Assignment 3
Finite Deformation Nonlinear Elasticity
Computational Solid Mechanics
Master of Science in Computational Mechanics 2016

Paris Dilip Mulye

June 9, 2016

Kirchhoff Saint-Venant Material Model

Problem 1

Given elastic energy function is,

$$\begin{aligned} W(\epsilon) &= \frac{\lambda}{2}(\epsilon_{ii})^2 + \mu\epsilon_{jk}\epsilon_{jk} \\ \sigma_{pq} &= \frac{\partial W}{\partial \epsilon_{pq}} = \lambda\epsilon_{ii}\frac{\partial \epsilon_{ii}}{\partial \epsilon_{pq}} + 2\mu\frac{\partial \epsilon_{jk}}{\partial \epsilon_{pq}}\epsilon_{jk} \\ &= \lambda\,\epsilon_{ii}\,\delta_{ip}\,\delta_{iq} + 2\mu\,\delta_{jp}\,\delta_{kq}\,\epsilon_{jk} \end{aligned}$$

Using property of δ operator, $\delta_{ab}M_{ac} = M_{bc}$,

$$\begin{aligned} \sigma_{pq} &= \lambda\epsilon_{ii}\delta_{pq} + 2\mu\epsilon_{pq} \\ \boldsymbol{\sigma} &= \lambda\, tr(\boldsymbol{\epsilon})\boldsymbol{I} + 2\mu\boldsymbol{\epsilon} \end{aligned}$$

Which agrees with usual linear elasticity expression.

Problem 2

Given elastic energy function is,

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

For law to be isotropic, $W(\boldsymbol{QF}) = W(\boldsymbol{F})$ which implies that the energy potential is rotation independent.

$$\begin{aligned} \boldsymbol{E} &= \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I}) = \frac{1}{2}(\boldsymbol{F^T F} - \boldsymbol{I}) \\ &= \frac{1}{2}(\boldsymbol{F^T Q^T Q F} - \boldsymbol{I}) = \frac{1}{2}(\boldsymbol{F^T F} - \boldsymbol{I}) \end{aligned}$$

This proves that $\boldsymbol{E}$ is independent of rotations. Since, given $W(\boldsymbol{E})$ is solely a function of $\boldsymbol{E}$, it can be concluded that given model is isotropic.

Problem 3

$$\begin{aligned} W(\boldsymbol{E}) &= \frac{\lambda}{2}(tr\boldsymbol{E})^2 + \mu\, tr(\boldsymbol{E^2}) \\ tr(E_{PQ}E_{QR}) &= E_{PQ}E_{QP} \end{aligned}$$

The second Piola Kirchhoff Tensor is given as,

$$\begin{aligned} S_{IJ} &= \frac{\partial W}{\partial E_{IJ}} = \frac{\lambda}{2}\frac{\partial E_{KK}^2}{E_{IJ}} + \mu\frac{\partial(E_{PQ}E_{QP})}{E_{IJ}} \\ &= \lambda E_{KK}\frac{\partial E_{KK}}{\partial E_{IJ}} + \mu\frac{\partial E_{PQ}}{E_{IJ}}E_{QP} + \mu E_{PQ}\frac{\partial E_{QP}}{\partial E_{IJ}} \end{aligned}$$

Using symmetry of E and $\delta_{ab}M_{ac} = M_{bc}$,

$$\begin{aligned} S_{IJ} &= \lambda E_{KK}\delta_{KI}\delta_{KJ} + 2\mu E_{PQ}\delta_{QI}\delta_{PJ} \\ &= \lambda E_{KK}\delta_{IJ} + 2\mu E_{IJ} \\ \boldsymbol{S} &= \lambda(tr\boldsymbol{E})\boldsymbol{I} + 2\mu\boldsymbol{E} \end{aligned}$$

