University of California, Los Angeles

Computational Mechanics of Solids and Structures

MAE 261 B

Final Report

Author:
Amit Singh

Instructor:
Dr. W.S. Klug

March 30, 2015

Contents

Chapter 1

Assignment 1

1.1 Problem 1

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

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

1.1.2 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);
```

1.1.3 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

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

1.1.4 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 **C** differs from **I** where strain **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.

1.1.5 Source Code Listing

The files related to Problem 1 are

1. HW1_1.m

1.2 Problem 2

1.2.1 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

$$w(\mathbf{C}) = \frac{\lambda_0}{2}\ln^2(J) - \mu_0 \ln(J) + \frac{\mu_0}{2}(I_1 - 3) \tag{1.1}$$

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,

$$\boxed{P_{iJ} = (\lambda_0 \ln(J) - \mu_0)F^{-1}_{Ji} + \mu_0 F_{iJ}} \tag{1.2}$$

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

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

$$\boxed{C_{iJkL} = \lambda_0 F^{-1}_{Ji}F^{-1}_{Lk} + \mu_0\delta_{ik}\delta_{JL} - (\lambda_0\ln(J) - \mu_0)F^{-1}_{Jk}F^{-1}_{Li}} \tag{1.3}$$

Equations 1.1, 1.2 and 1.3 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

$$T_N = \boldsymbol{N} \cdot P\boldsymbol{N} = P_{33}(F_{\alpha\beta}, \lambda) = 0 \quad \alpha, \beta \in 1, 2 \tag{1.4}$$

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

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

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

$$\boxed{C^{(2D)}_{\alpha\beta\gamma\delta} = C_{\alpha\beta\gamma\delta} - C_{\alpha\beta33} C_{33\gamma\delta} \left(C_{3333} \right)^{-1}}$$

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

1.2.2 Formulation of Numerical Methods

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.

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

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:

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

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
```

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.

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

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

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

Newton-Raphson method

Equation 1.4 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 1.4 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

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

Therefore, our Newton-Raphson iteration scheme is

$$
\boxed{\lambda_{n+1} = \lambda_n - P_{33}(\lambda_n)\left(C_{3333}(\lambda_n)\right)^{-1}} \tag{1.5}
$$

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

1.2.3 Calculations and Results

Consistency tests

As we discussed in section 1.2.2, 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

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

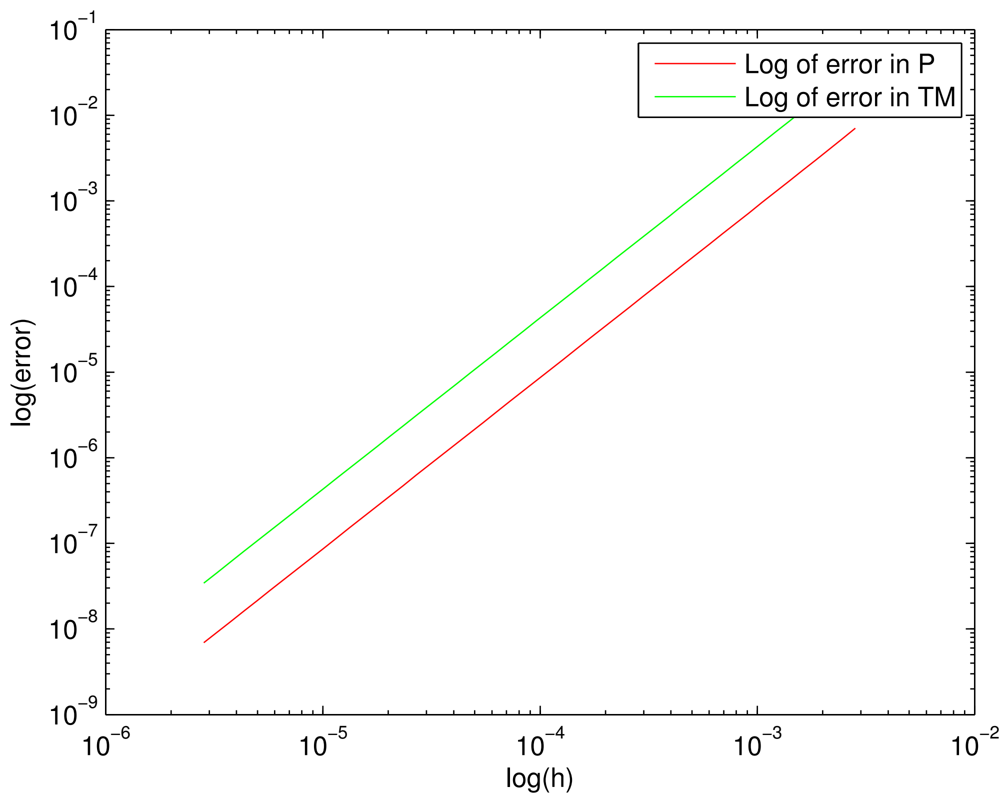


Figure 1.1: Error vs. Perturbation plot for 3D neoHookean implementation

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 1.2. 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
```

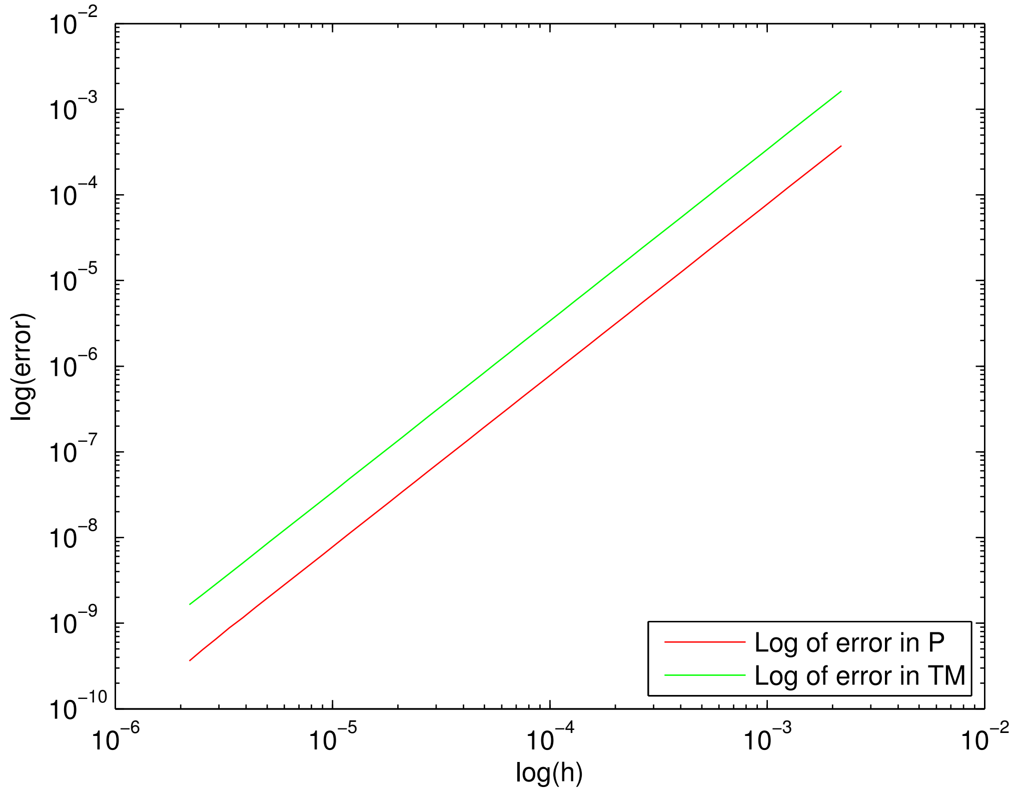


Figure 1.2: Error vs. Perturbation plot for plane stress neoHookean implementation

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

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

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
```

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

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

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
```

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 1.3, shows the stress-strain curves for uniaxial deformation.

- **Equibiaxial deformation** Figure 1.4, shows the stress-strain curves for equibiaxial deformation.

1.2.4 Discussion and Conclusions

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);
```

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 1.1 and 1.2. Thus, we pass the consistency requirements.

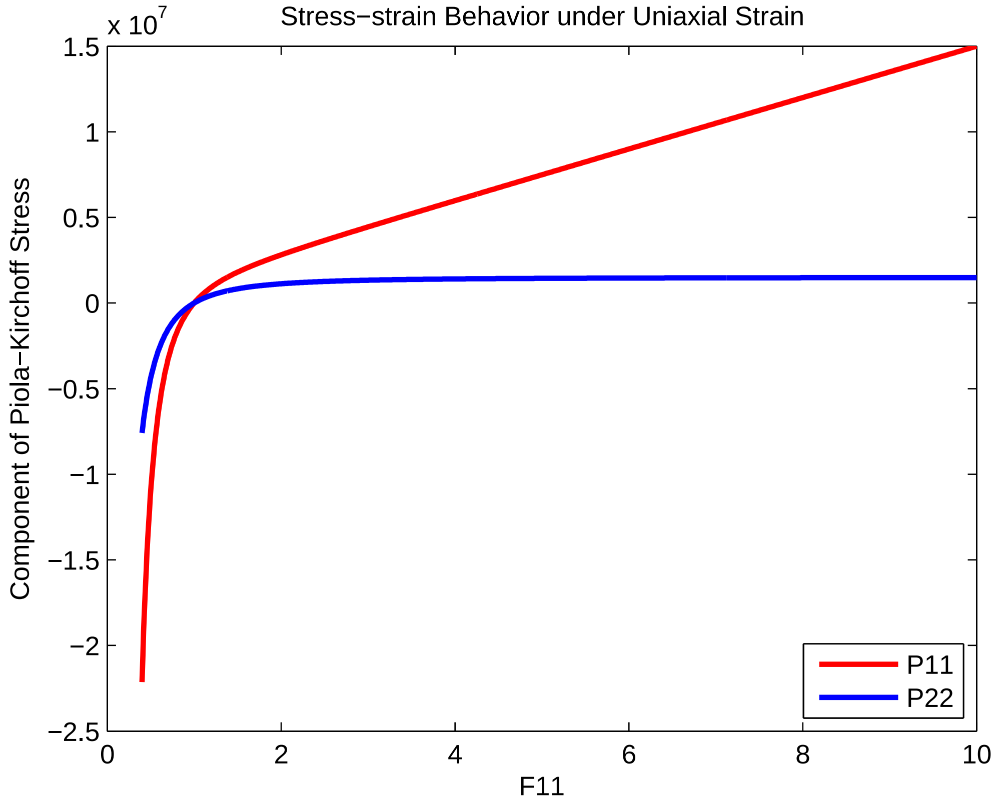


Figure 1.3: Stress vs Strain plot for uniaxial deformation

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.

