COLLEGE OF ENGINEERING
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES
UNIVERSITY OF MICHIGAN

Lecture 20
Mesh and Mesh Libraries

Prof. Brendan Kochunas
11/13/2019

NERS 590-004

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Outline

- Geometry and Mesh
- Mesh Formats and Libraries
- Meshing Tools
- CUBIT Tutorial
  - https://caen.engin.umich.edu/connect/windows-remote-desktop/

Why should I care about mesh?

- In engineering we typically have a governing equation that comes from physics.
  - These equations are most often some form of partial differential equation
- Being engineers we care about things out in the "real world"
  - To solve our governing equations on real world representations numerically, we need some kind of representation of the real world thing.
- The discretization of PDE's usually involves discretizing in space.
  - As you can imagine discretization schemes for various equations are fairly similar

- Hard to solve a PDE without a mesh!
- Don't reinvent the wheel!

Today's Learning Objectives

- Understand differences of "geometry" and "mesh"
- Types of discretization methods for PDE's
- Become familiar with what libraries are out there
  - And how they relate to one another
- Become aware of what tools are available
  - Often these are the "front end" of the meshing library.
- Tutorial for basics of using CUBIT

Geometry and Mesh

Modeling and Simulation of Physical Processes

- Define the physical problem.
- Create a mathematical (PDE) model
  - Systems of PDEs, ODEs, algebraic equations.
  - Define initial and or boundary conditions to get a well-posed problem.
- Create a Discrete (Numerical) model
  - Discretize the domain -> generate the grid -> obtain discrete model
  - Solve the discrete system
- Analyze errors in the discrete system
  - Consistency, stability and convergence analysis.

Fundamental Concepts

Geometry

- Description of the real-world physical problem
    - Sizes/dimensions of physical objects in space
    - AND the description of the materials occupying that space
- The geometry in the simulation is often an *approximation* of the real world.

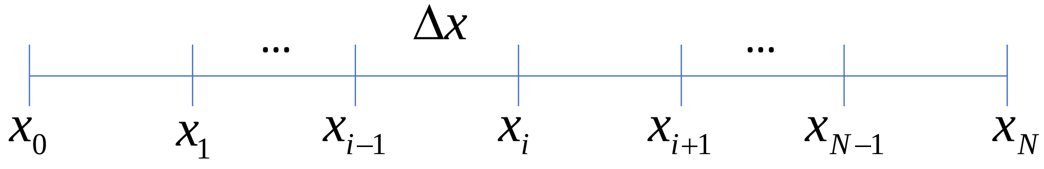
Mesh

- Discretized phase-space for dependent variable
    - Most commonly refers physical space
    - But, can also be applicable to other variables: speed, energy, time, etc.
- Type of mesh depends on the numerical method(s) used.
- Mesh & numerical method contain *additional approximations*

Geometry is *discretized* to produce a mesh

Types of Spatial Discretizations: Finite Difference

- From a spatial domain, represent solution at discrete points



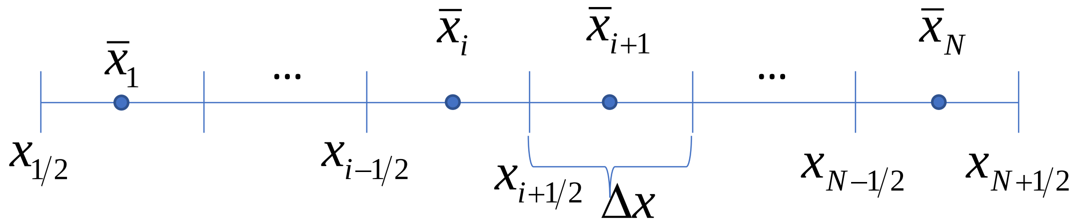
- Expressions can be derived from Taylor expansions about a point.
  - Discarded terms represent local truncation error.

$$f(x_i+\Delta x) = f(x_i) + \frac{df(x_i)}{dx}\frac{\Delta x}{1!} + \frac{d^2 f(x_i)}{dx^2}\frac{\Delta x^2}{2!}\ldots$$

- Derivatives represented as:
  - Forward $\dfrac{df(x_i)}{dx} \approx \dfrac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$
  - Backward $\dfrac{df(x_i)}{dx} \approx \dfrac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$
  - Central $\dfrac{df(x_i)}{dx} \approx \dfrac{f(x_{i+1}) - f(x_{i-1})}{2(x_{i+1} - x_{i-1})}$
- Resulting mesh are typically structured grids
  - Rectangular, triangular, etc.

