Meshing and Solvers

COMSOL Day Orange County

Andy Cai

Scope of the Minicourse

- Meshing Basics
- How do we improve mesh quality?
- Overview of the common numerical algorithms used
  - The Finite Element Method
  - Stationary, Nonlinear, and Time Dependent Solvers

Why Do We Need Meshing?

The Modeling Workflow

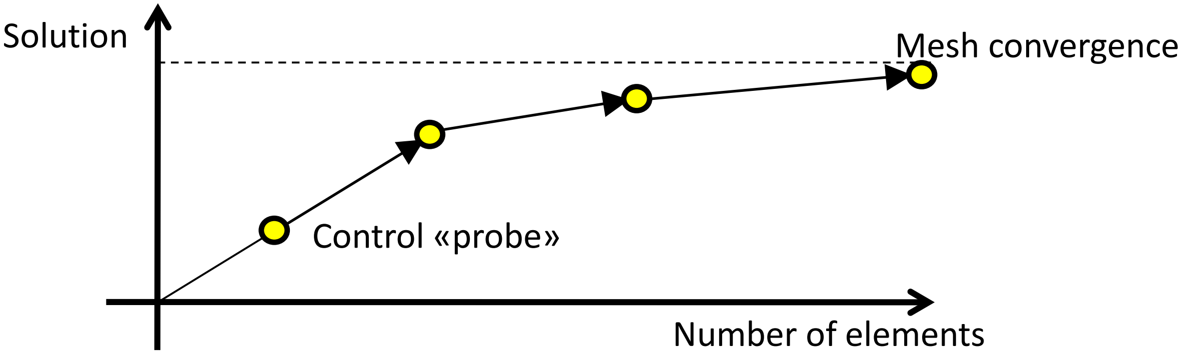
Geometry

Mesh

Results

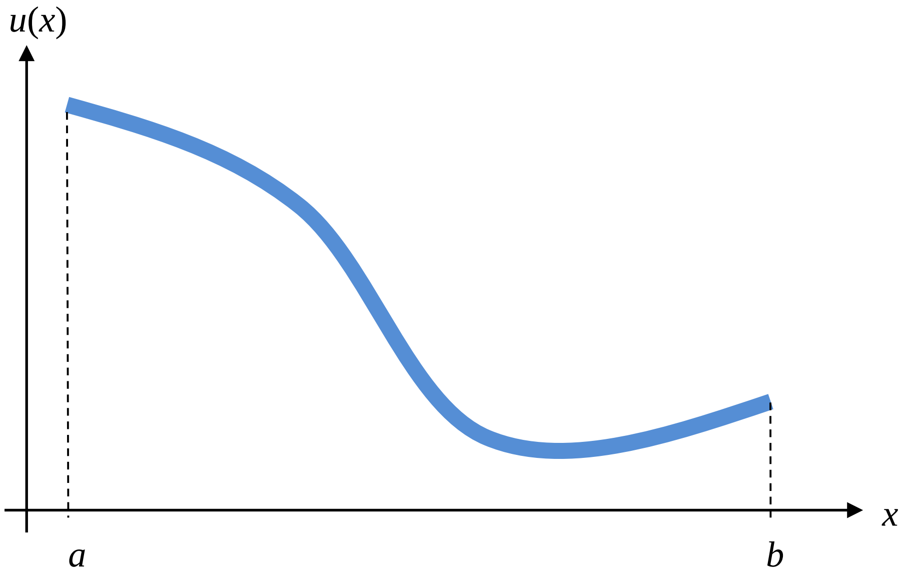
How Many Elements Do We Need?

- Very rarely known at the beginning
- Fine enough to map the geometry adequately
- Fine enough to resolve all gradients of the solution

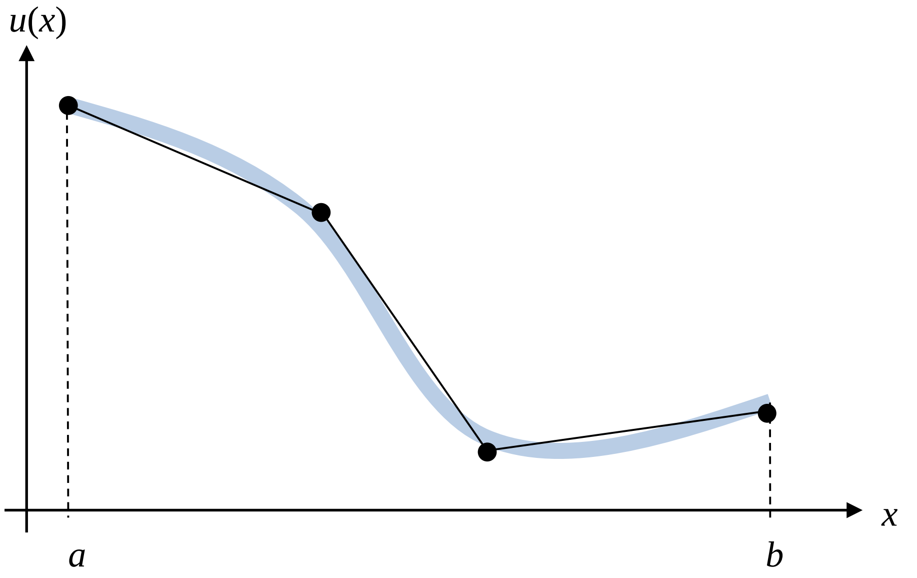


COMSOL

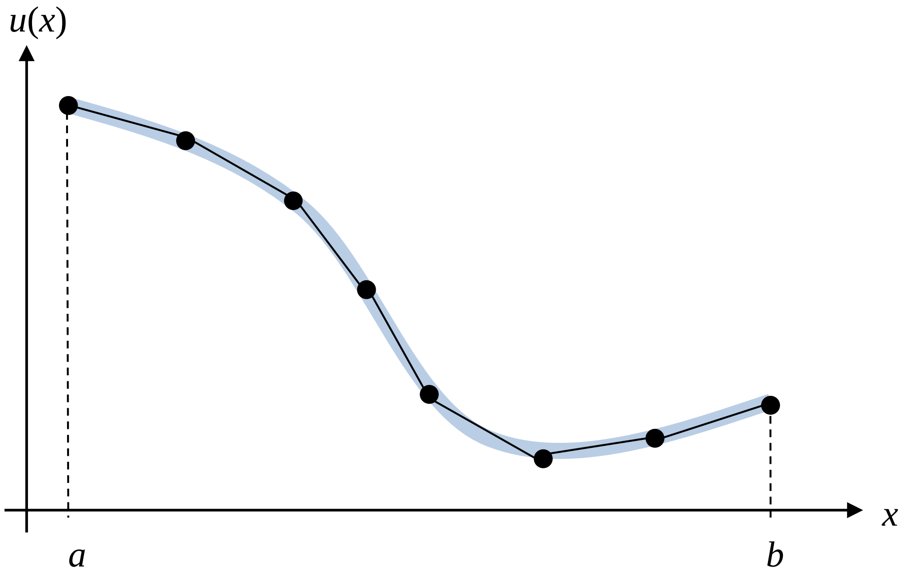
How Many Elements Do We Need?



