King Abdullah University of Science and Technology KAUST

CS 247 – Scientific Visualization
Lecture 4: Data Representation, Pt. 2

Markus Hadwiger, KAUST

Reading Assignment #2 (until Feb 7)

Read (required):

- Data Visualization book, finish Chapter 2
- Data Visualization book, Chapter 3 until 3.5 (inclusive)
- Data Visualization book, Chapter 4 until 4.1 (inclusive)

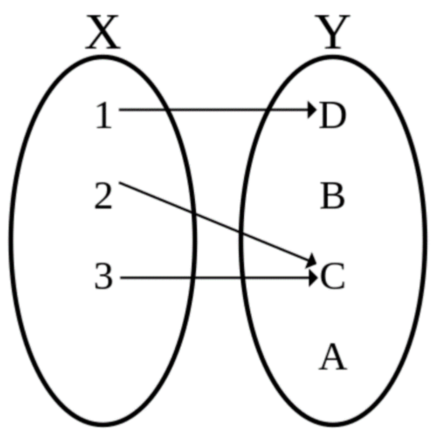
- Continue familiarizing yourself with OpenGL if you do not know it !

Data Representation

Mathematical Functions

Associates every element of a set (e.g., X) with *exactly one* element of another set (e.g., Y)

Maps from domain (X) to codomain (Y)



$$f: X \to Y$$
$$x \mapsto f(x)$$

Also important: *range/image*; *preimage*; continuity, differentiability, dimensionality, ...

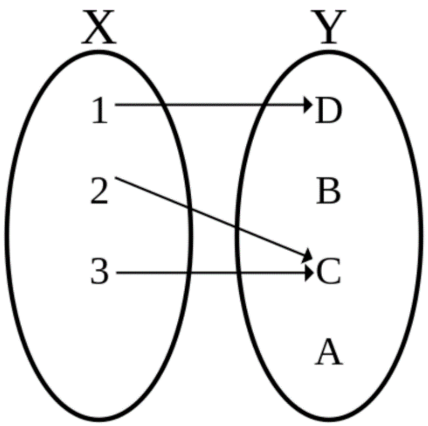
Graph of a function (mathematical definition):

$$G(f) := \{(x, f(x)) | x \in X\} \subset X \times Y$$

Mathematical Functions

Associates every element of a set (e.g., X) with *exactly one* element of another set (e.g., Y)

Maps from domain (X) to codomain (Y)



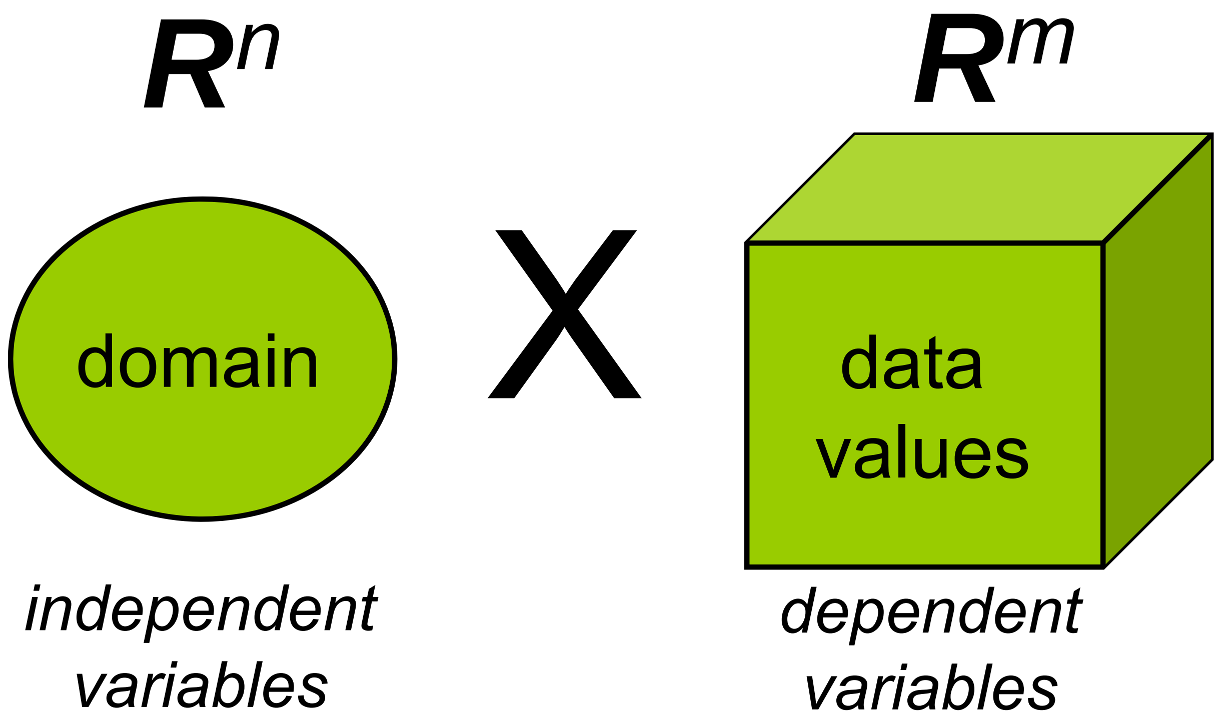
$$f : \mathbb{R}^n \to \mathbb{R}^m$$
$$x \mapsto f(x)$$

Also important: *range/image*; *preimage*; continuity, differentiability, dimensionality, ...

Graph of a function (mathematical definition):

$$G(f) := \{(x, f(x)) | x \in \mathbb{R}^n\} \subset \mathbb{R}^n \times \mathbb{R}^m \simeq \mathbb{R}^{n+m}$$

Data Representation



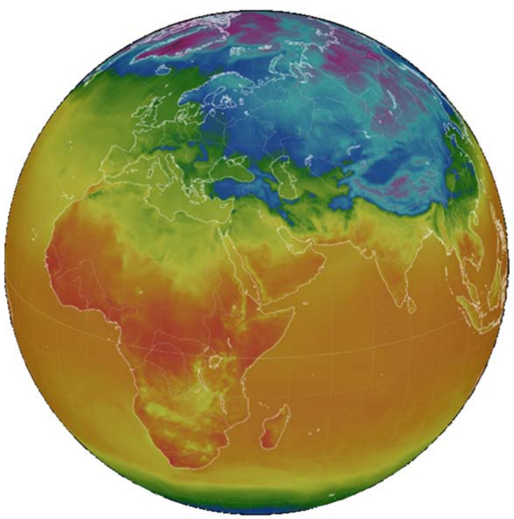
scientific data $\subseteq \boldsymbol{R}^{n+m}$

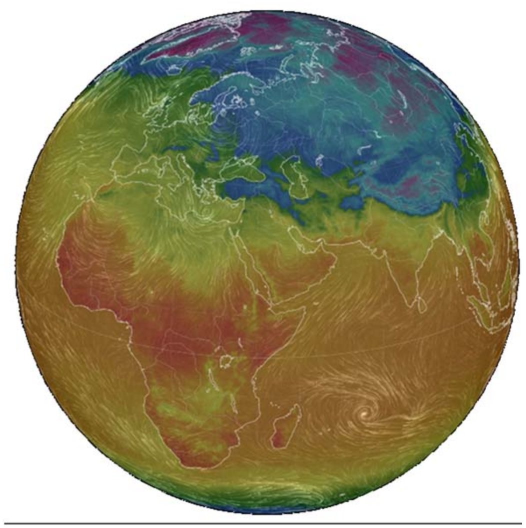
Domain Not Always Euclidean

Manifolds

...

- Scalar, vector, tensor fields on manifolds

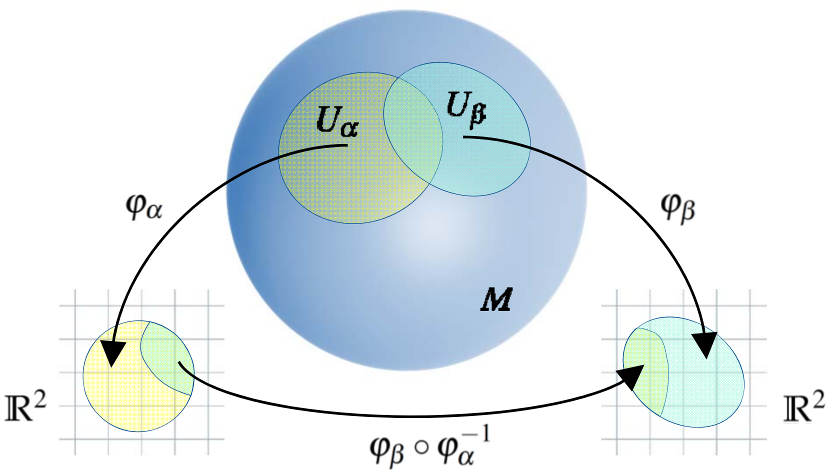


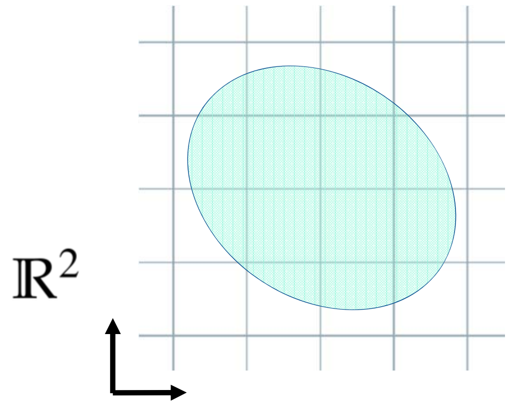


Topological Manifolds

Every point of an *n*-manifold is homeomorphic (topologically equivalent) to a region of $\mathbb{R}^n$

Think about being able to assign coordinates to a region: coordinate chart; (collection of charts: atlas)



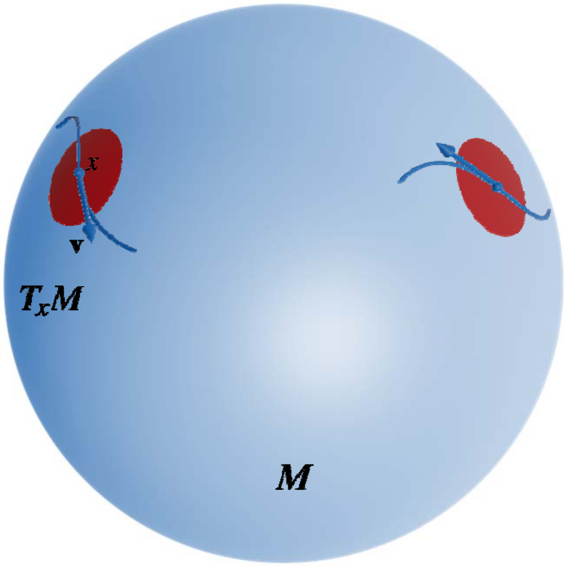


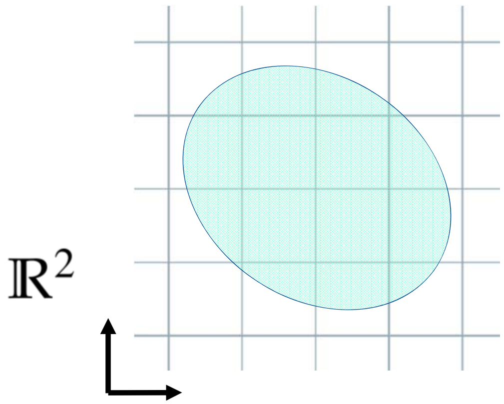
Smooth Manifolds

Well-defined tangent space at every point

- Dimensionality of each tangent space is the same as that of manifold

Enables calculus on manifolds (and vector fields, tensor fields, …)





Sampled Functions and Data Structures

Data Representation

- Discrete (sampled) representations
  - The objects we want to visualize are often ‘continuous’
  - But in most cases, the visualization data is given only at discrete locations in space and/or time
  - Discrete structures consist of samples, from which grids/meshes consisting of cells are generated
- Primitives in different dimensions

dimension	cell	mesh
0D	points	
1D	lines (edges)	polyline(–gon)
2D	triangles, quadrilaterals (rectangles)	2D mesh
3D	tetrahedra, prisms, hexahedra	3D mesh

