MAE 7750: HW #3

Hanfeng Zhai

April 11, 2023

Pseudocode *Design a detailed pseudocode for a nonlinear hyperelasticity problem in 2D. The pseudocode should refer to which functions and subroutines will be called for each steps. Discuss what happens in each step and also mention the relevant variables. (the file points to the contets of the "hyper" folder to find other subroutines and read input files. Specific emphasis should be placed on the Newton Raphson iterative solution scheme. What are the convergence criteria?*

In this problem, I write the pseudo code mainly based on "`test_cylinder.py`". The pseudo-code is shown in Algorithm 1. The emphasis on the Newton-Raphson iteration scheme begins in Line 13.

Base BVP *Run the code and plot the deformed mesh for the St. Venant and Neo-Hookean constitutive laws.*

We first do a brief description of the mechanics problem. For the Neo-Hookean model, the strain energy density function writes (according to the codes):

$$\phi = \frac{\mu}{2}\left(I_1 - 3 - 2\log J\right) + \frac{\lambda}{2}\log(J)^2 \tag{1}$$

where μ and λ are the first and second Lamé constants. We can further compute the second Piola-Kirchhoff stress $\mathbf{S}$ and the elasticity tensor $\mathbf{M} \equiv \mathbb{C}$ from $\mathbf{S} = 2\frac{\partial \phi}{\partial \mathbf{C}}$ and $\mathbf{M} = 2\frac{\partial \mathbf{S}}{\partial \mathbf{C}}$, respectively, where $\mathbf{C} = \mathbf{F}^\mathsf{T}\mathbf{F}$ is the right Cauchy-Green tensor.

We denote the prescribed displacement on the inner surface as u_0. The boundary condition writes:

$$\begin{aligned} u_r = u_{\text{inner}} &: u_r = u_0 = 1 \times 10^{-4} \\ u_r = u_{\text{outer}} &: \sigma_r = 0 \\ &\sigma_\theta = 0 \end{aligned} \tag{2}$$

Using both the coarse mesh and triangular refined mesh we generate Figure 1: the updated deformed cylinders are shown in red (current configuration $\mathbf{x}$) and undeformed cylinders are marked in blue ($\mathbf{X}$).

Alternate BVP *Change the boundary conditions and run an alternate BVP of your choice repeating the two runs of the previous task.*