Types of Spatial Discretizations: Finite Volume

- From a spatial domain, represent solution in discrete control volumes



- Makes use of conservation law for an integral over a finite volume

$$\bar{x}_i = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} f(x)dx$$

- Also typically applied to equations where the divergence theorem can be applied:

$$\int \nabla \cdot f(\vec{r})dV = \oint \vec{n} \cdot f(\vec{r})dS$$

- Resulting mesh may be arbitrary
- Methods typically require computing "fluxes" through surfaces between volumes

Types of Spatial Discretizations: Finite Element

- Solve "weak" formulations of equations

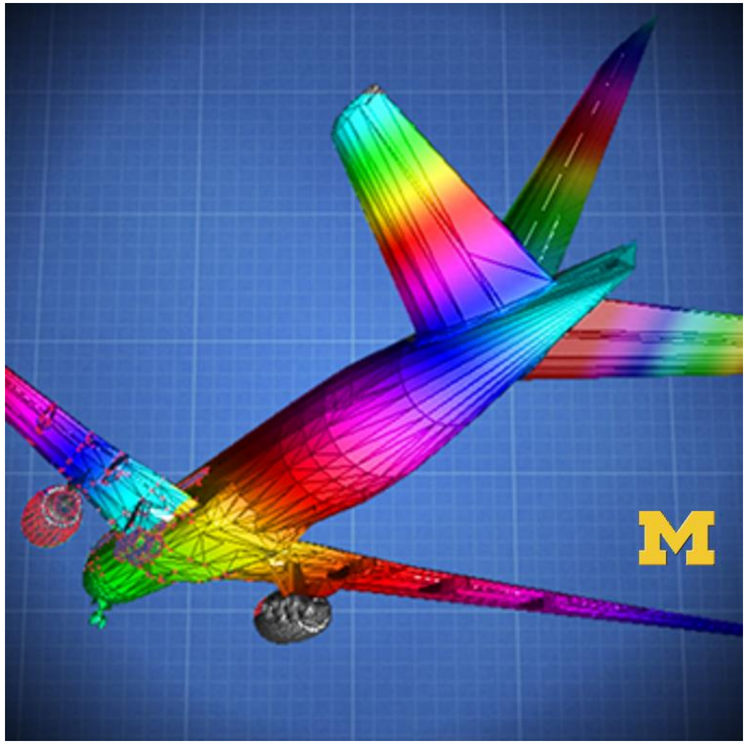
$$\frac{d^2 f}{dx^2} = q(x) \Rightarrow \int \frac{d^2 f}{dx^2}\xi(x)dx = \int q(x)\xi(x)dx$$

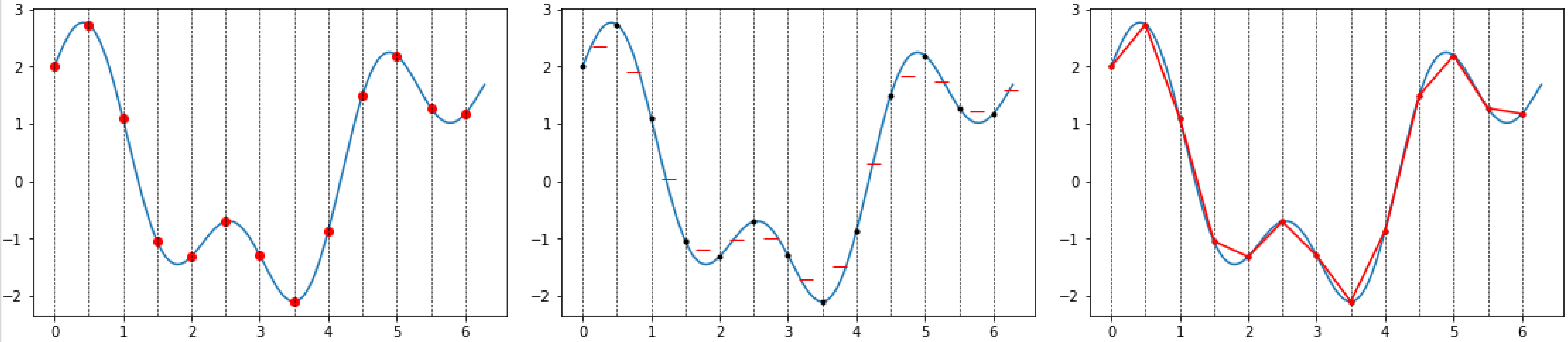
$$\int \frac{d^2 f}{dx^2}\xi(x)dx \Rightarrow -\int \frac{df}{dx}\frac{d\xi}{dx}(x)dx \equiv -\phi(f,\xi)$$

