UPC
MSc Computational Mechanics
Spring 2018

Computational Solid Mechanics

Homework 3

Prasad ADHAV
Aditya MANGAONKAR

Contents

1 Kirchhoff Saint-Venant material model

Isotropic linear elasticity can be derived from balance of linear momentum, the linearized strain displacement relation $\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)$*, and the stored elastic energy function*

$$W(\boldsymbol{\varepsilon}) = \frac{\lambda}{2} Tr(\boldsymbol{\varepsilon})^2 + \mu Tr(\boldsymbol{\varepsilon}^2)$$

1. *Check that, the stress tensor obtained from* $\boldsymbol{\sigma} = \partial W / \partial \boldsymbol{\varepsilon}$ *agrees with the usual linear elasticity expression.*

Let us use index notation to prove this result. Then,

$$\sigma_{ij} = \frac{\partial W}{\partial \varepsilon_{ij}} \quad ; \quad W = \frac{\lambda}{2}(\varepsilon_{ii})^2 + \mu \varepsilon_{jk}\varepsilon_{kj}$$

and thus,

$$\sigma_{ij} = \frac{\partial W}{\partial \varepsilon_{ij}} = \frac{\partial}{\partial \varepsilon_{ij}}\left[\frac{\lambda}{2}(\varepsilon_{kk})^2 + \mu \varepsilon_{pq}\varepsilon_{qp}\right] = \frac{\lambda}{2}\frac{\partial}{\varepsilon_{ij}}(\varepsilon_{kk})^2 + \mu \frac{\partial}{\varepsilon_{ij}}[\varepsilon_{pq}\varepsilon_{qp}] = \frac{\lambda}{2} 2\varepsilon_{kk}\frac{\partial \varepsilon_{kk}}{\varepsilon_{ij}} +$$

$$\mu \varepsilon_{qp}\frac{\partial \varepsilon_{pq}}{\partial \varepsilon_{ij}} + \mu \varepsilon_{pq}\frac{\partial \varepsilon_{qp;}}{\partial \varepsilon_{ij}} = \lambda \varepsilon_{kk}\frac{\partial \varepsilon_{kk}}{\varepsilon_{ij}} + \mu \varepsilon_{qp}\frac{\partial \varepsilon_{pq}}{\partial \varepsilon_{pq}}\delta_{pi}\delta_{qj} + \mu \varepsilon_{pq}\frac{\partial \varepsilon_{qp}}{\partial \varepsilon_{qp}}\delta_{qi}\delta_{pj} = \lambda \varepsilon_{kk}\frac{\partial \varepsilon_{kk}}{\partial \varepsilon_{kk}}\delta_{ik}\delta_{jk} +$$

$$\mu \varepsilon_{ji} + \mu \varepsilon_{ji} = \lambda \varepsilon_{kk}\delta_{ij} + 2\mu \varepsilon_{ij}$$

where we have taken into account that $\partial \varepsilon_{ii} / \partial \varepsilon_{ii} = 1$. Therefore, it has been shown that

$$\boldsymbol{\sigma} = \lambda Tr(\varepsilon)\boldsymbol{I} + 2\mu \boldsymbol{\varepsilon} \qquad\qquad Q.E.D. \tag{1}$$

Since the linearization of the Green-Lagrange strain tensor $\boldsymbol{E} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I})$ *is the small strain tensor* $\boldsymbol{\varepsilon}$*,it is natural to extend isotropic elasticity to nonlinear elasticity as*

$$W(\boldsymbol{E}) = \frac{\lambda}{2} Tr(\boldsymbol{E})^2 + \mu Tr(\boldsymbol{E}^2) \tag{2}$$

This hyperelastic model is called Kirchhoff Saint-Venant material model.

2. *According to the definition we gave in class about isotropy in nonlinear elasticity, is this model isotropic?*

According to the lecture, a material is isotropic if and only if any rotation is a material symmetry. In a material model, this definition implies that the strain energy function can be written in terms of the principal invariants of $\boldsymbol{C}$. The energy function for our case can be easily rewritten as

$$W(\boldsymbol{C}) = \frac{\lambda}{8} Tr(\boldsymbol{C} - \boldsymbol{I})^2 + \frac{\mu}{4} Tr\left[(\boldsymbol{C} - \boldsymbol{I})^2\right]$$

and, as one can tell, this function mainly depends on the trace of $\boldsymbol{C}$ which is known to be an invariant. Therefore, we conclude that this model is isotropic.

3. *Derive the second Piola-Kirchhoff stress* $\boldsymbol{S}$*.*

The second Piola-Kirchhoff stress tensor is obtained by the following expression,

$$\boldsymbol{S} = \frac{\partial W(\boldsymbol{E})}{\partial \boldsymbol{E}} \tag{3}$$

Let us use again index notion for the compution, as we did for the first exercise. Hence,

$$S_{ij} = \frac{\partial W}{\partial E_{ij}} = \frac{\partial}{\partial E_{ij}}\left[\frac{\lambda}{2}(E_{kk})^2 + \mu E_{pq}E_{qp}\right] = \lambda E_{kk}\frac{\partial E_{kk}}{\partial E_{ij}} + \mu E_{pq}\frac{\partial E_{qp}}{\partial E_{ij}} + \mu E_{qp}\frac{\partial E_{pq}}{\partial E_{ij}} =$$

$$= \lambda E_{kk}\frac{\partial E_{kk}}{\partial E_{kk}}\delta_{ik}\delta_{jk} + \mu E_{pq}\frac{\partial E_{qp}}{\partial E_{qp}}\delta_{iq}\delta_{jp} + \mu E_{qp}\frac{\partial E_{pq}}{\partial E_{pq}}\delta_{ip}\delta_{jq} = \lambda E_{kk}\delta_{ij} + 2\mu E_{ij}$$

Therefore, it yields

$$\boldsymbol{S} = \lambda Tr(\boldsymbol{E})\boldsymbol{I} + 2\mu\boldsymbol{E} \tag{4}$$

or, recalling the previous definition of the Green-Lagrange strain tensor, the second Piola-Kirchhoff stress tensor can be rewritten alternatively as

$$\boldsymbol{S} = \frac{\lambda}{2} Tr(\boldsymbol{C} - \boldsymbol{I})\boldsymbol{I} + \mu(\boldsymbol{C} - \boldsymbol{I}) \tag{5}$$

4. *For a uniform deformation of a rod aligned with the X axis ($x = \Lambda X$, $y = Y$, $z = Z$, whee Λ is the stretch ratio along the X direction), derive the relation between the nominal normal stress P (the xX component of the first Piola-Kirchhoff stress) and the stretch ratio Λ, P(Λ) and plot it.*

The deformation mapping is given by,

$$\boldsymbol{x} = \varphi(\boldsymbol{X}) = (\Lambda X, Y, Z)$$

and the first Piola-Kirchhoff stress tensor can be computed as,

$$\boldsymbol{P} = \boldsymbol{F}\boldsymbol{S} \tag{6}$$

Let us now start by computing the deformation gradient tensor, i.e.

$$F_{iJ} = \frac{\partial \varphi_i}{\partial X_J} \qquad \Rightarrow \qquad \boldsymbol{F} = \begin{bmatrix} \Lambda & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

In order to compute the second Piola-Kirchhoff stress tensor, according to equation 3, we need to compute $\boldsymbol{E}$ and therefore also the right Cauchy-Green deformation tensor $\boldsymbol{C}$. Then,

$$\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F} \quad \Rightarrow \quad \boldsymbol{C} = \begin{bmatrix} \Lambda & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Lambda & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \Lambda^2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and then, the Green-Lagrange strain tensor reads,

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I}) \quad \Rightarrow \quad \boldsymbol{E} = \frac{1}{2} \begin{bmatrix} (\Lambda^2 - 1) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Now, by making use of equation 4, we obtain,

$$\boldsymbol{S} = \frac{\lambda}{2}(\Lambda^2 - 1) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + 2\mu\frac{1}{2} \begin{bmatrix} (\Lambda^2 - 1) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = (\Lambda^2 - 1) \begin{bmatrix} (\lambda/2 + \mu) & 0 & 0 \\ 0 & \lambda/2 & 0 \\ 0 & 0 & \lambda/2 \end{bmatrix}$$