Domain

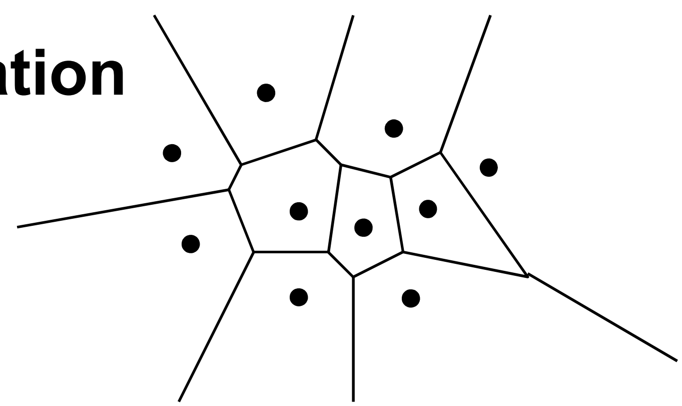
- The (geometric) shape of the domain is determined by the positions of sample points
- Domain is characterized by
  - Dimensionality: 0D, 1D, 2D, 3D, 4D, …
  - Influence: How does a data point influence its neighborhood?
  - Structure: Are data points connected? How? (Topology)

Domain

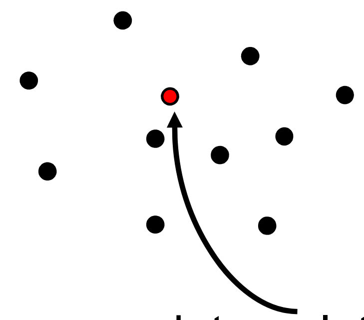
- Influence of data points
  - Values at sample points influence the data distribution in a certain region around these samples
  - To reconstruct the data at arbitrary points within the domain, the distribution of all samples has to be calculated
- Point influence
  - Only influence on point itself
- Local influence
  - Only within a certain region
    - Voronoi diagram
    - Cell-wise interpolation (see later in course)
- Global influence
  - Each sample might influence any other point within the domain
    - Material properties for whole object
    - Scattered data interpolation

Domain

- Voronoi diagram
  - Construct a region around each sample point that covers all points that are closer to that sample than to every other sample
  - Each point within a certain region gets assigned the value of the sample point
  - **Nearest-neighbor interpolation**



Domain



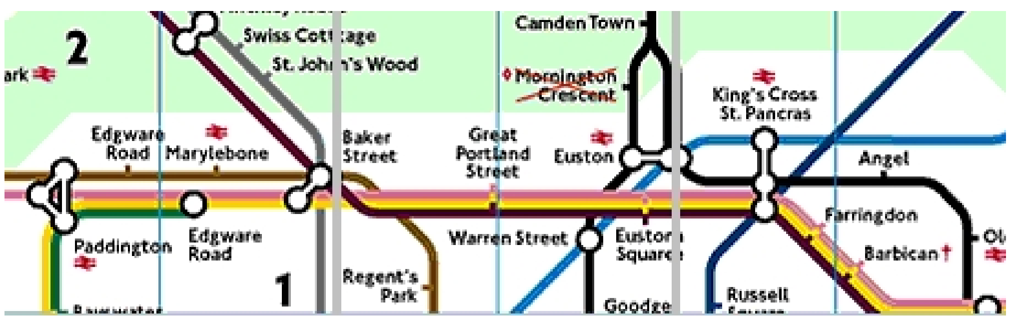
- Scattered data interpolation
  - At each point the weighted average of all sample points in the domain is computed
  - Weighting functions determine the support of each sample point
    - Radial basis functions simulate decreasing influence with increasing distance from samples
  - Schemes might be non-interpolating and expensive in terms of numerical operations

Data Structures

- Requirements:
  - Efficiency of accessing data
  - Space efficiency
  - Lossless vs. lossy
  - Portability
    - Binary – less portable, more space/time efficient
    - Text – human readable, portable, less space/time efficient
- Definition
  - If points are arbitrarily distributed and no connectivity exists between them, the data is called scattered
  - Otherwise, the data is composed of cells bounded by grid lines
  - **Topology** specifies the structure (**connectivity**) of the data
  - **Geometry** specifies the **position** of the data

Data Structures

- Some definitions concerning topology and geometry
  - In topology, qualitative questions about geometrical structures are the main concern
    - Does it have any holes in it?
    - Is it all connected together?
    - Can it be separated into parts?
- Underground map does not tell you how far one station is from the other, but rather how the lines are connected (topological map)



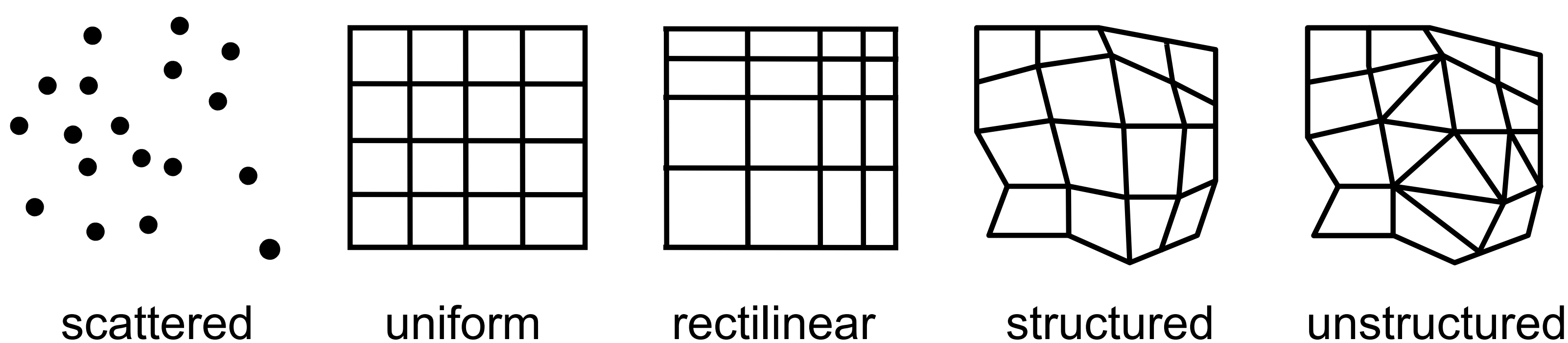
Grids – General Questions

Important questions:

- Which data organization is optimal?
- Where do the data come from?
- Is there a neighborhood relationship?
- How is the neighborhood info stored?
- How is navigation within the data possible?
- What calculations with the data are possible ?
- Are the data structured (regular/irregular topology)?

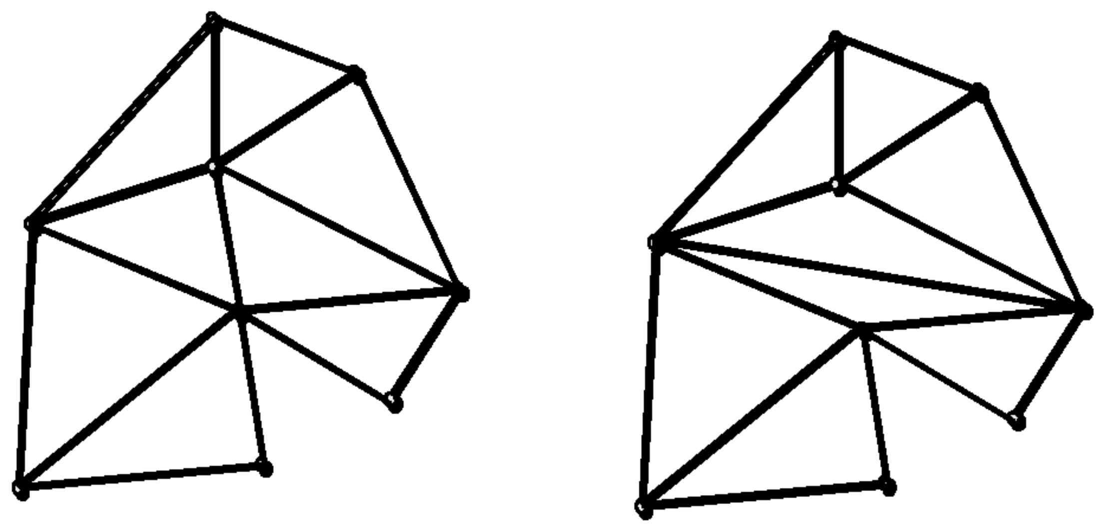
Data Structures

- Grid types
  - Grids differ substantially in the cells (basic building blocks) they are constructed from and in the way the topological information is given



Data Structures

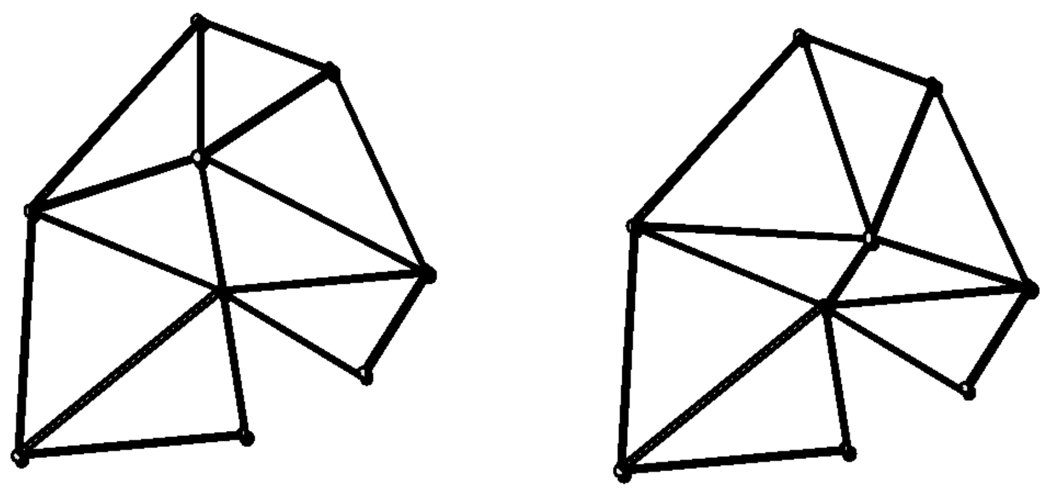
- Topology
  - Properties of geometric shapes that remain unchanged even when under distortion



Same geometry (vertex positions), different topology (connectivity)

Data Structures

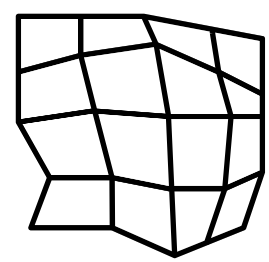
- Topologically equivalent
  - Things that can be transformed into each other by stretching and squeezing, without tearing or sticking together bits which were previously separated



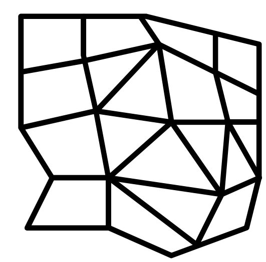
topologically equivalent

Data Structures

- Structured and unstructured grids can be distinguished by the way the elements or cells meet
- Structured grids
  - Have a regular topology and regular / irregular geometry
- Unstructured grids
  - Have irregular topology and geometry



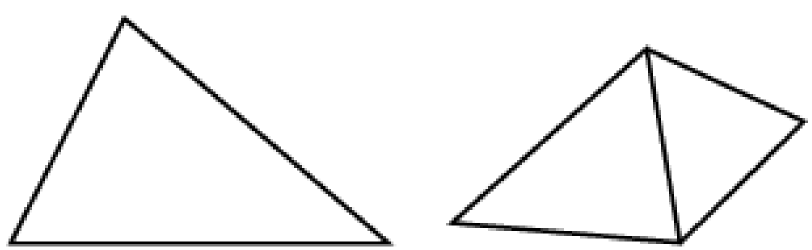
structured



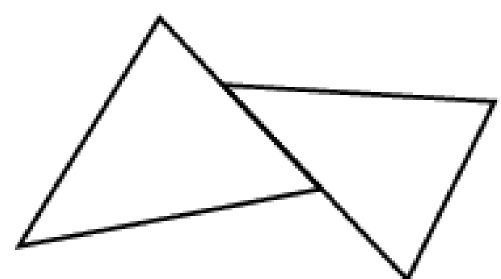
unstructured

Data Structures

- An *n*-simplex
  - The convex hull of $n + 1$ affinely independent points
  - Lives in R^m, with $n \leq m$
  - 0: points, 1: lines, 2: triangles, 3: tetrahedra
- Partitions via simplices are called triangulations
- Simplical complex *C* is a collection of simplices with:
  - Every face of an element of *C* is also in *C*
  - The intersection of two elements of *C* is empty or it is a face of both elements
- Simplical complex is a space with a triangulation



