Nonlinear Finite Elements for Continua and Structures, Second Edition

Solution Manual

Miguel A. Bessa

Department of Mechanical Engineering

Northwestern University

Evanston, Illinois, USA

Khalil I. Elkhodary

Department of Mechanical Engineering

The American University in Cairo

Cairo, Egypt

Wing Kam Liu

Department of Mechanical Engineering

Northwestern University

Evanston, Illinois, USA

Ted Belytschko

Department of Mechanical Engineering

Northwestern University

Evanston, Illinois, USA

Brian Moran

Physical Sciences and Engineering Division

King Abdullah University of Science and Technology

Thuwal, Kingdom of Saudi Arabia

Nonlinear Finite Elements for Continua and Structures, , Second Edition.

Acknowledgements

Special thanks to Jifeng Zhao, Patrick Lea and Rajiv Malhotra for providing feedback and correcting important parts of this manual. We would also like to thank Brendan Abberton, Ying Li, John Moore, and Wylie Stroberg.

The first author would like to thank his wife for her enormous patience and support, and even for typing some of the equations of this manual!

CHAPTER 1: Introduction

1.1. Show that the diffusion equation (heat conduction is one example) $u_{,xx} = \alpha u_{,t}$, where α is a positive constant, is parabolic.

Solution to 1.1.

$$u_{,xx} = \alpha u_{,t}$$

The above equation can be reduced to a first order form following the same procedure presented in Section 1.5. We let $f = u_{,x}$ and $g = u_{,t}$ from which we obtain the two first-order equations:

$$f_{,x} = \alpha g$$

$$f_{,t} = g_{,x}$$

Expressing the derivatives of the dependent variables as

$$f_{,s} = f_{,x}x_{,s} + f_{,t}t_{,s}$$

$$g_{,s} = g_{,x}x_{,s} + g_{,t}t_{,s}$$

And writing the above system in matrix form,

$$\boldsymbol{Az} = \begin{bmatrix} \alpha^{-1} & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ x_{,s} & t_{,s} & 0 & 0 \\ 0 & 0 & x_{,s} & t_{,s} \end{bmatrix} \begin{Bmatrix} f_{,x} \\ f_{,t} \\ g_{,x} \\ g_{,t} \end{Bmatrix} = \begin{Bmatrix} g \\ 0 \\ f_{,s} \\ g_{,s} \end{Bmatrix}$$

$$det(\boldsymbol{A}) = \alpha^{-1}\left(t_{,s}\right)^2 \Rightarrow t_{,s} = 0$$

Dividing by $x_{,s}$

$$t_{,x} = 0$$

From which we conclude that the diffusion equation is parabolic.

1.2. Determine the classification of the equation for the dynamics of beams, $u_{,xxxx} = \alpha u_{,tt}$.

Solution to 1.2.

Considering the solution for exercise 1, by inspection, the equation for the dynamics of beams is also parabolic. Note that, if $f = u_{,xx}$ and $g = u_{,t}$ we get

$$f_{,xx} = \alpha g_{,t}$$

CHAPTER 2: Lagrangian and Eulerian finite elements in one dimension

2.1. Transform the principle of virtual work to the principle of virtual power by letting $\delta u = \delta v$ and using the conservation of mass and the transformations for the stresses. (Note that this is possible since the admissibility conditions on the two sets of test and trial function spaces are identical).

Solution to 2.1.

The Principle of virtual work is:

$$\int_{X_1}^{X_2} (\delta u)_{,X}\, P A_0\, dX - (\delta u A_0 n^0 P)|_{\Gamma_t} - \int_{X_1}^{X_2} \delta u\, \rho_0 b A_0 dX + \int_{X_1}^{X_2} \delta u\, \rho_0 A_0 \ddot{u} dX = 0$$

The transformation to the principle of virtual power is possible by letting $\delta u = \delta v$, using the conservation of mass, $\rho_0 A_0 dX = \rho A dx$, the transformation for the stresses, $PA_0 = \sigma$A, and using the chain rule $\frac{\partial}{\partial X} = \frac{\partial}{\partial x}\frac{\partial x}{\partial X}$

$$\int_{x_1}^{x_2} (\delta v)_{,x} \frac{\partial x}{\partial X}\, \sigma A dX - (\delta v\, A\, n\, \sigma)|_{\Gamma_t} - \int_{x_1}^{x_2} \delta v \rho b A\, dx + \int_{x_1}^{x_2} \delta u \rho A \dot{v}\, dx = 0$$

$$\int_{x_1}^{x_2} [(\delta v)_{,x}\, \sigma A - \delta v(\rho b A + \rho A \dot{v})] dx - (\delta v\, A\, n\, \sigma)|_{\Gamma_t} = 0$$

2.2. **Consider a tapered two-node element with a linear displacement field as in Example 2.1 where the cross-sectional area $A_0 = A_{01}(1 - \xi) + A_{02}\xi$, where A_{01} and A_{02} are the initial cross-sectional areas at nodes 1 and 2. Assume that the nominal stress P is also linear in the element, i.e. $P = P_1(1 - \xi) + P_2\xi$.**

(a) **Using the total Lagrangian formulation, develop expressions for the internal nodal forces. For a constant body force, develop the external nodal forces. Compare the internal and external nodal forces for the case when $A_{01} = A_{02} = A_0$ and $P_1 = P_2$ to the results in Example 2.1.**

(b) **Develop the consistent mass matrix. Then obtain a diagonal form of the mass matrix by the row-sum technique. Find the frequencies of a single element with consistent mass and the diagonal mass by solving the eigenvalue problem**

$$\mathbf{Ky} = \omega^2 \mathbf{My} \quad \text{where } \mathbf{K} = \frac{E^{PF}(A_{01} + A_{02})}{2\ell_0} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Solution to 2.2a).

Similarly to example 2.1, the displacement field is given by the linear Lagrange interpolant expressed in terms of the material coordinates:

$$u(X,t) = \frac{1}{l_0} [X_2 - X \quad X - X_1] \begin{Bmatrix} u_1(t) \\ u_2(t) \end{Bmatrix}$$

where $\boldsymbol{N} = \frac{1}{l_0} [X_2 - X \quad X - X_1]$ and $l_0 = X_2 - X_1$.

The strain measure is evaluated in terms of the nodal displacements by using $\varepsilon = \sum_I \frac{\partial N_I}{\partial X} u_I^e = \boldsymbol{B_0}\, \boldsymbol{u^e}$

$$\varepsilon(X,t) = u_{,X} = \sum_I \frac{\partial N_I}{\partial X} u_I^e = \frac{1}{l_0} [-1 \quad 1] \begin{Bmatrix} u_1(t) \\ u_2(t) \end{Bmatrix}$$

where $\boldsymbol{B}_0 = \frac{1}{l_0}[-1 \quad 1]$.

However, the displacement field can also be expressed in parent coordinates

$$u(\xi,t) = [1 - \xi \quad \xi] \begin{Bmatrix} u_1(t) \\ u_2(t) \end{Bmatrix}$$

where $\boldsymbol{N}(\xi) = [1-\xi \quad \xi]$, with $\xi = \frac{X-X_1}{l_0} \Rightarrow X_{,\xi} = l_0$.

Using parent coordinates, the displacement field becomes

$$\varepsilon(\xi,t) = X_{,\xi}^{-1}\,\boldsymbol{N}_{,\xi}(\xi)\boldsymbol{u}^e(t) = \frac{1}{l_0}\,[-1 \quad 1]\begin{Bmatrix}u_1(t)\\u_2(t)\end{Bmatrix}$$

where $\boldsymbol{B}_0 = X_{,\xi}^{-1}\,\boldsymbol{N}_{,\xi}(\xi) = \frac{1}{l_0}[-1 \quad 1]$

We can now obtain the internal nodal forces:

$$\boldsymbol{f}_e^{int} = \int_{\Omega_0^e} \boldsymbol{B}_o^T P\; d\,\Omega_0 = \int_0^1 \boldsymbol{B}_0^T(\xi) P\; A_0\, X_{,\xi} d\xi$$

$$= \int_0^1 \frac{1}{l_0}\begin{bmatrix}-1\\1\end{bmatrix}\,(P_1(1-\xi) + P_2\xi)(A_{01}(1-\xi) + A_{02}\xi) l_0 d\xi$$

$$\boldsymbol{f}_e^{int} = \frac{A_{01}(2P_1+P_2) + A_{02}(P_1+2P_2)}{6}\begin{Bmatrix}-1\\1\end{Bmatrix}$$

The external nodal forces are:

$$\boldsymbol{f}_e^{ext} = \int_{\Omega_0^e} \rho_0 \boldsymbol{N}^T b\; A_0 dX = \int_0^1 \rho_0\, \boldsymbol{N}^T\,(\xi)\; bA_0\; X_{,\xi} d\xi$$

$$= \int_0^1 \rho_0 \begin{bmatrix}1-\xi\\ \xi\end{bmatrix} b(A_{01}(1-\xi) + A_{02}\xi) l d\xi$$

$$\boldsymbol{f}_e^{ext} = \frac{\rho_0 b\; l_0}{6}\begin{Bmatrix}2A_{01} + A_{02}\\ A_{01} + 2A_{02}\end{Bmatrix}$$

Finally, we can compare these results with the ones obtained in example 2.1. The internal and external nodal forces for the case when $A_{01} = A_{02} = A_0$ and $P_1 = P_2 = P$ become:

$$\boldsymbol{f}_e^{int} = A_0 P\begin{bmatrix}-1\\1\end{bmatrix}$$

$$\boldsymbol{f}_e^{ext} = \frac{\rho_0 \, A_0 l_0}{2} \begin{bmatrix} b \\ b \end{bmatrix}$$

Which are the same results obtained in example 2.1 (provided that $b_1 = \ b_2 = b$).

Solution to 2.2b).

Consistent mass matrix:

$$\boldsymbol{M}_{\boldsymbol{e}}^{C} = \int_{X_1}^{X_2} \rho_0 \boldsymbol{N}^T \boldsymbol{N} A_0 dX = \int_0^1 \rho_0 \, \boldsymbol{N}^T(\xi) \boldsymbol{N}(\xi) \, A_0(\xi) X_{,\xi} d\xi$$

$$\boldsymbol{M}_{\boldsymbol{e}}^{C} = \frac{\rho_0 l}{12} \begin{bmatrix} 3A_{01} + A_{02} & A_{01} + A_{02} \\ A_{01} + A_{02} & A_{01} + 3A_{02} \end{bmatrix}$$

Diagonal Mass matrix ($M_{II} = \sum M_{IJ}$)

$$\boldsymbol{M}_{\boldsymbol{e}}^{D} = \frac{\rho_0 l}{6} \begin{bmatrix} 2A_{01} + A_{02} & 0 \\ 0 & A_{01} + 2A_{02} \end{bmatrix}$$

In order to find the natural frequencies, we need to solve the eigenvalue problem using the stiffness matrix provided in the problem statement:

$$\mathbf{K}\,\mathbf{y} \ = \ \omega^2 \boldsymbol{M}\boldsymbol{y} \Rightarrow (\boldsymbol{k} - \ \omega^2 \boldsymbol{M})\boldsymbol{y} = 0$$

Hence,

$$\det(\mathbf{k} - \omega^2 \boldsymbol{M}) = 0$$

Solving for the consistent mass matrix $\boldsymbol{M}_{\boldsymbol{e}}^{C}$, we find (besides the trivial $\omega_C = 0$):

$$\omega_C = \frac{2\sqrt{6}(A_{01} + A_{02})}{l_0(A_{01} + 2A_{02})}\sqrt{\frac{E^{PF}}{\rho_0}}$$

Solving for the diagonal mass matrix $\boldsymbol{M}_{\boldsymbol{e}}^{D}$, we find (besides the trivial $\omega_D = 0$):

$$\omega_D = \frac{3(A_{01} + A_{02})}{l_0\sqrt{(A_{01} + 2A_{02})(2A_{01} + A_{02})}}\sqrt{\frac{E^{PF}}{\rho_0}}$$

Note that $\omega_C > \omega_D$.

2.3. Consider a tapered two-node element with a linear displacement field in the updated Lagrangian formulation as in Example 2.4. Let the current cross-sectional area be given by $A = A_1(1 - \xi) + A_2\xi$, **where** A_1 **and** A_2 **are the current cross-sectional areas at nodes 1 and 2. Develop the internal nodal forces in terms of the Cauchy stress for the updated Lagrangian formulation assuming** $\sigma = \sigma_1(1 - \xi) + \sigma_2\xi$ **where** σ_1 **and** σ_2 **are the Cauchy stresses at the two nodes. Develop the nodal external forces for a constant body force.**

Solution to 2.3.

Internal nodal forces:

$$\boldsymbol{f}^{int} = \int_{x_1}^{x_2} \frac{\partial \boldsymbol{N}^T}{\partial x}\sigma\, A dx = \int_0^1 \frac{\partial \boldsymbol{N}^T}{\partial \xi}\left(x_{,\xi}\right)^{-1}\sigma\, A(\xi)x_{,\xi}d\xi$$

$$\boldsymbol{N} = [1 - \xi \quad \xi]$$

$$x = \boldsymbol{N}^T\begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = (1 - \xi)x_1 + \xi x_2$$

$$x_{,\xi} = -x_1 + x_2 = l$$

$$\boldsymbol{f}^{int} = \int_0^1 \begin{bmatrix} -1 \\ 1 \end{bmatrix} \frac{1}{l} \left(\sigma_1(1-\xi) + \sigma_2\xi\right)\left(A_1(1-\xi) + A_2\xi\right) l \, d\xi$$

$$\boldsymbol{f}^{int} = \left[\frac{2A_1}{3}(4\sigma_1 - \sigma_2) + \frac{2A_2}{3}(\sigma_2 - \sigma_1)\right] \begin{Bmatrix} -1 \\ 1 \end{Bmatrix}$$

External nodal forces:

$$\boldsymbol{f}^{ext} = \int_{x_1}^{x_2} \rho \boldsymbol{N}^T b \, A dx = \int_0^1 \rho \begin{bmatrix} 1-\xi \\ \xi \end{bmatrix} b\left(A_1(1-\xi) + A_2\xi\right) l \, d\xi$$

$$\boldsymbol{f}^{ext} = \frac{\rho b l}{3} \begin{Bmatrix} A_1 + \dfrac{A_2}{2} \\ \dfrac{A_1}{2} + A_2 \end{Bmatrix}$$

2.4. Consider a 2-element mesh consisting of elements of length ℓ with constant cross-sectional area A. Assemble a consistent mass matrix and a stiffness matrix and obtain the frequency for the two element mesh with all nodes free (the eigenvalue problem is 3 × 3). The frequency analysis assumes a linear response so the initial and current geometry are identical. Repeat the same problem with a lumped mass. Compare the frequencies for the lumped and consistent mass matrices to the exact frequency for a free-free rod, $\omega = n\frac{\pi c}{L}$, where n = 0, 1, … Observe that the consistent mass frequencies are above the exact, whereas the diagonal mass frequencies are below the exact.

Solution to 2.4.