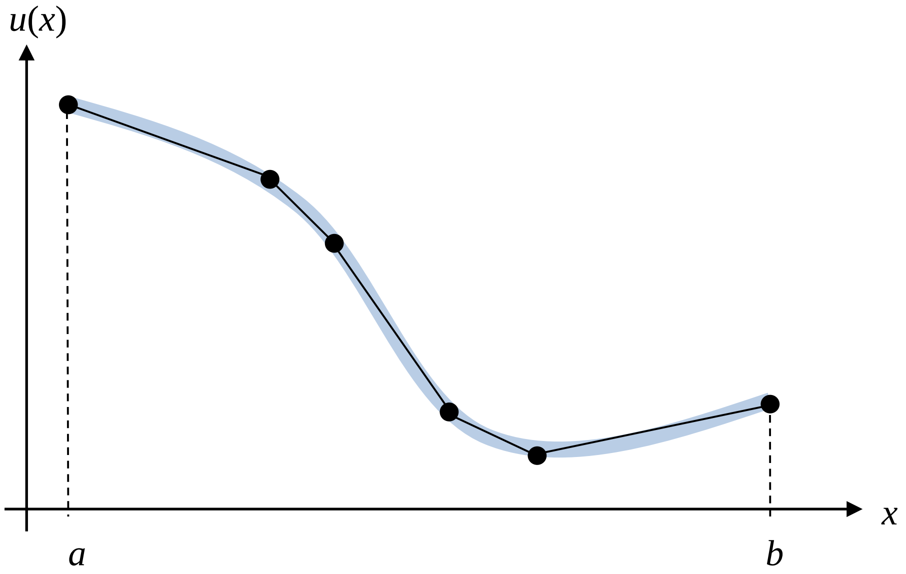
Elements Discretize the Solution



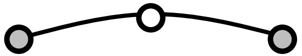
Finer Elements Decrease the Error



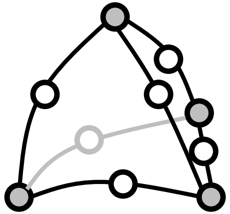
A Good Mesh Optimizes the Distribution of Elements



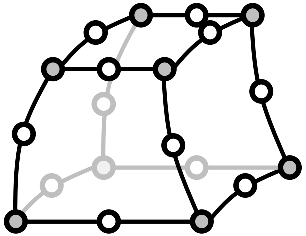
Available Elements



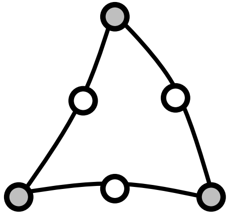
1D Line Element



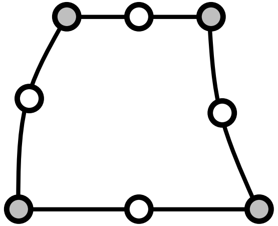
Tetrahedral



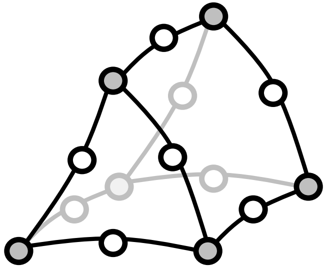
Hexahedral (Brick)



