CS 247 – Scientific Visualization
Lecture 5: Data Representation, Pt. 3
Scalar Fields, Pt. 1

Markus Hadwiger, KAUST

King Abdullah University of Science and Technology KAUST

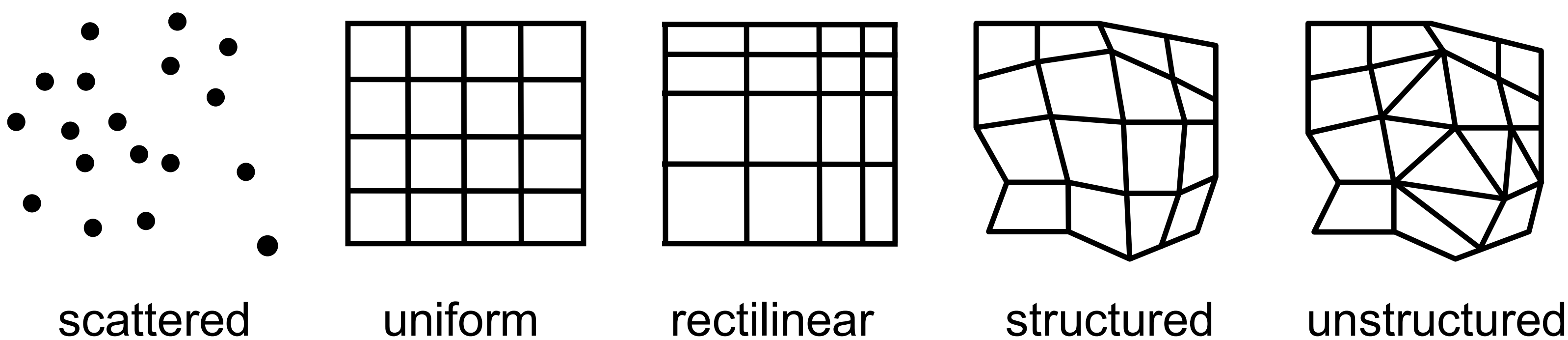
Reading Assignment #3 (until Feb 15)

Read (required):

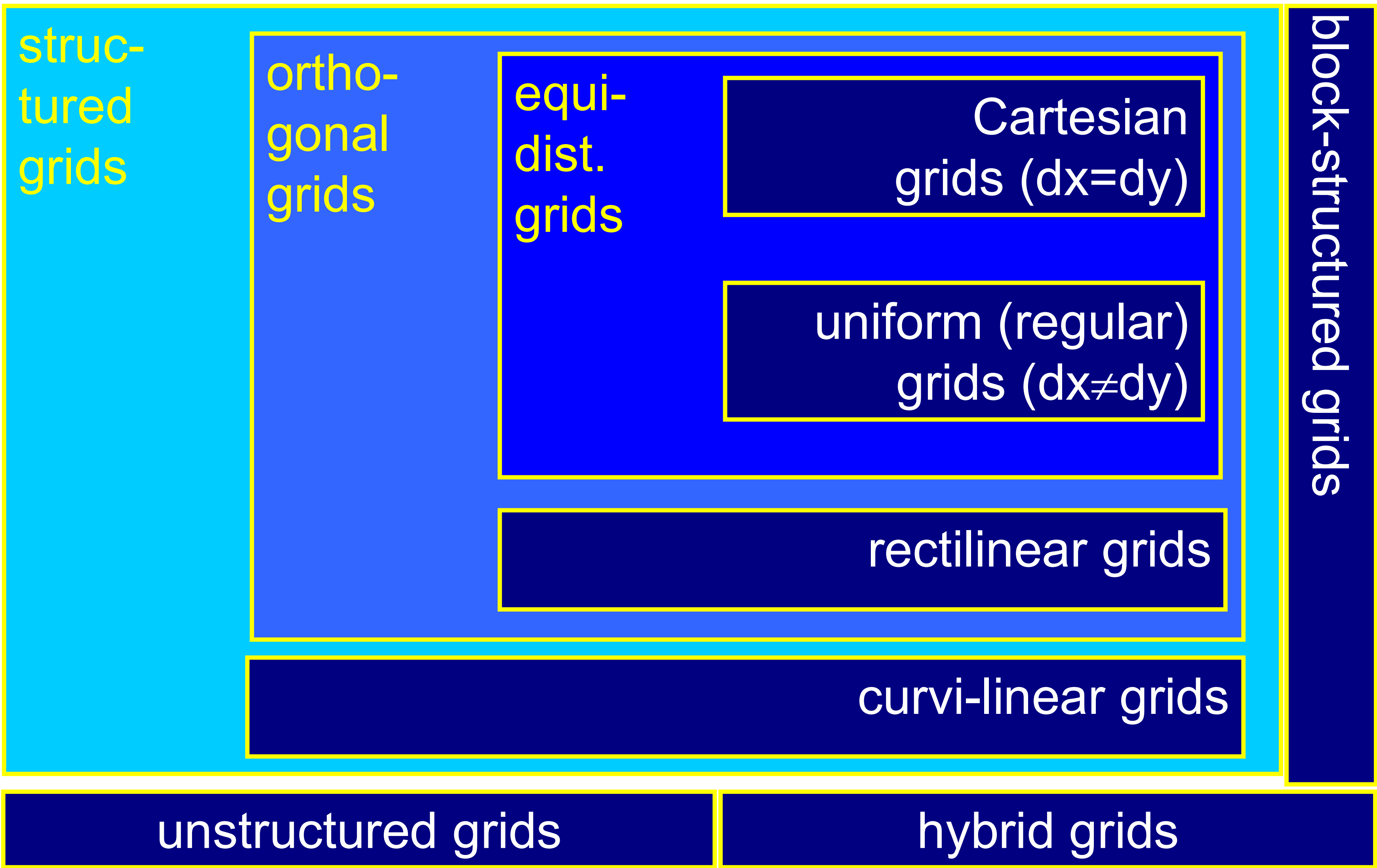
- Data Visualization book, finish Chapter 3 (read starting with 3.6)
- Data Visualization book, Chapter 5 until 5.3 (inclusive)

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



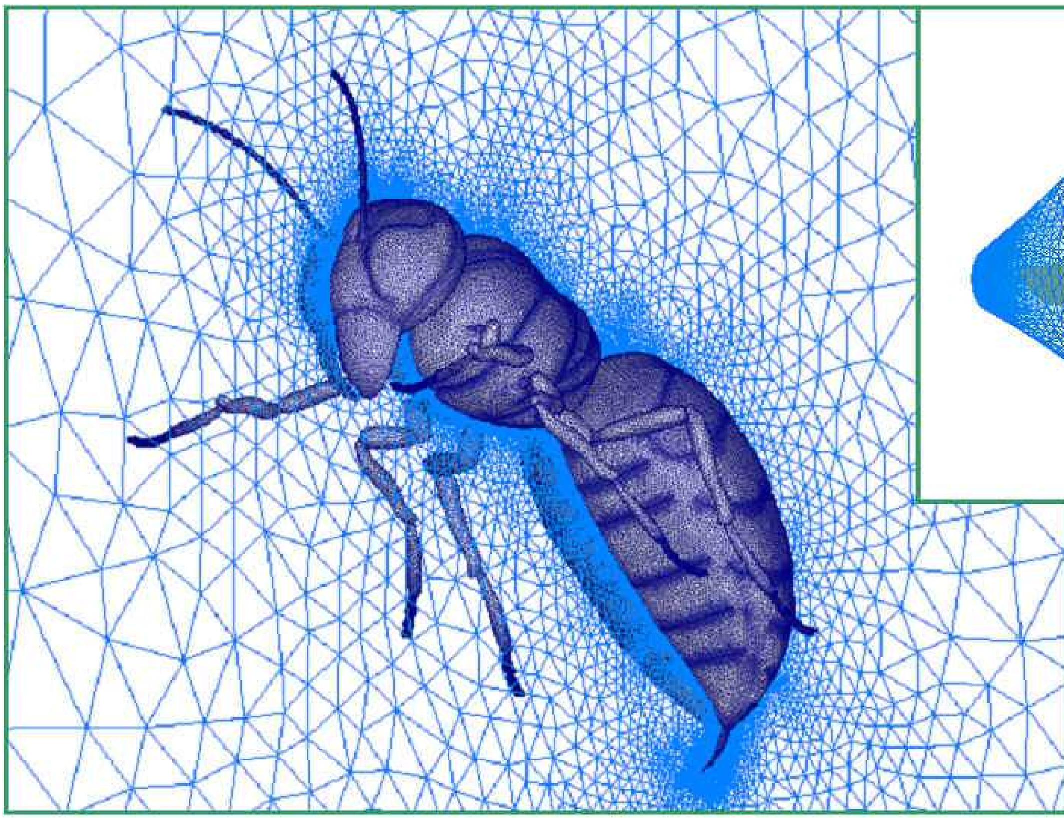
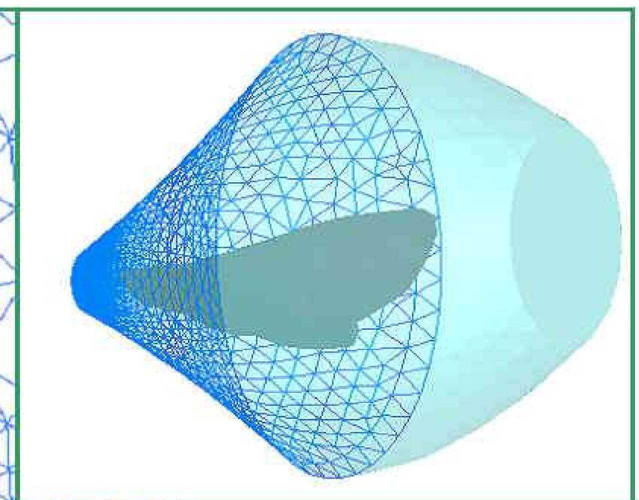
Grid Types - Overview