Calculating the consistent mass matrix for each element:

$$\boldsymbol{M}_{(1)}^{C} = \boldsymbol{M}_{(2)}^{C} = \boldsymbol{M}_{e}^{C} = \int_{x_1}^{x_2} \rho \boldsymbol{N}^{\boldsymbol{T}} \boldsymbol{N}\, A\, dx = \int_{0}^{1} \rho A \boldsymbol{N}^{\boldsymbol{T}} \boldsymbol{N}\, x_{,\xi}\, dA$$

For a two-node element with length l:

$$\boldsymbol{N} = [1-\xi \quad \xi]$$
$$x = [1-\xi \quad \xi] \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = (1-\xi)x_1 + \xi x_2$$
$$x_{,\xi} = -x_1 + x_2 = l$$

Then,

$$\boldsymbol{M}_{e}^{C} = \int_{x_1}^{x_2} \rho\, A \begin{bmatrix} 1-\xi \\ \xi \end{bmatrix} [1-\xi \quad \xi]\, l\, d\xi = \frac{\rho A l}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Assemble the mass matrix for the two element mesh according to the connectivity matrices:

$$\boldsymbol{M}^{C} = \boldsymbol{L}_{(1)} \boldsymbol{M}_{(1)}^{C} + \boldsymbol{L}_{(2)} \boldsymbol{M}_{(2)}^{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^{T} \frac{\rho A l}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{T} \frac{\rho A l}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$
$$\boldsymbol{M}^{C} = \frac{\rho A l}{6} \begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

The stiffness matrix for each element (considering linear elastic isotropic material):

$$\boldsymbol{K}_{(1)} = \boldsymbol{K}_{(2)} = \boldsymbol{K}_{e} = \int_{x_1}^{x_2} \frac{\partial \boldsymbol{N}^{T}}{\partial x} \frac{\partial \boldsymbol{N}}{\partial x} EA\, dx = \frac{E\, A}{l} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Assembling the stiffness matrix for the two element mesh:

$$\boldsymbol{K} = \boldsymbol{L}_{(1)} \boldsymbol{K}_{(1)} + \boldsymbol{L}_{(2)} \boldsymbol{K}_{(2)} = \frac{EA}{l} \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

We can now determine the natural frequencies solving the eigenvalue problem as in exercise 2:

$$\boldsymbol{K}\,\boldsymbol{y} = \omega^2\,\boldsymbol{M}\,\boldsymbol{y} \;\Rightarrow (\boldsymbol{k} - \omega^2\,\boldsymbol{M})\boldsymbol{y} = 0$$

Using the consistent mass matrix:

$$det(\boldsymbol{k} - \omega^2\boldsymbol{M}^C) = 0 \quad \Rightarrow \quad \omega_1^2 = 0 \;\vee\; \omega_2^2 = \frac{3E}{l^2\rho} \;\vee\; \omega_2^2 = \frac{12E}{l^2\rho}$$

$$\omega_1 = 0 \;\vee\; \omega_2 = \frac{\sqrt{3}}{l}\sqrt{\frac{E}{\rho}} \cong 0.551\,\frac{\pi c}{l} \;\vee\; \omega_3 = \frac{2\sqrt{3}}{l}\sqrt{\frac{E}{\rho}} \cong 1.103\,\frac{\pi c}{l}$$

Using the lumped mass matrix:

$$\boldsymbol{M}^D = \frac{\rho A l}{2}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\det(\boldsymbol{k} - \omega^2\boldsymbol{M}^D) = 0 \quad \Rightarrow \quad \omega_1^2 = 0 \vee \omega_2^2 = \frac{2E}{l^2\rho} \vee \omega_3^2 = \frac{4E}{l^2\rho}$$

$$\omega_1 = 0 \;\vee\; \omega_2 = \frac{\sqrt{2}}{l}\sqrt{\frac{E}{\rho}} \cong 0.450\,\frac{\pi c}{l} \;\vee\; \omega_3 = \frac{2}{l}\sqrt{\frac{E}{\rho}} \cong 0.637\,\frac{\pi c}{l}$$

The exact solution for the first 3 natural frequencies is ($L = 2l$):

$$\omega_1 = 0 \;\vee \omega_2 = 0.5\,\frac{\pi c}{l} \vee\; \omega_3 = \frac{\pi c}{l}$$

So we see that the frequencies obtained with the consistent mass matrix are above the exact solution and the frequencies for the lumped mass matrix are below.

2.5. Repeat Example 2.6 for spherical symmetry, where

$$\mathbf{D}=\begin{Bmatrix} D_{rr} \\ D_{\theta\theta} \\ D_{\phi\phi} \end{Bmatrix}, \boldsymbol{\sigma}=\begin{Bmatrix} \sigma_{rr} \\ \sigma_{\theta\theta} \\ \sigma_{\phi\phi} \end{Bmatrix}, D_{rr}=v_{r,r}, D_{\theta\theta}=D_{\phi\phi}=\frac{1}{r}vr$$

Solution to 2.5.

For spherical symmetry,

$$\boldsymbol{D}=\begin{Bmatrix} D_{rr} \\ D_{\theta\theta} \\ D_{\phi\phi} \end{Bmatrix}=\begin{Bmatrix} v_{r,r} \\ \frac{1}{r}v_r \\ \frac{1}{r}v_r \end{Bmatrix}$$

Momentum equation in spherical coordinates, for this problem:

$$\frac{\partial\sigma_{rr}}{\partial r}+\frac{1}{r}\left(2\sigma_{rr}-\sigma_{\theta\theta}-\sigma_{\phi\phi}\right)+\rho b_r=\rho\,\dot{v}_r$$

$$\delta P^{int}=\int_0^{2\pi}\int_0^{\pi}\int_{r_1^e}^{r_2^e}\left(\delta D_r\sigma_r+\delta D_{\theta\theta}+\delta\,D_{\emptyset}\sigma_{\emptyset}\right)r^2\sin\phi\;\;drd\phi d\theta$$

$$\delta P^{int}=\int_{r_1^e}^{r_2^e}\delta\boldsymbol{D}^T\,\boldsymbol{\sigma}\,4\pi r^2dr$$

Considering a linear velocity field:

$$v(\xi,t)=\begin{bmatrix}1-\xi & \xi\end{bmatrix}\begin{Bmatrix}v_1(t)\\ v_2(t)\end{Bmatrix}$$

with $\boldsymbol{N} = [1-\xi \quad \xi]$, the components of the rate-of-deformation can be determined as follows:

$$D_r = v_{r,r} = v_{r,\xi}\left(r_{,\xi}\right)^{-1}$$

$$v_{r,\xi} = [-1 \quad 1]\begin{Bmatrix} v_1(t) \\ v_2(t) \end{Bmatrix}$$

$$r = \boldsymbol{N}\begin{Bmatrix} r_1 \\ r_2 \end{Bmatrix} = (1-\xi)r_1 + \xi r_2$$

$$r_{,\xi} = r_2 - r_1$$

Thus,

$$D_r = \frac{1}{r_{21}}[-1 \quad 1]\begin{Bmatrix} v_1(t) \\ v_2(t) \end{Bmatrix}$$

with $r_{21} = r_2 - r_1$. The remaining components of the rate-of-deformation are:

$$D_\phi = D_\theta = \frac{v_r}{r} = \frac{1}{r}[1-\xi \quad \xi]\begin{Bmatrix} v_1(t) \\ v_2(t) \end{Bmatrix}$$

From which we obtain the rate-of-deformation:

$$\boldsymbol{D} = \begin{bmatrix} \frac{-1}{r_{21}} & \frac{1}{r_{21}} \\ \frac{1-\xi}{r} & \frac{\xi}{r} \\ \frac{1-\xi}{r} & \frac{\xi}{r} \end{bmatrix}\begin{Bmatrix} v_1(t) \\ v_2(t) \end{Bmatrix} = \boldsymbol{B}\,\boldsymbol{v_e}$$

with $r = (1-\xi)r_1 + \xi r_2$, and $r_{21} = r_2 - r_1$.

The internal forces can now be calculated:

$$\boldsymbol{f}^{int} = \int_{r_1}^{r_2} \boldsymbol{B}^T\{\boldsymbol{\sigma}\}\, 4\pi r^2 dr$$

$$\boldsymbol{f}^{int} = \int_0^1 \begin{bmatrix} \dfrac{-1}{r_{21}} & \dfrac{1-\xi}{(1-\xi)r_1+\xi r_2} & \dfrac{1-\xi}{(1-\xi)r_1+\xi r_2} \\ \dfrac{1}{r_{21}} & \dfrac{\xi}{(1-\xi)r_1+\xi r_2} & \dfrac{\xi}{(1-\xi)r_1+\xi r_2} \end{bmatrix} \begin{Bmatrix} \sigma_{rr} \\ \sigma_{\theta\theta} \\ \sigma_{\phi\phi} \end{Bmatrix} 4\pi\big((1-\xi)r_1+\xi r_2\big)^2 r_{21} dr$$

The consistent mass matrix can be calculated as:

$$\boldsymbol{M}_e = \int_0^1 \rho \begin{bmatrix} 1-\xi \\ \xi \end{bmatrix} \begin{bmatrix} 1-\xi & \xi \end{bmatrix} 4\pi\big((1-\xi)r_1+\xi r_2\big)^2 r_{21} dr$$

$$\boldsymbol{M}_e = \frac{\rho\pi}{15} \begin{bmatrix} 2r_{21}(6r_1^2+3r_1r_2+r_2^2) & r_{21}(3r_1^2+4r_1r_2+3r_2^2) \\ r_{21}(3r_1^1+4r_1r_2+3r_2^2) & 2r_{21}(r_1^2+3r_1r_2+6r_2^2) \end{bmatrix}$$

2.6. (a) Develop an expression for the principle of virtual power and derive the corresponding strong form.

(b) For a two node element with a linear velocity field, develop B, the internal nodal forces $\mathbf{f}_e^{\text{int}}$ in terms of the stresses, and the consistent mass matrix $\mathbf{M}_e$. For constant body force, develop an expression for the nodal external forces $\mathbf{f}_e^{\text{ext}}$.

Solution to 2.6a).

See sections 2.6 & 2.7.

Solution to 2.6b).

For a two node element with a linear velocity field, the shape functions are:

$$\boldsymbol{N} = \begin{bmatrix} 1-\xi & \xi \end{bmatrix}$$

From which we can write:

$$x = \begin{bmatrix} 1-\xi & \xi \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = (1-\xi)x_1 + \xi x_2$$

$$x_{,\xi} = -x_1 + x_2 = l$$

$$\boldsymbol{B} = \frac{\partial \boldsymbol{N}}{\partial x} = \frac{\partial \boldsymbol{N}}{\partial \xi}\frac{\partial \xi}{\partial x} = \begin{bmatrix} -1 & 1 \end{bmatrix}\frac{1}{l}$$

Therefore, the internal nodal forces are:

$$\boldsymbol{f}_e^{int} = \int_{x_1}^{x_2} \boldsymbol{B}^T \boldsymbol{\sigma}\ A dx = \int_0^1 \frac{1}{l}\begin{bmatrix} -1 \\ 1 \end{bmatrix} \sigma\ Al\ d\xi = \sigma A \begin{Bmatrix} -1 \\ 1 \end{Bmatrix}$$

The consistent mass matrix is:

$$\boldsymbol{M}_e = \int_0^1 \rho \begin{bmatrix} 1-\xi \\ \xi \end{bmatrix} \begin{bmatrix} 1-\xi & \xi \end{bmatrix} Al\ d\xi = \frac{\rho Al}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

And the external nodal forces are:

$$\boldsymbol{f}^{ext} = \int_0^1 \begin{bmatrix} 1-\xi \\ \xi \end{bmatrix} \rho bAl d\xi = \frac{\rho bAl}{2}\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

CHAPTER 3: Continuum mechanics

3.1. Consider the element shown in Figure 3.4. Let the motion be given by

$$x = X + Yt, \quad y = Y + \frac{1}{2}Xt$$

(a) Sketch the element at time *t* = 1. Evaluate the deformation gradient and the Green strain tensor at this time.

(b) Evaluate the velocity and acceleration of the element at *t* = 1.

(c) Evaluate the rate-of-deformation and the spin tensor of the element at *t* = 1.

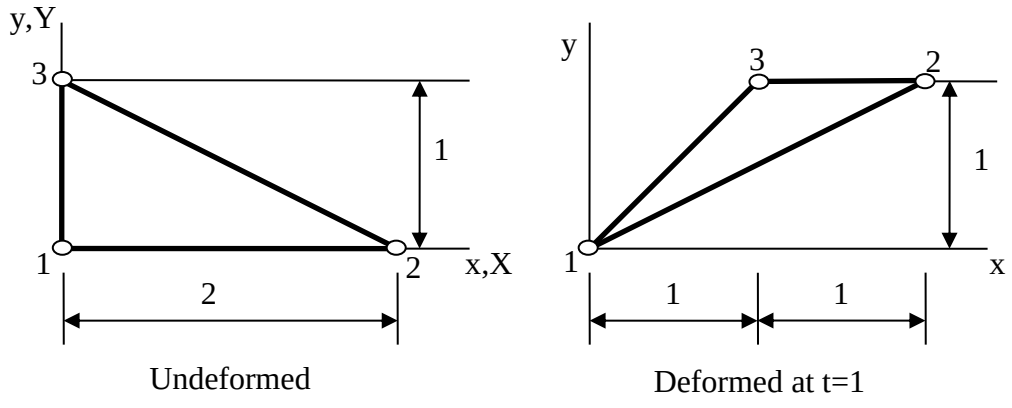
(d) Repeat the above at *t* = 0.5.

(e) Evaluate the Jacobian determinant as a function of time and determine for how long it remains positive. Sketch the element at the time that the Jacobian changes sign. What can you say about the motion at that time?

Solution to 3.1a).

For $t = 1$ the nodal coordinates are:

$$x_1 = 0 \quad ; \quad y_1 = 0$$
$$x_2 = 2 \quad ; \quad y_2 = 1$$
$$x_3 = 1 \quad ; \quad y_3 = 1$$



Sketches of the element in the undeformed configuration and the deformed configuration at t=1.

Deformation gradient:

$$\mathbf{F} = \begin{bmatrix} \frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} \\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} \end{bmatrix} = \begin{bmatrix} 1 & t \\ \frac{t}{2} & 1 \end{bmatrix}$$

$$\mathbf{F}|_{t=1} = \begin{bmatrix} 1 & 1 \\ 1/2 & 1 \end{bmatrix}$$

Green-Lagrange strain tensor:

$$\boldsymbol{E} = \frac{1}{2}\left(\boldsymbol{F}^T.\boldsymbol{F} - \boldsymbol{I}\right) = \begin{bmatrix} \frac{t^2}{8} & \frac{3t}{4} \\ \frac{3t}{4} & \frac{t^2}{2} \end{bmatrix}$$

$$\boldsymbol{E}\,|_{t=1} = \begin{bmatrix} 1/8 & 3/4 \\ 3/4 & 1/2 \end{bmatrix}$$

Solution to 3.1b).

$$\begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{bmatrix} 1 & t \\ \frac{t}{2} & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix}$$

The velocity is obtained by taking the derivative of this motion with respect to time,

$$\begin{Bmatrix} v_x \\ v_y \end{Bmatrix} = \begin{Bmatrix} \dot{x} \\ \dot{y} \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{1}{2} & 0 \end{bmatrix} \begin{Bmatrix} X \\ Y \end{Bmatrix}$$

The acceleration in the material description is obtained by taking the time derivatives of the velocities:

$$\begin{Bmatrix} a_x \\ a_y \end{Bmatrix} = \begin{Bmatrix} \dot{v}_x \\ \dot{v}_y \end{Bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} X \\ Y \end{Bmatrix}$$

Since it is asked for the velocity and acceleration of the element at t=1, we should substitute that value in t, but looking to the obtained velocities and accelerations we see that they are constant in time so:

$$\boldsymbol{v}|_{t=1} = \begin{bmatrix} 0 & 1 \\ 1/2 & 0 \end{bmatrix} \begin{Bmatrix} X \\ Y \end{Bmatrix} = \begin{Bmatrix} Y \\ X/2 \end{Bmatrix}$$

$$\boldsymbol{a}|_{t=1} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} X \\ Y \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

Solution to 3.1c).

Rate-of-deformation:

$$\boldsymbol{D} = \frac{1}{2}(\boldsymbol{L} + \boldsymbol{L}^T)$$

The velocity gradient can be calculated from:

$$\boldsymbol{L} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{X}} \frac{\partial \boldsymbol{X}}{\partial \boldsymbol{x}} = \dot{\boldsymbol{F}} \cdot \boldsymbol{F}^{-1}$$

$$\dot{\boldsymbol{F}} = \begin{bmatrix} 0 & 1 \\ 1/2 & 0 \end{bmatrix}$$

$$\boldsymbol{F}^{-1} = \frac{1}{1 - t^2/2} \begin{bmatrix} 1 & -t/2 \\ -t & 1 \end{bmatrix}$$

Therefore,

$$\boldsymbol{L} = \dot{\boldsymbol{F}} \cdot \boldsymbol{F}^{-1} = \begin{bmatrix} \frac{t}{t^2-2} & \frac{-2}{t^2-2} \\ \frac{-1}{t^2-2} & \frac{t}{t^2-2} \end{bmatrix}$$

$$\boldsymbol{D} = \begin{bmatrix} \frac{t}{t^2-2} & \frac{-\frac{3}{2}}{t^2-2} \\ \frac{-\frac{3}{2}}{t^2-2} & \frac{t}{t^2-2} \end{bmatrix}$$

$$\boldsymbol{D}\,|_{t=1} = \begin{bmatrix} -1 & 3/2 \\ 3/2 & -1 \end{bmatrix}$$

The spin tensor is simple to obtain:

$$\boldsymbol{W} = \frac{1}{2}(\boldsymbol{L} - \boldsymbol{L}^T) = \begin{bmatrix} 0 & \dfrac{-0.5}{t^2-2} \\ \dfrac{0.5}{t^2-2} & 0 \end{bmatrix}$$

$$\boldsymbol{W}_{|t=1} = \begin{bmatrix} 0 & \frac{1}{2} \\ -\frac{1}{2} & 0 \end{bmatrix}$$

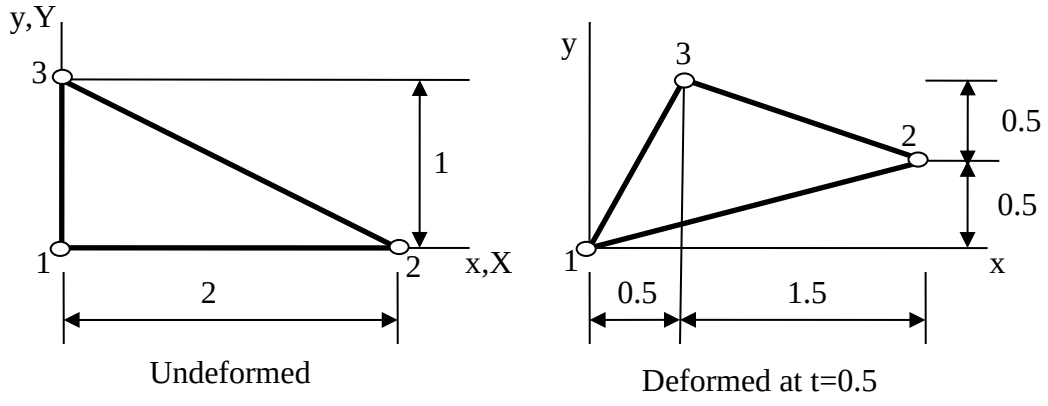
Solution to 3.1d).

