University of California, Los Angeles

Computational Mechanics of Solids and Structures

MAE 261 B

Homework 1

Author:
Amit Singh

Supervisor:
Dr. W.S. Klug

February 10, 2015

Contents

Chapter 1

Problem 1

1.1 Introduction

Our objective in this problem is to calculate the deformation gradient, right Cauchy-Green strain tensor and the Green strain tensor from a given deformed configuration map. The problem gives position map for a uniaxial deformation of a cylinder. Cylindrical coordinates are our natural choice of curvilinear coordinates for this problem. The cylindrical basis vectors in the lab-frame components are given as

$$[\theta_1, \theta_2, \theta_3] = [\mathbf{e}_R, \mathbf{e}_\phi, \mathbf{e}_Z] = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The reference configuration and the deformed configuration of this problem are independent of ϕ due to the axisymmetric nature of the problem. The reference configuration is given by

$$\mathbf{X} = R\mathbf{e}_R + Z\mathbf{e}_Z \quad a_0 < R < b_0$$

The deformed configuration is given as

$$\mathbf{x} = \lambda_1 R\mathbf{e}_R + \lambda_2 Z\mathbf{e}_Z \quad \lambda_1, \lambda_2 > 0$$

Now, determinant of deformation gradient for the given reference and deformed configuration is found to be $J = |\mathbf{F}| = \lambda_1^2\lambda_2$. A valid deformation should not invert the volume of the body. Hence $\lambda_1^2\lambda_2 > 0$.

The curvilinear tangent basis vectors are defined as follows:

$$\mathbf{G}_i = \frac{\partial \mathbf{X}}{\partial \theta_i}$$
$$\mathbf{g}_i = \frac{\partial \mathbf{x}}{\partial \theta_i}$$

The dual basis vectors can be derived from the tangent basis vectors using the following Kronecker-delta properties

$$\mathbf{G}_i \cdot \mathbf{G}^j = \delta_i^j$$
$$\mathbf{g}_i \cdot \mathbf{g}^j = \delta_i^j$$

The deformation gradient can be computed as

$$\mathbf{F} = \mathbf{g}_i \otimes \mathbf{G}^i$$

Further, the right Cauchy-Green deformation tensor is

$$\mathbf{C} = \mathbf{F}^T\mathbf{F}$$

The Green strain is given as

$$\mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I})$$

1.2 Formulation of Numerical Methods

We have used Matlab© for coding this problem. The code for solving this problem starts with clearing the workspace, command window and closing any open figure windows. Next, we initialize the parameters required for numerical calculations – $a_0, \lambda_1, \lambda_2, \phi$ and R.

```
% Clean the set-up
clear; close all; clc;

% Initialize parameters
a0 = 1; lambda1 = 1; lambda2 = 2; R = a0; phi = pi/4;
```

The next step is to create basis vectors in the lab-frame and curvilinear co-ordinates as obtained by hand calculations. Note that the curvilinear basis vectors have been stored as a 3×3 matrix where each column represents a basis vector.

```
% The cylindrical basis vectors
e_R = [cos(phi), sin(phi), 0]'; e_phi = [-sin(phi), cos(phi),
   0]';
e_Z = [0,0,1]';

% The basis vectors in reference configurations - Gt for tangent
```

```
% basis vectors and Gd for dual basis vectors
Gt =[e_R, R*e_phi, e_Z]; Gd =[e_R, e_phi/R,e_Z];

% The basis vectors in reference configurations - gt for tangent
% basis vectors and gd for dual basis vectors
gt = [lambda1*e_R,lambda1*R*e_phi, lambda2*e_Z]; gd =
[(1/lambda1)*e_R,(1/(lambda1*R))*e_phi, (1/lambda2)*e_Z];
```

The deformation gradient and the two strain tensors can be calculated using the equations we saw earlier

```
% Deformation Gradient, $ F = gt(:,i) \otimes Gd(:,i) $
F = gt(:,1)*Gd(:,1)' + gt(:,2)*Gd(:,2)' + gt(:,3)*Gd(:,3)';

% The right Cauchy-Green deformation tensor
C = F'*F;

% The Green strain
E = (C - eye(3))/2;
```

At last, we will display the calculated arrays and matrices on the output window

```
display(gt); display(gd); display(Gt); display(Gd); display(F);
display(C); display(E);
```

1.3 Calculations and Results

Using $R = a_0 = 1$, $\lambda_1 = 1$, $\lambda_2 = 2$, $\phi = \pi/4$

Analytical Results:

The tangent basis vectors

$$(\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3) = \begin{pmatrix} \cos(\phi) & -R\sin(\phi) & 0 \\ \sin(\phi) & R\cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \approx \begin{pmatrix} 0.7071 & -0.7071 & 0 \\ 0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The dual basis vectors

$$\left(\mathbf{G}^1, \mathbf{G}^2, \mathbf{G}^3\right) = \begin{pmatrix} \cos(\phi) & -\sin(\phi)/R & 0 \\ \sin(\phi) & \cos(\phi)/R & 0 \\ 0 & 0 & 1 \end{pmatrix} \approx \begin{pmatrix} 0.7071 & -0.7071 & 0 \\ 0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The tangent basis vectors

$$(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3) = \begin{pmatrix} \lambda_1\cos(\phi) & -\lambda_1 R\sin(\phi) & 0 \\ \lambda_1\sin(\phi) & \lambda_1 R\cos(\phi) & 0 \\ 0 & 0 & \lambda_2 \end{pmatrix} \approx \begin{pmatrix} 0.7071 & -0.7071 & 0 \\ 0.7071 & 0.7071 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

The dual basis vectors

$$\left(\mathbf{g}^1, \mathbf{g}^2, \mathbf{g}^3\right) = \begin{pmatrix} 1/\lambda_1 & -\sin(\phi)/\lambda_1 R & 0 \\ 0 & \cos(\phi)/\lambda_1 R & 0 \\ 0 & 0 & 1/\lambda_2 \end{pmatrix} \approx \begin{pmatrix} 0.7071 & -0.7071 & 0 \\ 0.7071 & 0.7071 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}$$

Numerical Results:

