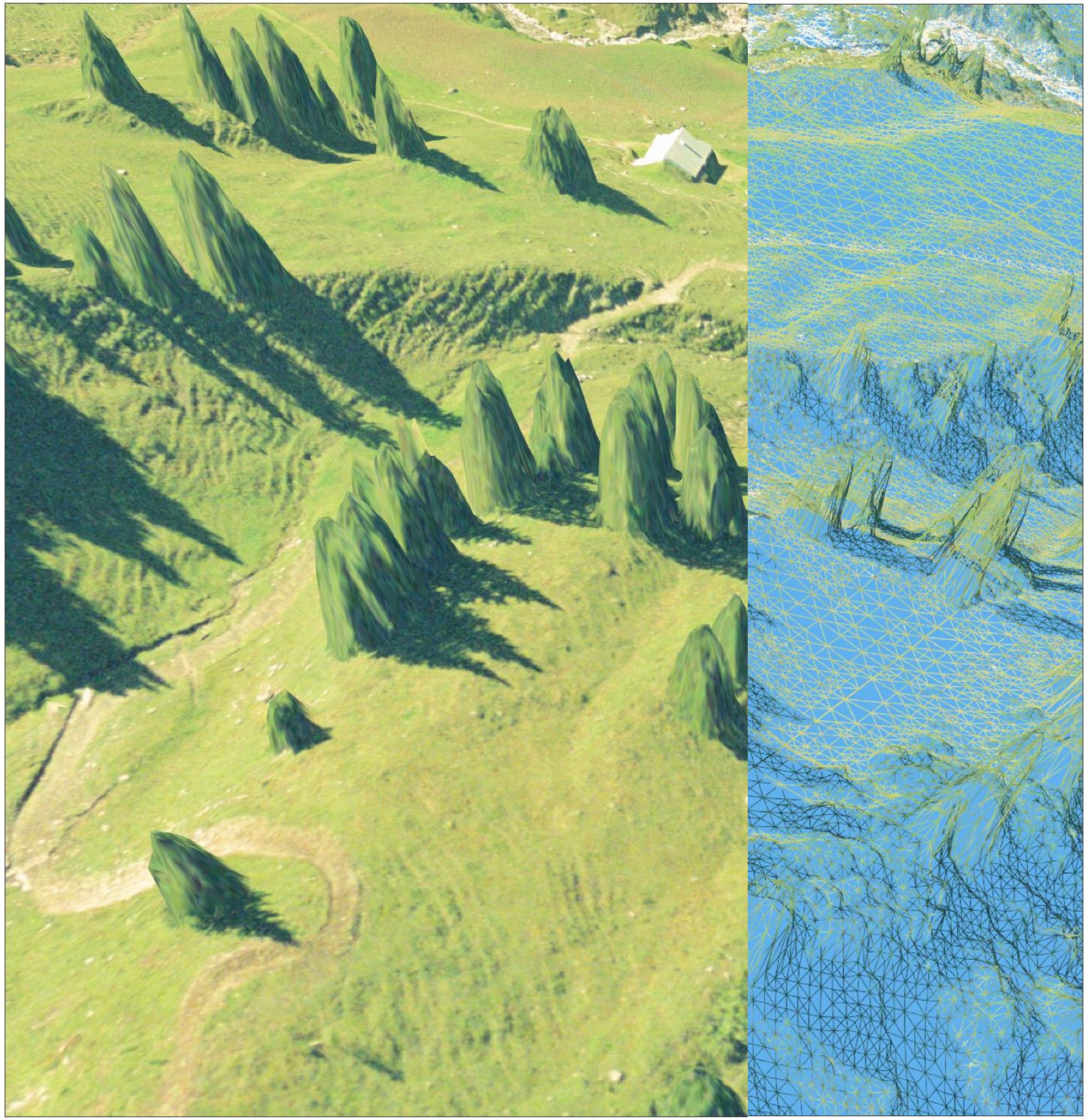
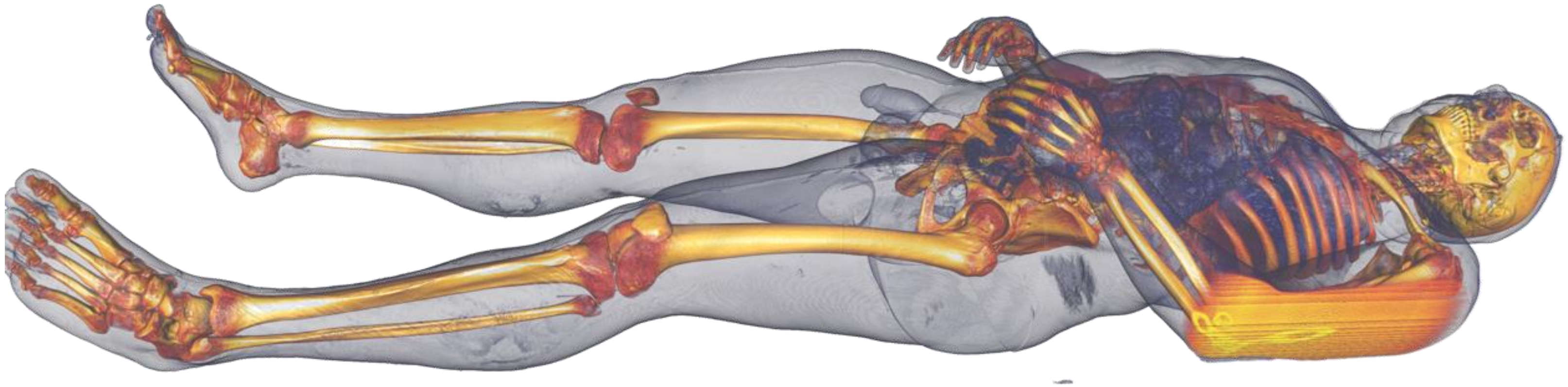
Visualization, DD2257