Problem 4

Using the given deformation map,

$$\boldsymbol{F} = \frac{d\boldsymbol{x}}{d\boldsymbol{X}} = \begin{bmatrix} \Lambda & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{C} = \boldsymbol{F^T F} = \begin{bmatrix} \Lambda^2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$\begin{aligned} \boldsymbol{S} &= \lambda(tr\boldsymbol{E})\boldsymbol{I} + 2\mu\boldsymbol{E} \\ &= \tfrac{1}{2}\begin{bmatrix} (\Lambda^2-1)(\lambda+2\mu) & 0 & 0 \\ 0 & (\Lambda^2-1)\lambda & 0 \\ 0 & 0 & (\Lambda^2-1)\lambda \end{bmatrix} \\ \boldsymbol{P} &= \boldsymbol{FS} \\ &= \tfrac{1}{2}\begin{bmatrix} \Lambda(\Lambda^2-1)(\lambda+2\mu) & 0 & 0 \\ 0 & (\Lambda^2-1)\lambda & 0 \\ 0 & 0 & (\Lambda^2-1)\lambda \end{bmatrix} \end{aligned}$$

$$\boldsymbol{P_{11}} = \frac{\Lambda(\Lambda^2-1)(\lambda+2\mu)}{2}$$

Figure 1 shows normalized value of P_{11}.

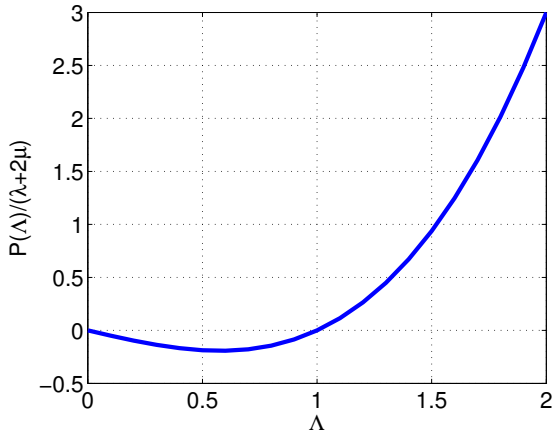

Figure 1: P_{11} vs Λ

Problem 5

P is not monotonic, which is evident from Figure 1. The optima of function are at,

$$\frac{dP}{d\Lambda} = \frac{(3\Lambda^2 - 1)(\lambda + 2\mu)}{2} = 0$$

which occurs at $\Lambda = \pm\frac{1}{\sqrt{3}}$. These values are independent of material parameters λ and μ. Since, negative values of Λ are not feasible, only one optima needs to be looked at which is at $\Lambda = \frac{1}{\sqrt{3}}$. Since $\frac{d^2P}{d\Lambda^2} => 0$ at this point, this is a minima. This is also evident from Figure 1, that the function has a minimum value at $\Lambda = \frac{1}{\sqrt{3}} \approx 0.58$.

Therfore, critical value at which model fails, since $\frac{dP}{d\Lambda} = 0$ is at $\Lambda = \frac{1}{\sqrt{3}}$. As explained above, this value is independent of material parameters λ and μ.

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

Hence, $J \to 0^+$ implies $\lambda \to 0^+$. This makes the energy potential $W \to 0^+$. Physically speaking, as one compresses material incrementally, more energy should be stored (increase in the potential) and in the limiting case, where a material becomes concentrated at single point, W should approach $+\infty$. This is not observed in Kirchhoff Saint-Venant Model.

Problem 6

For modified model,

$$W(\boldsymbol{E}) = \frac{\lambda}{2}(\ln J)^2 + \mu\, tr(\boldsymbol{E^2})$$
$$= \frac{\lambda}{2}(\ln \Lambda)^2 + \mu\left(\frac{\Lambda^2 - 1}{2}\right)^2$$

For this model, as $J \to 0^+$ or $\lambda \to 0^+$, $W \to +\infty$. Therefore, modified model removes this limitation of the original Kirchhoff Saint-Venant Model. As shown in Figure 2, the potential should have a minimum at natural length $\Lambda = 1$. Also, it can be seen that $W \to +\infty$ as $\lambda \to 0^+$ for modified model which is not the case for original model.

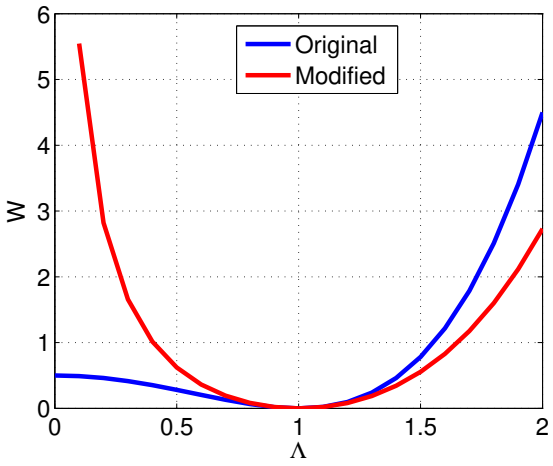

Figure 2: W vs Λ

Problem 7

The material model has been implemented, code is in Appendix. To observe the effects of only material nonlinearity (and not geometrical nonlinearity), Y DoF of all nodes was fised. Problem 1 in Matlab code was simulated. Λ was varied from 1 to 0.01. The displacement controlled simulation results are shown in Figure 3.