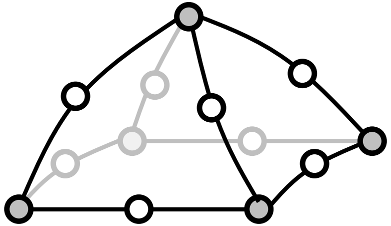
Triangular



Quadrilateral

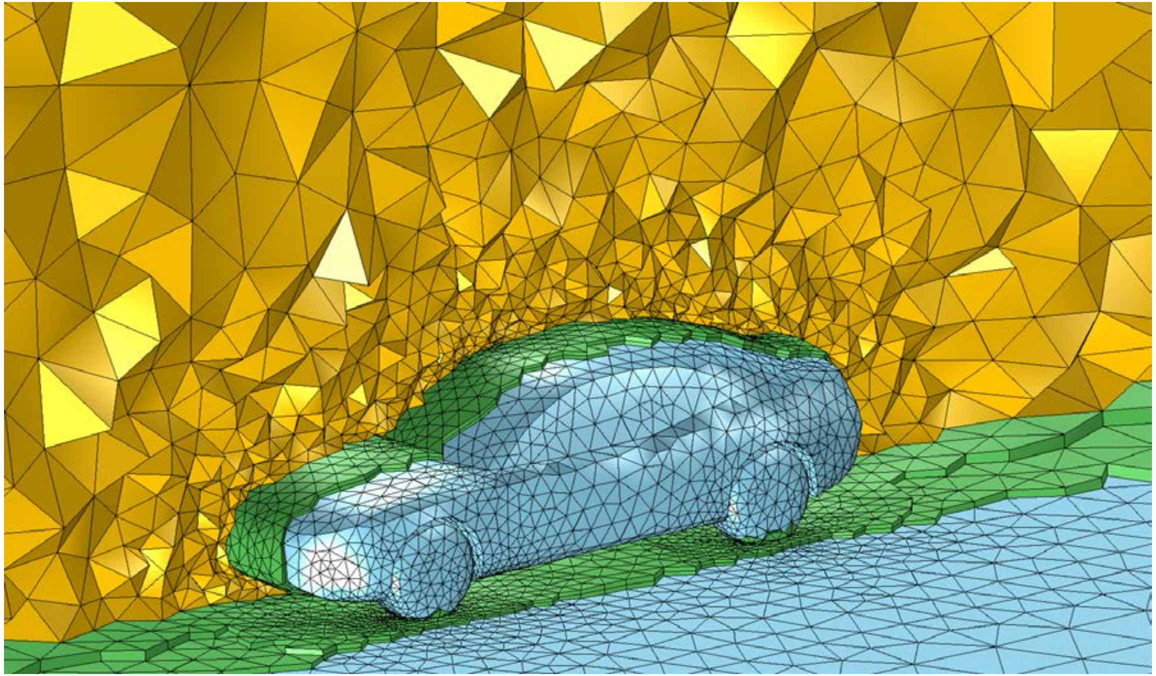


Prism



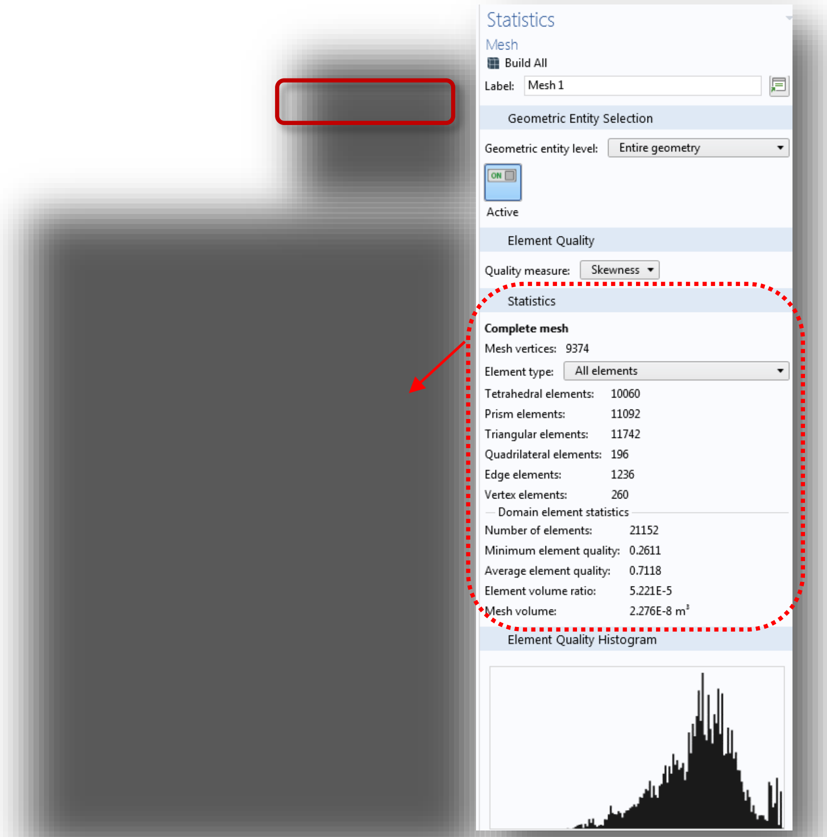
Pyramid

Physics-Controlled Mesh vs. User-Controlled Mesh



Mesh Statistics

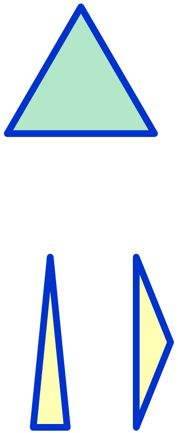
- The *Statistics* window includes information about:
  - The minimum and average mesh element quality
  - Element quality histogram



Mesh Quality

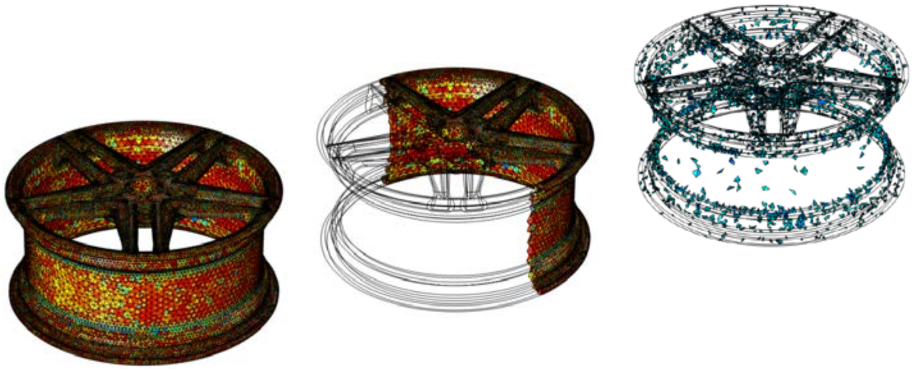
- Measures the regularity of the mesh elements' shape
- Low mesh resolution
  - Can lead to inaccurate results
- Low mesh quality
  - Can lead to inverted mesh elements
  - Can cause convergence issues
- Value between 0 and 1
  - min(q) > 0.3 in 2D
  - min(q) > 0.1 in 3D

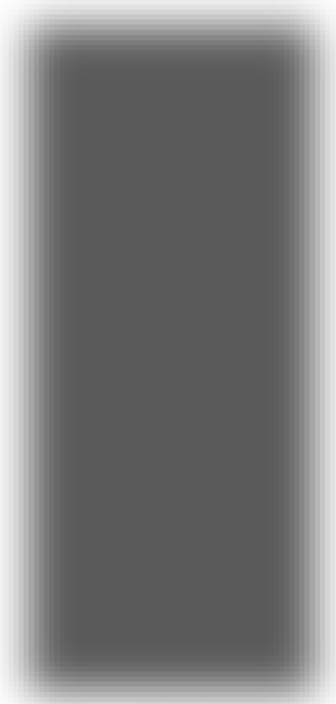
quality = 1 if element is equilateral



Mesh Visualization

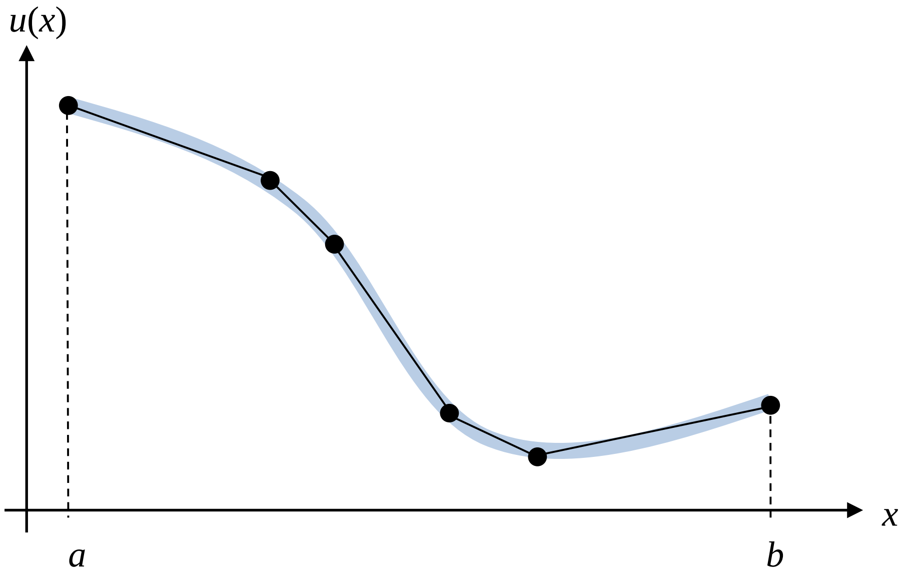
- Mesh plot
  - Plot various element types separately
  - Color elements according to quality
  - Show elements based on logical expressions
  - Shrink elements for better visualization



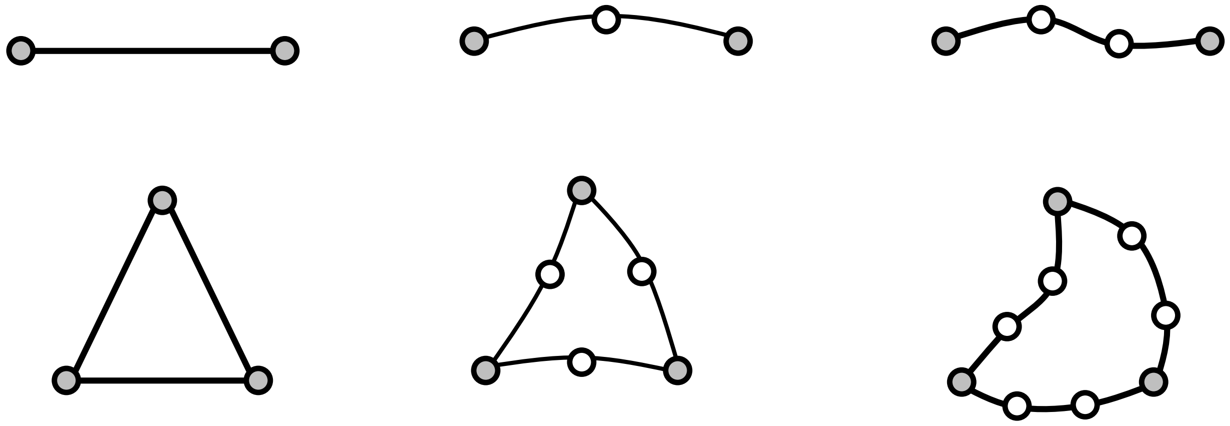


COMSOL

What Else Can We Do to Reduce the Error?