Repeating the exercise, now considering t=0.5:

$$x_1 = 0 + 0 = 0 \qquad y_1 = 0 + 0 = 0$$

$$x_2 = 2 + 0 = 2 \qquad y_2 = 0 + \frac{1}{2} \times 2 \times 0.5 = 0.5$$

$$x_3 = 0 + 1 \times 0.5 = 0.5 \qquad y_3 = 1 + 0 = 1$$



Sketches of the element in the undeformed configuration and the deformed configuration at t=0.5.

Deformation gradient and Green strain tensor at t=0.5:

$$\boldsymbol{F}|_{t=0.5} = \begin{bmatrix} 1 & 1/2 \\ 1/4 & 1 \end{bmatrix}$$

$$\boldsymbol{E}|_{t=0.5} = \begin{bmatrix} 1/32 & 3/8 \\ 3/8 & 1/8 \end{bmatrix}$$

Velocity and acceleration at t=0.5:

$$\boldsymbol{v}|_{t=0.5} = \boldsymbol{v} = \begin{Bmatrix} v_x \\ v_y \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ 1/2 & 0 \end{bmatrix} \begin{Bmatrix} X \\ Y \end{Bmatrix} = \begin{Bmatrix} Y \\ X/2 \end{Bmatrix}$$

$$\boldsymbol{a}|_{t=0.5} = \boldsymbol{a} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} X \\ Y \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

Rate of deformation:

$$\boldsymbol{D}|_{t=0.5} = \boldsymbol{v} = \begin{bmatrix} -2/7 & 6/7 \\ 6/7 & -2/7 \end{bmatrix}$$

Spin tensor:

$$\boldsymbol{W}|_{t=0.5} = \boldsymbol{v} = \begin{bmatrix} 0 & 2/7 \\ -2/7 & 0 \end{bmatrix}$$

Solution to 3.1e).

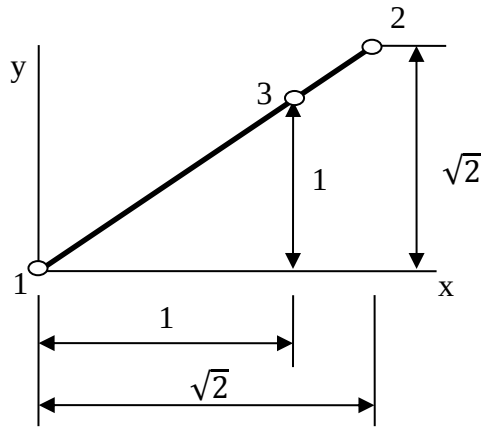
The Jacobian is obtained as:

$$J = \det(\boldsymbol{F}) = \begin{vmatrix} 1 & t \\ t/2 & 1 \end{vmatrix} = 1 - \frac{t^2}{2}, \quad t \geq 0$$

$$J > 0 \Rightarrow 1 - \frac{t^2}{2} > 0 \Rightarrow \left(t < \sqrt{2} \ \wedge \ t > \sqrt{2}\right) \ \wedge \ t \geq 0 \quad \Rightarrow \quad t \in \left[0, \sqrt{2}\right[$$

The Jacobian changes sign at $t = \sqrt{2}$, and the motion at that time is:

$$x_1 = 0 \qquad y_1 = 0$$
$$x_2 = 2 \qquad y_2 = \sqrt{2}$$
$$x_3 = \sqrt{2} \qquad y_3 = 1$$



Sketch of the element at the time the Jacobian changes sign.

We conclude that the element collapses in itself leading to a zero element area which causes a singularity because the deformation gradient cannot be inverted anymore.

3.2. Consider the motion given in Example 3.13, (E3.13.1). Find the velocity gradient L, the rate-of-deformation D, the spin tensor W and the angular velocity Ω as functions of time. Plot the spin and the angular velocity as function of time on the interval $t \in [0,4]$**. Does this shed any light on the difference between the Green-Naghdi and Jaumann material shown in Fig. 3.13?**

Solution to 3.2.

Motion of the element given in Example 3.13:

$$x = X + tY$$

$$y = Y$$

The deformation gradient, its time derivative and its inverse are:

$$\boldsymbol{F} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix};\ \dot{\boldsymbol{F}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix};\ \boldsymbol{F}^{-1} = \begin{bmatrix} 1 & -t \\ 0 & 1 \end{bmatrix}$$

The velocity gradient, the rate of deformation and the spin tensor are:

$$\boldsymbol{L} = \dot{\boldsymbol{F}} \cdot \boldsymbol{F}^{-1} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\boldsymbol{D} = \frac{1}{2}\left(\boldsymbol{L} + \boldsymbol{L}^{T}\right) = \begin{bmatrix} 0 & 1/2 \\ 1/2 & 0 \end{bmatrix}$$

$$\boldsymbol{W} = \frac{1}{2}\left(\boldsymbol{L} - \boldsymbol{L}^{T}\right) = \begin{bmatrix} 0 & 1/2 \\ -1/2 & 0 \end{bmatrix}$$

The angular velocity $\boldsymbol{\Omega}$ is calculated by:

$$\Omega = \dot{\boldsymbol{R}} \cdot \boldsymbol{R}^{T}$$

where $\boldsymbol{R} = \boldsymbol{F} \cdot \boldsymbol{U}^{-1}$ and $\boldsymbol{U} = \left(\boldsymbol{F}^{T} \cdot \boldsymbol{F}\right)^{\frac{1}{2}}$

Therefore,

$$\boldsymbol{C} = \boldsymbol{F}^{T} \cdot \boldsymbol{F} = \begin{bmatrix} 1 & t \\ t & t^2 + 1 \end{bmatrix}$$

The eigenvalues of $\boldsymbol{C}$ are obtained from:

$$\det(\boldsymbol{C} - \mu^2 \boldsymbol{I}) = 0 \Rightarrow$$

$$\Rightarrow \mu_{\mathrm{I}} = \sqrt{-\frac{1}{2}\left(t\sqrt{t^2+4} - t^2 - 2\right)} \quad \vee \mu_{\mathrm{II}} = \sqrt{\frac{1}{2}\left(t\sqrt{t^2+4} + t^2 + 2\right)}$$

The eigenvectors of $\boldsymbol{C}$ are obtained from:

$$\left(\boldsymbol{C} - \mu_N^2\, \boldsymbol{I}\right) \cdot \boldsymbol{u}^{N} = 0 \quad , \quad N = I, II$$

First eigenvector $\boldsymbol{u}^{I}$:

$$(1 - \mu_I)u_1^I + tu_2^I = 0 \Rightarrow u_2^I = -\frac{1}{2}\left(\sqrt{t^2+4} - t\right)u_1^I$$

Normalizing the vector to be a unit vector:

$$(u_1^I)^2 + \left[-\frac{1}{2}\left(\sqrt{t^2+4}-t\right)u_1^I\right]^2 = 1$$

$$u_1^I = \frac{\sqrt{2}}{(t^2+4)^{\frac{1}{4}}\cdot\sqrt{\sqrt{t^2+4}-t}} \quad \vee \quad u_2^I = \frac{-\frac{\sqrt{2}}{2}\left(\sqrt{t^2+4}-t\right)}{(t^2+4)^{1/4}\cdot\sqrt{\sqrt{t^2+4}-t}}$$

Second eigenvector $\boldsymbol{u}^{II}$:

$$(1-\mu_{II})u_1^{II} + tu_2^{II} = 0 \Rightarrow u_2^{II} = \frac{1}{2}\left(\sqrt{t^2+4}-t\right)u_1^{II}$$

Normalizing the vector to be a unit vector:

$$(u_1^{II})^2 + \left[\frac{1}{2}\left(\sqrt{t^2+4}-t\right)u_1^{II}\right]^2 = 1 \Rightarrow$$

$$\Rightarrow u_1^{II} = \frac{\sqrt{2}}{(t^2+4)^{\frac{1}{4}}\cdot\sqrt{\sqrt{t^2+4}+t}} \quad \vee \quad u_2^{II} = \frac{\frac{\sqrt{2}}{2}\left(\sqrt{t^2+4}+t\right)}{(t^2+4)^{\frac{1}{4}}\cdot\sqrt{\sqrt{t^2+4}+t}}$$

We can finally determine the stretch tensor $\boldsymbol{U}$ from:

$$\boldsymbol{U} = \begin{bmatrix} u_1^I & u_1^{II} \\ u_2^I & u_2^{II} \end{bmatrix} \cdot \begin{bmatrix} \mu_I & 0 \\ 0 & \mu_{II} \end{bmatrix} \cdot \begin{bmatrix} u_1^I & u_1^{II} \\ u_2^I & u_2^{II} \end{bmatrix}^T$$

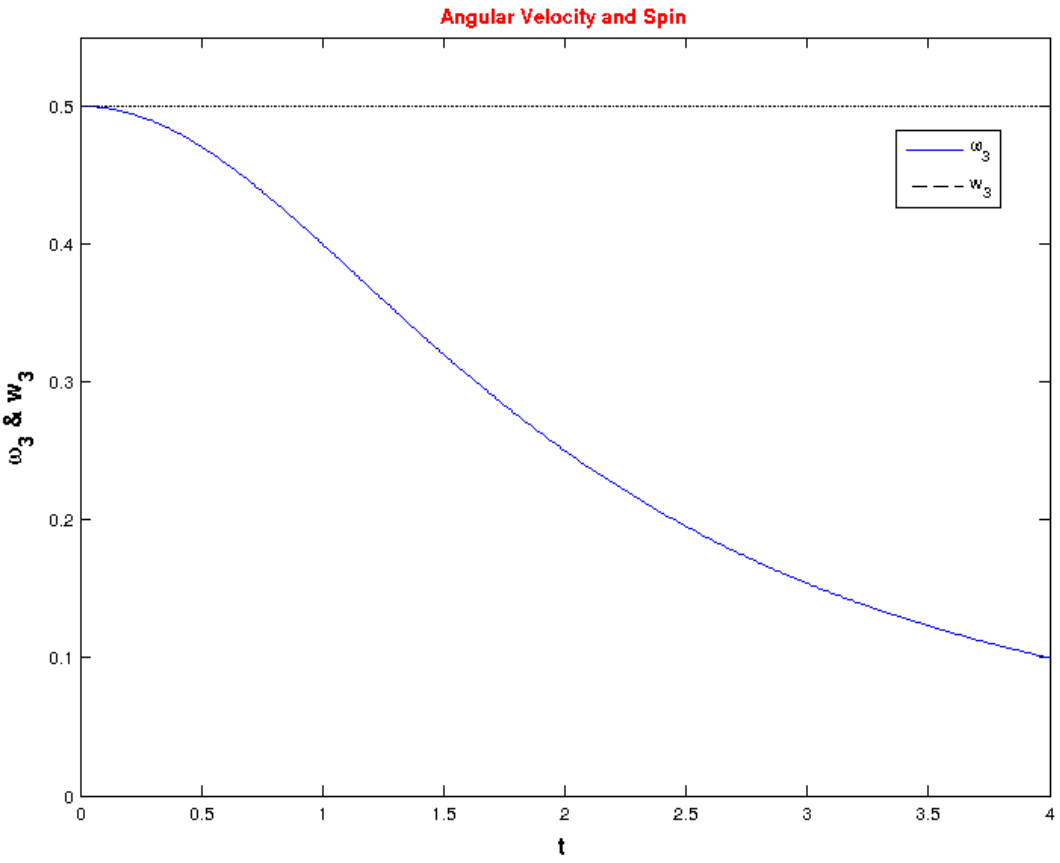
This results in very long components of the stretch tensor. The same happens for the computation of the rotation tensor, which requires to determine the inverse of **U** and to multiply that by the gradient tensor **F**:

$$\boldsymbol{R} = \boldsymbol{F}\cdot\boldsymbol{U}^{-1}$$

After computing the rotation tensor it is possible to obtain the angular velocity from:

$$\Omega = \dot{\boldsymbol{R}}\cdot\boldsymbol{R}^T$$

Due to the size of the components of each of the above referred tensors it is inadequate to write them here. Therefore, here we present the plot of the only nonzero component of the spin tensor W_{12} and the angular tensor Ω_{12} as a function of time on the interval $t \in [0,4]$.



Spin and angular velocity as function of time on the interval $\boldsymbol{t} \in [\mathbf{0}, \mathbf{4}]$.

Observing the plot it is clear that the spin tensor and the angular velocity tensor have very different values over time, although they start with the same value at t=0. The spin tensor, for this problem, is constant. On the other hand, the angular velocity is not. It decreases as time passes.

Therefore, looking to the expressions that define the Jaumann rate and the Green-Naghdi rate,

$$\boldsymbol{\sigma}^{\nabla J} = \frac{D\boldsymbol{\sigma}}{Dt} - \boldsymbol{W} \cdot \boldsymbol{\sigma} - \boldsymbol{\sigma} \cdot \boldsymbol{W}^{T}$$

$$\boldsymbol{\sigma}^{\nabla G} = \frac{D\boldsymbol{\sigma}}{Dt} - \Omega \cdot \boldsymbol{\sigma} - \boldsymbol{\sigma} \cdot \Omega^T$$

We see that the rates will have different values because they measure the rotation with different entities: the Jaumann rate uses the spin tensor, whereas the Green-Naghdi rate uses the angular velocity tensor. Hence, the latter will have a lower value when compared with the first.

3.3. Consider the three-node rod element shown in Figure 3.15. Use the standard 3-node shape functions for $\hat{v}_x$ and $\hat{v}_y$. The nodal coordinates are given by

$$x_1 = -r\sin\theta,\ x_2 = 0,\ x_3 = r\sin\theta \quad y_1 = 0,\ y_2 = r(1-\cos\theta),\ y_3 = 0$$

The nodal velocities at each node are in the radial direction as shown. Evaluate the corotational rate-of-deformation at node 2 in terms of the nodal velocities. For this point, the corotational coordinate system is coincident with the global system. Compare the result with the result obtained by using cylindrical coordinates, $D_{\theta\theta} = \frac{v_r}{r}$. Repeat the procedure at the Gauss quadrature point $\xi = -3^{-1/2}$ for $\theta = 0.1$ rad and $\theta = 0.05$ rad and compare to $D_{\theta\theta} = \frac{v_r}{r}$; the corotational system for the quadrature point is shown on the RHS of Figure 3.15.

Solution to 3.3.

Evaluating the figures it is possible to write the nodal coordinates with respect to a corotational system based on any given point with a parent coordinate ξ:

$$\hat{x}_I = r(\sin\emptyset_I - \sin\emptyset)\cos\emptyset + r(cos\emptyset_I - \cos\emptyset)\sin\emptyset$$
$$\hat{y}_I = -r(\sin\emptyset_I - \sin\emptyset)\sin\emptyset + r(cos\emptyset_I - \cos\emptyset)\cos\emptyset$$

where $\emptyset = \xi\theta$ and $\emptyset_I = \xi_I\,\theta$.