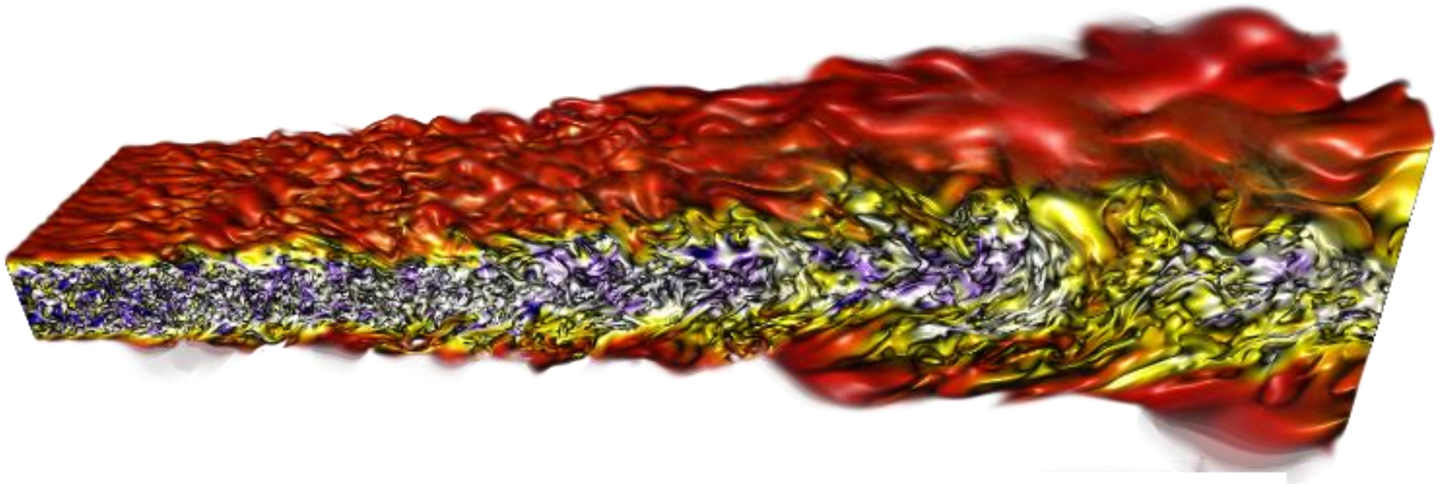
Prof. Dr. Tino Weinkauf

Data Description

Sampled Data



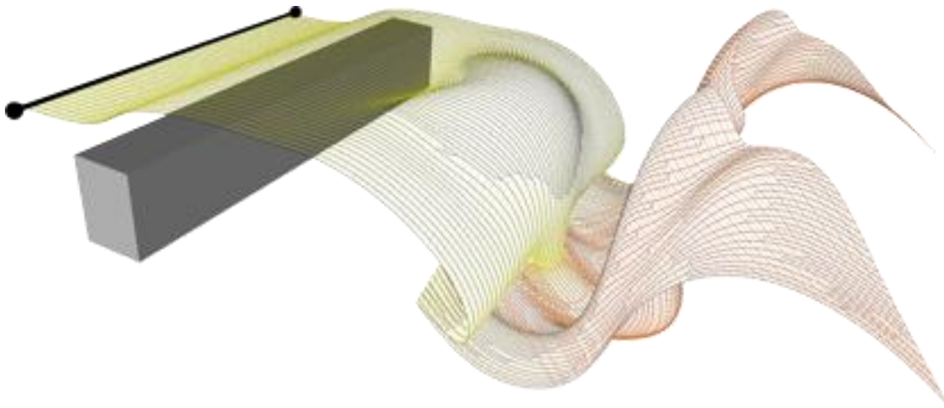


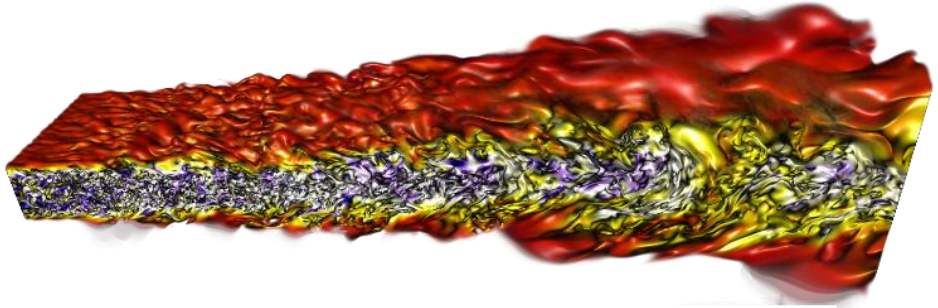


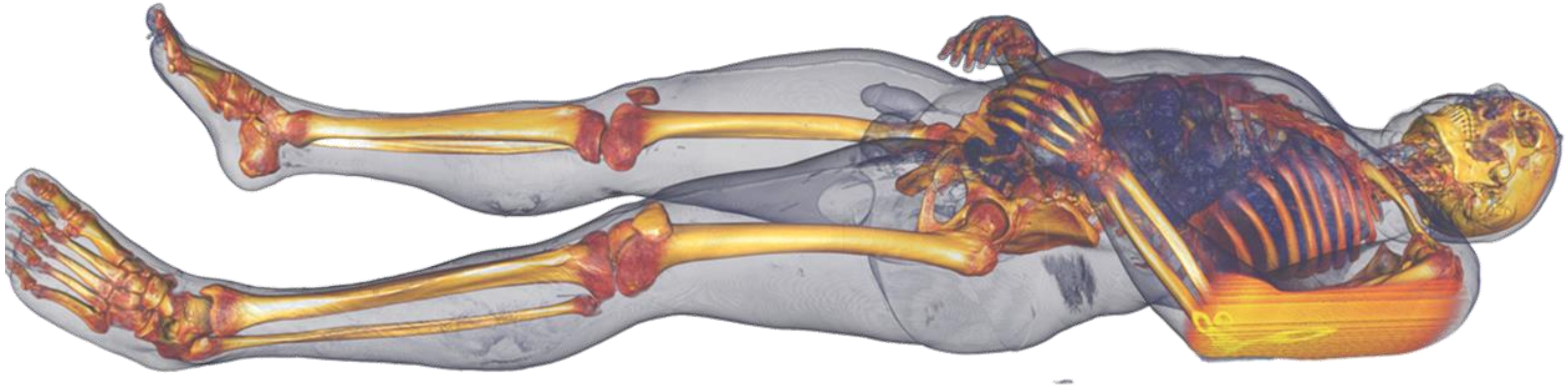
- In most cases, the visualization data represents a **continuous real object**, e.g., an oscillating membrane, a velocity field around a body, an organ, human tissue, etc.
  - This object lives in an n-dimensional space - the **domain** (aka. observation space)
- Usually, the data is only given at a 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 representation** of a continuous object

Discrete Representations

We usually deal with the **reconstruction** of a continuous real object from a given discrete representation.



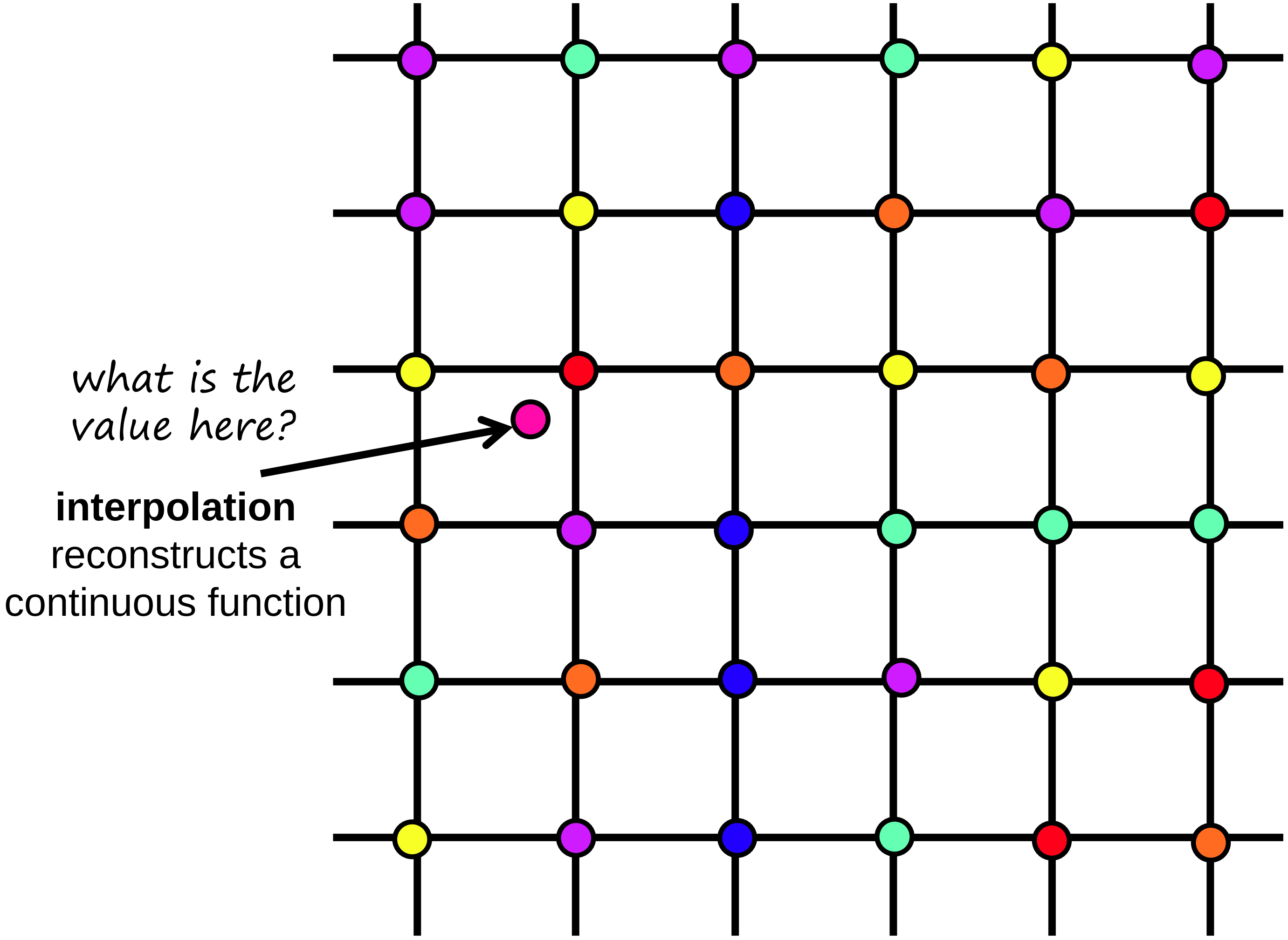




Discrete Representations

We usually deal with the **reconstruction** of a continuous real object from a given discrete representation.

Samples are connected to each other to form **grids / meshes**, covering the entire domain.



