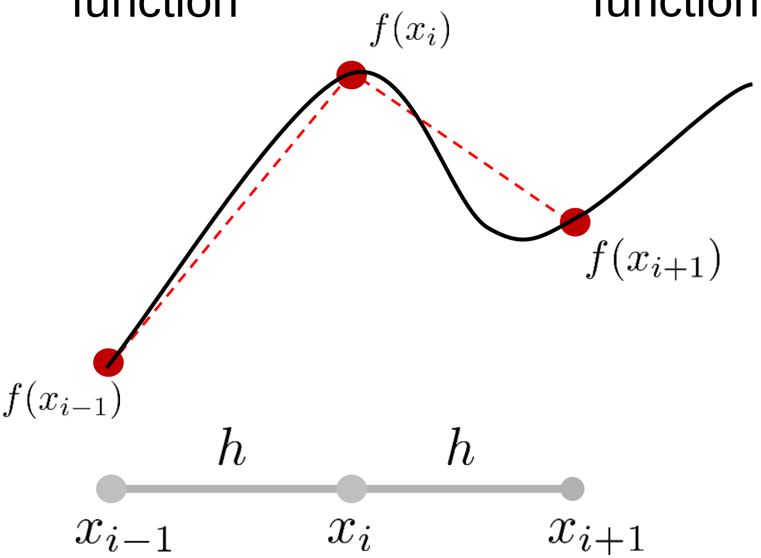
- Solution:
  - Use higher order interpolation scheme (cubic)
  - Use **finite difference estimation**

Finite Differences Schemes

- Apply Taylor series expansion around samples

Continuous function

Linear function



$f(x_{i+1})$

Taylor expansion

$$f(x_i + h) = f(x_i) + h\,\frac{\mathrm{d}f(x_i)}{\mathrm{d}x} + \frac{h^2}{2}\,\frac{\mathrm{d}^2 f(x_i)}{\mathrm{d}x^2} + O(h^3)$$

$$\rightarrow \quad \frac{\mathrm{d}f(x_i)}{\mathrm{d}x} \approx \frac{f(x_{i+1}) - f(x_i)}{h}$$

Forward difference

$$\rightarrow \quad \frac{\mathrm{d}f(x_i)}{\mathrm{d}x} \approx \frac{f(x_i) - f(x_{i-1})}{h}$$

Backward difference

- **Finite Differences Schemes**

$$f(x_{i+1}) = f(x_i) + h\,\frac{\mathrm{d}f(x_i)}{\mathrm{d}x} + \frac{h^2}{2}\,\frac{\mathrm{d}^2 f(x_i)}{\mathrm{d}x^2} + O(h^3)$$

$$f(x_{i-1}) = f(x_i) - h\,\frac{\mathrm{d}f(x_i)}{\mathrm{d}x} + \frac{h^2}{2}\,\frac{\mathrm{d}^2 f(x_i)}{\mathrm{d}x^2} + O(h^3)$$

Difference

$$\longrightarrow \quad (f(x_{i+1}) - f(x_i)) - (f(x_{i-1}) - f(x_i)) = 2h\,\frac{\mathrm{d}f(x_i)}{\mathrm{d}x} + O(h^3)$$

$$\longrightarrow \quad \frac{\mathrm{d}f(x_i)}{\mathrm{d}x} \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$$