Multi-dimensional

$$\nabla^2 f(\vec{r}) = q(\vec{r}) \Rightarrow -\phi(f,\xi) \equiv \int_\Omega \nabla f(\vec{r})\cdot\nabla\xi(\vec{r})dS$$

- Use "basis" (or "trial") functions to represent "shape" of solution <u>in a discrete spatial cell.</u>
- Very widely used in structural analysis



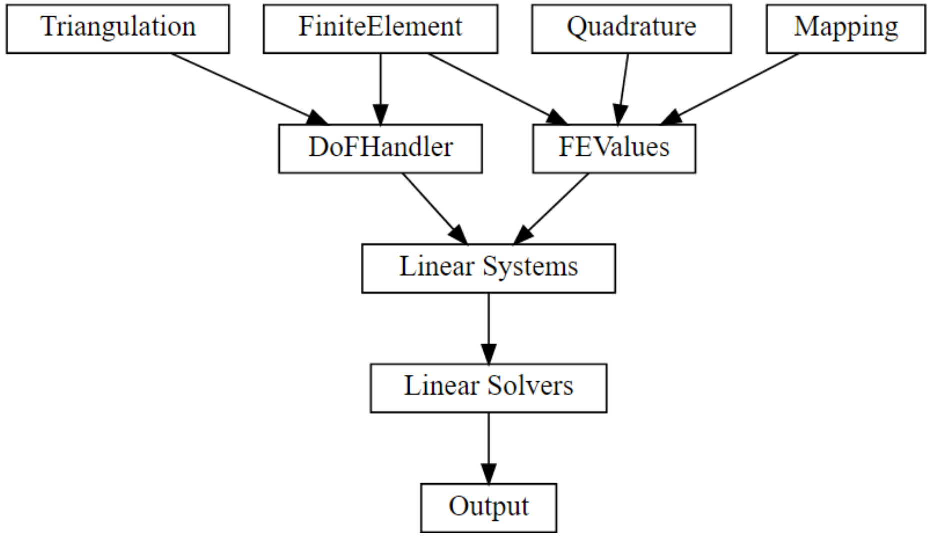


finite difference finite volume finite element.

Mesh Formats and Libraries

deal.II (https://www.dealii.org/)

- **D**ifferential **E**quations **A**nalysis **L**ibrary
  - (about 20 years old)
- Open source C++ FEA library.
- Interfaces to PETSc or Trilinos solvers
- Excellent translation support
  - Can process inputs from abaqus, unv, ucd, vtk, tecplot, netcdf CAD/IGES (OpenCASCADE)
- Lots well structured tutorials and videos



deal.II Architecture

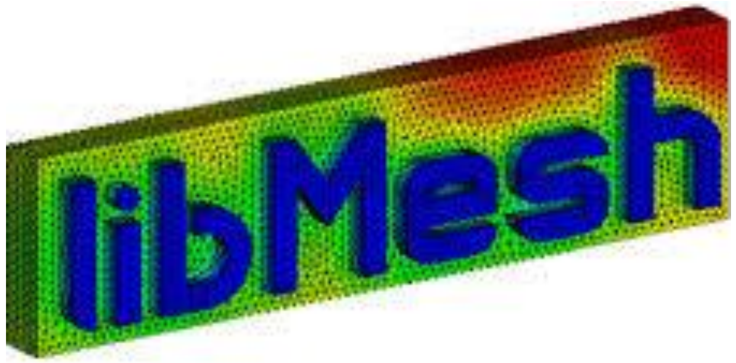
MFEM (https://mfem.org/)