Simplical complexes

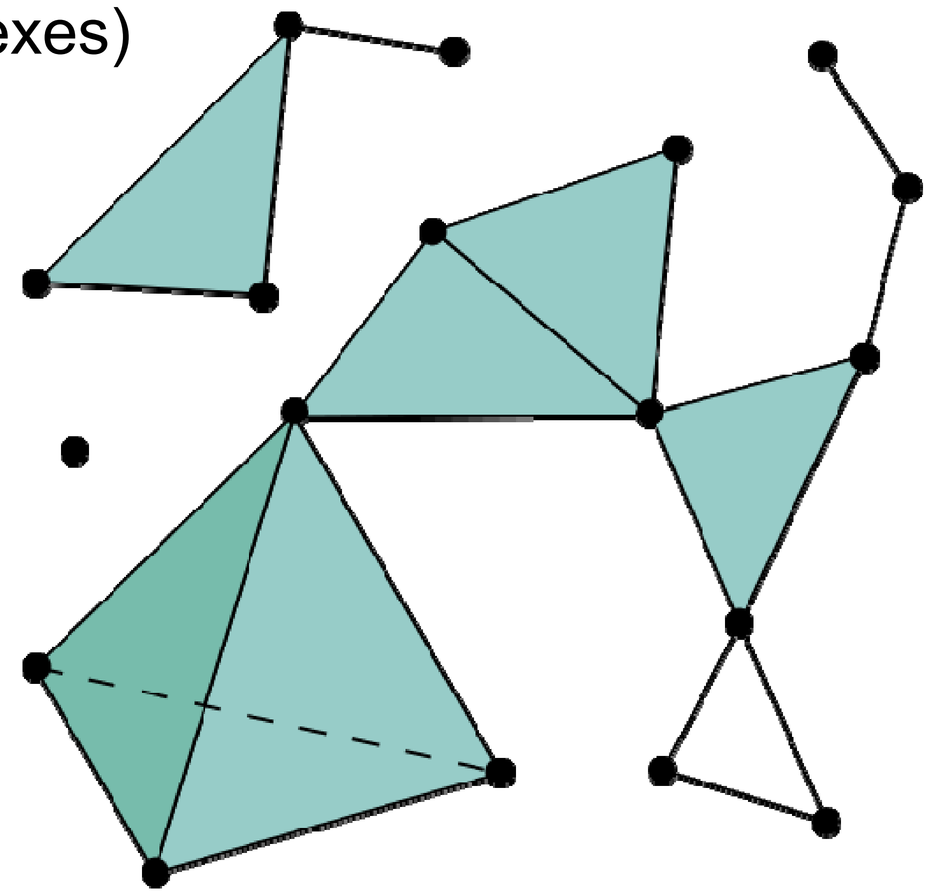


Not a simplical complex

Data Structures

- Simplicial complexes can be of mixed dimensions up to ≤ n
(except if "pure" complexes)

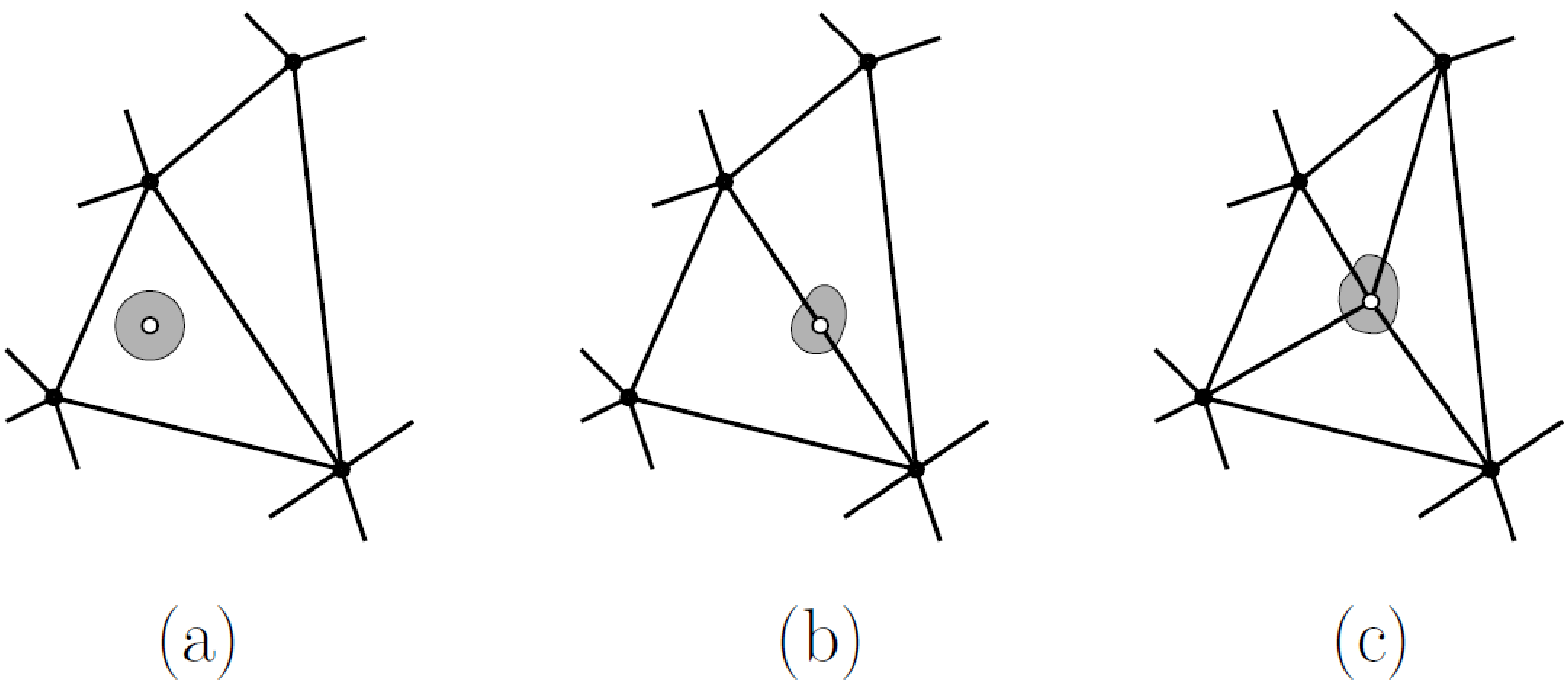
- Example:
Simplicial
3-complex



[Wikipedia.org]

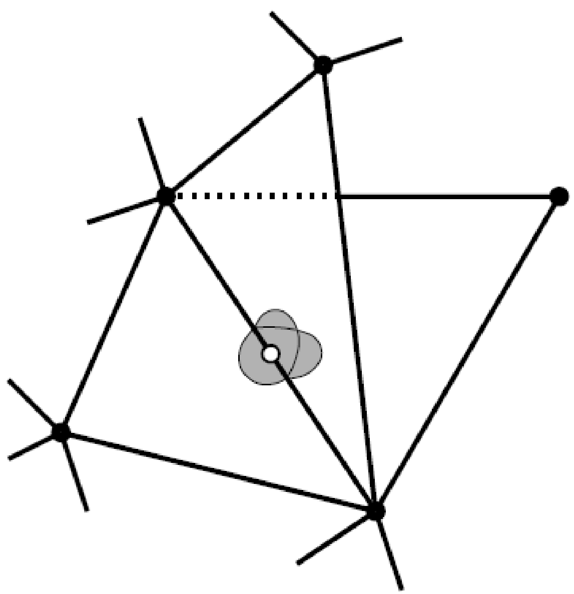
Data Structures

- 2-manifold meshes: neighborhood is 2-dimensional topological disc (or half disc for manifolds with boundary)

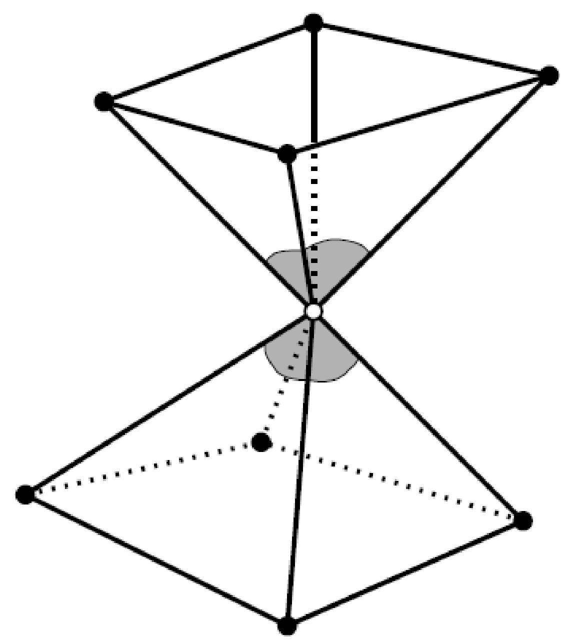


Data Structures

- Non-manifold meshes

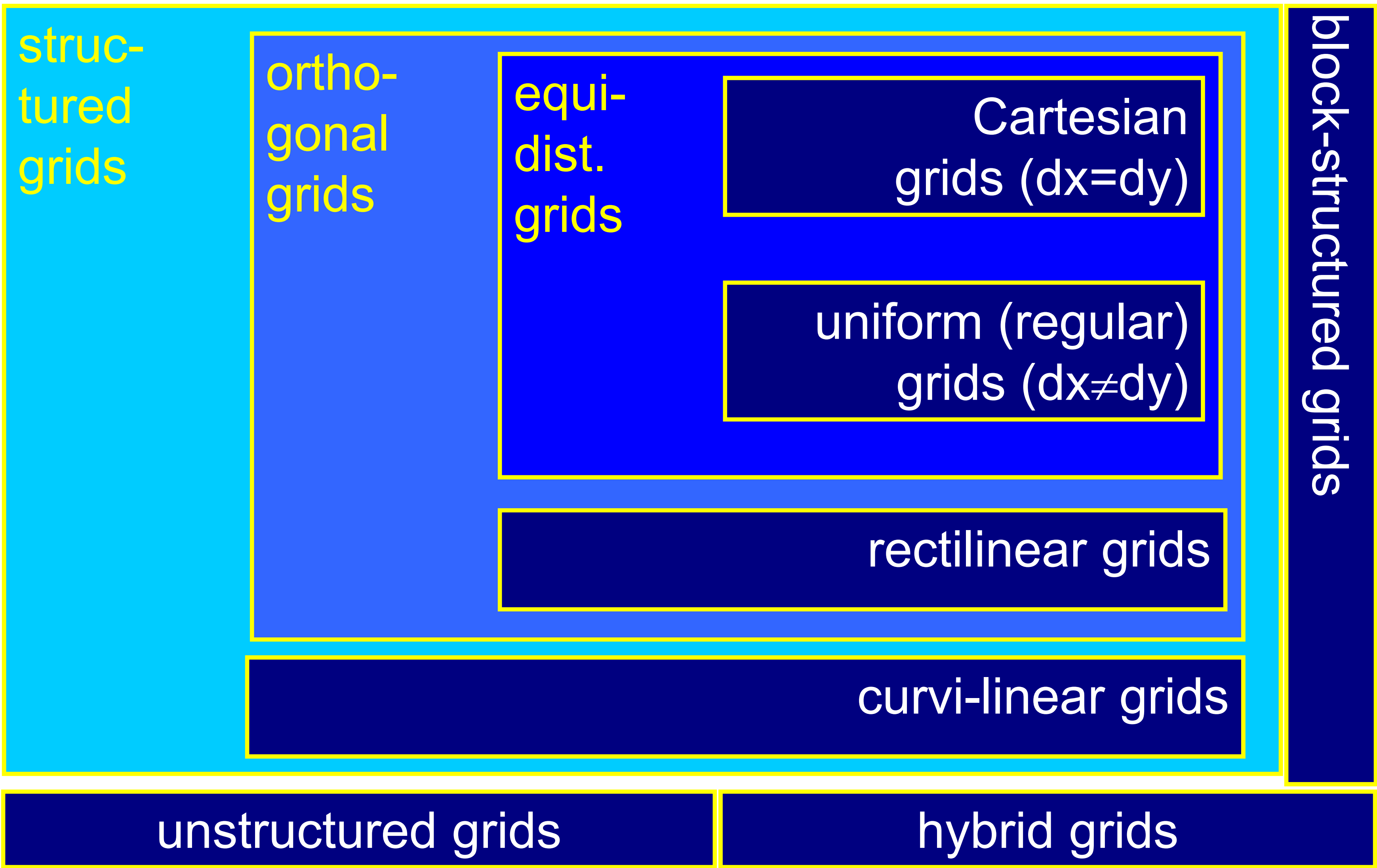


(d)