1.2.5 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 1.3.

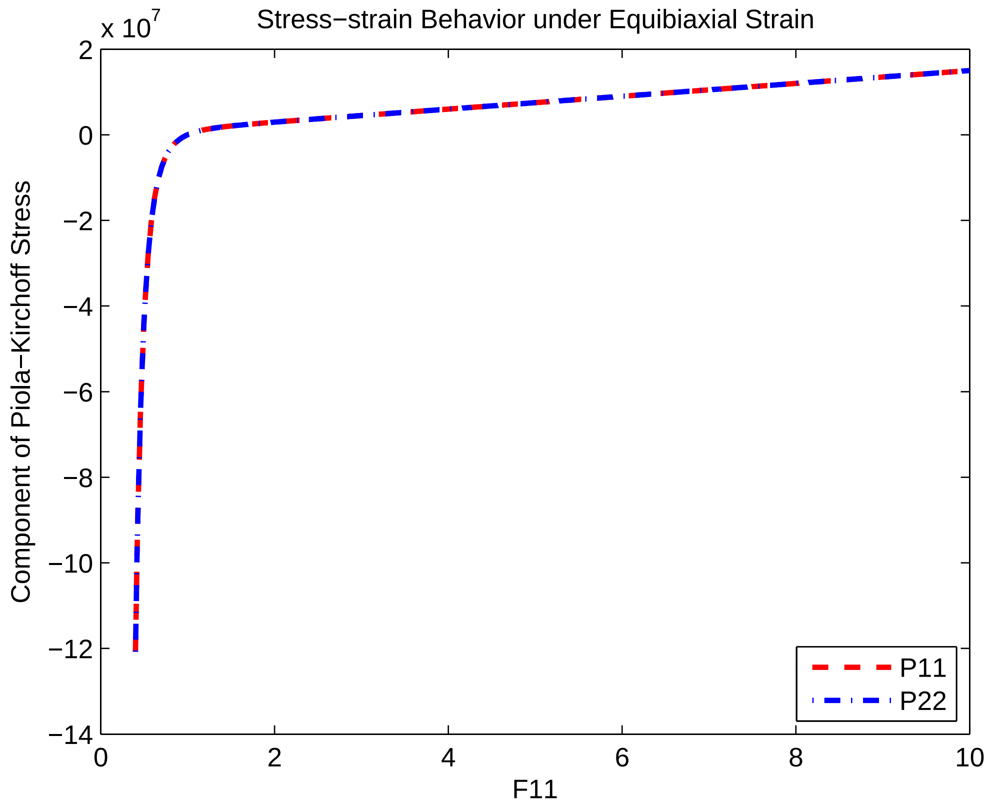


Figure 1.4: Stress vs Strain plot for equibiaxial deformation

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 1.4 where we find the graph of P_{11} and P_{22} overlapping for all values of F_{11}.

1.2.6 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.

Chapter 2

Assignment 2

2.1 Introduction

Our goal in this assignment is to code linear and quadratic *isoparametric* triagular finite elements.In finite element analysis, both the geometry and the unknown solution field are discretized using suitable interpolating functions. The term *parametric* indicates that we will develop shape functions for a standard parametric domain. Using suitable Jacobian matrix we will map these shape functions to the actual domain of integration. The term *iso* indicates that we will use the same shape functions for both the geometrical domain and the unknown solution field. To develop a numerical scheme for isoparametric triangular elements requires triangular shape functions and quadrature scheme for integration on triangular domains. These are the topics of the succeeding sections.

2.2 Triangular Shape Functions

We plan to develop shape functions for a linear 3-node triangular element and quadratic 6-node triangular element. The standard parametric triangular domains along with the nodes and their parametric coordinates for these elements are shown in figures 2.1 and 2.2. The next two subsections give the equations for shape functions and their derivatives for the two cases.

2.2.1 Linear 3-node Triangular Shape Functions and Their Derivatives

For three nodes we need three shape functions that satisfy the Kronecker-Delta property. Barycentric co-ordinates for triangles naturally satisfy this requirement. Hence the shape functions in terms of the barycentric co-ordinates are

$$\hat{N}_i = \lambda_i$$

For the node numbering shown in the standard (r, s) parametric domain of Figure 2.1, the barycentric coordinates, and thus the shape functions, are given by

$$\lambda_1 = s \qquad \lambda_2 = 1 - r \qquad \lambda_3 = r - s \tag{2.1}$$

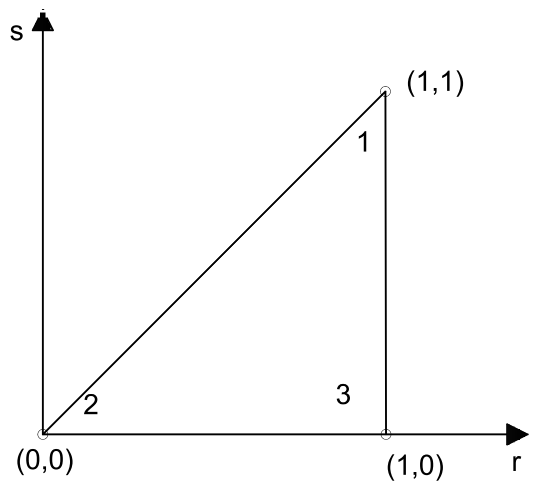


Figure 2.1: Nodal positions and numbering for linear parametric triangular domain

The derivatives are

$$\begin{aligned} \hat{N}_{1,r} &= 0 & \hat{N}_{1,s} &= 1 \\ \hat{N}_{2,r} &= -1 & \hat{N}_{2,s} &= 0 \\ \hat{N}_{3,r} &= 1 & \hat{N}_{3,s} &= -1 \end{aligned}$$

2.2.2 Quadratic 6-node Triangular Shape Functions and Their Derivatives

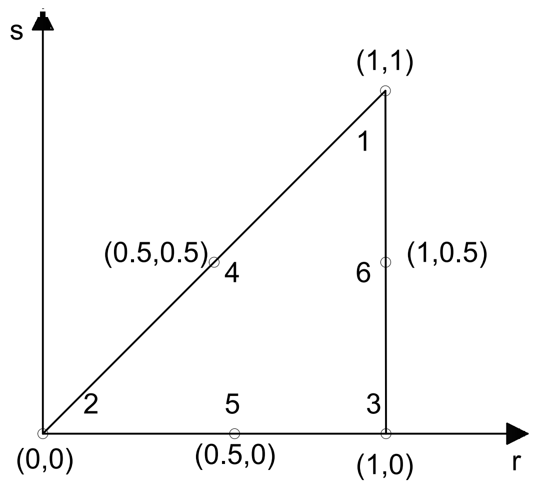


Figure 2.2: Node numbering and locations for quadratic parametric triangular domain

The node numbering is shown in Figure 2.2. As the node numbers $1, 2$ and 3 are same as for the linear 3-node case, we can use the same barycentric coordinates as in Equation 2.1. The six shape functions in terms of the barycentric coordinates are

Table 2.1: Sampling points and weights for 1-point and 3-point Gaussian quadrature

n_G	$\hat{w}_i$	(r_i, s_i)
1	$w_1 = 1$	$(2/3, 1/3)$
3	$w_1 = (1/3)$	$(1/2, 1/2)$
	$w_2 = (1/3)$	$(1, 1/2)$
	$w_3 = (1/3)$	$(1/2, 0)$

$$\begin{aligned}
\hat{N}_1 &= \lambda_1(2\lambda_1 - 1) = -s + 2s^2 \\
\hat{N}_2 &= \lambda_2(2\lambda_2 - 1) = 2r^2 - 3r + 1 \\
\hat{N}_3 &= \lambda_3(2\lambda_3 - 1) = 2r^2 - r - 4rs + s + 2s^2 \\
\hat{N}_4 &= 4\lambda_1\lambda_2 \qquad\quad = -4rs + 4s \\
\hat{N}_5 &= 4\lambda_2\lambda_3 \qquad\quad = -4(r^2 - r - rs + s) \\
\hat{N}_6 &= 4\lambda_3\lambda_1 \qquad\quad = 4rs - 4s^2
\end{aligned}$$

The derivatives are

$$\begin{aligned}
\hat{N}_{1,r} &= 0 & \hat{N}_{1,s} &= 4s - 1 \\
\hat{N}_{2,r} &= 4r - 3 & \hat{N}_{2,s} &= 0 \\
\hat{N}_{3,r} &= 4r - 4s - 1 & \hat{N}_{3,s} &= 1 - 4r + 4s \\
\hat{N}_{4,r} &= -4s & \hat{N}_{4,s} &= 4(1 - r) \\
\hat{N}_{5,r} &= 4(1 - 2r + s) & \hat{N}_{5,s} &= 4(r - 1) \\
\hat{N}_{6,r} &= 4s & \hat{N}_{6,s} &= 4(r - 2s)
\end{aligned}$$

2.3 Gaussian Quadrature on Triangular Domain

The equations for elastic response of isoparametric triangular elements have integrals evaluated over a triangular domain of integration. We will use Gaussian Quadrature rules for numerical integration. For the standard parametric domain of Figure 2.1 the general form of Gaussian quadrature is

$$\int_0^1 \int_0^r f(r,s)\, \mathrm{d}s\mathrm{d}r = \sum_{i=1}^{n_G} \hat{w}_i f(r_i, s_i)\hat{\Omega}$$

where n_G is the number of sampling points (r_i, s_i) and $\hat{w}_i$ is the corresponding weight normalized by the area of the domain $\hat{\Omega} = (1/2)$. Table 2.1 gives the values of n_G, $\hat{w}_i$ and (r_i, s_i) for the 1-point and 3-point quadrature.