- Modular Finite Element Methods
  - about 10 years old developed by DOE
- Designed for highly parallel computations
- Another C++ Library
- Interfaces to PETSc or HYPRE solvers
- Good translation support
  - Can process inputs from vtk, Gmsh, CUBIT (NetCDF)
- Good Tutorials
  - Lots of complex examples (miniapps)

Multi-mode Rayleigh-Taylor instability simulation using 4th order mixed elements in the MFEM-based BLAST shock hydrodynamics code. Visualization with VisIt.

libMesh (http://libmesh.github.io/)

- Another C++ FEA library.
- Provide support for adaptive mesh refinement (AMR) computations in parallel while allowing a research scientist to focus on the physics they are modeling.
- Interfaces to PETSc or Trilinos solvers
- Good translation support

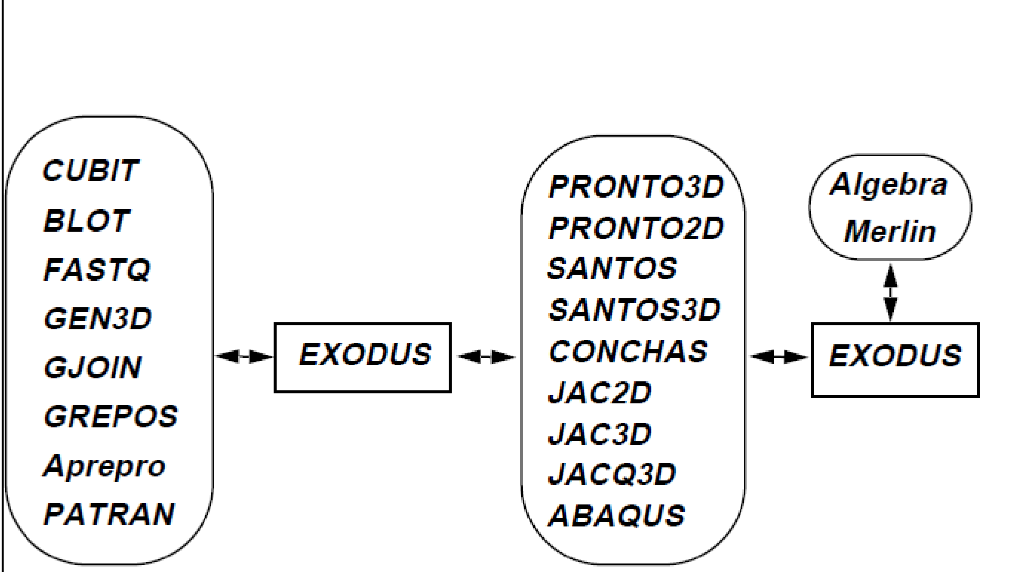


SEACAS/EXODUS

https://gsjaardema.github.io/seacas/html/index.html

- The Sandia Engineering Analysis Code Access System (SEACAS) is a suite of preprocessing, postprocessing, translation, and utility applications supporting finite element analysis software using the Exodus database file format.
  - Specifically for using Exodus II
  - Specifically for finite element analysis
- API supports C/C++ and Fortran