(e)

Grid Types - Overview



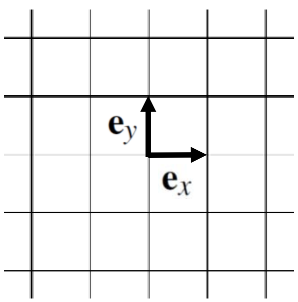
Naming / Definition Caveats

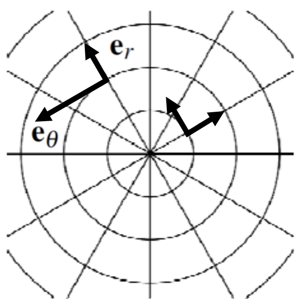
Beware of different naming conventions / different definitions

Example:

- On the previous slide, we used the term “orthogonal grid” in a simple, “global” way for the entire grid, i.e., different types of rectilinear grids, …
- In differential geometry, an orthogonal coordinate system is defined pointwise, i.e., a curvilinear grid with orthogonal basis vectors at each point is orthogonal

In differential geometry, both of these are orthogonal (in our context, the right one is not):

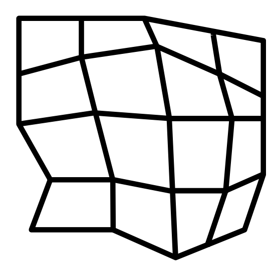


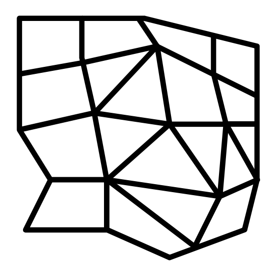


Structured Grids

Data Structures

- Characteristics of structured grids
  - Easier to compute with
  - Often composed of sets of connected parallelograms (hexahedra), with cells being equal or distorted with respect to (non-linear) transformations
  - May require more elements or badly shaped elements in order to precisely cover the underlying domain
  - Topology is represented implicitly by an *n*-vector of dimensions
  - Geometry is represented explicitly by an array of points
  - Every interior point has the same number of neighbors


structured


unstructured

Data Structures

- Characteristics of structured grids
  - Structured grids can be stored in a 2D / 3D array
  - Arbitrary samples can be directly accessed by indexing a particular entry in the array
  - Topological information is implicitly coded
    - Direct access to adjacent elements
  - Cartesian, uniform, and rectilinear grids are necessarily convex
  - Their visibility ordering of elements with respect to any viewing direction is given implicitly
  - Their rigid layout prohibits the geometric structure to adapt to local features
  - Curvilinear grids reveal a much more flexible alternative to model arbitrarily shaped objects
  - However, this flexibility in the design of the geometric shape makes the sorting of grid elements a more complex procedure

Data Structures

- Typical implementation of structured grids

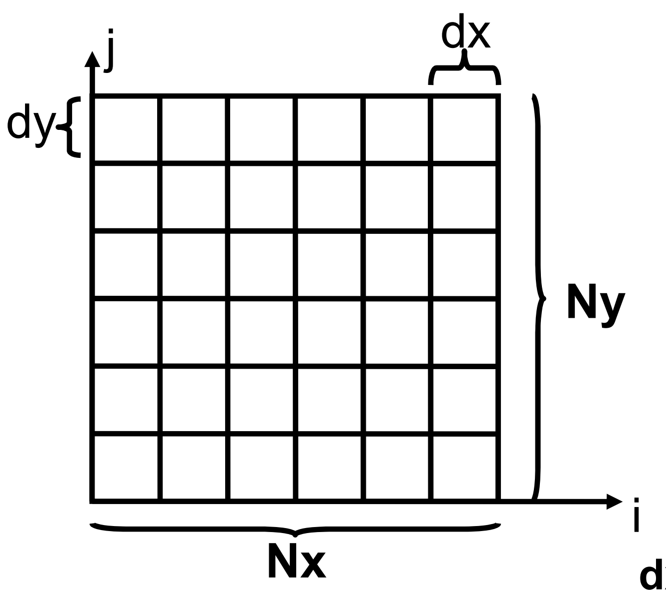
```
DataType *data = new DataType [Nx * Ny * Nz ];
val = data[ i + j * Nx + k * ( Nx * Ny ) ];
```

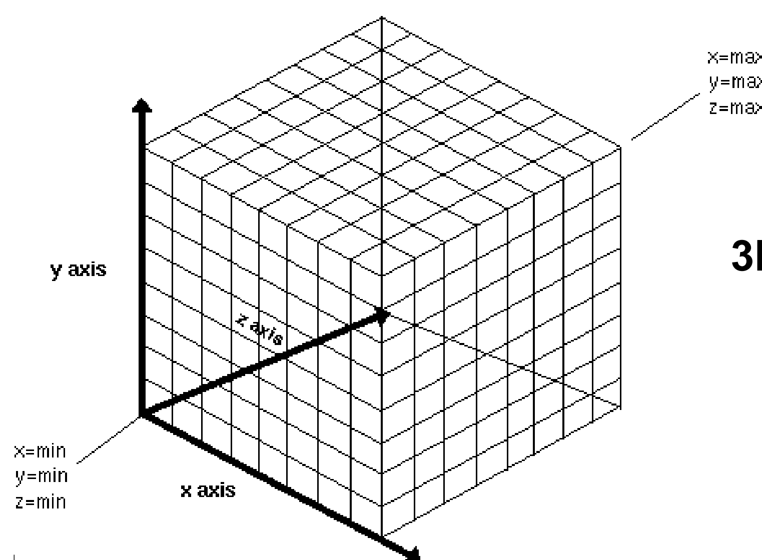
… code for geometry …

Data Structures

- Cartesian or equidistant grids
  - Structured grid
  - Cells and points are numbered sequentially with respect to increasing X, then Y, then Z, or vice versa
  - Number of points = Nx•Ny•Nz
  - Number of cells = (Nx-1)•(Ny-1)•(Nz-1)

2D





3D

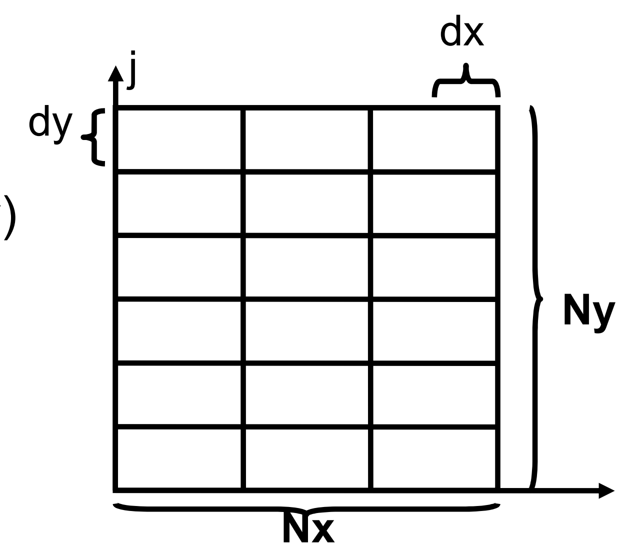
dx = dy = dz

Data Structures

- Cartesian grids
  - Vertex positions are given implicitly from [i,j,k]:
    - P[i,j,k].x = origin_x + i • dx
    - P[i,j,k].y = origin_y + j • dy
    - P[i,j,k].z = origin_z + k • dz
  - Global vertex index I[i,j,k] = k•Ny•Nx + j•Nx + i
    - k = I / (Ny•Nx)
    - j = (I % (Ny•Nx)) / Nx
    - i = (I % (Ny•Nx)) % Nx
  - Global index allows for linear storage scheme
    - Wrong access pattern might destroy cache coherence

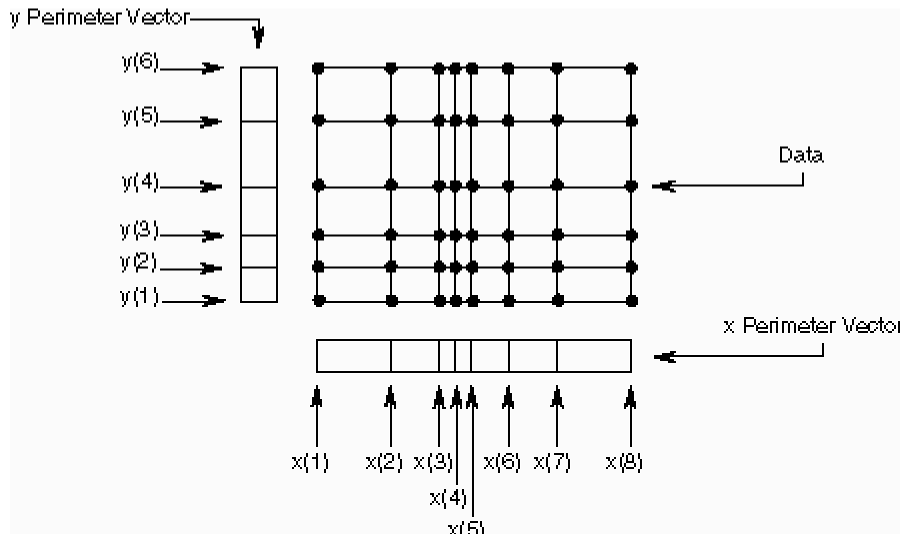
Data Structures

- Uniform grids
  - Similar to Cartesian grids
  - Consist of equal cells but with different resolution in at least one dimension ($dx \neq dy$ ($\neq dz$))
  - Spacing between grid points is constant in each dimension → same indexing scheme as for Cartesian grids
  - Most likely to occur in applications where the data is generated by a 3D imaging device providing different sampling rates in each dimension
  - Typical example: medical volume data consisting of slice images
    - Slice images with square pixels ($dx = dy$)
    - Larger slice distance ($dz > dx = dy$)



Data Structures

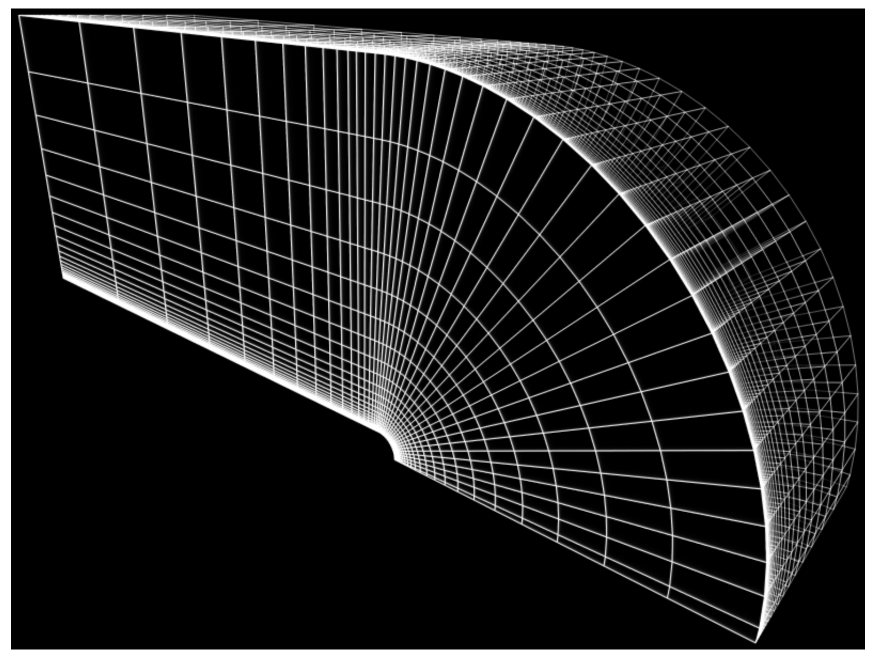
- Rectilinear grids
  - Topology is still regular but irregular spacing between grid points
    - Non-linear scaling of positions along either axis
    - Spacing, x_coord[L], y_coord[M], z_coord[N], must be stored explicitly
  - Topology is still implicit



(2D perimeter lattice:
rectilinear grid in IRIS Explorer)

