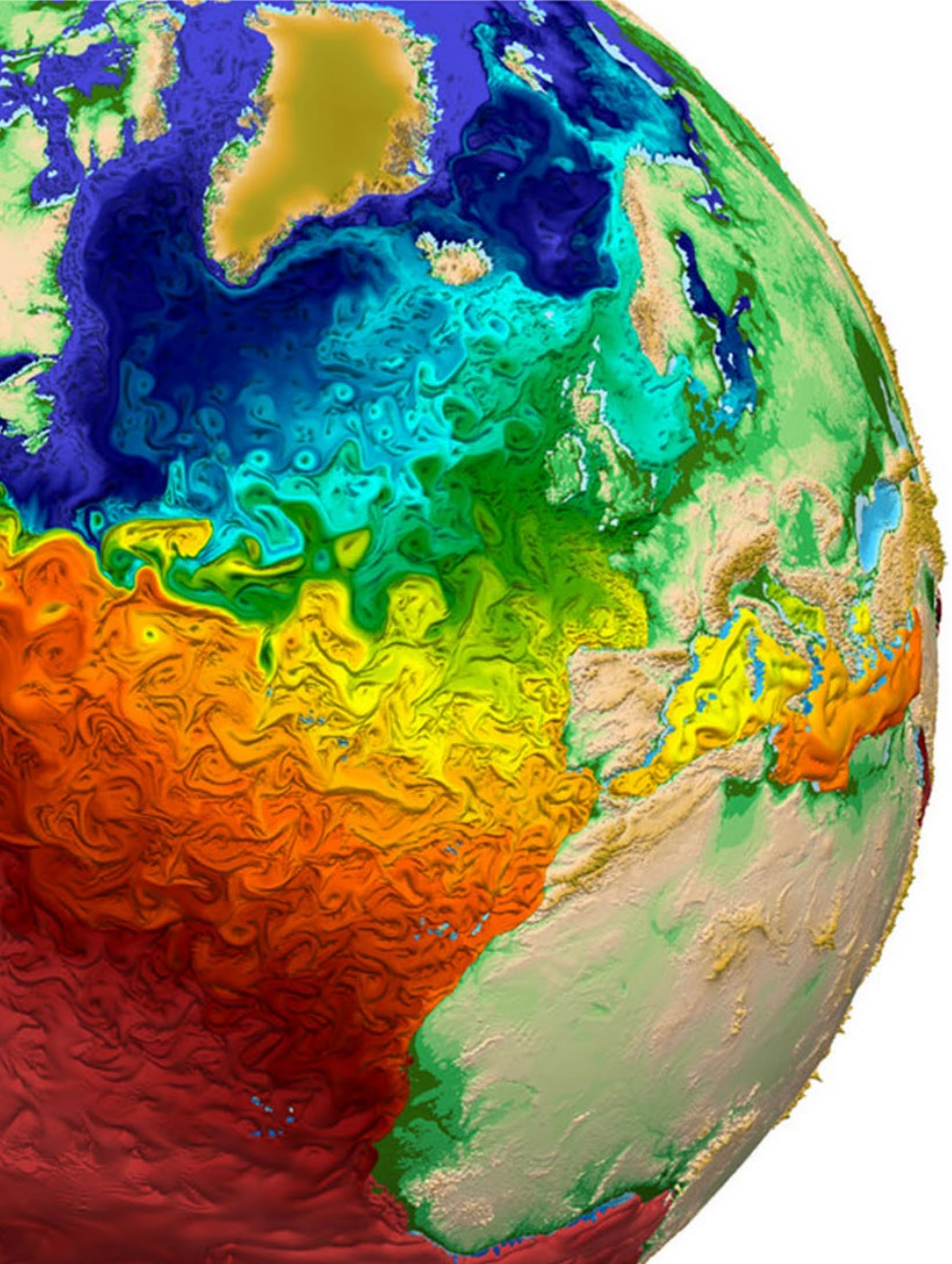


Domain Modeling

Meshes and Elements

Scientific Visualization

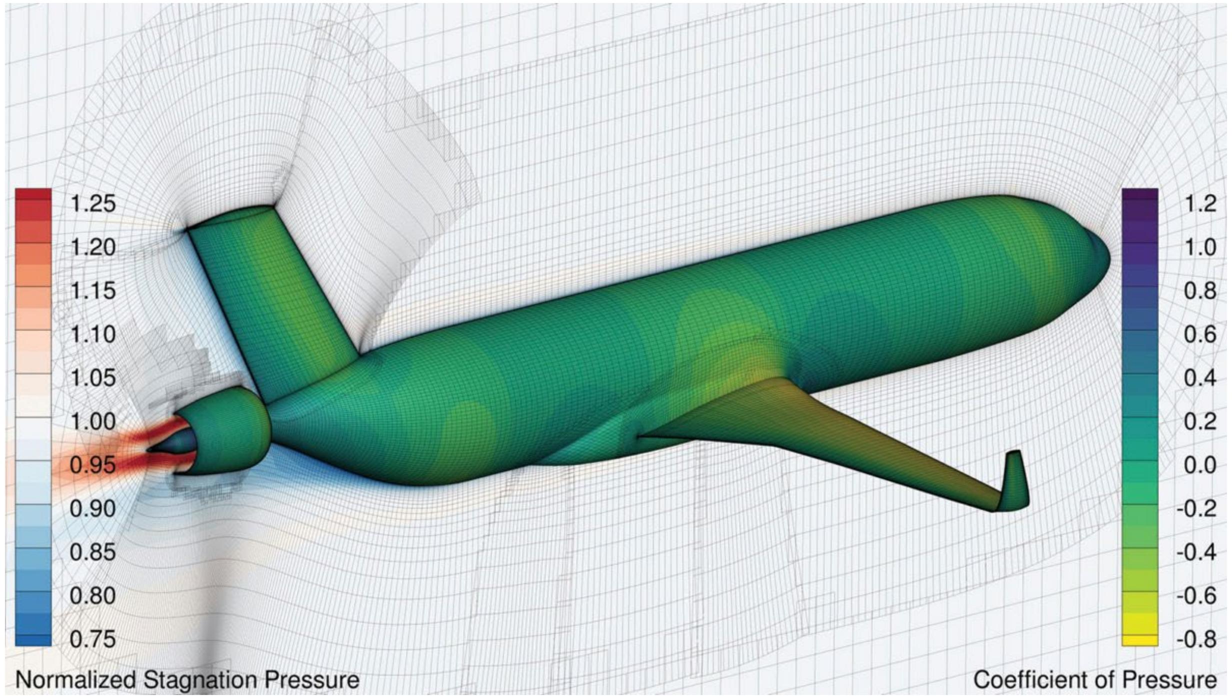
Professor Eric Shaffer

ILLINOIS

Domain Discretization

Scientific data often presented in a discretized domain

