Notes

- Assignment 2 is up

Modern FEM

- Galerkin framework (the most common)
- Find vector space of functions that solution (e.g. X(p)) lives in
  - E.g. bounded weak 1st derivative: H^1
- Say the PDE is L[X]=0 everywhere ("strong")
- The "weak" statement is ∫ Y(p)L[X(p)]dp=0 for every Y in vector space
- Issue: L might involve second derivatives
  - E.g. one for strain, then one for div sigma
  - So L, and the strong form, difficult to define for H^1
- Integration by parts saves the day

Weak Momentum Equation

- Ignore time derivatives - treat acceleration as an independent quantity
  - We discretize space first, then use "method of lines": plug in any time integrator

$$L[X] = \rho\ddot{X} - f_{body} - \nabla \cdot \sigma$$

$$\begin{aligned}
\int_\Omega Y L[X] &= \int_\Omega Y\left(\rho\ddot{X} - f_{body} - \nabla \cdot \sigma\right) \\
&= \int_\Omega Y\rho\ddot{X} - \int_\Omega Y f_{body} - \int_\Omega Y\nabla \cdot \sigma \\
&= \int_\Omega Y\rho\ddot{X} - \int_\Omega Y f_{body} + \int_\Omega \sigma\nabla Y
\end{aligned}$$

Making it finite

- The Galerkin FEM just takes the weak equation, and restricts the vector space to a finite-dimensional one
  - E.g. Continuous piecewise linear - constant gradient over each triangle in mesh, just like we used for Finite Volume Method
- This means instead of infinitely many test functions Y to consider, we only need to check a finite basis
- The method is defined by the basis
  - Very general: plug in whatever you want - polynomials, splines, wavelets, RBF's, …

Linear Triangle Elements

- Simplest choice
- Take basis $\{\phi_i\}$ where $\phi_i(p)=1$ at p_i and 0 at all the other p_j's
  - It's a "hat" function
- Then $X(p)=\sum_i x_i\phi_i(p)$ is the continuous piecewise linear function that interpolates particle positions
- Similarly interpolate velocity and acceleration
- Plug this choice of X and an arbitrary Y= ϕ_j (for any j) into the weak form of the equation
- Get a system of equations (3 eq. for each j)

The equations

$$\int_\Omega \phi_j \sum_i \rho \ddot{x}_i \phi_i - \int_\Omega \phi_j f_{body} + \int_\Omega \sigma \nabla \phi_j = 0$$

$$\sum_i \int_\Omega \rho \phi_j \phi_i \ddot{x}_i = \int_\Omega \phi_j f_{body} - \int_\Omega \sigma \nabla \phi_j$$

•Note that ϕ_j is zero on all but the triangles surrounding j, so integrals simplify
•Also: naturally split integration into separate triangles

Change in momentum term

- Let $m_{ij} = \int \rho \phi_i \phi_j$
- Then the first term is just $\sum_i m_{ji} \ddot{x}_i$
- Let $M=[m_{ij}]$: then first term is $M\ddot{x}$
- M is called the mass matrix
  - Obviously symmetric (actually SPD)
  - Not diagonal!
- Note that once we have the forces (the other integrals), we need to invert M to get accelerations

Body force term

- Usually just gravity: $f_{body}=\rho g$
- Rather than do the integral with density all over again, use the fact that ϕ_l sum to 1
  - They form a "partition of unity"
  - They represent constant functions exactly - just about necessary for convergence
- Then body force term is gM1
- More specifically, can just add g to the accelerations; don't bother with integrals or mass matrix at all

Stress term

- Calculate constant strain and strain rate (so constant stress) for each triangle separately
- Note $\nabla\phi_j$ is constant too
- So just take $\sigma\nabla\phi_j$ times triangle area
- [derive what $\nabla\phi_j$ is]
- Magic: exact same as FVM!
  - In fact, proof of convergence of FVM is often (in other settings too) proved by showing it's equivalent or close to some kind of FEM

The algorithm

- Loop over triangles
  - Loop over corners
  - Compute integral terms
    - only need to compute M once though - it's constant
  - End up with row of M and a "force"
- Solve Ma=f
- Plug this a into time integration scheme