Unstructured Grids

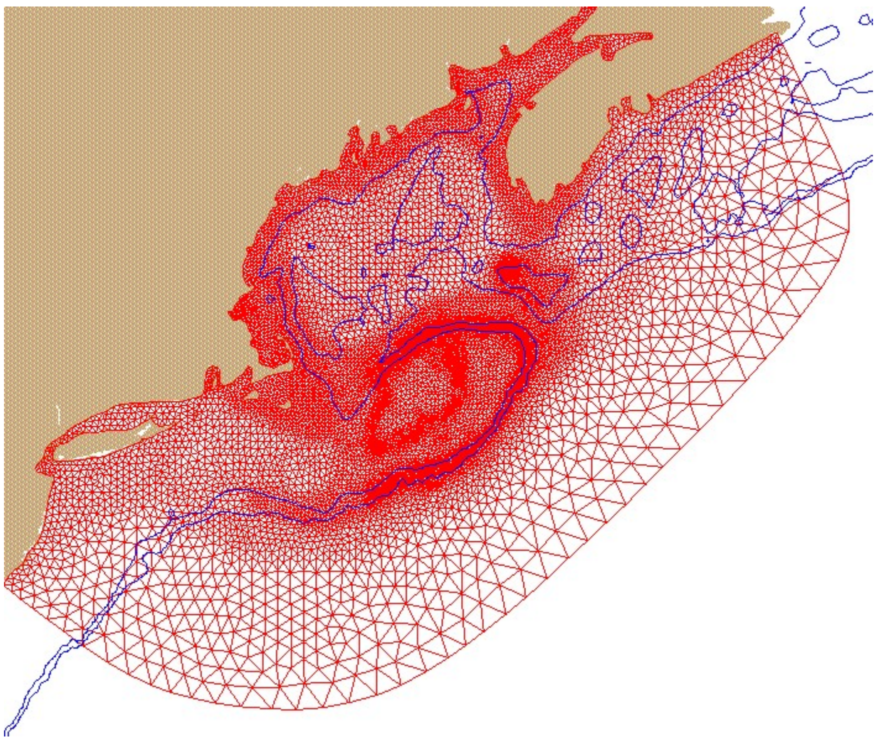
Data Structures

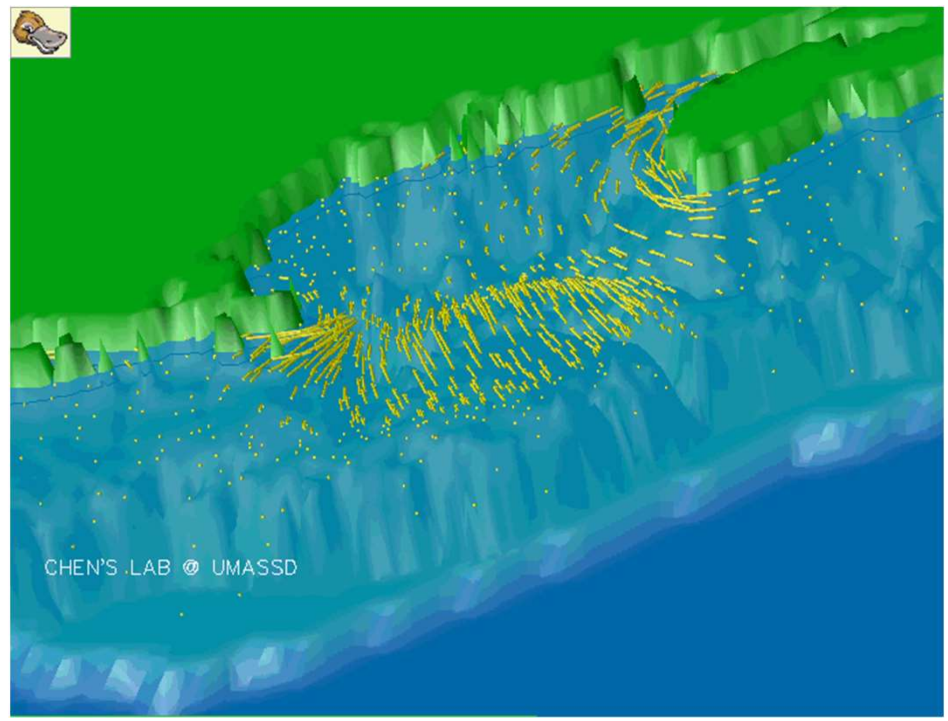
- Unstructured grids
  - Can be adapted to local features

Data Structures

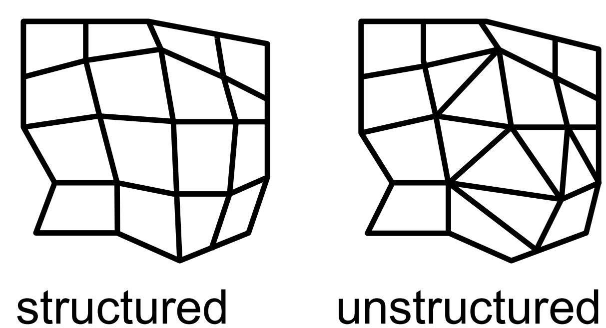
- Unstructured grids
  - Can be adapted to local features





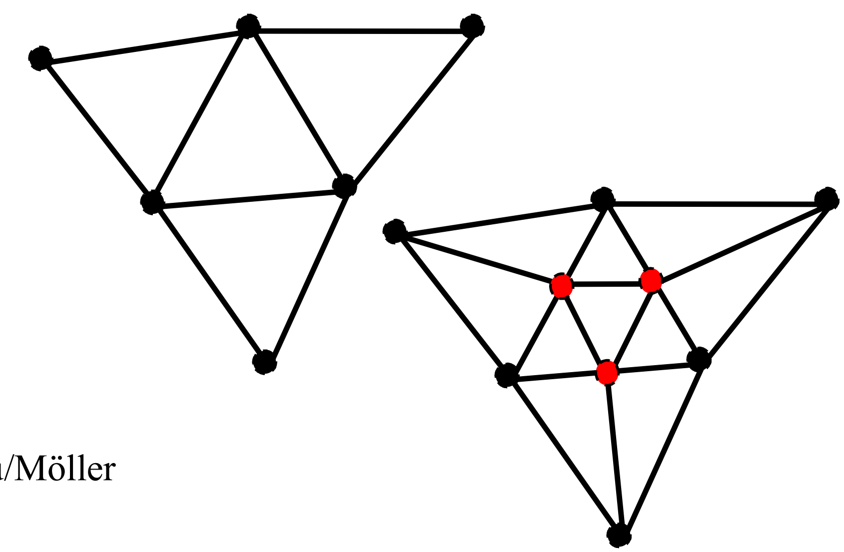
Data Structures

- 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



Data Structures

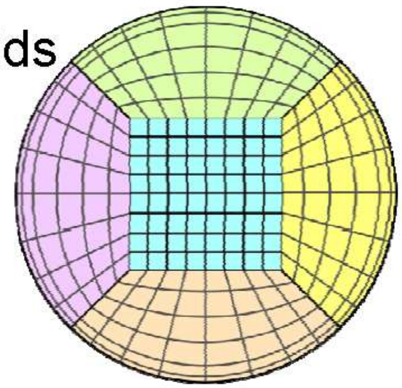
- Unstructured grids
  - Composed of arbitrarily positioned and connected elements
  - Can be composed of one unique element type or they can be hybrid (tetrahedra, hexas, prisms)
  - Triangle meshes in 2D and tetrahedral grids in 3D are most common
  - Can adapt to local features (small vs. large cells)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



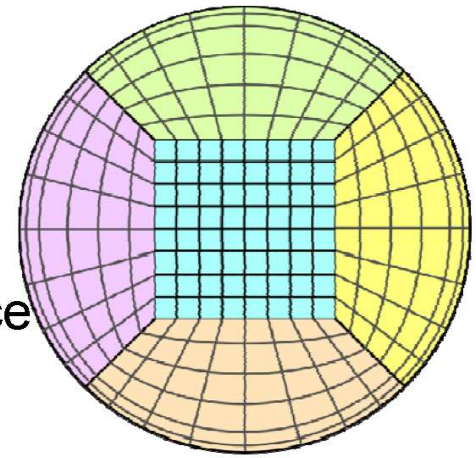
Data discretizations

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

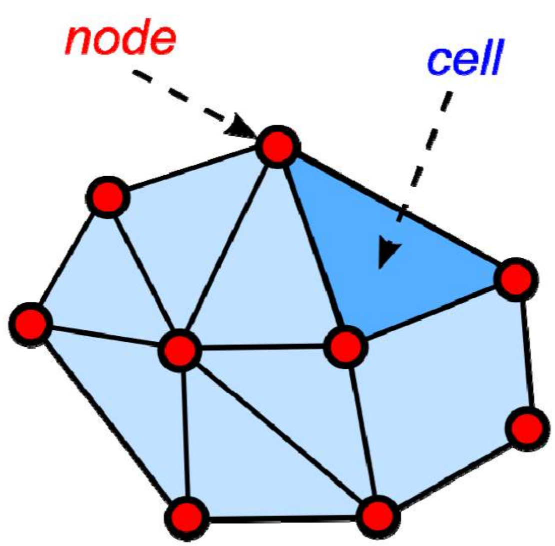


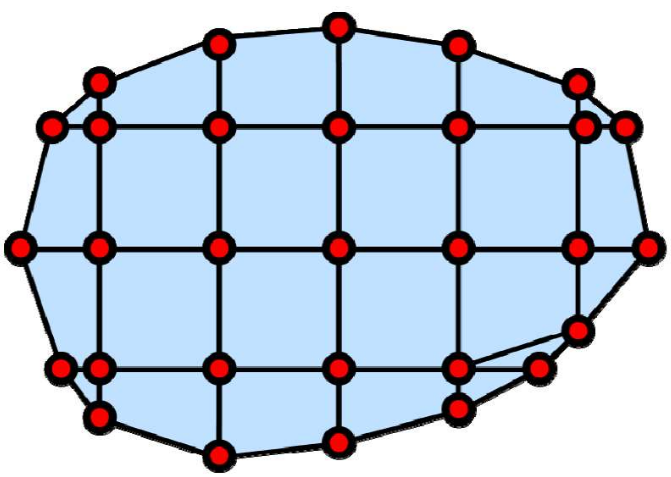
Unstructured grids



2D unstructured grids:

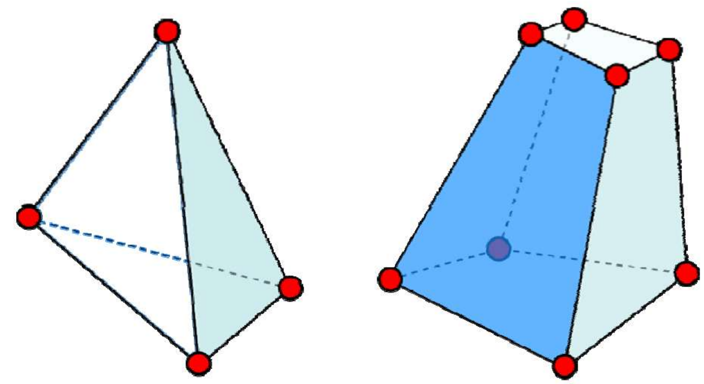
- cells are triangles and/or quadrangles
- domain can be a surface embedded in 3-space
  (distinguish n-dimensional from n-space)



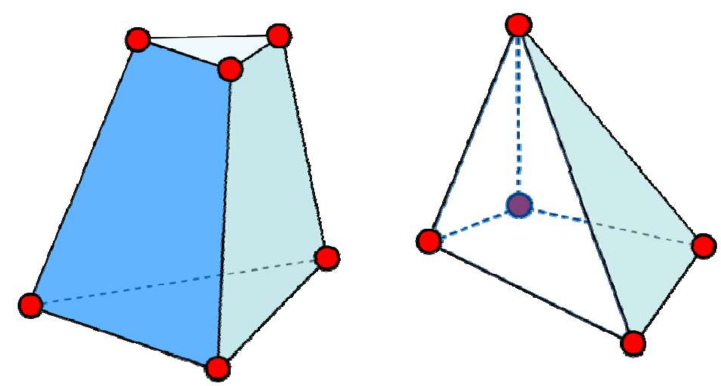


3D unstructured grids:

- cells are tetrahedra or hexahedra

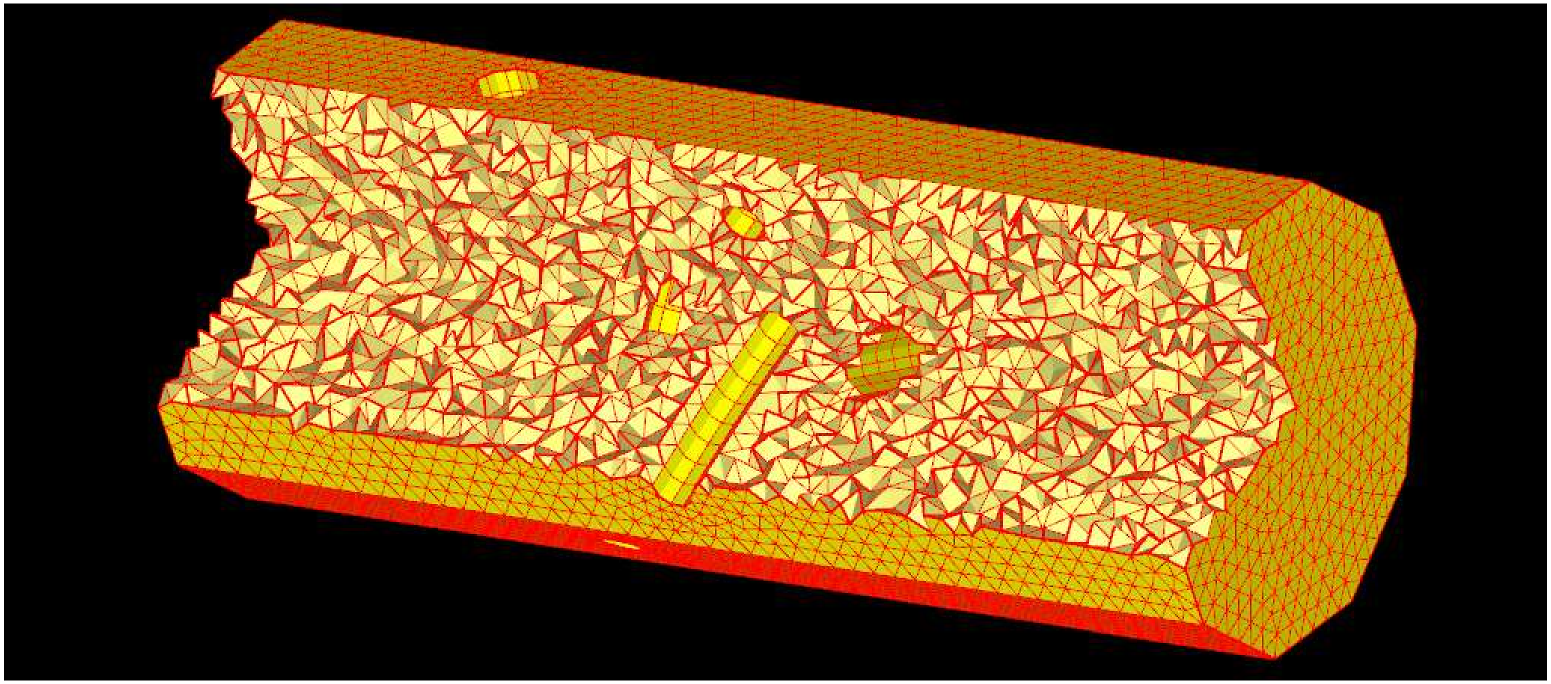


- mixed grids ("zoo meshes") require additional types: wedge (3-sided prism), and pyramid (4-sided)



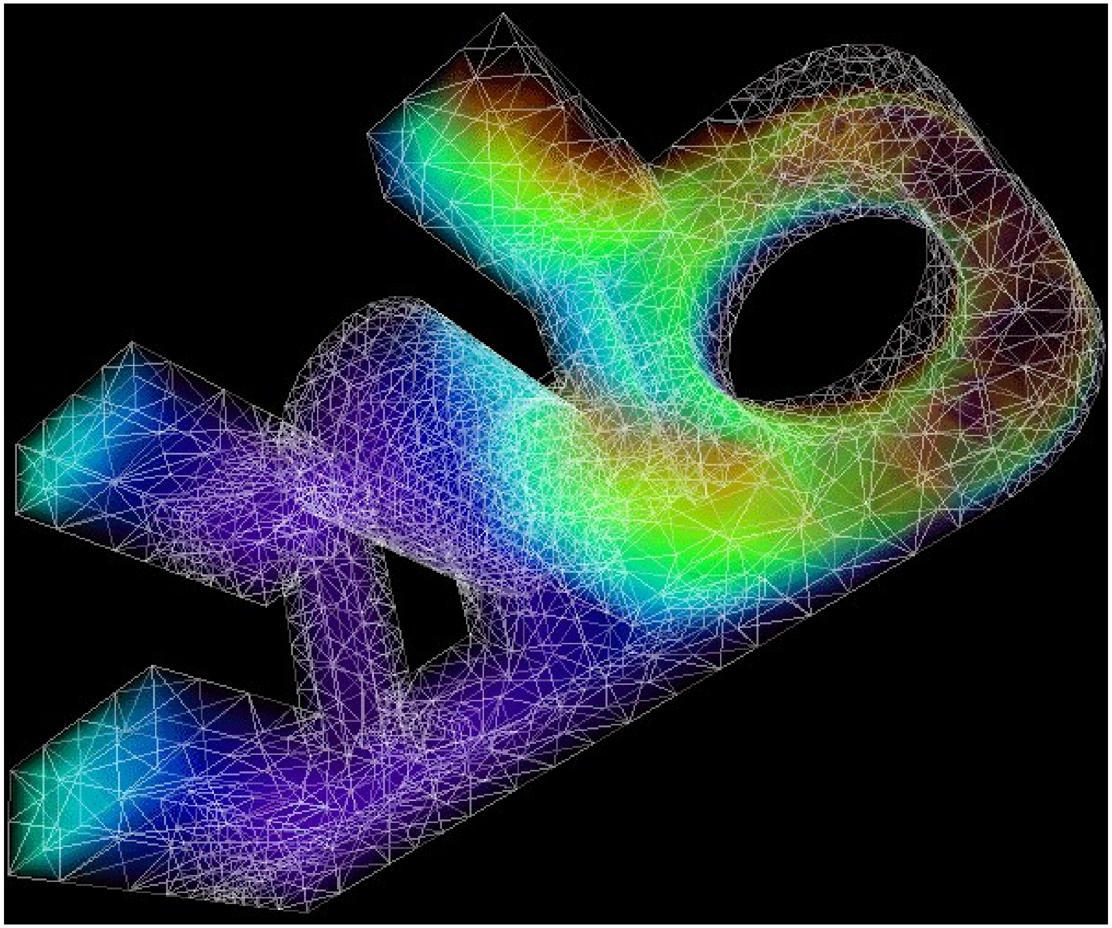
Common Unstructured Grid Types (1)