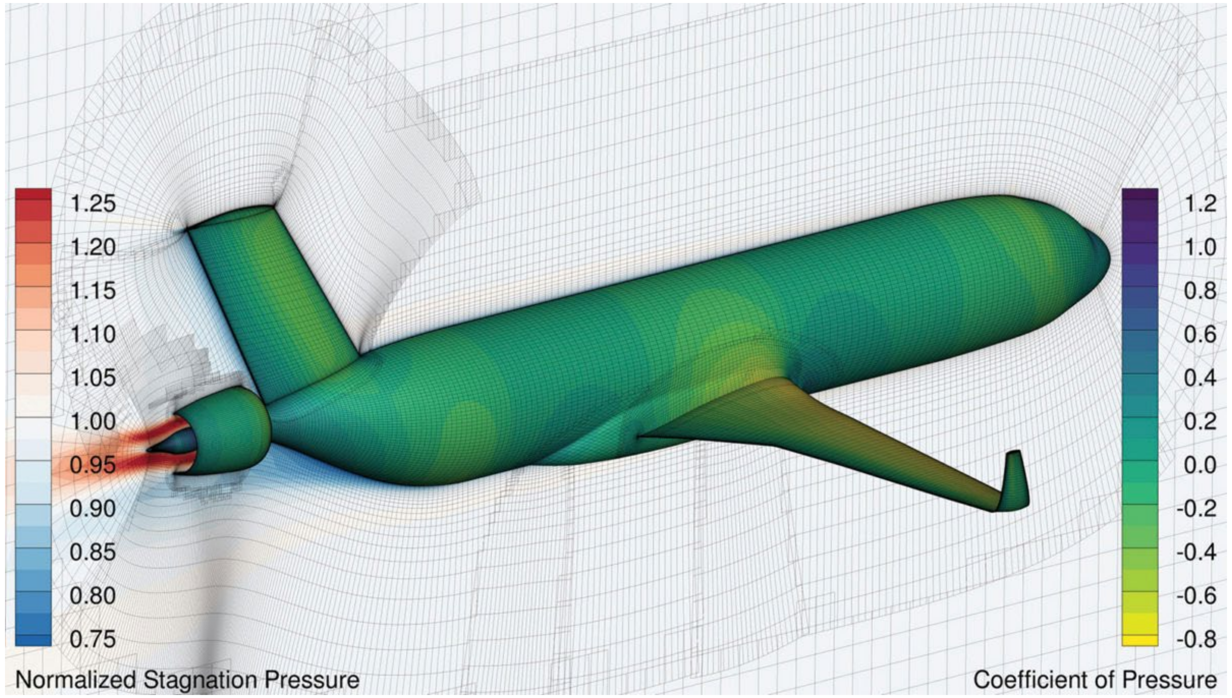
- Partitioned into discrete cells
- Each cell associated with some data
- Simulations often use domain discretization
- Data is easier to analyze, both visually and numerically