Lumped Mass

- Inverting mass matrix unsatisfactory
  - For particles and FVM, each particle had a mass, so we just did a division
  - Here mass is spread out, need to do a big linear solve - even for explicit time stepping
- Idea of lumping: replace M with the "lumped mass matrix"
  - A diagonal matrix with the same row sums
  - Inverting diagonal matrix is just divisions - so diagonal entries of lumped mass matrix are the particle masses
  - Equivalent to FVM with centroid-based volumes

Consistent vs. Lumped

- Original mass matrix called "consistent"
- Turns out its strongly diagonal dominant (fairly easy to solve)
- Multiplying by mass matrix = smoothing
- Inverting mass matrix = sharpening
- Rule of thumb:
  - Implicit time stepping - use consistent M (counteract over-smoothing, solving system anyways)
  - Explicit time stepping - use lumped M (avoid solving systems, don't need extra sharpening)

Locking

- Simple linear basis actually has a major problem: locking
  - But graphics people still use them all the time…
- Notion of numerical stiffness
  - Instead of thinking of numerical method as just getting an approximate solution to a real problem,
  - Think of numerical method as exactly solving a problem that's nearby
  - For elasticity, we're exactly solving the equations for a material with slightly different (and not quite homogeneous/isotropic) stiffness
- Locking comes up when numerical stiffness is MUCH higher than real stiffness

Locking and linear elements

- Look at nearly incompressible materials
- Can a linear triangle mesh deform incompressibly?
  - [derive problem]
- Then linear elements will resist far too much: numerical stiffness much too high
- Numerical material "locks"
- FEM isn't really a black box!
- Solutions:
  - Don't do incompressibility
  - Use other sorts of elements (quads, higher order)
  - …

Quadrature

- Formulas for linear triangle elements and constant density simple to work out
- Formulas for subdivision surfaces (or high-order polynomials, or splines, or wavelets…) and varying density are NASTY
- Instead use "quadrature"
  - I.e. numerical approximation to integrals
- Generalizations of midpoint rule
  - E.g. Gaussian quadrature (for intervals, triangles, tets) or tensor products (for quads, hexes)
- Make sure to match order of accuracy [or not]

Accuracy

- At least for SPD linear problems (e.g. linear elasticity) FEM selects function from finite space that is "closest" to solution
  - Measured in a least-squares, energy-norm sense
- Thus it's all about how well you can approximate functions with the finite space you chose
  - Linear or bilinear elements: $O(h^2)$
  - Higher order polynomials, splines, etc.: better

Hyper-elasticity

- Another common way to look at elasticity
  - Useful for handling weird nonlinear compressibility laws, for reduced dimension models, and more
- Instead of defining stress, define an elastic potential energy
  - Strain energy density W=W(A)
  - W=0 for no deformation, W>0 for deformation
  - Total potential energy is integral of W over object
- This is called hyper-elasticity or Green elasticity
- For most (the ones that make sense) stress-strain relationships can define W
  - E.g. linear relationship: $W=\sigma:\varepsilon=\text{trace}(\sigma^T\varepsilon)$

Variational Derivatives

- Force is the negative gradient of potential
  - Just like gravity
- What does this mean for a continuum?
  - W=W(∂X/∂p), how do you do -d/dX?
- Variational derivative: $W_{total}[X+\varepsilon Y]=\int W\left(\frac{\partial X}{\partial p}+\varepsilon\frac{\partial Y}{\partial p}\right)$
  - So variational derivative is -∇•∂W/∂A
  - And f=∇•∂W/∂A
  - Then stress is ∂W/∂A
- Easy way to do reduced dimensional objects (cloth etc.)

$$\approx \int W\left(\frac{\partial X}{\partial p}\right)+\varepsilon\frac{\partial W}{\partial A}\frac{\partial Y}{\partial p}$$

$$= W_{total}+\varepsilon\int\frac{\partial W}{\partial A}\frac{\partial Y}{\partial p}$$

$$= W_{total}-\varepsilon\int Y\nabla\cdot\frac{\partial W}{\partial A}$$

Numerics

- Simpler approach: find discrete W_{total} as a sum of W's for each element
  - Evaluate just like FEM, or any way you want
- Take gradient w.r.t. positions $\{x_i\}$
  - Ends up being a Galerkin method
- Also note that an implicit method might need Jacobian = negative Hessian of energy
  - Must be symmetric, and at least near stable configurations must be negative definite