2.4 Isoparametric Triangular Elements

Using either of the shape functions and one of the quadrature rules discussed in previous sections we can write the equations for strain energy, internal force and stiffness modulus of isoparametric triangular elements. Let $\mathbf{X}$ and $\mathbf{x}$ represent the reference and deformed configurations of the element. Also, we will denote the parametric domain as θ^α where $\theta^1 = r$ and $\theta^2 = s$. Let n be the number of nodes in the element. In all the following equations $i, J, k, L, \alpha, \beta = 1, 2$ and $a = 1, 2 \ldots n$. The Jacobian matrix for transformation from parametric domain to reference configuration is then given as

$$\mathbb{J}_{I\alpha} = \frac{\partial X_I}{\partial \theta^\alpha}$$

The derivative of the shape functions with respect to $\mathbf{X}$ are

$$\begin{aligned} N_{a,J} &= \frac{\partial N_a}{\partial X_{aJ}} \\ &= \frac{\partial \hat{N}_a}{\partial \theta_\beta} \frac{\partial \theta_\beta}{\partial X_J} \\ &= \hat{N}_{a,\beta} \mathbb{J}^{-1}_{\beta J} \end{aligned}$$

The deformation gradient can now be written as

$$F_{iJ} = x_{ia} N_{a,J}$$

It is important that the sequence of vertices of the triangular domain $\mathbf{X}$ and $\mathbf{x}$ must be same as formed by the node numbering of the standard domain in Figures 2.1 and 2.2. If this requirement is not met, the determinant of $\mathbf{F}$ will become negative and give rise to imaginary Piola-Kirchhoff stress tensor. Also, the triangles should not have very high asspect ratio as that can cause determinant of $\mathbf{F}$ approach 0 and cause problems in calculating $\mathbf{F}^{-1}$.

Once we have a valid deformation gradient we can calculate the strain energy density w, the Piola-Kirchhoff stress tensor P_{iJ} and the consistent two-dimensional stiffness tensor C_{iJkL} using the plane stress neo-Hookean constitutive relationship formulated in HW 1. We need the following integral equations to find strain energy W, the internal force f^{int}_{ia} and the stiffness modulus K_{iakb} for the element Ω^e

$$\begin{aligned} W &= \int_{\Omega^e} w \, \mathrm{d}\Omega^e \\ &= \sum_{q=1}^{n_G} w \bigg|_{\theta_q} \hat{w}_q |\mathbb{J}| \hat{\Omega} \end{aligned}$$

$$\begin{aligned} f^{\text{int}}_{ia} &= \int_{\Omega^e} P_{iJ} N_{a,J} \, \mathrm{d}\Omega^e \\ &= \sum_{q=1}^{n_G} \left(P_{iJ} \hat{N}_{a,\beta} \mathbb{J}^{-1}_{\beta J} \right) \bigg|_{\theta_q} \hat{w}_q |\mathbb{J}| \hat{\Omega} \end{aligned}$$

$$\begin{aligned} K_{iakb} &= \int_{\Omega^e} C_{iJkL} N_{a,J} N_{b,L} \, \mathrm{d}\Omega^e \\ &= \sum_{q=1}^{n_G} \left(C_{iJkL} \hat{N}_{a,\beta} \mathbb{J}^{-1}_{\beta J} \hat{N}_{b,\alpha} \mathbb{J}^{-1}_{\alpha L} \right) \bigg|_{\theta_q} \hat{w}_q |\mathbb{J}| \hat{\Omega} \end{aligned}$$

2.5 Verification Tests for Shape Functions, Quadrature and Elements

To verify the correctness of our computer implementations, we will check for some known properties that the shape functions, quadrature and elements should satisfy.

2.5.1 Verification of Shape Functions

- Partition of unity: At any point (r_1, s_1) in the standard domain, the shape functions should have

  $$\sum_{a=1}^{n} \hat{N}_a(r_1, s_1) = 1$$

  We generated 1000 random points in the domain and checked for this property for the linear 3-node and quadratic 6-node triangular shape functions. It was satisfied with a numberical tolerance of 1×10^{-15} in all cases.

- Partition of nullity: The shape function derivatives should satisfy

  $$\sum_{a=1}^{n} \hat{N}_{a,\alpha}(r, s) = 0 \qquad \alpha = 1, 2$$

  at all points in the standard domain. For 1000 randomly generated points in the domain this property was found to be satisfied with a numerical tolerance of 1×10^{-15} for both the 3-node linear and 6-node quadratic shape functions.

- Consistency: The shape function derivatives calculated by our code should tally with finite difference approximations of the derivatives within a suitable numerical tolerance. To get finite difference approximation of derivative with respect to one of the coordiantes r and s we perturb that co-ordinate with a suitably small value while holding the other coordinate constant. For example, using central finite differences we can write

  $$\hat{N}_{a,r} \approx \frac{\hat{N}_a(r+h, s) - \hat{N}_a(r-h, s)}{2h}$$

  where h is the perturbation. The value of h was chosen to be of the order 10^{-6}. For values of h smaller than this, large rounding errors make the approximation to have errors larger than $\mathcal{O}(h)$. For both the 3-node linear and 6-node quadratic shape functions the derivatives calculated by our code tallied with the corresponding central finite difference approximations to an acceptable tolerance of 10^{-6}.

- C^0-completeness: For a finite element formulation to converge to a solution, one of the requirements is that the shape functions should be able to exactly reproduce a linear field. Irrespective of the actual functional form of the solution field over the full domain, if it is discretized over sufficiently small elements then the local form of the solution over the element can always be reasonably approximated using linear functions. This is the rationale behind C^0-completeness requirement. To test it we will use a linear polynomial in (r, s), that is, $p(r, s) = ar + bs + c$ where a, b and c are randomly chosen co-efficients. We will evaluate

p at the nodes of the standard domain to get $p(r_a, s_a)$. For a randomly chosen point (r_1, s_1) in the domain, C^0-completeness requires

$$p(r_1, s_1) = \sum_{a=1}^{n} \hat{N}_a(r_a, s_a)p(r_a, s_a)$$

For both the 3-node linear and 6-node quadratic shape functions we found this to be true with an error of the order 10^{-14}. The error is small enough to consider this verification test as passed.

2.5.2 Verification of Quadrature Implementation

A Gaussian quadrature rule that uses n points should exactly reproduce integration of a polynomial of degree less than or equal to $2n-1$. We have implemented 1-point and 3-point quadrature rules. But our shape functions are only linear or quadratic. So we will check, if our 1-point quadrature code can reproduce integration of a linear polynomial in (r, s) and if our 3-point quadrature code can reproduce integration of a quadratic polynomial in (r, s) exactly over the standard domain of Figure 2.1.

- For a linear polynomial,

$$\begin{aligned} I_1 &= \int_0^1 \int_0^r (ar + bs + c)\,\mathrm{d}s\mathrm{d}r \\ &= \frac{a}{3} + \frac{b}{6} + \frac{c}{2} \end{aligned}$$

  For 1-point scheme the sampling point is $(r_1, s_1) = ((2/3), (1/3))$ and $\hat{w}_1 = 1$. Area of the standard domain is $\hat{\Omega} = (1/2)$. We want to check if our code returns

$$(ar_1 + bs_1 + c)\hat{w}_1\hat{\Omega} = I_1$$

  We found this to satisfied for each of 1000 random choices for a, b and c with a tolerance of the order 10^{-15}.

- For a quadratic polynomial,

$$\begin{aligned} I_3 &= \int_0^1 \int_0^r (ar^2 + bs^2 + crs + dr + es + f)\,\mathrm{d}s\mathrm{d}r \\ &= \frac{1}{24}(6a + 2b + 3c + 8d + 4e + 12f) \end{aligned}$$

  For 3-point scheme the sampling points and weights are given in Table 2.1. Area of the standard domain is $\hat{\Omega} = (1/2)$. We want to check if our code returns

$$\sum_{q=1}^{3}(ar_q^2 + bs_q^2 + cr_qs_q + dr_q + es_q + f)\hat{w}_q\hat{\Omega} = I_3$$

  We found this to be satisfied for each of 1000 random choices for a, b, c, d, e and f with a tolerance of the order 10^{-15}.

2.5.3 Verification Tests for Isoparametric Triangular Elements

Consistency

We have implemented the calculation of strain energy, internal force and stiffness modulus for the isoparametric triangular elements. These quantities are interrelated as

$$f_{ia}^{\text{int}} = \frac{\partial W}{\partial x_{ia}} \qquad \text{and}$$

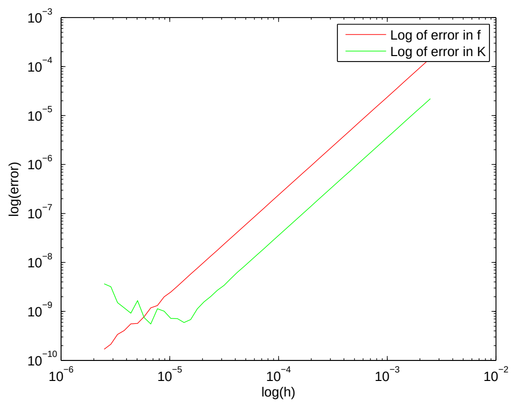
$$K_{iakb} = \frac{\partial f_{ia}^{\text{int}}}{\partial x_{kb}}$$

We can use these two equations to check our implementation for bugs. If there are no bugs, the finite difference approximation of derivatives of W and f_{ia}^{int} with respect to x_{ia} and x_{kb} should match f_{ia}^{int} and K_{iakb}, obtained directly by our code, respectively to acceptable numerical tolerance. The central finite difference approximation of the above equations are given as

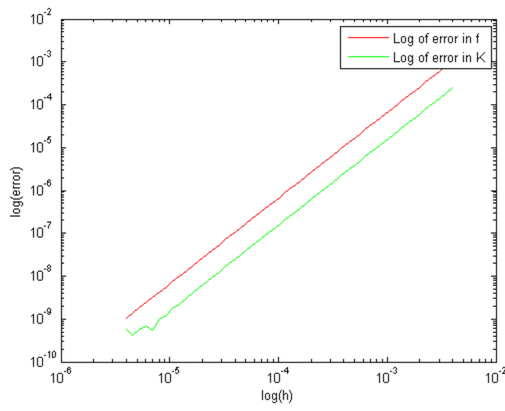
$$f_{ia}^{\text{int}} \approx \frac{W(x_{ia}+h) - W(x_{ia}-h)}{2h}$$

$$K_{iakb} \approx \frac{f_{ia}^{\text{int}}(x_{kb}+h) - f_{ia}^{\text{int}}(x_{kb}-h)}{2h}$$

We chose 50 values for perturbation $h \in (10^{-6}, 10^{-3})$. The error in the numerical derivatives scaled quadratically with h as can be seen from Figure 2.3. For the smaller values of h the effect of large rounding errors can be noticed in Figure 2.3a.