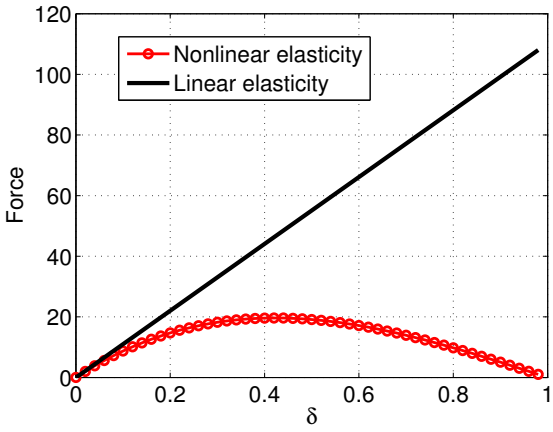

Figure 3: Forces vs δ

As expected, the force reaches a maximum value at $\Lambda = \frac{1}{\sqrt{3}}$, $\delta = 1 - \Lambda = 0.43$. points between $\delta = 0$ to $\delta = \frac{1}{\sqrt{3}}$ are stable. But if sufficient energy is provided (increment in F), the equilibrium point shifts on the other side of maxima, which would be unstable. Therefore, the model would snap to some unknown configuration which indicated the instability.

To verify this hypothesis, a force controlled model was simulated, with F varying from 0 to 20 in compression. As expected, the solution was unstable at point close to this. Both displacement and forced controlled simulations are plotted in the same plot (Figure 4) for comparison. (Note, displacement controlled solution used for comparison was simulated only until $\delta = 0.8$)

Since, there is no other feasible solution, the model snaps inside out ($\delta > 1$). This final configuration is shown in Figure 5, where blue mesh is initial configuration and red is final.

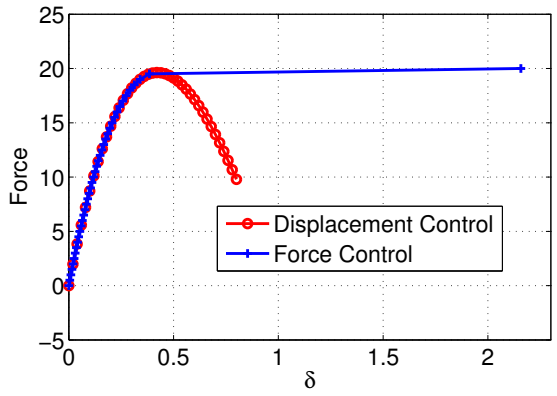

Figure 4: Compressive Force vs Compressive Displacement

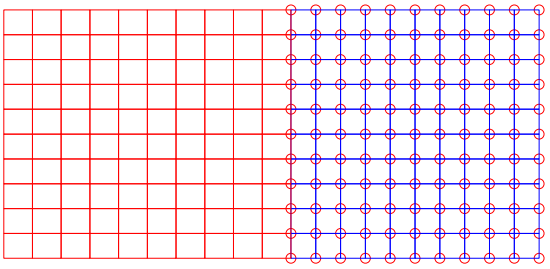

Figure 5: Final Configuration: Force Control

Implementation of Line-Search

Line Search has been implemented in MATLAB. The relevant code can be found in Appendix. The direction of p was reversed if it was found to move in the uphill direction where p was defined as,
`p = -grad_E* inv(Hess_E)`. This was checked using `sum(grad_E.*p) > 0`, if found true, direction was reversed.

The results for slender beam under compressive displacement load are shown in Figure 6 which depicts solution without line search implementation (blue) and with line search (red). It is evident that line search method was able to capture buckling. Even after adding random perturbations, solver without line search was unable to capture the buckling effect.