Finally, by replacing previous equations for $\boldsymbol{F}$ and $\boldsymbol{S}$ into expression 6, it yields

$$\boldsymbol{P} = \begin{bmatrix} \Lambda & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} (\Lambda^2 - 1) \begin{bmatrix} (\lambda/2 + \mu) & 0 & 0 \\ 0 & \lambda/2 & 0 \\ 0 & 0 & \lambda/2 \end{bmatrix} = (\Lambda^2 - 1) \begin{bmatrix} \Lambda(\lambda/2 + \mu) & 0 & 0 \\ 0 & \lambda/2 & 0 \\ 0 & 0 & \lambda/2 \end{bmatrix}$$

Then, the relation between the nominal normal stress and the stretch ratio is,

$$P(\Lambda) = \Lambda(\Lambda^2 - 1)(\lambda/2 + \mu) \tag{8}$$

5. *Is the relation $P(\Lambda)$ monotonic? If not, derive the critical stretch Λ_{crit} at which the model fails with zero stiffness. Does this critical stretch depend on the elastic constants? Show that the material does not satisfy the growth conditions*

$$W(\boldsymbol{E}) \longrightarrow +\infty \qquad when \qquad J \longleftrightarrow 0^{+}$$

In Figure 1.1 we show the plot for $P(\Lambda)$ versus the values of Λ in the interval $[-1, 1]$, given a value for λ and μ. As we can see from this graph, the derivative (slope) of the function does change its sign. Thus the relation is clearly non–monotonic.

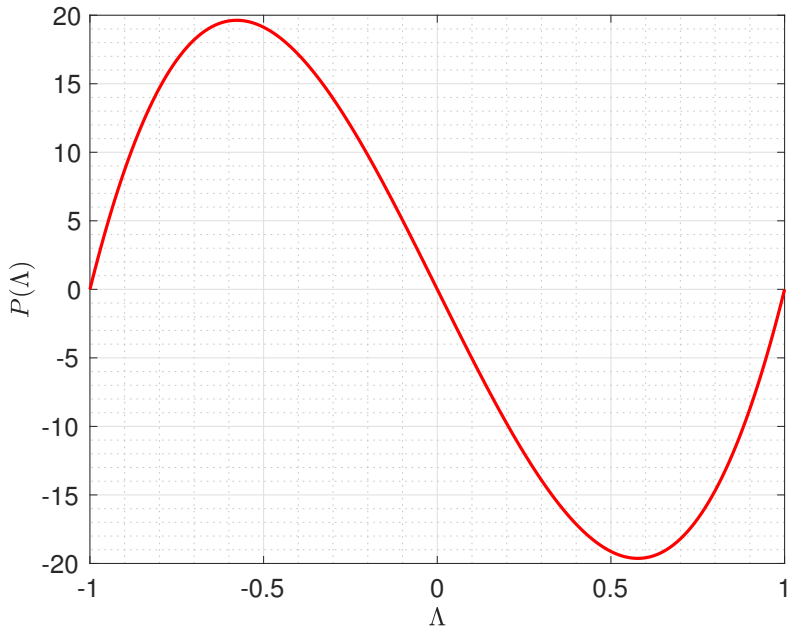


Figure 1.1: $P(\Lambda)$ vs Λ in the interval $[-1, 1]$

Let us now then derive a critical value for Λ. For this purpose, we have the condition

$$\frac{dP(\Lambda)}{d\Lambda} = 0$$

Therefore, from Eq. 8 one can easily obtain

$$\frac{dP(\Lambda)}{d\Lambda} = \left(\frac{\lambda}{2} + \mu\right)[3\Lambda^2 - 1] = 0 \qquad \Rightarrow \qquad \Lambda_{crit} = \pm\frac{\sqrt{3}}{3}$$

and clearly, Λ_{crit} does not depend on the elastic constant. Now, let us check the grow conditions. First of all, the Jacobian of the transformation would be,

$$J = \det \boldsymbol{F} = \det \begin{bmatrix} \Lambda & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \Lambda$$

From the expression of $\boldsymbol{E}$, we have

$$Tr(\boldsymbol{E}) = \frac{1}{2}(\Lambda - 1) \quad \Rightarrow \quad Tr(\boldsymbol{E})^2 = \frac{1}{4}(\Lambda - 1)^2$$

Also, one can easily obtain

$$\boldsymbol{E}^2 = \frac{1}{4}\begin{bmatrix} (\Lambda^2 - 1)^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \Rightarrow \quad Tr(\boldsymbol{E}^2) = \frac{1}{4}(\Lambda^2 - 1)^2$$

and therefore, the energy function now reads

$$W(\boldsymbol{E}) = \frac{\lambda}{2}Tr(\boldsymbol{E})^2 + \mu Tr(\boldsymbol{E}^2) = \left(\frac{\lambda}{8} + \frac{\mu}{4}\right)(\Lambda^2 - 1)^2 = \left(\frac{\lambda}{8} + \frac{\mu}{4}\right)(J^2 - 1)^2$$

where we have taken into account the value of the Jacobian. Taking the limit of the energy function, it yields,

$$\lim_{J\to 0} W(\boldsymbol{E}) = \lim_{J\to 0} \left(\frac{\lambda}{8} + \frac{\mu}{4}\right)(J^2-1)^2 = \left(\frac{\lambda}{8} + \frac{\mu}{4}\right)$$

As we can see, as the Jacobian goes to zero, the energy function yields into a finite value, and does not grow to infinite. Hence, the grow conditions are not fulfilled.

6. *Consider now the modified Kirchhoff Saint-Venant material model:*

$$W(\boldsymbol{E}) = \frac{\lambda}{2}(ln\ J)^2 + \mu Tr(\boldsymbol{E}^2)$$

Does this model circumvent the drawbacks of the previous model?

The model is different now, but we can still use

$$J = \det F = \Lambda \qquad ; \qquad Tr(\boldsymbol{E}^2) = \frac{1}{4}(\Lambda^2-1)^2$$

and thus, for this model we can write

$$W(\boldsymbol{E}) = \frac{\lambda}{2}(\ln\ J)^2 + \frac{\mu}{4}(J^2-1)^2$$

and taking the limit

$$\lim_{J\to 0}\left[\frac{\lambda}{2}(\ln\ J)^2 + \frac{\mu}{4}(J^2-1)^2\right] = +\infty$$

and hence, the growth conditions are now satisfied.

7. *Implement the material model in Eq. 2 in the Matlab code. Perform the consistency test to check your implementation. Try to demonstrate the material instabilities of this model with a numerical example.*

We have implemented the material model from Eq. 2 in the Matlab code provided for the assignment. In Box 1, 2 and 3 in Appendix A we collect a detailed implementation for this model. For the implementation, we need to derived both the second Piola–Kirchhoff $\boldsymbol{S}$ stress tensor and the constitutive tensor $\mathbb{C}$.

$$S_{ij} = \frac{\partial W}{\partial E_{ij}} = \frac{\partial}{\partial E_{ij}}\left[\frac{\lambda}{2}(\ln J)^2 + \mu E_{pq}E_{qp}\right] = \lambda E_{kk}\delta_{ij} + 2\mu E_{ij} = \frac{\lambda}{2}(C_{kk}-\delta_{kk})\delta_{ij} + \mu(C_{ij}-\delta_{ij}) \quad \forall\ i,j$$

where we have used a similar procedure as for previous models. Now, for the case of $\mathbb{C}$, we have

$$\mathbb{C}_{ijkl} = 2\frac{\partial \boldsymbol{S}_{ij}}{\partial \boldsymbol{C}_{kl}} = \lambda\frac{\partial(C_{mm}-\delta_{mm})}{\partial C_{kl}}\delta_{ij} + 2\mu\frac{\partial C_{ij}}{\partial C_{kl}} = \lambda\frac{\partial C_{mm}}{\partial C_{kl}}\delta_{ij} + 2\mu\frac{\partial C_{ij}}{\partial C_{kl}} =$$

$$= \lambda\frac{\partial C_{mm}}{\partial C_{mm}}\delta_{ij}\delta_{mk}\delta_{ml} + 2\mu\frac{\partial C_{ij}}{\partial C_{ij}}\delta_{ik}\delta_{lj} = \lambda\delta_{lk}\delta_{ij} + 2\mu\delta_{ik}\delta_{lj}$$

This model was checked using the `Check_Derivatives` function provided. The results of the test yielded no error larger than the set tolerance (10^{-3}) for both the gradient and the Hessian. That proves that the derivation of the tensors and the implementation of the model was correct.