Using the above result we can write the current coordinates in the corotational system by using the shape functions to interpolate the nodal values:

$$\hat{x}_i = \hat{x}_{iI}\, N_I$$

Also by evaluating the figure, the nodal velocities in the corotational frame can be written as:

$$\hat{v}_{x_I} = v_r \sin(\emptyset_I - \theta)$$
$$\hat{v}_i = \hat{v}_{iI} N_I$$

If we want to determine the corotational rate-of-deformation at node 2, the corotational system is located at:

$$\xi = 0 \;\Rightarrow\; \emptyset = 0$$

For this point, the nodal coordinates have the following value:

$$\hat{x}_I = r \sin \emptyset_I \;\; ; \;\; \hat{y}_I = r\,(\cos \emptyset_I - 1)$$

So, each node has the coordinates:

$$Node\ 1 \rightarrow \emptyset_I = -\theta \;\Rightarrow\; \hat{x}_1 = -r \sin\theta \;\; ; \;\; \hat{y}_1 = r\,(\cos\theta - 1)$$
$$Node\ 2 \rightarrow \emptyset_I = 0 \;\Rightarrow\; \hat{x}_2 = 0 \;\; ; \;\; \hat{y}_2 = 0$$
$$Node\ 3 \rightarrow \emptyset_I = \theta \;\Rightarrow\; \hat{x}_3 = r \sin\theta \;\; ; \;\; \hat{y}_3 = r\,(\cos\theta - 1)$$

And the nodal velocities are:

$$Node\ 1 \rightarrow \hat{v}_{x_1} = -v_r \sin\theta \;\; ; \;\; \hat{v}_{y_1} = v_r \cos\theta$$
$$Node\ 2 \rightarrow \hat{v}_{x_2} = 0 \;\; ; \;\; \hat{v}_{y_2} = v_r$$
$$Node\ 3 \rightarrow \hat{v}_{x_3} = v_r \sin(\theta) \;\; ; \;\; \hat{v}_{y_3} = v_r \cos(\theta)$$

Now, using the shape functions for a 3 node rod element:

$$\boldsymbol{N}^T = \left[\frac{1}{2}\,\xi(\xi - 1) \quad 1 - \xi^2 \quad \frac{1}{2}\,\xi(\xi + 1)\right]$$

We can determine D_{xx} at node 2 in order to compare it with the result obtained using cylindrical coordinates $D_{\theta\theta}$.

$$D_{xx} = \frac{\partial \hat{v}_x}{\partial \hat{x}} = \hat{v}_{xI} \frac{\partial N_I}{\partial \hat{x}} = v_{xI} \frac{\partial N_I}{\partial \xi} \frac{\partial \xi}{\partial \hat{x}} = v_{xI} \frac{\partial N_I}{\partial \xi} \left(\frac{\partial \hat{x}}{\partial \xi}\right)^{-1}$$

$$\frac{\partial \hat{x}}{\partial \xi} = \hat{x}_I \frac{\partial N_I}{\partial \xi} = \{-r\sin\theta \quad 0 \quad r\sin\theta\} \begin{bmatrix} -\frac{1}{2} + \xi \\ -2\xi \\ \frac{1}{2} + \xi \end{bmatrix}_{\xi=0} = r\sin\theta$$

Since,

$$\left(\frac{\partial \hat{x}}{\partial \xi}\right)^{-1} = \frac{1}{r\sin\theta}$$

the corotational rate-of-deformation at node 2,

$$D_{xx} = \frac{\partial \hat{v}_x}{\partial \hat{x}} = v_{xI} \frac{\partial N_I}{\partial \xi} \left(\frac{\partial \hat{x}}{\partial \xi}\right)^{-1} = v_{xI} \frac{\partial N_I}{\partial \xi} \frac{1}{r\sin\theta}$$

$$D_{xx} = \frac{\partial \hat{v}}{\partial \hat{x}} = \{-\sin\theta \quad 0 \quad \sin\theta\} \begin{bmatrix} -\frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix} \frac{1}{r\sin\theta} = \frac{v_r}{r}$$

is then proven to be the same as the rate-of-deformation using cylindrical coordinates $\widehat{D}_{xx} = \frac{v_r}{r} = D_{\theta\theta}$.

Now we need to repeat the problem for $\xi = -3^{-1/2} = -\frac{1}{\sqrt{3}}$

For this point the nodal coordinates in the corotational system can be calculated as before. For instance, the x-component of the corotational coordinate for node 1 is:

$$\hat{x}_1 = r\left(\sin(-\theta) - \sin\left(-\frac{\theta}{\sqrt{3}}\right)\right)\cos\left(-\frac{\theta}{\sqrt{3}}\right) + r\left(\cos(-\theta) - \cos\left(-\frac{\theta}{\sqrt{3}}\right)\right)\sin\left(-\frac{\theta}{\sqrt{3}}\right)$$

Calculating the values for $\theta = 0.1\,rad$:

$Node\ 1 \rightarrow\ \emptyset_I = -\theta\ \Rightarrow\quad \hat{x}_1|_{\theta=0.1} \cong -0.04187r\ ;\ \hat{y}_1|_{\theta=0.1} \cong -0.005755r$

$Node\ 2 \rightarrow\ \emptyset_I = 0\quad \Rightarrow\quad \hat{x}_2|_{\theta=0.1} \cong 0.05751r\ ;\ \hat{y}_2|_{\theta=0.1} \cong 0.004993r$

$Node\ 3 \rightarrow\ \emptyset_I = \theta\quad \Rightarrow\quad \hat{x}_3|_{\theta=0.1} \cong 0.15747r\ ;\ \hat{y}_3|_{\theta=0.1} \cong 0.005766r$

Calculating the values for $\theta = 0.05\,rad$:

$Node\ 1 \rightarrow\ \emptyset_I = -\theta\ \Rightarrow\quad \hat{x}_1|_{\theta=0.05} \cong -0.02108r\ ;\ \hat{y}_1|_{\theta=0.05} \cong -0.0014422r$

$Node\ 2 \rightarrow\ \emptyset_I = 0\quad \Rightarrow\quad \hat{x}_2|_{\theta=0.05} \cong 0.02884r\ ;\ \hat{y}_2|_{\theta=0.05} \cong 0.0012496r$

$Node\ 3 \rightarrow\ \emptyset_I = \theta\quad \Rightarrow\quad \hat{x}_3|_{\theta=0.05} \cong 0.078834r\ ;\ \hat{y}_3|_{\theta=0.05} \cong 0.001443r$

Also, following the same procedure as before, the nodal velocities for $\theta = 0.1\,rad$ can be determined:

$Node\ 1 \rightarrow \hat{v}_{x_1}\big|_{\theta=0.1} \cong -0.04225\ v_r\ ;\ \hat{v}_{y_1}\big|_{\theta=0.1} \cong 0.999107\ v_r$

$Node\ 2 \rightarrow \hat{v}_{x_2}\big|_{\theta=0.1} \cong 0.057703\ v_r\ ;\ \hat{v}_{y_2}\big|_{\theta=0.1} \cong 0.99833\ v_r$

$Node\ 3 \rightarrow \hat{v}_{x_3}\big|_{\theta=0.1} \cong 0.15708\ v_r\ ;\ \hat{v}_{y_3}\big|_{\theta=0.1} \cong 0.98759\ v_r$

While the nodal velocities for $\theta = 0.05\,rad$ are:

$Node\ 1 \rightarrow \hat{v}_{x_1}\big|_{\theta=0.05} \cong -0.02113\ v_r\ ;\ \hat{v}_{y_1}\big|_{\theta=0.05} \cong 0.999777\ v_r$

$Node\ 2 \rightarrow \hat{v}_{x_2}\big|_{\theta=0.05} \cong 0.028864\ v_r\ ;\ \hat{v}_{y_2}\big|_{\theta=0.05} \cong 0.99958\ v_r$

$Node\ 3 \rightarrow \hat{v}_{x_3}\big|_{\theta=0.05} \cong 0.078786\ v_r\ ;\ \hat{v}_{y_3}\big|_{\theta=0.05} \cong 0.99689\ v_r$

We can now calculate $\widehat{D}_{xx}$:

$$\widehat{D}_{xx} = \frac{\partial \hat{v}_x}{\partial \hat{x}} = \hat{v}_{xI} \frac{\partial N_I}{\partial \xi} \frac{\partial \xi}{\partial \hat{x}}$$

$$\frac{\partial \xi}{\partial \hat{x}} = \left(\frac{\partial \hat{x}}{\partial \xi}\right)^{-1}$$

$$\left.\frac{\partial \hat{x}}{\partial \xi}\right|_{\theta=0.1} = \hat{x}_I \frac{\partial N_I}{\partial \xi} = r\{-0.04187 \quad 0.05751 \quad 0.15744\} \begin{Bmatrix} -\frac{1}{2} + \left(-\frac{1}{\sqrt{3}}\right) \\ -2\left(-\frac{1}{\sqrt{3}}\right) \\ \frac{1}{2} + \left(-\frac{1}{\sqrt{3}}\right) \end{Bmatrix}$$

$$\left.\frac{\partial \hat{x}}{\partial \xi}\right|_{\theta=0.1} \cong 0.09934r$$

$$\left.\frac{\partial \hat{x}}{\partial \xi}\right|_{\theta=0.05} \cong 0.049914r$$

$$\left.\frac{\partial \hat{v}_x}{\partial \hat{x}}\right|_{\theta=0.1} = v_r\{-4225 \quad 0.057703 \quad 0.99833\} \begin{Bmatrix} -\frac{1}{2} + \left(-\frac{1}{\sqrt{3}}\right) \\ \frac{2}{\sqrt{3}} \\ \frac{1}{2} - \frac{1}{\sqrt{3}} \end{Bmatrix} \frac{1}{0.09934\, r} \cong \frac{v_r}{r}$$

$$\left.\frac{\partial \hat{v}_x}{\partial \hat{x}}\right|_{\theta=0.05} \approx \frac{v_r}{r}$$

Therefore, we again conclude that $\left.\widehat{D}_x\right|_{\theta=0.1} = \left.\widehat{D}_x\right|_{\theta=0.05} = \frac{v_r}{r} = D_{\theta\theta}$.

3.4. Use Nanson's relation (3.4.5) to show that the material time derivative of a surface integral is given by

$$\frac{d}{dt}\int_S g\mathbf{n}\, dS = \int_S \left[(\dot{g} + g\nabla \cdot \mathbf{v})\mathbf{I} - g\mathbf{L}^T\right] \cdot \mathbf{n}\, dS$$

This result is used in Chapter 6 in the derivation of load stiffness.

Solution to 3.4.

From Nanson's relation:

$$\boldsymbol{n}d\Gamma = J\boldsymbol{n}_0.\boldsymbol{F}^{-1}d\Gamma_0$$

We can write:

$$\frac{d}{dt}\int_S g\boldsymbol{n}dS = \frac{d}{dt}\int_{S_0} gJ\boldsymbol{n}_0.\boldsymbol{F}^{-1}dS_0$$

$$= \int_{S_0}\left(\dot{g}J\boldsymbol{n}_0\cdot\boldsymbol{F}^{-1} + g\,\dot{J}\boldsymbol{n}_0\cdot\boldsymbol{F}^{-1} + gJ\boldsymbol{n}_0\cdot\dot{\boldsymbol{F}}^{-1}\right)dS_0$$

$$= \int_{S_0}\left[(\dot{g} + g\nabla\cdot\boldsymbol{v})J\boldsymbol{n}_0\cdot\boldsymbol{F}^{-1} + gJ\boldsymbol{n}_0\cdot\dot{\boldsymbol{F}}^{-1}\right]dS_0$$

Considering $\boldsymbol{L} = \dot{\boldsymbol{F}}\cdot\boldsymbol{F}^{-1} \Rightarrow \dot{\boldsymbol{F}}^{-1}\cdot\boldsymbol{L} = \boldsymbol{F}^{-1} \Rightarrow \dot{\boldsymbol{F}}^{-1} = \boldsymbol{F}^{-1}\cdot\boldsymbol{L}^{-1}$

$$\frac{d}{dt}\int_S g\boldsymbol{n}ds = \int_S[(\dot{g} + g\nabla\cdot\boldsymbol{v})\boldsymbol{n} + g\boldsymbol{n}\cdot\boldsymbol{L}^{-1}]\,dS$$

$$= \int_S[(\dot{g} + g\nabla\cdot\boldsymbol{v})\,\boldsymbol{n} + g\boldsymbol{L}^{-T}\cdot\boldsymbol{n}]\,ds$$

$$= \int_S[(\dot{g} + g\nabla\cdot\boldsymbol{v})\,\boldsymbol{I} + g\boldsymbol{L}^{-T}]\cdot\boldsymbol{n}ds$$

Noting that:

$$\frac{D}{Dt}(\boldsymbol{F}\cdot\boldsymbol{F}^{-1}) = 0 \Rightarrow \dot{\boldsymbol{F}}\cdot\boldsymbol{F}^{-1} + \boldsymbol{F}\cdot\dot{\boldsymbol{F}}^{-1} = 0 \Rightarrow \boldsymbol{L}^{-1} = -\boldsymbol{L} \Rightarrow \boldsymbol{L}^{-T} = -\boldsymbol{L}^{T}$$

Therefore,

$$\frac{d}{dt}\int_S g\boldsymbol{n}\,ds = \int_S[(\dot{g} + g\nabla\cdot\boldsymbol{v})\boldsymbol{I} - g\boldsymbol{L}^{T}]\cdot\boldsymbol{n}\,dS \qquad q.e.d.$$

3.5. (a) Show that for any two second order tensors A and B, the Jaumann rate has the property that

$$(\mathbf{A}:\mathbf{B}) = \dot{\mathbf{A}}:\mathbf{B} + \mathbf{A}:\dot{\mathbf{B}} = \mathbf{A}^{\nabla J}:\mathbf{B} + \mathbf{A}:\mathbf{B}^{\nabla J}$$

(b) Show that for symmetric tensors A and B, the additional results

$$\mathbf{A}:\mathbf{B} = \mathbf{A}:\mathbf{B}^{\nabla J} \quad \text{or} \quad \dot{\mathbf{A}}:\mathbf{B} = \mathbf{A}^{\nabla J}:\mathbf{B}$$

hold if A and B commute (i.e., are coaxial or have the same principal directions).

(c) Finally, show that the results in a) and b) hold for any spin-based rate, i.e.,

$$\mathbf{A}^{\nabla} = \dot{\mathbf{A}} - \boldsymbol{\Omega}\cdot\boldsymbol{\Lambda} - \mathbf{A}\cdot\boldsymbol{\Omega}^{T}$$

where $\boldsymbol{\Omega} = -\boldsymbol{\Omega}^{T}$ is a spin tensor.

These results, due to Prager, are used in Chapter 5 in developing the elasto-plastic tangent modulus.

Solution to 3.5a).

From the definition of Jaumann rate:

$$\mathbf{A}^{\nabla \mathrm{J}} = \frac{\mathrm{D}\mathbf{A}}{\mathrm{Dt}} - \mathbf{W}\,.\,\mathbf{A} - \mathbf{A}\,.\,\mathbf{W}^{\mathbf{T}}$$

$$\mathbf{B}^{\nabla \mathrm{J}} = \frac{\mathrm{D}\mathbf{B}}{\mathrm{Dt}} - \mathbf{W}.\mathbf{B} - \mathbf{B}.\mathbf{W}^{\mathbf{T}}$$

We want to show that

$$\frac{D}{Dt}(\mathbf{A}:\mathbf{B}) = \dot{\mathbf{A}}:\mathbf{B} + \mathbf{A}:\dot{\mathbf{B}} = \mathbf{A}^{\nabla \mathrm{J}}:\mathbf{B} + \mathbf{A}:\mathbf{B}^{\nabla \mathrm{J}}$$

Developing the RHS:

$$\mathbf{A}^{\nabla \mathrm{J}}:\mathbf{B} + \mathbf{A}:\mathbf{B}^{\nabla \mathrm{J}} = \left(\dot{\mathbf{A}} - \mathbf{W}\cdot\mathbf{A} - \mathbf{A}\cdot\mathbf{W}^{\mathrm{T}}\right):\mathbf{B} + \mathbf{A}:\left(\dot{\mathbf{B}} - \mathbf{W}\cdot\mathbf{B} - \mathbf{B}\cdot\mathbf{W}^{\mathrm{T}}\right)$$

$$= \dot{\mathbf{A}}:\mathbf{B} - (\mathbf{W}\cdot\mathbf{A}):\mathbf{B} - (\mathbf{A}\cdot\mathbf{W}^{\mathrm{T}}):\mathbf{B} + \mathbf{A}:\dot{\mathbf{B}} - \mathbf{A}:(\mathbf{W}\cdot\mathbf{B}) - \mathbf{A}:(\mathbf{B}\cdot\mathbf{W}^{\mathrm{T}})$$

$$= \dot{\mathbf{A}}:\mathbf{B} - \mathbf{B}:(\mathbf{W}\cdot\mathbf{A}) + \mathbf{B}:(\mathbf{A}\cdot\mathbf{W}) + \mathbf{A}:\dot{\mathbf{B}} - \mathbf{A}:(\mathbf{W}\cdot\mathbf{B}) + \mathbf{A}:(\mathbf{B}\cdot\mathbf{W})$$

Where in the last line we used the definition of skew-symmetric tensor for $\mathbf{W}$ and the property of the double dot product $\mathbf{U}:\mathbf{V} = \mathbf{V}:\mathbf{U}$.

Recalling that any skew-symmetric second-order tensor can be expressed in terms of the components of a vector $\boldsymbol{\omega}$, called axial vector, we can rewrite the above equation as:

$$\mathbf{A}^{\nabla \mathrm{J}}\colon \mathbf{B} + \mathbf{A} : \mathbf{B}^{\nabla \mathrm{J}} = \dot{\mathbf{A}}\colon \mathbf{B} - (\boldsymbol{\omega} \times \mathbf{A})\colon \mathbf{B} + \mathbf{B}\colon (\mathbf{A} \times \boldsymbol{\omega}) + \mathbf{A}\colon \dot{\mathbf{B}} - (\boldsymbol{\omega} \times \mathbf{B})\colon \mathbf{A} + \mathbf{A}\colon (\mathbf{B} \times \boldsymbol{\omega})$$

Therefore, we can now permute the order of the entities accordingly (odd permutations need a negative sign):

$$\begin{aligned}\mathbf{A}^{\nabla \mathrm{J}}\colon \mathbf{B} + \mathbf{A} : \mathbf{B}^{\nabla \mathrm{J}} &= \dot{\mathbf{A}}\colon \mathbf{B} + (\boldsymbol{\omega} \times \mathbf{B})\colon \mathbf{A} - \mathbf{A}\colon (\mathbf{B} \times \boldsymbol{\omega}) + \mathbf{A}\colon \dot{\mathbf{B}} - (\boldsymbol{\omega} \times \mathbf{B})\colon \mathbf{A} + \mathbf{A}\colon (\mathbf{B} \times \boldsymbol{\omega}) \\ &= \dot{\mathbf{A}}\colon \mathbf{B} + \mathbf{A}\colon \dot{\mathbf{B}}\end{aligned}$$

Hence,

$$\frac{D}{Dt}(\mathbf{A}\colon \mathbf{B}) = \mathbf{A}^{\nabla \mathrm{J}}\colon \mathbf{B} + \mathbf{A} : \mathbf{B}^{\nabla \mathrm{J}} = \dot{\mathbf{A}}\colon \mathbf{B} + \mathbf{A}\colon \dot{\mathbf{B}} \qquad \text{q. e. d}$$

Solution to 3.5b).