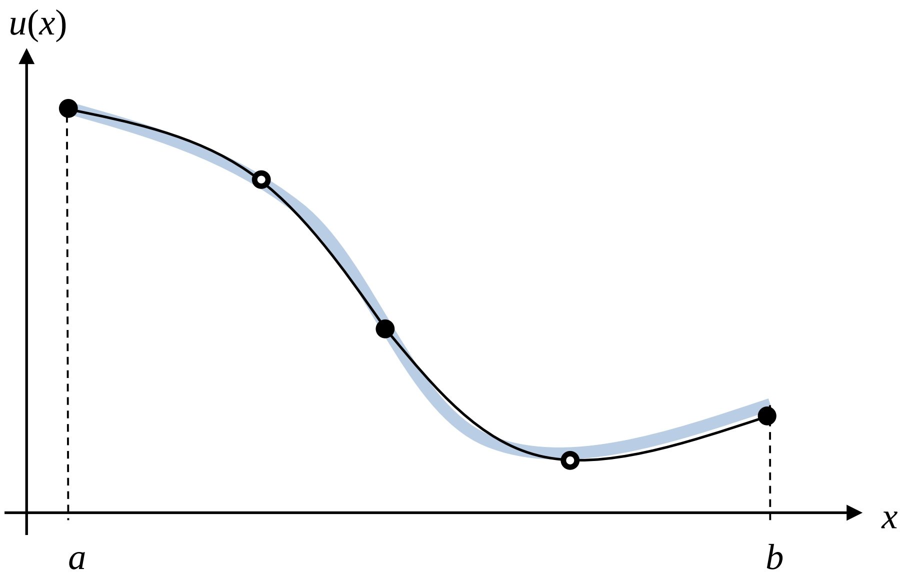


Each Element Can Have a Different Order

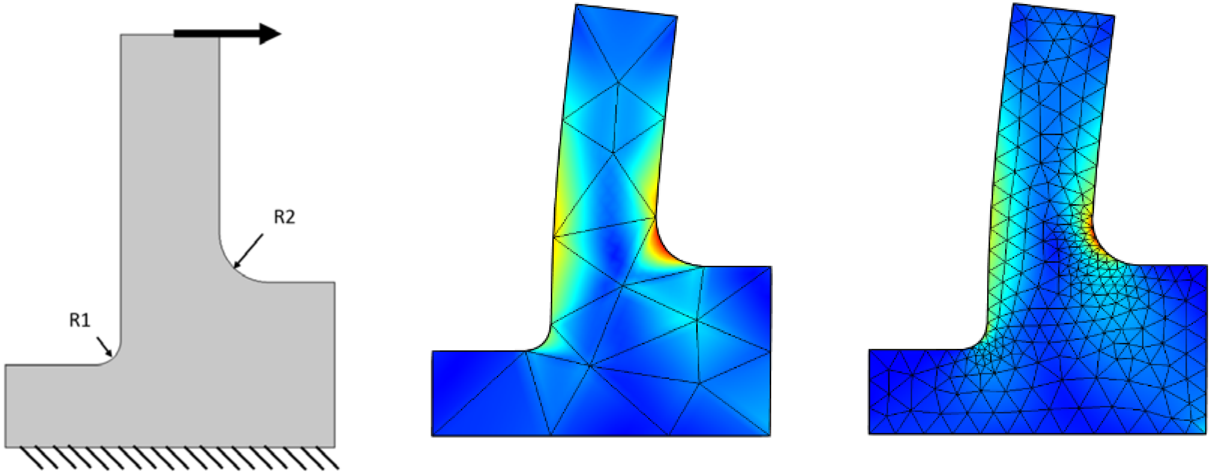


1st order   2nd order   3rd order

Only Need 2 High-Order Elements to Resolve the Solution



Adaptive Mesh Refinement



COMSOL

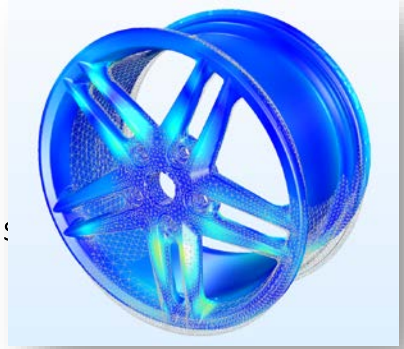
Numerical Methods in COMSOL Multiphysics

- Finite element method (FEM)
  - Most single physics and multiphysics simulations
- Boundary element method (BEM)
  - Certain electrostatics, magnetostatics, corrosion, and acoustics simulations
- Discontinuous Galerkin (DG)
  - Certain acoustics and electromagnetic wave propagation simulations
- Finite volume method (FVM)
  - Certain plasma and semiconductor simulations
- Particle tracing and ray tracing methods
- Some combinations of the above methods

Numerical Methods in COMSOL Multiphysics

Focus of this minicourse is mostly about solvers for FEM

- Finite element method (FEM)
  - Most single physics and multiphysics simulations
- Boundary element method (BEM)
  - Certain electrostatics, magnetostatics, corrosion, and acoustics simulations
- Discontinuous Galerkin (DG)
  - Certain acoustics and electromagnetic wave propagation simulations
- Finite volume method (FVM)
  - Certain plasma and semiconductor simulations
- Particle tracing and ray tracing methods
- Some combinations of the above methods



A FEM solution based on a tetrahedral mesh.

COMSOL

Going From Continuum to Discrete

- Given a heat transfer, structural, acoustics, electromagnetic, CFD, or chemical problem, typically stated as a partial differential equation (PDE), for heat transfer we got the heat equation

$$\rho C \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = Q$$

- We want to approximate this continuous equation with a system of linear equations
  - Approximate the continuum T with finite-sized vector u
  - For now: assume linear and ignore boundary conditions

$$D \frac{du}{dt} + Ku = b$$

- Where u and b are N-by-1 vectors and D and K are N-by-N matrices
- N =degrees of freedom (DOFs)

Stationary vs. Time Dependent

- A time-dependent problem gives rise to a system of ordinary differential equations (ODEs)

$$\rho C \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = Q \qquad \rightarrow \qquad D \frac{du}{dt} + Ku = b$$

- A stationary problem gives rise to a system of linear equations

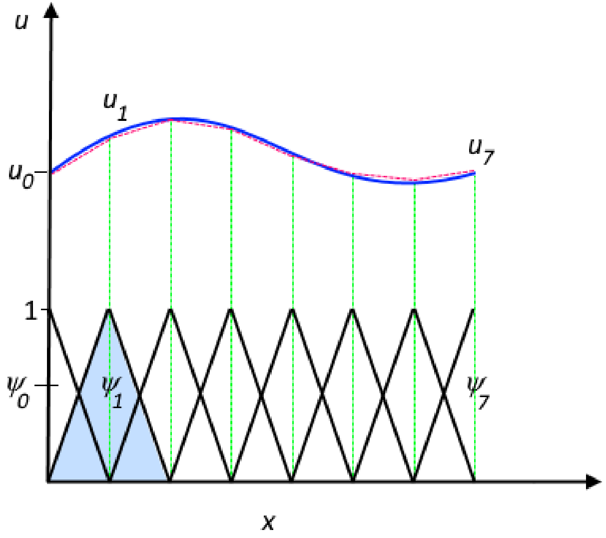
$$-\nabla \cdot (k \nabla T) = Q \qquad \rightarrow \qquad Ku = b$$

- Turns out it is good to know how to solve the stationary problem in order to solve the time-dependent problem, in general

How to Approximate: The Finite Element Method (1D)