Algorithm 1 Nonlinear Finite Element Analysis of Hyperelasticity

function `cut`($\mathbf{M}$, bcDofs, freeDofs)
    Cut the degree of freedom of the elasticity tensor $\mathbf{M}$ into the boundary and free degree of freedom parts.
function `verify`($\mathbf{U}$, $\mathbf{U}_{ref}$, bcDofs, tolerance)
    Checks if the values of $\mathbf{U}$ at the indices specified in bcDofs are the same as the corresponding values in $\mathbf{U}_{ref}$ up to the specified tolerance. If they are not, it throws an error or warning.
Define the material properties: Set Young's modulus and Poisson's ratio for rubber (and steel — here we consider rubber). Recall the constitutive model of Neo Hookean, Gent, and St. Venant Kirchhoff Hyperelasticity.
Define the iteration parameters: Specify the number of steps in the loop, the size of each step, and the magnitude of the radial displacement applied to the interior boundary of the cylinder (I further change it to the exterior cylinder).
Define the solver parameters: Set the precision, tolerance, and the maximum number of iterations.
Specify the file and folders to save the output files. Create something like "`/dirout/`".
Generate mesh based on geometry and recall `gmsh` *python module.* One can either load pre-generated mesh in a specific directory or generate using the existing module.
Set boundary conditions and force vectors.
Assembly the global stiffness matrix. Solve based on $\mathbf{M} \cdot \mathbf{U} = \mathbf{F}$ (internal + external).
Apply the loading and update the information. Apply the iteration using the Newton-Raphson iteration solution scheme. Initialize the Residual, Stiffness, etc.
while Residual > Precision **do**
    Compute the internal stiffness matrix $\mathbf{K}$ from displacements.
    Compute the stiffness matrix $\mathbf{M}$ from $\mathbf{K}$.
    Solve the unknown displacement $\mathbf{U}$ from $\mathbf{M}$ and Residual $\mathbf{R}$.
    Compute Force $\mathbf{F}$ from $\mathbf{U}$; Compute external force $\mathbf{F}_{\text{ext}} = \mathbf{M}_{11}\mathbf{U}_1 + \mathbf{M}_{12}\mathbf{U}_2$.
    Update Residual $\mathbf{R} = \mathbf{F}_{\text{ext}} - \mathbf{F}_{\text{int}}$; Cut $\mathbf{R}$ into $\mathbf{R}_1$ & $\mathbf{R}_2$.
    Verify the boundary conditions satisfy preset tolerance.
    if normalized test function < precision **then**
        Update iteration.
    end if
    for Local nodes in Displacement nodes **do**
        Compute the local displacement $\mathbf{u}$, Obtain the reference configuration $\mathbf{X}$; $\rightarrow$ Get current configuration $\mathbf{x} = \mathbf{X} + \mathbf{u}$; $\rightarrow$ Obtain radial strain ϵ_r.
        Update nodal force and internal displacement from ϵ_r.
    end for
    for Local element in Overall elements **do**
        Compute different kinds of Stresses (i.e., Cauchy, First Piola-Kirchhoff, ...), and Cauchy Green strains $\mathbf{E}$.
    end for
end while
Plot the figures and postprocessing.

