Project #1

assigned: Wednesday, September 28th, 2017
due: Wednesday, October 12th, 2016, 17:00
please hand in to one of the TAs in LEE N203

(Theory) Problem 1: continuum mechanics, indicial notation (50 points).

Let us derive the constitutive relations of elastic solids from their strain energy densities W:

(a) First, consider a *linear elastic* solid whose energy density is given by

$$W(\boldsymbol{\varepsilon}) = \frac{\lambda}{2}(\operatorname{tr}\boldsymbol{\varepsilon})^2 + \mu\,\boldsymbol{\varepsilon}\cdot\boldsymbol{\varepsilon} = \frac{\lambda}{2}(\varepsilon_{ii})^2 + \mu\,\varepsilon_{ij}\varepsilon_{ij}$$

with known material constants $\lambda, \mu > 0$.

Derive the components of the stress tensor $\boldsymbol{\sigma}$ and of the incremental modulus tensor $\mathbb{C}$:

$$\sigma_{ij} = \frac{\partial W}{\partial \varepsilon_{ij}}, \qquad \mathbb{C}_{ijkl} = \frac{\partial \sigma_{ij}}{\partial \varepsilon_{kl}}.$$

(b) For the above material model, consider a uniaxial stress state with $\sigma_{11} = \sigma$ and $\sigma_{ij} = 0$ else, and by symmetry we know that the transverse strains satisfy $\varepsilon_{22} = \varepsilon_{33}$. Solve for the strain components ε_{ij} and compute Young's modulus E and Poisson's ratio ν as functions of λ and μ, using the definitions

$$E = \frac{\sigma_{11}}{\varepsilon_{11}}, \qquad \nu = -\frac{\varepsilon_{22}}{\varepsilon_{11}}.$$

(c) Next, consider a hyperelastic rubbery solid whose behavior is describe by a so-called *compressible Neo-Hookean* model with energy density

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

with known material constants $\lambda, \mu > 0$.

Derive the components of the first Piola-Kirchhoff stress tensor $\boldsymbol{P}$ and of the incremental modulus tensor $\mathbb{C}$:

$$P_{iJ} = \frac{\partial W}{\partial F_{iJ}}, \qquad \mathbb{C}_{iJkL} = \frac{\partial P_{iJ}}{\partial F_{kL}}.$$

(d) Show that the modulus tensor in the undeformed ground state (i.e., for $\boldsymbol{F} = \boldsymbol{I}$) agrees with that of the linear elastic model above (therefore, for small strains both models agree).

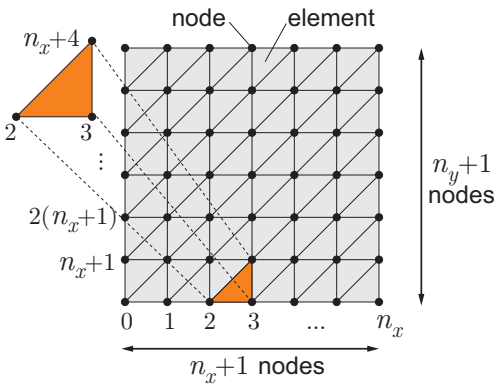
Helpful relations:

When taking tensor derivatives, the following relations are helpful and do not need to be derived:

$$\frac{\partial J}{\partial F_{iJ}} = J\,F^{-1}_{Ji} \qquad \text{and} \qquad \frac{\partial F^{-1}_{Ji}}{\partial F_{kL}} = -F^{-1}_{Jk}F^{1}_{Li}.$$

(Coding) Problem 2: vectors, arrays, Paraview (50 points).

To get started with C++, let us begin with a geometric exercise and write the code that creates what we will later call a finite element mesh (we do not need to know anything about finite elements yet). Shown on the right is a rectangular body subdivided into triangular elements.



For given outer side lengths a and b and given numbers of elements per side n_x and n_y, we would like to determine where all vertices (called *nodes*) are located and which three nodes jointly form each triangle (called *element*). For convenience, let us agree on the node numbering scheme shown in the figure.

We would like to store the following information:

(i) for each **node**, we need
– its ID, i.e., its unique node number between 0 and $(n_x+1)(n_y+1)-1$, stored as an `int`,
– its position (x, y) stored as a `Vector2D`.

(ii) for each **element**, we need
– the IDs of the three nodes forming the triangle (in any counter-clockwise order). These three numbers are generally referred to as the element's *connectivity* and stored as an array of three integers $(i,\ j,\ k)$, i.e., as an `array<int,3>`.

Because we will want to hand all of the above information in a convenient way to other parts of our code, we introduce a class called `Mesh` which contains these two items:

```
class Mesh {
   vector<Node>           _nodes;          (this is the list of all (x,y) position vectors)
   vector<array<int,3> > _connectivity;    (this is the list of all (i,j,k) connectivities);
}
```

`Furthermore, we introduce a class Node` which contains the nodal ID and position:

```
class Node {
   int      _ID;          (this is the unique nodal ID)
   Vector2D _position;    (this is the nodal position (x,y))
}
```

Our tasks are the following: we need to create a mesh, create all the nodes (with their positions and IDs) and add them to `_nodes` and likewise add all the triples of element nodes to `_connectivity`. You can check your mesh by accessing and checking individual nodes or element connectivities. Finally, we visually check if the mesh is correct by creating a file that can opened and visualized by Paraview, our post-processing software that will be used for later projects as well.

total: 100 points

General Hints and Remarks:

- You are encouraged to discuss the problems and coding tasks with each other. However, writing down the solutions and the code must be done by everyone on their own (projects with several names on them will not be accepted; identical code among participants will not be accepted either). When handing in your projects, you may be asked some quick questions about your code.
- For all problems, feel free to use Mathematica or Matlab if it helps. If you do, please submit a printout to show your work. For all theory problems, please show your work.
- If you need help accessing the server, editing and compiling files, etc., have a look at the practical tips found on the course website and/or contact us.
- Your home directory *(home/yourUsername/)* has a folder `project1` which contains all files required for this project. There, you will find an executable (ending in .cc) and utility files (ending in .h).
- These files compile without errors but do nothing as of now. They contain a few example commands that are meant as demonstrations of C++ syntax and can be changed in any way you want.
- `ToDo` comments will show you were your input is required.
- Our code uses the library `Eigen` which defines, e.g., `Vector2D` and has myriads of vector and tensor operations available (such as trace, determinant, inverse, transpose, etc.) and has substantial online documentation, see e.g., here. Feel free to take full advantage of those where needed.
- Please feel free to take full advantage of our *office hours twice a week*. We are happy to help!
- When submitting your project, please include a **print-out** of all the code files you modified (this will also allow us to leave comments and grades on your project sheets).