The tangent basis vectors

$$(\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3) = \begin{pmatrix} 0.7071 & -0.7071 & 0 \\ 0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The dual basis vectors

$$\left(\mathbf{G}^1, \mathbf{G}^2, \mathbf{G}^3\right) = \begin{pmatrix} 0.7071 & -0.7071 & 0 \\ 0.7071 & 0.7071 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The tangent basis vectors

$$(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3) = \begin{pmatrix} 0.7071 & -0.7071 & 0 \\ 0.7071 & 0.7071 & 0 \\ 0 & 0 & 2.0 \end{pmatrix}$$

The dual basis vectors

$$\left(\mathbf{g}^1, \mathbf{g}^2, \mathbf{g}^3\right) = \begin{pmatrix} 0.7071 & -0.7071 & 0 \\ 0.7071 & 0.7071 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}$$

Analytical Results:

The deformation gradient

$$\mathbf{F} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_2 \end{pmatrix} \approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

The right Cauchy-Green Strain tensor

$$\mathbf{C} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_1^2 & 0 \\ 0 & 0 & \lambda_2^2 \end{pmatrix} \approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

The Green Strain tensor

$$\mathbf{E} = \begin{pmatrix} (\lambda_1 - 1)/2 & 0 & 0 \\ 0 & (\lambda_1^2 - 1)/2 & 0 \\ 0 & 0 & (\lambda_2^2 - 1)/2 \end{pmatrix} \approx \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1.5 \end{pmatrix}$$

Numerical Results:

The deformation gradient

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

The right Cauchy-Green Strain

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

The Green Strain Tensor

$$\mathbf{E} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1.5 \end{pmatrix}$$

1.4 Discussion and Conclusions

Looking at the equations for reference configuration and deformed configurations

$$\mathbf{X} = R\mathbf{e}_R + Z\mathbf{e}_Z \quad a_0 < R < b_0$$
$$\mathbf{x} = \lambda_1 R\mathbf{e}_R + \lambda_2 Z\mathbf{e}_Z \quad \lambda_1, \lambda_2 > 0$$

we can see that the deformed configuration is simply an axisymmetric radial scaling of reference configuration along with a scaling in the $\mathbf{e}_Z$ direction. Due to axisymmetry we expect

$$\mathbf{F}_{11} = \mathbf{F}_{22}$$

Also, because λ_1 is the stretching ratio we expect

$$\mathbf{F}_{11} = \mathbf{F}_{22} = \lambda_1$$

The stretching ratio along $\mathbf{e}_Z$ is clearly λ_2 therefore we expect

$$\mathbf{F}_{33} = \lambda_2$$

Our expectation is confirmed by both the hand calculations and numerical results because we obtain

$$\begin{aligned}\mathbf{F} &= \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_2 \end{pmatrix} \\ &\approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}\end{aligned}$$

for the chosen values $\lambda_1 = 1$, $\lambda_2 = 2$. Physically, the right-Cauchy strain tensor gives the square of local change in distances on deformation

$$d\mathbf{x} = \mathbf{X} \cdot \mathbf{C}\mathbf{X}$$

This is consistent with our results

$$\begin{aligned}\mathbf{C} &= \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_1^2 & 0 \\ 0 & 0 & \lambda_2^2 \end{pmatrix} \\ &\approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{pmatrix}\end{aligned}$$

The Green-strain indicates how much $\mathbf{C}$ differs from $\mathbf{I}$ where strain $\mathbf{I}$ means that reference and deformed configurations are the same. Once again, this is seen in our results.

$$\begin{aligned}\mathbf{E} &= \begin{pmatrix} (\lambda_1 - 1)/2 & 0 & 0 \\ 0 & (\lambda_1^2 - 1)/2 & 0 \\ 0 & 0 & (\lambda_2^2 - 1)/2 \end{pmatrix} \\ &\approx \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1.5 \end{pmatrix}\end{aligned}$$

As we see, because the radial stretch has been chosen to be $\lambda_1 = 1$, which means no change in the radial direction, in the Green-strain tensor, the $(1, 1)$ and $(2, 2)$ components are 0.

Thus, we find that our calculations and results are consistent with physical expectations.

1.5 Source Code Listing

The files related to Problem 1 are

1. HW1_1.m

Chapter 2

Problem 2

2.1 Introduction

In this problem, we will implement a program that calculates stress response of a planar sheet of non-linear hyperelastic material subjected to a given deformation. We have been given a compressible neo-Hookean hyperelastic model which has strain energy defined as

$$w(\mathbf{C}) = \frac{\lambda_0}{2}\ln^2(J) - \mu_0 \ln(J) + \frac{\mu_0}{2}(I_1 - 3) \tag{2.1}$$

where

$$I_1 = tr(\mathbf{C}) = C_{kk}, \quad J = det(\mathbf{F}), \quad \mathbf{C} = \mathbf{F}^T\mathbf{F}.$$

By definition, first Piola-Kirchhoff stress tensor is given as

$$\begin{aligned} P_{iJ} &= \frac{\partial w(\mathbf{C})}{\partial F_{iJ}} \\ &= (\lambda_0 \ln(J) - \mu_0)\frac{\partial}{\partial F_{iJ}}(\ln(J)) + \frac{\mu_0}{2}\left(\frac{\partial}{\partial F_{iJ}}(F_{lK}F_{lK})\right) \end{aligned}$$

Now, we can simplify above equation using following two results

$$\begin{aligned} \frac{\partial}{\partial F_{iJ}}(\ln(J)) &= \frac{1}{J}\frac{det(\mathbf{F})}{\partial F_{iJ}} \\ &= \frac{1}{J}JF^{-1}_{Ji} \\ &= F^{-1}_{Ji} \\ \frac{\partial}{\partial F_{iJ}}(F_{lK}F_{lK}) &= 2F_{lK}\delta_{li}\delta_{KJ} \\ &= 2F_{iJ} \end{aligned}$$

Thus we get,

$$\boxed{P_{iJ} = (\lambda_0 \ln(J) - \mu_0) F^{-1}_{Ji} + \mu_0 F_{iJ}} \tag{2.2}$$

The tangent modulus is a derivative of first Piola-Kirchhoff stress tensor with respect to the deformation gradient. Using the last two quations we can calculate the derivative as follows

$$\begin{aligned} C_{iJkL} &= \frac{\partial P_{iJ}}{\partial F_{kL}} \\ &= (\lambda_0 \ln(J) - \mu_0) \frac{\partial F^{-1}_{Ji}}{\partial F_{kL}} + \frac{\lambda_0}{J} \left(\frac{\partial det(\mathbf{F})}{\partial F_{kL}} \right) F^{-1}_{Ji} + \mu_0 \delta_{ki} \delta_{LJ} \end{aligned}$$