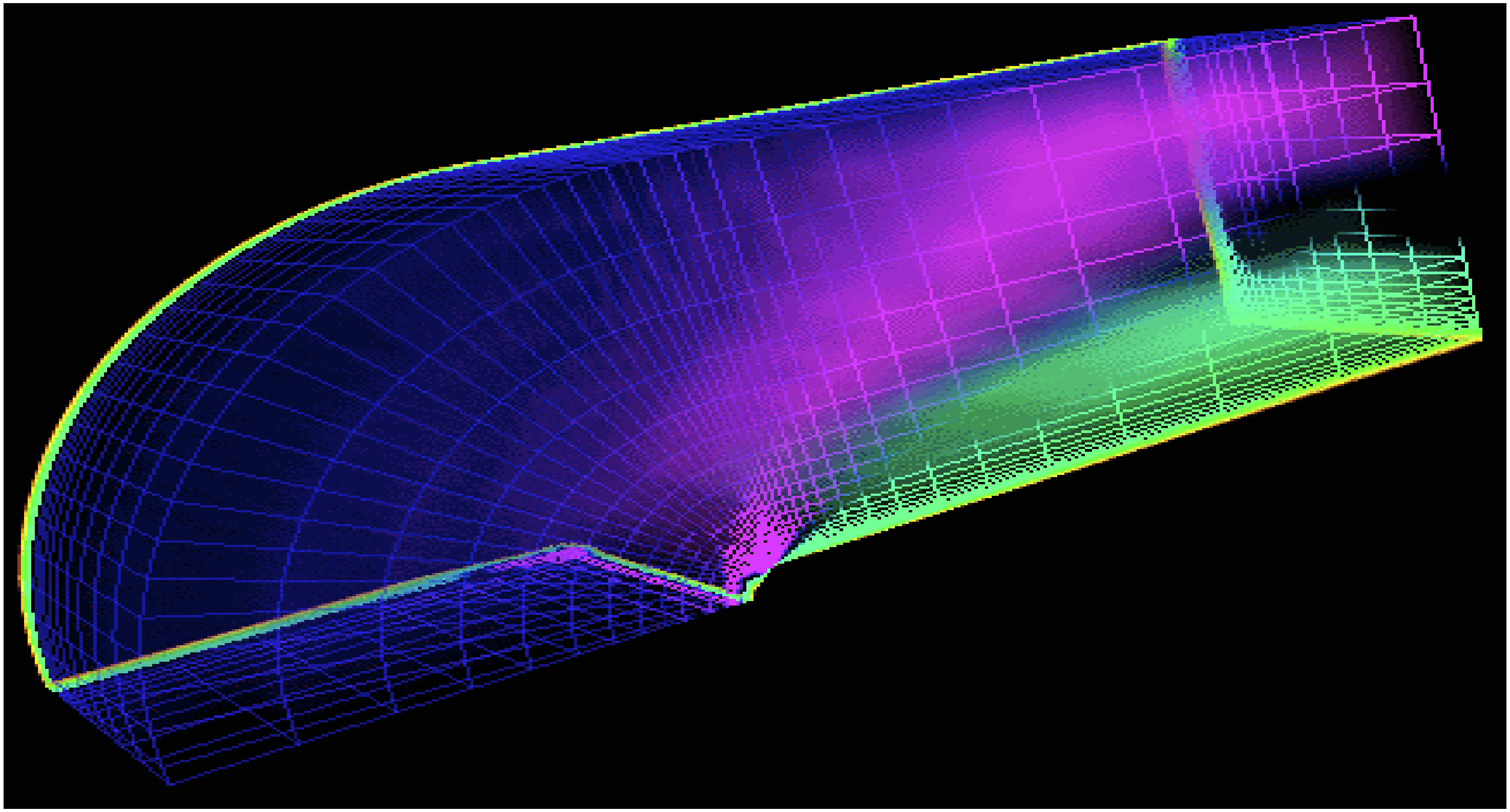
Data Structures

- Curvilinear grids
  - Topology is still regular but irregular spacing between grid points
    - Positions are non-linearly transformed
  - Topology is still implicit, but vertex positions are explicitly stored
    - x_coord[L,M,N]
    - y_coord[L,M,N]
    - z_coord[L,M,N]
  - Geometric structure might result in concave grids



Data Structures

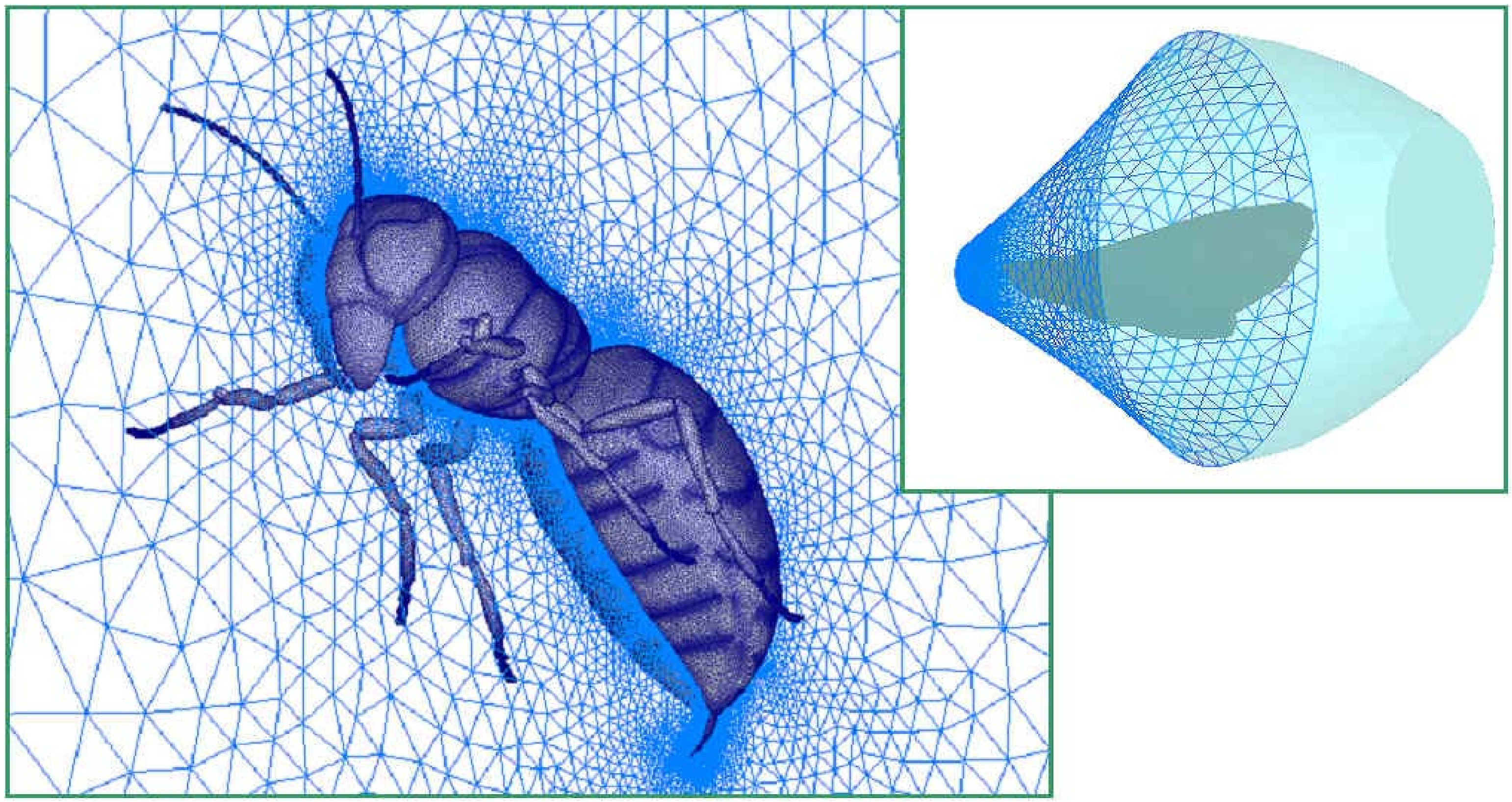
- Curvilinear grids



Unstructured Grids

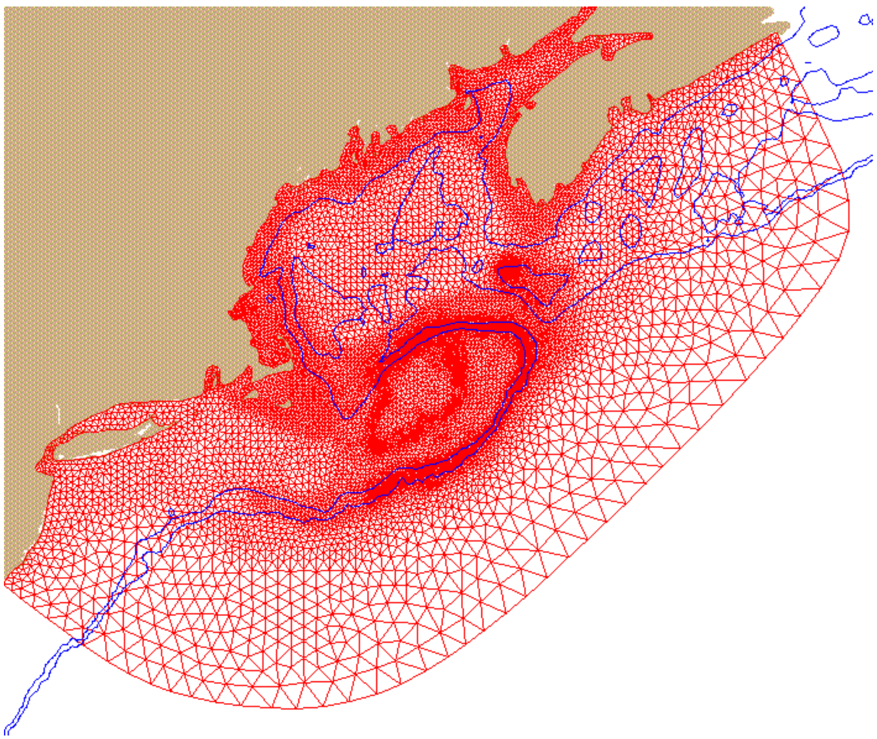
Data Structures

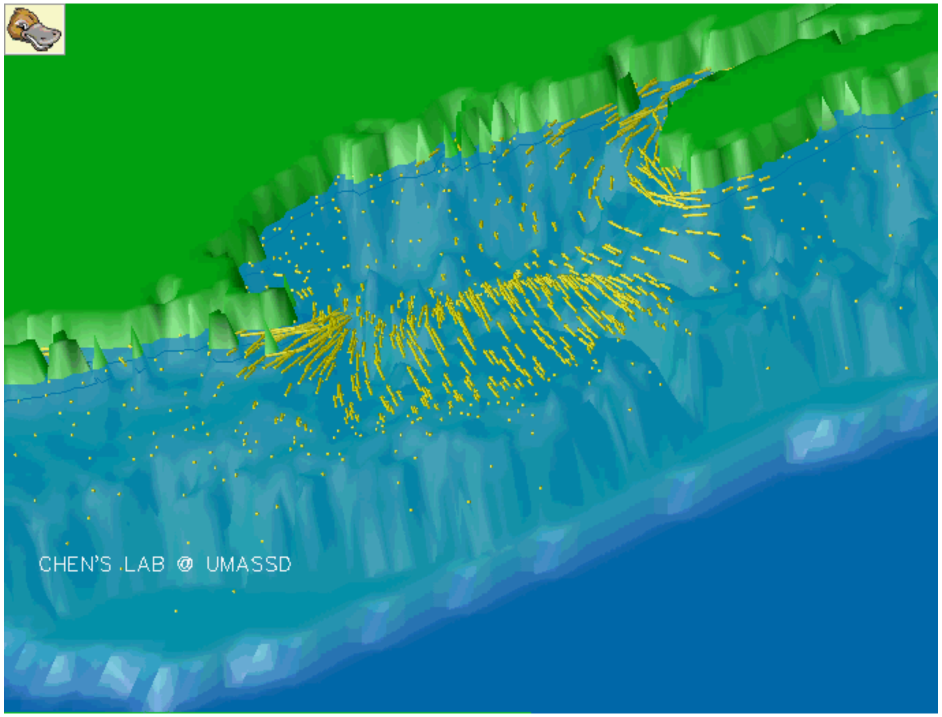
- Unstructured grids
  - Can be adapted to local features



Data Structures

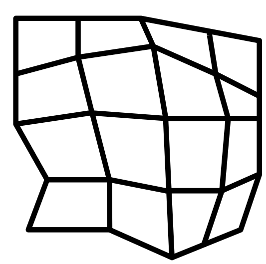
- Unstructured grids
  - Can be adapted to local features