Domain Discretization

Many choices for how one can discretize

- Impacts space required for data storage
- Efficiency of operations on the data

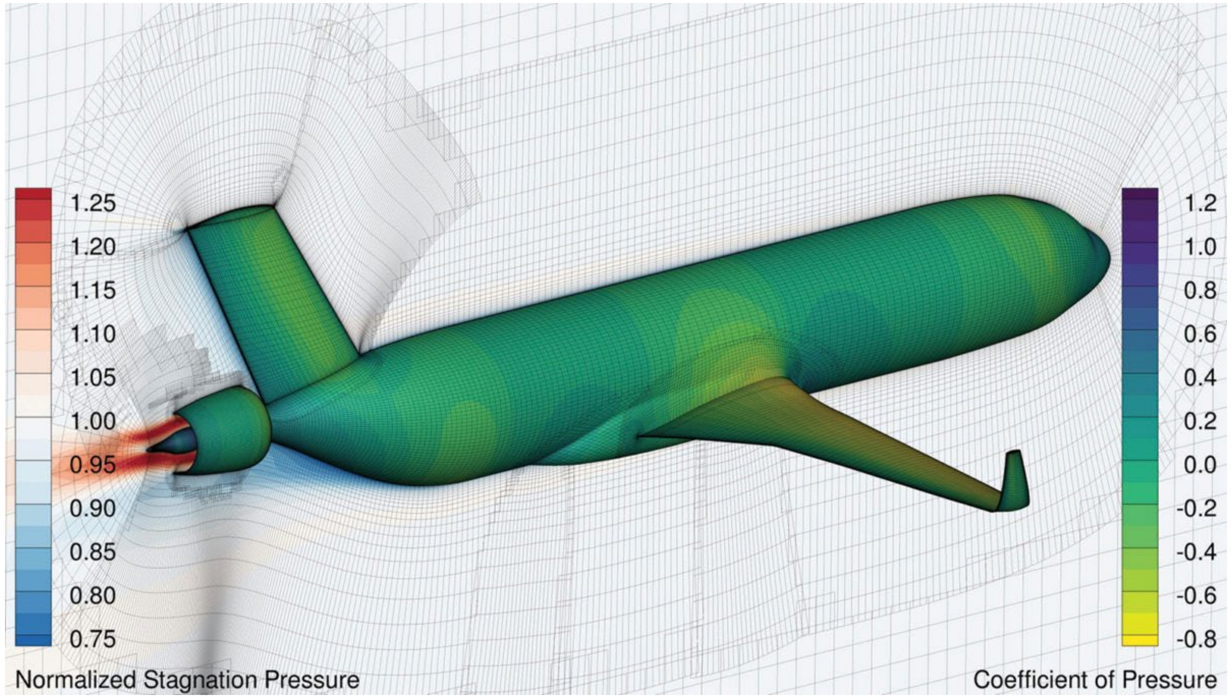


ILLINOIS

Domain Discretization

Discretization May Need to Change for Rendering

- Rendering may require processing the data mesh
- e.g. Given a hexahedral mesh may need to
    - Identify surface faces of the hexahedra
    - Triangulate them
    - Render the triangles



ILLINOIS

Terminology

Geometry

Positions of the vertices in space

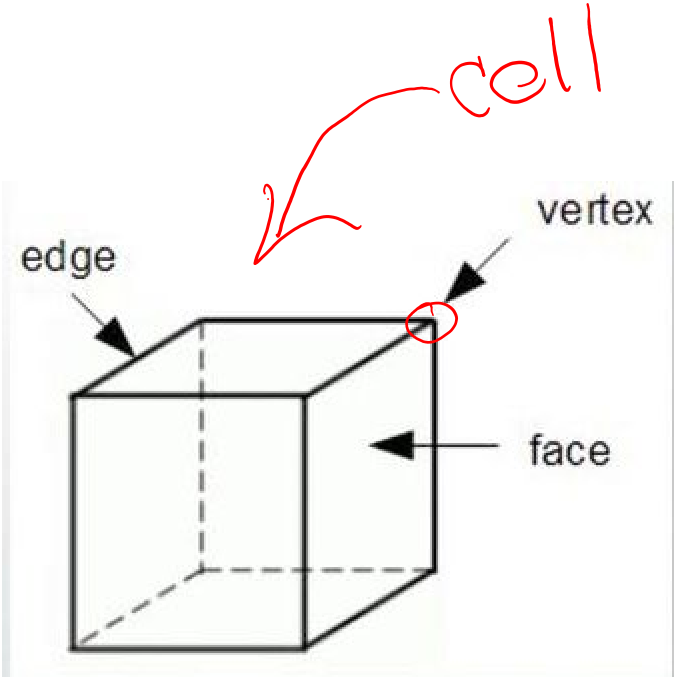
Can be structured or unstructured

Topology

Cells

Connectivity information

Can be structured or unstructured



Equivalent Terminology

- Grid = Mesh
- Nodes = Vertices
- Elements = Cells
- …lots of discipline specific terminology
  - e.g. in computational fluid dynamics (CFD) a zone is a group of cells
  - e.g. in geographic information systems (GIS) a triangulated irregular network (TIN) is a surface mesh of triangles

Cells and Grids

Cells
- provide interpolation over a small, simple-shaped spatial region

Grids (or meshes)
- partition our complex data domain D into cells
- allow applying per-cell interpolation (as described so far)