We can simplify this equation using the result

$$\frac{\partial F^{-1}_{Ji}}{\partial F_{kL}} = -F^{-1}_{Jk} F^{-1}_{Li}$$

$$\boxed{C_{iJkL} = \lambda_0 F^{-1}_{Ji} F^{-1}_{Lk} + \mu_0 \delta_{ik} \delta_{JL} - (\lambda_0 \ln(J) - \mu_0) F^{-1}_{Jk} F^{-1}_{Li}} \tag{2.3}$$

Equations 2.1, 2.2 and 2.3 give 3-D behavior of a compressible neo-Hookean material. Next, we will adapt these equations for 2-D *plane stress* case. Consider a planar sheet of compressible neo-Hookean material lying in the 1-2 plane of the lab frame. Thus, the normal to the surface is $\boldsymbol{N} = \boldsymbol{E}_3$. A planar deformation implies there should not be any shearing in the 3-direction. But in response to the in-plane stretching we can expect a compression in the 3-direction. Thus, our deformation gradient can be assumed to look like

$$[F_{iJ}] = \begin{pmatrix} F_{11} & F_{12} & 0 \\ F_{21} & F_{22} & 0 \\ 0 & 0 & \lambda \end{pmatrix}$$

In order that there is no stress in the 3-direction, we should set the component of traction vector in that direction to be zero in the deformed configuration. But because of planar stress we also expect the surface normal in deformed and reference configuration to be in the same direction i.e. $\boldsymbol{n} = \boldsymbol{N}$. We also assume that the traction vectors are same in the reference and deformed configuration. Hence, we can write

$$T_N = \boldsymbol{N} \cdot P\boldsymbol{N} = P_{33}(F_{\alpha\beta}, \lambda) = 0 \quad \alpha, \beta \in 1, 2 \tag{2.4}$$

In general, $\mathbf{P}$ is a function of components of $\mathbf{F}$. Hence, for plane stress we can write $P_{33}(F_{\alpha\beta}, \lambda))$. We need to solve for λ using the known values of $F_{\alpha\beta}$. This will be discussed in next section.

Let us define the strain energy density, first Piola-Kirchhoff stress and the tangent modulus for plane-stress condition as follows

$$
\begin{aligned}
w^{(2D)}(\mathbf{F}) &\equiv w^{(3D)}(F_{\alpha\beta}, \lambda(F_{\alpha\beta})) \\
P^{(2D)}_{\alpha\beta}(\mathbf{F}) &\equiv \frac{\partial}{\partial F_{\alpha\beta}}(w^{(2d)}) \\
&= \frac{\partial w^{(3D)}}{\partial F_{\alpha\beta}} + \frac{\partial w}{\partial \lambda}\frac{\partial \lambda}{\partial F_{\alpha\beta}} \qquad P_{33} = 0 \\
&= \frac{\partial w^{(3D)}}{\partial F_{\alpha\beta}} \\
C^{(2D)}_{\alpha\beta\gamma\delta}(\mathbf{F}) &\equiv \frac{\partial}{\partial F_{\gamma\delta}}\left(P^{(2D)}_{\alpha\beta}\right) \\
&= \frac{\partial P_{\alpha\beta}}{\partial F_{\gamma\delta}} + \frac{\partial P_{\alpha\beta}}{\partial F_{33}}\frac{\partial \lambda}{\partial F_{\gamma\delta}}
\end{aligned}
$$

Using plane strain constraint we find that

$$
\begin{aligned}
P_{33}(F_{\alpha\beta}, \lambda) &= 0 \\
dP_{33} = \frac{\partial P_{33}}{\partial F_{\alpha\beta}} dF_{\alpha\beta} + \frac{\partial P_{33}}{\partial F_{33}} d\lambda &= 0 \\
\implies \quad C_{33\alpha\beta} dF_{\alpha\beta} + C_{3333} d\lambda &= 0 \\
C_{33\alpha\beta} dF_{\alpha\beta} + C_{3333}\frac{\partial \lambda}{\partial F_{\alpha\beta}} dF_{\alpha\beta} &= 0 \\
\left(C_{33\alpha\beta} + C_{3333}\frac{\partial \lambda}{\partial F_{\alpha\beta}}\right) &= 0 \\
\implies \quad \boxed{\frac{\partial \lambda}{\partial F_{\alpha\beta}} = -\frac{C_{33\alpha\beta}}{C_{3333}}}
\end{aligned}
$$

Therefore, the 2-D tangent modulus is given as

$$
\boxed{C^{(2D)}_{\alpha\beta\gamma\delta} = C_{\alpha\beta\gamma\delta} - C_{\alpha\beta 33} C_{33\gamma\delta} \left(C_{3333}\right)^{-1}}
$$

Thus we have obtained all theoretical results required to solve our problem. Now, let's look at some numerical aspects of the solution.

2.2 Formulation of Numerical Methods

2.2.1 Neo-Hookean function

First of all, we will implement a function that takes as input a valid deformation gradient $\mathbf{F}$ and material constants, λ_0 and μ_0, as input and returns the strain engergy density w, first Piola-Kirchhoff stress tensor $\mathbf{P}$ and the tangent modulus C_{iJkL} as output using the analytical results

derived in the introduction. We use multidimensional array to store the fourth order tensor, C_{iJkL}. We also implemented a small function *kronDel()* that takes two indices i and j as input and returns 1 if $i = j$ and 0 otherwise.

```
function [w,P,TM] = neoHookean(F,lambda,mu)
% NEOHOOKEAN takes the deformation gradient as an input and returns the
% strain energy density w, first Piola-Kirchhoff stress tensor P and
% tangent modulus TM as output.User must also specify the values of
% material constants 'lambda' and 'mu'

J = det(F);
Fi = inv(F);
C = F'*F;
I1 = trace(C);

% Calculating strain energy density
w = (lambda/2)*(log(J))^2 - mu*(log(J)) + (mu/2)*(I1-3);

% Calculating the first Piola-Kirchoff Stress tensor
P = (lambda*log(J)- mu)*Fi' + mu*F;

% Assming 3-dimensions for a fourth-order tangent modulus
TM = zeros(3,3,3,3);

% Calculating the TM components
for l=1:3
    for M=1:3
        for n=1:3
            for O=1:3
                TM(l,M,n,O) = lambda*Fi(M,l)*Fi(O,n) + mu*...
                              kronDel(l,n)*kronDel(M,O) - (lambda*
                                 log(J)...
                              - mu)*Fi(M,n)*Fi(O,l);
            end
```

```
30            end
31        end
32    end
33    end
```