Figure 1.2 down below displays the evolution of the mesh with the final shape of the deformation for the case of a slender beam under compression. For this simulation, we have chosen to stretch the bar to its half length, so that we make sure to go through the theoretical critical value.

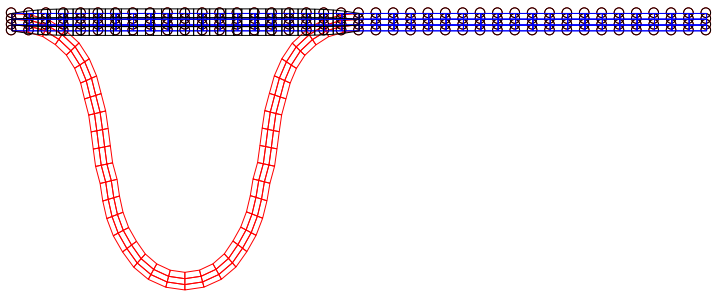


Figure 1.2: Mesh for a slender beam under compression using the Kirchhoff Saint–Venant material model.

Figure 1.3 shows the Force–displacement graph and the error plot for this model. We observe that this plot really differs from the linear elasticity solution, as expected. As we can see, there is a region where the force increases with the displacement but, a point is reached where the forces decrease to small values remaining then almost constant until the end of the simulation. This basically means that, keeping the same force, the beam could get infinitely deformed.

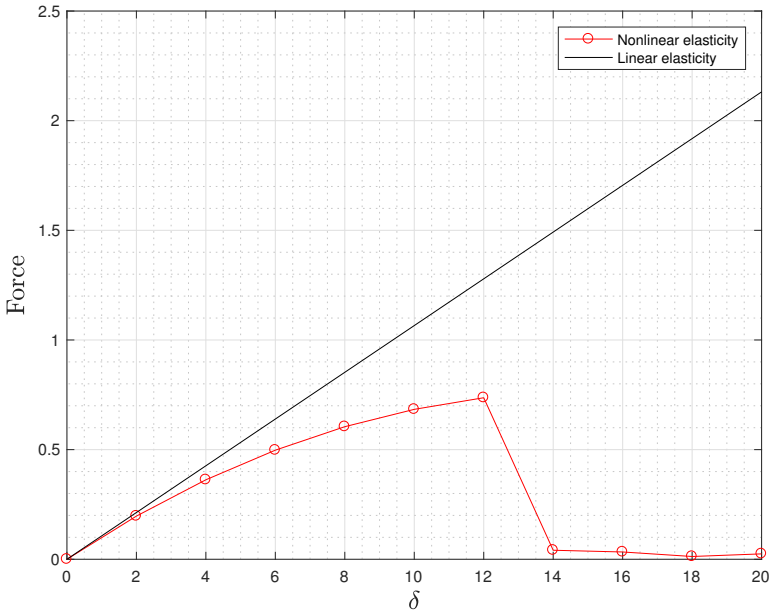


Figure 1.3: Force - displacement curve for a slender beam under compression using the Kirchhoff Saint–Venant material model.

Now one should recall, that for this Kirchhoff Saint–Venant model, we have derived a theoretical critical value of stretch for which the model should fail in successfully representing the real behavior of

the material. But also note that it was obtained previously that the model does not satisfy the so–called growth conditions. Therefore, as we reach the critical point, the model keeps providing a finite solution for the problem and it does not fail, what we know that should be a realistic representation of the problem.

Further, we have decided to implement the modified Kirchoff Saint–Venant model, so that we can numerically check how this model now improves the results obtained for this problem. The implementation details are provided in Box 4, 5 and 6. First, we need to derive both $\boldsymbol{S}$ and $\mathbb{C}$. Then, the calculations yield

$$S_{ij} = \frac{\partial W}{\partial E_{ij}} = -\mu\delta_{ij} + \mu C_{ij} + \lambda \ln J C_{ij}^{-1}$$

for the case of the second Piola–Kirchhoff stress tensor, and

$$\mathbb{C}_{ijkl} = 2\frac{\partial S_{ij}}{\partial C_{kl}} = 2\mu\delta_{ik}jl + \lambda C_{ij}^{-1}C_{kl}^{-1} - \ln J\left(C_{ik}^{-1}C_{jl}^{-1} + C_{il}^{-1}C_{jk}^{-1}\right)$$

Let us now present a short discussion on the results. Figure 1.4 shows the deformed mesh. As we can see, the model has completely offers a really distorted mesh, once we overpass the critical point.

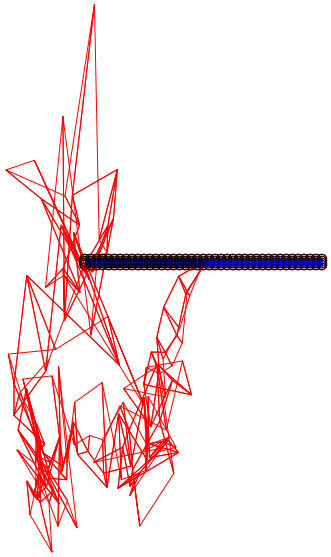


Figure 1.4: Mesh for a slender beam under compression using the modified Kirchhoff Saint–Venant material model.

Figure 1.5 shows the force–displacement plot. As one can tell, there is a points from which the values of the force start increasing without control. This a direct consequence of the growth conditions, which were demonstrated to be fulfilled for this model.

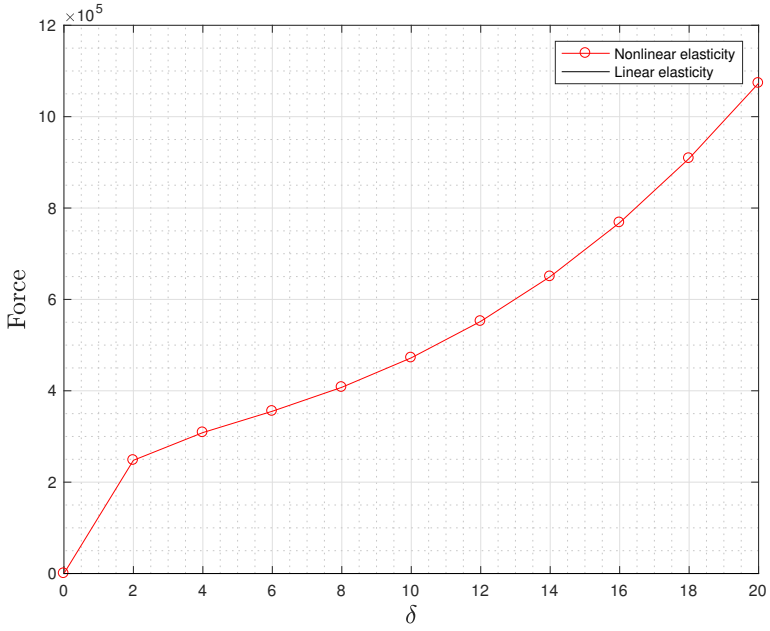


Figure 1.5: Force - displacement curve for a slender beam under compression using the modified Kirchhoff Saint–Venant material model.

2 Implementation of line-search

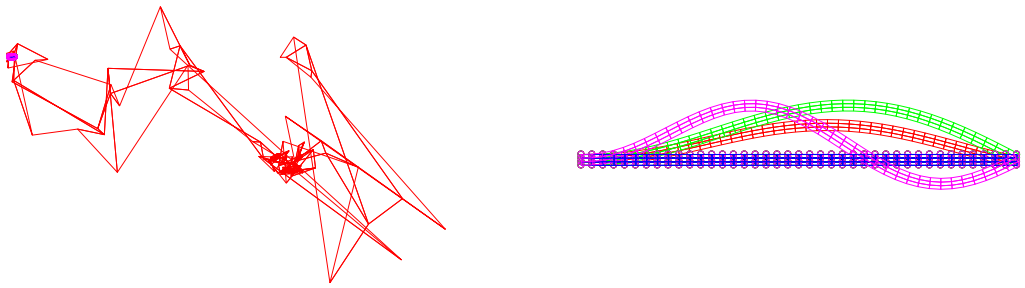
Implement a line-search algorithm to be used in combination with Newton's method. For this, I suggest you resort to Matlab's function ***`fminbd`****, which performs 1D nonlinear minimization with bounds. You need to define a function* ***`Ener_1D`*** *that evaluates the energy along the line that passes through x in the direction of* ***`p`*** *(the descent search direction). The function* ***`LineSearch`*** *may include lines like the ones suggested next:*

```
t=1;
opts=optimset('TolX',options.TolX,'MaxIter',options.n_iter_max_LS);
t = fminbnd(@(t) Ener_1D(t,x_short,p),0,2,opts);
x_short=x_short+t*p;
```

Test the code with the examples where you expect buckling (the compression of the beams, or the deflection of the arch), and compare the results with and without line–search.