Grid terminology

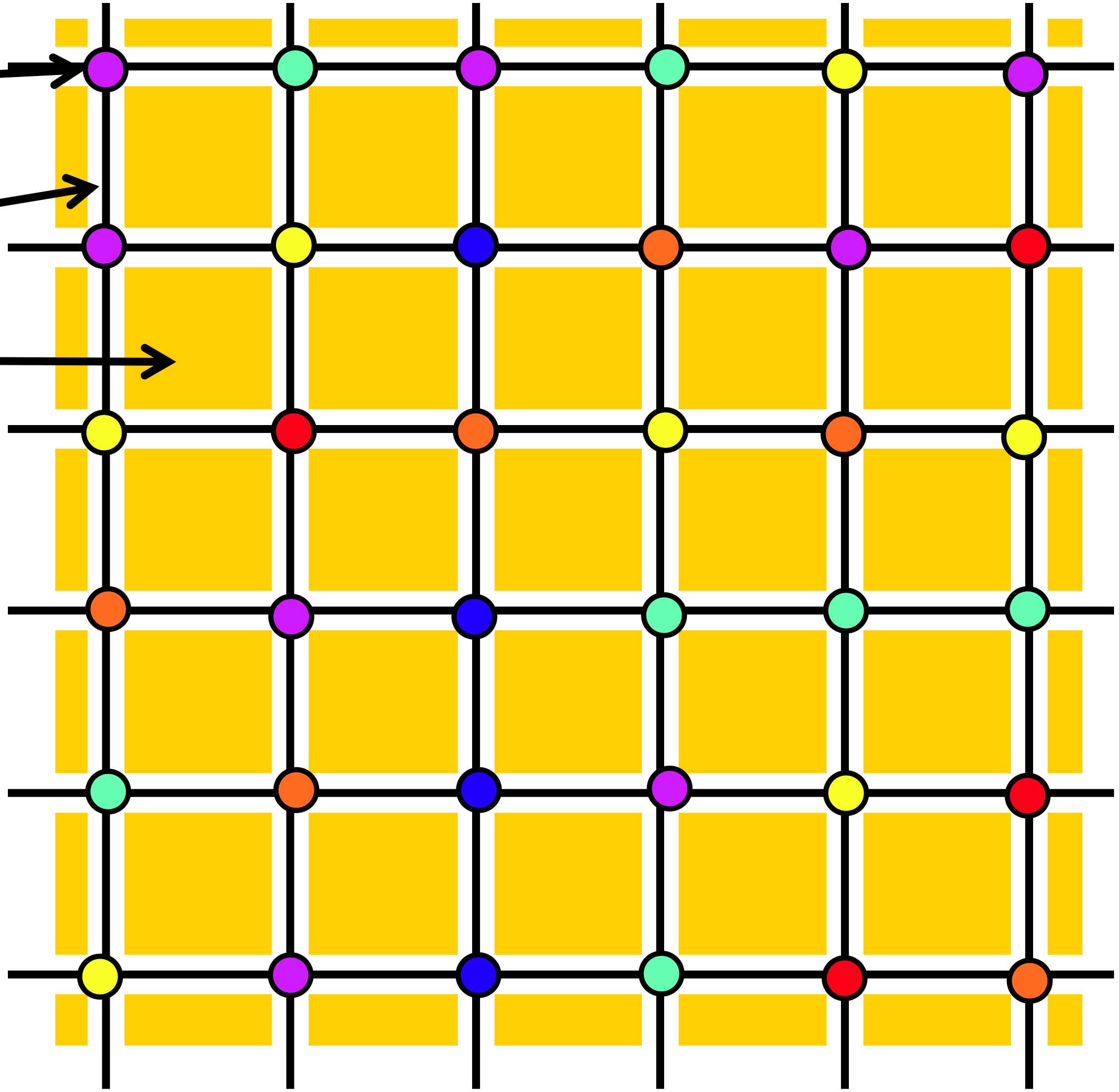
grid cell: largest-dimensional element in a grid
2D: grid face
3D: grid voxel

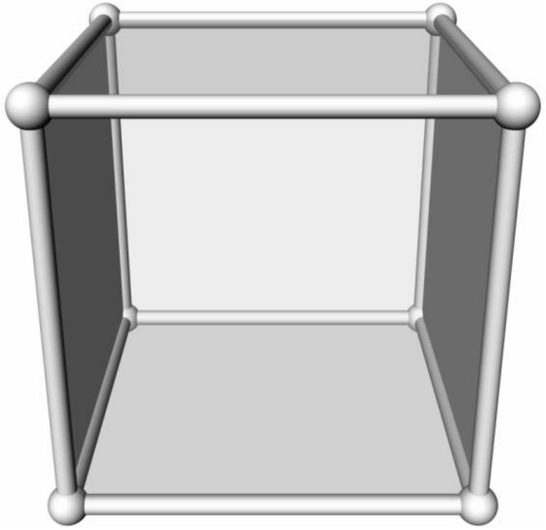
0D: grid vertex
(grid point)

1D: grid line

2D: grid face

3D: grid voxel





grid vertices
grid lines
grid faces
grid voxel

Operations on Grids

- Determine the **data value** at a position
  - Easy at the grid vertices
  - At other positions: Interpolation Schemes
- Determine neighbors
- Convert to other grid types
- Compute metrics
  - Distance, Area, Volume
- Compute Bounding Box

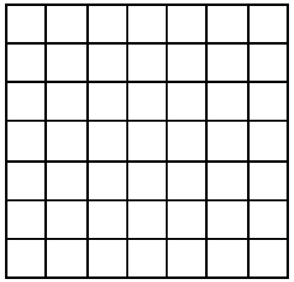
Data Connectivity

- 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

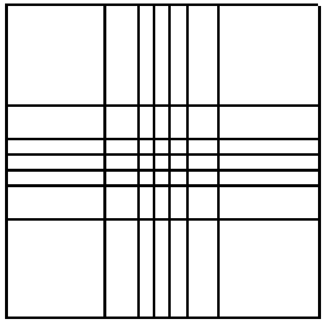
very important

Structured grids

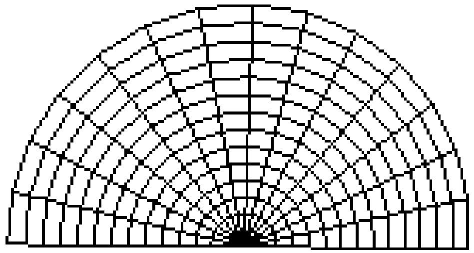
- “Structured” refers to the matrix-like connectivity between the grid vertices
- We distinguish different types of structured grids regarding the alignment to the coordinate system and the size of cells



uniform grid
axis-aligned, identical cells
implicitly given coordinates



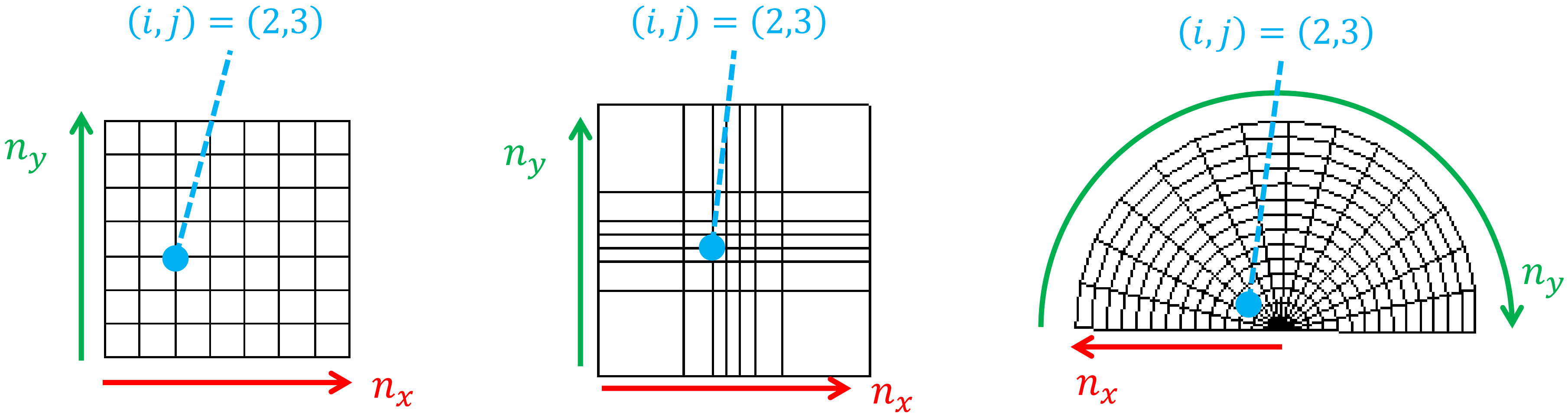
rectilinear grid
axis-aligned, cells different size
semi-implicitly given coordinates



curvilinear grid
not axis-aligned, cells different size
explicitly given coordinates

Structured grids

- Number of grid vertices: n_x, n_y, n_z
- We can address every **grid vertex** with an index tuple (i, j, k)
  - $0 \leq i < n_x$ $\quad$ $0 \leq j < n_y$ $\quad$ $0 \leq k < n_z$