Both the methods, try to approach the minimum of energy function, but since line search is searching along direction of gradient for a longer distance (usage of t), there is a greater chance that it finds the global minima. On the other hand, there is a high chance that solver without line search will not find the global minima. (but will reach local minima, or even local maxima, if positive definiteness of Hessian is not checked)

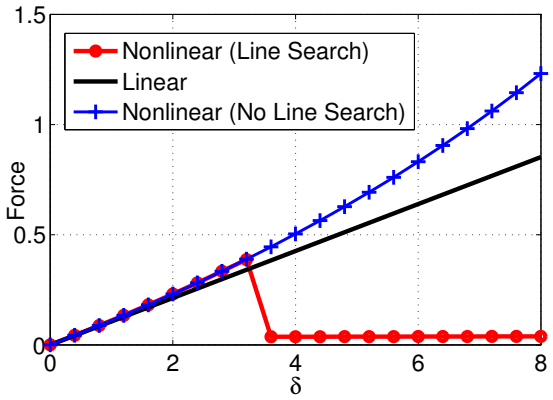

Figure 6: Line Search Effect

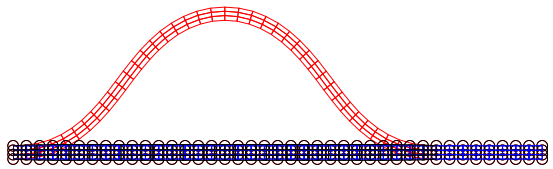

Figure 7: Solution With Line Search

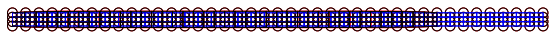

Figure 8: Solution Without Line Search

A similar observation was made for problem with deflection of Arch (displacement controlled). Solver without line search cannot capture buckling and snaps to infeasible configuration which can be seen in Figure 11. On the other hand, line search can capture the buckling effect (Figure 10). Figure 9 shows load-displacement of line search.

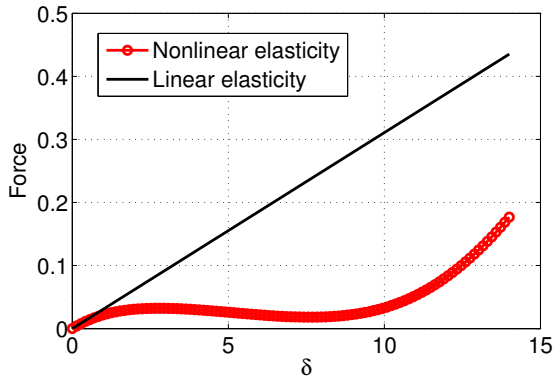

Figure 9: Load vs Displacement, Line Search

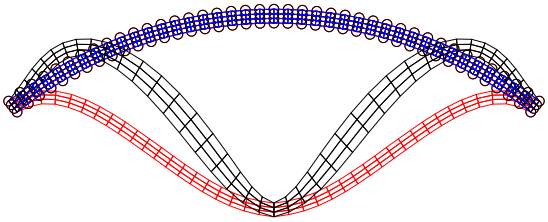

Figure 10: Solution With Line Search

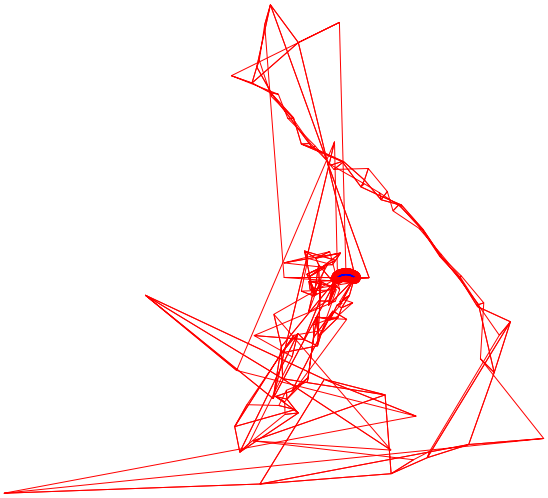

Figure 11: Solution Without Line Search

Implementation of a Material Model

Given, W, it is needed to calculate S and CC which are second Piola Stress Tensor and Tangent Modulus respectively. The derivation of these quantities is as follows. Some preliminaries are,

$$I_1 = C_{II}, \qquad \frac{\partial I_1}{\partial C} = I$$

$$I_3 = det(C), \qquad \frac{\partial I_3}{\partial C} = I_3 C^{-1}$$

$$I_4 = n_I n_J C_{IJ}$$

$$\frac{\partial I_4}{\partial C} = \delta_{IK}\delta_{JL} n_I n_J = \begin{bmatrix} n_1 n_1 & n_2 n_1 \\ n_2 n_1 & n_2 n_2 \end{bmatrix} = N$$