2.2.2 Generating valid arbitrary deformation gradient

We generate a random matrix $\mathbf{H}$ of size 3×3. A valid deformation gradient should have $J = det(\mathbf{F}) > 0$. We can ensure this if $\mathbf{F}$ is a positive definite matrix. Hence, we construct the deformation gradient as $\mathbf{F} = \mathbf{H}\mathbf{H}^T$

```
1    F = rand(3);
2    % We will ensure that F is positive definite so that det(F)>0
3    F = F*F';
4    if(det(F)<0)
5          error('Invalid deformation gradient generated!');
6    end
```

2.2.3 Consistency test

As we saw in the introduction, $\mathbf{P}$ is derivative of w with respect to $\mathbf{F}$ and C_{iJkL} is derivative of P_{iJ} with respect to F_{kL} for any valid deformation. We also derived the analytical expressions for P_{iJ} and C_{iJkL}. We can validate our code, by testing if numerical differentiation of w and $\mathbf{P}$ tallies with the analytical equations for $\mathbf{P}$ and C_{iJkL} respectively. This is called consistency test. We will use 3-point finite difference scheme for numerical differentiation. As per this scheme, if h is small perturbation in the value x, the derivative of function $f(x)$ is given as

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} = f'_h(x)$$

This method has error $\mathcal{O}(h^2)$.

$$\begin{aligned} \text{error} = |f'_h(x) - f'(x)| &= Kh^2 \\ \log((erro)r) &= \log(K) + 2\log(h) \end{aligned}$$

where K is some constant. So on a log-log plot of error vs h should give us a straight line of slope 2. For our problem, the numerical derivatives are given as follows:

$$P_{iJ}^{(h)}(F_{iJ}) = \frac{w(F_{iJ}+h) - w(F_{iJ}-h)}{2h}$$
$$C_{iJkL}^{(h)}(F_{kL}) = \frac{P_{iJ}(F_{kL}+h) - P_{iJ}(F_{kL}-h)}{2h}$$

The implementation looks as follows:

```
lambda = rand(1)*10;
mu = rand(1)*10;
%Calculate the analytical P and TM
[w_exact,P_exact,TM_exact] = neoHookean(F,lambda,mu);

%% Calculate P and TM using 3-point finite difference scheme
h = norm(F)*logspace(-6,-3);

errP = zeros(1,length(h));
errTM = zeros(1,length(h));

for z=1:length(h)
    P_approx = zeros(3,3);
    for i=1:3
        for j=1:3
            F(i,j) = F(i,j) + h(z);
            [Wplus,~,~] = neoHookean(F,lambda,mu);
            F(i,j) = F(i,j)-2*h(z);
            [Wminus,~,~] = neoHookean(F,lambda,mu);
            P_approx(i,j) = (Wplus - Wminus)/(2*h(z));
            F(i,j) = F(i,j) + h(z); %Reset F
        end
    end
    errP(z)= max(max(abs(P_exact-P_approx)))/norm(P_exact);

    % Caculate TM using 3-point finite difference scheme
    TM_approx = zeros(3,3,3,3);
    index = 1;
```

```
    for i=1:3
        for J=1:3
            for k=1:3
                for L=1:3
                    F(k,L) = F(k,L) + h(z);
                    [~,Pplus,~] = neoHookean(F,lambda,mu);
                    F(k,L) = F(k,L) - 2*h(z);
                    [~,Pminus,~] = neoHookean(F,lambda,mu);
                    TM_approx(i,J,k,L) = (Pplus(i,J)-Pminus(i,J))
                       /(2*h(z));
                    F(k,L) = F(k,L) + h(z); %Reset F
                end
            end
        end
    end
    errTM(z) = max(max(max(max((TM_exact - TM_approx)))))/...
        max(max(max(max(TM_exact))));
end

%% Log-log plot of error vs h
loglog(h,errP,'r',h,errTM,'g');
xlabel('log(h)');
ylabel('log(error)')
legend('Log of error in P','Log of error in TM');

slope1 = (log(errP(length(h)-5))-log(errP(5)))/(log(h(length(h)-5))
   -...
    log(h(5)));
fprintf('Slope of log(errP) vs log(h) = %4.3e\n',slope1);
slope2 = (log(errTM(length(h)-5))-log(errTM(5)))/(log(h(length(h)
   -5))-...
    log(h(5)));
fprintf('Slope of log(errTM) vs log(h) = %4.3e\n',slope2);
```

Similarly, we implemented consistency check for plane-stress neoHookean function.

2.2.4 Generating a random rotation matrix

We used Rodrigues equation to otbain rotation matrix $\mathbf{Q}$ for rotation around a randomly generated axis $\mathbf{v}$ by a random angle θ. We check that the rotation matrix actually belongs to SO^3 by checking if $det(\mathbf{Q}) = 1$ and $\mathbf{Q}^{-1} = \mathbf{Q}^T$

```
v = rand(3,1);
n = v/norm(v);
theta = rand(1)*pi;
n_hat = [0,-n(3),n(2); n(3),0,-n(1);-n(2),n(1),0];
Q = eye(3) + sin(theta)*n_hat + (1- cos(theta))*(n*n' - eye(3));
tol = 10^-7;

% Check if Q belongs to SO3
if (abs(det(Q)-1) > tol || norm(abs(inv(Q) - Q')) > tol)
    error('Q does not belong to SO3.\n');
end
```

2.2.5 Material Frame Indifference test

A valid constitutive material model must satisfy material frame indifference, that is, the stress response should be independent of the coordinate representation used in calculation. One way to check this is to compare w, $\mathbf{P}$ and C_{iJkL} obtained for a deformation gradient $\mathbf{F}$ with the same quantities calculated for $\mathbf{F}$ in a rigidly rotated co-ordinate system. Theoretically, the two set of quantities should compare as follows.