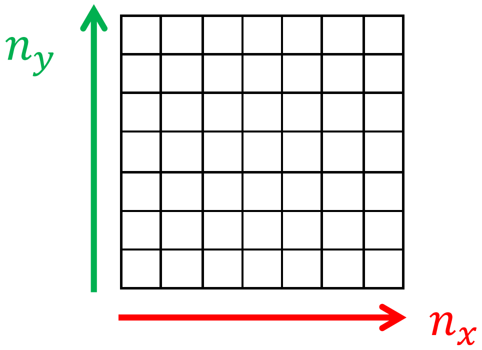
uniform grid
axis-aligned, identical cells
implicitly given coordinates

rectilinear grid
axis-aligned, cells different size
semi-implicitly given coordinates

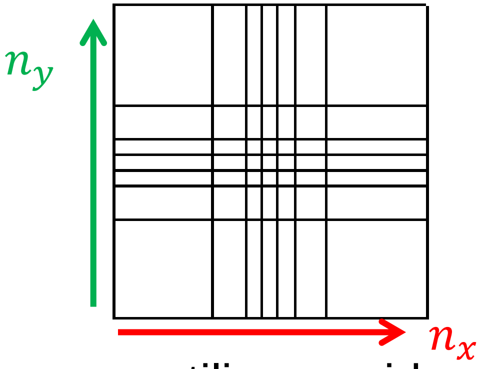
curvilinear grid
not axis-aligned, cells different size
explicitly given coordinates

Structured grids

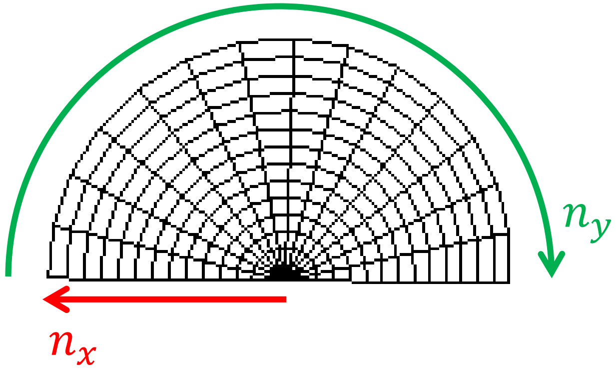
- Number of grid vertices: n_x, n_y, n_z
- We can address every **grid cell** with an index tuple (i, j, k)
  - $0 \leq i < n_x - 1$ $\quad$ $0 \leq j < n_y - 1$ $\quad$ $0 \leq k < n_z - 1$
- ➔ Number of cells: $(n_x - 1) \times (n_y - 1) \times (n_z - 1)$



uniform grid
axis-aligned, identical cells
implicitly given coordinates



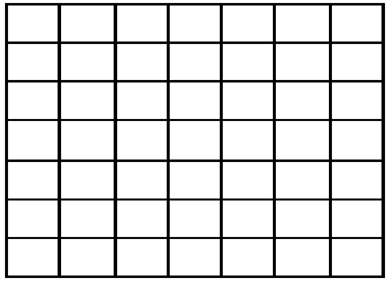
rectilinear grid
axis-aligned, cells different size
semi-implicitly given coordinates

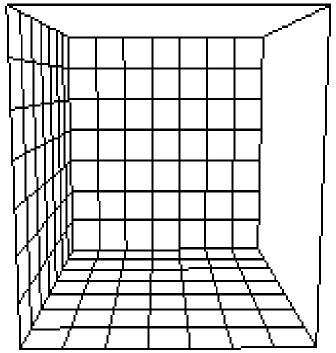


curvilinear grid
not axis-aligned, cells different size
explicitly given coordinates

Regular or uniform grids

- Cells are rectangles or rectangular cuboids of the same size
- All grid lines are parallel to the axes

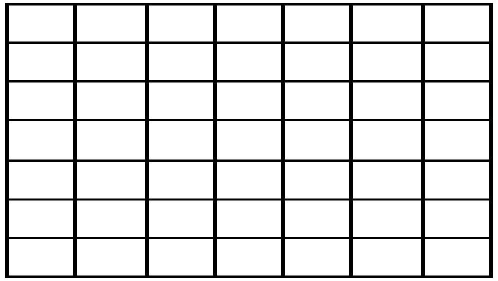


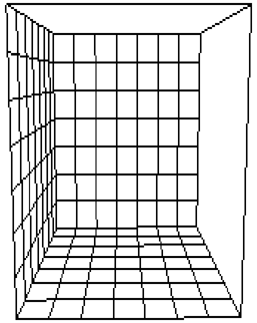


- To define a uniform grid, we need the following:
  - Bounding box: $(x_{min}, y_{min}, z_{min}) - (x_{max}, y_{max}, z_{max})$
  - Number of grid vertices in each dimension: n_x, n_y, n_z

➔ from that information we can derive the Cell size: d_x, d_y, d_z

Regular or uniform grids



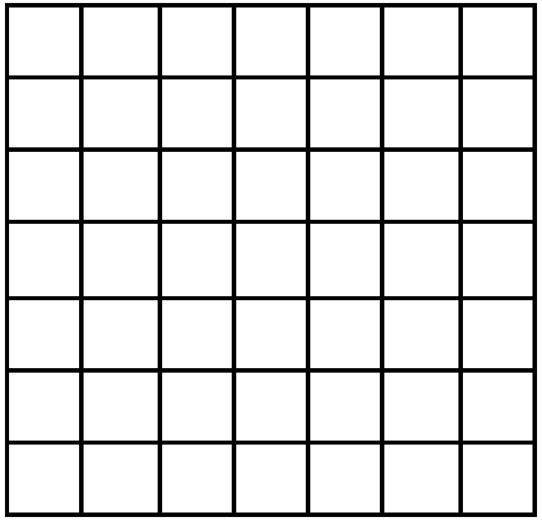


- Well suited for image data (medical applications)
- Coordinate ➔ cell is very simple and cheap
  - Global search is good enough; local search not required
- Coordinate of a grid vertex:

$$(i \cdot d_x, j \cdot d_y, k \cdot d_z) + (x_{min}, y_{min}, z_{min})$$

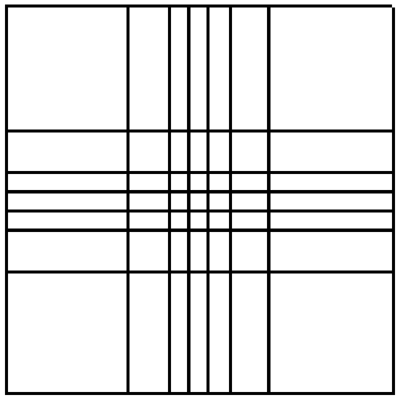
Cartesian grid

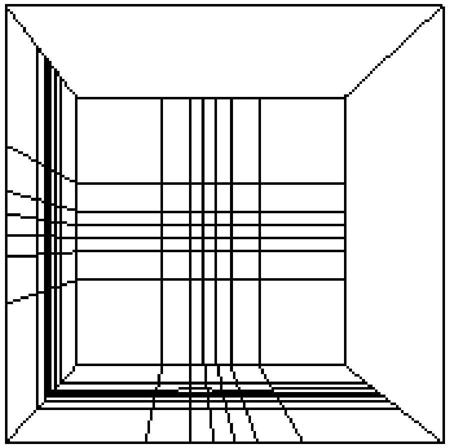
- Special case of a uniform grid: $d_x = d_y = d_z$
- Consists of squares (2D), cubes (3D)