Central difference

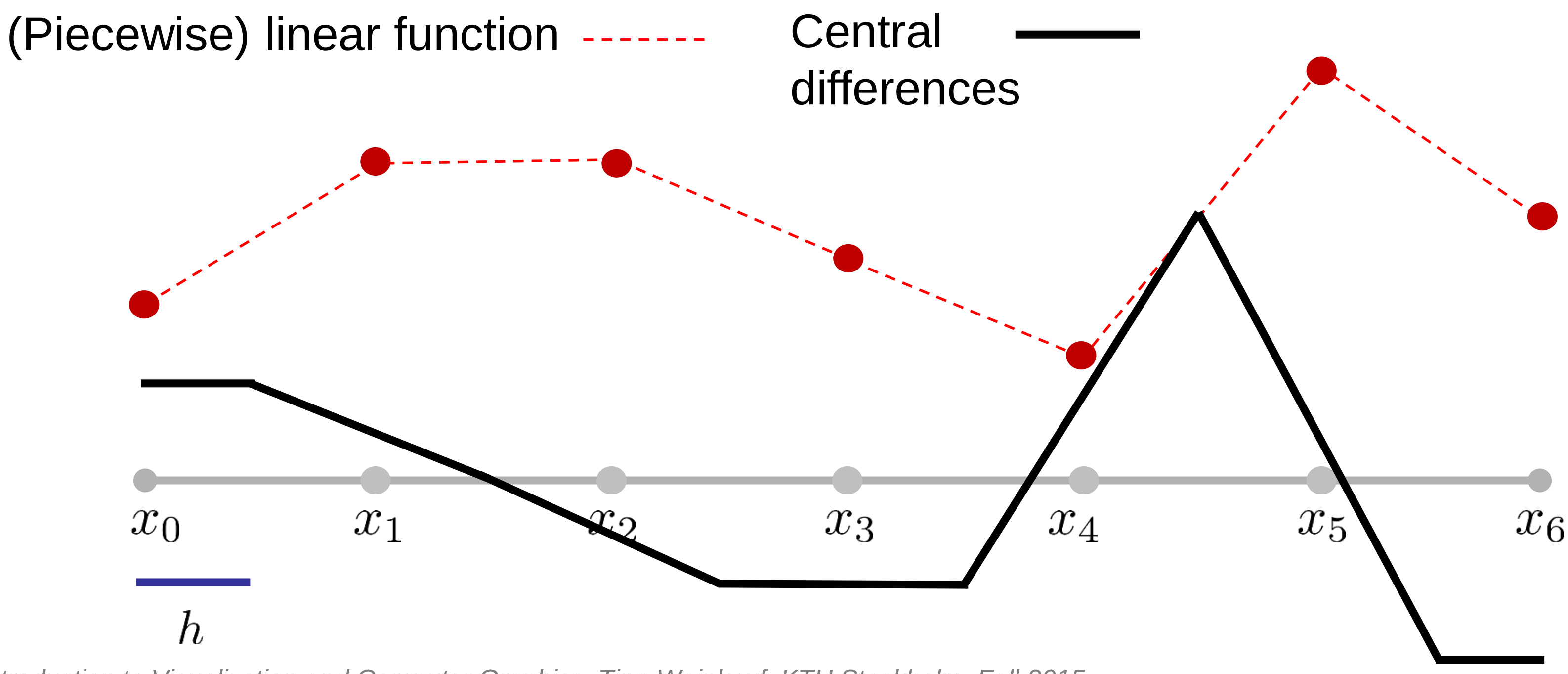
- Central differences have higher approximation order than forward / backward differences

- **Finite Differences Schemes,** Higher order derivatives

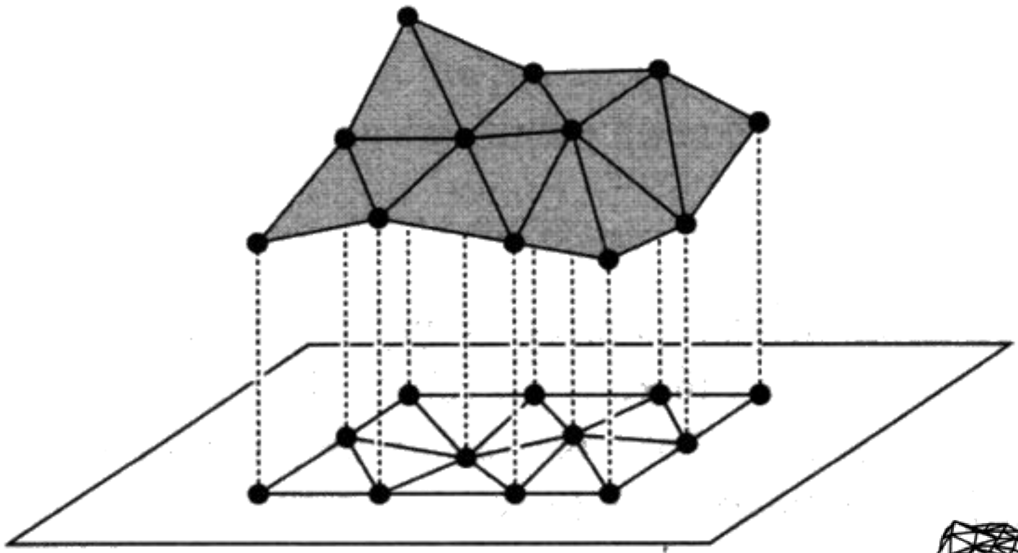
$$\frac{\mathrm{d}^2 f(x_i)}{\mathrm{d}x^2} \approx \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}$$