It is enough to prove one of the results, let's prove that $\mathbf{A}\colon \mathbf{B}^{\nabla \mathrm{J}} = \mathbf{A} : \dot{\mathbf{B}}$

$$\begin{aligned}\mathbf{A}\colon \mathbf{B}^{\nabla \mathrm{J}} &= \mathbf{A}\colon \left(\dot{\mathbf{B}} - \mathbf{W} \cdot \mathbf{B} - \mathbf{B} \cdot \mathbf{W}^{\mathrm{T}}\right) = \mathbf{A}\colon \dot{\mathbf{B}} - \mathbf{A}\colon (\mathbf{W} \cdot \mathbf{B}) - \mathbf{A}\colon (\mathbf{B} \cdot \mathbf{W}^{\mathrm{T}}) \\ &= \mathbf{A}\colon \dot{\mathbf{B}} - \mathbf{A}\colon (\mathbf{W} \cdot \mathbf{B}) + \mathbf{A}\colon (\mathbf{B} \cdot \mathbf{W})\end{aligned}$$

Recalling $\mathbf{U}\colon \mathbf{V} = \mathbf{U}^{\mathrm{T}}\colon \mathbf{V}^{\mathrm{T}}$ as a property of the double dot product, and noting that $\mathbf{A}$ and $\mathbf{B}$ are symmetric tensors, we can rewrite the above equation as:

$$\begin{aligned}\mathbf{A}\colon \mathbf{B}^{\nabla \mathrm{J}} &= \mathbf{A}\colon \dot{\mathbf{B}} - \mathbf{A}^{\mathrm{T}}\colon (\mathbf{W} \cdot \mathbf{B})^{\mathrm{T}} + \mathbf{A}\colon (\mathbf{B} \cdot \mathbf{W}) = \mathbf{A}\colon \dot{\mathbf{B}} - \mathbf{A}^{\mathrm{T}}\colon (\mathbf{B}^{\mathrm{T}} \cdot \mathbf{W}^{\mathrm{T}}) + \mathbf{A}\colon (\mathbf{B} \cdot \mathbf{W}) \\ &= \mathbf{A}\colon \dot{\mathbf{B}} + \mathbf{A}\colon (\mathbf{B} \cdot \mathbf{W}) + \mathbf{A}\colon (\mathbf{B} \cdot \mathbf{W}) = \mathbf{A}\colon \dot{\mathbf{B}} + 2\mathbf{A}\colon (\mathbf{B} \cdot \mathbf{W})\end{aligned}$$

Rewriting the above result using the axial vector $\boldsymbol{\omega}$:

$$\mathbf{A}^{\nabla \mathrm{J}}\colon \mathbf{B} + \mathbf{A} : \mathbf{B}^{\nabla \mathrm{J}} = \dot{\mathbf{A}}\colon \mathbf{B} + 2\mathbf{A}\colon (\mathbf{B} \times \boldsymbol{\omega})$$

If **A** and **B** are co-axial tensors they have the same eigenvectors; therefore, they can be written as:

$$\mathbf{A} = \sum_{a=1}^{3} A_a \mathbf{n}^{(a)} \otimes \mathbf{n}^{(a)} \qquad ; \qquad \mathbf{B} = \sum_{a=1}^{3} B_a \mathbf{n}^{(a)} \otimes \mathbf{n}^{(a)}$$

where $\otimes$ stands for the dyadic product.

We can then write:

$$\mathbf{A}: \mathbf{B}^{\nabla J} = \dot{\mathbf{A}}: \mathbf{B} + 2 \sum_{a=1}^{3} A_a \mathbf{n}^{(a)} \otimes \mathbf{n}^{(a)} : \left(\sum_{b=1}^{3} B_b \mathbf{n}^{(b)} \otimes \mathbf{n}^{(b)} \times \boldsymbol{\omega} \right)$$

$$= \dot{\mathbf{A}}: \mathbf{B} + 2 \sum_{a=1}^{3} A_a \sum_{a=1}^{3} B_a \left(\mathbf{n}^{(a)} \cdot \mathbf{n}^{(b)} \right) \left(\mathbf{n}^{(a)} \cdot \mathbf{n}^{(b)} \times \boldsymbol{\omega} \right) = \dot{\mathbf{A}}: \mathbf{B} \qquad \text{q. e. d.}$$

Where the last equality is obtained if we note that:

- For $a \neq b \Rightarrow \mathbf{n}^{(a)} \cdot \mathbf{n}^{(b)} = 0$, because the eigenvectors are orthogonal;
- For $a = b \Rightarrow \mathbf{n}^{(a)} \cdot \mathbf{n}^{(b)} \times \boldsymbol{\omega} = \mathbf{n}^{(a)} \cdot \mathbf{n}^{(a)} \times \boldsymbol{\omega} = 0$, due to the fact that the cross product of $\mathbf{n}^{(a)}$ with $\boldsymbol{\omega}$ produces a perpendicular vector to both these vectors, which is also perpendicular to $\mathbf{n}^{(a)}$.

Solution to 3.5c).

Trivial considering the previous answers.

3.6. (a) Use the results in Problem 3 and the expressions for the principal invariants of a tensor in Box 5.2 to show that the material time derivatives of the principal invariants can be written as

$$\dot{I}_1 = \dot{\boldsymbol{\sigma}} : \mathbf{I} = \boldsymbol{\sigma}^{\nabla J} : \mathbf{I}$$
$$\dot{I}_2 = I_1 \dot{\boldsymbol{\sigma}} : \mathbf{I} - (\dot{\boldsymbol{\sigma}} \cdot \dot{\boldsymbol{\sigma}}) : \mathbf{I} = \boldsymbol{\sigma}^{\nabla J} : \mathbf{I} - (\boldsymbol{\sigma}^{\nabla J} \cdot \boldsymbol{\sigma}) : \mathbf{I}$$
$$\dot{I}_3 = I_3 \text{ trace } (\dot{\boldsymbol{\sigma}} \cdot \boldsymbol{\sigma}^{-1}) = I_3 \text{ trace } (\boldsymbol{\sigma}^{\nabla J} \cdot \boldsymbol{\sigma}^{-1})$$

It follows that if the Jaumann rate of Cauchy stress vanishes, i.e., $\boldsymbol{\sigma}^{\nabla J} = 0$, then the principal invariants of the Cauchy stress are stationary.

(b) Show that if the material time-derivative of the Cauchy stress is deviatoric then the Jaumann rate of Cauchy stress is deviatoric.

From Problem 5(c), it follows that these results also hold for any symmetric tensor and for any spin-based rate.

Solution to 3.6a).

Let's start with the first invariant:

$$I_1 = \text{trace}(\boldsymbol{\sigma}) = \sigma_{ii} = \sigma_{ij}\delta_{ij} = \boldsymbol{\sigma} : \mathbf{I}$$

Therefore, calculating the time derivative of I_1:

$$\dot{I}_1 = \frac{D}{Dt}(\boldsymbol{\sigma} : \mathbf{I}) = \dot{\boldsymbol{\sigma}} : \mathbf{I} + \boldsymbol{\sigma} : \dot{\mathbf{I}}$$

Since $\dot{\mathbf{I}} = 0$, and using the result obtained in Problem 5b):

$$\dot{I}_1 = \dot{\boldsymbol{\sigma}} : \mathbf{I} = \boldsymbol{\sigma}^{\nabla \mathrm{J}} : \mathbf{I} \qquad q.e.d.$$

Now, considering the second invariant:

$$I_2 = \frac{1}{2}\{[\text{trace}(\boldsymbol{\sigma})]^2 - \text{trace}(\boldsymbol{\sigma}^2)\} = \frac{1}{2}[I_1^2 - \boldsymbol{\sigma} : \boldsymbol{\sigma}]$$

Calculating the time derivative of this invariant:

$$\dot{I}_2 = \frac{1}{2}\left[2I_1\dot{I}_1 - \dot{\boldsymbol{\sigma}} : \boldsymbol{\sigma} - \boldsymbol{\sigma} : \dot{\boldsymbol{\sigma}}\right]$$

Recalling that for any two tensors $\mathbf{A}$ and $\mathbf{B}$, the following holds: $\mathbf{A}:\mathbf{B} = (\mathbf{A}\cdot\mathbf{B}^T):\mathbf{I}$, we can rewrite the above expression as:

$$\dot{I}_2 = \frac{1}{2}\left[2I_1\dot{I}_1 - (\dot{\boldsymbol{\sigma}}\cdot\boldsymbol{\sigma}^T):\mathbf{I} - (\boldsymbol{\sigma}\cdot\dot{\boldsymbol{\sigma}}^T):\mathbf{I}\right]$$

From the following property of the double dot product of two tensors: $\mathbf{A}^{\mathrm{T}}:\mathbf{B} = \mathbf{A}:\mathbf{B}^{\mathrm{T}}$, and recalling that the Cauchy stress is symmetric:

$$\dot{I}_2 = \frac{1}{2}\left[2I_1\dot{I}_1 - (\dot{\boldsymbol{\sigma}}\cdot\boldsymbol{\sigma}):\mathbf{I} - (\dot{\boldsymbol{\sigma}}\cdot\boldsymbol{\sigma}^T):\mathbf{I}^{\mathrm{T}}\right] = \frac{1}{2}\left[2I_1\dot{I}_1 - (\dot{\boldsymbol{\sigma}}\cdot\boldsymbol{\sigma}):\mathbf{I} - (\dot{\boldsymbol{\sigma}}\cdot\boldsymbol{\sigma}):\mathbf{I}\right]$$

$$\dot{I}_2 = I_1\dot{I}_1 - (\dot{\boldsymbol{\sigma}}\cdot\boldsymbol{\sigma}):\mathbf{I}$$

Now, we can rewrite the above as:

$$\dot{I}_2 = I_1\dot{\boldsymbol{\sigma}}:\mathbf{I} - \dot{\boldsymbol{\sigma}}:\boldsymbol{\sigma}^T = I_1\dot{\boldsymbol{\sigma}}:\mathbf{I} - \dot{\boldsymbol{\sigma}}:\boldsymbol{\sigma}^T = I_1\dot{\boldsymbol{\sigma}}:\mathbf{I} - \dot{\boldsymbol{\sigma}}:\boldsymbol{\sigma}$$

Therefore, using the results of Problem 5b):

$$\dot{I}_2 = I_1\dot{\boldsymbol{\sigma}}:\mathbf{I} - (\dot{\boldsymbol{\sigma}}\cdot\boldsymbol{\sigma}):\mathbf{I} = I_1\boldsymbol{\sigma}^{\nabla\mathrm{J}}:\mathbf{I} - (\boldsymbol{\sigma}^{\nabla\mathrm{J}}\cdot\boldsymbol{\sigma}):\mathbf{I} \qquad \text{q. e. d.}$$

Finally, let's consider the third invariant:

$$I_3 = \det(\boldsymbol{\sigma})$$

Recalling that the derivative of the determinant of a square matrix $\mathbf{A}$ can be expressed using Jacobi's formula:

$$\frac{D}{Dt}[\det(\mathbf{A})] = \mathrm{trace}\left[\mathrm{adj}(\mathbf{A})\frac{D\mathbf{A}}{Dt}\right] = \det(\mathbf{A})\mathrm{trace}\left[\mathbf{A}^{-1}\cdot\frac{D\mathbf{A}}{Dt}\right]$$

We can write the time derivative of the third invariant as:

$$\dot{I}_3 = \det(\boldsymbol{\sigma})\mathrm{trace}[\boldsymbol{\sigma}^{-1}\cdot\dot{\boldsymbol{\sigma}}] = I_3\mathrm{trace}\left[\dot{\boldsymbol{\sigma}}^T\cdot\boldsymbol{\sigma}^{-\mathrm{T}}\right] = I_3\mathrm{trace}[\dot{\boldsymbol{\sigma}}\cdot\boldsymbol{\sigma}^{-1}]$$

We can rewrite the above as:

$$\dot{I}_3 = I_3(\dot{\boldsymbol{\sigma}} \cdot \boldsymbol{\sigma}^{-1}) : \mathbf{I} = I_3 \dot{\boldsymbol{\sigma}} : \boldsymbol{\sigma}^{-\mathrm{T}}$$

From the results of Problem 5b)

$$\dot{I}_3 = I_3 \boldsymbol{\sigma}^{\nabla \mathrm{J}} : \boldsymbol{\sigma}^{-\mathrm{T}} = I_3(\boldsymbol{\sigma}^{\nabla \mathrm{J}} \cdot \boldsymbol{\sigma}^{-1}) : \mathbf{I} = I_3 \mathrm{trace}[\boldsymbol{\sigma}^{\nabla \mathrm{J}} \cdot \boldsymbol{\sigma}^{-1}] \qquad q.e.d.$$

Solution to 3.6b).

The first invariant of a deviatoric tensor is zero; therefore, if $\dot{\boldsymbol{\sigma}}$ is deviatoric:

$$I_1(\dot{\boldsymbol{\sigma}}) = \mathrm{trace}(\dot{\boldsymbol{\sigma}}) = \dot{\boldsymbol{\sigma}} : \mathbf{I} = 0$$

However, in part (a) we derived the following result:

$$\dot{\boldsymbol{\sigma}} : \mathbf{I} = \boldsymbol{\sigma}^{\nabla \mathrm{J}} : \mathbf{I}$$

Therefore,

$$\boldsymbol{\sigma}^{\nabla \mathrm{J}} : \mathbf{I} = 0 \Rightarrow \mathrm{trace}(\boldsymbol{\sigma}^{\nabla \mathrm{J}}) = 0 \qquad q.e.d.$$

Proving that the Jaumann rate of Cauchy stress is also deviatoric.

3.7. Starting from Eqs. (3.3.4) and (3.3.12), show that

$$2d\mathbf{x} \cdot \mathbf{D} \cdot d\mathbf{x} = 2d\mathbf{x} \cdot \mathbf{F}^{-T} \cdot \dot{\mathbf{E}} \cdot \mathbf{F}^{-1} \cdot d\mathbf{x}$$

and hence that Eq. (3.3.22) holds.

Solution to 3.7.

Equation (3.3.4) is:

$$d\boldsymbol{X} \cdot \left(\boldsymbol{F}^T \cdot \boldsymbol{F} - \boldsymbol{I} - 2\boldsymbol{E}\right) \cdot d\boldsymbol{X} = 0$$

Equation (3.3.12) is:

$$\frac{\partial}{\partial t}\,(ds^2) = \frac{\partial}{\partial t}(d\boldsymbol{x}\cdot d\boldsymbol{x}) = 2\,d\boldsymbol{x}\cdot\boldsymbol{D}\cdot d\boldsymbol{x} \qquad \forall dx$$

From equation (3.3.4) it is possible to define $d\boldsymbol{x}\cdot d\boldsymbol{x}$ as:

$$d\boldsymbol{X}\cdot(\boldsymbol{F}^T\cdot\boldsymbol{F}-\boldsymbol{I}-2\boldsymbol{E})\cdot d\boldsymbol{X}=0$$
$$d\boldsymbol{X}\cdot\boldsymbol{F}^T\cdot\boldsymbol{F}\cdot d\boldsymbol{X}-d\boldsymbol{X}\cdot\boldsymbol{I}\cdot d\boldsymbol{X}-2d\boldsymbol{X}\cdot\boldsymbol{E}\cdot d\boldsymbol{X}=0$$
$$d\boldsymbol{x}\cdot d\boldsymbol{x}-d\boldsymbol{X}\cdot d\boldsymbol{X}-2d\boldsymbol{X}\cdot\boldsymbol{E}\cdot d\boldsymbol{X}=0$$
$$d\boldsymbol{x}\cdot d\boldsymbol{x}=d\boldsymbol{X}\cdot d\boldsymbol{X}+2d\boldsymbol{X}\cdot\boldsymbol{E}\cdot d\boldsymbol{X}$$

Going back to equation (3.3.12) and using this result:

$$\frac{\partial}{\partial t}(d\boldsymbol{x}\cdot d\boldsymbol{x}) = \frac{\partial}{\partial t}(d\boldsymbol{X}\cdot d\boldsymbol{X}+2d\boldsymbol{X}\cdot\boldsymbol{E}\cdot d\boldsymbol{X}) = 2d\boldsymbol{X}\cdot\dot{\boldsymbol{E}}\cdot d\boldsymbol{X}$$

since the reference configuration does not change with time. Recalling that $\boldsymbol{F}=\frac{d\boldsymbol{x}}{d\boldsymbol{X}}\Rightarrow d\boldsymbol{X}=\boldsymbol{F}^{-1}\cdot d\boldsymbol{x}$, and substituting this result in the previous equation:

$$\frac{\partial}{\partial t}(d\boldsymbol{x}\cdot d\boldsymbol{x}) = 2d\boldsymbol{x}\cdot\boldsymbol{F}^{-T}\cdot\dot{\boldsymbol{E}}\cdot\boldsymbol{F}^{-1}\cdot d\boldsymbol{x} = 2d\boldsymbol{x}\cdot\boldsymbol{D}\cdot d\boldsymbol{x} \qquad q.e.d.$$

3.8. Using the statement of the conservation of momentum in the Lagrangian description in the initial configuration, show that it implies

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

Solution to 3.8.