(a) Linear 3-node triangular element



(b) Quadratic 6-node triangular element

Figure 2.3: Log-log plot of error versus h shows straight lines with slope 2 indicating error of $\mathcal{O}(h^2)$ as expected for a central finite difference numerical differentiation. Smaller vlues of h introduce large rounding errors as seen in Figure 2.3a

Rank of Stiffness Matrix

The fourth order stiffness modulus K_{iakb} can be *unrolled* into a square 2-D matrix of size $ia \times kb$. For linear 3-node triangular element this will give a 6×6 matrix and for the quadratic 6-node triangular element we get a 12×12 matrix. A $n \times n$ matrix can have maximum of n non-zero

Table 2.2: Rank of stiffness matrix for various combinations of shape functions and order of Gaussian quadrature

Deformation	Shape Function	Quadrature	Rank
Zero	Linear	1-point	3
	Quadratic	1-point	3
	Quadratic	3-point	9
Finite	Linear	1-point	4
	Quadratic	1-point	4
	Quadratic	3-point	10

eigenvalues. Then the *rank* of that matrix is said to be n. Each zero eigenvalue reduces the rank by 1 which makes the matrix *rank deficient*. We calculated the rank of the stiffness matrix for the case of zero deformation i.e. $\mathbf{X} = \mathbf{x}$ and for finite deformation for different combinations of shape function and quadrature order choices. The results are presented in Table 3.1. A discrete finite-element model must give unique solution in the same way as the corresponding continuous mathematical model. In a elasticity problem, rigid body motions do not result in any internal forces and thus are called zero-energy modes. Zero-energy modes make the solution degenerate i.e. non-unique. Each zero eigen-value of the stiffness matrix should represents a rigid-body mode and we should not have any extra zero-energy modes in our finite element implementation than as physically expected. For zero deformation case, there are three degrees of freedom — two translations and one rotation. Hence, we expect the stiffness matrix to be rank-deficient by 3. For finite deformation case there are only two degrees of freedom — the translations. Hence, we expect the matrix to be rank deficient by 2. The results presented in Table 3.1 are consistent with our expectations for all but the two cases when we use 1-point Gaussian quadrature rule with a quadratic shape function. A 1-point Gaussian quadrature rule can exactly reproduce an integration of a polynomial of degree less than or equal to 1. For exactly reproducing integration of quadratic shape function we will need at least 3-point quadrature. By using lesser number of sampling points we are introducing extra non-physical zero body modes.

2.6 Source Code Listing

The source code files for this assignment are listed below.

1. `aspectRation.m`:This function takes coordinates of vertices of a triangle as input and returns its aspect ratio as an output. We use it to ensure that we do not generate skewed triangles which can cause problems due determinant of F approaching zero.
2. `checkOutwardNormal.m`: This function takes coordinates of vertices of a triangle as an input and checks if the sequence of the vertices is consistent with the node numbering of the standard parametric triangle. This is used to ensure that we don't get deformation gradients with negative determinant.
3. `findStiffnessRank.m`: This function accepts a four dimensional array as an input and unrolls it into a 2D matrix and returns the number of non-zero eigenvalues of the unrolled matrix as output.
4. `neoHookean.m` This is the 3D neo-Hookean constitutive model implementation from HW 1.

5. `planeStressNH.m` This is the plane-stress implementation of the 3D neo-Hookean consitutive model as developed for HW 1.

6. `Stiffness_rank.m` This is a script file that calculates and prints a table for the rank of stiffness matrix for different combination of shape function and quadrature rule choices.

7. `T3Lin.m` This function takes (r, s) coordinates as input and returns the shape function and its derivatives evaluated for a linear 3-node triangular element at that point.

8. `T6quad.m` This function takes (r, s) coordinates as input and returns the shape function and its derivatives evaluated for a quadratic 6-node triangular element at that point.

9. `T3Lin_Verification.m` This script runs the various tests for the linear 3-node shape functions.

10. `T6quad_Verification.m` This script runs the various tests for the quadratic 6-node shape functions.

11. `T3LinEle.m` This function takes X, x, quadrature order and the material properties μ and λ as input and returns the strain energy, internal force and stiffness modulus for a linear 3-node triangular finite element.

12. `T6QuadEle.m` This function takes X, x, quadrature order and the material properties μ and λ as input and returns the strain energy, internal force and stiffness modulus for a quadratic 6-node triangular finite element.

13. `T3LinEle_Verification.m`: This script runs and prints the results of various tests for the linear 3-node triangular element.

14. `T6QuadEle_Verification.m`: This script runs and prints the results of various tests for the quadratic 6-node triangular element.

15. `TriGaussQuad.m`: This function takes the order of Gaussian quadrature as input and returns the corresponding sampling points and weights as output for a standard triangular domain.

16. `TriGaussQuad_Verification.m`: This script runs verification test for the Gaussian quadrature implementation and prints its results.

Chapter 3

Assignment 3

Introduction

In this assignment, our goals are to

- implement 3-node and 6-node triangular membrane elements
- implement assembly of force and stiffness matrices for a mesh of these elements
- implement finite element analysis of some example problems using the membrane elements

3.1 Finite Element Formulation of a Membrane

Membrane theory can be derived from shell theory with the assumption that the shell is very thin and does not offer resistance to bending or transverse shear. We also assume that behaviour of the membrane through its thickness is same as that for the mid-surface and hence it is sufficient to calculate the behaviour of just the mid-surface. The change in the thickness direction can be represented by a single parameter, the thickness stretch, λ.

Let $\mathbf{a}$ and $\mathbf{A}$ denote the curvilinear basis vectors in the current and reference configurations of the mid-surface respectively. If x_{ia} and X_{ia} denote the co-ordinates of the nodes of the triangular membrane element in current and reference configuration respectively then tangent basis vectors in curvilinear frame are given as

$$a_\alpha = \mathbf{x}_{,\alpha} = x_{ia}N_{a,\alpha}$$
$$A_\alpha = \mathbf{X}_{,\alpha} = X_{ia}N_{a,\alpha}$$

where N_a are the shape functions for triangular elements derived in HW 2 and $\alpha = 1, 2$. The third curvilinear tangent basis vector is assumed to always be normal to the other two. This means,

$$\mathbf{a}_3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{|\mathbf{a}_1 \times \mathbf{a}_2|}$$
$$\mathbf{A}_3 = \frac{\mathbf{A}_1 \times \mathbf{A}_2}{|\mathbf{A}_1 \times \mathbf{A}_2|}$$

For further calculations, we require the dual basis vectors in the refernce configuration of mid-surface, $\mathbf{A}^i$. These can be derived as follows

$$\begin{aligned} A_{ij} &= \mathbf{A}_i \cdot \mathbf{A}_j \\ A^{ij} &= A_{ij}^{-1} \\ \mathbf{A}^i &= A^{ij}\mathbf{A}_j \end{aligned}$$

where A_{ij} is the metric tensor and A^{ij} is the inverse metric tensor. It should be pointed out that as a consequence of our assumptions in membrane theory we have $\mathbf{A}_3 = \mathbf{A}^3$.

3.1.1 Deformation Gradient

Using the curvilinear basis vectors derived in the previous section we can write the deformation gradient of the element as

$$\mathbf{F} = \mathbf{a}_\alpha \otimes \mathbf{A}^\alpha + \lambda \mathbf{a}_3 \otimes \mathbf{A}^3$$

where λ is the thickness stretch ratio. We need to provide an initial guess for λ to start with. Then, using the deformation gradient and the Lamé coefficients, we can calculate strain energy density, w, the first Piola-Kirchhoff stress tensor $\mathbf{P}$ and the stiffness modulus C_{iJkL} based on the compressible neo-Hookean constitutive model developed in HW 1. Now, we are in position to implement plane stress condition for the triangular element.

3.1.2 Plane-stress condition

The component of traction vector in the direction of surface normal in the current configuration can be written as

$$T(\lambda) = \mathbf{a}_3 \cdot (\mathbf{P}(\lambda)\mathbf{A}_3)$$

To enforce planes stress we need to solve for λ such that $T(\lambda) = 0$. We will use Newton-Raphson method to solve for λ. We need Taylor series expansion of $\mathbf{P}(\lambda)$ and $T(\lambda)$.

$$\begin{aligned} \mathbf{P}(\lambda + d\lambda) &\approx \mathbf{P}(\lambda) + \frac{\partial \mathbf{P}}{\partial \lambda}\mathrm{d}\lambda \\ &= \mathbf{P}(\lambda) + \frac{\partial \mathbf{P}}{\partial \mathbf{F}} : \frac{\partial \mathbf{F}}{\partial \lambda}\mathrm{d}\lambda \\ &= \mathbf{P}(\lambda) + \mathbb{C} : \frac{\partial \mathbf{F}}{\partial \lambda}\mathrm{d}\lambda \\ &= \mathbf{P}(\lambda) + \mathbb{C} : (\mathbf{a}_3 \otimes \mathbf{A}^3)\mathrm{d}\lambda \end{aligned}$$

Therefore,

$$\begin{aligned} T(\lambda + \mathrm{d}\lambda) &\approx \mathbf{a}_3 \cdot (\mathbf{P}(\lambda + \mathrm{d}\lambda)\mathbf{A}_3) \\ &= T(\lambda) + \mathbf{a}_3 \cdot \left(\mathbb{C} : (\mathbf{a}_3 \otimes \mathbf{A}^3)\mathbf{A}^3\right)\mathrm{d}\lambda \\ &= T(\lambda) + (\mathbf{a}_3 \otimes \mathbf{A}^3) : \mathbb{C} : (\mathbf{a}_3 \otimes \mathbf{A}^3)\mathrm{d}\lambda \end{aligned}$$

If $T(\lambda + \mathrm{d}\lambda) = 0$ then

$$\mathrm{d}\lambda = -T(\lambda)\left[(\mathbf{a}_3 \otimes \mathbf{A}^3) : \mathbb{C} : (\mathbf{a}_3 \otimes \mathbf{A}^3)\right]^{-1}$$

This gives the update to λ for the Newton-Raphson interations. The term in the bracket is C_{3333}, component of stiffness modulus in the curvilinear frame. It can be computed in terms of C_{iJkL} which is in the lab-frame as

$$C_{3333} = (\mathbf{a}_3)_i(\mathbf{A}_3)_J C_{iJkL}(\mathbf{a}_3)_k(\mathbf{A}_3)_L$$

3.1.3 Strain Energy

After solving for λ using Newton-Raphson method for plane stress, we can re-calculate $\mathbf{F}$. Thus, we can calculate strain energy density, w, as per the compressible neo Hookean constitutive model of HW 2. The strain energy of the element, W, can be written as

$$\begin{aligned} W &= \int_{V^e} w \, \mathrm{d}V^e \\ &= \int_{\Omega^e} w \, \mathrm{d}\Omega^e H \end{aligned}$$

where H is membrane thickness and Ω is the mid-surface. We need to evaluate the integral using Gauss quadrature for triangular region developed in HW 2. Let n_G be the number of Gauss quadrature points, $\hat{\Omega}$ be the area of standard parametric triangle and $\hat{w}$ be the weight for the Gauss quadrature point. Then we can write

$$\begin{aligned} W &= \int_{\Omega^e} w \, \mathrm{d}\Omega^e H \\ &= \int_{\hat{\Omega}} wH\sqrt{A} \, \mathrm{d}\hat{\Omega} \\ &= \sum_{q=1}^{n_G} w\hat{w}_q H\sqrt{A}\hat{\Omega} \end{aligned}$$

where $A = |A_{ij}|$ is determinant of the metric tensor.