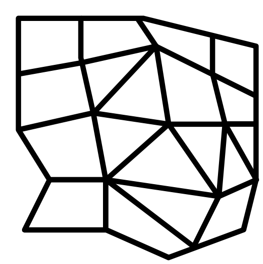




Data Structures

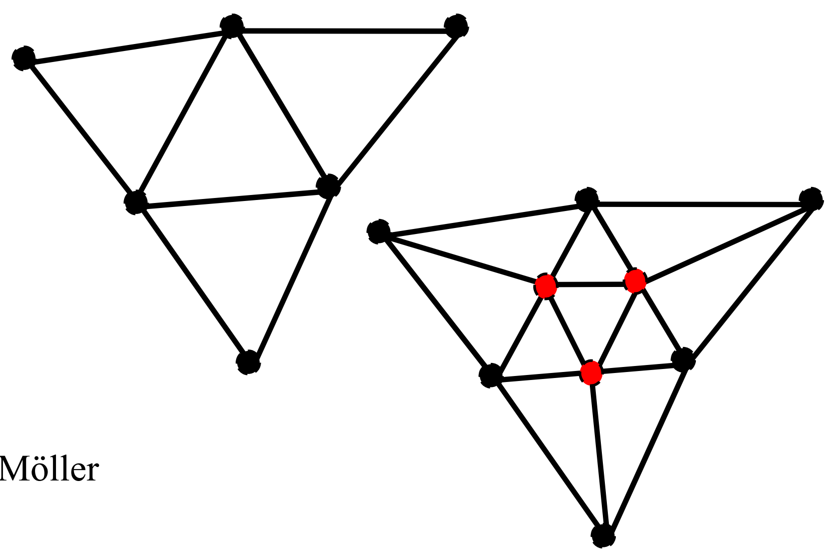
- If no implicit topological (connectivity) information is given, the grids are called unstructured grids
  - Unstructured grids are often computed using quadtrees (recursive domain partitioning for data clustering), or by triangulation of point sets
  - The task is often to create a grid from scattered points
- Characteristics of unstructured grids
  - Grid point geometry **and** connectivity must be stored
  - Dedicated data structures needed to allow for efficient traversal and thus data retrieval
  - Often composed of triangles or tetrahedra
  - Typically, fewer elements are needed to cover the domain


structured


unstructured

Data Structures

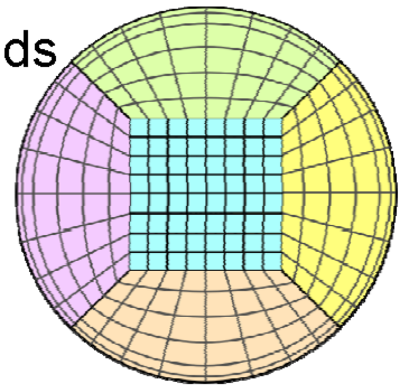
- Unstructured grids
  - Composed of arbitrarily positioned and connected elements
  - Can be composed of one unique element type or they can be hybrid (tetrahedra, hexas, prisms)
  - Triangle meshes in 2D and tetrahedral grids in 3D are most common
  - Can adapt to local features (small vs. large cells)
  - Can be refined adaptively
  - Simple linear interpolation in simplices



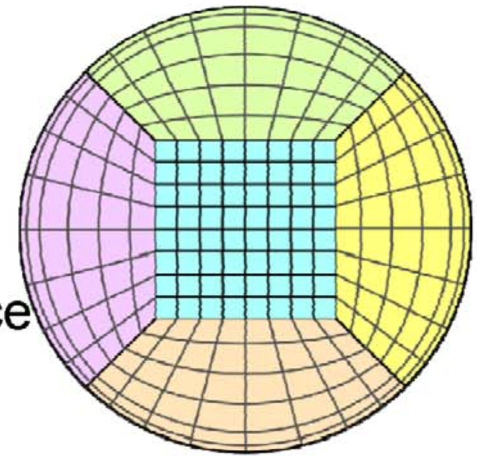
Data discretizations

Types of data sources have typical types of discretizations:

- Measurement data:
  - typically scattered (no grid)
- Numerical simulation data:
  - structured, block-structured, unstructured grids
  - adaptively refined meshes
  - multi-zone grids with relative motion
  - etc.
- Imaging methods:
  - uniform grids
- Mathematical functions:
  - uniform/adaptive sampling on demand

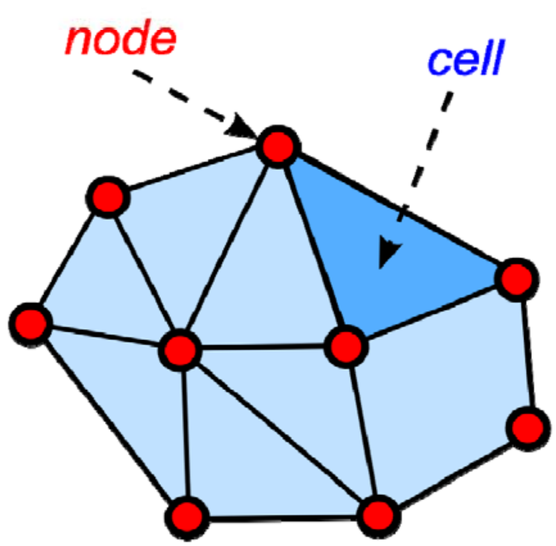


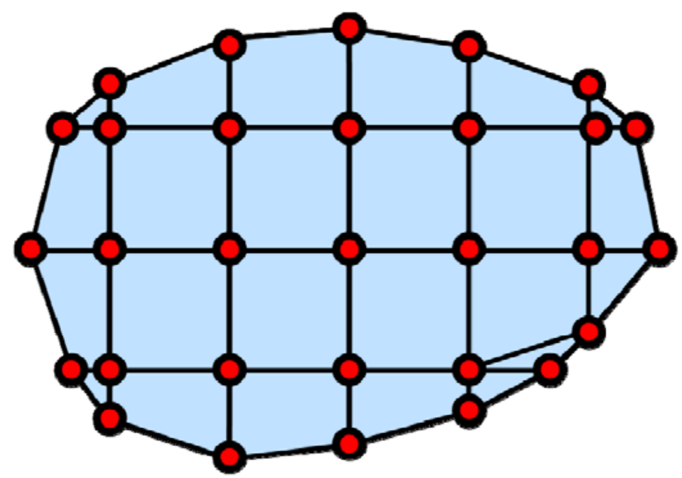
Unstructured grids



2D unstructured grids:

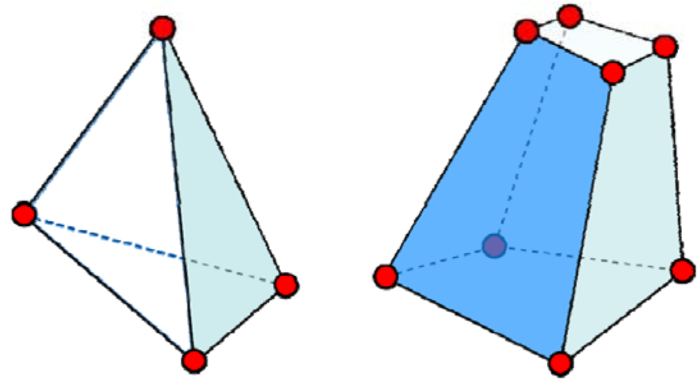
- cells are triangles and/or quadrangles
- domain can be a surface embedded in 3-space
  (distinguish n-dimensional from n-space)



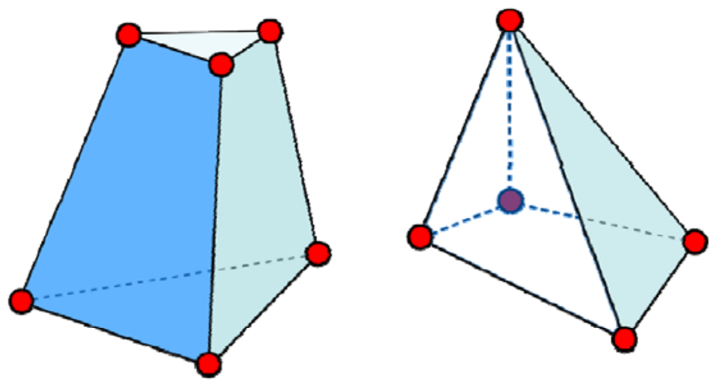


3D unstructured grids:

- cells are tetrahedra or hexahedra

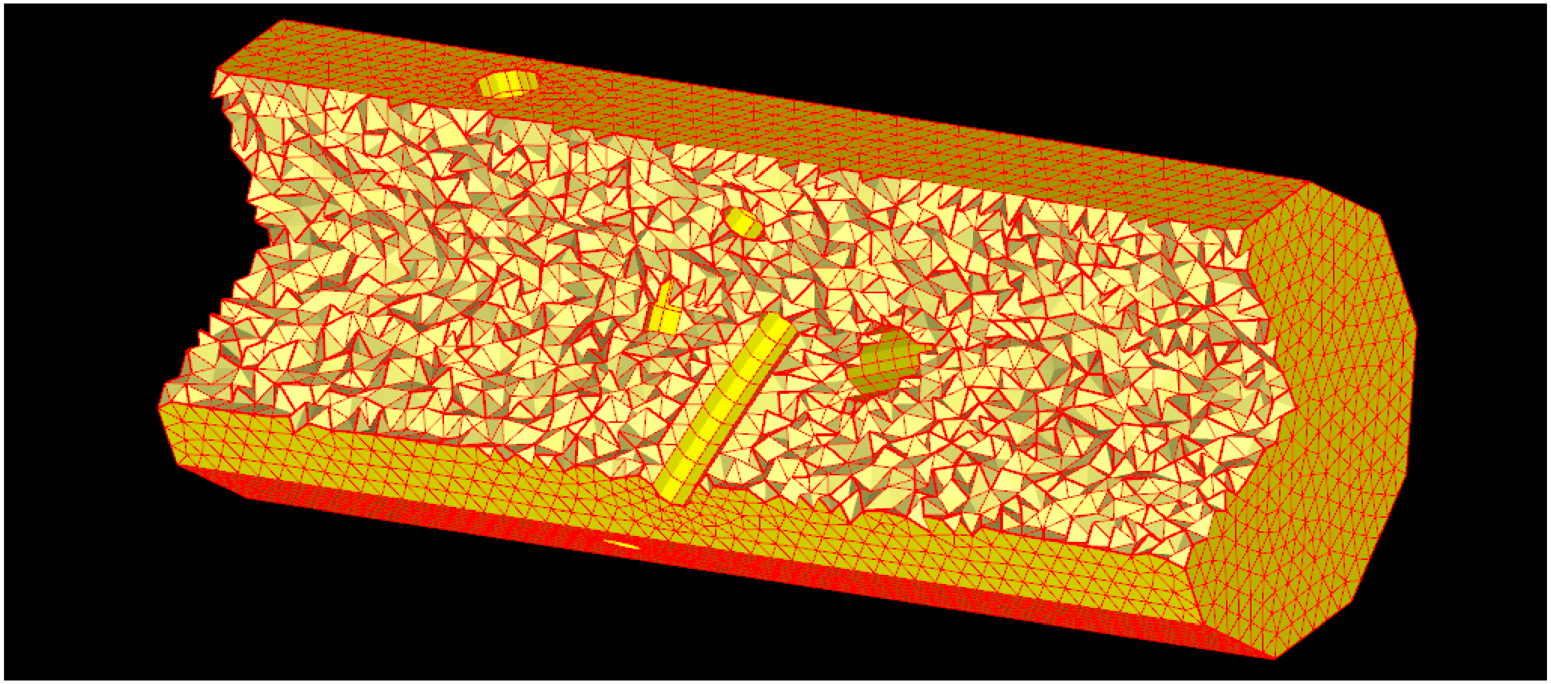


- mixed grids ("zoo meshes") require additional types: wedge (3-sided prism), and pyramid (4-sided)