From the conservation of the linear momentum in the Lagrangian description in the initial configuration, one can write:

$$\frac{D}{Dt}\int_{\Omega_0}\rho_0\boldsymbol{v}d\Omega_0 = \int_{\Omega_0}\rho_0\boldsymbol{b}d\Omega_0 + \int_{\Gamma_0}\boldsymbol{t}_0 d\Gamma_0$$

The integral form of the conservation of the angular momentum can then be obtained by taking the cross product of each term in the corresponding linear momentum principle with the position vector **x** (in the current configuration, even though the linear momentum is expressed in terms of the initial configuration):

$$\frac{D}{Dt}\int_{\Omega_0} \boldsymbol{x} \times \rho_0 d\,\Omega_0 = \int_{\Omega_0} \boldsymbol{x} \times \rho_0 \boldsymbol{b} d\Omega_0 + \int_{\Gamma_0} \boldsymbol{x} \times \boldsymbol{t}_0 d\Gamma_0$$

The last term on the RHS can be rewritten as:

$$\int_{\Gamma_0} \boldsymbol{x} \times \boldsymbol{t}_0 d\Gamma_0 = \int_{\Gamma_0} \boldsymbol{x} \times (\boldsymbol{n}_0 \cdot \boldsymbol{P}) d\Gamma_0 = \int_{\Gamma_0} \boldsymbol{n}_0 \cdot (\mathbf{P} \times \boldsymbol{x})\, d\Gamma_0 = \int_{\Omega_0} \nabla \cdot (\mathbf{P} \times \boldsymbol{x})\, d\Omega_0$$

Or using the component form:

$$\int_{\Gamma_0} \varepsilon_{ijk} x_i t_j^0 \; d\Gamma_0 = \int_{\Gamma_0} \varepsilon_{ijk} x_i n_l^0 P_{lj}\, d\Gamma_0 = \int_{\Gamma_0} \varepsilon_{ijk} \frac{\partial}{\partial X_l}\left(x_i P_{lj}\right)\, d\Gamma_0$$

$$= \int_{\Gamma_0} \varepsilon_{ijk} \left(\frac{\partial x_i}{\partial X_l} P_{lj} + x_i \frac{\partial P_{lj}}{\partial X_l}\right) d\Omega_0$$

Therefore, going back to the equation of the conservation of angular momentum, using the above obtained result, and using the component form:

$$\int_{\Omega_0} \left[\varepsilon_{ijk}\rho_0 v_i v_j + \varepsilon_{ijk}\rho_0 x_i \frac{Dv_j}{Dt} - \varepsilon_{ijk}\rho_0 b_i - \varepsilon_{ijk}\left(F_{il}P_{lj} + x_i \frac{\partial P_{lj}}{\partial X_l}\right)\right] d\Omega_0 = 0$$

Noting that the first term is zero (cross product of parallel vectors), and rearranging the remaining terms:

$$\int_{\Omega_0} \varepsilon_{ijk}\, x_i \left(\rho_0 \frac{Dv_j}{Dt} - \rho_0 b_j - \frac{\partial P_{lj}}{\partial X_l}\right) d\Omega_0 - \int_{\Omega_0} \varepsilon_{ijk}\, F_{il} P_{lj} \cdot d\Omega_0 = 0$$

Noting that the expression inside the brackets is the linear momentum conservation, we get:

$$-\int_{\Omega_0} \varepsilon_{ijk}\; F_{il} P_{lj}\, d\Omega_0 = 0$$

For an arbitrary volume:

$$\varepsilon_{ijk} F_{il} P_{lj} = 0$$

Multiplying by ε_{pqk}:

$$\varepsilon_{pqk}\varepsilon_{ijk}F_{il}P_{lj} = 0$$

$$\left(\delta_{pi}\delta_{qj} - \delta_{pj}\delta_{qi}\right)F_{il}P_{lj} = 0$$

$$F_{pl}P_{lg} - F_{lq}^{T}P_{pl}^{T} = 0 \Rightarrow \boldsymbol{F}\cdot\boldsymbol{P} = \boldsymbol{P}^{T}\cdot\boldsymbol{F}^{T} \qquad q.e.d.$$

3.9. Extend Example 3.3 by finding the conditions at which the Jacobian becomes negative at the Gauss quadrature points for 2 × 2 quadrature when the initial element is rectangular with dimension $a \times b$. Repeat for one-point quadrature, with the quadrature point at the center of the element.

Solution to 3.9.

Since this element is not a square, the motion is not the same as in Example 3.3. Although the motion can be easily determined for this problem we will use the isoparametric mapping for a four node element to find the Jacobian.

The nodal coordinates in the reference configuration:

$$X_1 = 0\ ;\ Y_1 = 0\ ;\ X_2 = a\ ;\ Y_2 = 0\ ;\ X_3 = a\ ;\ Y_3 = b\ ;\ X_4 = 0\ ;\ Y_4 = b$$

The nodal coordinates in the deformed configuration are:

$$x_1 = 0\ ;\ y_1 = 0\ ;\ x_2 = a\ ;\ y_2 = 0\ ;\ x_3 = a + u_{x3}\ ;\ y_3 = b + u_{y3}\ ;$$

$$x_4 = 0\ ;\ y_4 = b$$

The shape functions for this element are:

$$\boldsymbol{N} = \frac{1}{4}[(1-\xi)(1-\eta) \quad (1+\xi)(1-\eta) \quad (1+\xi)(1+\eta) \quad (1-\xi)(1+\eta)]$$

We can then write:

$$X_i = X_{iI}N_I \Rightarrow \boldsymbol{X} = \begin{Bmatrix} X \\ Y \end{Bmatrix} = \begin{bmatrix} 0 & a & a & 0 \\ 0 & 0 & b & b \end{bmatrix} \frac{1}{4} \begin{bmatrix} (1-\xi)(1-\eta) \\ (1+\xi)(1-\eta) \\ (1+\xi)(1+\eta) \\ (1-\xi)(1+\eta) \end{bmatrix} = \frac{1}{2} \begin{Bmatrix} a(1+\xi) \\ b(1+\eta) \end{Bmatrix}$$

$$x_i = x_{iI}N_I \Rightarrow \boldsymbol{x} = \begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{bmatrix} 0 & a & a+u_{x3} & 0 \\ 0 & 0 & b+u_{y3} & b \end{bmatrix} \frac{1}{4} \begin{bmatrix} (1-\xi)(1-\eta) \\ (1+\xi)(1-\eta) \\ (1+\xi)(1+\eta) \\ (1-\xi)(1+\eta) \end{bmatrix}$$

$$= \frac{1}{4} \begin{Bmatrix} a(1-\eta)(1+\xi) + (a+u_{x3})(1+\eta)(1+\xi) \\ b(1+\eta)(1-\xi) + (b+u_{y3})(1+\eta)(1+\xi) \end{Bmatrix}$$

From the result for the Lagrangian coordinates $\boldsymbol{X}$ for this problem it is possible to find the motion by explicitly writing the parent coordinates $\boldsymbol{\xi}$ in terms of the Lagrangian coordinates $\boldsymbol{X}$ (note that this is not necessary to solve the problem; we could solve it by writing the derivatives using the chain rule, as usual):

$$\boldsymbol{\xi} = \begin{Bmatrix} \xi \\ \eta \end{Bmatrix} = \begin{Bmatrix} -1 + \dfrac{2X}{a} \\ -1 + \dfrac{2Y}{b} \end{Bmatrix}$$

Substituting this result in the current coordinates we determine the motion:

$$\boldsymbol{x} = \begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} \dfrac{(a+u_{x3})XY}{ab} + \dfrac{1}{2}X\left(2 - \dfrac{2Y}{b}\right) \\ \dfrac{(b+u_{y3})XY}{ab} + \dfrac{1}{2}Y\left(2 - \dfrac{2X}{a}\right) \end{Bmatrix}$$

We can now determine the deformation gradient:

$$\boldsymbol{F} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} = \begin{bmatrix} \dfrac{(a+u_{x3})Y}{ab} + \dfrac{1}{2}\left(2 - \dfrac{2Y}{b}\right) & \dfrac{-X}{b} + \dfrac{(a+u_{x3})X}{ab} \\ \dfrac{-Y}{a} + \dfrac{(b+u_{y3})Y}{ab} & \dfrac{(b+u_{y3})X}{ab} + \dfrac{1}{2}\left(2 - \dfrac{2X}{a}\right) \end{bmatrix}$$

Since we have the expression for the deformation gradient as a function of the Lagrangian coordinates, to find the value of the Jacobian for the 4 Gauss points we need to determine the coordinates of the Gauss points in the reference domain:

First Gauss point in parent coordinates: $\xi_{GP1} = -\frac{1}{\sqrt{3}} \quad ; \quad \eta_{GP1} = -\frac{1}{\sqrt{3}}$

In the reference domain using the mapping obtained for $\boldsymbol{X}$:

$$X_{GP1} = \frac{a}{2}\left(1 - \frac{1}{\sqrt{3}}\right) \quad ; \quad Y_{GP1} = \frac{b}{2}\left(1 - \frac{1}{\sqrt{3}}\right)$$

Calculating the Jacobian at this point:

$$J_{GP1} = \det(\boldsymbol{F}) = 1 + \frac{\left(3 - \sqrt{3}\right)u_{x3}}{6a} + \frac{\left(3 - \sqrt{3}\right)u_{y3}}{6b}$$

Therefore, the Jacobian at the first Gauss point becomes negative for:

$$J_{GP1} < 0 \Rightarrow u_{y3} < -\frac{6b}{3 - \sqrt{3}} - \frac{b}{a}u_{x3}$$

Doing the same for the remaining Gauss points:

$$\boldsymbol{\xi}_{GP2} = \begin{Bmatrix} \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{3}} \end{Bmatrix} \quad ; \quad \boldsymbol{\xi}_{GP3} = \begin{Bmatrix} \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{Bmatrix} \quad ; \quad \boldsymbol{\xi}_{GP4} = \begin{Bmatrix} -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{Bmatrix}$$

We arrive to the following conditions for the respective Jacobians:

$$J_{GP2} < 0 \Rightarrow u_{y3} < -\frac{6b}{3 + \sqrt{3}} - \left(2 - \sqrt{3}\right)\frac{b}{a}u_{x3}$$

$$J_{GP3} < 0 \Rightarrow u_{y3} < -\frac{6b}{3 + \sqrt{3}} - \frac{b}{a}u_{x3}$$

$$J_{GP4} < 0 \Rightarrow u_{y3} < -\frac{6b}{3 - \sqrt{3}} - \left(\sqrt{3} - 2\right)\frac{b}{a}u_{x3}$$

Finally, it was asked to calculate when the Jacobian becomes negative for the case of a single Gauss point in the middle of the element:

$$\boldsymbol{\xi}_{GP_{single}} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

So, the following condition arises:

$$J_{GP_{single}} < 0 \Rightarrow u_{y3} < -2b - \frac{b}{a} u_{x3}$$

3.10. Derive (3.2.19).

Solution to 3.10.

Trivial from the textbook.

CHAPTER 4: Lagrangian meshes

4.1. Consider the element shown in Figure 3.4 with the motion

$$x(\mathbf{X},t) = (1+at)X\cos\frac{\pi}{2}t - (1+bt)Y\sin\frac{\pi}{2}t$$

$$y(\mathbf{X},t) = (1+at)X\sin\frac{\pi}{2}t + (1+bt)Y\cos\frac{\pi}{2}t$$

Sketch the element in the deformed configuration at $t = 1$ **(this was already done in Exercise 3.1).**

(a) Let the only nonzero PK2 stress component in the deformed configuration be S_{11}**. Find the nodal internal forces.**

(b) For the same state of stress, find the nodal internal forces in the under-formed configuration. What is the effect of rotating the body on the nodal internal forces?

(c) Repeat the above parts a **and** b **with the only non-zero components being** S_{22} **and** S_{12}**. Explain the nodal internal forces in the undeformed and deformed configurations.**

Solution to 4.1a).

$$x_1|_{t=1} = 0\,; \qquad y_1|_{t=1} = 0$$

$$x_2|_{t=1} = 0 \quad ; \quad y_2|_{t=1} = 2\,(1+a)$$

$$x_3|_{t=1} = -(1+b) \quad ; \quad y_3|_{t=1} = 0$$

The map between the parent element and the initial configuration is:

$$\begin{Bmatrix} X \\ Y \\ 1 \end{Bmatrix} = \begin{bmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{Bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{Bmatrix}$$

$$with\ X_1 = 0; Y_1 = 0\,; X_2 = 2; Y_2 = 0; X_3 = 0; Y_3 = 1$$

The inverse of this relation is given by:

$$\begin{Bmatrix}\xi_1\\ \xi_2\\ \xi_3\end{Bmatrix} = \frac{1}{2A_0}\begin{bmatrix}Y_{23} & X_{32} & X_2Y_3 - X_3Y_2\\ Y_{31} & X_{13} & X_3Y_1 - X_1Y_3\\ Y_{12} & X_{21} & X_1Y_2 - X_2Y_1\end{bmatrix}\begin{Bmatrix}X\\ Y\\ 1\end{Bmatrix}$$

where $X_{IJ} = X_I - X_J$, $Y_{IJ} = Y_I - Y_J$, and $2A_0 = X_{32}Y_{12} - X_{12}Y_{32}$.

The shape functions for the linear displacement triangle are the triangular coordinates so $N_I = \xi_I$. This way, the $\boldsymbol{\mathcal{B}}_0$ matrix is given by:

$$\boldsymbol{\mathcal{B}}_I^0 = \left[\boldsymbol{\mathcal{B}}_{JI}^0\right] = \left[\frac{\partial N_I}{\partial X_j}\right]$$

$$\boldsymbol{\mathcal{B}}_0 = \left[\boldsymbol{\mathcal{B}}_1^0\, \boldsymbol{\mathcal{B}}_2^0 \boldsymbol{\mathcal{B}}_3^0\right] = \begin{bmatrix}\frac{\partial N_1}{\partial X} & \frac{\partial N_2}{\partial X} & \frac{\partial N_3}{\partial X}\\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \frac{\partial N_3}{\partial y}\end{bmatrix} = \frac{1}{2A_0}\begin{bmatrix}Y_{23} & Y_{31} & Y_{12}\\ Y_{32} & X_{13} & X_{21}\end{bmatrix}$$

$$\boldsymbol{\mathcal{B}}_0 = \frac{1}{2}\begin{bmatrix}-1 & 1 & 0\\ -2 & 0 & 2\end{bmatrix}$$

Now, using Voigt notation the expression to calculate the internal nodal forces is:

$$\boldsymbol{f}_I^{int} = \int_{\Omega_0} \boldsymbol{B}_{0_I}^{\,T}\{\boldsymbol{S}\}\; d\Omega_0$$

where $\boldsymbol{B}_{OI} = \begin{bmatrix}\frac{\partial N_I}{\partial X}\frac{\partial x}{\partial X} & \frac{\partial N_I}{\partial X}\frac{\partial y}{\partial X}\\ \frac{\partial N_I}{\partial Y}\frac{\partial x}{\partial Y} & \frac{\partial N_I}{\partial Y}\frac{\partial y}{\partial Y}\\ \frac{\partial N_I}{\partial X}\frac{\partial x}{\partial Y} + \frac{\partial N_I}{\partial Y}\frac{\partial x}{\partial X} & \frac{\partial N_I}{\partial X}\frac{\partial x}{\partial Y} + \frac{\partial N_I}{\partial Y}\frac{\partial y}{\partial X}\end{bmatrix}$.