$$\frac{\partial C^{-1}}{\partial C} = -0.5(C^{-1}_{IK} C^{-1}_{JL} + C^{-1}_{IL} C^{-1}_{JK}) = -0.5A$$

$$P = \exp(c_1(\sqrt{I_4} - 1)^4)$$

$$\frac{\partial P}{\partial C} = 2c_1 \frac{P(\sqrt{I_4} - 1)^3}{\sqrt{I_4}} N = (\text{P\_1\_fact})N$$

$$\frac{\partial (P\_1\_fact)}{\partial C} = (\text{P\_2\_fact})N$$

$$S = 2\frac{\partial W}{\partial C}, \qquad CC = 4\frac{\partial^2 W}{\partial C^2}$$

Using the above expressions and given potential,

$$\begin{aligned} W =& 0.5\mu(I_1 - 2) - \mu \ln(\sqrt{I_3}) + \\ & 0.25\kappa(I_3 - 1 - 2\ln(\sqrt{I_3})) + c_0(P-1) \\ S =& 2(0.5\mu I - 0.5\mu C^{-1} + 0.25\kappa I_3 C^{-1} \\ & - 0.25\kappa C^{-1} + c_0(\text{P\_1\_fact})N) \\ CC =& 4(0.25\mu A - 0.125\kappa I_3 A + \\ & 0.25\kappa C^{-1} I_3 C^{-1} + 0.125\kappa A \\ & + c_0(\text{P\_2\_fact})NN) \end{aligned}$$

Where, NN means $N_{IJ}N_{KL}$, $N_{IJ} = N(I,J)$.

Part A

The model has been implemented in MATLAB, code can be found in Appendix. The material parameters used for the simulations are as follows,

```
mod1.mu=1;
mod1.kappa = 100;
mod1.c0 = 80;
mod1.c1 = 5;
```

As asked, four simulations were performed for θ = 90, 45, 30 and 0. For all simulations, line search with Newton-Raphson method was used. The example used for solving was Example 1 from MATLAB code. Tensile applied load was 0:0.09:3.

Part B

The derivative check was performed and was found to be successful. The convergence plot for error in f using Newton-Raphson (with Line Search) is shown in Figure 12. The corresponding data points are, x=[1, 2, 3, 4] and y=[-1.56, -1.74, -4.07, -7.36]. From this, it can be seen that when the iterations double from 2 to 4, the error reduces almost 4 times. This implies a quadratic convergence.

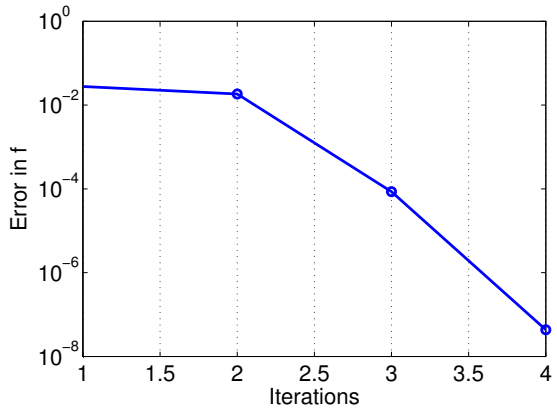

Figure 12: Convergence Plot

Part C

As explained previously, four simulations were performed. The results are documented from Figure 13 to Figure 20.

Angle = 90

In this case, material is stronger in Y than in X. But as load increases, material fibers get squeezed in form of diamonds, in order to resist load. Once this transformation of shape gets completed, the stiffness of material changes. This can be seen from jump in the slope of load vs displacement curve. The deformed shape confirms the hypothesis about change in shape. The decrease in the stiffness can be accounted by the fact, once perturbed, the shear components will come into picture and due to very less shear resistance, they would readily deform, reducing the overall stiffness.