$$w(\mathbf{QF}) = w(\mathbf{F})$$
$$\mathbf{P}(\mathbf{QF}) = \mathbf{QP}(\mathbf{F})$$
$$C_{iJkL}(\mathbf{QF}) = Q_{im}Q_{kn}C_{mJnL}(\mathbf{F})$$

The snippet that validates this is as follows

```
%% Checking Material Frame Indifference
F_rot = Q*F;
[w_rot,P_rot,TM_rot] = neoHookean(F_rot,lambda,mu);

fprintf('Relative change in w due to rotation = %4.3e\n',...
    abs(w_rot - w_exact)/norm(w_exact));
fprintf('Relative change in P due to rotation = %4.3e\n',...
```

```
    max(max((abs(P_rot - Q*P_exact))))/norm(P_exact));
TM_rot_exact = zeros(3,3,3,3);
for i=1:3
    for J=1:3
        for k=1:3
            for L=1:3
                for m=1:3
                    for n=1:3
                        TM_rot_exact(i,J,k,L) = ...
                            TM_rot_exact(i,J,k,L) + Q(i,m)*Q(k,n)*
                               ...
                            TM_exact(m,J,n,L);
                    end
                end
            end
        end
    end
end
fprintf('Relative change in TM due to rotation = %4.3e\n',...
    max(max(max(max(abs(TM_rot - TM_rot_exact)))))/...
    max(max(max(max(TM_exact)))));
```

2.2.6 Isotropy test

Compressible neo-Hookean model is designed to be isotropic. The stress-response of the material should not change arbitrarily by just a rigid rotation of the deformation. Rather, a rotated deformation should yield quantities that are related to their counterparts for original deformation as follows

$$w(\mathbf{FQ}) = w(\mathbf{F})$$
$$\mathbf{P}(\mathbf{FQ}) = \mathbf{P}(\mathbf{F})\mathbf{Q}$$
$$C_{iJkL}(\mathbf{QF}) = Q_{MJ}Q_{NL}C_{iMkN}(\mathbf{F})$$

Following code validates this requirement

```
%% Symmetry test
F_sym = F*Q;
```

```
3
4  [w_sym,P_sym,TM_sym] = neoHookean(F_sym,lambda,mu);
5
6  fprintf('Relative change in w for symmetry test = %4.3e\n',...
7      abs(w_sym - w_exact)/norm(w_exact));
8  fprintf('Relative change in P for symmetry test = %4.3e\n',...
9      max(max((abs(P_sym - P_exact*Q))))/norm(P_exact));
10
11 TM_sym_exact = zeros(3,3,3,3);
12 for i=1:3
13     for J=1:3
14         for k=1:3
15             for L=1:3
16                 for M=1:3
17                     for N=1:3
18                         TM_sym_exact(i,J,k,L) = ...
19                             TM_sym_exact(i,J,k,L) + Q(M,J)*Q(N,L)*
                                   ...
20                             TM_exact(i,M,k,N);
21                     end
22                 end
23             end
24         end
25     end
26 end
27
28 fprintf('Relative change in TM for symmetry test = %4.3e\n',...
29     max(max(max(max(abs(TM_sym - TM_sym_exact)))))/...
30     max(max(max(max(TM_exact)))));
```

2.2.7 Newton-Raphson method

Equation 2.4 is non-linear and we need to solve for λ in terms of $F_{\alpha\beta}$. This can be done using *Newton-Raphson* iterative method. Since $F_{\alpha\beta}$ are known we can consider $P_{33}(F_{\alpha\beta}, \lambda) = P_{33}(\lambda)$ Suppose, λ_{n+1} is a root of the non-linear equation 2.4 and λ_n is our guess for the root such that

$\lambda_{n+1} = \lambda_n + \Delta\lambda$ where $\Delta\lambda$ is small. Using Taylor expansion we can write

$$0 = P_{33}(\lambda_n + \Delta\lambda) = P_{33}(\lambda_n) + \Delta\lambda \cancel{\frac{dP_{33}}{d\lambda}}^{C_{3333}} + \dots$$
$$\implies \quad \Delta\lambda \approx -P_{33}(\lambda_n)\left(C_{3333}(\lambda_n)\right)^{-1}$$

Therefore, our Newton-Raphson iteration scheme is

$$\boxed{\lambda_{n+1} = \lambda_n - P_{33}(\lambda_n)\left(C_{3333}(\lambda_n)\right)^{-1}} \tag{2.5}$$

There can be situations when the Newton-Raphson method does not converge. For example, if the error in initial guess is large then subsequent iterations will take the scheme away from actual root. Also, after certain number of iterations, the adjustment $\Delta\lambda$ may become very small and there is no tangible improvement in accuracy of the solution inspite of further iterations. Hence, we need to terminate the scheme the number of iteration crosses a reasonable threshold limit even if the error is not below our chosen tolerance limit. If our initial guess is close to the solution, the scheme converges quadratically. Hence, we don't need very large number of iterations. For our implementation we have following termination criteria:

1. $|P_{33}(\lambda_n)| < 10^{-8}$
2. Number of iterations > 100
3. $\Delta\lambda < 10^3 \times$ machine precision

The snippet looks like this

```
%Initial guess for F33 i.e. gamma and initialize P
gamma = 1;
tol = 10^-8;
iterLimit =100;
iterCount = 1;
%Newton iteration to solve for F33 such that P33 = 0
while(1)
    F = [D(1,1),D(1,2),0; D(2,1),D(2,2),0; 0,0,gamma];
    [~,P,TM] = neoHookean(F,lambda,mu);
    dgamma = -(P(3,3))/(TM(3,3,3,3));
    gamma = gamma + dgamma;
    iterCount = iterCount + 1;
    if(~isreal(P))
        error('P has become imaginary. Calculation terminated.');
```

```
    end
    if(abs(P(3,3)) < tol || iterCount>iterLimit || dgamma < eps
       *10^3)
        fprintf(['Newton iteration terminated.\n'...
            'abs(P33) = %4.3e dgamma = %4.3e iterCount = %d\n'],...
            abs(P(3,3)),dgamma,iterCount);
        break;
    end
end
```