The terms $\partial N_I/\partial X$ and $\partial N_I/\partial Y$ were calculated for the $\boldsymbol{\mathcal{B}}_0$ matrix; the terms of the gradient tensor are calculated by:

$$\frac{\partial x}{\partial X} = x_{,X} = \frac{\partial N_I}{\partial X}\, x_I = \frac{1}{2A_0}\,(Y_{23}\, x_1 + Y_{31}x_2 + Y_{12}x_3)$$

$$\left.\frac{\partial x}{\partial X}\right|_{t=1} = 0$$

$$\frac{\partial x}{\partial Y} = \frac{\partial N_I}{\partial Y} x_I = \frac{1}{2A_0}(X_{32}x_1 + X_{13}x_2 + X_{21}x_3)$$

$$\left.\frac{\partial x}{\partial Y}\right|_{t=1} = -1 - b$$

$$\frac{\partial y}{\partial x} = \frac{\partial N_I}{\partial X} y_I = \frac{1}{2A_0}(Y_{23}Y_1 + Y_{31}Y_2 + Y12\,Y_3)$$

$$\left.\frac{\partial y}{\partial X}\right|_{t=1} = 1 + a$$

$$\frac{\partial y}{\partial y} = \frac{\partial N_I}{\partial Y} Y_I = \frac{1}{2A_0}(X_{32}Y_1 + X_{13}Y_2 + X_{21}Y_3)$$

$$\left.\frac{\partial y}{\partial Y}\right|_{t=1} = 0$$

Thus,

$$\boldsymbol{B}_0|_{t=1} = \begin{bmatrix} 0 & \frac{-1-a}{2} & 0 & \frac{1+a}{2} & 0 & 0 \\ 1+b & 0 & 0 & 0 & -1-b & 0 \\ \frac{1+b}{2} & -1-a & \frac{-1-b}{2} & 0 & 0 & 1+a \end{bmatrix}$$

We can now calculate the nodal internal forces at t=1 for a stress state at that time with the only nonzero component being S_{11}. Note that the element is considered to have a thickness of h_0.

$$\boldsymbol{f}^{int}\big|_{t=1} = \begin{Bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \\ f_{x3} \\ f_{y3} \end{Bmatrix} = \int_{\Omega_0} \boldsymbol{B}_0^T \begin{Bmatrix} S_{11} \\ S_{22} \\ S_{12} \end{Bmatrix} d\Omega_0 = h_0 A_0 \boldsymbol{B}_0^T \begin{Bmatrix} S_{11} \\ 0 \\ 0 \end{Bmatrix}$$

$$\boldsymbol{f}^{int}\big|_{t=1} = \frac{h_0 S_{11}}{2} \begin{Bmatrix} 0 \\ -a-1 \\ 0 \\ a+1 \\ 0 \\ 0 \end{Bmatrix}$$

Solution to 4.1b).

For this part, it is necessary to recalculate the $\boldsymbol{\mathcal{B}}_0$matrix for time t=0 (undeformed configuration).

$$\boldsymbol{F}|_{t=0} = \boldsymbol{I} \Rightarrow \begin{matrix} \left.\frac{\partial x}{\partial X}\right|_{t=0} = 1\,; & \left.\frac{\partial x}{\partial Y}\right|_{t=0} = 0\,; \\ \left.\frac{\partial y}{\partial X}\right|_{t=0} = 0\,; & \left.\frac{\partial y}{\partial Y}\right|_{t=0} = 1\,; \end{matrix}$$

Therefore,

$$\boldsymbol{B}_0|_{t=0} = \begin{bmatrix} -1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ -1 & -1/2 & 0 & 1/2 & 1 & 0 \end{bmatrix}$$

Thus,

$$f^{int}\big|_{t=0} = \begin{Bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \\ f_{x3} \\ f_{y3} \end{Bmatrix} = \int_{\Omega_0} \boldsymbol{B}_0^T \begin{Bmatrix} S_{11} \\ 0 \\ 0 \end{Bmatrix} d\Omega_0 = \frac{h_0 S_{11}}{2} \begin{Bmatrix} -1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

We conclude that the effect of rotating the body on the nodal internal forces does not change the direction of those forces relative to the body. This is explained by the fact that the PK2 stress is an objective stress. However, the nodal internal forces change in intensity due to the fact that the body is being stretched.

Solution to 4.1c).

If the only nonzero component is S_{22}:

$$\boldsymbol{f}^{int}\big|_{t=1} = \int_{\Omega_0} \boldsymbol{B}_0^T|_{t=1} \begin{Bmatrix} 0 \\ S_{22} \\ 0 \end{Bmatrix} d\Omega_0 = h_0 S_{22} \begin{Bmatrix} b+1 \\ 0 \\ 0 \\ -b-1 \\ 0 \end{Bmatrix}$$

$$\boldsymbol{f}^{int}\big|_{t=0} = \int_{\Omega_0} \boldsymbol{B}_0^T|_{t=0} \begin{Bmatrix} 0 \\ S_{22} \\ 0 \end{Bmatrix} d\Omega_0 = h_0 S_{22} \begin{Bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 1 \end{Bmatrix}$$

If the only nonzero component is S_{12}:

$$\boldsymbol{f}^{int}\big|_{t=1} = \int_{\Omega_0} \boldsymbol{B}_0^T|_{t=1} \begin{Bmatrix} 0 \\ 0 \\ S_{12} \end{Bmatrix} d\Omega_0 = \frac{h_0 S_{12}}{2} \begin{Bmatrix} b+1 \\ -2(a+1) \\ -b-1 \\ 0 \\ 0 \\ a(a+1) \end{Bmatrix}$$

$$\boldsymbol{f}^{int}\big|_{t=0} = \int_{\Omega_0} \boldsymbol{B}_0^T|_{t=0} \begin{Bmatrix} 0 \\ 0 \\ S_{12} \end{Bmatrix} d\Omega_0 = \frac{h_0 S_{12}}{2} \begin{Bmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \\ 0 \end{Bmatrix}$$

We conclude that the same effect is caused on the nodal internal forces by the rotation of the body: the nodal forces follow the rotation, changing the intensity due to the stretching applied to the body.

4.2. Consider the block under shear shown in Figure 3.13 with the motion given in (E3.13.1). Evaluate the Green strain as a function of time. Plot E_{12} and E_{22} for

$t \in [0,4]$**; explain why** E_{22} **is nonzero. Evaluate the PK2 stress for a Kirchhoff material, using a** $[C^{SE}]$ **given by (5.4.58) (the matrix given in (5.4.58) is** $[C^{\tau}]$**, but use the same matrix).**

Solution to 4.2.

The motion of the element is given by

$$x = X + tY, \qquad y = Y$$

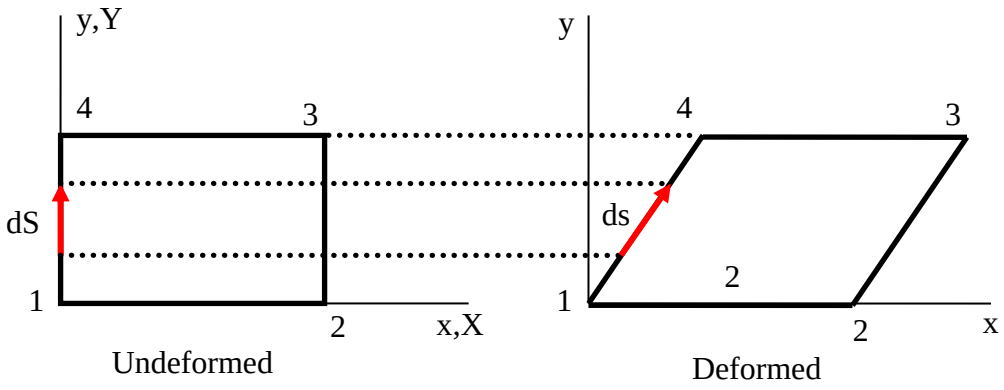
The deformation gradient is simple to calculate:

$$\boldsymbol{F} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$$

The Green strain as a function of time is:

$$\boldsymbol{E} = \frac{1}{2}\left(\boldsymbol{F}^T \cdot \boldsymbol{F} - \boldsymbol{I}\right) = \frac{1}{2}\begin{bmatrix} 0 & t \\ t & t^2 \end{bmatrix}$$

The explanation for the fact that the E_{22} is nonzero is simple to understand by drawing the undeformed and deformed configurations and knowing what the Green strain measures:



Sketch of the deformation of the block under shear.

The motion to which the block is subjected imposes that the y coordinate does not change. On the other hand, since the block is under shear, if one follows what

happens to an infinitesimal line segment in the Y direction in the undeformed configuration and to what happens to the length of that infinitesimal line segment after deformation we see that it increases such that the upper face of the block remains at the same height.

Knowing that the Green strain measures the difference of the square of the length of an infinitesimal segment in the deformed configuration and the undeformed configuration, it is then intuitive to understand that E_{22} is nonzero.

Finally, we need to evaluate the PK2 stress using a tangent modulus tensor $\boldsymbol{C}^{SE}$

$$\boldsymbol{C}^{SE} = \begin{bmatrix} \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda + 2\mu & 0 \\ 0 & 0 & \mu \end{bmatrix}$$

Thus:

$$\begin{Bmatrix} S_{11} \\ S_{22} \\ S_{12} \end{Bmatrix} = \begin{bmatrix} \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda + 2\mu & 0 \\ 0 & 0 & \mu \end{bmatrix} \begin{Bmatrix} E_{11} \\ E_{22} \\ 2E_{12} \end{Bmatrix} = \frac{1}{2} \begin{Bmatrix} \lambda t^2 \\ t^2(2\mu + \lambda) \\ 2t\mu \end{Bmatrix}$$

4.3. (a) Use Nanson's relation (3.4.5)

$$\mathbf{n}\, d\Gamma = J\mathbf{n}_0 \cdot \mathbf{F}^{-1} d\Gamma_0 \qquad n_i\, d\Gamma = J n_j^0 F_{ji}^{-1}\, d\Gamma_0$$

to show that the external nodal forces for an applied pressure *p* **acting on the plane** $\varsigma = -1$ **are given by**

$$\mathbf{f}_I^{\text{ext}} = -\int_{-1}^{1}\int_{-1}^{1} pN_I J_\xi \mathbf{F}_\xi^{-T} \cdot \mathbf{n}_{0\xi}\, d\xi d\eta,$$

where $\mathbf{n}_{0\xi} = -\hat{\mathbf{e}}_3$ **is the unit normal vector in parent coordinates to the plane** $\varsigma = -1$ **in the parent element,**

(b) By using the definition of the inverse of a tensor in terms of Cramer's rule, i.e.

$$\mathbf{F}_{\xi}^{-1} = \frac{1}{\det(\mathbf{F}_{\xi})}\left[\mathbf{F}_{\xi}^{*}\right]$$

where $\mathbf{F}_{\xi}^{*}$ is the adjoint (transpose of the matrix of co-factors) of $\mathbf{F}_{\xi}$, show that the above expression for the external nodal forces reduces to (E4.3.16).

Solution to 4.3a).

The pressure force can be written as:

$$\boldsymbol{t} = -p\boldsymbol{n}$$

With $\boldsymbol{n}$ being the normal vector to the lower face of the element corresponding to the parent element face $\varsigma = -1$, and p being the applied pressure.

Therefore, the external nodal forces are given by:

$$\boldsymbol{f}_I^{ext} = -\int_{\Gamma} pN_I\boldsymbol{n}\, d\boldsymbol{\Gamma}$$

Using Nanson's formula to relate the current normal to the reference normal in the parent domain:

$$\boldsymbol{n}d\Gamma = J_{\xi}\boldsymbol{n}_{0\xi}\cdot\boldsymbol{F}_{\xi}^{-1}d\boldsymbol{\Gamma}_{\xi}$$

We can express the external nodal forces as:

$$\boldsymbol{f}_I^{ext} = -\int_{-1}^{1}\int_{-1}^{1} pN_I J_{\xi}\boldsymbol{n}_{0\xi}\cdot\boldsymbol{F}_{\xi}^{-1}d\xi d\eta = -\int_{-1}^{1}\int_{-1}^{1} pN_I J_{\xi}\boldsymbol{F}_{\xi}^{-T}\cdot\boldsymbol{n}_{0\xi}d\xi d\eta \qquad q.e.d.$$

Solution to 4.3b).

From the definition of the inverse of a tensor in terms of Cramer's rule, we can express the external nodal forces as:

$$\boldsymbol{f}_I^{ext} = -\int_{-1}^{1}\int_{-1}^{1} pN_I J_{\xi}\boldsymbol{n}_{0\xi}\cdot\boldsymbol{F}_{\xi}^{-1}d\xi d\eta = \int_{-1}^{1}\int_{-1}^{1} pN_I\hat{\boldsymbol{e}}_3\cdot\boldsymbol{F}_{\xi}^{*}d\xi d\eta$$

$$= \int_{-1}^{1}\int_{-1}^{1} pN_I\left(\boldsymbol{F}_{\xi}^{*}\right)^{T}\cdot\hat{\boldsymbol{e}}_3 d\xi d\eta$$

Since $J_{\xi} = \det(\boldsymbol{F}_{\xi})$, and $\boldsymbol{n}_{0\xi} = -\hat{\boldsymbol{e}}_3$. The deformation gradient, which is a two-point tensor, is given by:

$$\boldsymbol{F}_{\xi} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial z}{\partial \varsigma} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \varsigma} \\ \frac{\partial z}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \varsigma} \end{bmatrix} = \frac{\partial x_i}{\partial \xi_j} \boldsymbol{e}_i \hat{\boldsymbol{e}}_j$$

where $\boldsymbol{e}_i$ are the unit vectors in the current configuration and $\hat{\boldsymbol{e}}_j$ are the unit vectors in the parent configuration. The transpose of the adjoint matrix, which is the matrix of co-factors is given by:

$$\left(\boldsymbol{F}_{\xi}^{*}\right)^{T} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \varsigma} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \varsigma} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \varsigma} \end{bmatrix} = \begin{bmatrix} \begin{vmatrix} \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \varsigma} \\ \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \varsigma} \end{vmatrix} \boldsymbol{e}_1 \hat{\boldsymbol{e}}_1 & - \begin{vmatrix} \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \varsigma} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \varsigma} \end{vmatrix} \boldsymbol{e}_1 \hat{\boldsymbol{e}}_2 & \begin{vmatrix} \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} \end{vmatrix} \boldsymbol{e}_1 \hat{\boldsymbol{e}}_3 \\ - \begin{vmatrix} \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \varsigma} \\ \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \varsigma} \end{vmatrix} \boldsymbol{e}_2 \hat{\boldsymbol{e}}_1 & \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \varsigma} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \varsigma} \end{vmatrix} \boldsymbol{e}_2 \hat{\boldsymbol{e}}_2 & - \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} \end{vmatrix} \boldsymbol{e}_2 \hat{\boldsymbol{e}}_3 \\ \begin{vmatrix} \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \varsigma} \\ \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \varsigma} \end{vmatrix} \boldsymbol{e}_3 \hat{\boldsymbol{e}}_1 & - \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \varsigma} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \varsigma} \end{vmatrix} \boldsymbol{e}_3 \hat{\boldsymbol{e}}_2 & \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{vmatrix} \boldsymbol{e}_3 \hat{\boldsymbol{e}}_3 \end{bmatrix}$$

Therefore, from the orthonormality of the unit vectors within the same configuration, $\hat{\boldsymbol{e}}_i \cdot \hat{\boldsymbol{e}}_j = \delta_{ij}$, it is clear that the following equality holds:

$$\left(\boldsymbol{F}_{\xi}^{*}\right)^{T} \cdot \hat{\boldsymbol{e}}_3 = \begin{vmatrix} \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \\ \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \eta} \end{vmatrix} \boldsymbol{e}_1 - \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \eta} \end{vmatrix} \boldsymbol{e}_2 + \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix} \boldsymbol{e}_3 = \begin{vmatrix} \boldsymbol{e}_1 & \boldsymbol{e}_2 & \boldsymbol{e}_3 \\ \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \end{vmatrix}$$

Substituting back in the equation obtained for the external nodal forces we get:

$$\boldsymbol{f}_I^{ext} = \int_{-1}^{1}\int_{-1}^{1} p N_I \left(\boldsymbol{F}_{\xi}^{*}\right)^{T} \cdot \hat{\boldsymbol{e}}_3 d\xi d\eta = \int_{-1}^{1}\int_{-1}^{1} p N_I \begin{vmatrix} \boldsymbol{e}_1 & \boldsymbol{e}_2 & \boldsymbol{e}_3 \\ \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \end{vmatrix} d\xi d\eta \qquad q.e.d.$$

4.4. To illustrate the flexibility in choice of reference configuration for the formulation of the finite element equations, consider the tensor quantity, $\mathbf{P}_\xi = J_\xi \mathbf{F}_\xi^{-1} \cdot \boldsymbol{\sigma}$, which can be thought of as the nominal stress tensor on the parent element domain. Show that the equilibrium equation and boundary conditions can be written as

$\nabla_{\xi .} \mathbf{P}_\xi = 0$ **in** $\cup \Delta_e$ **(union of parent element domains)**

$\mathbf{n}_{0\xi .} \mathbf{P}_\xi = \mathrm{t}_\xi$ **on** Γ_t **(traction boundary)**

and derive the corresponding weak form. Introduce parent element shape functions $N_I(\xi)$ **and show that the element internal force vector can be written directly in terms of the parent element domain as**

$$f_{iI}^{int} = \int_{\square} P_{ji}^{\xi} \frac{\partial N_I}{\partial \xi_j} d\square \qquad q.e.d.$$

Solution to 4.4.