- Simplest: purely tetrahedral



Grid Structures

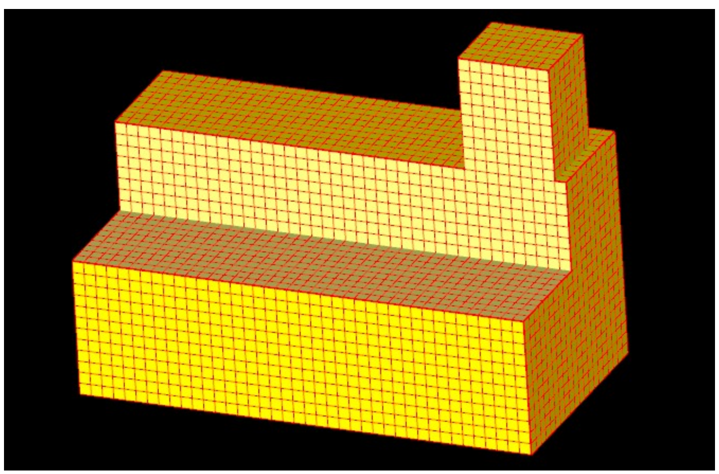
Tet grid example

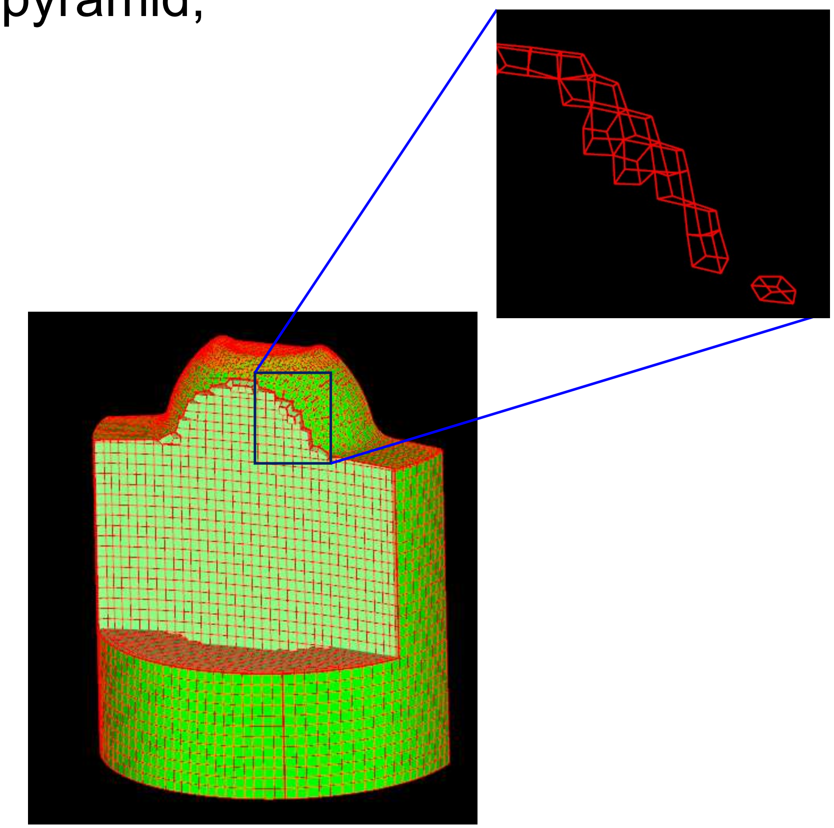


Common Unstructured Grid Types (2)

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

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

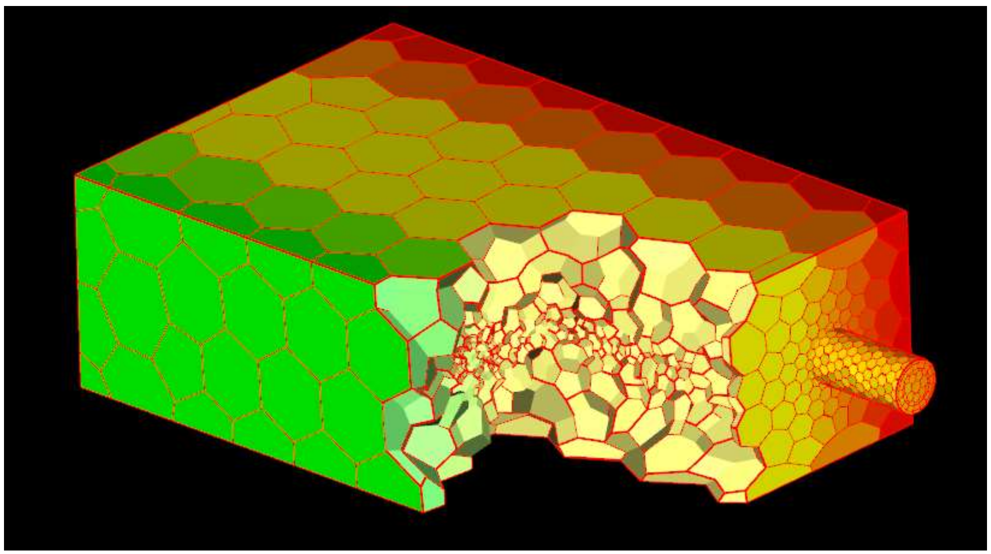


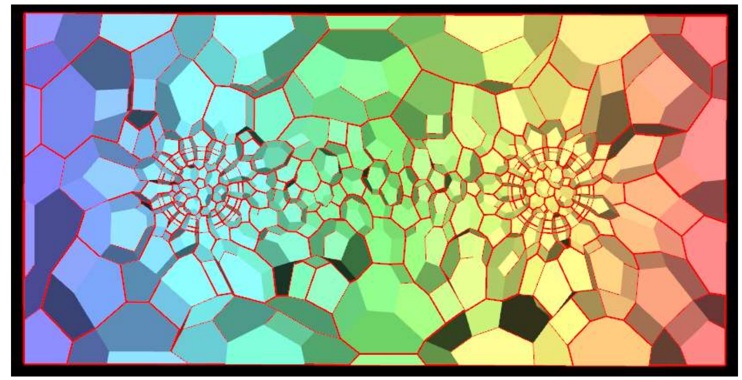


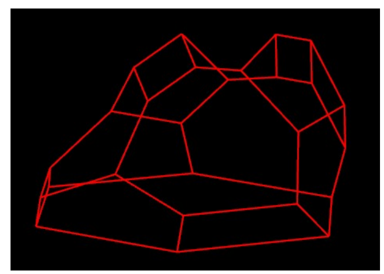
Common Unstructured Grid Types (3)