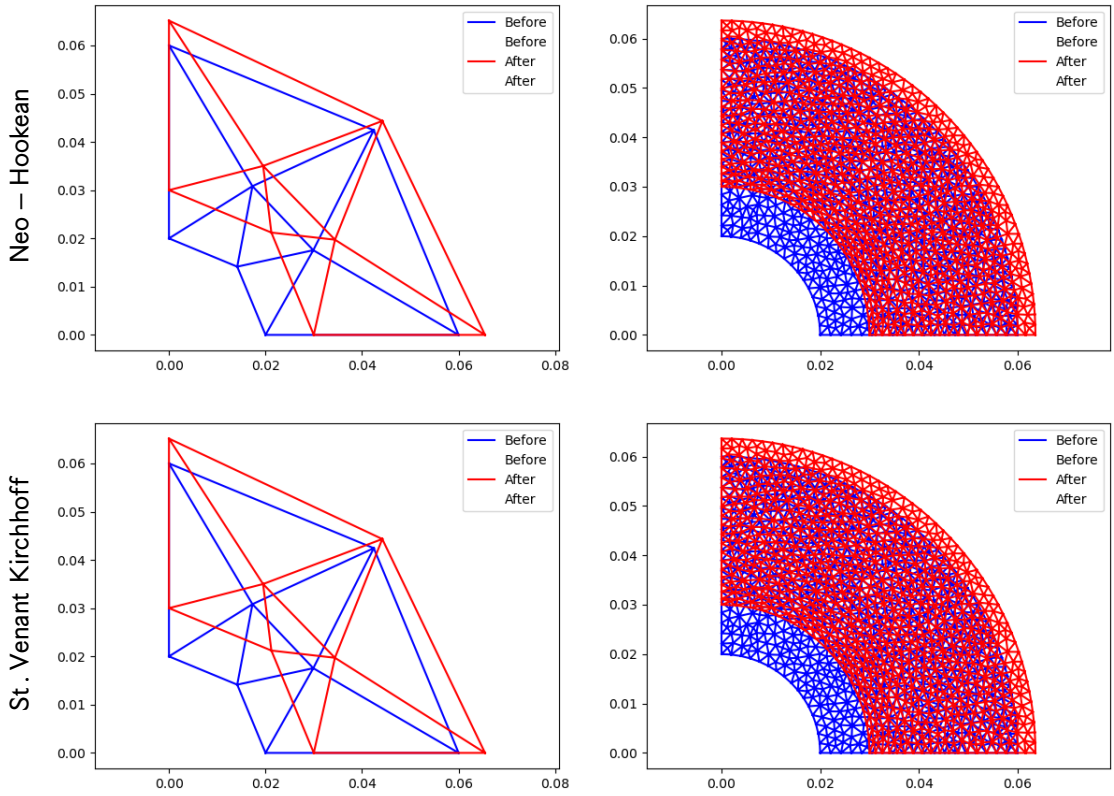

Figure 1: The original (blue) and deformed (red) mesh for the cylinder using both the coarse (left) and fine (right) triangular meshes using the Neo-Hookean (up) and St. Venant (bottom) constitutive laws for the original given boundary conditions.

I modify the new boundary conditions as one applies external displacement towards to radial center, shown in Figure 2.

The new boundary condition writes:

$$\begin{aligned} u_r = u_{\text{outer}} : u_r &= -u_0 = -1 \times 10^{-4} \\ u_r = u_{\text{inner}} : \sigma_r &= 0 \\ \sigma_\theta &= 0 \end{aligned} \tag{3}$$

The results for using the Neo-Hookean and St. Venant Kirchhoff hyperelasticity constitutive models are shown in Figure 3. For both the previous and this problem, we adopt the rubber material as the two constitutive laws are mainly used for rubber.

Implementation of another constitutive law *Implement the necessary code for another hyperelastic constitutive law. It is suggested to work with the compressible Gent law, but any other hyperelastic law of your choice (such as Arruda-Boyce 8-chain) is also acceptable.*

Here, we use the Gent hyperelasticity constitutive model. Recalling the potential energy

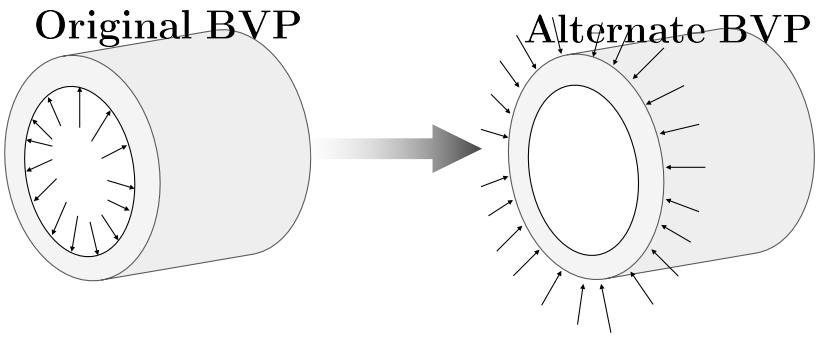

Figure 2: The schematic for my alternate boundary conditions.

model by Horgan and Saccomandi [1]:

$$\phi = -\frac{\mu}{2}\left[J_m \log\left(\left(1+\frac{3}{J_m}\right) - \frac{\mathrm{tr}(\mathbf{C})}{J_m}\right) + 2\log J\right] \tag{4}$$

We can hence compute the 2nd Piola-Kirchhoff stress from the strain energy density function:

$$\begin{aligned}
\mathbf{S} &= 2\frac{\partial \phi}{\partial \mathbf{C}} \\
&= -2\frac{\mu}{2}\left[\frac{-J_m\mathbf{I}}{J_m+3-\mathrm{tr}(\mathbf{C})} + 2\frac{1}{2}\mathbf{C}^{-1}\right] \\
&= -\mu\left[\frac{-J_m\mathbf{I}}{J_m+3-\mathrm{tr}(\mathbf{C})} + \mathbf{C}^{-1}\right] \\
&= \frac{\mu J_m\mathbf{I}}{J_m+3-\mathrm{tr}(\mathbf{C})} - \mu\mathbf{C}^{-1}
\end{aligned} \tag{5}$$