The line–search strategy for solving non–linear systems of equations, updates the value of the new iteration based on a descent direction and a relaxation parameter, obtained from a 1D minimization problem. The line–search ensures that the solution of a non–linear system is stable when Newton's method generates unstable solutions. This technique is already provided as a material for the assignment and it is based on the *Matlab* function `fminbd`. Two cases where buckling is expected have been analyzed using the `NeoHookean` material, and the results are presented next.

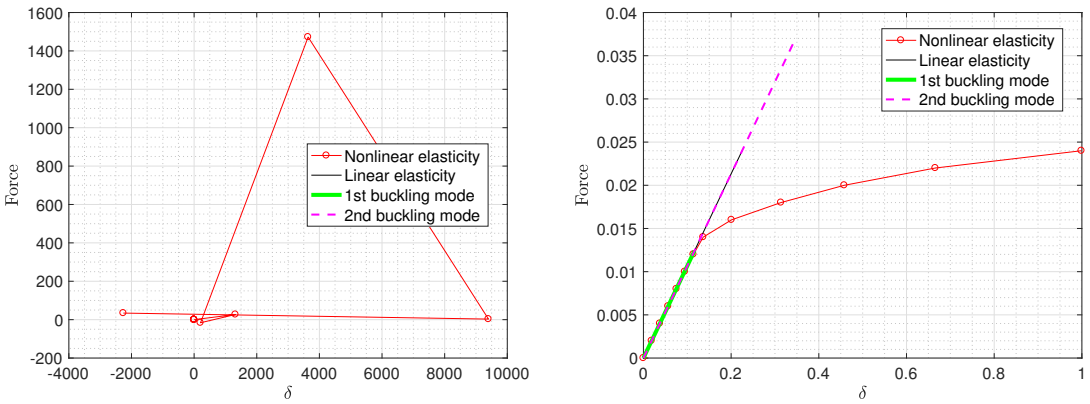
The first example solved is the case of a slender beam under dead load. Figure 2.1 shows the result obtained for the deformed mesh of the slender beam without line search (2.1a) and with line search (2.1b). As it can be clearly seen, the algorithm without line search activated is not able to compute the solution (it diverges) and the provided result is completely unrealistic. The solution obtained using the algorithm with line search is the expected one, as this improvement of the solver allows to overcome the instabilities of the case and yields the appropriate results.



(a) Mesh deformation without `LineSearch` (b) Mesh deformation considering `LineSearch`

Figure 2.1: Mesh for the slender beam under dead load with a NeoHookean material model.

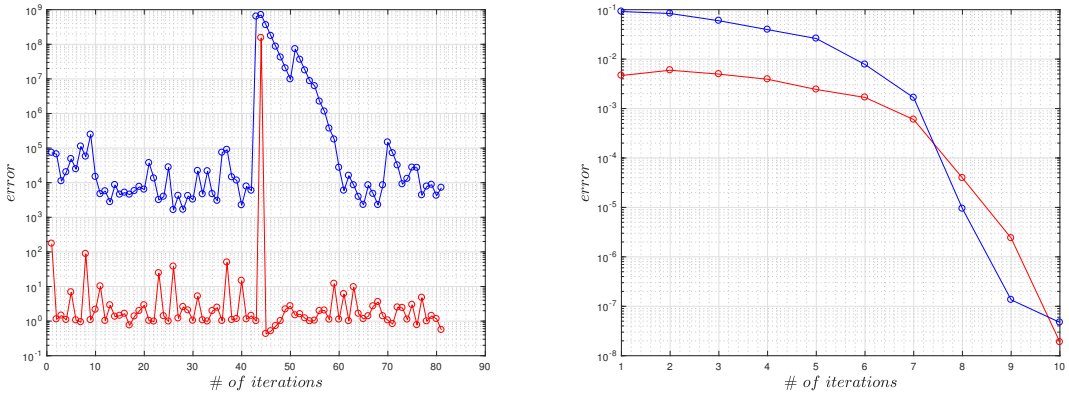
Figure 2.2 shows the force - displacement curves, for the algorithm without line search (2.2a) and with line search (2.2b). Once again, it is straight forward to see that the force obtained when solving the problem without line search is unrealistic, as it suddenly snaps from a low value to a very large one and then again to a low one. The solver is then unable to compute what actually happens with the beam. When the line search is activated, the instabilities are overcome and the results obtained are reasonable and expected, with the force increasing linearly with the displacement in the elastic region and then increasing in a non-linear trend once it enters the hyperelastic region.



(a) Force - displacement graph without `LineSearch` (b) Force - displacement graph using `LineSearch`

Figure 2.2: Force–displacement curve for the slender beam under dead load with a NeoHookean material model.

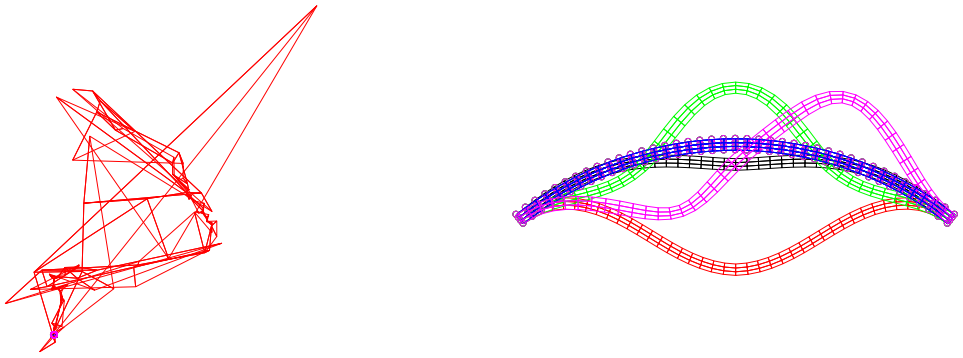
Figure 2.3 allows to compare the error obtained for the solver without line search and the solver with line search. The error obtained without using line search (represented in 2.3a) confirms the instabilities of the method, as it does not show any kind of convergence. The error obtained using line search (in 2.3b shows that the method is stable and it converges quadratically to the solution.



(a) Error without `LineSearch` (b) Error using `LineSearch`

Figure 2.3: Error obtained for the slender beam under dead load with a NeoHookean material model.

The other example solved is the arch under dead load, which is also a buckling problem. Figure 2.4 shows the meshes obtained without using line search (2.4a) and activating line search (2.4b). Again, for this problem the solution diverges if the line search improvement is not activated, as it can be clearly seen in the mesh plots. The solution obtained without line search diverges and has no physical meaning, while the algorithm with line search yields the expected displacement of the arch.

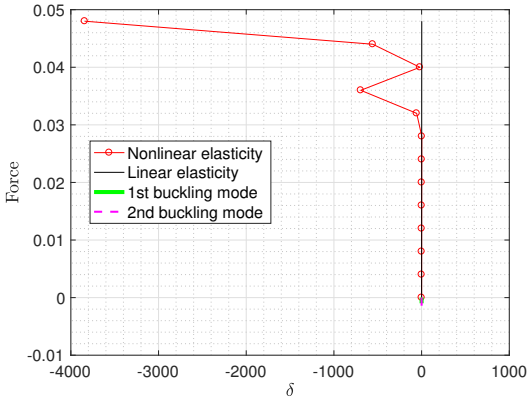


(a) Mesh deformation without using `LineSearch` (b) Mesh deformation when using `LineSearch`

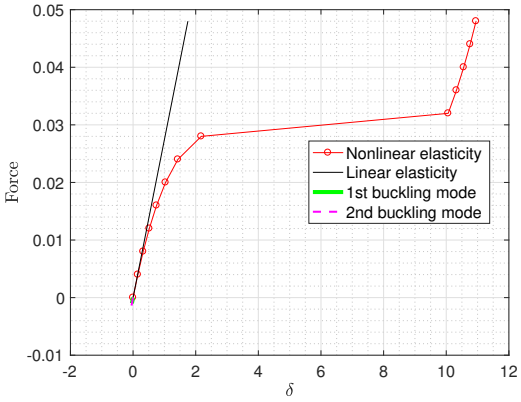
Figure 2.4: Meshes for arch with a dead load at the center modeled with a NeoHookean material.