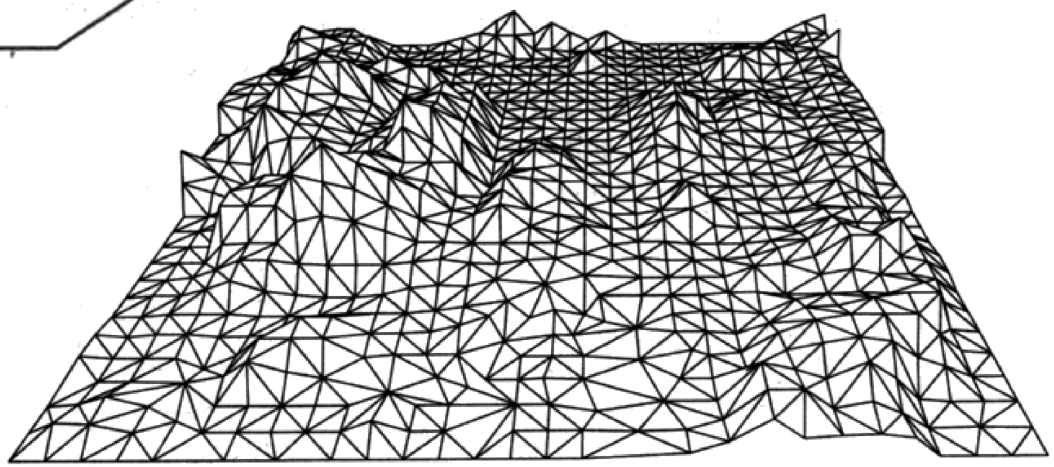
$$\frac{\partial^2 f(x_i, y_j)}{\partial xy} \approx \frac{f(x_{i+1}, y_{j+1}) - f(x_{i+1}, y_{j-1}) - f(x_{i-1}, y_{j+1}) + f(x_{i-1}, y_{j-1})}{4\, h_x\, h_y}$$

- 1D Example, linear interpolation

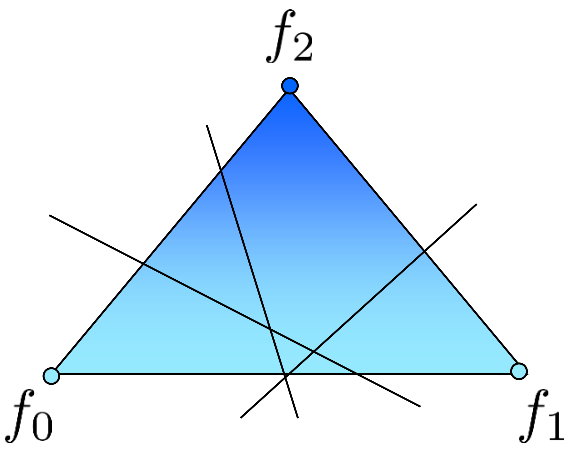


- **Piecewise Linear Interpolation in Triangle Meshes**





- **Linear Interpolation in a Triangle**
  - There is exactly one linear function that satisfies the interpolation constraint