Based on the 2nd PK stress tensor, one can compute the elasticity tensor in terms of $\mathbf{C}$:

$$\begin{aligned}
\mathbf{M} &= 2\frac{\partial \mathbf{S}}{\partial \mathbf{C}} \\
&= 2\frac{\partial\left[\frac{\mu J_m\mathbf{I}}{J_m+3-\mathrm{tr}(\mathbf{C})} - \mu\mathbf{C}^{-1}\right]}{\partial \mathbf{C}} \\
&= \frac{2\mu J_m\mathbf{I}\otimes\mathbf{I}}{\left[J_m+3-\mathrm{tr}(\mathbf{C})\right]^2} - 2\mu\frac{\partial \mathbf{C}^{-1}}{\partial \mathbf{C}}
\end{aligned} \tag{6}$$

Implement this in `Python` we generate the following codes:

```
class GentElasticity(Elasticity):

    def __init__(self, E, nu):
        super(GentElasticity, self).__init__(E, nu)

    def potential(self, C):
        lamda = self.get1LAME()
        mu = self.get2LAME()
```

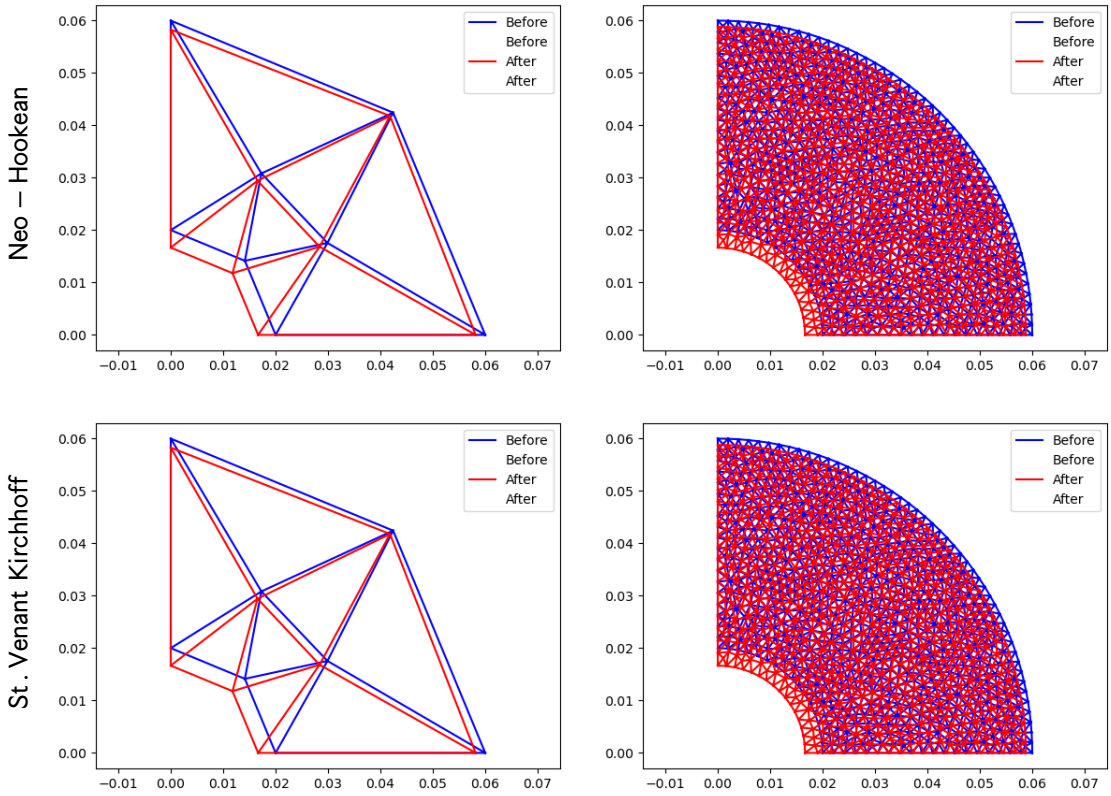

Figure 3: The original (blue) and deformed (red) mesh for the cylinder using both the coarse (left) and fine (right) triangular meshes using the Neo-Hookean (up) and St. Venant (bottom) constitutive laws for the alternate (Figure 2) boundary conditions.

```
        dim = len(C)
        if dim == 2:
            K = np.copy(C)
            C = np.zeros((3, 3))
            C[:2, :2] = K[:, :]
        J = np.sqrt(tensor.det(C))
        jm = 50
        phi = - (mu / 2) * ( jm * np.log( (1 + (3/jm)) - (tensor.trace(C)/jm) ) + 2 * np.log
(J))
        return phi

    def stress(self, C):
        dim = len(C)
        if dim == 2:
            K = np.copy(C)
            C = np.eye(3)
