Project #2

assigned: Thursday, October 12th, 2017
due: Thursday, October 26th, 2017, 17:00
please hand in to one of the TAs in LEE N203

Review: tensor vs. vector notation

Since there is no straight-forward way to deal with higher-order tensors in code, we use **vector notation**. That is, here and in the following, our code will work with a displacement gradient *vector* and stress *vector*. For example, in 2D we convert tensors into vectors according to

$$\boldsymbol{\beta} = \nabla \boldsymbol{u} = \begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{pmatrix} \quad \to \quad \tilde{\boldsymbol{\beta}} = \begin{pmatrix} u_{1,1} \\ u_{1,2} \\ u_{2,1} \\ u_{2,2} \end{pmatrix}, \qquad \boldsymbol{\sigma} = \begin{pmatrix} \sigma_{1,1} & \sigma_{1,2} \\ \sigma_{2,1} & \sigma_{2,2} \end{pmatrix} \quad \to \quad \tilde{\boldsymbol{\sigma}} = \begin{pmatrix} \sigma_{11} \\ \sigma_{12} \\ \sigma_{21} \\ \sigma_{22} \end{pmatrix}.$$

The finite-deformation version is analogous with $\tilde{\boldsymbol{F}}$ and $\tilde{\boldsymbol{P}}$. The fourth-order incremental tangent tensor now becomes a simple matrix defined via

$$\tilde{\mathbb{C}} = \frac{\partial \tilde{\boldsymbol{\sigma}}}{\partial \tilde{\boldsymbol{\beta}}} \qquad \text{and, e.g.,} \quad \tilde{\mathbb{C}}_{11} = \mathbb{C}_{1111},\ \tilde{\mathbb{C}}_{12} = \mathbb{C}_{1112},\ \tilde{\mathbb{C}}_{13} = \mathbb{C}_{1113}, \text{etc.}$$

That is, according to the stress and strain vectors where indices are ordered as $\{(11),(12),(21),(22)\}$, the indices of $\tilde{\mathbb{C}}_{ij}$ (with $i,j = 1,\dots,4$ in 2D) stand for a pair of indices according to the above order.

For convenience, **utility functions** are available for the simple conversion between vector and standard notation. This way we can first define $\mathbb{C}_{ijkl}$ in the classical way and then convert to the above vector form. Of course, you are welcome to use those utility tools in your code.

Finally, recall that we had shown in class that (the finite-deformation version is analogous)

$$\sigma_{ij} = \frac{\partial W}{\partial (\nabla u)_{ij}} \quad \Leftrightarrow \quad \tilde{\sigma}_i = \frac{\partial W}{\partial \tilde{\beta}_i} \qquad \text{and} \qquad \mathbb{C}_{ijkl} = \frac{\partial \sigma_{ij}}{\partial (\nabla u)_{kl}} \quad \Leftrightarrow \quad \tilde{\mathbb{C}}_{ij} = \frac{\partial \tilde{\sigma}_i}{\partial \tilde{\beta}_j}.$$

Problem 1: material model, linear elasticity (30 points).

Let us implement the 3D isotropic linear elastic material model from Project #1, defined by the energy density

$$W(\nabla \boldsymbol{u}) = \frac{\lambda}{2}(\operatorname{tr}\boldsymbol{\varepsilon})^2 + \mu\,\boldsymbol{\varepsilon}\cdot\boldsymbol{\varepsilon} \qquad \text{with} \qquad \boldsymbol{\varepsilon} = \frac{1}{2}\left[\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\mathsf{T}}\right], \qquad \boldsymbol{\beta} = \nabla \boldsymbol{u}.$$

Let us implement the linear elastic material model as a C++ class that has the following components:

- Use `typedef` to define the vectors/matrices `DisplacementGradient`, `Stress`, `Strain`, and `TangentMatrix` as `Eigen::Matrix<double,?,?>` with their correct dimensions in 3D.