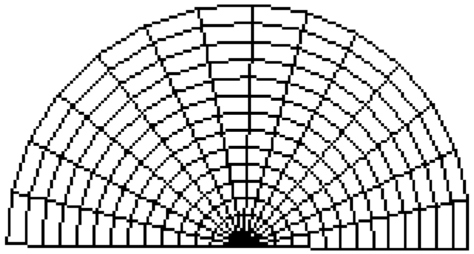
Rectilinear grids

- 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

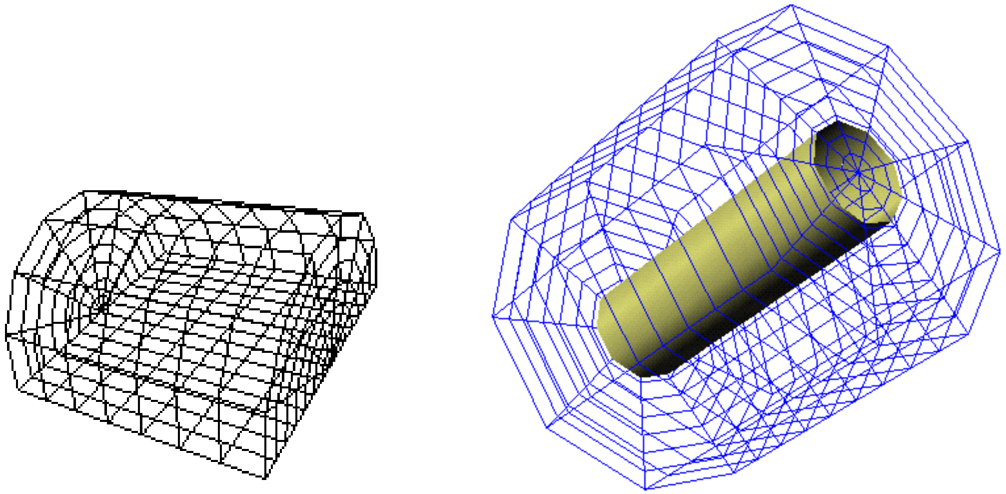




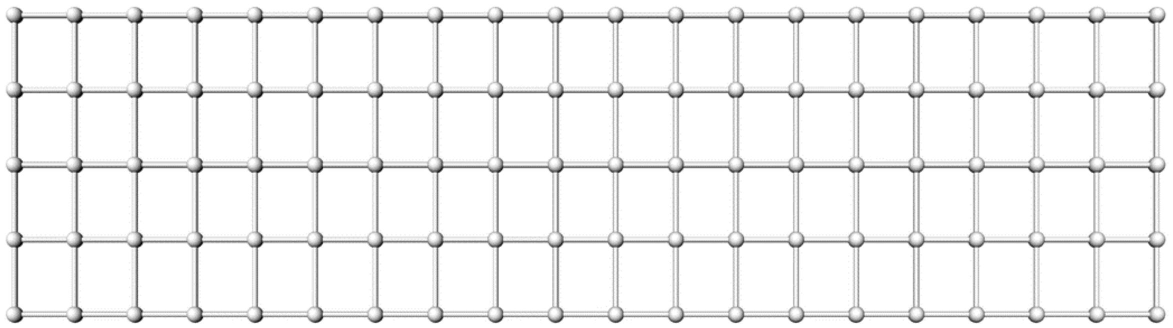
- **Curvilinear grids**
- Vertex locations are explicitly given
  - XYZLoc = {(0.0, -1.0, 3.0), (1.5, 0.3, 3.5), (2.0, 1.0, 3.6), …}
- Cells are quadrilaterals or cuboids
- Grid lines are not (necessarily) parallel to the axes



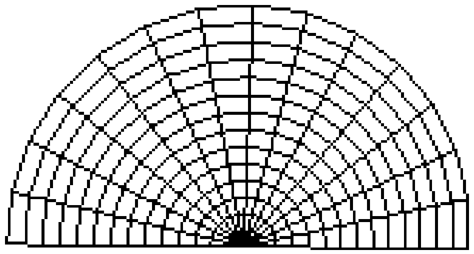
2D curvilinear grid



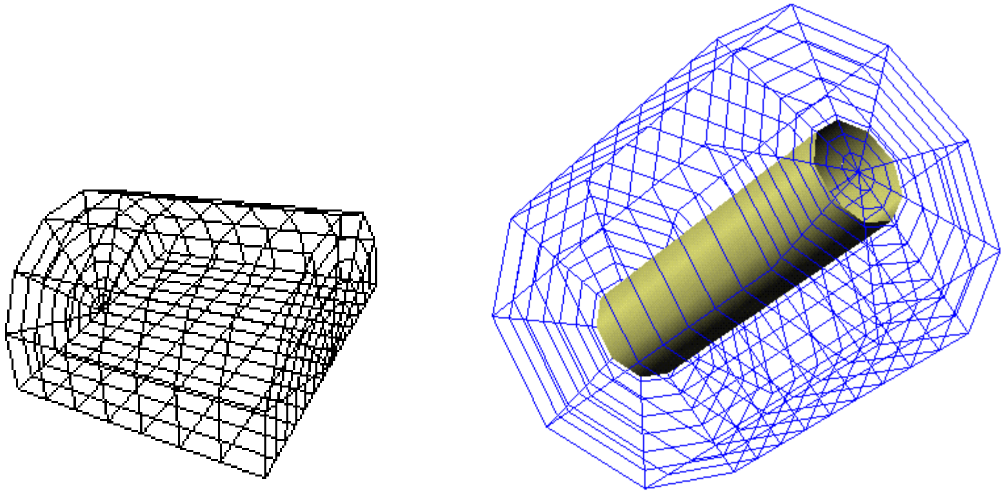
3D curvilinear grids



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



2D curvilinear grid

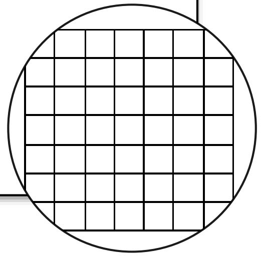


3D curvilinear grids

Data structures for structured grids

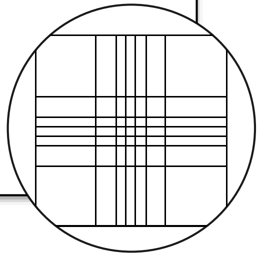
- dependent variable: array
- positions: implicit, no storage needed

uniform grid



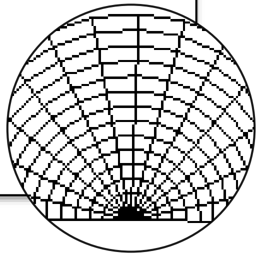
- dependent variable: array
- positions: 2/3 location vectors