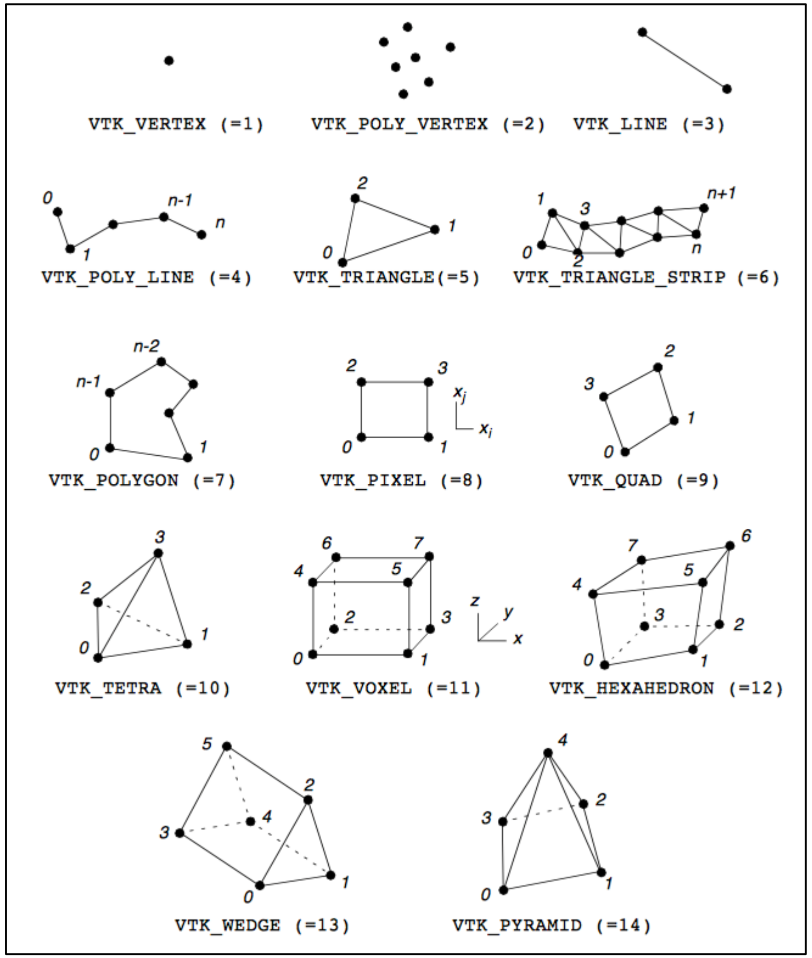
FIGURE 2. Expanded Modular View of SEACAS



Visualization ToolKit

(www.vtk.org)

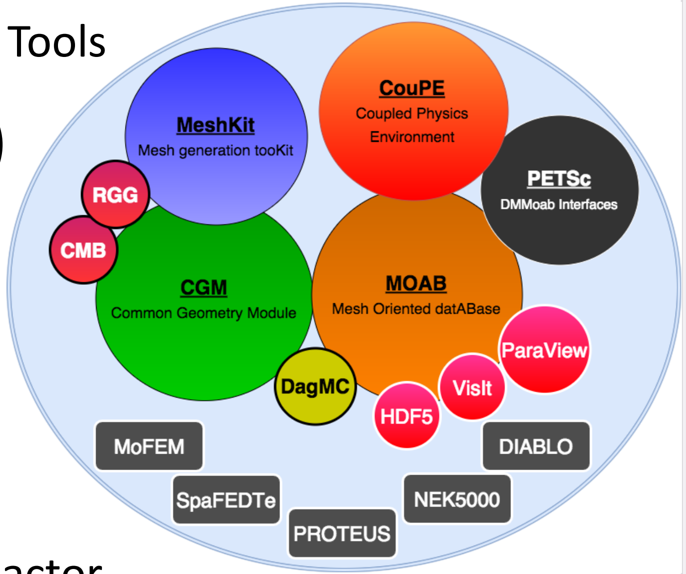
- Open source C++ library developed by Kitware (the Cmake people).
- Primarily a library for visualization
  - However core data model is very similar to mesh
  - Furthermore, previously discussed visualization tools that visualize work with the aforementioned mesh libraries
  - Consequently, there are several tools available that now how to go from VTK to a mesh or vice-versa.
- Does not presume discretization of type of numerical scheme



VTK primitives

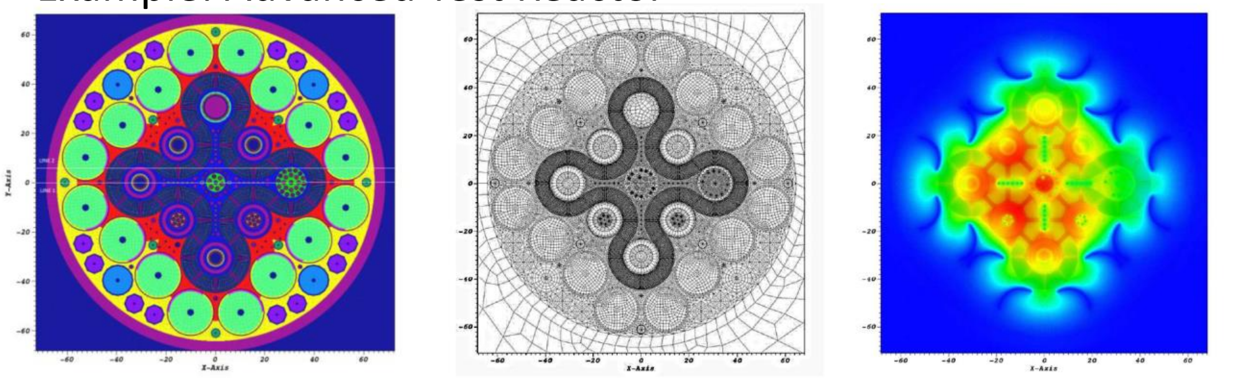
SIGMA Tools (sigma.mcs.anl.gov—deprecated?)