Figure 2.5 shows the force - displacement curves for the arch problem without line search (2.5a) and with line search (2.5b). The force obtained without activating line search makes no sense and does not represent the actual behavior of the arch. The force obtained when line search is activated shows the expected behavior: the model is not able to exactly reproduce the evolution of the force during the buckling phase but it is able to yield a good approximation. The force increases as the displacement increases and the buckling is modeled as a big jump of displacement with a small increase in force exactly when the arch rapidly snaps from one stable position to the next one.

Figure 2.6 represents the error the algorithm without line search (2.6a) and the algorithm with line search (2.6b). Once again, the error plot without line search proves that the solution diverges and the results are not representative of the actual behavior of the arch. The error for the algorithm with line search proves that the solution rapidly converges to the result, as it would be expected.

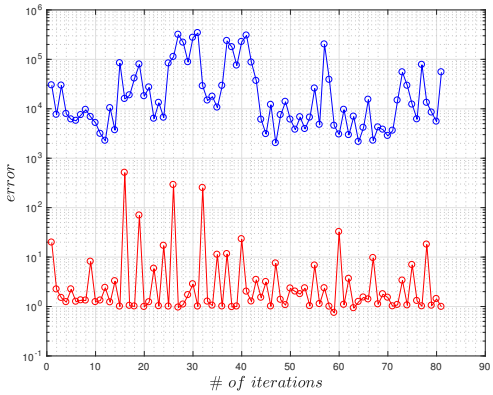


(a) F–δ graph without using `LineSearch`

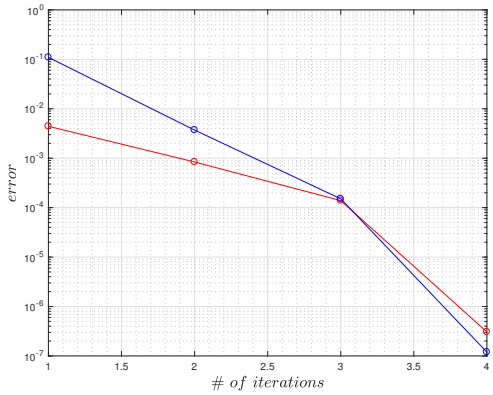


(b) F–δ graph using `LineSearch`

Figure 2.5: Force–displacement curve for the arch under dead load at the center modeled with a NeoHookean material.



(a) Convergence plot without using `LineSearch`



(b) Convergence plot when using `LineSearch`

Figure 2.6: Error curve for the arch under dead load at the center modeled with a NeoHookean material.

3 Implementation of a material model

The code you are given implements a plane-strain finite element method for finite deformation elasticity. A compressible NeoHookean material is already in place (modeling a slightly porous rubber for instance), whose strain energy density (or hyper-elastic potential) is

$$W(\boldsymbol{C}) = \frac{1}{2}\lambda_0(\ln J)^2 - \mu_0 \ln J + \frac{1}{2}\mu_0(Tr(\boldsymbol{C}) - 3)$$

This constitutive model is isotropic. Note that, since we are considering plane strain, we can use a 2×2 reduced right Cauchy-Green deformation tensor and replace trace $\boldsymbol{C} - 3$ by trace $\boldsymbol{C} - 2$ in the above equation.

We want to consider now an anisotropic material, more specifically, a transversely isotropic material. We consider a material constitutive law for a rubber reinforced by fibers, all aligned in the same direction in such a way that perpendicular to the fibers, the material remains isotropic. The orientation of the fibers is given in the reference configuration by a unit vector $\boldsymbol{N}^{fib}$. Such a model depends on the principal invariants of $\boldsymbol{C}$, and additionally by the fourth invariant.

$$I_4(\boldsymbol{C}) = \boldsymbol{N}^{fib} \cdot \boldsymbol{C} \cdot \boldsymbol{N}^{fib} = C_{IJ} N_I^{fib} N_J^{fib}$$

More specifically,

$$W(\boldsymbol{C}) = \frac{1}{2}\mu_0(Tr(\boldsymbol{C}) - 3) - \mu_0 \ln J + \kappa \mathcal{G}(J) + c_0 \left\{ \exp\left[c_1 \left(\sqrt{I_4(\boldsymbol{C})} - 1 \right)^4 \right] \right\}$$

where μ_0,κ, c_0, c_1 are material parameters, and $\mathcal{G}(J)$ provides the volumetric response of the material. We consider,

$$\mathcal{G}(J) = \frac{1}{4}\left(J^2 - 1 - 2\ln J\right)$$

The last term in the strain energy function specifies the contribution to the deformation of the fibers, and as typical in biological fibers, with this model these become stiffer the more deformed they are.

(a) *Implement this material model into the code. The code is prepared for this model (`material=2`), including the definition of the material parameters in `preprocessing.m`.*

In order to implement this model in the code, it is necessary to develop some derivations to obtain the second Piola–Kirchhoff tensor and the constitutive tensor.

The second Piola–Kircchoff tensor can be obtained as $\boldsymbol{S} = 2\frac{\partial W}{\partial \boldsymbol{C}}$, so deriving from the provided expression of W, the tensor is

$$\boldsymbol{S} = 2\frac{\partial}{\partial \boldsymbol{C}}\left[\frac{1}{2}\mu_0(Tr(\boldsymbol{C}) - 3) - \mu_0 \ln J + \kappa\mathcal{G}(J) + c_0\left\{\exp\left[c_1\left(\sqrt{I_4(\boldsymbol{C})} - 1\right)^4\right]\right\}\right]$$

Then,

$$\boldsymbol{S} = 2\left[\frac{\mu_0}{2}\frac{\partial Tr(\boldsymbol{C})}{\partial \boldsymbol{C}} - \frac{\mu_0}{J}\frac{\partial J}{\partial \boldsymbol{C}} + \frac{\kappa}{4}\left(2J\frac{\partial J}{\partial \boldsymbol{C}} - \frac{2}{J}\frac{\partial J}{\partial \boldsymbol{C}}\right) + c_0\frac{\exp\left[{}_1(\sqrt{I_4} - 1)^4\right](\sqrt{I_4} - 1)^3}{\sqrt{I_4}}\frac{\partial I_4}{\partial \boldsymbol{C}}\right] \quad (9)$$

Considering the following relations and identities

$$\frac{\partial Tr(\boldsymbol{C})}{\partial \boldsymbol{C}} = \boldsymbol{I} \quad ; \quad \frac{\partial J}{\partial \boldsymbol{C}} = \frac{J}{2}\boldsymbol{C}^{-1} \quad ; \quad \frac{\partial I_4}{\partial C_{ij}} = N_i N_j$$

$$\frac{\partial G}{\partial J} = \frac{1}{2}\left(J - \frac{1}{J}\right) \quad ; \quad \frac{\partial C^{-1}_{ij}}{\partial C_{kl}} = -\frac{1}{2}(C^{-1}_{ik}C^{-1}_{jl} + C^{-1}_{il}C^{-1}_{jk})$$

the tensor can be written using index notation as

$$S_{ij} = \mu_0\frac{\partial C_{kk}}{\partial C_{ij}} - 2\mu_0\frac{1}{J}\frac{1}{2}JC^{-1}_{ij} + \kappa\frac{2}{2}\left(J - \frac{1}{J}\right)\frac{1}{2}JC^{-1}_{ij} + 4c_0c_1\frac{\exp\left[{}_1(\sqrt{I_4} - 1)^4\right](\sqrt{I_4} - 1)^3}{\sqrt{I_4}}N_iN_j \quad (10)$$

and after some algebraic manipulation, the tensor $\boldsymbol{S}$ can be expressed using as

$$\boldsymbol{S}_{ij} = \mu_0\left(\delta_{ij} - C^{-1}_{ij}\right) + \frac{\kappa}{2}\left(J^2 - 1\right) - C^{-1}_{ij} + 4c_0c_1\frac{\exp\left[{}_1(\sqrt{I_4} - 1)^4\right](\sqrt{I_4} - 1)^3}{\sqrt{I_4}}N_iN_j \quad (11)$$