- A linear function can be written as

$$f(x,y) = a + b\,x + c\,y$$

  - Polynomial can be obtained by

    solving the linear system

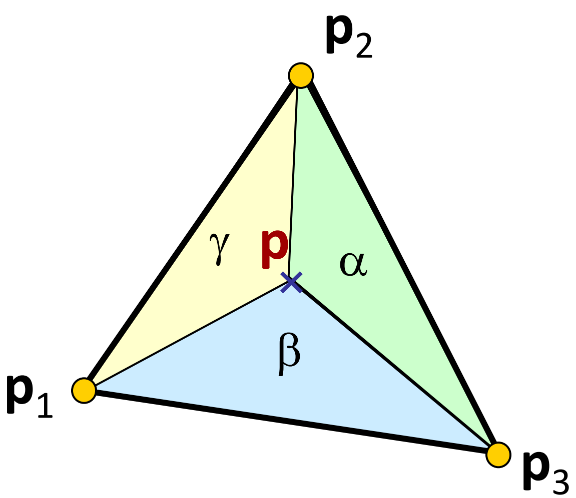
$$\begin{bmatrix} 1 & x_0 & y_0 \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix}$$

- Linear in x and y
  - Interpolated values along any ray in the plane spanned by the triangle are linear along that ray

- Barycentric Coordinates:
  - Planar case:
    Barycentric combinations of 3 points

$$\mathbf{p} = \alpha \mathbf{p}_1 + \beta \mathbf{p}_2 + \gamma \mathbf{p}_3, \text{with} : \alpha + \beta + \gamma = 1$$

$$\gamma = 1 - \alpha - \beta$$

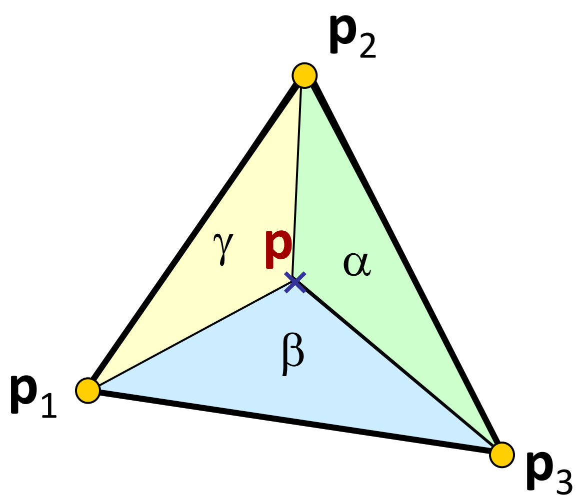


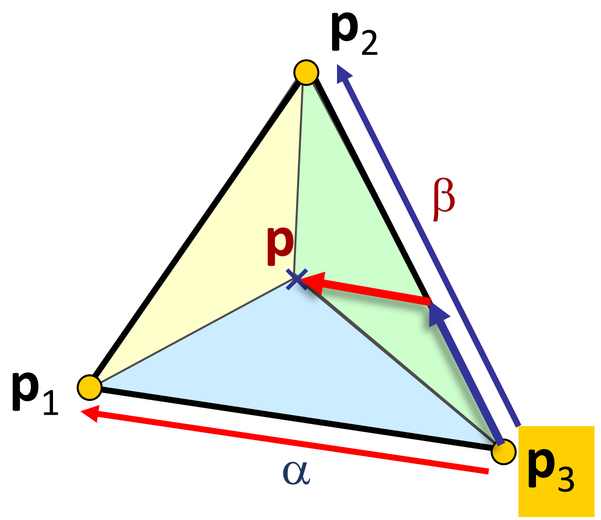
  - Area formulation:

$$\alpha = \frac{area(\Delta(\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}))}{area(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3))}, \beta = \frac{area(\Delta(\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}))}{area(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3))}, \gamma = \frac{area(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}))}{area(\Delta(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3))}$$