Data
– per vertex
– or –
– per cell

Given a domain D…

A grid G = {ci} is a set of cells such that

$c_i \cap c_j = \emptyset, \forall i \neq j$ no two cells overlap

$\bigcup_i c_i = D$ the cells cover all our domain

The dimension of the domain D constrains which cell types we can use

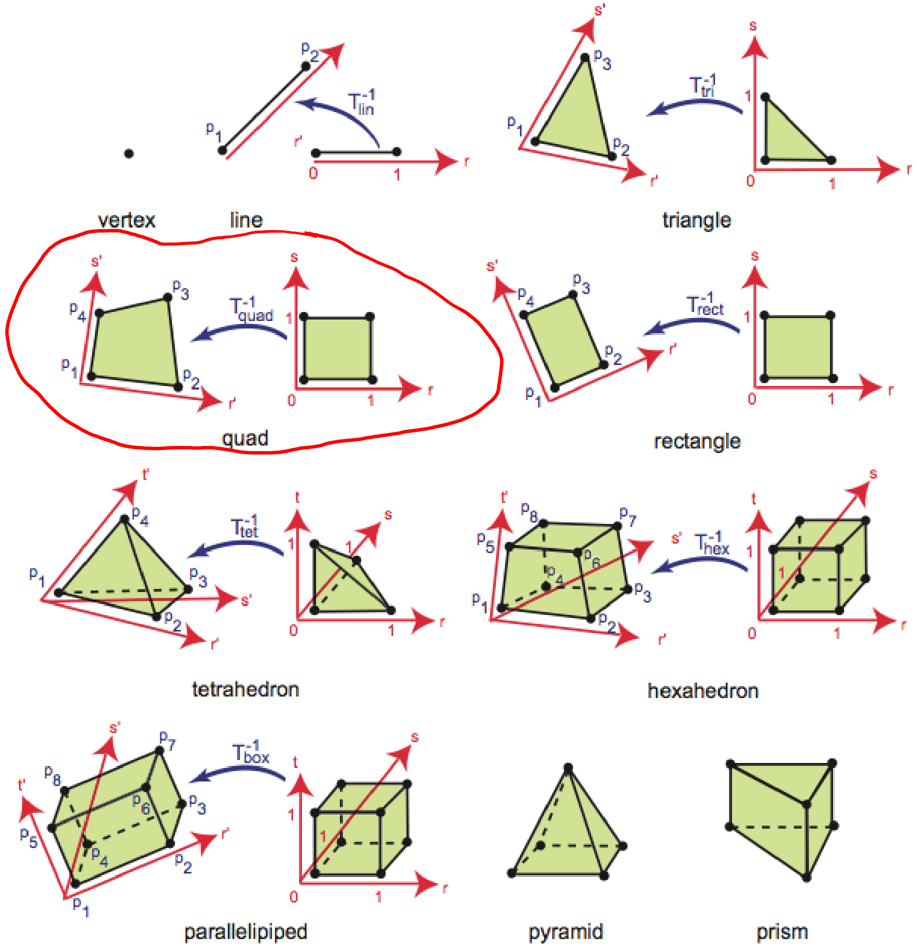
Cell Types

0D
- point

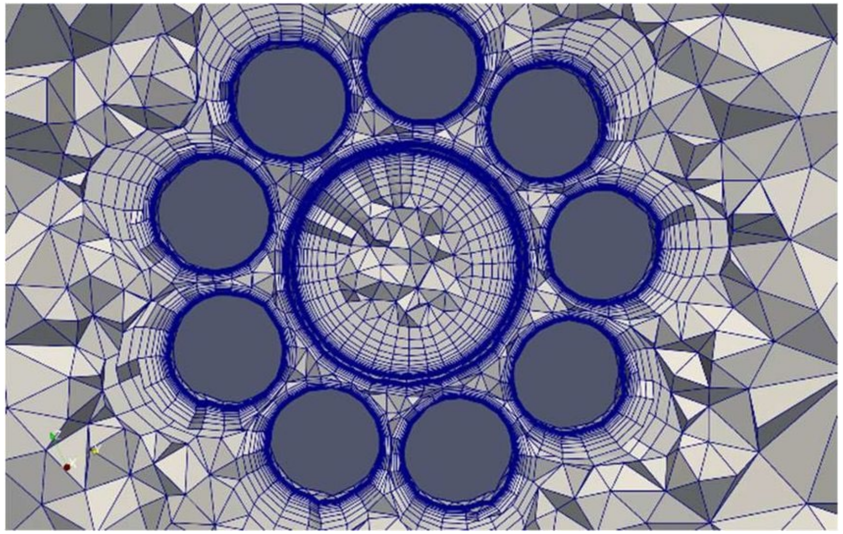
1D
- line

2D
- triangle, quad, rectangle

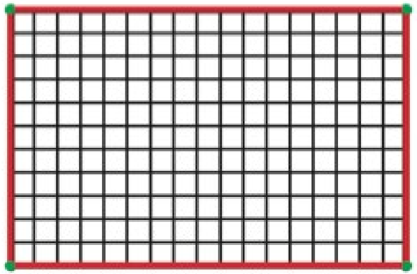
3D
- tetrahedron, parallelepiped, box, pyramid, prism, …

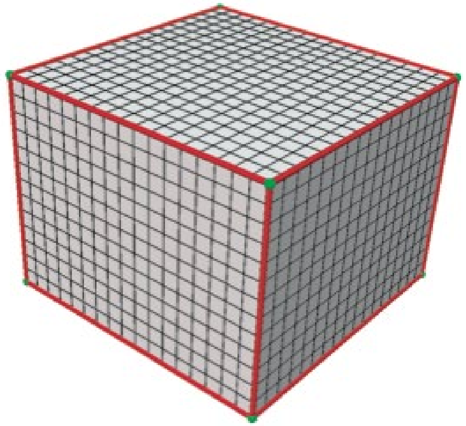


Hybrid Meshes contain more than 1 type of cell

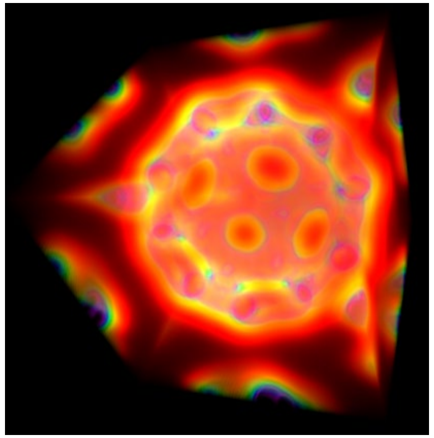


Uniform Grids (Images)