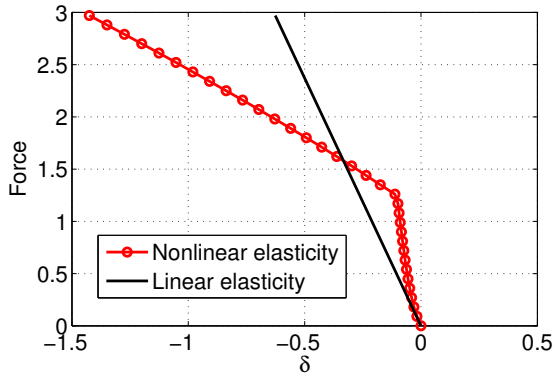

Figure 13: Load vs Displacement, $\theta = 90$

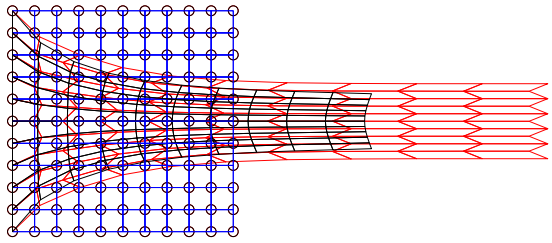

Figure 14: Final Configuration, $\theta = 90$

Angle = 45

In this case, material fibers are strong along 45 degrees with X axis. So Applied load along $[1\ 0]^T$ can be split using vectors along 45 degree line $a = [1\ 1]^T$ and one perpendicular to it $b = [1\ -1]^T$. Since, stiffness along vector a is more, there would be less deflection, but the stiffness is less along vector b, which results in more deformation along b. This is evident from shape plot.

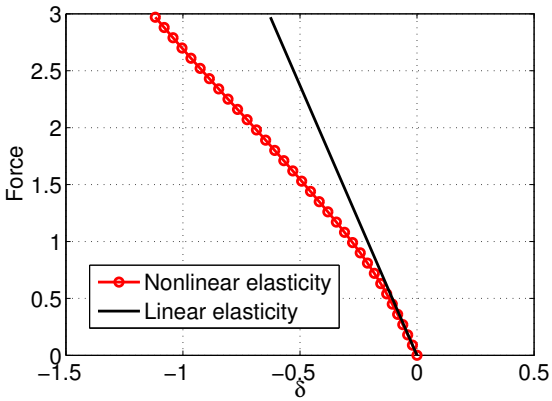

Figure 15: Load vs Displacement, $\theta = 45$

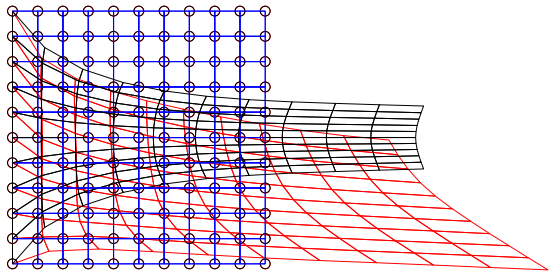

Figure 16: Final Configuration, $\theta = 45$

Angle = 30

In this case, material fibers are strong along 30 degrees with X axis. So Applied load along $[1\ 0]^T$ can be split using vectors along 30 degree line $a = [0.8660\ 0.5000]^T$ and one perpendicular to it $b = [0.5000\ -0.8660]^T$. Since, stiffness along vector a is more, there would be less deflection, but the stiffness is less along vector b, which results in more deformation along b. This is evident from shape plot.

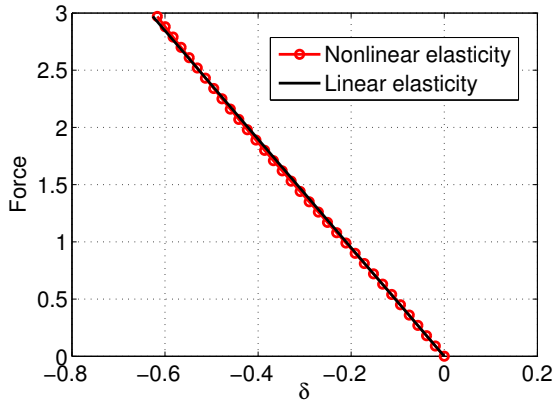

Figure 17: Load vs Displacement, $\theta = 30$

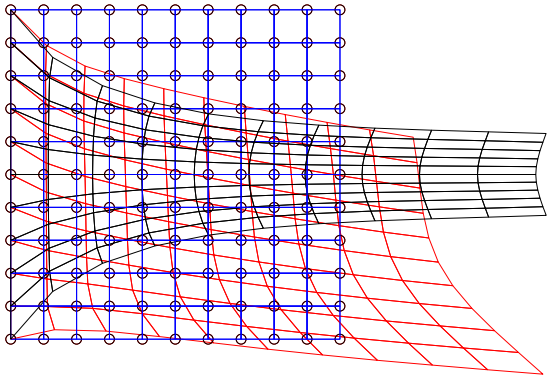

Figure 18: Final Configuration, $\theta = 30$

Angle = 0

In this case, material fibers are strong along X direction, so compared to all the cases, displacements should be less in X direction.