2.3 Calculations and Results

2.3.1 Consistency tests

As we discussed in section 2.2.3, we expect our numerical differentiation scheme to have an error of $\mathcal{O}(h^2)$. Also, a log-log plot of error vs. h would validate this if it is a line with *slope* $= 2$. The error in calculation of $\mathbf{P}$ by numerical differentiation of w has been calculated as

$$\text{error} = \frac{\|\mathbf{P}_h - \mathbf{P}\|}{\|\mathbf{P}\|}$$

The error in calculation of C_{iJkL} by numerical differentiation of $\mathbf{P}$ has been calculated as

$$\text{error} = \frac{\max\limits_{\forall i,J,k,L}\left(|C^{(h)}_{iJkL} - C_{iJkL}|\right)}{\max\limits_{\forall i,J,k,L}\left(C_{iJkL}\right)}$$

We also calculated the slope of the two error plots numerically. Their values are in the range from 1.99 to 2.02. Following shows a sample output for the slopes from the script `neoHookeanCorrectness.m`

```
Slope of log(errP) vs log(h) = 2.000e+00
Slope of log(errTM) vs log(h) = 2.000e+00
```

We tested the plane-stress version of neo-Hookean implementation for consistency separately. The error in $\mathbf{P}$ has been calculated using same equation as for the 3-D neo-Hookean implementation. But the error in tangent modulus is given as

$$\text{error} = \frac{\max\limits_{\forall \alpha,\beta,\gamma,\delta}\left(|C^{(h)(2D)}_{\alpha\beta\gamma\delta} - C^{(2D)}_{\alpha\beta\gamma\delta}|\right)}{\max\limits_{\forall i,J,k,L}\left(C_{\alpha\beta\gamma\delta}\right)}$$

This is because $C^{(2D)}_{\alpha\beta\gamma\delta}$ is not merely a subset of components of C_{iJkL} but requires a correction in order to be consistent with plane stress condition. The error plot is as shown in figure

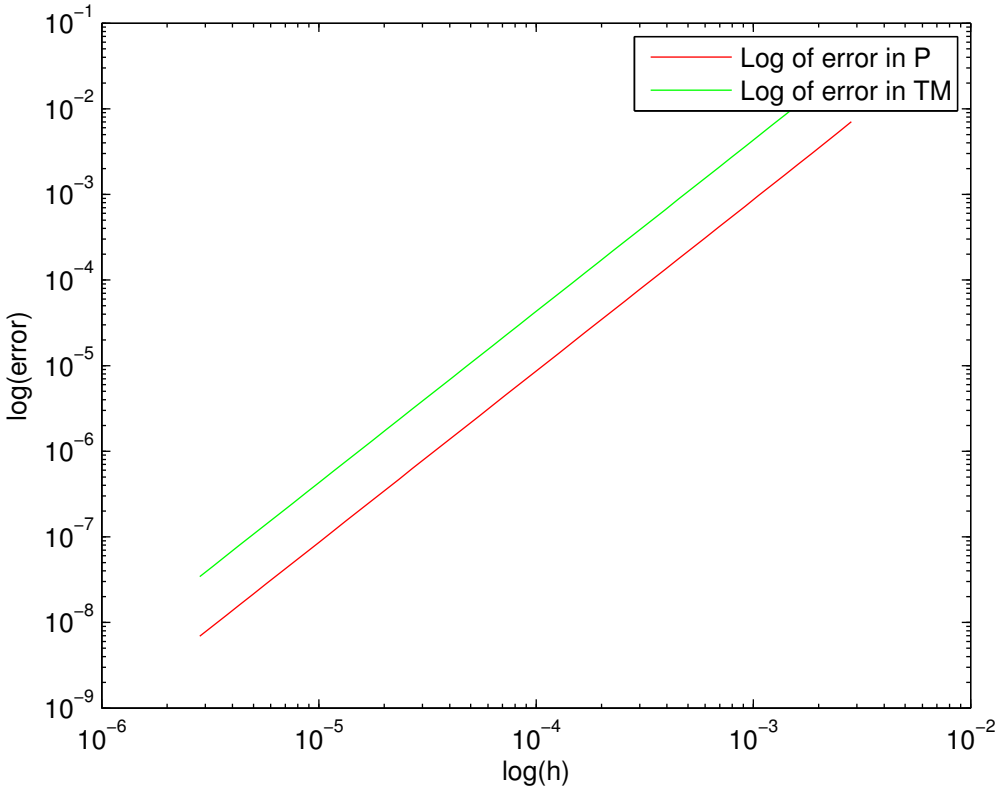

Figure 2.1: Error vs. Perturbation plot for 3D neoHookean implementation

2.2. As can be seen from the plot and also verified from numerically calculations, slopes of the error-plots are always around 2 ± 0.02 Following is a sample slope output from the script `planeStressConsistency.m`

```
Slope of log(errP) vs log(h) = 1.997e+00
Slope of log(errTM) vs log(h) = 2.012e+00
```

2.3.2 Material Frame Indifference test

The error in MFI test for w is calculated as

$$\text{error} = \frac{|w(\mathbf{QF}) - w(\mathbf{F})|}{|w(\mathbf{F})|}$$

The error in MFI test for $\mathbf{P}$ is calculated as

$$\text{error} = \frac{\|\mathbf{P}(\mathbf{QF}) - \mathbf{QP}(\mathbf{F})\|}{\|\mathbf{QP}(\mathbf{F})\|}$$