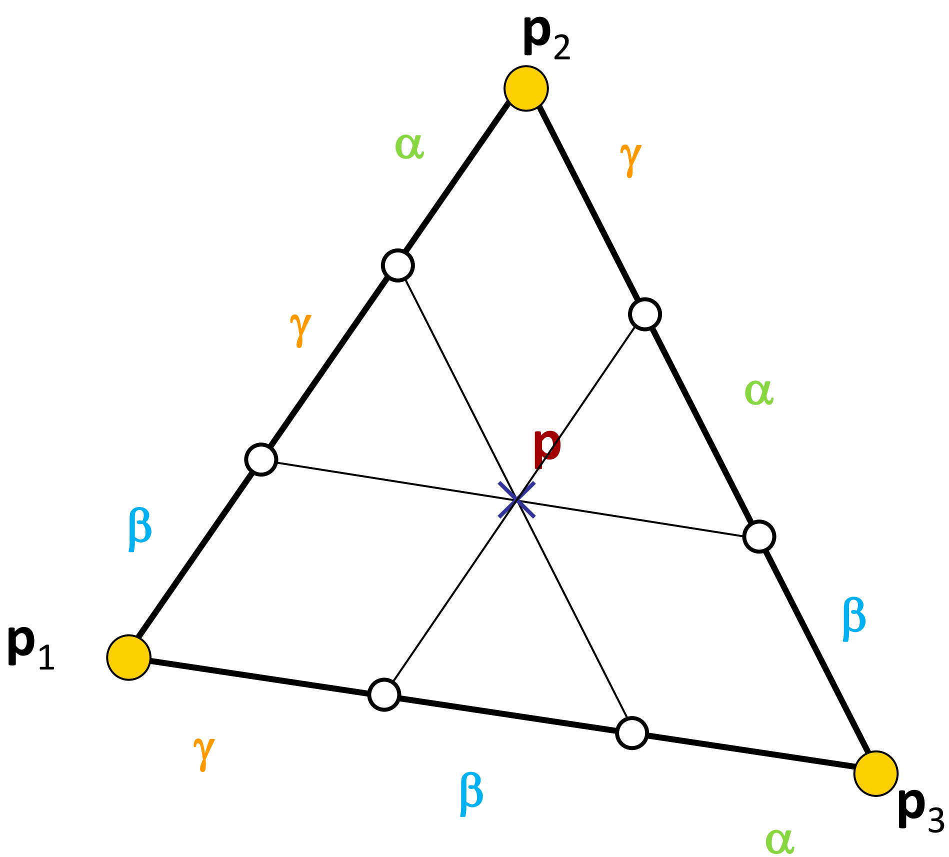
- Barycentric Coordinates:
  - Linear formulation:

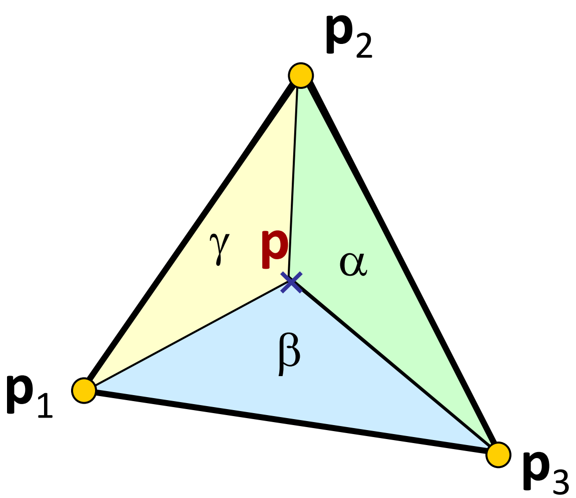
$$\begin{aligned}\mathbf{p} &= \alpha\mathbf{p}_1 + \beta\,\mathbf{p}_2 + \gamma\,\mathbf{p}_3\\ &= \alpha\mathbf{p}_1 + \beta\,\mathbf{p}_2 + (1-\alpha-\beta)\,\mathbf{p}_3\\ &= \alpha\mathbf{p}_1 + \beta\,\mathbf{p}_2 + \mathbf{p}_3 - \alpha\,\mathbf{p}_3 - \beta\,\mathbf{p}_3\\ &= \boxed{\mathbf{p}_3} + \underline{\alpha\left(\mathbf{p}_1 - \mathbf{p}_3\right)} + \underline{\beta\left(\mathbf{p}_2 - \mathbf{p}_3\right)}\end{aligned}$$





$$\mathbf{p} = \alpha\mathbf{p}_1 + \beta\,\mathbf{p}_2 + \gamma\,\mathbf{p}_3, \text{with} : \alpha + \beta + \gamma = 1$$