image

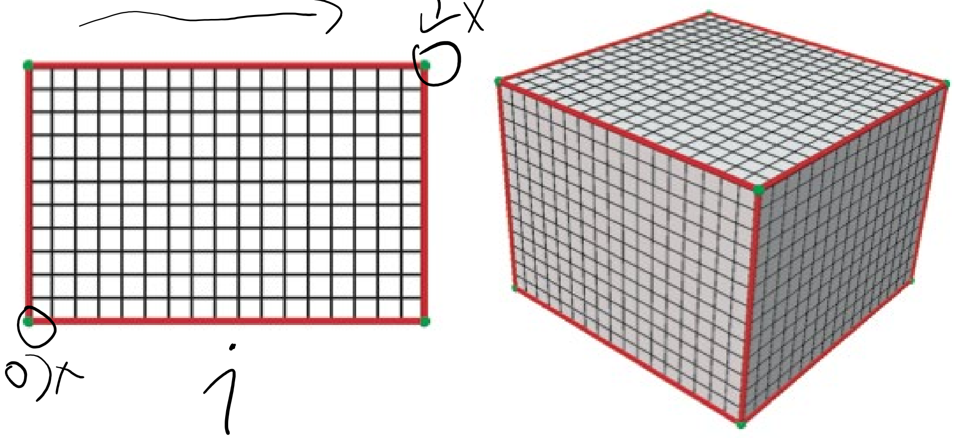


volume

- all cells have identical size and type (typically, square or cubic)
- cannot model non-axis-aligned domains

Uniform Grids (Images)

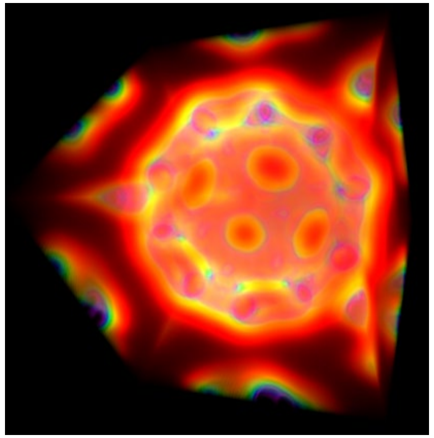
j



P(0,0)

i

image



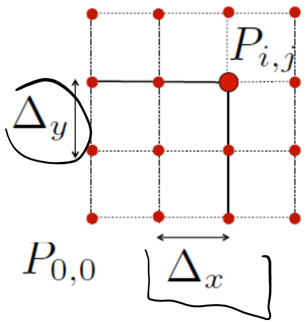
volume

- all cells have identical size and type (typically, square or cubic)
- cannot model non-axis-aligned domains

Storage requirements for the structure

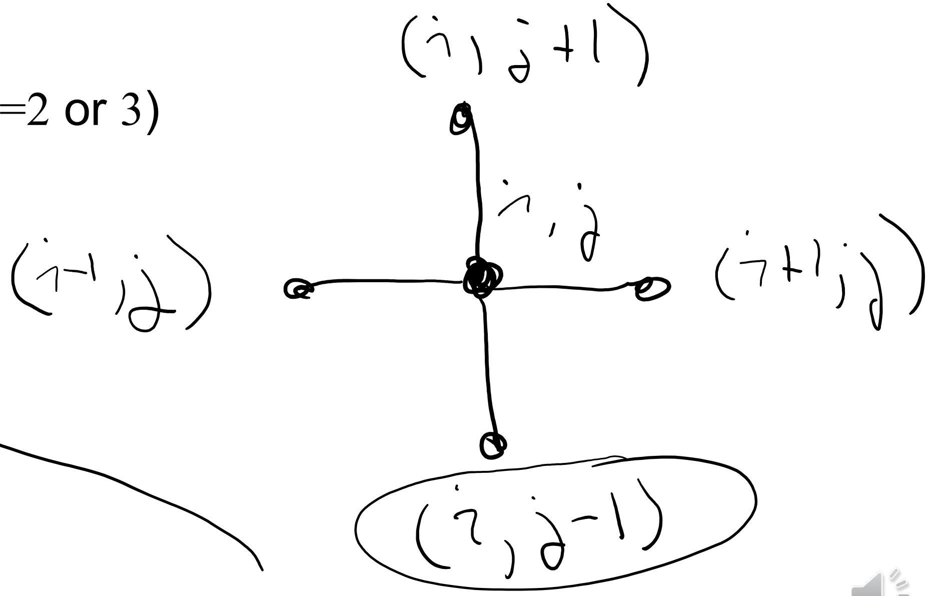
- m integers for the #vertices along each of the m dimensions of D (e.g. m=2 or 3)
- two corner points

Node positions can be computed instead of stored

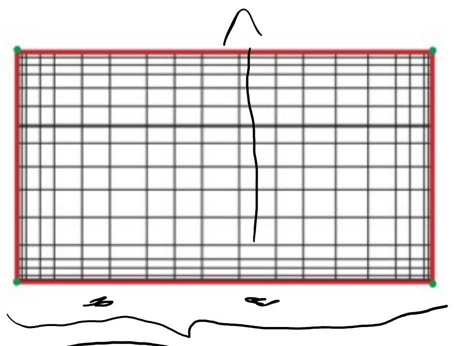


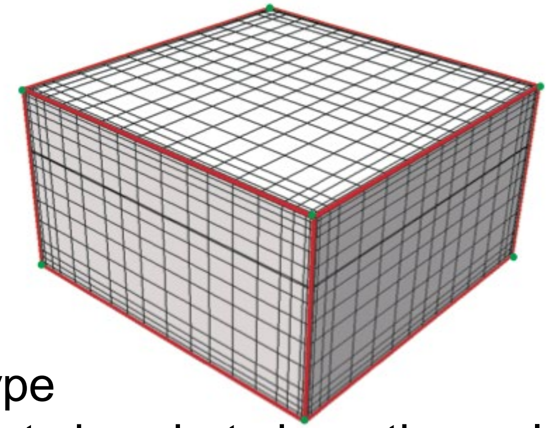
$$\boxed{P_{i,j} = P_{0,0} + i\Delta_x\vec{e}_x + j\Delta_y\vec{e}_y}$$