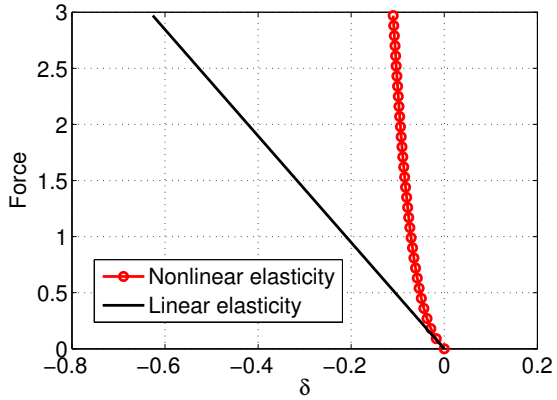

Figure 19: Load vs Displacement, $\theta = 0$

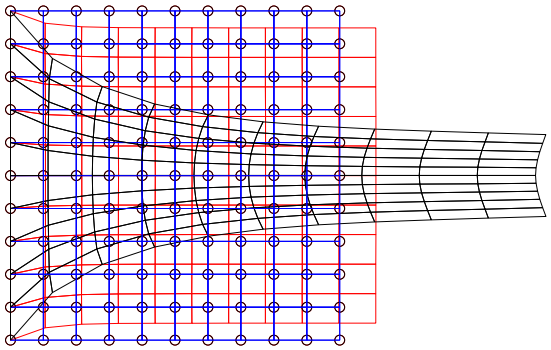

Figure 20: Final Configuration, $\theta = 0$

Comparison

Except, Angle =90, all other graphs could be explained with simple logic, As angle increases from 0 to 90, the stiffness along X direction decreases, which is evident from the decreasing slopes of curves. Possible casue of anomaly in the behavior of angle=90 case, is mentioned above already.

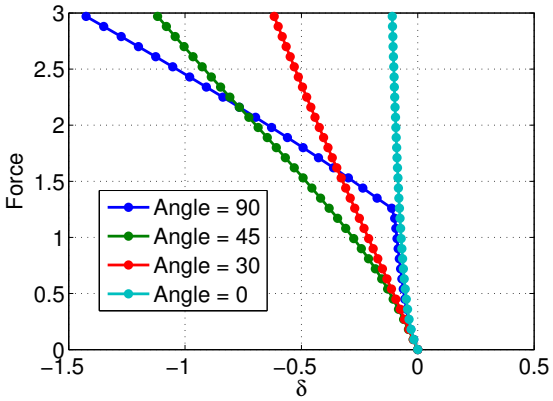

Figure 21: Load vs Displacement, $\theta = 90,42,30,0$

Appendix

Kirchhoff Saint-Venant Material Model

The material model `NeoHookean_3` was modified. Also, the calls to `NeoHookean_1` and `NeoHookean_2` were redirected to `NeoHookean_3`.

```
function [W,S,CC]=NeoHookean_3...
                  (C,lambda,mu,icode)
CMat = [C(1), C(3); C(3), C(2)];
E = 0.5*(CMat-eye(2));
W = lambda/2*trace(E) + mu*trace(E*E);
S = lambda*eye(2)*trace(E) + 2*mu*E;
S = [S(1,1), S(2,2), S(1,2)];
CC=zeros(3);
CC(1,1) = lambda+2*mu;
CC(2,2) = lambda+2*mu;
CC(1,2) = lambda;
CC(2,1) = CC(1,2);
CC(3,3) = 2*mu;
end
```

Line-Search

Line Search with Newton-Raphson was implemented in file `Equilibrate` as a child of command `switch options.method`

```
case 1,   %Newton-Raphson
  iter=0;
  err_x=100;
  err_f=100;
  [Ener,grad_E,Hess_E]=Ener_short(x_short,3);
  while (iter<=options.n_iter_max) & ...
        ((err_f>options.tol_f))
  [Ener,grad_E,Hess_E]=Ener_short(x_short,3);
  iter=iter+1;
  p = -Hess_E\grad_E;
  if sum(grad_E.*p) > 0
     p = -p;
  end
  t=1;
  opts=optimset('TolX',options.TolX,...
     'MaxIter',options.n_iter_max_LS);
  t = fminbnd(@(t) Ener_1D(t,x_short,p),...
      0,10,opts);
  x_short=x_short+t*p;
  err_x=norm(t*p)/norm(x_short);
  err_f=norm(grad_E);
  err_plot=[err_plot err_x];
  err_plot1=[err_plot1 err_f];
end
```