rectilinear grid

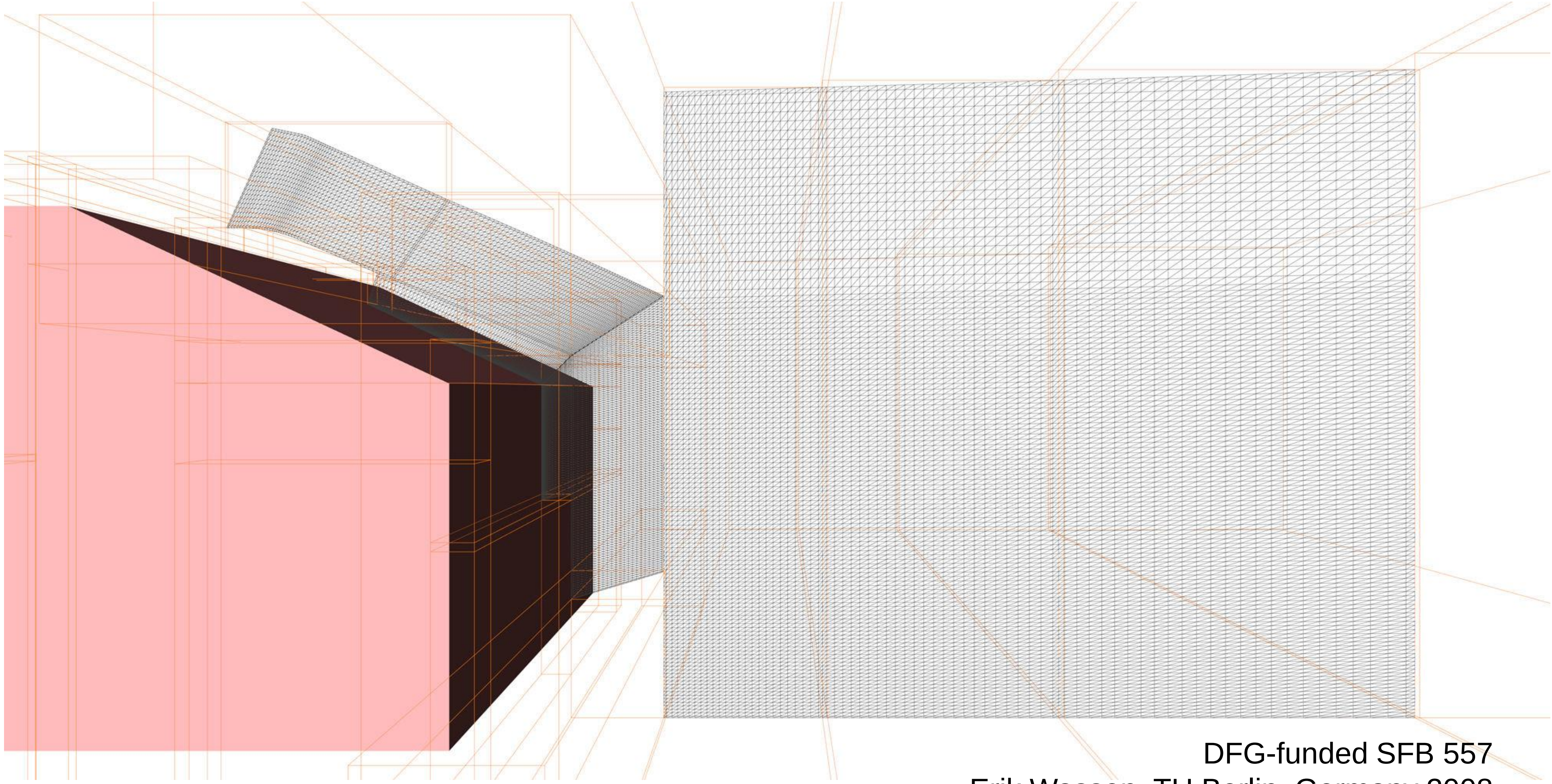


- dependent variable: array
- positions: array

curvilinear grid

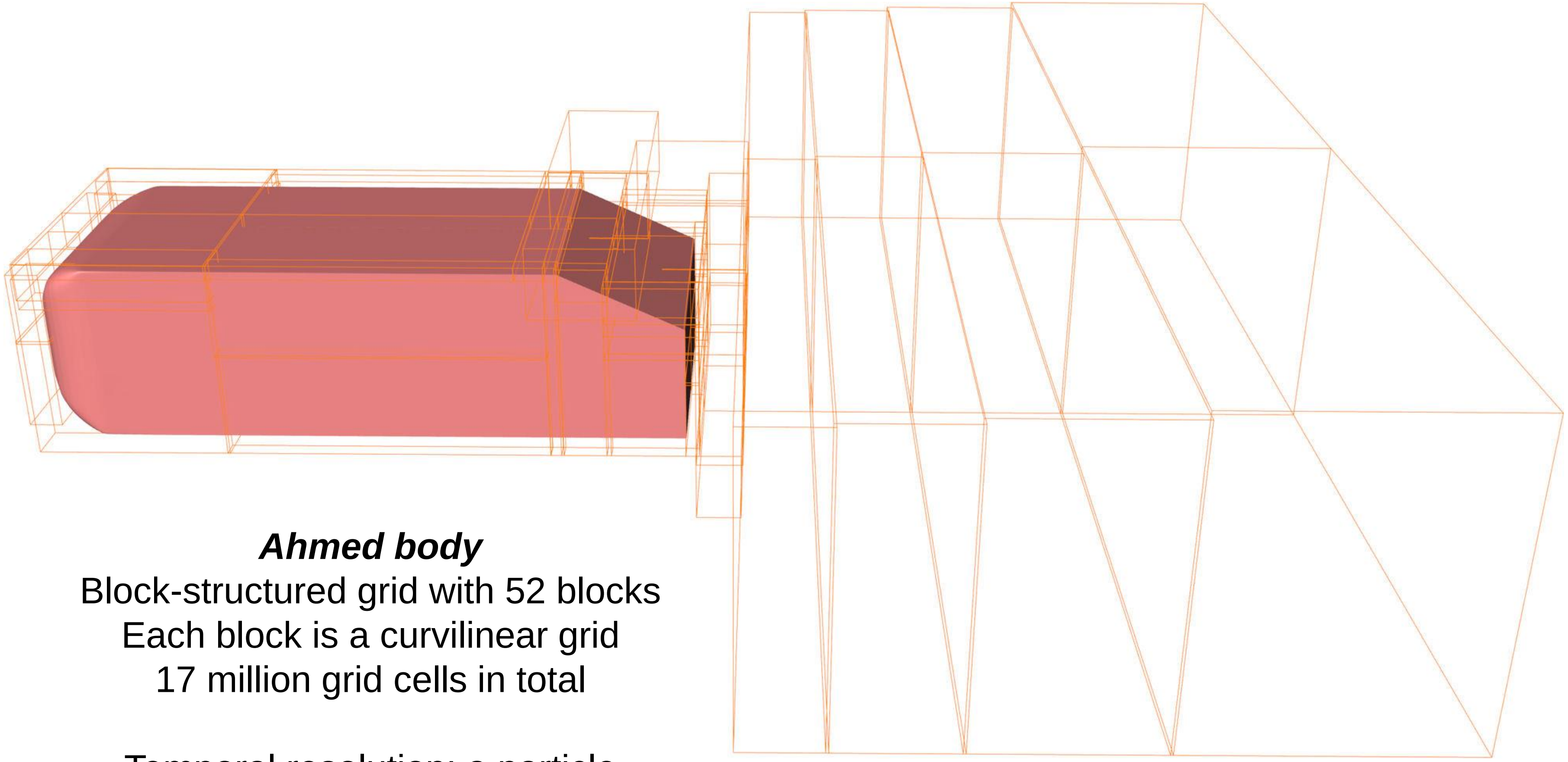


- **Block-structured grids**
- combination of several structured grids



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

- Demands on data storage, an example:

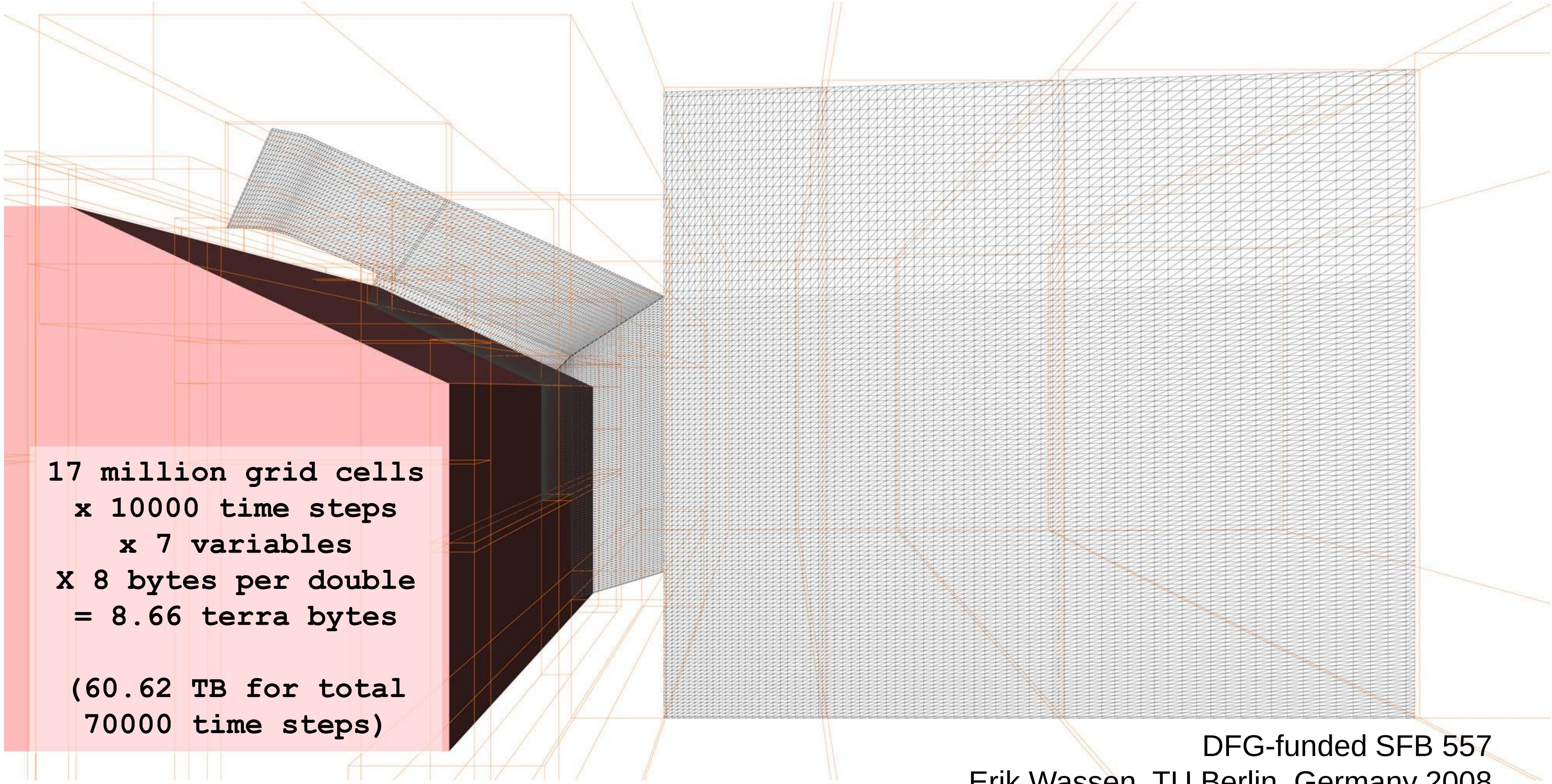


Ahmed body
Block-structured grid with 52 blocks
Each block is a curvilinear grid
17 million grid cells in total