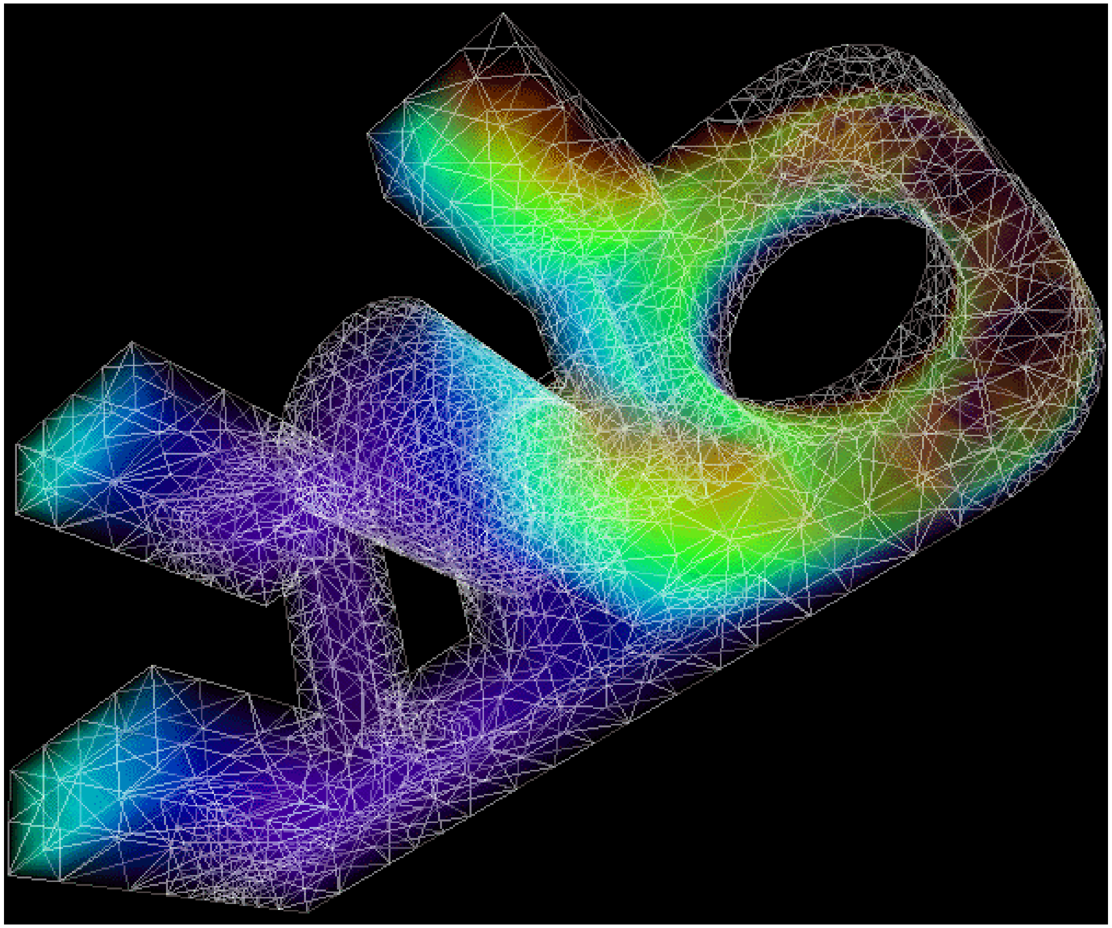
Common Unstructured Grid Types (1)

- Simplest: purely tetrahedral



Grid Structures

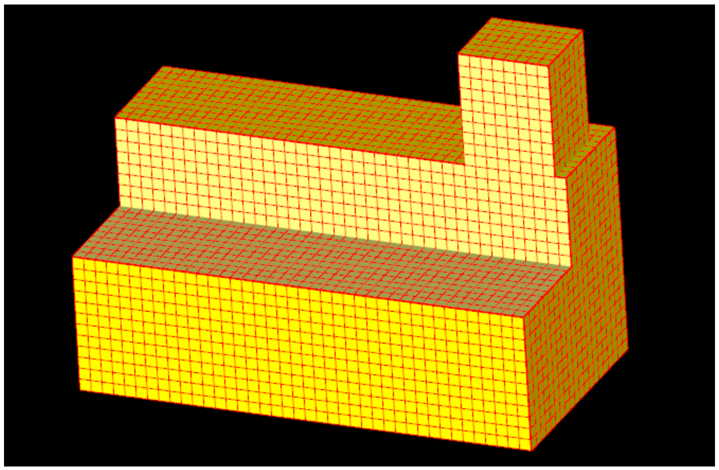
Tet grid example

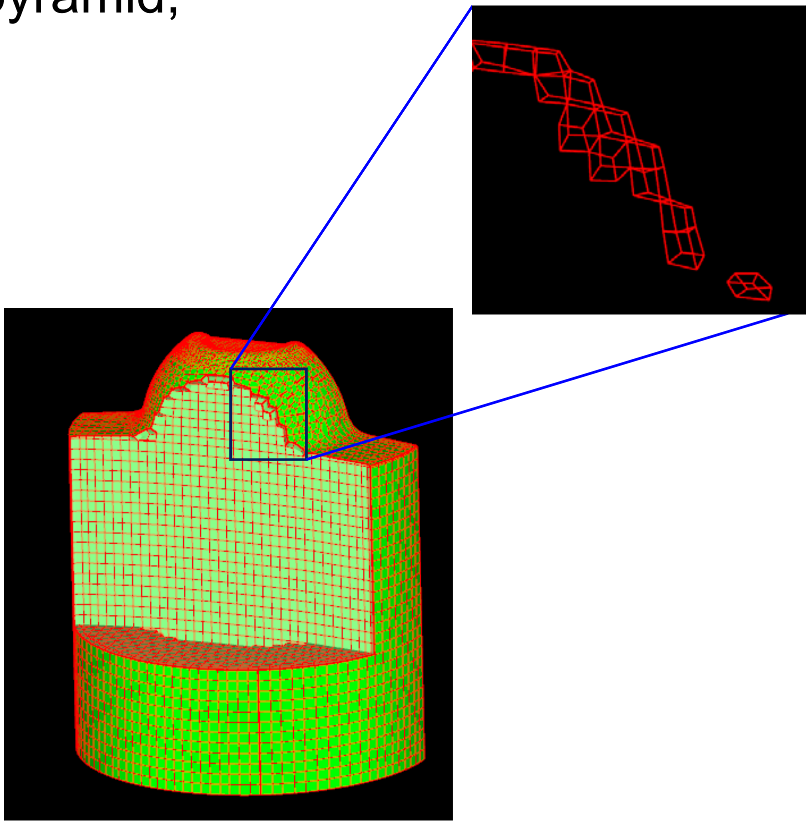


Common Unstructured Grid Types (2)

Pre-defined cell types
(tetrahedron, triangular prism, quad pyramid, hexahedron, octahedron)

- Only triangle / quad faces
- Planar / non-planar faces

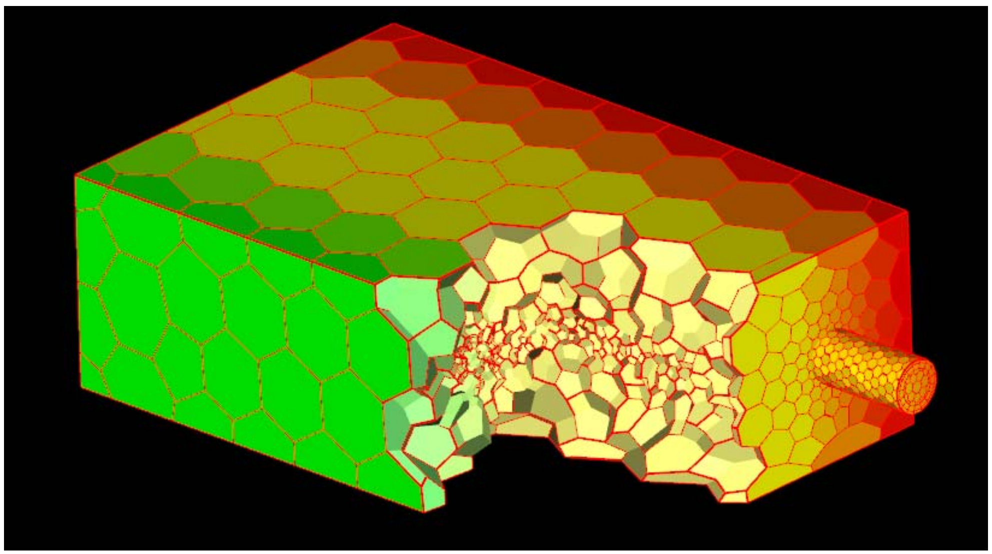


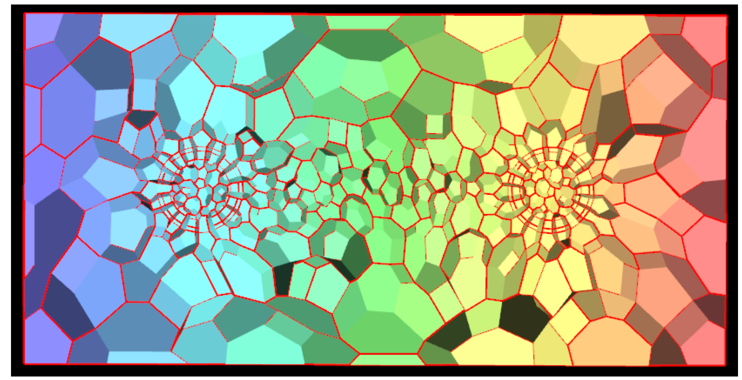


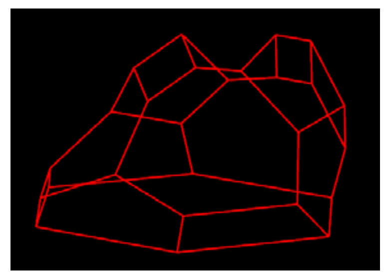
Common Unstructured Grid Types (3)

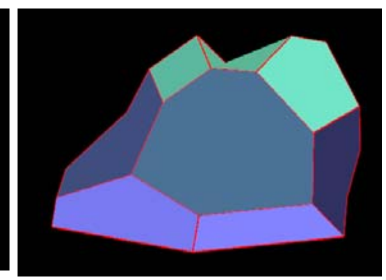
(Nearly) arbitrary polyhedra

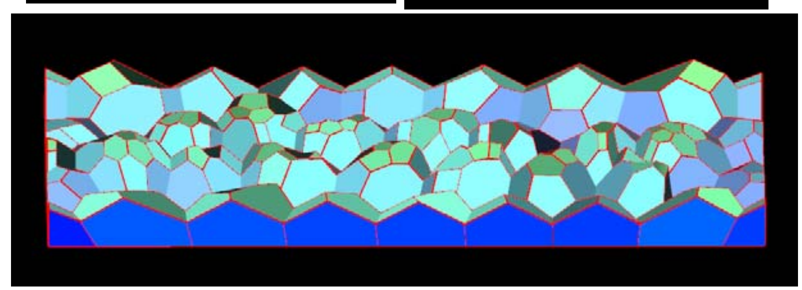
- Possibly non-planar faces







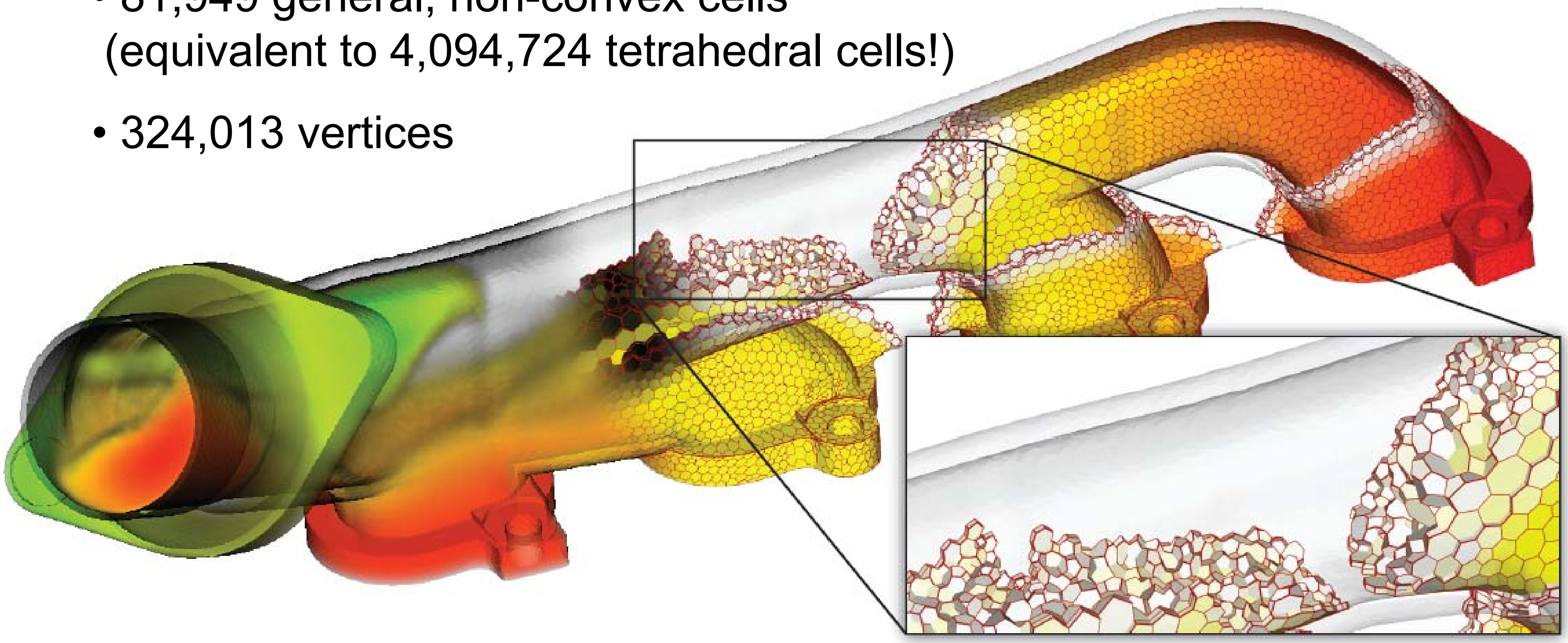




Example: General Polyhedral Cells

Exhaust manifold

- 81,949 general, non-convex cells (equivalent to 4,094,724 tetrahedral cells!)
- 324,013 vertices