Following the same procedure, the constitutive tensor $\mathbb{C}$ can be derived as

$$\mathbb{C}_{ijkl} = 2\frac{\partial S_{ij}}{\partial C_{kl}}$$

So from the previous expression for the second Piola–Kirchhoff tensor

$$\begin{aligned}\mathbb{C}_{ijkl} = 0 &+ \kappa 2J\frac{1}{2}JC^{-1}_{kl}C^{-1}_{ij} - \frac{\kappa}{2}\left(J^2 - 1\right)\left(C^{-1}_{ik}C^{-1}_{jl} + C^{-1}_{il}C^{-1}_{jk}\right) + \mu_0\left(C^{-1}_{ik}C^{-1}_{jl} + C^{-1}_{il}C^{-1}_{jk}\right) \\ &+ 8c_0c_1\frac{(\sqrt{I_4} - 1)^2}{I_4}\exp\left[c_1(\sqrt{I_4} - 1)^4\right]\left(2c_1(\sqrt{I_4} - 1)^4 + 3\sqrt{I_4} - \frac{\sqrt{I_4} - 1}{2\sqrt{I_4}}\right)N_iN_jN_kN_l \quad (12)\end{aligned}$$

As before, operating and doing some algebraic manipulation, a simplified expression can be obtained:

$$\begin{aligned}\mathbb{C}_{ijkl} = \kappa J^2C^{-1}_{kl}C^{-1}_{ij} - \frac{\kappa}{2}\left(J^2 - 1 - \mu_0\right)\left(C^{-1}_{ik}C^{-1}_{jl} + C^{-1}_{il}C^{-1}_{jk}\right) \\ +8c_0c_1\frac{(\sqrt{I_4} - 1)^2}{I_4}\exp\left[c_1(\sqrt{I_4} - 1)^4\right]\left(2c_1(\sqrt{I_4} - 1)^4 + 3\sqrt{I_4} - \frac{\sqrt{I_4} - 1}{2\sqrt{I_4}}\right)N_iN_jN_kN_l\end{aligned}$$

(b) *Check the correctness (consistency test) of your implementation by running the script* `Check_Derivatives.m` *with* `material=2`*..This script checks the gradient of the energy (out-of-balance forces) and the Hessian of the energy (tangent stiffness matrix) by numerical differentiation. Check also that when solving a mechanical problem with this mode, Newton's method converges quadratically.*

The correctness of the derived model was tested using the provided function `Check_Derivatives`. The gradient yields no error, meaning that up to the second Piola–Kirchhoff tensor the model is correctly implemented. However, the Hessian showed errors slightly larger than the tolerance, which implies that there is a possible error in the implementation of the constitutive tensor.

The value of `h` (the step for the finite difference computation) was changed to eliminate the possibility of an error associated to the finite differences. The error in the Hessian was not affected, proving that the finite difference computation was not the source of the error.

The derivation of the constitutive tensor was throughly reviewed and the derivative of the exponential term was checked using symbolic calculus software. Nevertheless, no mistakes in the derivation were found and the same result was obtained every time.

In conclusion, we cannot find the source of the error in the Hessian. However, as the errors were small and the first part of the model was checked to be correctly implemented, the model was used to solve the following cases.

Finally, we present an image for the error after running one example for the suggested problem of the next section (`example=0`). As one can tell from Figure 3.1, the error convergence is merely linear. Note that the Newton–Raphson method is used to solve the non–linearity and one would expect quadratic convergence to the solution. Still, we recall that convergence of the Newton method relies on providing a quite good initial guess for the iterations. When this is not the case, the convergence deteriorates. Apart from that, there might be some error in the implementation of the material model, which makes the convergence of the method to be mainly linear. Though, after reviewing both the mathematical derivation and implementation several times, we did not find any major source of error.

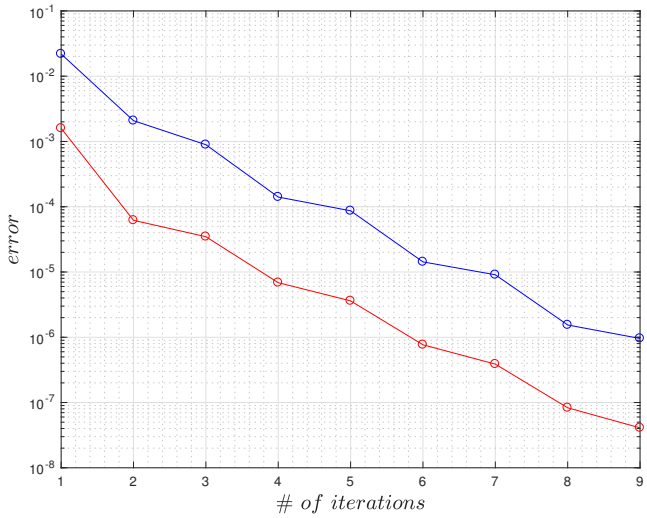


Figure 3.1: Convergence plot for the transverse isotropic material model for the `example=0`.

(c) *Solve* `example=0` *a dead load applied on an elastic block in tension, with a few representative orientations of the fibers. Consider* $\theta = 0$ *(fibers aligned with the loading direction),* $\theta = \pi/6$, $\theta = \pi/4$, $\theta = \pi/2$ *(fibers perpendicular to the loading direction), where*

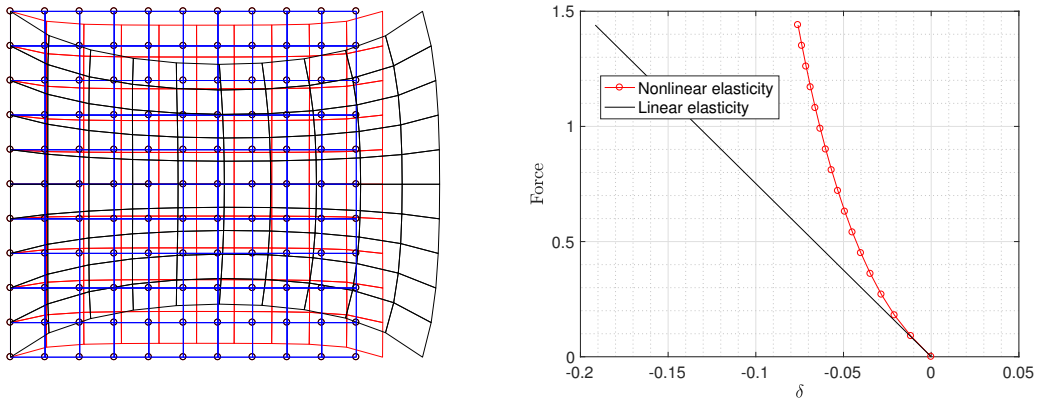
$$\boldsymbol{N}^{fib} = [\cos\theta, \sin\theta]^T$$

Explain the results from a mechanical viewpoint.

This section shows the results obtained when stretching a plate to 1.5 times its length for different fiber orientations. The distorted mesh and the force–displacement curve are presented for $\theta = 0$, $\theta = \frac{\pi}{6}$, $\theta = \frac{\pi}{4}$ and $\theta = \frac{\pi}{2}$. It is important to recall that the model represents a material reinforced with fibers aligned all in the same direction and that the material's behavior remains isotropic in the direction

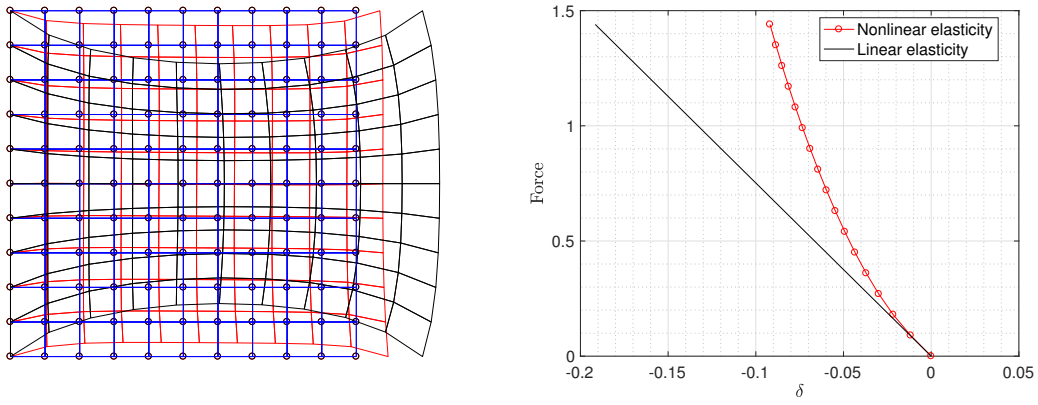
perpendicular to the fibers.