            C[:2, :2] = K[:, :]
        PK2 = tensor.tensor(dim)
        lamda = self.get1LAME()
        mu = self.get2LAME()
        invC = tensor.inv(C)
        I = tensor.I(3)
        jm = 50
        PK2 = ( ( mu * jm * I )/( jm + 3 - tensor.trace(C) ) ) - mu * invC
```

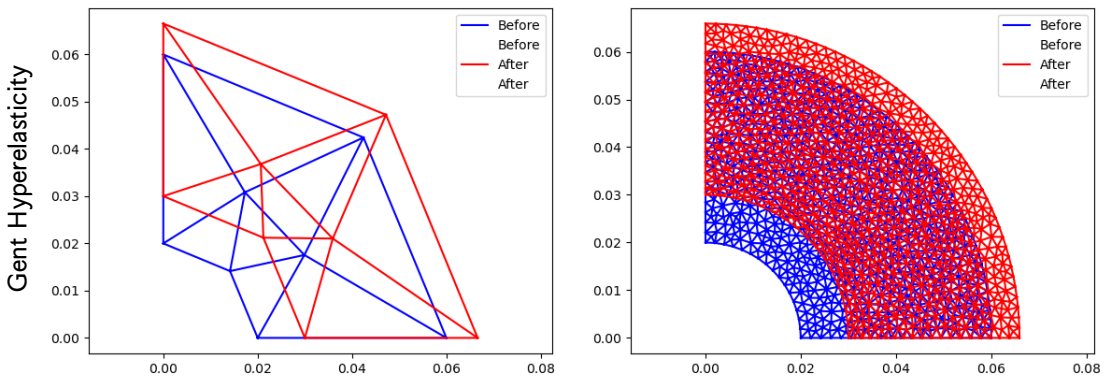

Figure 4: The original (blue) and deformed (red) mesh for the cylinder using both the coarse (left) and fine (right) triangular meshes using the Gent constitutive laws for the original given boundary conditions.

```
        if dim == 2:
            return PK2[:2, :2]
        return PK2

    def stiffness(self, C):
        d = len(C)
        dim = len(C)
        lamda = self.get1LAME()
        mu = self.get2LAME()
        invC = tensor.inv(C)
        dinvC = tensor.tensor4(d)
        for i in range(d):
            for j in range(d):
                for k in range(d):
                    for l in range(d):
                        part1 = invC[i, k] * invC[j, l]
                        part2 = invC[i, l] * invC[j, k]
                        dinvC[i, j, k, l] = -(part1 + part2) / 2

        I = tensor.I(dim)
        IxI = tensor.outerProd4(I, I)
        jm = 50
        M = ( (2 * mu * jm * IxI) / (jm + 3 - tensor.trace(C))**2 ) - 2 * mu * dinvC
        return M
```

References

[1] C. O. Horgan and G. Saccomandi, *Constitutive models for compressible nonlinearly elastic materials with limiting chain extensibility*, Journal of Elasticity, 77 (2004), pp. 123–138.