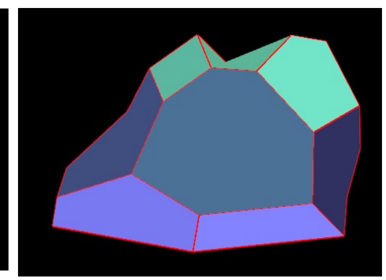
(Nearly) arbitrary polyhedra

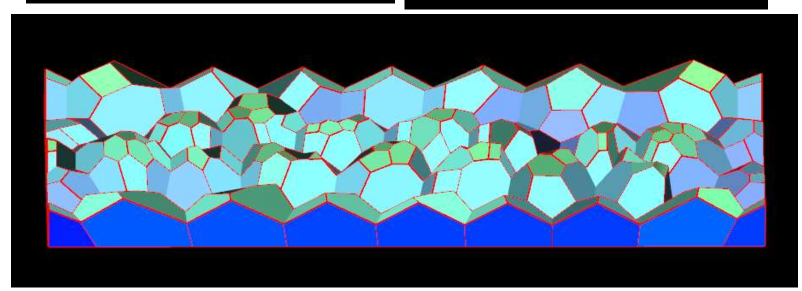
- Possibly non-planar faces







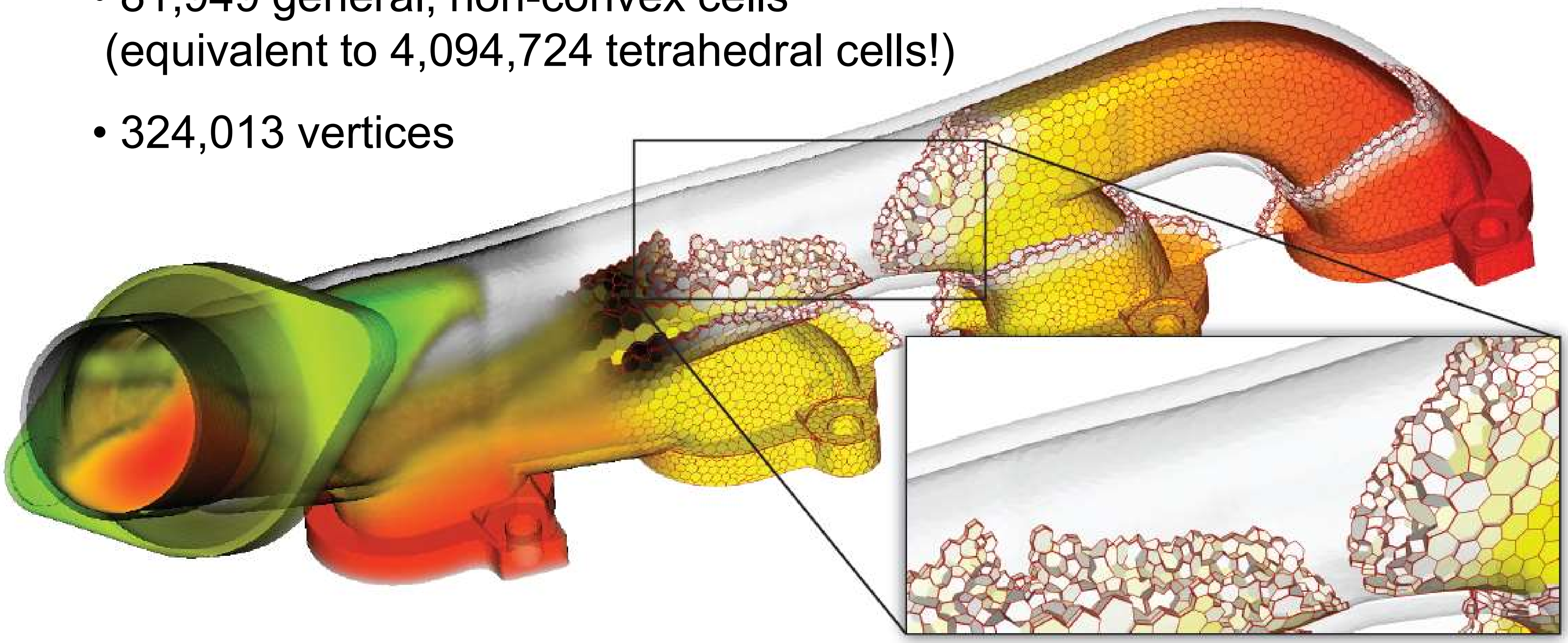




Example: General Polyhedral Cells

Exhaust manifold

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

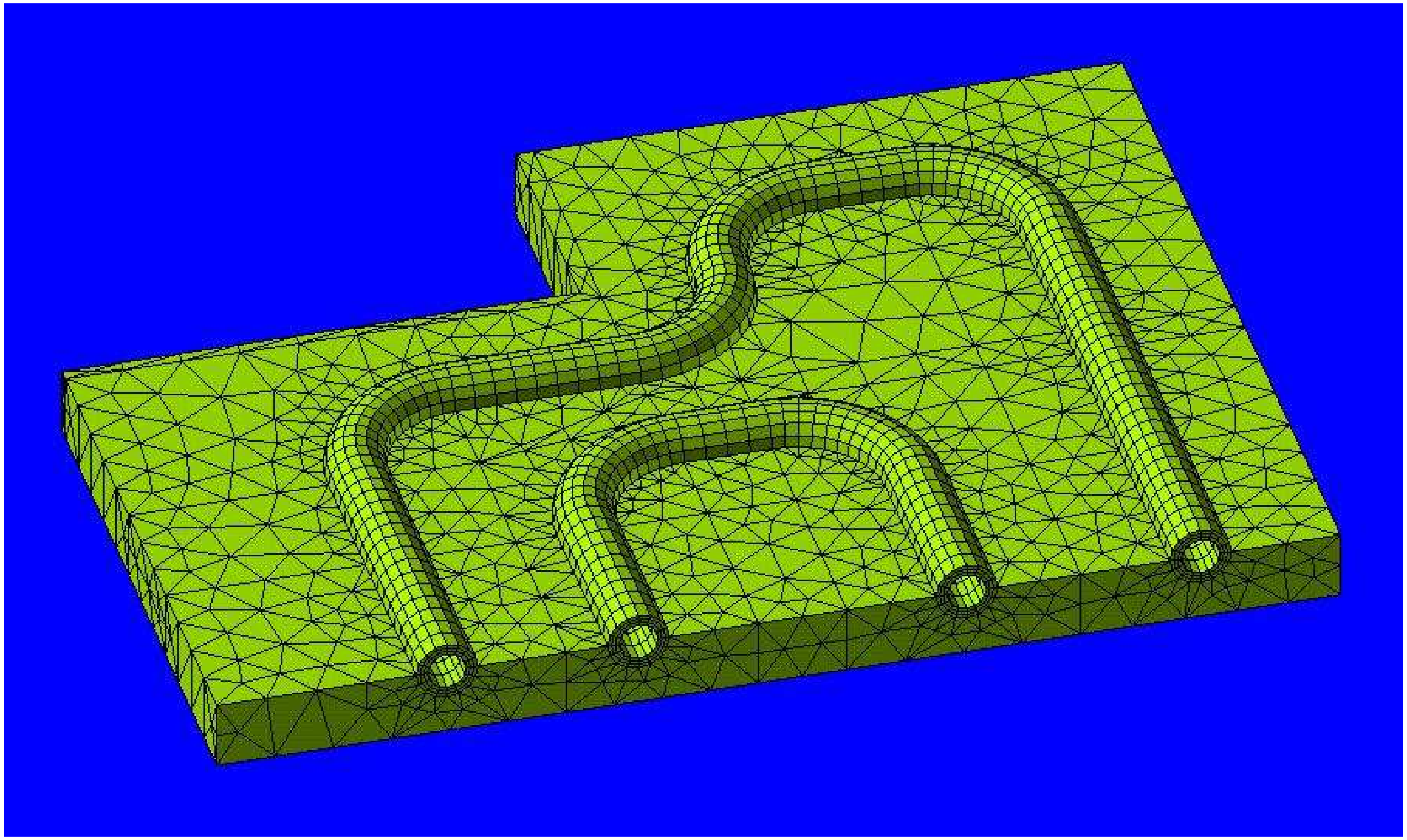


- Color coding: temperature distribution

Hybrid Grids

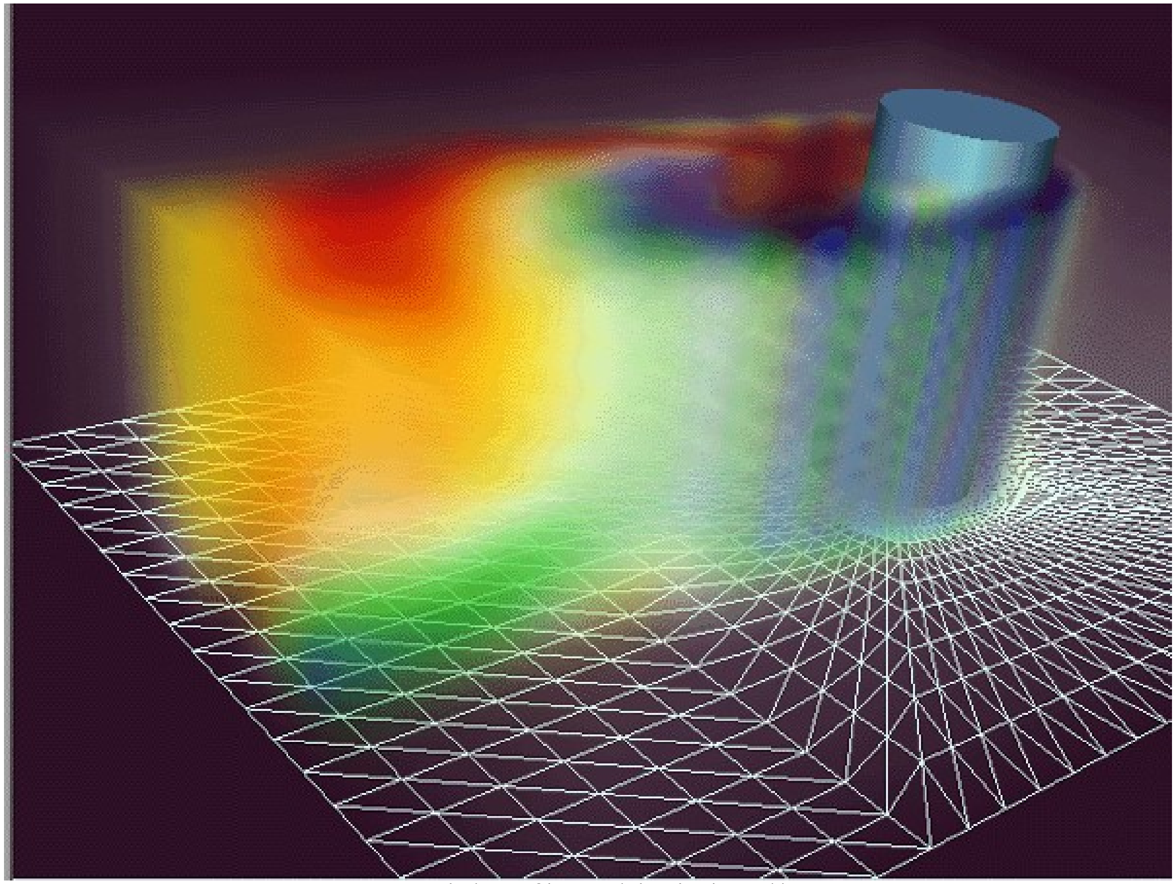
Data Structures

- Hybrid grids
  - Combination of different grid types



Data Structures

Hybrid grid example



© Weiskopf/Machiraju/Möller

Data Structures

Data Structures

- Typical implementations of unstructured grids
  – Direct form

Coords for
vertex 1

```
x1,y1,(z1)
x2,y2,(z2)    face 1
x3,y3,(z3)
x2,y2,(z2)
x3,y3,(z3)    face 2
x4,y4,(z4)
...
```

```
struct face                      2D
  float verts[3][2]
  DataType val;

struct face
  float verts[3][3]              3D
  DataType val;
```