Figure 3.2 shows the results obtained for an angle of $\theta = 0$. In this case the orientation of the fibers coincides with the direction of the applied force. Figure 3.3 shows the results obtained for $\theta = \frac{\pi}{2}$, in which the force applied is perpendicular to the fiber's direction. As mentioned before, the material remains isotropic in the direction orthogonal to the fibers, meaning that the behavior should be the same for perpendicular directions. The results prove that the model correctly represents this characteristic, as the mesh deformation and the force–displacement are equivalent for $\theta = 0$ and $\theta = \frac{\pi}{2}$.



(a) Mesh (b) Force–displacement curve

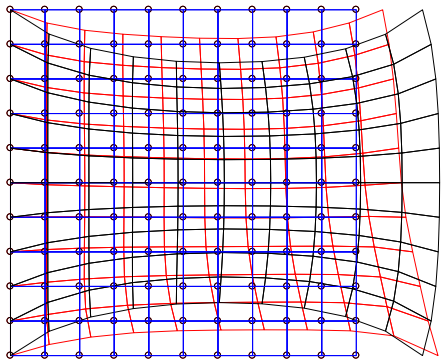
Figure 3.2: Mesh and force–displacement curve for $\theta = 0$



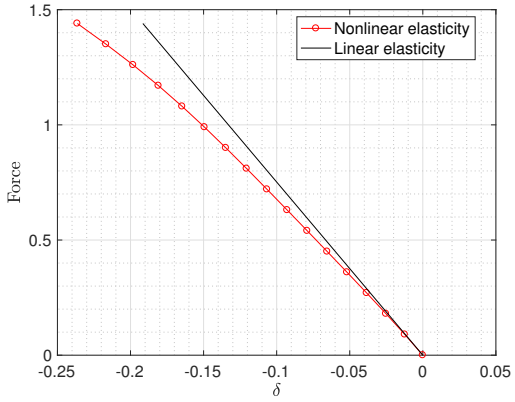
(a) Mesh (b) Force–displacement curve

Figure 3.3: Mesh and force–displacement curve for $\theta = \frac{\pi}{2}$

The following Figures 3.4 and 3.5 show the results for $\theta = \frac{\pi}{6}$ and $\theta = \frac{\pi}{4}$ respectively. The main particularity of this cases is that the fibers are now arbitrarily oriented with respect to the direction of the applied force. For this orientation of the fibers the material is less stiff than for the previous ones, as less force is required to generate the same displacement. This means that this directions are non-optimal if the material wants to be used to its maximum capacity. To use the material to its maximum capacity the applied load should be either aligned or perpendicular to the fibers orientation.

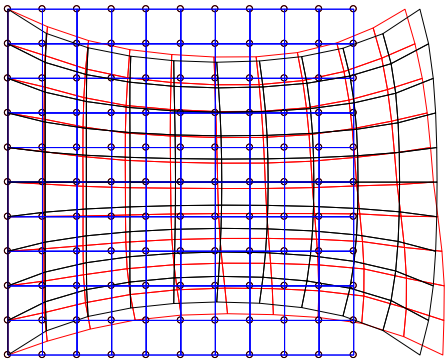


(a) Mesh

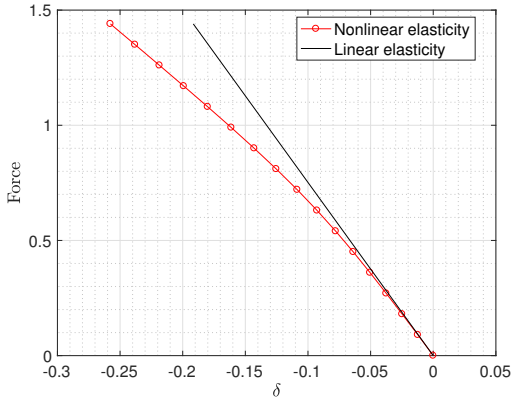


(b) Force–displacement curve

Figure 3.4: Mesh and force–displacement curve for $\theta = \frac{\pi}{6}$



(a) Mesh



(b) Force–displacement curve

Figure 3.5: Mesh and force–displacement curve for $\theta = \frac{\pi}{4}$

A Matlab Codes

Box 1: Kirchhoff Saint-Venant: Function 1

```
function [W]=KSV_1(C,lambda,mu,icode)
% Strain tensor and its trace
E=(C-[1 1 0])/2;
trE=E(1)+E(2);
% Square of strain tensor and its trace
E2=E.^2;
trE2=E2(1)+E2(2);

% Energy function
W=lambda/2*(trE)^2+mu*trE2;
end
```

Box 2: Kirchhoff Saint-Venant: Function 2

```
function [W,S]=KSV_2(C,lambda,mu,icode);
% Strain tensor and its trace
E=(C-[1 1 0])/2;
trE=E(1)+E(2);
% Square of strain tensor and its trace
E2=E.^2;
trE2=E2(1)+E2(2);

% Energy function
W=lambda/2*(trE)^2+mu*trE2;

% Second Piola-Kirchhoff tensor
S=[];
S=lambda*trE*[1 1 0]+2*mu*E;
end
```

Box 3: Kirchhoff Saint-Venant: Function 3

```
function [W,S,CC] = KSV_3(C,lambda,mu,icode)

% Strain tensor and its trace
E=(C-[1 1 0])/2;
trE=E(1)+E(2);
% Square of strain tensor and its trace
E2=E.^2;
trE2=E2(1)+E2(2);

% Energy function
W=lambda/2*(trE)^2+mu*trE2;

% Second Piola-Kirchhoff tensor
S = zeros(1,3);
S = lambda*trE*[1 1 0]+2*mu*E;

% Constitutive tensor
CC = zeros(3);

CC(1,1) = lambda+2*mu;
CC(1,2) = lambda;
CC(1,3) = 0;
CC(2,1) = lambda;
```

```
CC(2,2) = lambda+2*mu;
CC(2,3) = 0;
CC(3,1) = 0;
CC(3,2) = 0;
CC(3,3) = 2*mu;
end
```

Box 4: Modified Kirchhoff Saint-Venant: Function 1

```
function [W]=KSVmodified_1(C,lambda,mu,icode)
% Strain tensor and its trace
E = (C-[1 1 0])/2;
trE = E(1)+E(2);
% Square of strain tensor and its trace
E2 = E.^2;
trE2 = E2(1)+E2(2);

% Jacobian
J2 = C(1)*C(2)-C(3)*C(3);
J = sqrt(J2);

% Energy function
W = lambda/2*(log(J))^2+mu*trE2;


end
```

Box 5: Modified Kirchhoff Saint-Venant: Function 2

```
function [W,S]=KSVmodified_2(C,lambda,mu,icode);
% Strain tensor and its trace
E = (C-[1 1 0])/2;
trE = E(1)+E(2);
% Square of strain tensor and its trace
E2 = E.^2;
trE2 = E2(1)+E2(2);

% Jacobian
J2 = C(1)*C(2)-C(3)*C(3);
J = sqrt(J2);

% Energy function
W = lambda/2*(log(J))^2+mu*trE2;

% Inverse of C
C_inv = [C(2) C(1) -C(3)]/J2;

% Second Piola-Kirchhoff tensor
S = zeros(1,3);
S(1) = mu*(C(1)-1)+lambda*log(J)*C_inv(1);
S(2) = mu*(C(2)-1)+lambda*log(J)*C_inv(2);
S(3) = mu*(C(3))+lambda*log(J)*C_inv(3);

end
```