3.1.4 Internal and External Forces

The internal nodal forces can be written in terms of stress-resultant $\mathbf{n}^\alpha$.

$$\mathbf{n}^\alpha = \mathbf{P} \cdot \mathbf{A}^\alpha H$$

The internal forces are

$$\begin{aligned} f_{ia}^{\text{int}} &= \int_{\hat{\Omega}} n_i^\alpha N_{a,\alpha}\sqrt{A} \, \mathrm{d}\hat{\Omega} \\ &= \sum_{q=1}^{n_G} n_i^\alpha N_{a,\alpha}\Big|_{\theta_q} \hat{w}_q\sqrt{A}\hat{\Omega} \end{aligned}$$

The external forces are obtained from the distributed transverse load on the element, $\mathbf{f}$,

$$\begin{aligned} f_{ia}^{\text{ext}} &= \int_{\hat{\Omega}} f_i N_a \sqrt{A}\, \mathrm{d}\hat{\Omega} \\ &= \sum_{q=1}^{n_G} f_i N_a \bigg|_{\theta_q} \hat{w}_q \sqrt{A} \hat{\Omega} \end{aligned}$$

3.1.5 Stiffness Modulus

To calculate the element stiffness modulus, K_{iakb}, we need to calculate the curvilinear components of Kirchhoff stress tensor, $\boldsymbol{\tau}$,

$$\begin{aligned} \boldsymbol{\tau} &= \mathbf{P}\mathbf{F}^T \\ &= \mathbf{F}(\mathbf{F}^{-1}\mathbf{P})\mathbf{F}^T \\ &= \mathbf{F}\mathbf{S}\mathbf{F}^T \end{aligned}$$

where $\mathbf{S} = \mathbf{F}^{-1}\mathbf{P}$ is the second Piola-Kirchhoff stress tensor. $\mathbf{S}$ and $\boldsymbol{\tau}$ have the same components but different basis vectors. The relationship can be written as

$$\mathbf{S} = \tau^{\alpha\beta} \mathbf{A}_\alpha \otimes \mathbf{A}_\beta$$

where $\tau^{\alpha\beta}$ are the components of $\boldsymbol{\tau}$ in the curvilinear frame which can be obtained as

$$\tau^{\alpha\beta} = \mathbf{A}^\alpha \cdot \left(\mathbf{S}\mathbf{A}^\beta\right)$$

We also need the components of stiffness modulus in terms of $\boldsymbol{\tau}$

$$C^{ijkl} = \frac{\partial \tau^{ij}}{\partial a_{kl}}$$

where a_{kl} are components of metric tensor in current configuration. We can obtain this using the stiffness modulus in terms of the second Piola-Kirchoff stress tensor

$$C_{IJKL} = \frac{\partial S_{IJ}}{\partial C_{KL}}$$

where C_{KL} are components of the left Cauchy Green strain tensor. We can calculate C_{IJKL} from C_{iJkL} which is the stiffness modulus in terms of the first Piola-Kirchoff stress tensor.

$$C_{IJKL} = \frac{1}{2} F^{-1}_{Ii} F^{-1}_{Kk} \left(C_{iJkL} - \delta_{ik} S_{JL}\right)$$

Now,

$$C^{ijkl} = C_{IJKL} (\mathbf{A}^i)_I (\mathbf{A}^j)_J (\mathbf{A}^k)_K (\mathbf{A}^l)_L$$

where $(\cdot)_I$ represents the I^{th} component in lab-frame of the vector in the brackets. When we impose the plane-stress condition, we force $\tau^{33} = 0$. This requires modifying $C^{\alpha\beta\gamma\delta}$ such that it is consistent, that is, it can be obtained by differentiating $\tau^{\alpha\beta}$ with respect to $a^{\gamma\delta}$. The modified form is given as

$$\tilde{C}^{\alpha\beta\gamma\delta} = C^{\alpha\beta\gamma\delta} - \frac{C^{\alpha\beta 33}}{C^{3333}} C^{33\gamma\delta}$$

Now we have all components we need to write the element stiffness modulus.

$$
\begin{aligned}
K_{iakb} &= \underbrace{\int_{\hat{\Omega}} \left(\tilde{C}^{\alpha\beta\mu\nu} (\mathbf{a}_\beta \otimes \mathbf{a}_\nu)_{ik} N_{a,\alpha} N_{b,\mu} \right) \sqrt{A} \, \mathrm{d}\hat{\Omega}}_{\text{Material Stiffness}} \\
&+ \underbrace{\int_{\hat{\Omega}} \left(\tau^{\alpha\beta} N_{a,\alpha} N_{b,\beta} \right) \sqrt{A} \, \mathrm{d}\hat{\Omega}}_{\text{Geometric Stiffness}} \\
&= \sum_{q=1}^{n_G} \left(\tilde{C}^{\alpha\beta\mu\nu} (\mathbf{a}_\beta \otimes \mathbf{a}_\nu)_{ik} N_{a,\alpha} N_{b,\mu} \right) \Bigg|_{\theta_q} \hat{w}_q \sqrt{A} \hat{\Omega} \\
&+ \sum_{q=1}^{n_G} \left(\tau^{\alpha\beta} N_{a,\alpha} N_{b,\beta} \right) \Bigg|_{\theta_q} \hat{w}_q \sqrt{A} \hat{\Omega}
\end{aligned}
$$

It is important to note that in computer implementation the geometric and material stiffness components need to be computed in different loops because they don't have same number of repeated indices.

3.1.6 Verification Tests

We can verify our computer implementation of the triangular membrane elements using consistency test and by calculating rank of stiffness modulus in zero and finite deformation cases.

Consistency Test

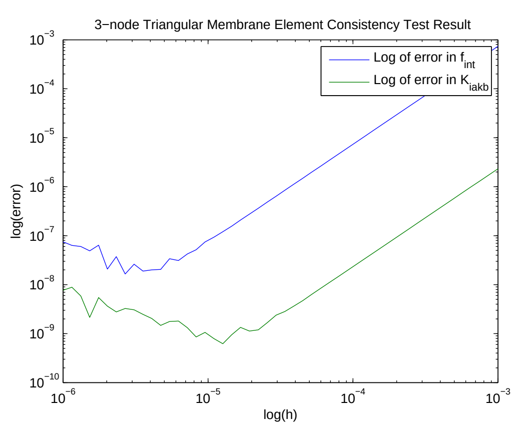
We have implemented the calculation of strain energy, internal force and stiffness modulus for the isoparametric triangular membrane elements. These quantities are interrelated as

$$
\begin{aligned}
f_{ia}^{\text{int}} &= \frac{\partial W}{\partial x_{ia}} \qquad \text{and} \\
K_{iakb} &= \frac{\partial f_{ia}^{\text{int}}}{\partial x_{kb}}
\end{aligned}
$$

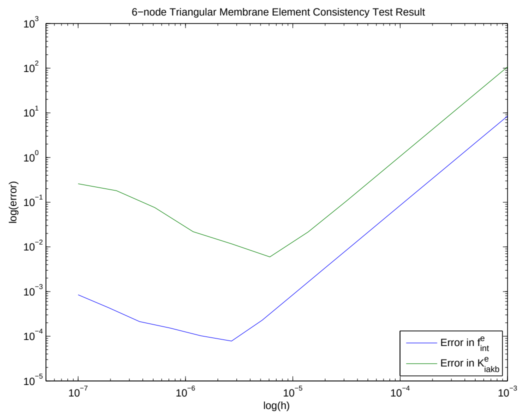
We can use these two equations to check our implementation for bugs. If there are no bugs, the finite difference approximation of derivatives of W and f_{ia}^{int} with respect to x_{ia} and x_{kb} should match f_{ia}^{int} and K_{iakb}, obtained directly by our code, respectively to acceptable numerical tolerance. The central finite difference approximation of the above equations are given as

$$
\begin{aligned}
f_{ia}^{\text{int}} &\approx \frac{W(x_{ia} + h) - W(x_{ia} - h)}{2h} \\
K_{iakb} &\approx \frac{f_{ia}^{\text{int}}(x_{kb} + h) - f_{ia}^{\text{int}}(x_{kb} - h)}{2h}
\end{aligned}
$$

We chose several values for perturbation $h \in (10^{-7}, 10^{-3})$. The error in the numerical derivatives scaled quadratically with h as can be seen from Figure 3.1. For the smaller values of h the effect of large rounding errors can be noticed in Figure 3.1a.



(a) 3-node triangular membrane element



(b) 6-node triangular membrane element

Figure 3.1: Log-log plot of error versus h shows straight lines with slope 2 indicating error of $\mathcal{O}(h^2)$ as expected for a central finite difference numerical differentiation. Smaller vlues of h introduce large rounding errors as seen in Figure 3.1a

Table 3.1: Rank of stiffness matrix for linear and quadratic membrane elements

Deformation	Shape Function	Quadrature	Rank
Zero	Linear	1-point	3
	Quadratic	3-point	9
Finite	Linear	1-point	6
	Quadratic	3-point	15

Stiffness Matrix Rank

The fourth order stiffness modulus K_{iakb} can be *unrolled* into a square 2-D matrix of size $ia \times kb$. For linear 3-node triangular element this will give a 9×9 matrix and for the quadratic 6-node triangular element we get a 18×18 matrix. We calculated the rank of the stiffness matrix for the case of zero deformation i.e. $\mathbf{X} = \mathbf{x}$ and for finite deformation for both linear and quadratic shape functions. The results are presented in Table 3.1. In case of zero deformation for 3-node element we have only 1 quadrature points. Thus, in 3-D we have 3 rotational and 3 translational degree of freedom for that point. Hence, the stiffness matrix is rank-deficient by 6 which gives rank as $9 - 6 = 3$. For zero-deformation case for 6-node element we have 3 quadrature points which form a triangle. We need to count number of unique rigid body motions such that this triangle remains undeformed. We can have 3 rigid-body translations, 3 rotations about the 3 sides of this triangle and 3 rotations about axis passing through each of the quadrature point and perpendicular to plane of the triangle. Thus, we have 9 degrees of freedom. This we should have rank as $18 - 9 = 9$. In case of finite deformation case, the rotational degrees of freedom are no longer zero-energy modes. So, we expect the rank to increase by 3 and 6 for the 3-node and 6-node elements respectively. Thus, the results presented in Table 3.1 are consistent with our expectations.