  – Additionally store the data values
  – Problems: storage space, redundancy

Data Structures

- Typical implementations of unstructured grids
  - Indirect form

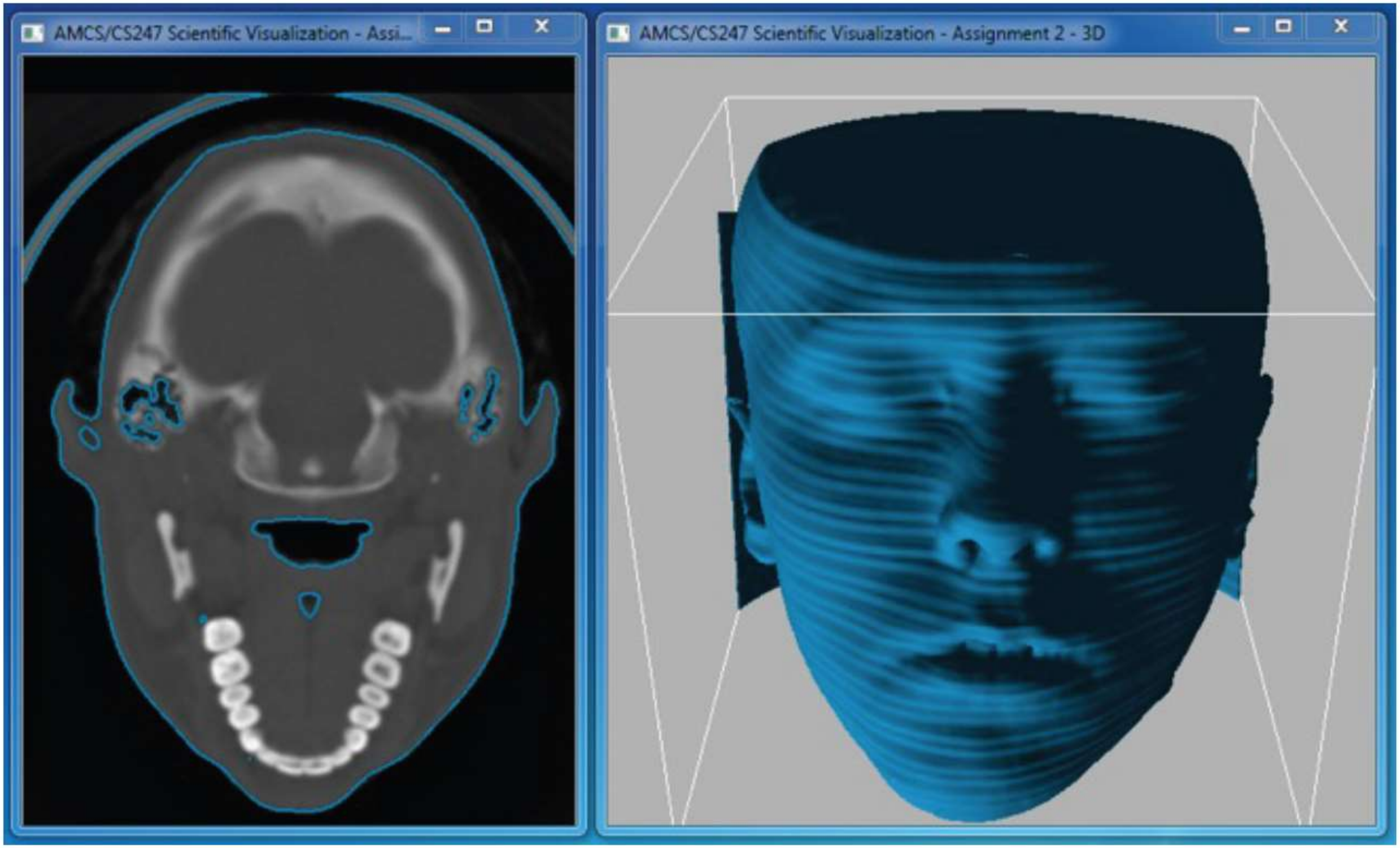
vertex list	face list
`x1,y1,(z1)`	`1,2,3`
`x2,y2,(z2)`	`1,2,4`
`x3,y3,(z3)`	`3,2,4`
`x4,y4,(z4)`	
	`...`

Coords for vertex 1

  - Indexed face set
  - More efficient than direct approach in terms of memory requirements
  - But still have to do global search to find local information (i.e. what faces share an edge)

Scalar Fields

Programming Assignment 2 + 3



Scalar Fields are Functions

- 1D scalar field: $\Omega \subseteq R \rightarrow R$
- 2D scalar field: $\Omega \subseteq R^2 \rightarrow R$
- 3D scalar field: $\Omega \subseteq R^3 \rightarrow R$

  $\rightarrow$ **volume visualization!**

more generally: $\Omega \subseteq$ n-manifold

Basic Visualization Strategies

Mapping to geometry

- Function plots
- Height fields
- Isocontours/isolines, isosurfaces

Color mapping

Specific techniques for 3D data

- Indirect volume visualization
- Direct volume visualization
- Slicing

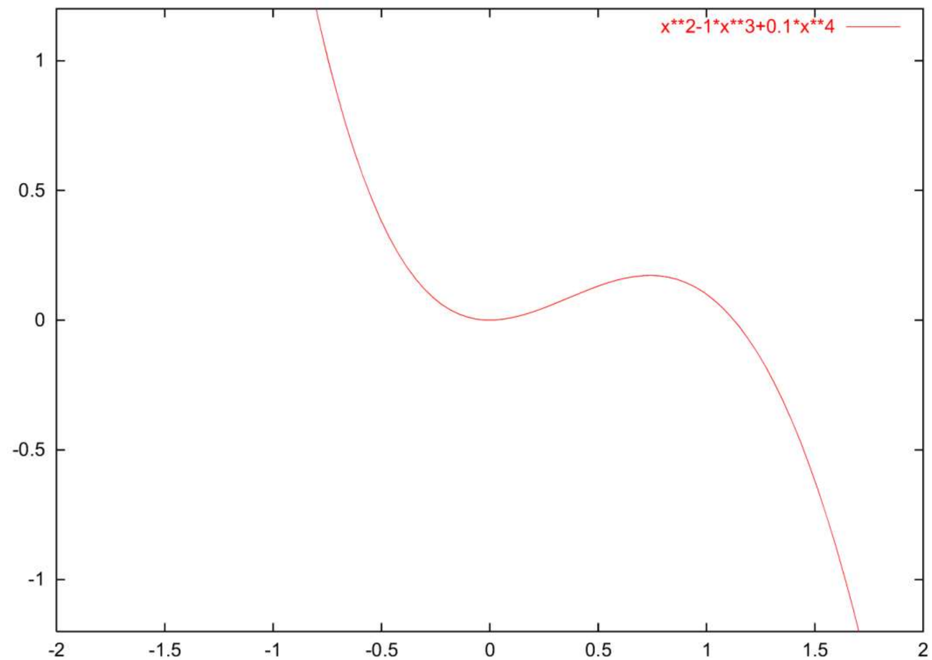
Visualization methods depend heavily on dimensionality of domain

Function Plots and Height Fields (1)

Function plot for a 1D scalar field

$$\{(x, f(x)) | x \in \mathbb{R}\}$$

- Points
- 1D manifold: line

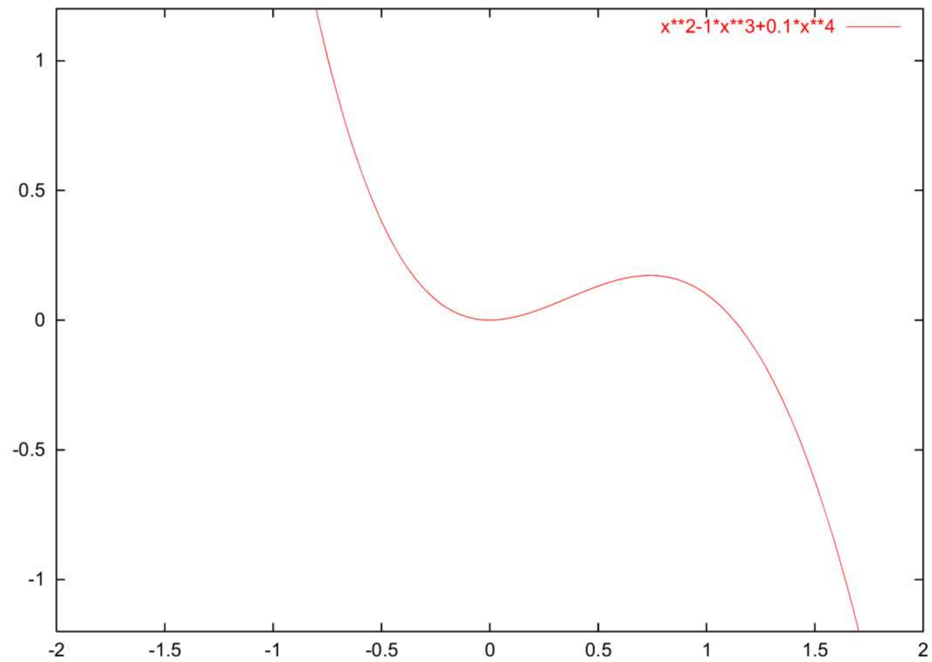


Function Plots and Height Fields (1)

Function plot for a 1D scalar field

$$\{(s, f(s)) \,|\, s \in \mathbb{R}\}$$

- Points
- 1D manifold: line



Function Plots and Height Fields (2)

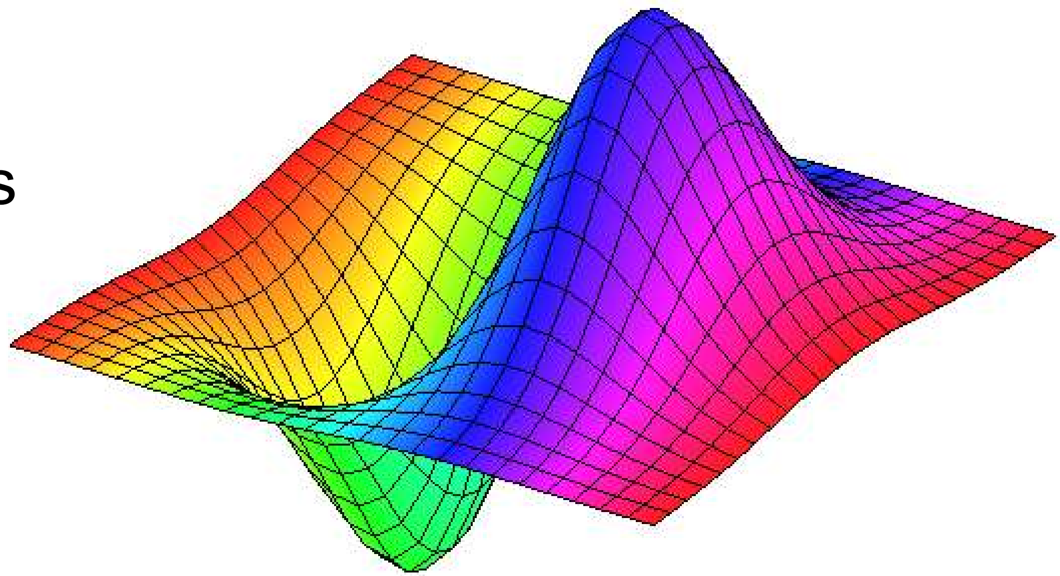
Function plot for a 2D scalar field

$$\{(x, f(x)) | x \in \mathbb{R}^2\}$$

- Points
- 2D manifold: surface

Surface representations

- Wireframe
- Hidden lines
- Shaded surface



Function Plots and Height Fields (2)