SIGMA Tools



- Scalable Interfaces for Geometry and Mesh based Applications
  - SIGMA provides interfaces and tools to access geometry data, create high quality unstructured meshes along with unified data-structures to load and manipulate parallel computational meshes for various applications to enable efficient physics solver implementations.
  - All libraries implemented in C++ and Python interfaces are provided.
- MeshKit – Provides meshing algorithms
- CGM – Common Geometry Model
  - Provides geometry functionality to other applications and mesh libraries. Includes some features not in solid modeling engines like decomposition tools
- MOAB – Mesh Oriented dAtaBase
  - library for representing unstructured and structured mesh, and field data on a mesh. Includes "Zoo" of finite elements
  - Facilitates solution transfer

Example: Advanced Test Reactor



Geometry/BC Setup — Generate unstructured mesh and link with solvers — Check-point/Analyze/Visualize

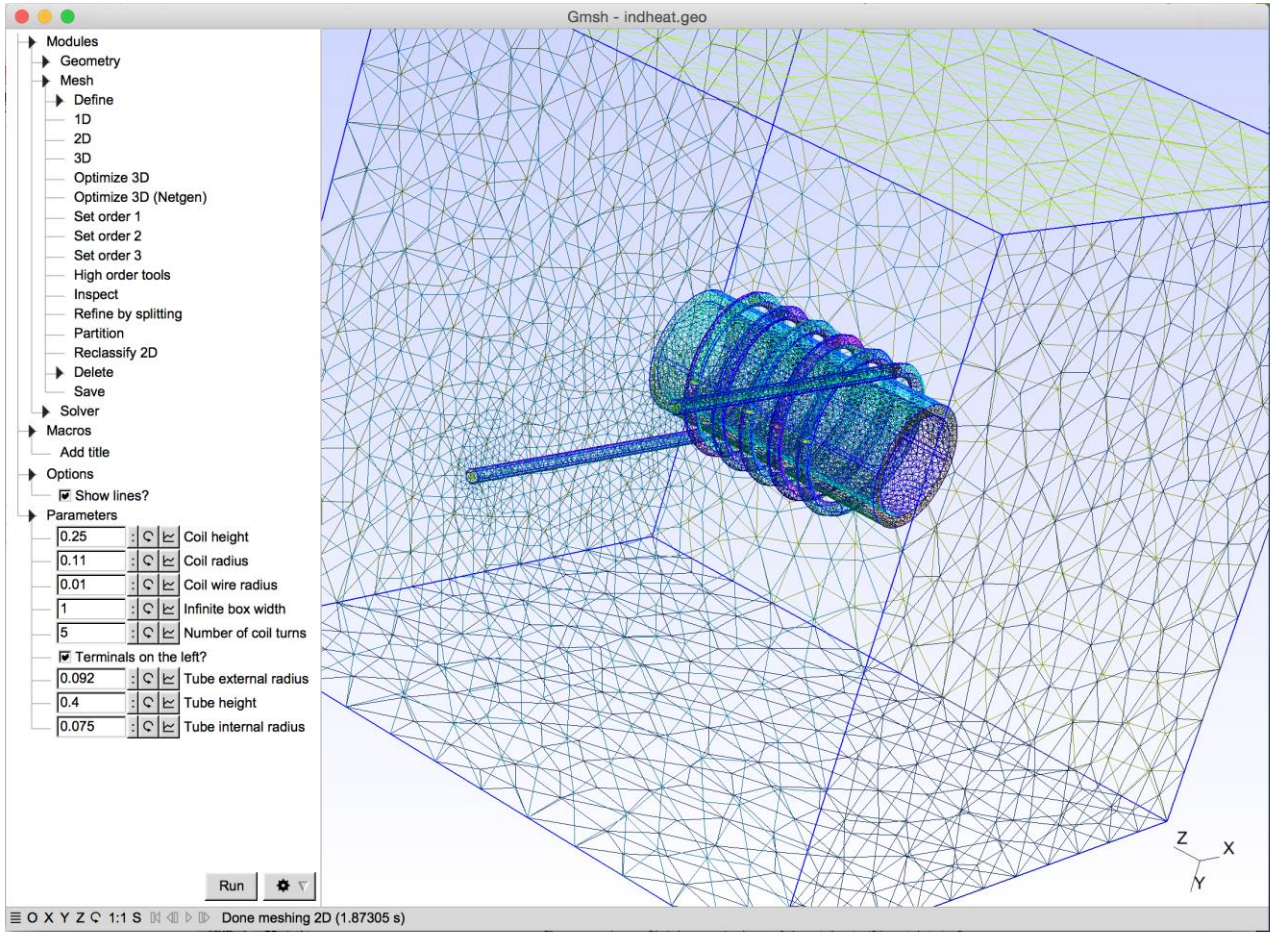
Which one is best for me?