(1,0) (0,1)



Rectilinear Grids





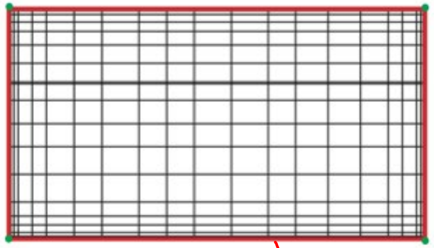
- all cells have same type
- cells can have different sizes but share them along axes

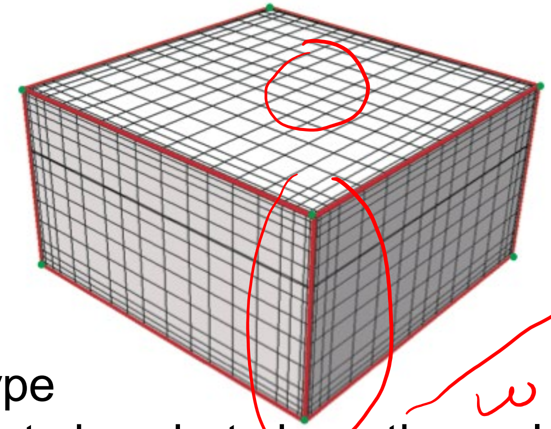
$$[\Delta x_0, \Delta x_1, \dots, \Delta x_n]$$

Rectilinear Grids

u x coords
v y coords
2 numbers

$u + v + 2$ numbers





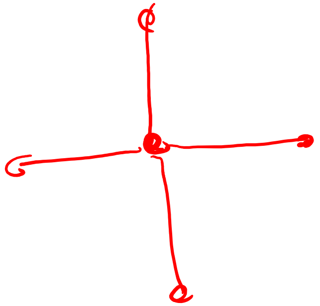
- all cells have same type
- cells can have different sizes but share them along axes

Storage requirements

$\sum_{i=1}^{m} d_i$ floats (coordinates of vertices along each of the m axes of D)

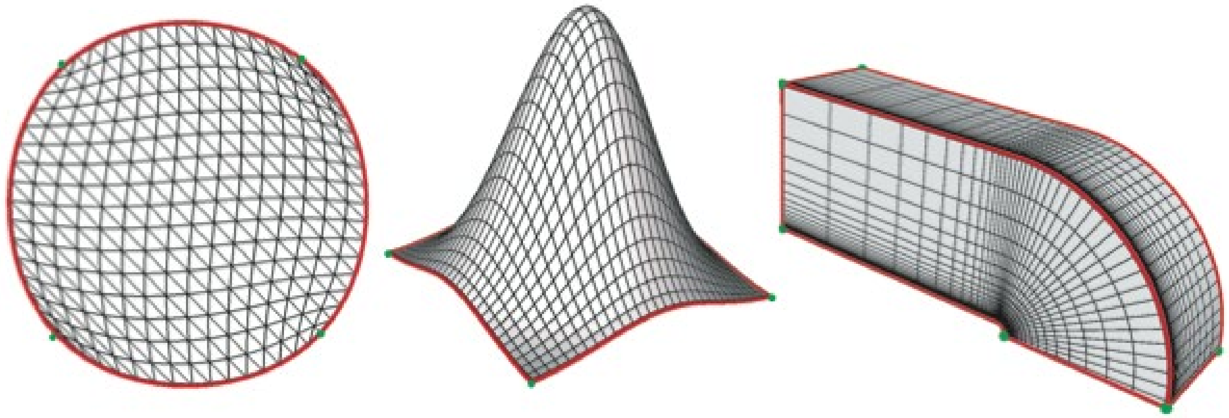
The number of vertices along each axis

Can compute node positions from stored coordinate information



$p(i,j) = (P_x[i], P_y[j])$

Curvilinear Grids

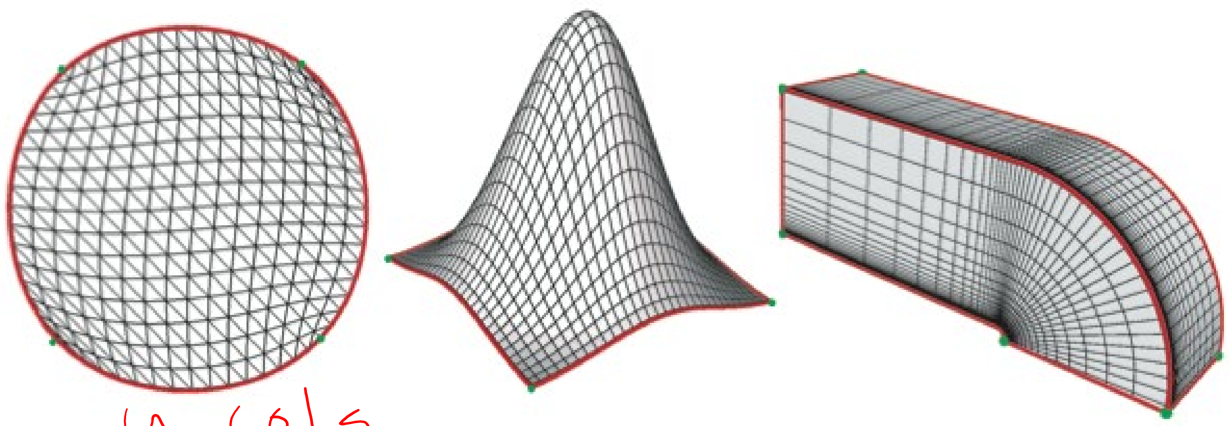


- all cells have same type
- cell vertex coordinates are freely (explicitly) specifiable…
- …as long as cells assemble in a matrix-like structure
- can approximate more complex shapes than rectilinear/uniform grids