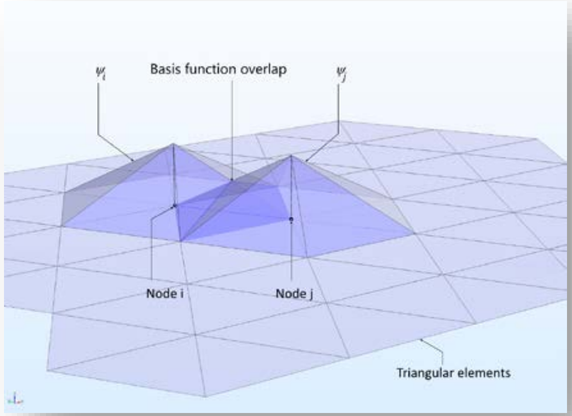
- Assume we know the solution u
  - In the picture: solid blue line
  - Has an infinite number of degrees of freedom
- Approximate with u_h
  - In the picture: dashed red line
  - Has a finite number of degrees of freedom
- u_h is a linear combination of linear basis functions ψ_i, represented by the solid black lines
- The coefficients are denoted by u_0 through u_7
- The finite element method is based on this type of approximation

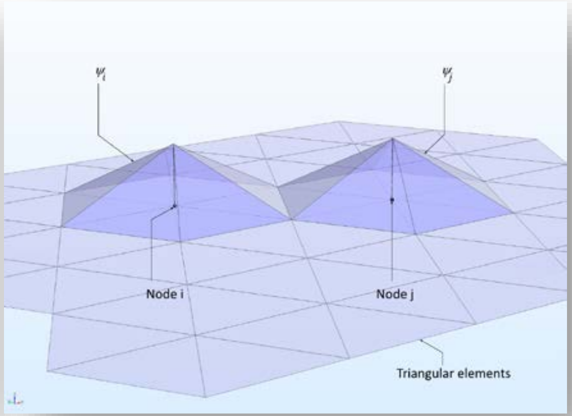
- *Note: linear has two meanings here: "linear basis functions" is unrelated to "linear problem": We can a) solve a nonlinear problem with linear basis functions or b) solve a linear problem with higher-order, nonlinear, basis functions*



Basis Functions (2D) and Sparse Matrices

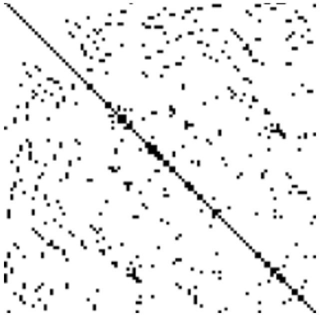
- Basic FEM uses tent-shaped linear basis functions that have a value of 1 at the corresponding node and zero on all other nodes
- Left picture: two base functions that share an element have a basis function overlap
- Right picture: two basis functions that share one element vertex but do not overlap in a 2D domain
- Lots of non-overlapping basis functions results in a *sparse* "stiffness" matrix K





$$-\nabla\cdot(k\nabla T)=Q \;\Rightarrow\; K\,\mathrm{u}=b$$

K={



}

Non-zero matrix entries are black

Solving the Linear Equation System

- How do we solve the system of linear equations $Ku = b$?
- By inverting? $u = K^{-1}b$
  - Doesn't work numerically
- Two main approaches
  - Direct solvers
    - Split K into factors so that $Ku = LUu = b$ and then solve in steps
  - Iterative solvers
    - For example $u_j = u_{j-1} + M(b - Ku_{j-1})$

Direct or Iterative Solvers

Direct	vs	Iterative
Most general-purpose and robust		Different physics have different recommended iterative solver types
Most memory-intensive		Uses least memory, *usually* faster
No feedback from solver		Non-convergence of iterative solver indicates a problem in the linear model

Winner? *Iterative* for most large 3D modeling, *Direct* for most 1D and 2D modeling. Reasonable default solvers are set up automatically.

Direct Solvers in COMSOL Multiphysics

- MUMPS
  - From a French consortium
  - Written for distributed (clusters) computation
  - Multithreaded parallel computation
  - Out-of-core
  - Partial pivoting makes it robust
- PARDISO
  - From Intel®
  - Written for multithreaded parallel computation
  - Out-of-core
  - No partial pivoting makes it less robust, but often faster, than MUMPS
- SPOOLES
  - Memory lean
  - Relatively slow

Iterative Linear Solvers

- Want to solve $Ku = b$ iteratively
- For example, iterate $u_j = u_{j-1} + P\ (b - Ku_{j-1})$
  where $P^{-1} = diag(K)$ are the diagonal entries of the matrix K
  until the error is below some tolerance: $error(u_j) < tol$
- The matrix P is called a preconditioner

COMSOL

Nonlinear Solvers

- Stationary nonlinear problem, for example

$$-\nabla \cdot (k\nabla T) = Q \quad \rightarrow \quad K(u)u = b \quad u{\sim}T$$

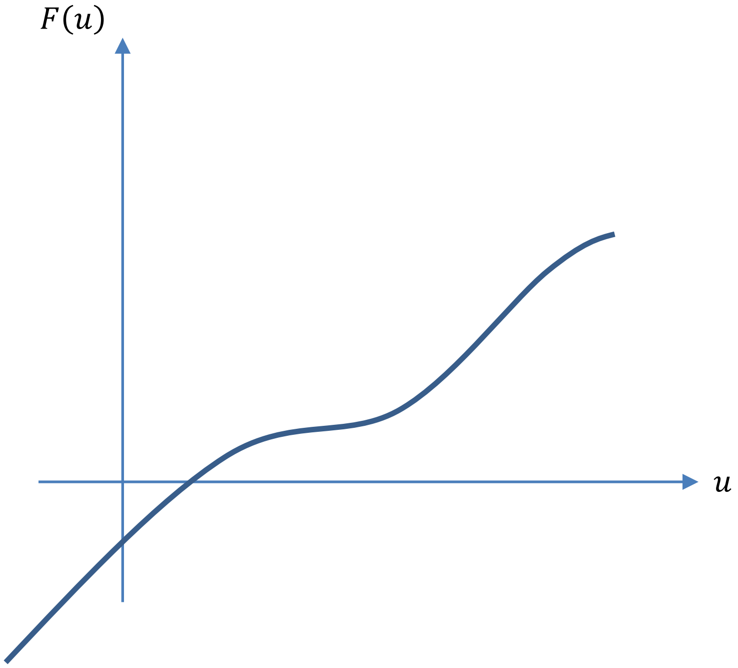
- The stiffness matrix K is here a function of the unknown field u
- Need to solve iteratively
  - Newton's method!
- Generalize:

$$K(u)u = b \quad \rightarrow \quad K(u)u - b = 0 \quad \rightarrow \quad F(u) = 0$$

- The equation form $F(u) = 0$ covers "all" physics
  - Apply Newton's method to this