- The class constructor should receive two `doubles` as input (E and ν) and store those within the class as `private` members. (Recall the conversion between λ, μ and E, ν derived in Project #1.)
- The class should have a method `computeEnergy`, which receives the `DisplacementGradient` vector $\tilde{\beta}$ and returns the energy density W as a `double`.
- The class should have a method `computeStress`, which receives a `DisplacementGradient` vector $\tilde{\beta}$ and returns a `Stress` vector $\tilde{\boldsymbol{\sigma}}$.
- The class should have a method `computeTangentMatrix`, which receives a `DisplacementGradient` vector $\tilde{\beta}$ and returns a `TangentMatrix` $\tilde{\mathbb{C}}$.
- The class should have a method `computeStrain`, which receives a `DisplacementGradient` vector $\tilde{\beta}$ and returns the strain vector $\tilde{\varepsilon}$ as `Strain`.

Problem 2: material model, hyperelasticity (35 points).

Let us repeat the above implementation for the compressible Neo-Hookean material model from Project #1, which had the energy density

$$W(\boldsymbol{F}) = \frac{\mu}{2}(\operatorname{tr}\boldsymbol{C} - 3) + \frac{\lambda}{2}(\ln J)^2 - \mu \ln J.$$

Now working with $\tilde{\boldsymbol{\beta}}$, $\tilde{\boldsymbol{P}}$ and $\tilde{\mathbb{C}}$, let us implement this material model as a new class having the same methods as the linear elastic one implemented above.

Problem 3: derivative test (35 points).

Let us verify the correct implementation of the above material models by checking if the stresses are indeed the correct derivatives of the energy, and analogously the tangent matrix the derivative of the stresses. To this end, we use the forward-Euler finite-difference approximation

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

where $|h| \ll 1$ is a small perturbation. Let us implement a function that verifies if $\tilde{\boldsymbol{P}} = \partial W/\partial\tilde{\boldsymbol{\beta}}$ and $\tilde{\mathbb{C}} = \partial\tilde{\boldsymbol{P}}/\partial\tilde{\boldsymbol{\beta}}$ are implemented correctly, as follows.

Let your new function pick a random (reasonable) $\tilde{\boldsymbol{\beta}}$ and numerically compute each component of $\tilde{\boldsymbol{P}}$ and $\tilde{\mathbb{C}}$ using the above finite difference approximation (by perturbing each component of $\tilde{\boldsymbol{\beta}}$ by a small h, one component at a time). Compare those numerical values to the stress and stiffness values returned by the material model. The material model passes the test if the relative error between numerical and analytical derivatives (e.g., $||\tilde{\boldsymbol{P}}_{\text{analytical}} - \tilde{\boldsymbol{P}}_{\text{numerical}}||/||\tilde{\boldsymbol{P}}_{\text{analytical}}||$) is below a specified tolerance tol.

Use the test function to verify the correct implementation of your own linear elastic and Neo-Hookean models. For testing, use $h = 10^{-6}$ and $tol = 10^{-4}$ and pick some positive material constants.

total: 100 points

Hints for the derivative test:

For example, the components of the stress vector $\tilde{\boldsymbol{\sigma}}$ can be checked as follows:

- Pick a random $\tilde{\boldsymbol{\beta}} \neq \mathbf{0}$.
- Compute the energy $W_0 = W(\tilde{\boldsymbol{\beta}})$.
- Next for each component i of $\tilde{\boldsymbol{\beta}}$:

  Compute a perturbed $\tilde{\boldsymbol{\beta}}'$ such that $\tilde{\boldsymbol{\beta}}' = \tilde{\boldsymbol{\beta}}$ except for one perturbed component $\tilde{\beta}_i' = \tilde{\beta}_i + h$.

  Compute the perturbed energy $W' = W(\tilde{\boldsymbol{\beta}}')$.

  Compute the stress component
  $$\tilde{\sigma}_{i,\text{numerical}} = \frac{W' - W_0}{h}.$$
- Repeat the above procedure for all components i of $\tilde{\boldsymbol{\beta}}$ to compute all components of $\tilde{\boldsymbol{\sigma}}$.

The calculation of $\tilde{\mathbb{C}}$ follows analogously: compute $\tilde{\boldsymbol{\sigma}}_0$, then perturb each component j of $\tilde{\boldsymbol{\beta}}'$ and compute $\tilde{\boldsymbol{\sigma}}'$ for each j. Then $\mathbb{C}_{ij,\text{numerical}} = (\tilde{\sigma}_{0,i} - \tilde{\sigma}_i')/h$.