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

sample points are not laid out in a matrix-like fashion

unstructured grids connect neighboring samples

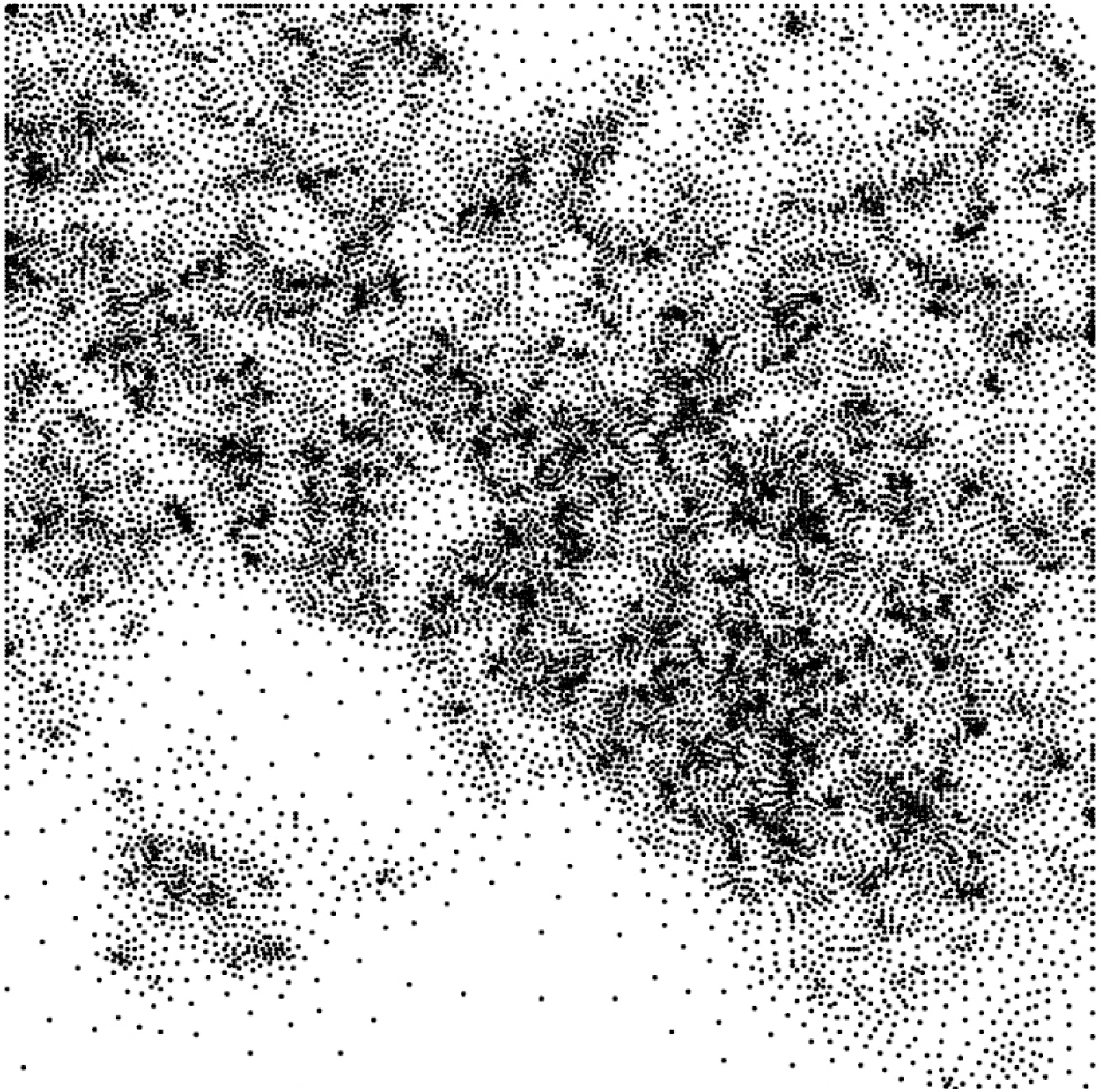
many possibilities how to do this

triangle/tetrahedral mesh

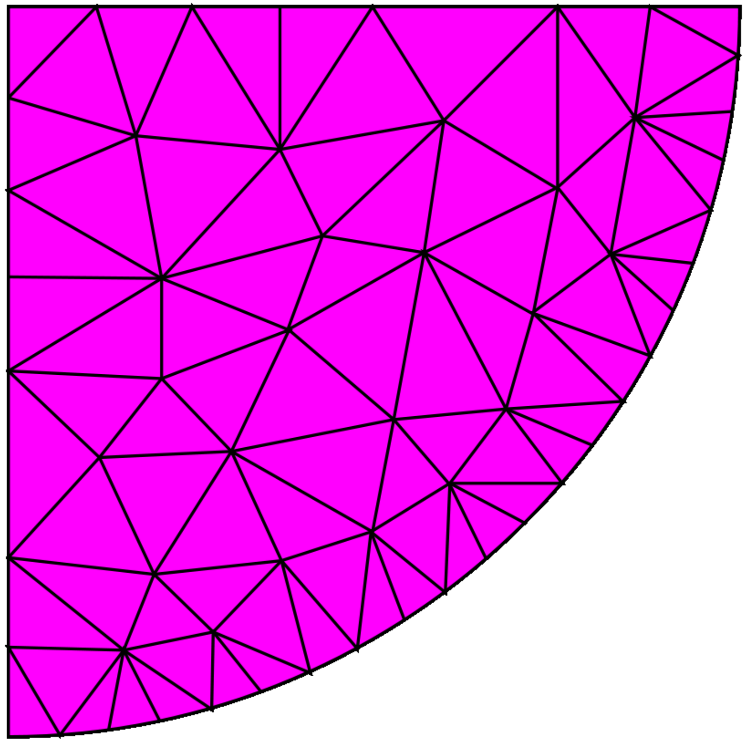
2D/3D linear cells

quad/hexahedral mesh

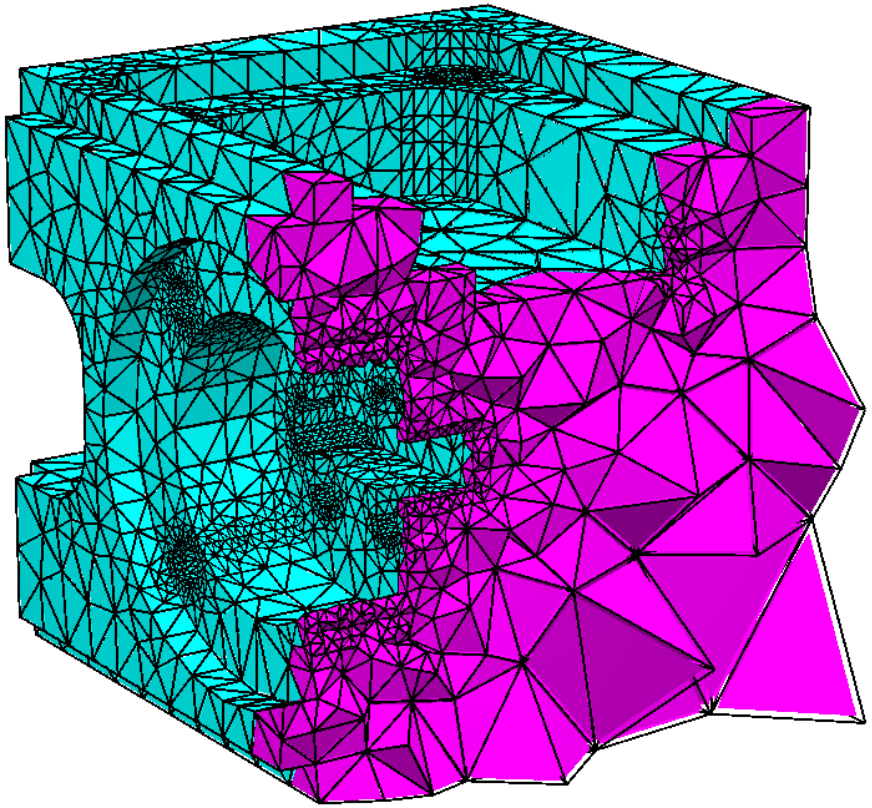
2D/3D cube-like cells



Unstructured Grids: Triangle / Tetrahedral Meshes



2D unstructured grid
consisting of triangles



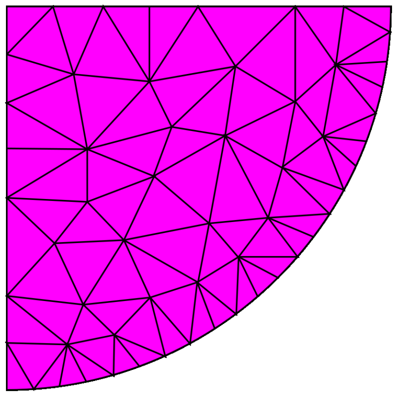
3D unstructured grid
consisting of tetrahedra
(from TetGen user manual)