- **Barycentric Interpolation in a Triangle**

very important

  - The linear function of a triangle can be computed at any point as

    $$f(x,y) = \alpha_0(x,y) f_0 + \alpha_1(x,y) f_1 + \alpha_2(x,y) f_2$$

    with $\alpha_0 + \alpha_1 + \alpha_2 = 1$ (**Barycentric Coordinates**)

  - This also holds for the coordinate $\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$ of the triangle:

    $$\mathbf{x} = \alpha_0\,\mathbf{x}_0 + \alpha_1\,\mathbf{x}_1 + \alpha_2\,\mathbf{x}_2$$

    → Can be used to solve for unknown coefficients α_i :

$$\begin{bmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

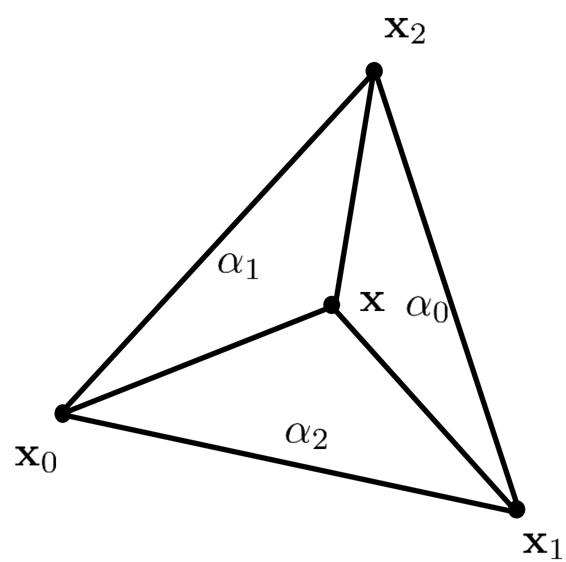
• Barycentric Interpolation in a Triangle

- Solution of $\begin{bmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ (e.g. Cramer's rule):

$$\alpha_0 = \frac{1}{2A} \det\left(\begin{bmatrix} x & x_1 & x_2 \\ y & y_1 & y_2 \\ 1 & 1 & 1 \end{bmatrix} \right) \qquad \alpha_0 = \frac{\mathrm{Area}([\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2])}{\mathrm{Area}([\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2])}$$

$$\alpha_1 = \frac{1}{2A} \det\left(\begin{bmatrix} x_0 & x & x_2 \\ y_0 & y & y_2 \\ 1 & 1 & 1 \end{bmatrix} \right) \qquad \alpha_1 = \frac{\mathrm{Area}([\mathbf{x}_0, \mathbf{x}, \mathbf{x}_2])}{\mathrm{Area}([\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2])}$$

$$\alpha_2 = \frac{1}{2A} \det\left(\begin{bmatrix} x_0 & x_1 & x \\ y_0 & y_1 & y \\ 1 & 1 & 1 \end{bmatrix} \right) \qquad \alpha_2 = \frac{\mathrm{Area}([\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}])}{\mathrm{Area}([\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2])}$$



with

$$A = \frac{1}{2} \det\left(\begin{bmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{bmatrix} \right)$$

Inside triangle criteria

$$0 \leq \alpha_0,\ \alpha_1,\ \alpha_2 \leq 1$$

- **Barycentric Interpolation in a Tetrahedron**
- Analogous to the triangle case

Gradient of a linearly interpolated function in a triangle/tetrahedron

- **Constant!**