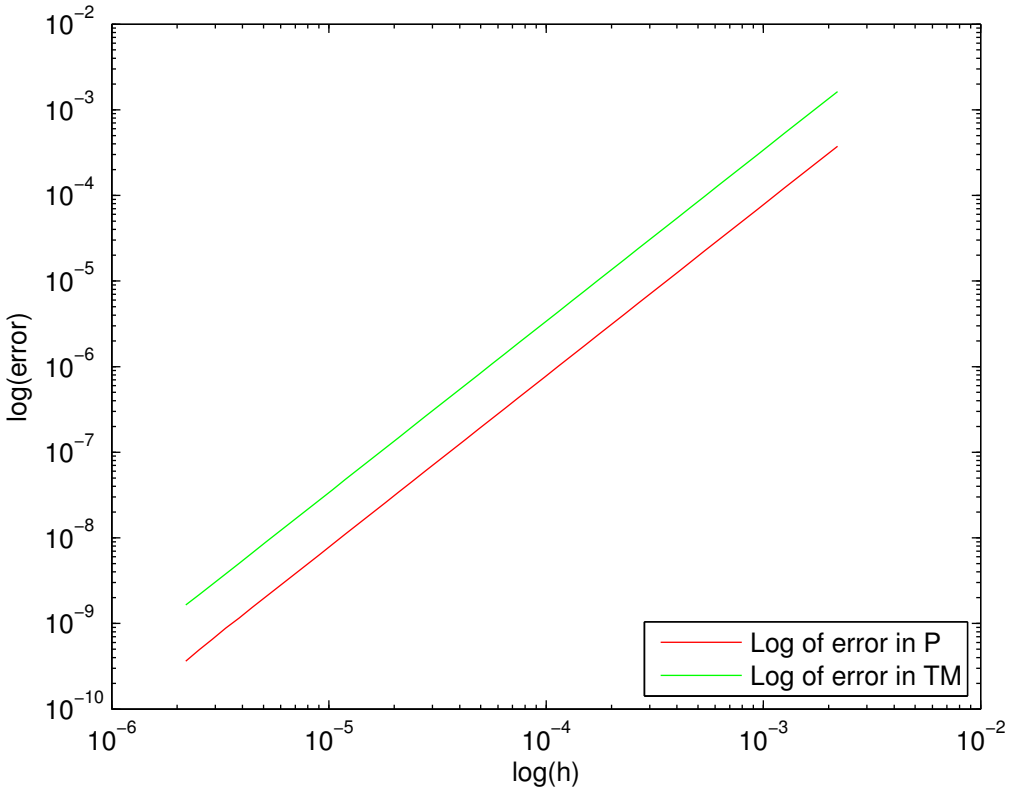

Figure 2.2: Error vs. Perturbation plot for plane stress neoHookean implementation

The error in C_{iJkL} for MFI test has been calculated as

$$\text{error} = \frac{\max\limits_{\forall i,J,k,L} \left(|C_{iJkL}(\mathbf{QF}) - Q_{ij}Q_{kl}C_{jJlL}(\mathbf{F})|\right)}{\max\limits_{\forall i,J,k,L} \left(Q_{ij}Q_{kl}C_{jJlL}(\mathbf{F})\right)}$$

Following is a sample output for the MFI test from the script `neoHookeanCorrectness.m`

```
Relative error in w due to coordinate rotation = 8.513e-15
Relative error in P due to coordinate rotation = 1.65e-14
Relative error in TM due to coordinate rotation = 4.359e-14
```

2.3.3 Isotropy Test

The error in isotropy test for w is calculated as

$$\text{error} = \frac{|w(\mathbf{FQ}) - w(\mathbf{F})|}{|w(\mathbf{F})|}$$

The error in MFI test for $\mathbf{P}$ is calculated as

$$\text{error} = \frac{\|\mathbf{P}(\mathbf{FQ}) - \mathbf{P}(\mathbf{F})\mathbf{Q}\|}{\|\mathbf{P}(\mathbf{F})\mathbf{Q}\|}$$

The error in C_{iJkL} for MFI test has been calculated as

$$\text{error} = \frac{\max\limits_{\forall i,J,k,L} (|C_{iJkL}(\mathbf{FQ}) - Q_{MJ}Q_{NL}C_{iMkN}(\mathbf{F})|)}{\max\limits_{\forall i,J,k,L} (Q_{MJ}Q_{NL}C_{iMkN}(\mathbf{F}))}$$

The following shows a sample output from the script `neoHookeanCorrectness.m`

```
Relative error in w for symmetry test = 9.222e-15
Relative error in P for symmetry test = 2.059e-15
Relative error in TM for symmetry test = 3.012e-15
```

2.3.4 Plane Stress Examples

For both the plane stress examples we have used the material constants for neoHookean model as follows:

$$\lambda_0 = 5 \times 10^8$$
$$\mu_0 = 1.5 \times 10^6$$

The F_{11} values used are in the closed interval $[0.1, 1.5]$. For both cases, we have plotted the P_{11} and P_{22} components of $\mathbf{P}$ on the y-axis and F_{11} on the x-axis.

Uniaxial deformation

Figure 2.3, shows the stress-strain curves for uniaxial deformation.

Equibiaxial deformation

Figure 2.4, shows the stress-strain curves for equibiaxial deformation.

2.4 Discussion and Conclusions

2.4.1 Consistency

For finite difference scheme the values of perturbation h are assumed to be very small compared to the value of variable x so that we can use Taylor expansion. But in numerical implementation for values $h << \sqrt{\epsilon}x$ where ϵ is machine precision there is significant amount of rounding error due to floating point arithmetic. We are generating our arbitrary $\mathbf{F}$ in code as

```
F = rand(3);
```