3.2 Assembly

In finite element analysis, the geometrical domain of the problem is discretized into a mesh of finite elements. Our mesh will be using the finite elements we formulated in the previous section. We need to compute the combined resultant energy, force and stiffness of the whole mesh based on contribution from each element. This is done in the process of assembly in finite element method.

3.2.1 Meta Arrays

We make use of two "meta" arrays during the assembly process.

- **IEN**: This is the ***E**lement **N**odes* array. It stores the mesh connectivity information. Every node in the mesh is assigned a *global node number.* Each row of **IEN** represents a single triangular element in the mesh. The columns of **IEN** store the global node numbers for the vertices of the triangle represented by that row. In case of 3-node triangular elements **IEN** is of size $n_{el} \times 3$ where n_{el} is the total number of elements in the mesh. For the 6-node triangular membrane element the size of **IEN** is $n_{el} \times 6$.

- **ID**: This is the ***D**estination* array. It stores the *global equation number* corresponding to each *global degree of freedom.* For our elements, each node in the mesh has 3 degrees of freedom associated with it. We call these as the 3 *local degrees of freedom* of each node. Each element has $3 \times n_{nodes\ per\ element}$ local degrees of freedom. So the total number of degrees of freedom in a mesh is given by

  $$\text{Total degrees of freedom} = 3 \times \text{total number of nodes}$$

  The degree of freedom numbers associated with a global node numbered a can be obtained as

  $$GDOF(a) = 3(a-1) + i \qquad \text{where} \quad i = 1, 2, 3$$

  In, each row of **ID** array the first column is the global degree of freedom number and the second column is the *global equation number.* If a particular degree of freedom in the mesh is constrained due to a prescribed boundary condition then the second column in **ID** array for that global degree of freedom is set to 0. *Thus, we assemble the force and stiffness matrices only for the unknown degrees of freedom.*

3.2.2 Potential Energy Assembly

Total *strain energy* of the mesh is the algebraic sum of strain energy of each element. The total *potential energy* of the mesh is the the difference between the total strain energy and the work done by external forces. Since, we know the external force acting on each node of each element and we can calculate the nodal displacements, we can calculate the external work done on the element as

$$W_e^{ext} = \mathbf{f}_e^{ext} \cdot \mathbf{u}_e$$

Therefore, we can write

$$\Pi = \sum_{e=1}^{n_{el}} (W_e^{\text{int}} - W_e^{\text{ext}})$$

3.2.3 Force Assembly

Using the **IEN** and **ID** arrays, we can get the global equation number for each local degree of freedom for each element. Consider the case of 3-node triangular element. It has 9 local degrees of freedom. The element force vectors will be column vectors with 9 elements, $\mathbf{f}^e$. Suppose the 4^{th} local degree of freedom corresponds to 115^{th} global equation number. Certainly, as the global equation number is non-zero, this is an unknown degree of freedom and thus we have to assemble its contribution to the global force matrix, $\mathbf{f}$. The force (either internal or external) assembly entails that

$$f_{115} \leftarrow f_{115} + f_4^e$$

where $\leftarrow$ indicates assignment operation in a computer program. After assembling both the internal and external forces we can calculate the global residual force vector as

$$\mathbf{r} = \mathbf{f}^{\text{int}} - \mathbf{f}^{\text{ext}}$$

3.2.4 Stiffness Assembly

Consider the case of 3-node triangular element. It has 9 local degrees of freedom. The stiffness matrix, $\mathbf{K}^e$ will, therefore, be of size 9×9. Suppose the 4^{th} and 6^{th} local degrees of freedom correspond to 115^{th} and 227^{th} global equation numbers respectively. Certainly, as the global equation numbers are non-zero, these are unknown degrees of freedom and thus we have to assemble their contribution to the global stiffness matrix, $\mathbf{K}$. The assembly entails that

$$\begin{aligned}
K_{115,115} &\leftarrow K_{115,115} + K^e_{4,4} \\
K_{115,227} &\leftarrow K_{115,227} + K^e_{4,6} \\
K_{227,115} &\leftarrow K_{227,115} + K^e_{6,4} \\
K_{227,227} &\leftarrow K_{227,227} + K^e_{6,6}
\end{aligned}$$

where $\leftarrow$ indicates assignment operation in a computer program. Since both K^e and K are symmetric, we could have used only three assignments instead of four.

3.2.5 Verification Tests

We will test our implementation for bugs using consistency test and by checking the rank of stiffness matrix for zero and finite deformations.

Consistency Test

The global potential energy Π, residual force $\mathbf{r}$ and the global stiffness modulus $\mathbf{K}$ are inter-related as

$$\begin{aligned}
r_{ia} &= \frac{\partial \Pi}{\partial x_{ia}} \\
K_{iakb} &= \frac{\partial r_{ia}}{\partial x_{kb}}
\end{aligned}$$

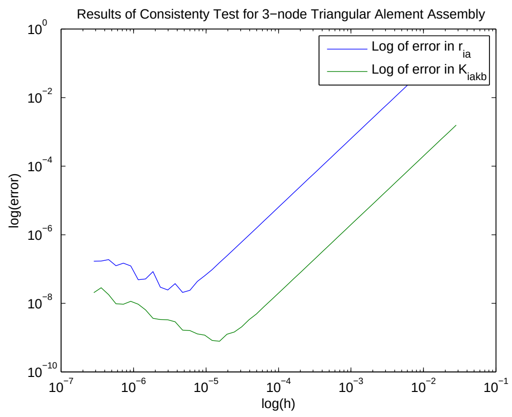
Table 3.2: Rank of Global Stiffness matrix after assembly for Linear and Quadratic membrane elements

Shape Function	Deformation	Rank
Linear	Zero	9
	Finite	9
Quadratic	Zero	24
	Finite	24

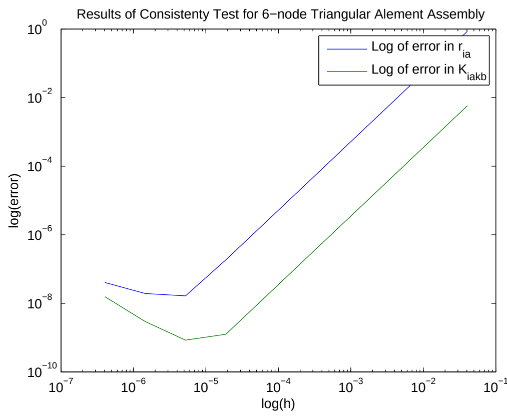
We can use these to test our implementation by comparing the results obtained from our assembly sub-routine with the numerical derivatives calculated using 3-point finite difference scheme. The equations for numerical derivatives are

$$r_{ia} \approx \frac{W(x_{ia}+h) - W(x_{ia}-h)}{2h}$$
$$K_{iakb} \approx \frac{r_{ia}(x_{kb}+h) - r_{ia}(x_{kb}-h)}{2h}$$

We chose several values for perturbation $h \in (10^{-7}, 10^{-3})$. The error in the numerical derivatives scaled quadratically with h as can be seen from Figure 3.2.



(a) Assembly of 3-node triangular elements

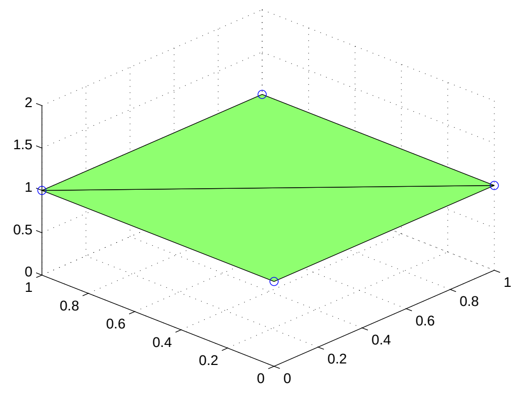


(b) Assembly of 6-node triangular elements

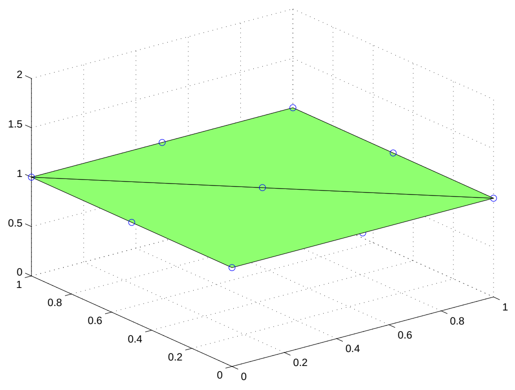
Figure 3.2: Log-log plot of error versus h shows straight lines with slope 2 indicating error of $\mathcal{O}(h^2)$ as expected for a central finite difference numerical differentiation. Smaller vlues of h introduce large rounding errors.

Rank of Stiffness Matrix

The global stiffness matrix is a square matrix with number of rows and columns equal to the number of unknown global degrees of freedom. We chose a square mesh of two triangular elements, as shown in Figure 3.3 and constrained one of the corner nodes to have zero displacement in all three directions. Then we prescribed zero and finite displacements for other nodes of the element and calculated the rank of stiffness matrix in both cases. We did this for both the 3-node and the 6-node element. The results are summarized in the table 3.2 For the 3-node element mesh, there are 4 global nodes and 12 global degrees of freedom. We have constrained 3 of these degrees



(a) Mesh of 3-node elements



(b) Mesh of 6-node elements

Figure 3.3: Meshes used to verify the rank of stiffness matrix obtained after assembly. The nodes in the mesh have been circled.

of freedom. So we are left with 9 unknown degrees of freedom. Each row and column of the 9×9 assembled stiffness matrix corresponds to one of the unknown degree of freedom. Since, we want our system of equations to give us a unique answer for each unknown degree of freedom we expect that the assembled stiffness matrix has full rank. In our example, it should have rank 9. Similar, analysis for the mesh of 6-node elements shows that we should expect the assembled stiffness matrix to have rank 24. The results shown in Table 3.2 are thus consistent with our expectations. It is very important to highlight at this point that in the zero deformation case, it was necessary to perturb the unconstrained degrees of freedom in the transverse direction by a small amount. Otherwise, the membrane mesh would be perfectly flat and offer no resistance in the transverse direction giving rise to unwanted rank deficiency in the stiffness matrix.