- People seem to love writing open source finite element libraries
- If your group is already using one, go with it.
  - Unless…
- If starting new things to consider
  - What language am I programming in and does the library have an API to this language?
  - Is the API sufficient for my needs?
  - Do I find the API intuitive and/or easy to use?
  - Is there good tool support for pre- and post- processing?
  - Interoperability with commercial tools

Meshing Tools

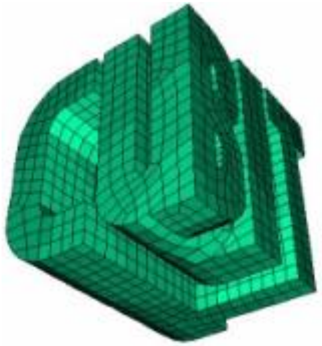
Gmsh (http://gmsh.info/)

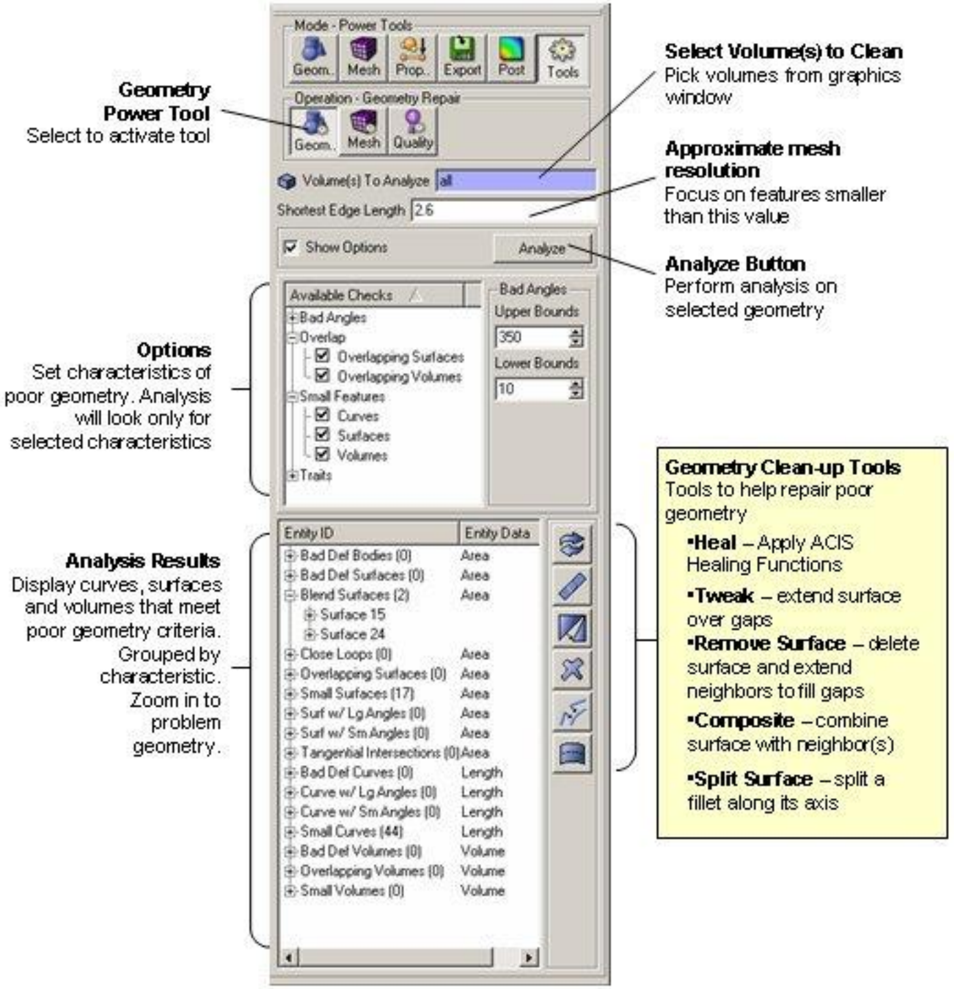
- Open Source GUI
  - Has good support in previously mentioned libraries
- Capabilitie
  - Built-in CAD engine (geometry)
  - Post-processing
  - Generating extruded, or repetitive meshes
  - Interface to external solvers via sockets (TCP/IP)
- Some drawbacks
  - Limited functionality in scripting and GUI