Curvilinear Grids

3uv numbers

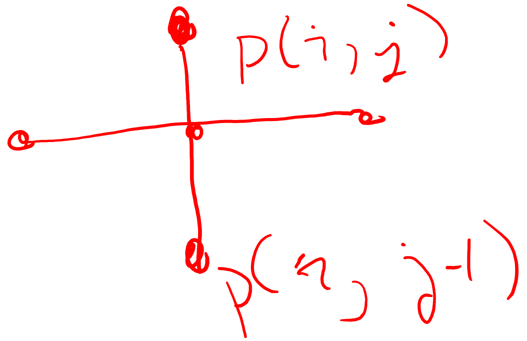
n rows



n cols

$p[i][j][k] = (x,y,z)$

- all cells have same type
- cell vertex coordinates are freely (explicitly) specifiable…
- …as long as cells assemble in a matrix-like structure
- can approximate more complex shapes than rectilinear/uniform grids



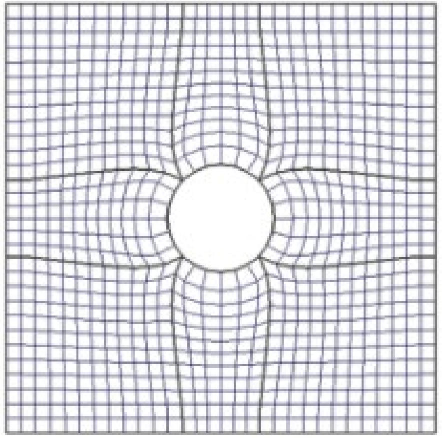
Storage requirements

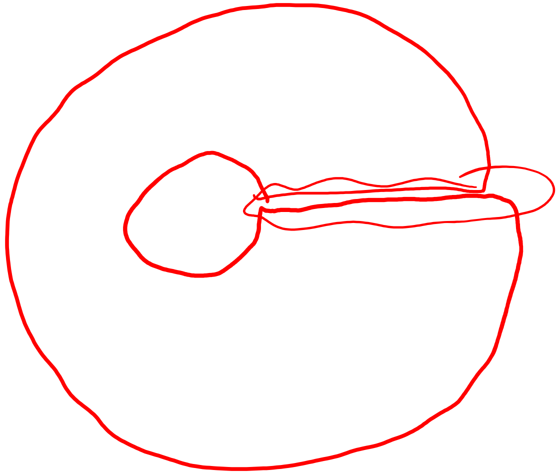
$\prod_{i=1}^{m} d_i$ floats (coordinates of all vertices)

Also 1 number for each axis (the number of vertices along each axis)

Unstructured Grids

Consider the domain D: a square with a hole in the middle

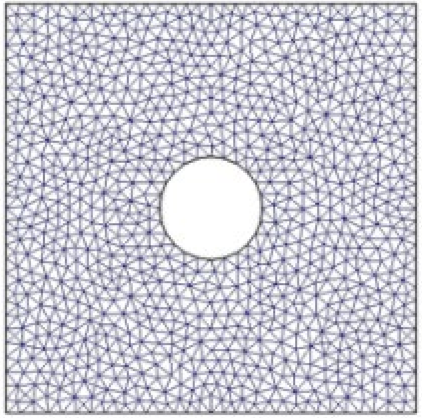




We cannot cover such a domain with a single structured grid (why?)

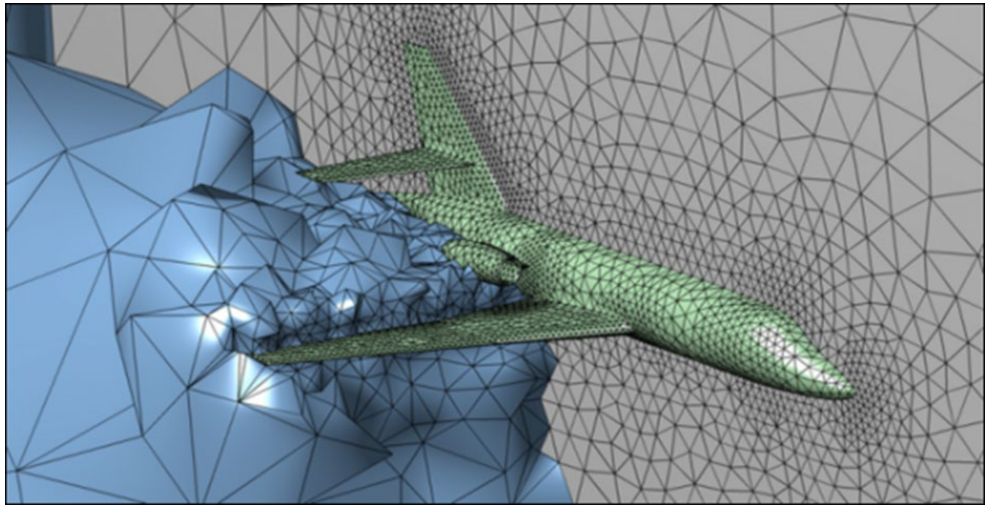
- it's not of genus 0, so cannot be covered with a matrix-like distribution of cells

For this, we need unstructured grids



Unstructured Grids

- most flexible grid type for modeling complex geometry
- both vertex coordinates and cell themselves are freely (explicitly) specifiable
- one storage implementation
  - vertex set
  - cell set



Storage requirements

$$V = \{v_i\}$$
$$C = \{c_i = (indices\ of\ vertices\ in\ V)\}$$

$$m\|V\| + s\|C\|$$

for a m-dimensional grid with cells having s vertices each

What operation is hard to do with just this information?