3.3 Solving the Equilibrium Problem

The object of our finite element analysis is to calculate nodal displacements that lead to the system being in equilibrium characterized by zero residual force vector. The residual force vector is a non-linear function of the current configuration and we can use Newton-Raphson method to solve for equilibrium displacement as follows

3.3.1 Newton-Raphson Method for Equilibrium

$$\begin{aligned}\mathbf{r}(\mathbf{x}+\mathbf{u}) &\approx \mathbf{r}(\mathbf{x}) + \frac{\partial \mathbf{r}}{\partial \mathbf{x}}\mathbf{u} \\ &= \mathbf{r}(\mathbf{x}) + \mathbf{K}\mathbf{u}\end{aligned}$$

We want $\mathbf{r}(\mathbf{x}+\mathbf{u}) = 0$, therefore,

$$\mathbf{u} = \mathbf{K}^{-1}\mathbf{r}$$

The $\mathbf{u}$ represents the update required to our current guess for $\mathbf{x}$. We repeat this till the residual is driven to zero to acceptable numerical tolerance.

3.3.2 Incremental Solution Strategy

When the mesh is subjected to relatively large forces or displacements, the Newton iterations will not converge unless we have a very good initial guess. To resolve this problem we can adopt an incremental solution strategy. We can divide the total prescribed force or displacement in small increments such that for the first increment from zero our Newton iterations for equilibrium converge with the reference configuration itself as our initial guess for the solution. For subsequent increments in load (force or displacement) we can use the solution obtained for the previous increment as the initial guess.

3.4 Applications

3.4.1 Planar Isotropic Stretch

We have to solve the problem of a square sheet stretched by a planar isotropic deformation($F_{11} = F_{22} = \lambda, F_{12} = F_{21} = 0$) imposed by prescribing displacements of nodes on the boundaries. Figure 3.4 shows the reference and deformed configuration mesh, obtained by our finite element analysis, superimposed. Using the incremental solution strategy discussed in previous section, we

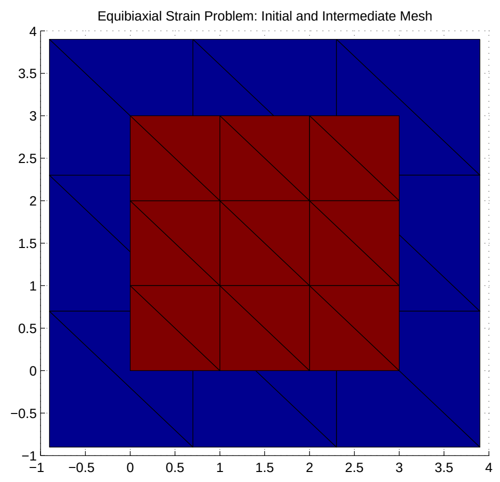


Figure 3.4: The reference configuration mesh has been superimposed on the deformed mesh. Brown mesh represents the reference configuration and blue mesh represents the deformed configuration obtained by our finite element analysis. The square sheet has stretched isotropically as expected.

solved for the deformed configuration for various planar isotropic stretches and compressions of the initial mesh. As a post-processing step for every isotropic stretch, we calculated the mean deformation gradient for all elements and checked if the deformation gradient for any element deviated significantly from the mean. In all cases, the deviation was within numerical tolerance and thus we found that the deformation gradient was uniform across the mesh. It should be noted that we had to prescribe the transverse displacement of all nodes to be zero. Otherwise, as the reference configuration is a perfectly flat mesh, the stiffness matrix has zero stiffness components in the transverse direction which leads to a singular matrix that cannot be inverted as required for our equilibrium Newton iterations. By prescribing the transverse displacement we ensure that those degrees of freedom are not assembled into our global stiffness matrix.

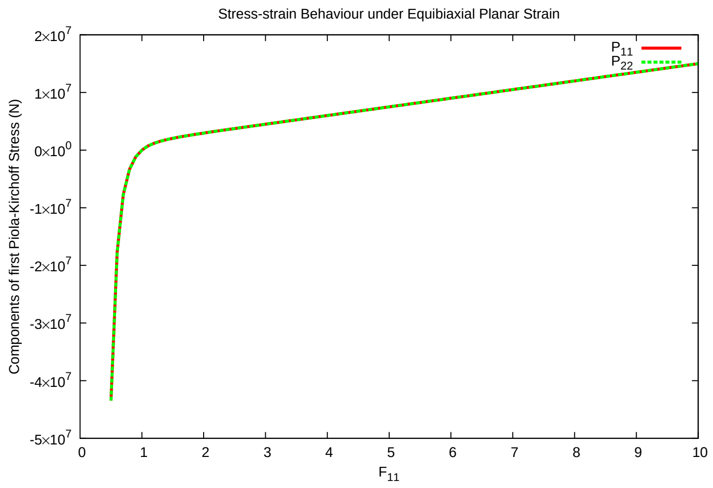
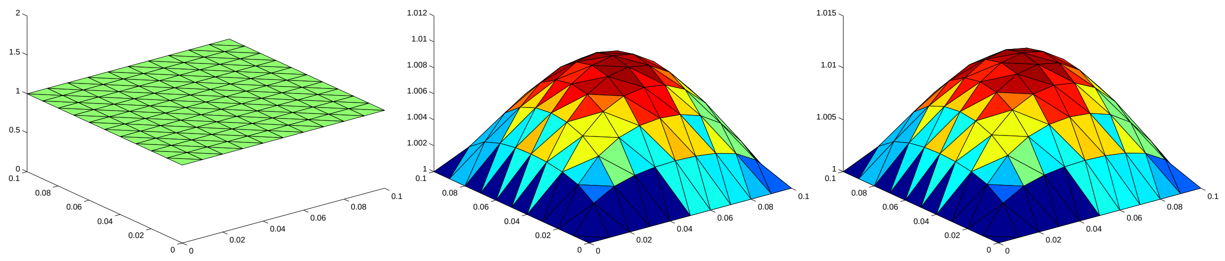


Figure 3.5: P_{11} and P_{22} components of first Piola-Kirchhoff stress tensor plotted against a component of deformation gradient F_{11}. This result is same as obtained in HW 1 for compressible neo Hookean material.

3.4.2 Simply Supported Membrane under Transverse Load

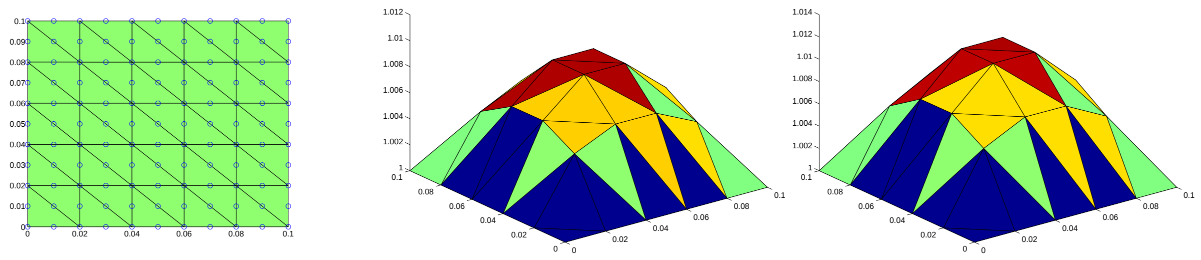
We have to solve the problem of a simply supported square membrane under a uniform transverse load. Side length of the square mesh was chosen to be $L = 10\ cm$. Reference thickness $H = 0.1\ cm$, shear modulus, $\mu_0 = 4 \times 10^5\ N/m^2$ and $\lambda_0 = 10\mu_0$. We will use incremental solution strategy for applied transverse load. We will increase it in increments of $10\ N/m^2$ from 0 to $1000\ N/m^2$. Figure 3.6 shows the mesh at zero, half the maximum and the maximum load full load. We can



(a) Undeformed mesh, 200 elements (b) Deformed mesh under half of maximum load (c) Deformed mesh under maximum load

Figure 3.6: The reference configuration mesh and the deformed mesh at half load and full load using 3-node triangular membrane elements are shown. The maximum deflection at half load is only slightly less than the deflection at full-load.

observe that the deformed mesh at half-load is almost identical to the deformed mesh at full load. It indicates that the increase in deflection goes on decreasing rapidly as the transverse load increases. This is expected because as the membrane deflects under the transverse load its area increases and it has to stretch. The more it stretches, the stiffer it becomes and thus it takes larger force to stretch it further. This relationshiip between maximum deflection andthe transverse load can also be noticed in Figure 3.9. Figure 3.7 shows the mesh of 6-node triangular membrane elements at zero, half of the maximum and maximum transverse load. Figure 3.7a is the top-view of the



(a) Top view of undeformed mesh, 50 elements (b) Deformed mesh under half of maximum load (c) Deformed mesh under maximum load

Figure 3.7: The reference configuration mesh and the deformed mesh at half load and full load using 6-node triangular membrane elements are shown. The deformed mesh plots are misleading because the triangles in the mesh don't show the mid-side nodes.

initial mesh. It also shows the mid-side nodes. 6-node triangular elements have mid-side nodes which make the sides of the triangle quadratic functions instead of straight lines. But this detail is missing from Figure 3.7. A better representation of the displacement field is shown in Figure 3.8. It is obtained by doing a Delaunay triangulation of the cloud of points formed by the nodes of the original mesh and then plotting the displacement field over it. Like Figure 3.6, this representation also has the drawback that the triangles in the mesh are not actually the elements but it gives better visualization of the displacement field. 6-node quadratic element mesh gives more accurate solution with lesser number of elements as compared to the mesh of 3-node triangular elements.