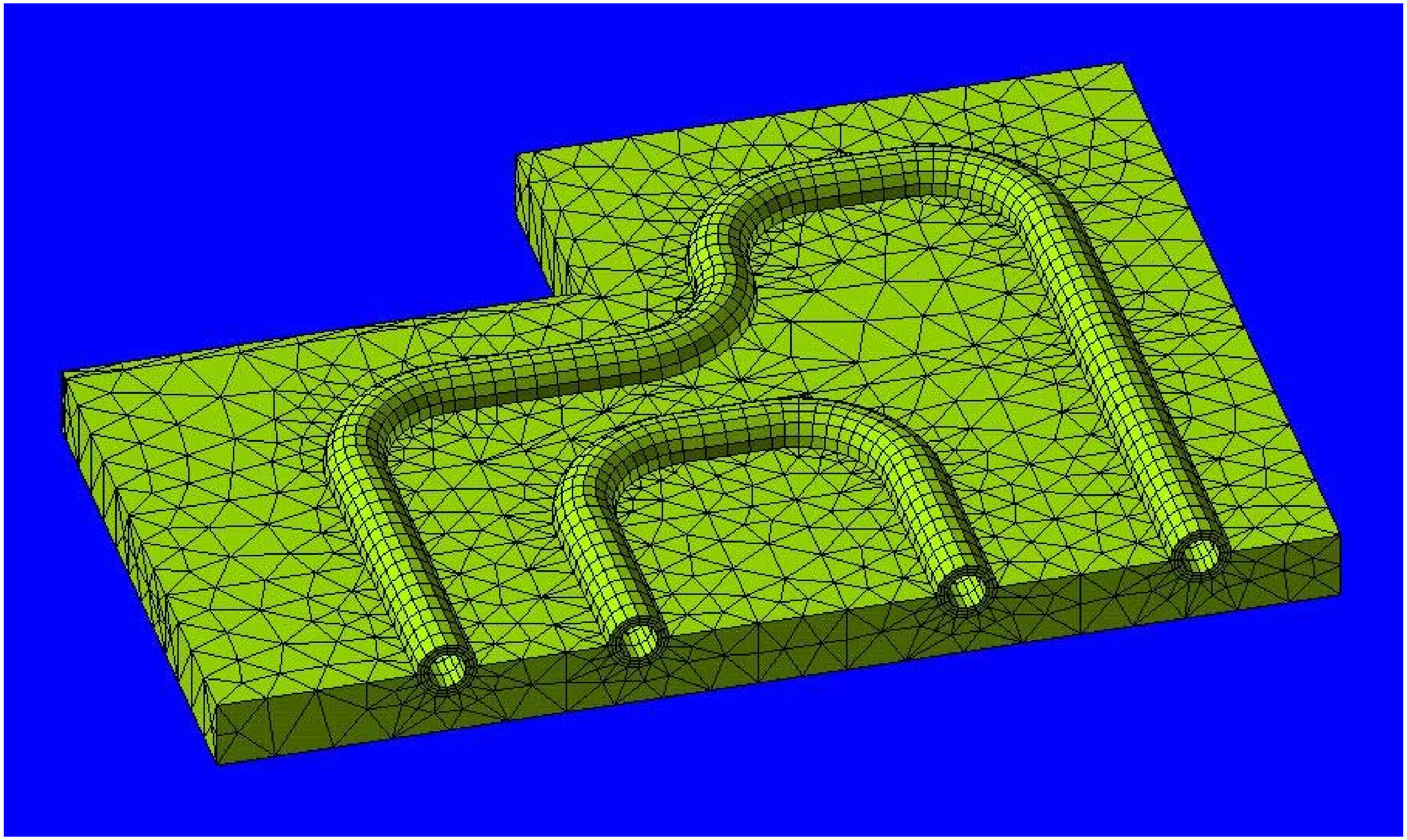


- Color coding: temperature distribution

Hybrid Grids

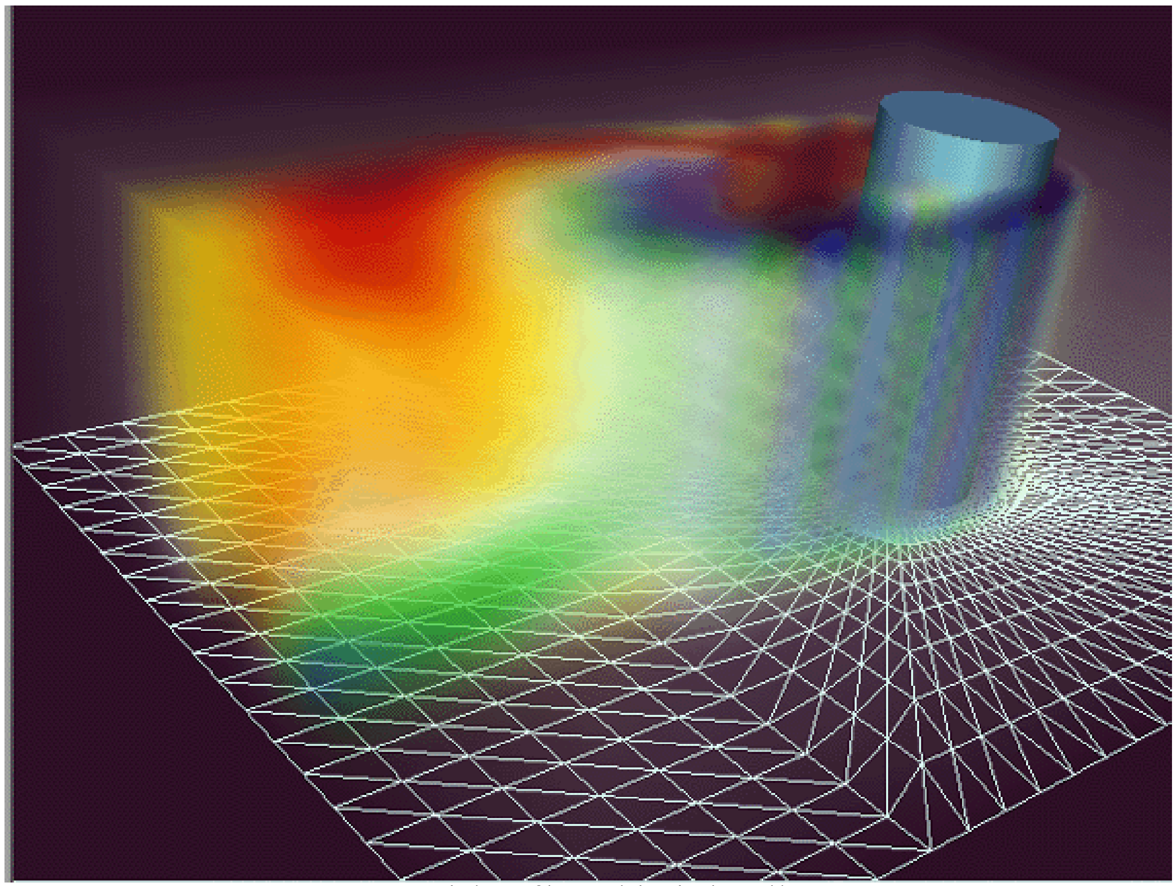
Data Structures

- Hybrid grids
  - Combination of different grid types

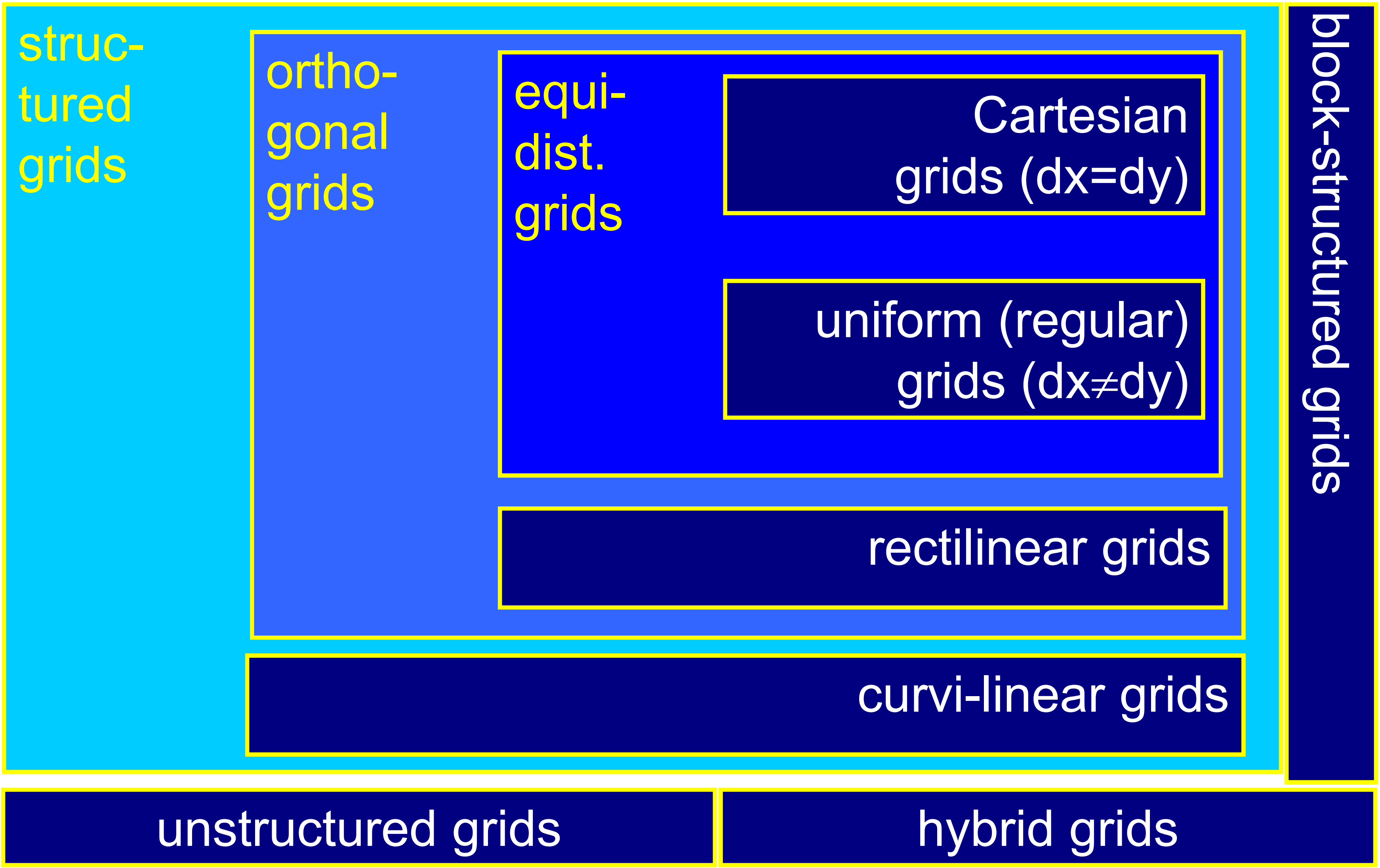


Data Structures

Hybrid grid example



Grid Types - Overview



Thank you.

Thanks for material

- Helwig Hauser
- Eduard Gröller
- Daniel Weiskopf
- Torsten Möller
- Ronny Peikert
- Philipp Muigg
- Christof Rezk-Salama