$c_1 = 1, 10, 15$

point in cell

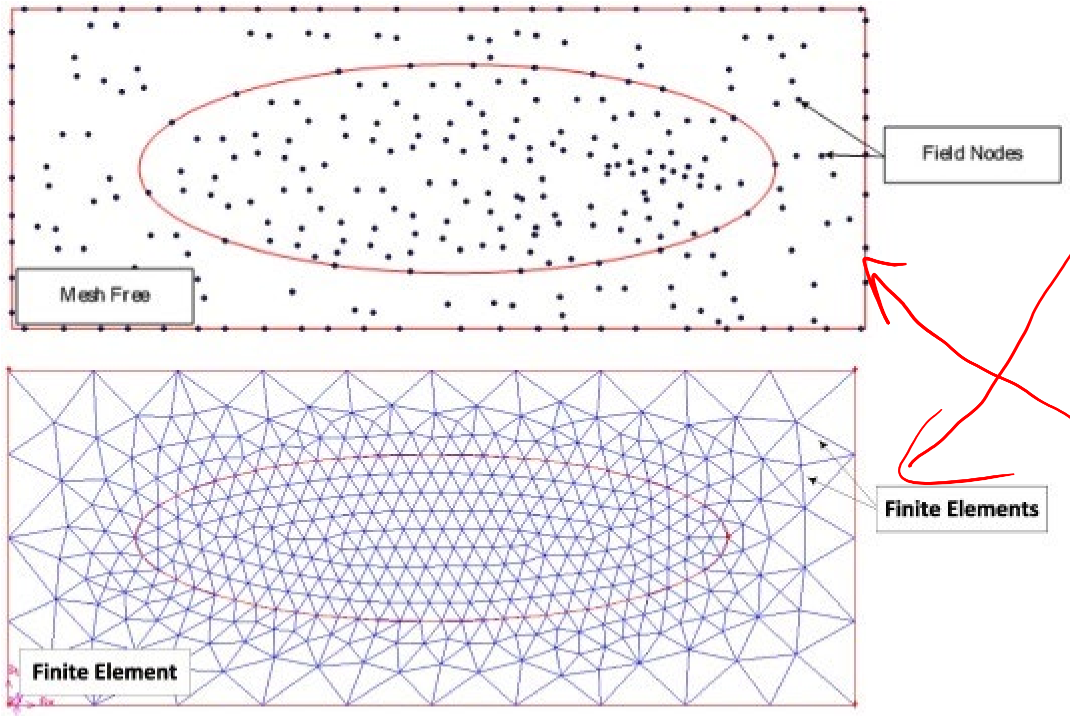
cell neighbors

Grid Type Summary

Grid Type	Topology	Geometry
Uniform	Structured	Structured
Rectilinear	Structured	Semi-structured
Curvilinear	Structured	Unstructured
Unstructured	Unstructured	Unstructured

- Geometry can structured or unstructured
- Topology can be structured or unstructured

Simulation Data: Mesh-Free Methods



Finite Element Methiod

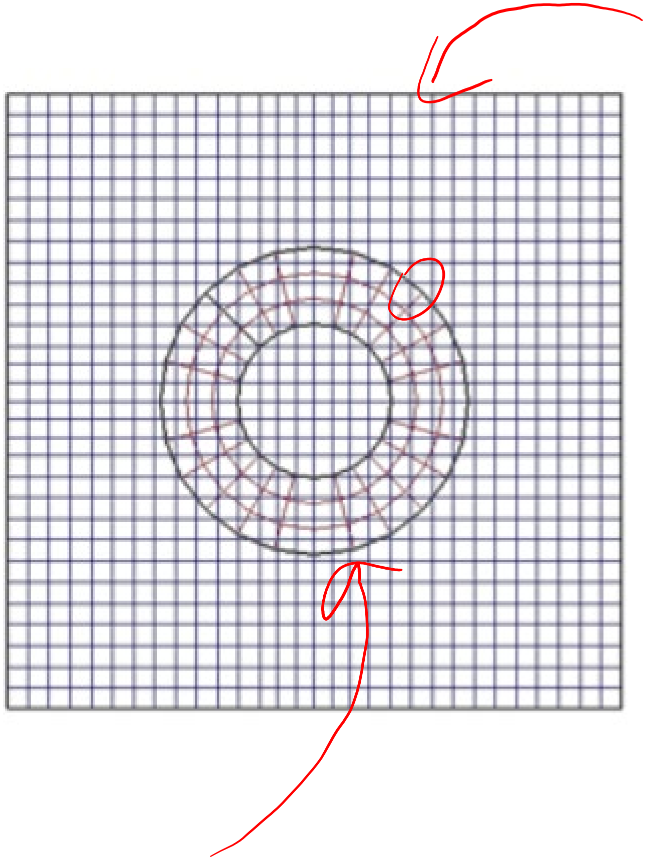
mesh-based method for solving partial differential equations

Mesh-free methods exist as well

Uses nodes and neighbor computations

Rendering solution may require meshing and interpolation

Simulation Data: Overset Meshes



- Used to solve partial differential equations
- Use two or more overlapping methods
- Data transfer between meshes via interpolation
- Challenging to visualize effectively
  - e.g. render multiple views