Box 6: Modified Kirchhoff Saint-Venant: Function 3

```
function [W,S,CC]=KSVmodified_3(C,lambda,mu,icode);
% Strain tensor and its trace
```

```
E = (C-[1 1 0])/2;
trE = E(1)+E(2);
% Square of strain tensor and its trace
E2 = E.^2;
trE2 = E2(1)+E2(2);

% Jacobian
J2 = C(1)*C(2)-C(3)*C(3);
J = sqrt(J2);

% Energy function
W = lambda/2*(log(J))^2+mu*trE2;

% Inverse of C
C_inv = [C(2) C(1) -C(3)]/J2;

% Second Piola-Kirchhoff tensor
S = zeros(1,3);
S(1) = mu*(C(1)-1)+lambda*log(J)*C_inv(1);
S(2) = mu*(C(2)-1)+lambda*log(J)*C_inv(2);
S(3) = mu*(C(3))+lambda*log(J)*C_inv(3);

% Constitutive tensor
CC = zeros(3,3);

aux1=2*mu;
aux2=lambda;
aux3=-lambda*log(J);

CC(1,1) = aux1 + aux2*C_inv(1)*C_inv(1)-...
    aux3*(C_inv(1)*C_inv(1)+C_inv(1)*C_inv(1));

CC(1,2) = aux2*C_inv(2)*C_inv(1) - ...
    aux3*(C_inv(3)*C_inv(3)+C_inv(3)*C_inv(3));

CC(1,3) = aux2*C_inv(3)*C_inv(1) -...
    aux3*(C_inv(1)*C_inv(3)+C_inv(1)*C_inv(3));

CC(2,2) = aux1+ aux2*C_inv(2)*C_inv(2)-...
    aux3*(C_inv(2)*C_inv(2)+C_inv(2)*C_inv(2));

CC(2,3) = aux2*C_inv(3)*C_inv(2) -...
    aux3*(C_inv(2)*C_inv(3)+C_inv(2)*C_inv(3));

CC(3,3) = aux1+ aux2*C_inv(3)*C_inv(3)-...
    aux3*(C_inv(1)*C_inv(2)+C_inv(3)*C_inv(3));

CC(2,1) = CC(1,2);

CC(3,1) = CC(1,3);

CC(3,2) = CC(2,3);
end
```

Box 7: Transversely isotropic model: Function 1

```
function [W]=transv_isotr_1(C,c0,c1,kappa,mu,N)
% Auxiliar variables necessary for the computations
J2 = C(1)*C(2)-C(3)*C(3);                % Square of Jacobian
```

```
J = sqrt(J2);                                   % Jacobian
logJ = log(J);                                  % Logarithm of the Jacobian
traceC = C(1)+C(2);                             % Trace of C


% Volumetric response of the material
G = 0.25*(J2-1-2*logJ);

% Fourth invariant
I4 =  C(1)*N(1)*N(1)+C(2)*N(2)*N(2)+2*C(3)*N(1)*N(2);

% Energy function
W =  mu*(traceC-2)/2-mu*logJ+kappa*G+c0*(exp(c1*(sqrt(I4)-1)^4)-1);
end
```

Box 8: Transversely isotropic model: Function 2

```
function [W,S]=transv_isotr_2(C,c0,c1,kappa,mu,N)
% Auxiliar variables necessary for the computations
J2 = C(1)*C(2)-C(3)*C(3);                       % Square of Jacobian
J = sqrt(J2);                                   % Jacobian
logJ = log(J);                                  % Logarithm of the
    Jacobian
Cinv = [C(2) C(1) -C(3)]/J2;                    % Inverse of C
Id = [1 1 0];                                   % Identity matrix in voigt
     notation
traceC = C(1)+C(2);                             % Trace of C
Ntensor = [N(1)*N(1), N(2)*N(2), N(1)*N(2)]; % Derivative of I4


% Volumetric response of the material
G = 0.25*(J2-1-2*logJ);

% Fourth invariant
I4 =  C(1)*N(1)*N(1)+C(2)*N(2)*N(2)+2*C(3)*N(1)*N(2);

% Energy function
W =  mu*(traceC-2)/2-mu*logJ+kappa*G+c0*(exp(c1*(sqrt(I4)-1)^4)-1);

% Second Piola-Kirchhoff tensor
S = zeros(1,3);
S = mu*(Id-Cinv)+kappa/2*(J2*Cinv-Cinv)+4*c0*c1*((sqrt(I4)-1)^3/sqrt(I4
    )*exp(c1*(sqrt(I4)-1)^4))*Ntensor;
end
```

Box 9: Transversely isotropic model: Function 3

```
function [W,S,CC]=transv_isotr_3(C,c0,c1,kappa,mu,N)
% Auxiliar variables necessary for the computations
J2 = C(1)*C(2)-C(3)*C(3);                       % Square of Jacobian
J = sqrt(J2);                                   % Jacobian
logJ = log(J);                                  % Logarithm of the
    Jacobian
Cinv = [C(2) C(1) -C(3)]/J2;                    % Inverse of C
Id = [1 1 0];                                   % Identity matrix in voigt
     notation
traceC = C(1)+C(2);                             % Trace of C
Ntensor = [N(1)*N(1), N(2)*N(2), N(1)*N(2)]; % Derivative of I4

```

```
11
12 % Volumetric response of the material
13 G = 0.25*(J2-1-2*logJ);
14
15 % Fourth invariant
16 I4 =  C(1)*N(1)*N(1)+C(2)*N(2)*N(2)+2*C(3)*N(1)*N(2);
17
18 % Energy function
19 W =  mu*(traceC-2)/2-mu*logJ+kappa*G+c0*(exp(c1*(sqrt(I4)-1)^4)-1);
20
21 % Second Piola-Kirchhoff tensor
22 S = mu*(Id-Cinv)+kappa/2*(J2*Cinv-Cinv)+4*c0*c1*((sqrt(I4)-1)^3/sqrt(I4
   )*exp(c1*(sqrt(I4)-1)^4))*Ntensor;
23
24 % Constitutive tensor
25 const=(8*c0*c1*((sqrt(I4)-1)^2)/I4)*exp(c1*((sqrt(I4)-1)^4))*(2*c1*((
   sqrt(I4)-1)^4)...
26     +3*sqrt(I4)-((sqrt(I4)-1)/(2*sqrt(I4))));
27
28 dC11=-1/2*(Cinv(1)*Cinv(1)+Cinv(1)*Cinv(1));
29 CC(1,1)=(-2*mu + kappa*(J2-1))*dC11 + kappa*J2*Cinv(1)*Cinv(1) +const*
   Ntensor(1)*Ntensor(1);
30
31 dC22=-1/2*(Cinv(2)*Cinv(2)+Cinv(2)*Cinv(2));
32 CC(2,2)=(-2*mu + kappa*(J2-1))*dC22 + kappa*J2*Cinv(2)*Cinv(2) +const*
   Ntensor(2)*Ntensor(2);
33
34 dC33=-1/2*(Cinv(1)*Cinv(2)+Cinv(3)*Cinv(3));
35 CC(3,3)=(-2*mu + kappa*(J2-1))*dC33 + kappa*J2*Cinv(3)*Cinv(3) +const*
   Ntensor(3)*Ntensor(3);
36
37 dC12=-1/2*(Cinv(3)*Cinv(3)+Cinv(3)*Cinv(3));
38 CC(1,2)=(-2*mu + kappa*(J2-1))*dC12 + kappa*J2*Cinv(2)*Cinv(1) +const*
   Ntensor(1)*Ntensor(2);
39
40 dC13=-1/2*(Cinv(1)*Cinv(3)+Cinv(1)*Cinv(3));
41 CC(1,3)=(-2*mu + kappa*(J2-1))*dC13 + kappa*J2*Cinv(1)*Cinv(3) +const*
   Ntensor(1)*Ntensor(3);
42
43 dC23=-1/2*(Cinv(2)*Cinv(3)+Cinv(2)*Cinv(3));
44 CC(2,3)=(-2*mu + kappa*(J2-1))*dC23 + kappa*J2*Cinv(2)*Cinv(3) +const*
   Ntensor(2)*Ntensor(3);
45
46 CC(2,1)=CC(1,2);
47 CC(3,1)=CC(1,3);
48 CC(3,2)=CC(2,3);
49
50 end
```