Transversely Isotropic Material

A new file was written to simulate this material model.

```
function [W,S,CC]=transv_isotr_3...
(C,c0,c1,kappa,mu,N_fib)
C = [C(1), C(3); C(3), C(2)];
Cinv = eye(2)/C;
I1 = trace(C);
I3 = det(C);
I4 = N_fib'*C*N_fib;
J = sqrt(I3);
N1 = N_fib(1);
N2 = N_fib(2);
N = [N1*N1, N2*N1; N1*N2, N2*N2];
P = exp(c1*(sqrt(I4)-1)^4);
P_1_fact = 4*c1*P*(sqrt(I4)-1)^3 /2/sqrt(I4);
P_2_fact = 2*c1*(sqrt(I4)*(P_1_fact*(sqrt(I4)-1)^3...
   + 3*P*(sqrt(I4)-1)^2/2/sqrt(I4))...
   - P*((sqrt(I4)-1)^3)/2/sqrt(I4))/I4;
%values for fourth order tensor N_i N_j N_k N_l
N11N11 = N(1,1)*N(1,1);
N22N22 = N(2,2)*N(2,2);
N12N12 = N(1,2)*N(1,2);
N11N22 = N(1,1)*N(2,2);
N11N12 = N(1,1)*N(1,2);
N22N12 = N(2,2)*N(1,2);
%values of derivative of Cinv wrt C
A1111 = Cinv(1,1)*Cinv(1,1) + Cinv(1,1)*Cinv(1,1);
A2222 = Cinv(2,2)*Cinv(2,2) + Cinv(2,2)*Cinv(2,2);
A1212 = Cinv(1,1)*Cinv(2,2) + Cinv(1,2)*Cinv(2,1);
A1122 = Cinv(1,2)*Cinv(1,2) + Cinv(1,2)*Cinv(1,2);
A1112 = Cinv(1,1)*Cinv(1,2) + Cinv(1,2)*Cinv(1,1);
A2212 = Cinv(2,1)*Cinv(2,2) + Cinv(2,2)*Cinv(2,1);
W = 0.5*mu*(I1-2) - mu*log(J)...
    + kappa/4*(I3-1-2*log(J)) + c0*(P-1);
S_tensor = 0.5*mu*eye(2) - mu/2*Cinv...
    + kappa/4*I3*Cinv - kappa/4*Cinv + c0*P_1_fact*N;
S_tensor = 2*S_tensor;
S = [S_tensor(1,1) S_tensor(2,2) S_tensor(1,2)];
CC = zeros(3,3);
f1 = mu/4+kappa/8;
f2 = kappa/4;
f3 = kappa/8;
CC(1,1) = f1*A1111 + f2*Cinv(1,1)*Cinv(1,1)*I3 ...
- f3*I3*A1111 + c0*P_2_fact*N11N11;
CC(2,2) = f1*A2222 + f2*Cinv(2,2)*Cinv(2,2)*I3 ...
- f3*I3*A2222 + c0*P_2_fact*N22N22;
CC(3,3) = f1*A1212 + f2*Cinv(1,2)*Cinv(1,2)*I3 ...
- f3*I3*A1212 + c0*P_2_fact*N12N12;
CC(1,2) = f1*A1122 + f2*Cinv(1,1)*Cinv(2,2)*I3 ...
- f3*I3*A1122 + c0*P_2_fact*N11N22;
CC(1,3) = f1*A1112 + f2*Cinv(1,1)*Cinv(1,2)*I3 ...
- f3*I3*A1112 + c0*P_2_fact*N11N12;
CC(2,3) = f1*A2212 + f2*Cinv(2,2)*Cinv(1,2)*I3 ...
- f3*I3*A2212 + c0*P_2_fact*N22N12;
CC(2,1) = CC(1,2);
CC(3,1) = CC(1,3);
CC(3,2) = CC(2,3);
CC = 4*CC;
end
```

THE END