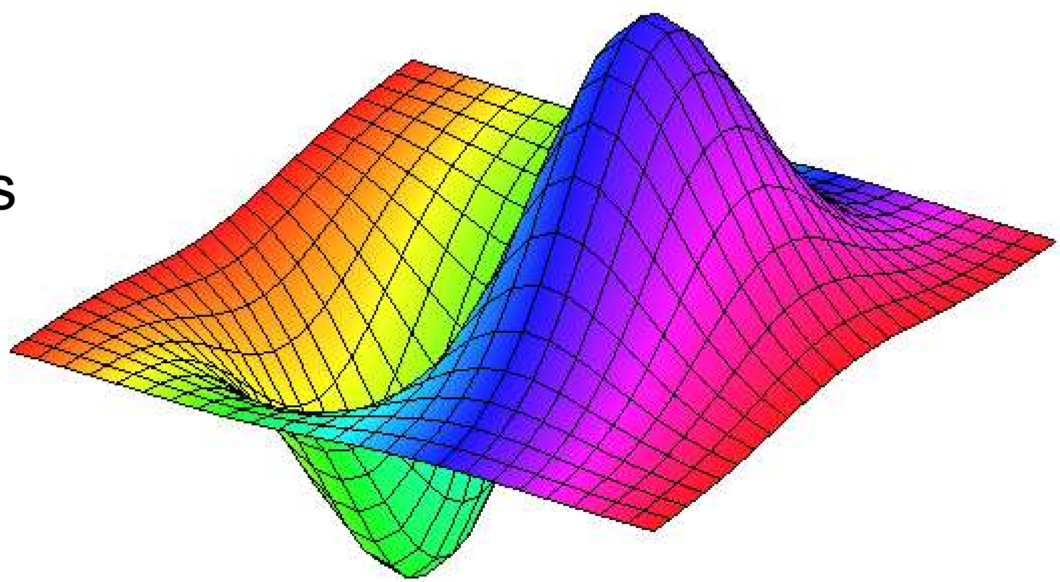
Function plot for a 2D scalar field

$$\{(s,t,f(s,t))\,|\,(s,t)\in\mathbb{R}^2\}$$

- Points
- 2D manifold: surface

Surface representations

- Wireframe
- Hidden lines
- Shaded surface

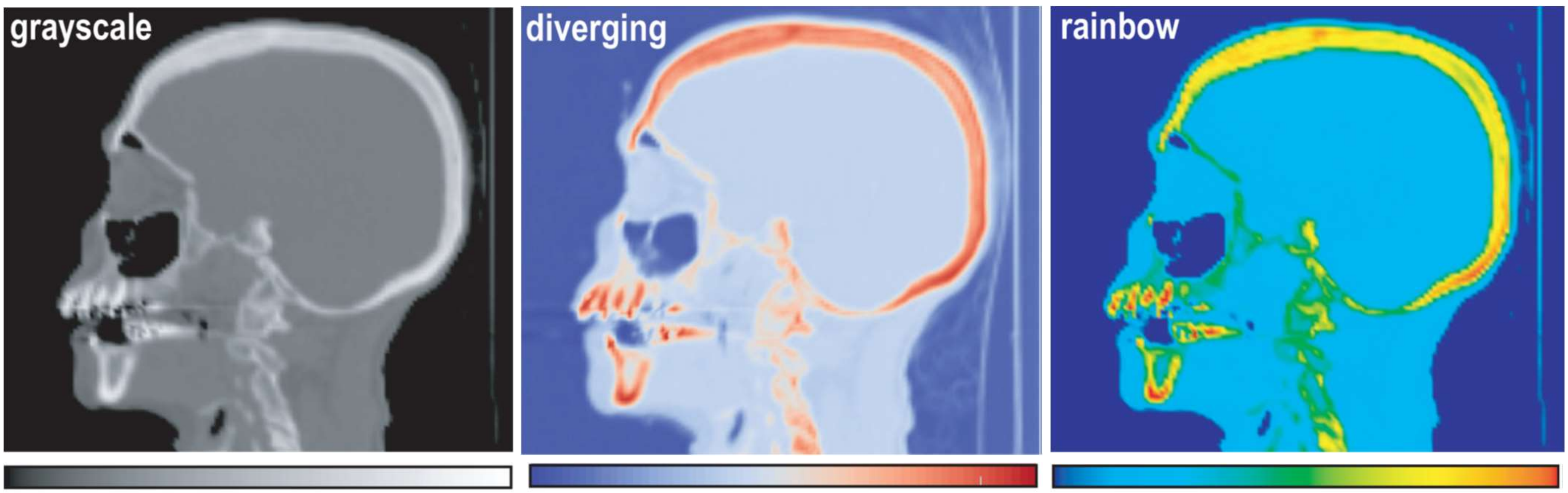


Color Mapping / Color Coding

Map scalar value to color

- Color table (e.g., array with RGB entries)
- Procedural computation; manual specification

With opacity (alpha value “A”): 1D *transfer function* (RGBA table, …)



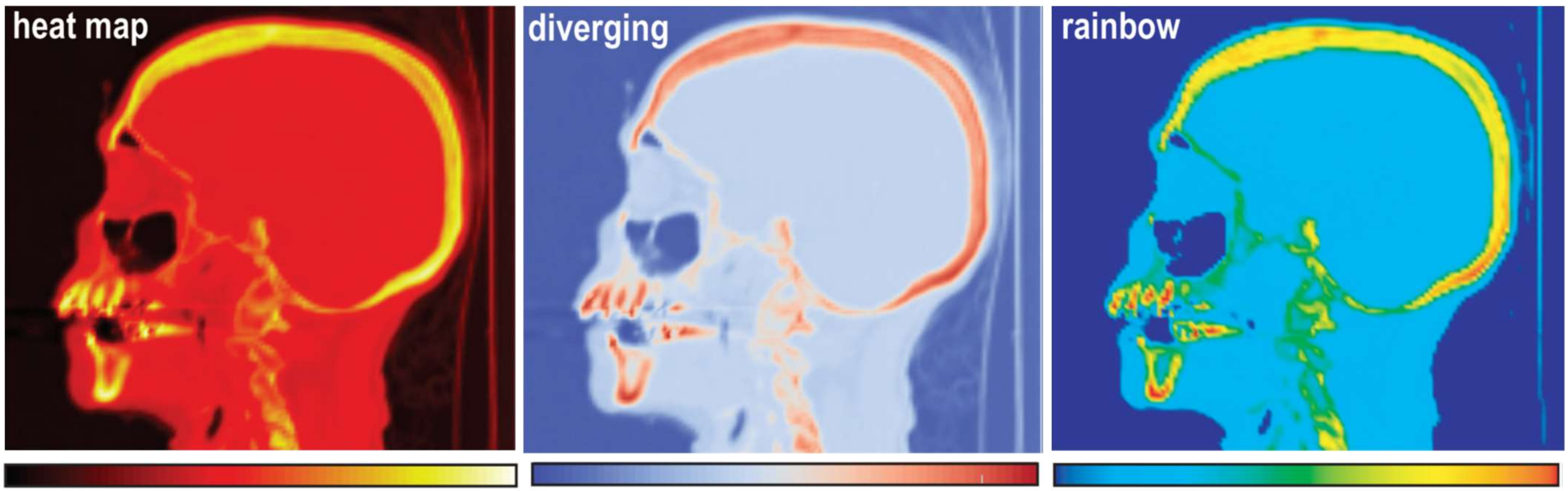
not recommended!

Color Mapping / Color Coding

Map scalar value to color

- Color table (e.g., array with RGB entries)
- Procedural computation; manual specification

With opacity (alpha value “A”): 1D *transfer function* (RGBA table, …)



not recommended!

Contours

Set of points where the scalar field *s* has a given value *c*:

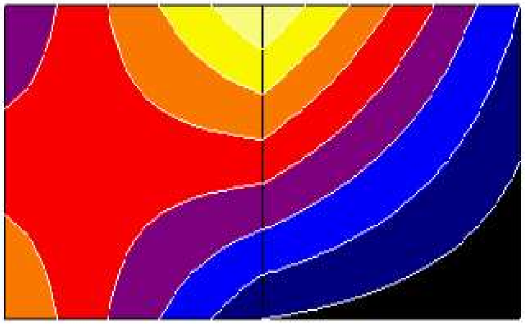
$$S(c) := f^{-1}(c) \qquad S(c) := \{x \in \mathbb{R}^n : f(x) = c\}$$

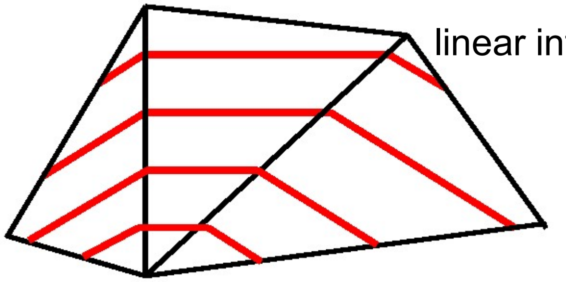
Common contouring algorithms

- 2D: marching squares, marching triangles
- 3D: marching cubes, marching tetrahedra