CUBIT (cubit.sandia.gov)

- CUBIT is a full-featured software toolkit for robust generation of two- and three-dimensional finite element meshes (grids) and geometry preparation.
  - Pretty much the only DOE sponsored open source meshing GUI.

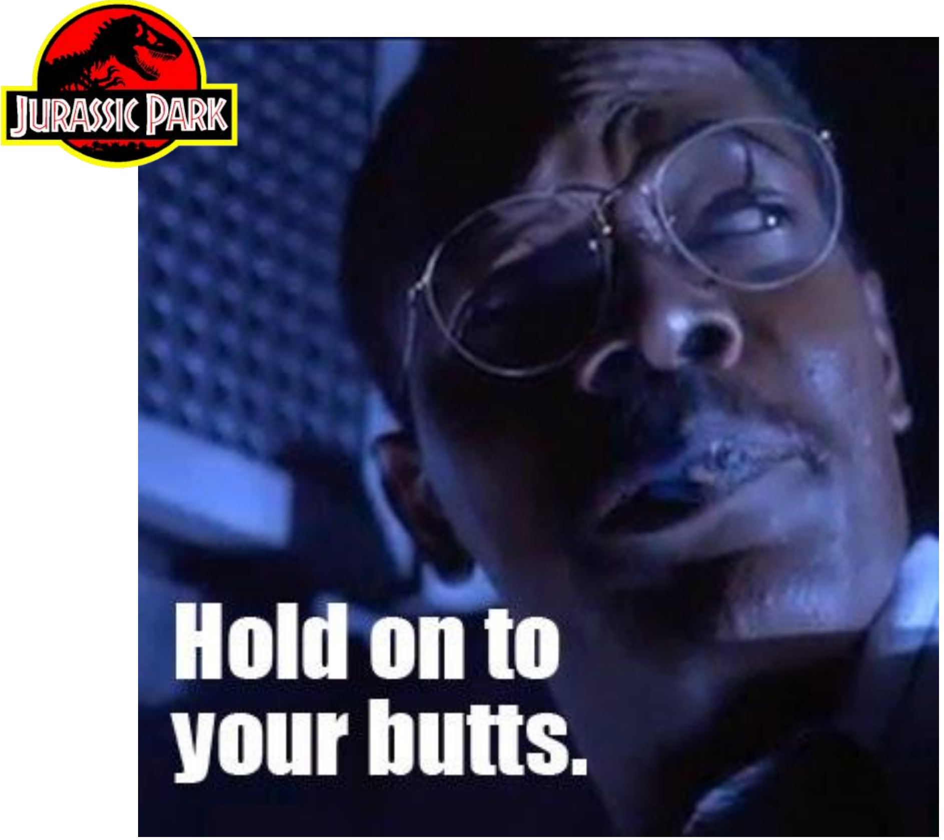




CUBIT Tutorial

"Hold on to your butts…"

- Arnold



https://cubit.sandia.gov/public/15.1/help_manual/WebHelp/step_by_step_tutorials/gui/overview.htm