Newton’s Method

- Graph of $F(u)$
- For example $F(u) = K(u)u - b$



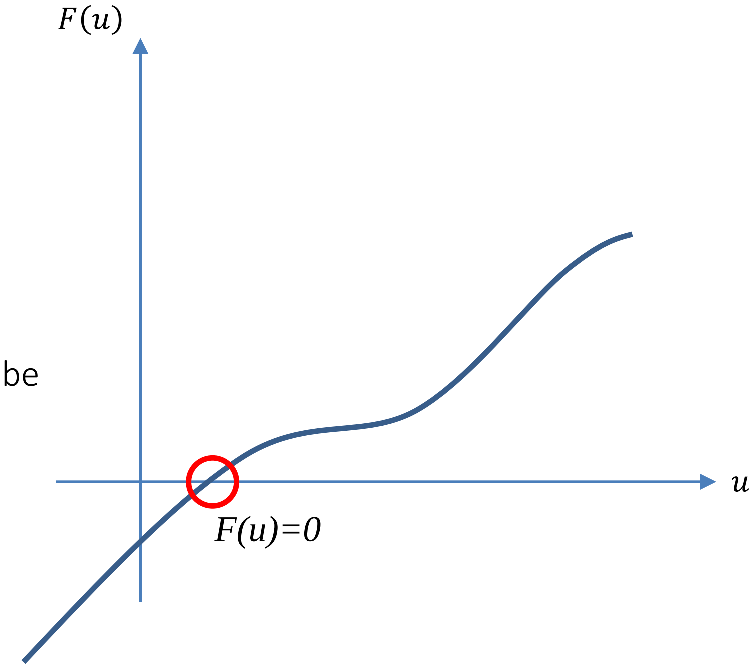
Newton's Method

- Want to find $F(u) = 0$
- For example $K(u)u - b = 0$

- For the interest
  - A solution $v \approx u$ such that

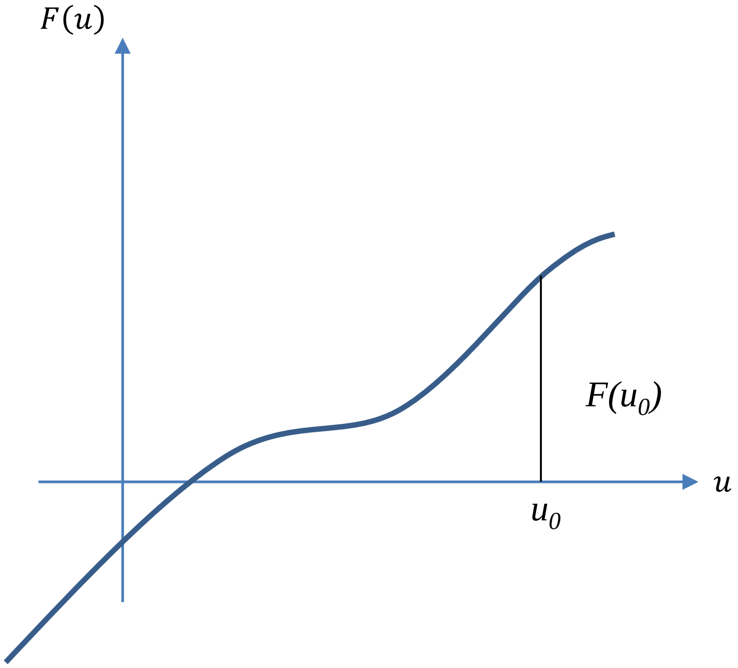
$$\|K(u)u - b\| < \delta$$
$$\|u - v\| < \varepsilon$$

where ε, δ are some small numbers may be good enough for us



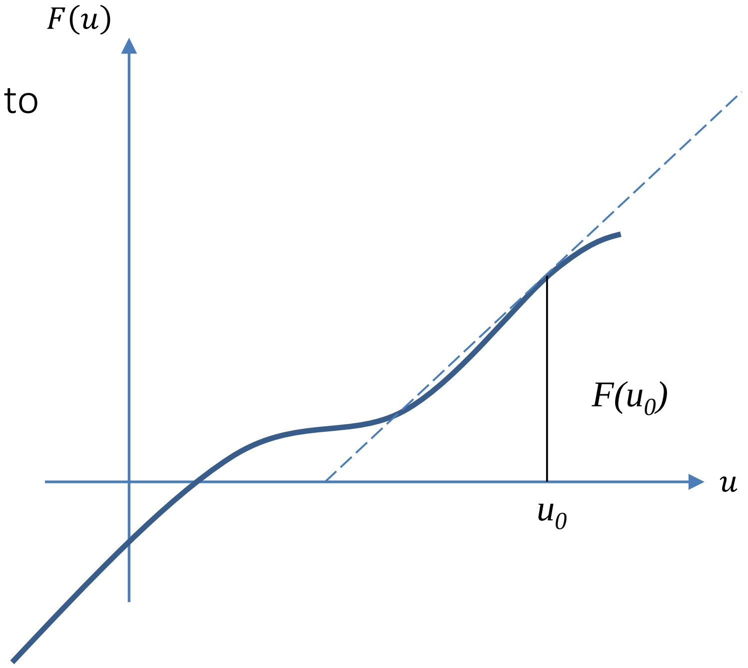
Newton’s Method

- Guess a solution $u = u_0$



Newton's Method

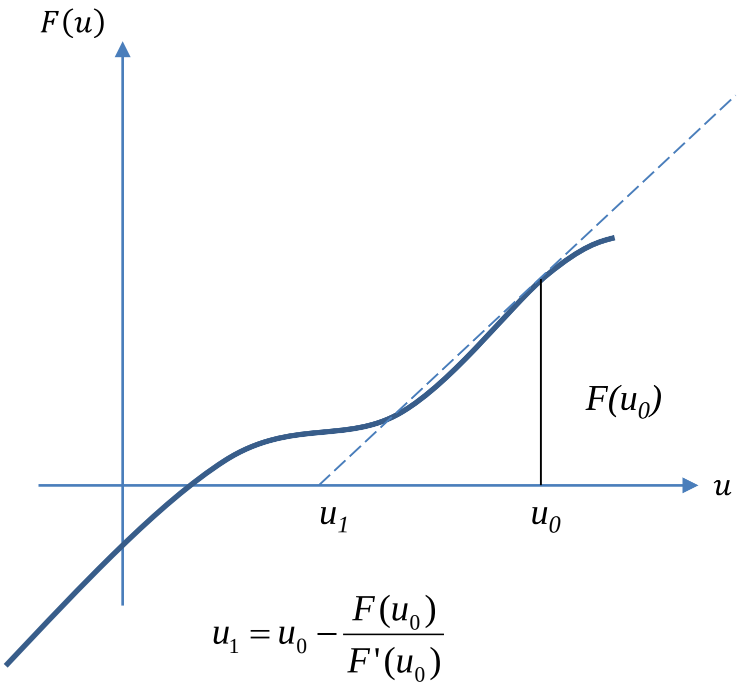
- Construct the tangent at $(u_0, F(u_0))$
- Hope that it intersect the u-axis closer to the solution $F(u)$