Implicit methods

- Point-on-contour test
- Isosurface ray-casting



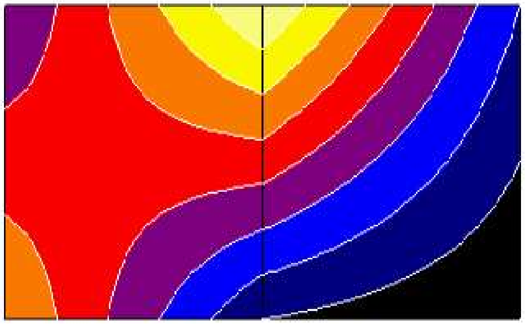


Contours

Set of points where the scalar field *s* has a given value *c*:

$$S(c) := f^{-1}(c) \qquad S(c) := \{x \in \mathbb{R}^2 : f(x) = c\}$$

bilinear interpolation

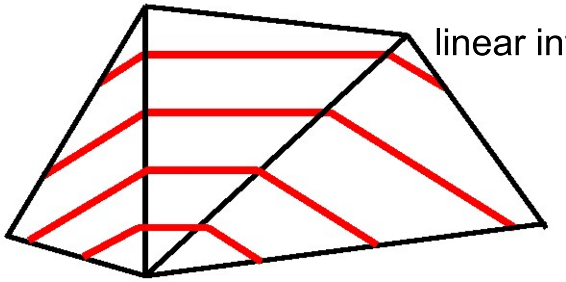


Common contouring algorithms

- 2D: marching squares, marching triangles
- 3D: marching cubes, marching tetrahedra

Implicit methods

- Point-on-contour test
- Isosurface ray-casting



linear interpolation

Contours

Set of points where the scalar field *s* has a given value *c*:

$$S(c) := f^{-1}(c) \qquad S(c) := \{x \in \mathbb{R}^3 : f(x) = c\}$$

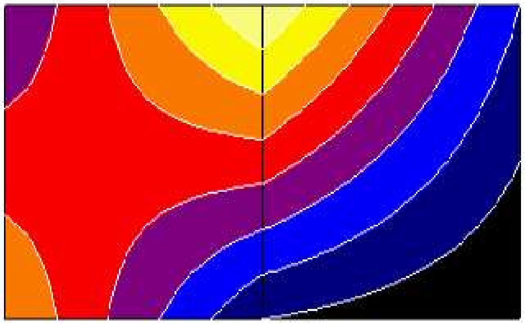
Common contouring algorithms

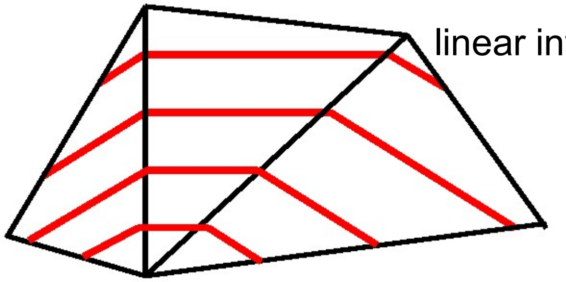
- 2D: marching squares, marching triangles
- 3D: marching cubes, marching tetrahedra

Implicit methods

- Point-on-contour test
- Isosurface ray-casting

bilinear interpolation





What are contours?

Set of points where the scalar field *s* has a given value *c*:

$$S(c) := \{x \in \mathbb{R}^n : f(x) = c\}$$

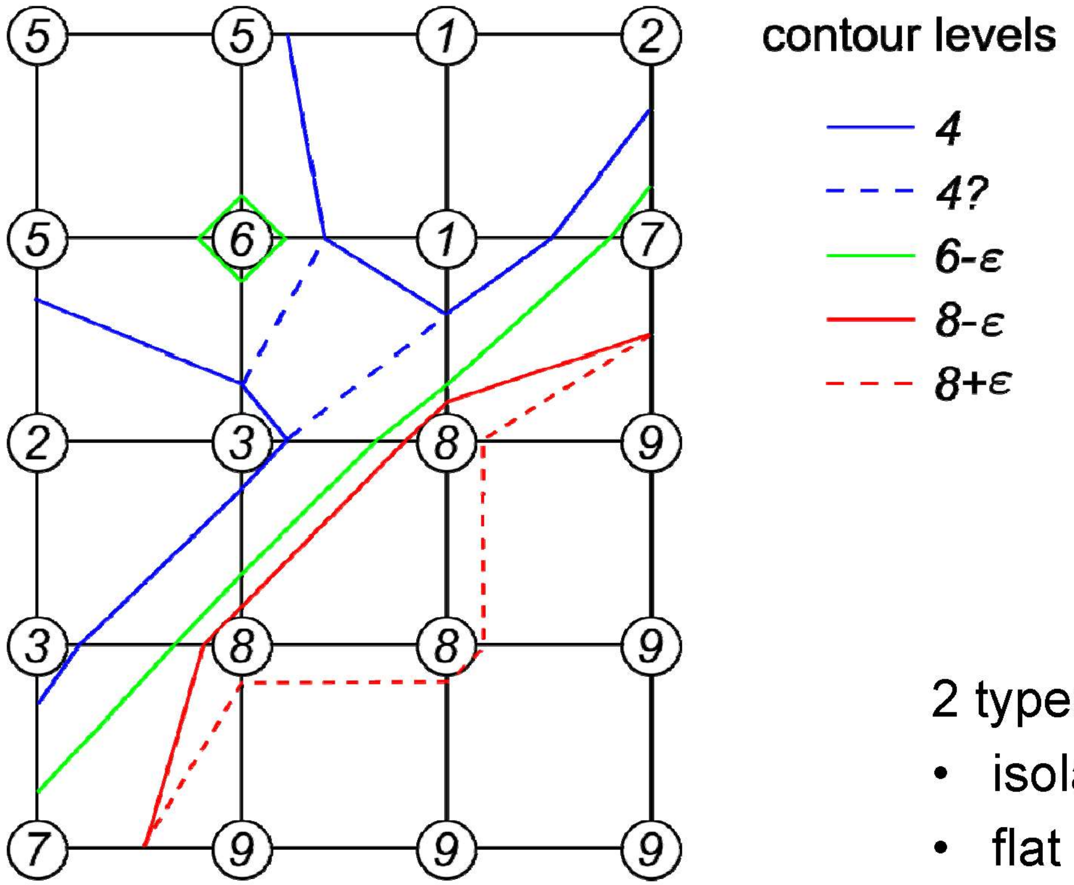
Examples in 2D:

- height contours on maps
- isobars on weather maps

Contouring algorithm:

- find intersection with grid edges
- connect points in each cell

Example



2 types of degeneracies:

- isolated points (c=6)
- flat regions (c=8)

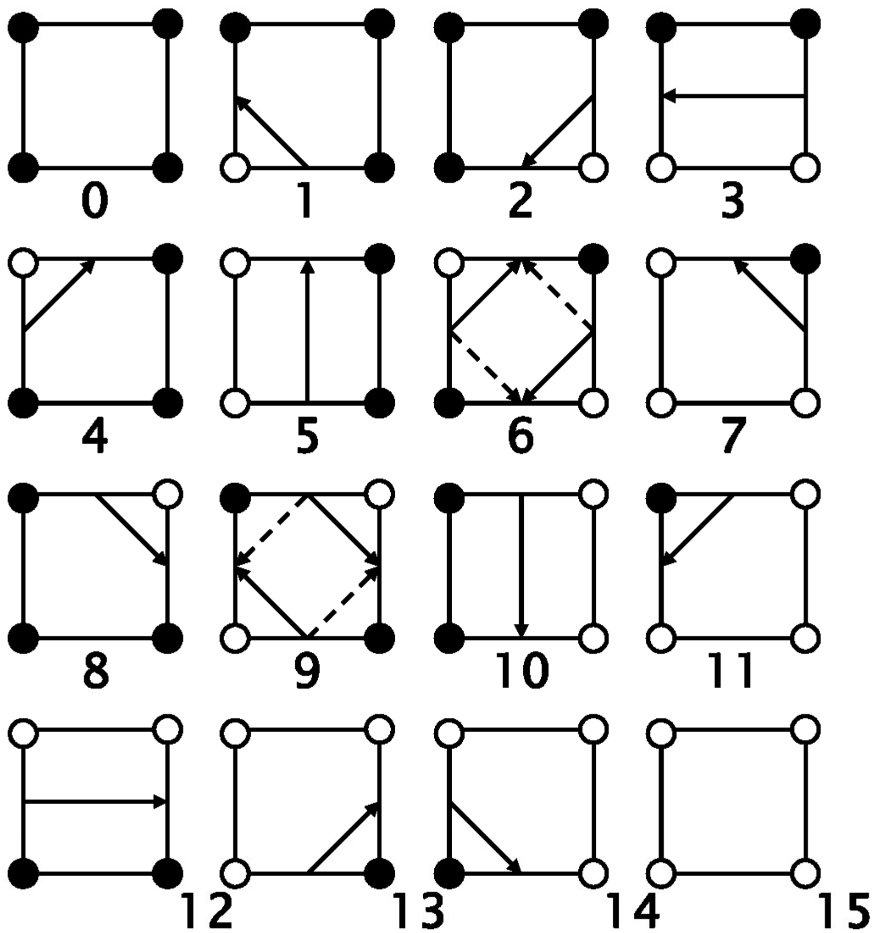
Basic contouring algorithms:

- cell-by-cell algorithms: simple structure, but generate disconnected segments, require post-processing
- contour propagation methods: more complicated, but generate connected contours

"Marching squares" algorithm (systematic cell-by-cell):

- process nodes in ccw order, denoted here as x_0, x_1, x_2, x_3
- compute at each node $\mathbf{x}_i$ the reduced field $\tilde{f}(x_i) = f(x_i) - (c - \varepsilon)$ (which is forced to be nonzero)
- take its sign as the i^{th} bit of a 4-bit integer
- use this as an index for lookup table containing the connectivity information:

Contours in a quadrangle cell



● $\tilde{f}(x_i) < 0$

○ $\tilde{f}(x_i) > 0$

- Alternating signs exist in cases 6 and 9.
- Choose the solid or dashed line?
- Both are possible for topological consistency.
- This allows to have a fixed table of 16 cases.

Thank you.

Thanks for material

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