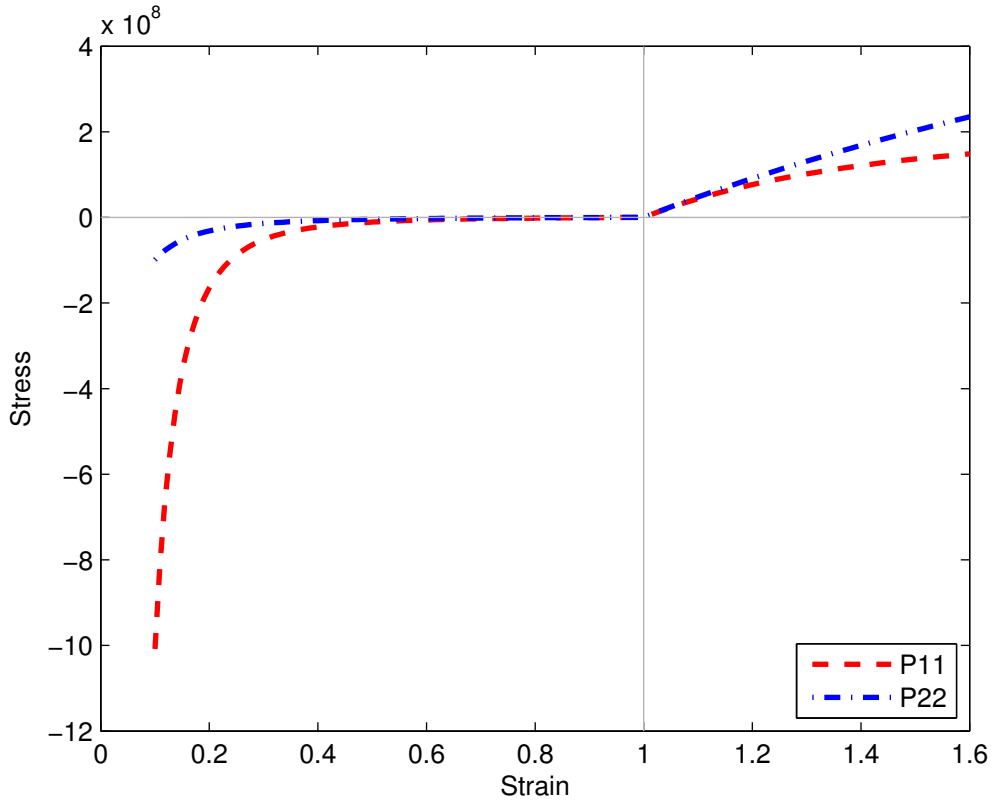

Figure 2.3: Stress vs Strain plot for uniaxial deformation

So components of F belong to the interval $(0, 1)$. We are using the values h as

$$h = \|\mathbf{F}\| \times 10^{-r} \quad \text{where} r \in (-6, -3) \quad \text{and} r \in \mathbb{Z}$$

Thus, we avoid prominent rounding errors in our scheme and obtain a quadratic scaling as expected. It can be verified from figures 2.1 and 2.2. Thus, we pass the consistency requirements.

2.4.2 MFI and Isotropy

The errors we calculate in these two tests are relative errors. Therefore, values closer to 0 are better. We get values less than 10^{-13}. These are very small and close to zero. Thus, we are sure that our implementation passes these two tests.

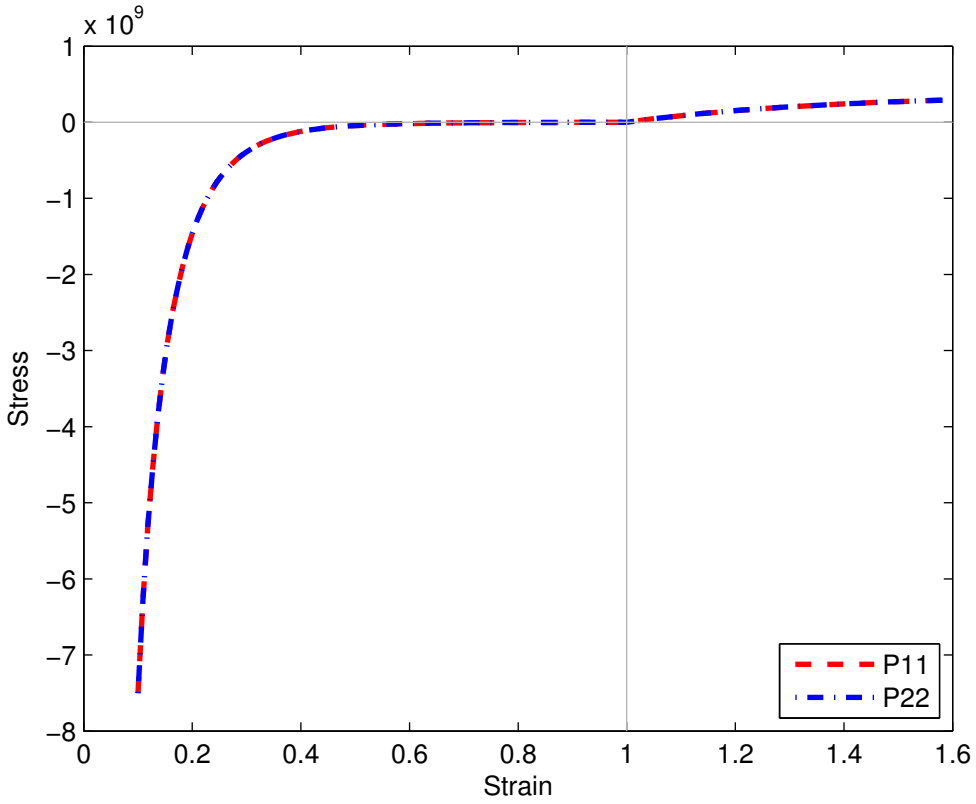

Figure 2.4: Stress vs Strain plot for equibiaxial deformation

2.4.3 Plane Stress Examples

Uniaxial Deformation

In this example, we have

$$[F_{\alpha\beta}] = \begin{bmatrix} F_{11} & 0 \\ 0 & 1 \end{bmatrix}$$

We are constraining the 2-direction to remain undeformed while 1-direction is stretched by ratio F_{11}. When $F_{11} < 1$ i.e. compression along direction 1, direction 2 would like to expand but it is constrained and thus also undergoes compression. Therefore when $P_{11} < 0$, we expect $P_{22} < 0$. Similar argument is true for elongation along direction 1. But the magnitudes of the stress along the two directions are generally not expected to be the same expect for very small values of F_{11}. This trend can be clearly validated from figure 2.3.

Equibiaxial Deformation

In this example, we have

$$[F_{\alpha\beta}] = \begin{bmatrix} F_{11} & 0 \\ 0 & F_{11} \end{bmatrix}$$

We are imposing same deformation in both 1 and 2 directions. Since the material is isotropic, we would expect same stress response in both directions. This is obvious from figure 2.4 where we find the graph of P_{11} and P_{22} overlapping for all values of F_{11}.

2.5 Source Code Listing

The list of files for problem 2 is

1. `kronDel.m`: Kronecker-delta function
2. `neoHookean.m`: 3-D constitutive model. It is a function that takes $\mathbf{F}, \lambda_0$ and μ_0 as input and gives $w, \mathbf{P}$ and C_{iJkL} as output.
3. `neoHookeanCorrectness.m`:This script tests the 3-D neo-Hookean constitutive model for consistency, material frame indifference and isotropy.
4. `planeStressNH.m`: 2-D plane stress neo-Hookean model. It is a function that takes $F_{\alpha\beta}, \lambda_0$ and μ_0 as input and gives $w, P_{\alpha\beta}$ and $C^{(2D)}_{\alpha\beta\gamma\delta}$ as output.
5. `planeStressConsistency.m`:This script tests the 2-D plane stress neoHookean model for consistency.
6. `planeStressExamples.m`:This script implements the uniaxial deformation and equibiaxial deformation examples for plane-stress.