COMSOL

Newton’s Method

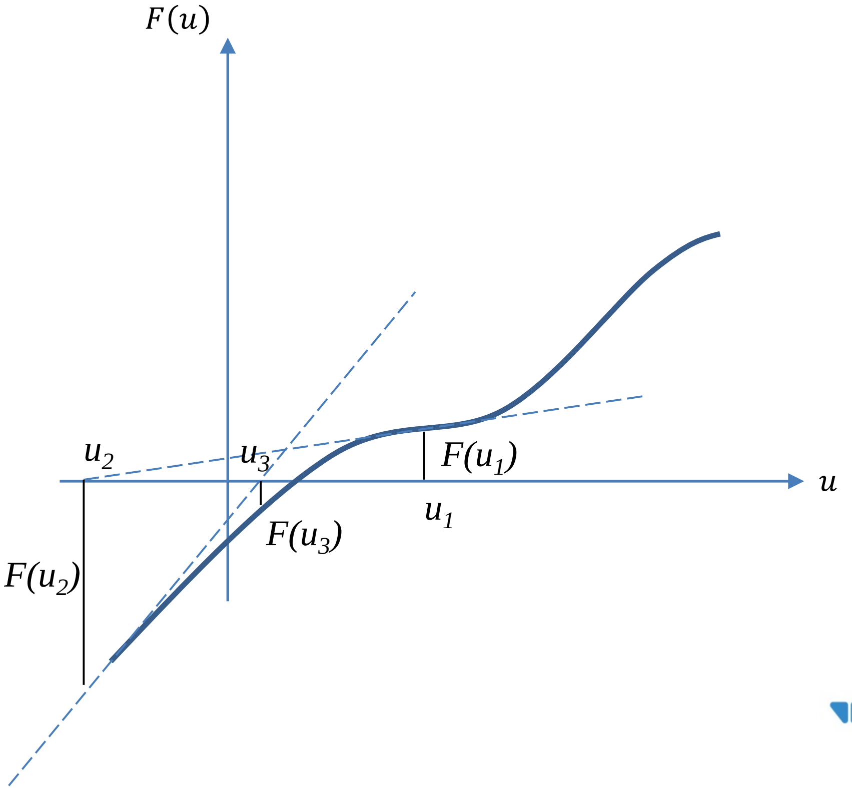
- Compute the intersection point
- Call it u_1



$$u_1 = u_0 - \frac{F(u_0)}{F'(u_0)}$$

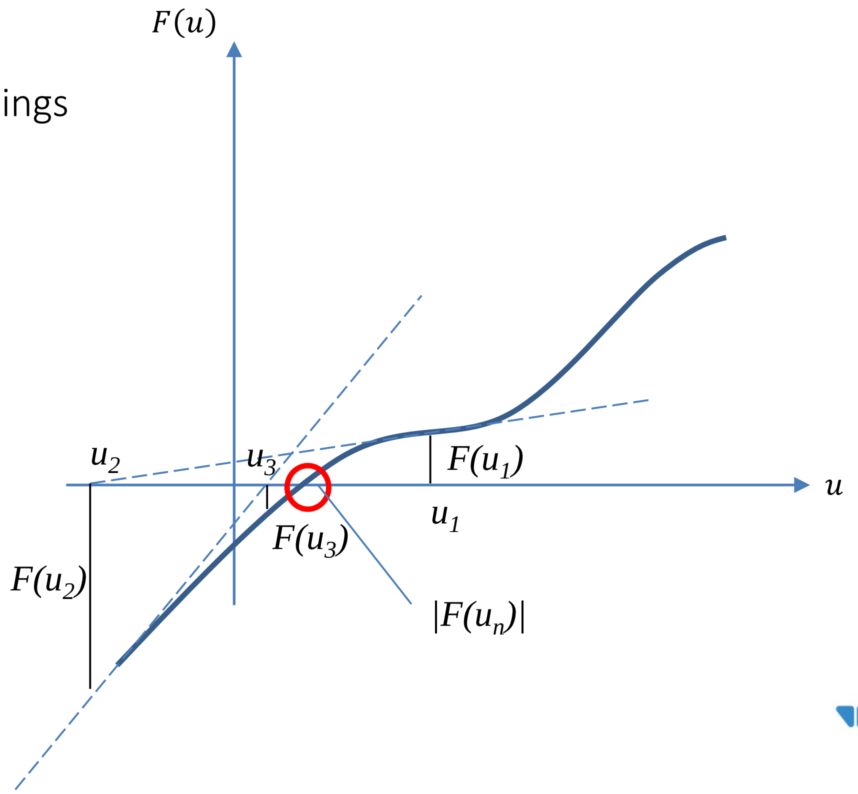
Newton’s Method

- Repeat …
- $u_n = u_{n-1} - \frac{F(u_{n-1})}{F'(u_{n-1})}$



Newton's Method

- … until $\mathrm{s}\|u_n - u_{n-1}\| < tol$
- Where s depends on multiple things
- Now $u = u_n$ is our solution



COMSOL

Newton's Method for Finite Elements

- The equation $$u_n = u_{n-1} - \frac{F(u_{n-1})}{F'(u_{n-1})}$$
  on matrix form $$u_n = u_{n-1} - [F'(u_{n-1})]^{-1} F(u_{n-1})$$

- Setting $$\Delta u_n = u_n - u_{n-1}$$
  gives $$-F'(u_{n-1})\Delta u_n = F(u_{n-1})$$
  where $F'(u)$ is the Jacobian matrix
- In the linear case $-F'(u) = K$ so that in each iteration we solve, again something similar to
  $$Ku = b$$
  but with Δu_n instead of u
- In practice, a damped method is used where: $$\Delta u_n = \gamma(u_n - u_{n-1})$$

COMSOL

When it Doesn’t Converge

- Newton’s method may not converge if, for example
  - “the starting guess is too far from the solution”
    - Use parametric solver (continuation method) or auxiliary solver
  - “something goes wrong with computing the tangent”
  - “ill-posed problem”
    - there is no solution: $F(u) = 0$ will never happen

COMSOL

Time Dependent Solvers

- Recall that the transient equation, for example

$$\rho C \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = Q$$

- Becomes

$$D \frac{du}{dt} + Ku = b \qquad u \sim T$$

where u and b are N-by-1 vectors and D and K are N-by-N matrices
N =degrees of freedom (DOFs)

Ordinary Differential Equations (ODEs)

- The equation system

$$D\frac{du}{dt} + Ku = b \qquad u \sim T$$

- is a system of ODEs.
- Solving
  - Discretize in space using, typically, finite elements
  - Solve in time by integrating the resulting system of ODEs: the “Method of Lines”
    - The system is typically implicit so in each step solve we need to solve a potentially nonlinear system with Newton’s method + the linear algebra solvers (direct or iterative)

The Time Dependent Solver can Solve Nonlinear Equations with Both 1st and 2nd Time Derivatives

$$\frac{1}{\rho c^2}\frac{\partial^2 p}{\partial t^2} + \nabla \cdot \left(\frac{1}{\rho}\nabla p\right) = Q_m$$

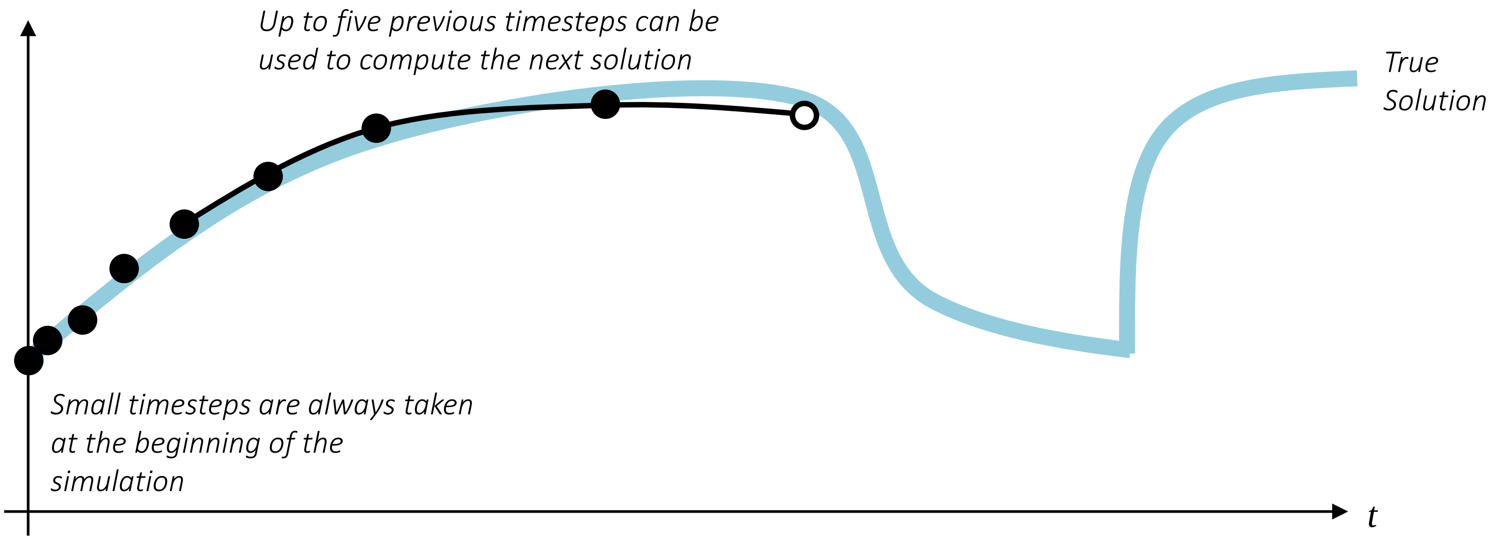
$$\rho C_p \frac{\partial T}{\partial t} + \nabla \cdot (-k(T)\nabla T) = Q$$