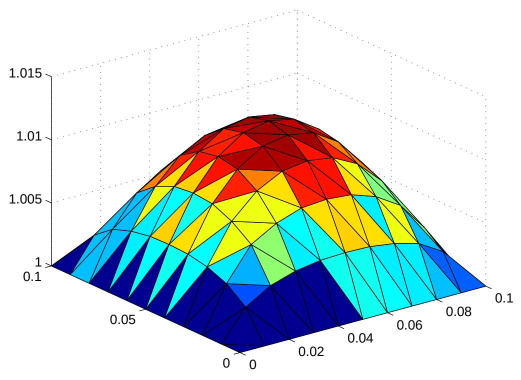
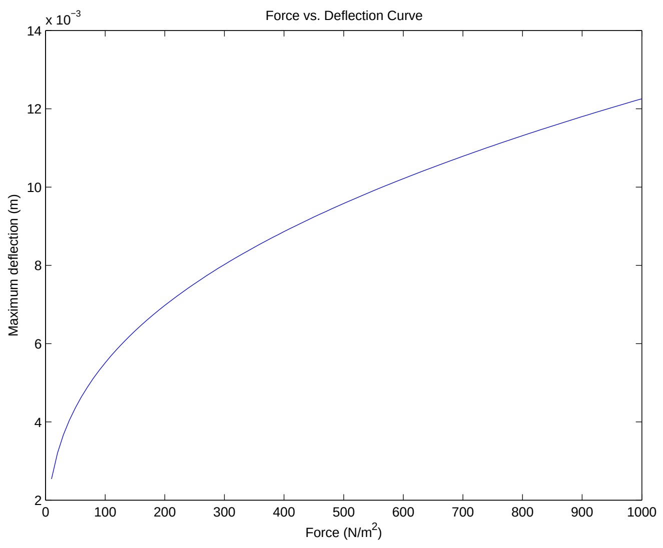


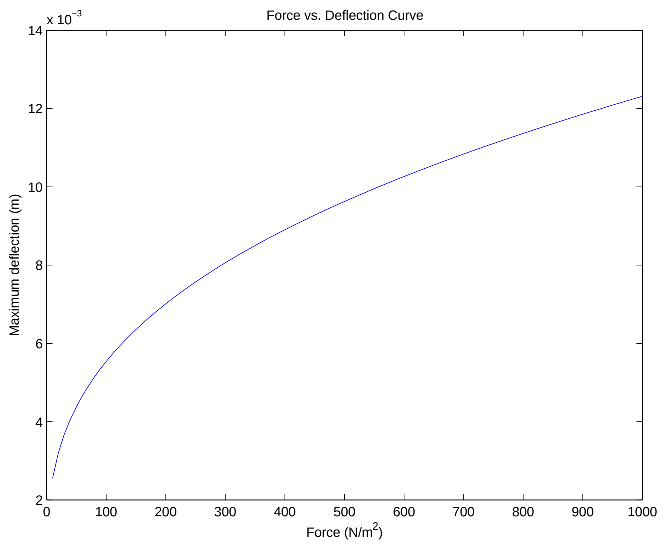
Figure 3.8: Alternative representation of deformed 6-node triangular element mesh. But the triangles in this figure are not the elements.

3.4.3 Spherical balloon

We have to sovle the problem of a spherical balloon of initial radius $R = 10\ cm$ and thickness $H = 0.1\ cm$ under a uniform internal pressure. We used incremental solution strategy as before. The initial mesh and several deformed mesh are shown in Figure 3.10. Figure 3.11 shows the variation in stretch-ratio defined as $\lambda = r/R$ where r and R are the current and reference configuration radii. The plot is similar to Figure 3.5. The Newton-Raphson iterations for enforcing plane-stress



(a) For 3-node element mesh



(b) For 6-node elements mesh

Figure 3.9: Force-deflection curve for the 3-node and 6-node element meshes. The slope decreases as the maximum deflection increases. This shows that it requires more force to bring about equal increase in maximum deflection as the membrane deforms.

at element level could not drive the normal component of traction vector below 1×10^{-6} after a pressure of 60 N/m^2 because the Newton updates became of the order of machine precision. But as the component is still much smaller than the traction vector in the plane of membrane, we continued to solve for equilibrium for larger pressures till 220 N/m^2 when the Newton iterations for the equilibrium stopped converging.

3.5 Source Code Listing

- `assemblyT3Lin.m`: This function performs assembly for a mesh of 3-node triangular membrane elements.
- `assemblyT6Quad.m`: This function performs assembly for a mesh of 6-node triangular membrane elements.
- `BalloonProblem.m` This is the driver script from the balloon problem.
- `calcAllCs.m`: This function performs the plane-stress enforcement at element level and calculates various stiffness moduli required to calculate the membrane element stiffness modulus.
- `calcFandP.m`: This function is a post-processing function used to calculate deformation gradient and first Piola-Kirchoff stress tensor. It is used to check for uniform deformation gradient.
- `convertToT6Mesh.m`: This function is used to add mid-side nodes to a 3-node triangular element mesh and regenerate a connectivity matrix to get a 6-node triangular element mesh.
- `EquibiaxialStrain.m`: This is the driver script for the isotropic stretch problem.

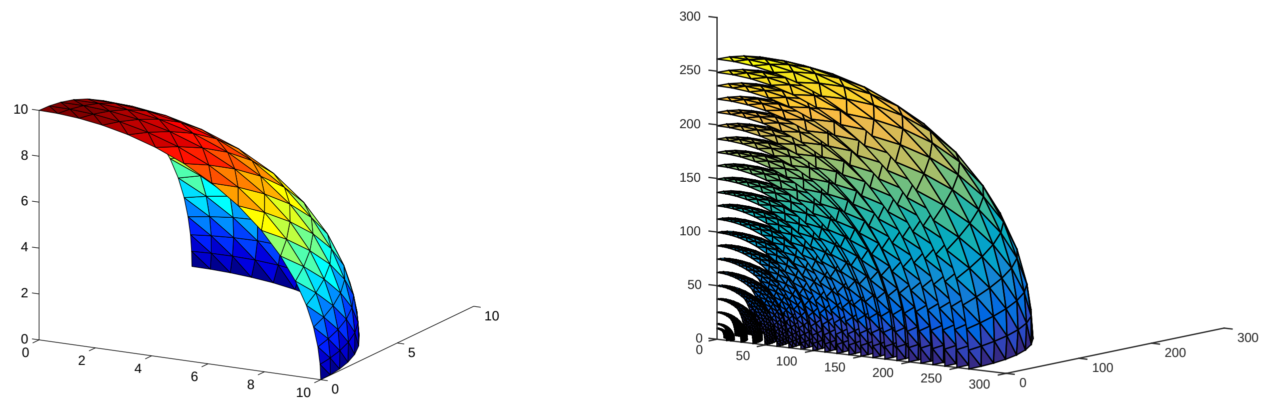

(a) Initial Mesh of radius $R = 10\ cm$ (b) Several deformed mesh of increasing radii

Figure 3.10: The initial mesh and many deformed meshes of an octant of a spherical membrane subjected to uniform internal pressure.

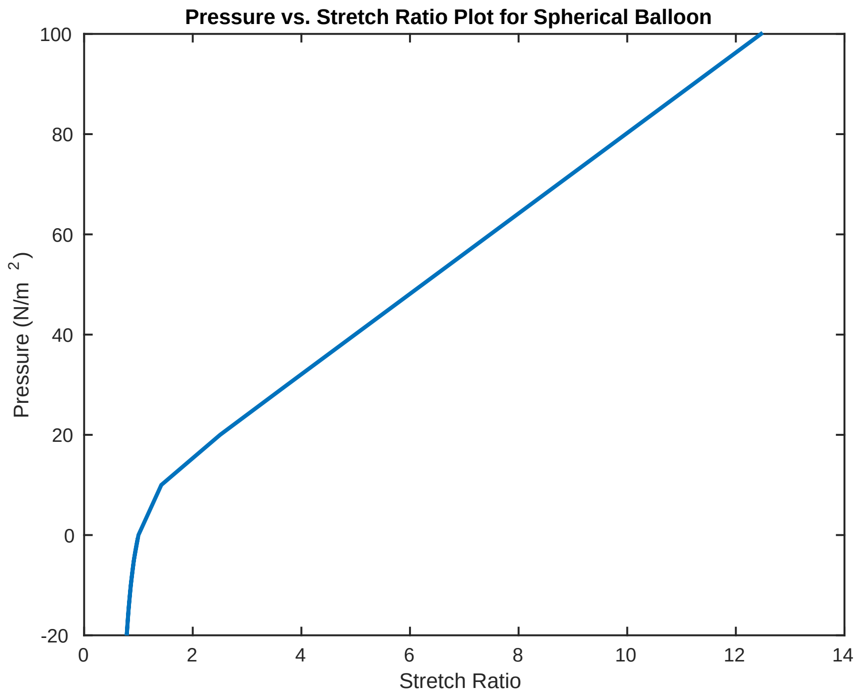


Figure 3.11: Pressure versus stretch-ratio plot for the balloon problem. It shows similar behavior as the equibiaxial strain problem in Figure 3.5

- `equiTriMesh.m`: This fucntion creates a mesh of triangles in a large equilateral triangles. It is used to generate the spherical balloon octant mesh.
- `findStiffnessRank.m`: This function calculates rank of a matrix.
- `getBCmatrix.m`: This function accepts a cell-array containing infformation about boundary condition in terms of nodal co-ordinates and local degree of freedom for a mesh and generates

a matrix containing the boundary condition specified with respect to global degree of freedom number.

- `neoHookean.m`: This function calculates the strain energy density, first Piola-Kirchhoff stress tensor and the corresponding tangent modulus.
- `plotMesh.m`: This function generates a 3-D plot of a mesh and saves it to file.
- `plotQuiver.m`: This function plots force vector field on a mesh for visualization.
- `Element_Stiffness_rank.m`: This script calculates the rank of stiffness matrix for 3-node and 6-node membrane elements under zero or finite deformations.
- `T3_assy_consistency_check.m`: This script performs consistency check for `assemblyT3Lin.m`
- `T3_assy_Stiffness_Rank.m`: This script calculates the stiffness rank for 3-node triangular element mesh under zero and finite deformation.
- `T3Lin.m`: This function calcuates the 3-node linear triangular shape functions and its derivatives.
- `T3MembraneEle.m`: This function calculates strain energy, nodal force vectors and stiffness modulus for a 3-node triangular membrane element.
- `T3MembraneEle_Verification.m`: This script checks `T3MembraneEle.m` for consistency.
- `T6_assy_consistency_check.m`: This script performs consistency check for `assemblyT6Quad.m`
- `T6_assy_Stiffness_Rank.m`: This script calculates the stiffness rank for 6-node triangular element mesh under zero and finite deformation.
- `T6Quad.m`: This function calcuates the 6-node linear triangular shape functions and its derivatives.
- `T6MembraneEle.m`: This function calculates strain energy, nodal force vectors and stiffness modulus for a 6-node triangular membrane element.
- `T6MembraneEle_Verification.m`: This script checks `T6MembraneEle.m` for consistency.
- `TransverseLoadExample.m`: This is the driver script for transverse load problem.
- `TriGaussQuad.m`: Returns the Gauss-quadrature points and weights for 1-point and 3-point schemes.