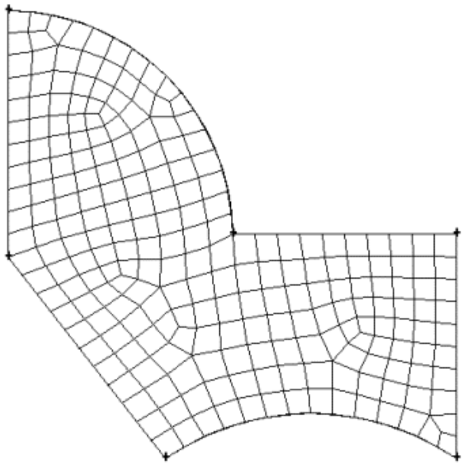
Unstructured Grids

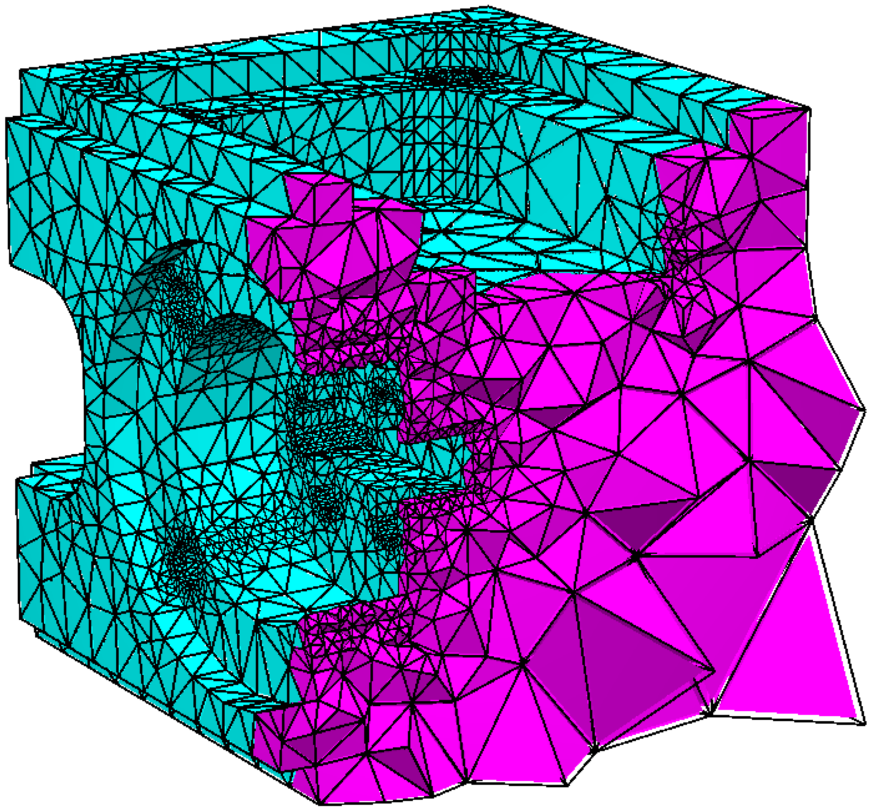
Vertex locations and connectivity explicitly given

Coordinate ➔ cell:

- Local search within last cell or its immediate neighbors
- Global search via quadtree/octree







Data structures for Triangles Meshes

shared vertex data structure

vertex table

stores positions

triangle table

stores indices into vertices

gives “triangle soup”

from this we can derive:

list of edges

vertex neighbors

Vertices
x_1 y_1 z_1
...
x_V y_V z_V

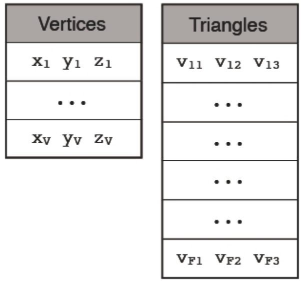
Triangles
v_{11} v_{12} v_{13}
...
...
...
...
v_{F1} v_{F2} v_{F3}

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

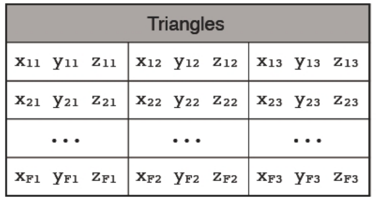
no neighborhood info

Data structures for Triangles Meshes

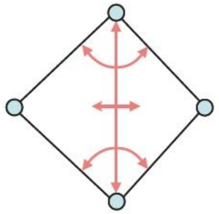
many other options exist



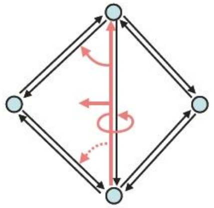
12 B/v + 12 B/f = 36 B/v
no neighborhood info



36 B/f = 72 B/v
no connectivity!

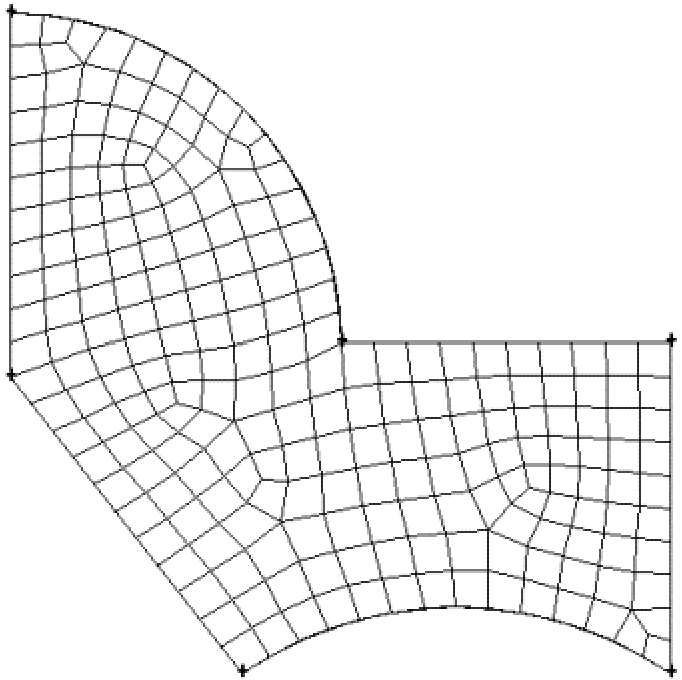


120 B/v
edge orientation?



96 to 144 B/v
no case distinctions
during traversal

Unstructured Grids: Quad Mesh

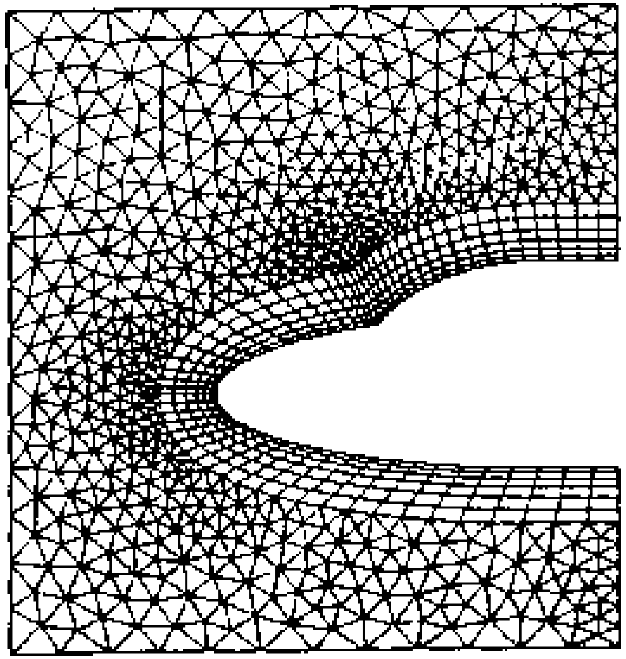


2D unstructured grid consisting of quads

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

Hybrid Grids

Combination of different grid types



2D hybrid grid

Summary

- Continuous objects are often the subject of study
  - Need to be sampled to be represented in the computer
- Data is stored at samples (most often points)
- Structured grids: matrix-like organization of neighborhood
  - uniform
  - rectilinear
  - curvilinear
- Unstructured grids: arbitrary organization of neighborhood
  - triangle/tetrahedral meshes
  - quad/hexahedral meshes