$$\mathbf{M}(\mathbf{u}, t)\frac{\partial^2 \mathbf{u}}{\partial t^2} + \mathbf{C}(\mathbf{u}, t)\frac{\partial \mathbf{u}}{\partial t} + \mathbf{K}(\mathbf{u}, t)\mathbf{u} - \mathbf{b}(\mathbf{u}, t) = 0$$

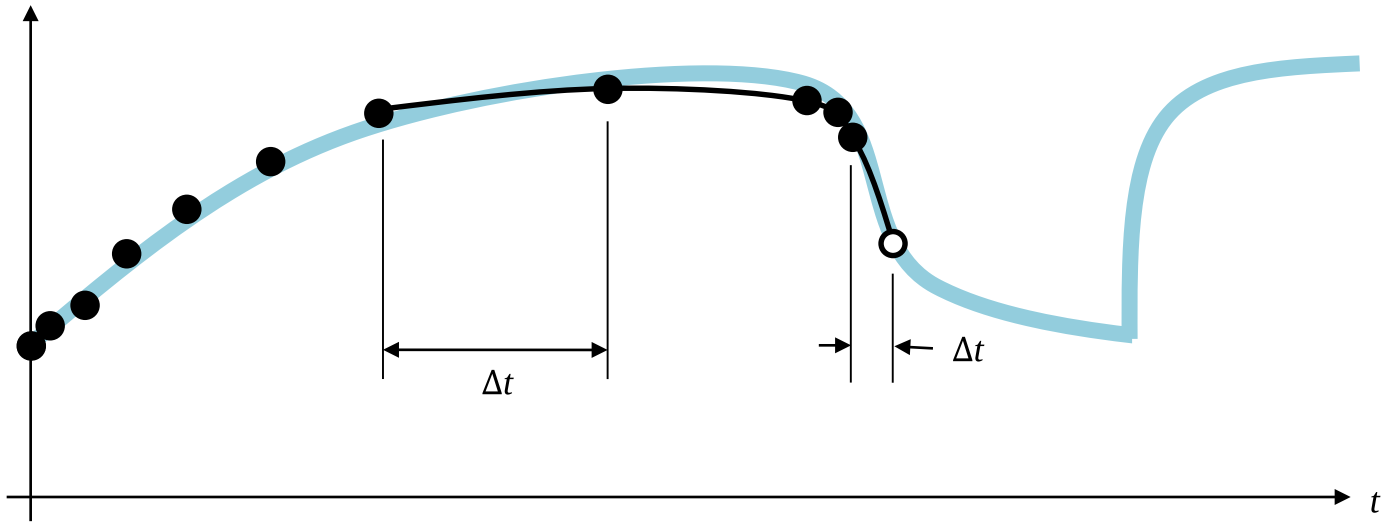
Solve an implicit set of nonlinear equations for future timesteps

$$\mathbf{L}\mathbf{u}_{t+1} = \mathbf{u}_t + \mathbf{b}$$

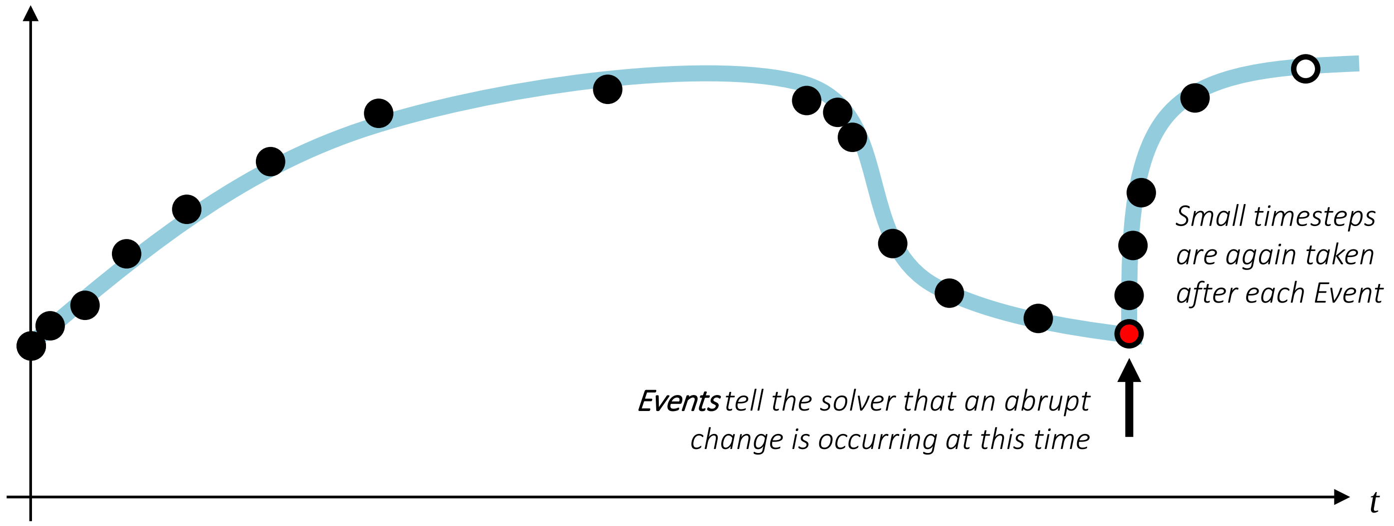
The Time Step is Automatically Computed to Satisfy the Specified Tolerance



Smaller Time Steps are Taken When the Solver Recognizes that the Solution is Varying Quickly



Instantaneous Changes, such as due to Changing Loads, are Handled via *Events*



Modeling Issues for the Time-dependent Solver

- Defining inputs (loads, properties) that vary too abruptly in time
  - Use Events if there are, in fact, abrupt changes
- Too loose a tolerance: Errors can accumulate over time
  - Verify a transient model by re-solving with finer tolerance
- Too coarse a mesh: Solution can vary abruptly in time and space
  - Re-solve with a finer mesh, and re-verify the tolerance study

Topics Not Covered

- Termination criteria and tolerances
- Segregated vs. Full Solver
- Adaptive Solver and Error Estimates
- Time-Stepping Algorithms
- Parametric Solver and Parametric Sweeps
- Auxiliary Solver

Q & A