The given stress quantity $\boldsymbol{P}_\xi$ can be expressed in terms of the Cauchy stress as:

$$\sigma_{ij} = J_\xi^{-1} F_{im}^{\xi} P_{mj}^{\xi}$$

The equilibrium equation in the absence of body forces is given by:

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} = 0 \quad \Rightarrow \quad \frac{\partial \sigma_{ij}}{\partial x_i} = 0$$

Using the result obtained for the Cauchy stress in terms of $\boldsymbol{P}_\xi$, and substituting in the above equation:

$$\frac{\partial \left(J_\xi^{-1} F_{im}^{\xi} P_{mj}^{\xi} \right)}{\partial x_i} = 0$$

$$\frac{\partial \left(J_\xi^{-1} \right)}{\partial x_i} F_{im}^{\xi} P_{mj}^{\xi} + J_\xi^{-1} \frac{\partial \left(F_{im}^{\xi} \right)}{\partial x_i} P_{mj}^{\xi} + J_\xi^{-1} F_{im}^{\xi} \frac{\partial \left(P_{mj}^{\xi} \right)}{\partial x_i} = 0$$

Evaluating each term separately, starting with the first derivative and using the definition of determinant of a tensor in index notation:

$$\frac{\partial\left(J_\xi^{-1}\right)}{\partial x_i} = -J_\xi^{-2}\frac{\partial\left(J_\xi\right)}{\partial x_i}$$

$$= -J_\xi^{-2}\frac{\partial\left(e_{jkl}F_{1j}^{\xi}F_{2k}^{\xi}F_{3l}^{\xi}\right)}{\partial x_i}$$

$$= -J_\xi^{-2}\frac{\partial\left(e_{jkl}\frac{\partial x_1}{\partial\xi_j}\frac{\partial x_2}{\partial\xi_k}\frac{\partial x_3}{\partial\xi_l}\right)}{\partial x_i}$$

$$= -J_\xi^{-2}e_{jkl}\left[\frac{\partial\left(\frac{\partial x_1}{\partial\xi_j}\right)}{\partial x_i}\frac{\partial x_2}{\partial\xi_k}\frac{\partial x_3}{\partial\xi_l} + \frac{\partial x_1}{\partial\xi_j}\frac{\partial\left(\frac{\partial x_2}{\partial\xi_k}\right)}{\partial x_i}\frac{\partial x_3}{\partial\xi_l} + \frac{\partial x_1}{\partial\xi_j}\frac{\partial x_2}{\partial\xi_k}\frac{\partial\left(\frac{\partial x_3}{\partial\xi_l}\right)}{\partial x_i}\right]$$

$$= -J_\xi^{-2}e_{jkl}\left[\frac{\partial\left(\frac{\partial x_1}{\partial x_i}\right)}{\partial\xi_j}\frac{\partial x_2}{\partial\xi_k}\frac{\partial x_3}{\partial\xi_l} + \frac{\partial x_1}{\partial\xi_j}\frac{\partial\left(\frac{\partial x_2}{\partial x_i}\right)}{\partial\xi_k}\frac{\partial x_3}{\partial\xi_l} + \frac{\partial x_1}{\partial\xi_j}\frac{\partial x_2}{\partial\xi_k}\frac{\partial\left(\frac{\partial x_3}{\partial x_i}\right)}{\partial\xi_l}\right] = 0$$

Since $\frac{\partial x_n}{\partial x_i} = \delta_{nj}$.

For the same reason,

$$\frac{\partial\left(F_{im}^{\xi}\right)}{\partial x_i} = 0$$

Thus,

$$\frac{\partial\left(J_\xi^{-1}F_{im}^{\xi}P_{mj}^{\xi}\right)}{\partial x_i} = 0 \Rightarrow J_\xi^{-1}F_{im}^{\xi}\frac{\partial\left(P_{mj}^{\xi}\right)}{\partial x_i} = 0$$

$$F_{im}^{\xi}\frac{\partial\left(P_{mj}^{\xi}\right)}{\partial x_i} = 0$$

$$\frac{\partial x_i}{\partial\xi_m}\frac{\partial\left(P_{mj}^{\xi}\right)}{\partial x_i} = 0$$

$$\frac{\partial\left(P_{mj}^{\xi}\right)}{\partial\xi_m} = 0 \Rightarrow \boldsymbol{\nabla_\xi}\cdot\boldsymbol{P}_\xi = 0 \quad \text{in } \bigcup\Delta_e \qquad q.e.d.$$

The boundary conditions can be obtained by converting:

$$d\boldsymbol{f} = \boldsymbol{\sigma}\cdot\boldsymbol{n}d\boldsymbol{\Gamma} = \mathbf{t}\mathrm{d}\boldsymbol{\Gamma} \quad \text{on} \quad \boldsymbol{\Gamma}_\mathrm{t}$$

Using Nanson's relation, $\boldsymbol{n}d\Gamma = J_\xi \boldsymbol{n}_{0\xi} \cdot \boldsymbol{F}_\xi^{-1} d\boldsymbol{\Gamma}_\xi$, and using the definition for $\boldsymbol{P}_\xi$ given in the problem statement. Starting with the LHS of the above equation:

$$d\boldsymbol{f} = \boldsymbol{\sigma} \cdot \boldsymbol{n}d\boldsymbol{\Gamma} = J_\xi^{-1} \boldsymbol{F}_\xi \cdot \boldsymbol{P}_\xi \cdot \boldsymbol{n}d\boldsymbol{\Gamma}$$
$$= J_\xi^{-1} \boldsymbol{F}_\xi \cdot \boldsymbol{P}_\xi \cdot \boldsymbol{n}_{0\xi} \cdot \boldsymbol{F}_\xi^{-1} J_\xi d\boldsymbol{\Gamma}_\xi$$
$$= \boldsymbol{F}_\xi \cdot \boldsymbol{P}_\xi \cdot \boldsymbol{n}_{0\xi} \cdot \boldsymbol{F}_\xi^{-1} d\boldsymbol{\Gamma}_\xi$$

From conservation of angular momentum $\boldsymbol{F}_\xi \cdot \boldsymbol{P}_\xi = \boldsymbol{P}_\xi^T \cdot \boldsymbol{F}_\xi^T$, so:

$$d\boldsymbol{f} = \boldsymbol{P}_\xi^T \cdot \boldsymbol{F}_\xi^T \cdot \boldsymbol{F}_\xi^{-T} \cdot \boldsymbol{n}_{0\xi} d\boldsymbol{\Gamma}_\xi = \boldsymbol{P}_\xi^T \cdot \boldsymbol{n}_{0\xi} d\boldsymbol{\Gamma}_\xi$$

Since,

$$d\boldsymbol{f} = \mathbf{t}\mathrm{d}\boldsymbol{\Gamma} = \mathbf{t}_\xi \mathrm{d}\boldsymbol{\Gamma}_\xi$$

We finally obtain:

$$\boldsymbol{P}_\xi^T \cdot \boldsymbol{n}_{0\xi} = \mathbf{t}_\xi \quad \text{on} \quad \boldsymbol{\Gamma}_\mathrm{t}^\xi \qquad\qquad q.e.d.$$

Deriving the weak form is then trivial, leading to:

$$\delta W_{int} - \delta W_{ext} + \delta W_{kin} = 0$$
$$\delta W_{int} = \int_{\Omega_\xi} \delta \boldsymbol{F}_\xi^T \cdot \boldsymbol{P}_\xi d\Omega = \int_{\Omega_\xi} \left(\delta F_{ij}^\xi\right)^T P_{ji}^\xi d\Omega$$
$$\delta W_{ext} = \int_{\Gamma_\xi} \delta u_i t_i^\xi d\Gamma$$
$$\delta W_{kin} = \int_{\Omega_\xi} \delta u_i \rho_\xi \ddot{u}_i d\Omega$$

The internal nodal forces can then be obtained by discretizing the expression for the internal energy:

$$\delta W_{int} = \int_{\Omega_\xi} \left(\delta F_{ij}^\xi\right)^T P_{ji}^\xi d\Omega$$

$$\delta u_{iI} f_{iI}^{int} = \delta u_{iI} \int_{\Omega_\xi} \frac{\partial N_I}{\partial \xi_j}^T P_{ji}^{\xi} d\Omega$$

Denoting the parent domain as the problem statement:

$$f_{iI}^{int} = \int_{\square} P_{ji}^{\xi} \frac{\partial N_I}{\partial \xi_j} d\square \qquad q.e.d.$$

CHAPTER 5: Constitutive models

5.1. Show that if p is the pressure, the relations $3Jp = \boldsymbol{\tau}:\mathbf{g} = \bar{\mathbf{S}}:\bar{\mathbf{C}}^e$ hold.

Solution to 5.1.

From Equation (5.7.4)

$$\mathbf{S} = \left(\mathbf{F}^e\right)^{-1} \cdot \boldsymbol{\tau} \cdot \left(\mathbf{F}^e\right)^{-T}$$

From Equation (5.7.3)

$$\overline{\mathbf{C}^e} = \mathbf{F}^{eT} \cdot \mathbf{g} \cdot \mathbf{F}^e$$

Therefore,

$$\bar{\mathbf{S}} : \bar{\mathbf{C}}^e = \left(\mathbf{F}^e\right)^{-1} \cdot \boldsymbol{\tau} \cdot \left(\mathbf{F}^e\right)^{-T} : \mathbf{F}^{eT} \cdot \mathbf{g} \cdot \mathbf{F}^e = \boldsymbol{\tau} : \mathbf{g}$$

In Euclidean space $\mathbf{g} = \mathbf{I}$; therefore,

$$\boldsymbol{\tau} : \mathbf{g} = \boldsymbol{\tau} : \mathbf{I} = J\boldsymbol{\sigma} : \mathbf{I} = J(\sigma_{11} + \sigma_{22} + \sigma_{33}) = 3Jp \quad \text{q.e.d.}$$

where, $p = (\sigma_{11} + \sigma_{22} + \sigma_{33})/3$

5.2. Show that $(\text{sym}\ \bar{\mathbf{r}}):\left(\bar{\mathbf{C}}^e\right)^{-1} = 0$ **and hence that** $\bar{\mathbf{S}}^{\text{dev}}:\bar{\mathbf{D}}^p = \bar{\mathbf{S}}:\bar{\mathbf{D}}^p$. **See (5.7.39) and (5.7.40).**

Solution to 5.2.

From (5.7.40),

$$\bar{\mathbf{D}}^p = \dot{\lambda}\ \text{sym}\ \bar{\mathbf{r}} = \dot{\lambda}\frac{3}{2\bar{\sigma}}\bar{\mathbf{C}}^e \cdot \bar{\mathbf{S}}^{dev} \cdot \bar{\mathbf{C}}^e$$

Therefore,

$$\text{sym}\ \bar{\mathbf{r}} = \frac{3}{2\bar{\sigma}}\bar{\mathbf{C}}^e \cdot \bar{\mathbf{S}}^{dev} \cdot \bar{\mathbf{C}}^e$$

And

$$\text{sym}\,\bar{\mathbf{r}} : \left(\bar{\mathbf{C}}^{e}\right)^{-1} = \frac{3}{2\sigma}\bar{\mathbf{C}}^{e} \cdot \bar{\mathbf{S}}^{dev} \cdot \bar{\mathbf{C}}^{e} : \left(\bar{\mathbf{C}}^{e}\right)^{-1} = \frac{3}{2\sigma}\bar{\mathbf{C}}^{e} \cdot \bar{\mathbf{S}}^{dev} : \mathbf{I}$$

Since the trace of a deviatoric tensor is zero

$$\text{sym}\,\bar{\mathbf{r}} : \left(\bar{\mathbf{C}}^{e}\right)^{-1} = \frac{3}{2\sigma}\bar{\mathbf{C}}^{e} \cdot \bar{\mathbf{S}}^{dev} : \mathbf{I} = 0 \qquad \text{q.e.d.}$$

5.3. Derive expressions for the Lie derivatives $L_v\boldsymbol{\tau}^{\text{dev}}$ and $L_v\boldsymbol{\tau}^{\text{hyd}}$ in terms of the material time derivative of the stress and the spatial velocity gradient L.

Solution to 5.3.

From equation (5.10.5), we know that

$$L_v\boldsymbol{\tau} = \dot{\boldsymbol{\tau}} - \mathbf{L}\cdot\boldsymbol{\tau} - \boldsymbol{\tau}\cdot\mathbf{L}^{T}$$

Since $\boldsymbol{\tau}^{dev} = \boldsymbol{\tau} - \boldsymbol{\tau}^{hyd} = \boldsymbol{\tau} - Jp\mathbf{I}$, and noting how equation (5.10.5) was derived, we can calculate the Lie derivative of the hydrostatic part of the Kirchhoff stress by replacing $\boldsymbol{\tau}$ by $\boldsymbol{\tau}^{hyd} = Jp\mathbf{I}$:

$$L_v\boldsymbol{\tau}^{hyd} = Jp\mathbf{I}\nabla\cdot\mathbf{v} - Jp\left(\mathbf{L}+\mathbf{L}^{T}\right) = Jp\mathbf{I}\ \text{trace}\left(\mathbf{L}\right) - Jp\left(\mathbf{L}+\mathbf{L}^{T}\right),$$

Since $\dfrac{D\left(Jp\mathbf{I}\right)}{Dt} = Jp\mathbf{I}\nabla\cdot\mathbf{v} = Jp\mathbf{I}\text{trace}\left(\mathbf{L}\right)$.

Therefore, we can now determine the Lie derivative of the deviatoric part of the Kirchhoff stress:

$$L_v\boldsymbol{\tau}^{dev} = L_v\boldsymbol{\tau} - L_v\boldsymbol{\tau}^{hyd} = \dot{\boldsymbol{\tau}} - \mathbf{L}\cdot\boldsymbol{\tau} - \boldsymbol{\tau}\cdot\mathbf{L}^{T} - Jp\mathbf{I}\ \text{trace}\left(\mathbf{L}\right) + Jp\left(\mathbf{L}+\mathbf{L}^{T}\right)$$

Note that we could simplify the last term by using $\mathbf{D} = \frac{1}{2}\left(\mathbf{L}+\mathbf{L}^{T}\right)$.

CHAPTER 6: Solution methods and stability

6.1. Use Nanson's law (3.4.5) and the result obtained in exercise 4 of Chapter 3 for the material time derivative of a surface integral,

$$\frac{d}{dt}\int_S g\mathbf{n}\,dS = \int_S \left[(\dot{g} + g\nabla\cdot\mathbf{v})\mathbf{I} - g\mathbf{L}^T\right]\cdot\mathbf{n}\,dS\,,$$

to develop a linearization of the load stiffness $\mathbf{K}^{ext} = \partial\mathbf{f}^{ext}/\partial\mathbf{d}$ for a pressure load applied on a surface mapped from the biunit square in the parent element.

Solution to 6.1.

The external nodal forces are given by

$$\mathbf{f}_I^{ext} = -\int_\Gamma N_I\, p\mathbf{n}d\Gamma$$

Using the relation derived in exercise 4.3,

$$\frac{d}{dt}\int_S g\mathbf{n}\,dS = \int_S \left[(\dot{g} + g\nabla\cdot\mathbf{v})\mathbf{I} - g\mathbf{L}^T\right]\cdot\mathbf{n}\,dS$$

We obtain the following expression for the time derivative of the external nodal forces:

$$\dot{\mathbf{f}}_I^{ext} = -\int_\Gamma N_I\left[(\dot{p} + p\nabla\cdot\mathbf{v})\mathbf{I} - p\mathbf{L}^T\right]\cdot\mathbf{n}\,d\Gamma$$

Similarly to what was done in the textbook in equation (6.4.29), we omit the term with the rate of change of the pressure ($\dot{p}\approx 0$). Thus, the external load stiffness is given by:

$$\dot{\mathbf{f}}_I^{ext} = \mathbf{K}_{IJ}^{ext}\mathbf{v}_J = -\int_\Gamma pN_I\left[\left(\nabla\cdot\mathbf{v}\right)\mathbf{I} - \mathbf{L}^T\right]\cdot\mathbf{n}\,d\Gamma$$

Rewriting the integral with respect to the parent domain using $\mathbf{n}\,d\Gamma = \mathbf{x}_{,\xi}\times\mathbf{x}_{,\eta}d\xi d\eta$

$$\dot{\mathbf{f}}_I^{ext} = \mathbf{K}_{IJ}^{ext}\mathbf{v}_J = -\int_{-1}^{1}\int_{-1}^{1} pN_I\left[\left(\nabla\cdot\mathbf{v}\right)\mathbf{I} - \mathbf{L}^T\right]\cdot\left(\mathbf{x}_{,\xi}\times\mathbf{x}_{,\eta}\right)d\xi d\eta$$

Next, we switch to index notation and use $v_{i,k} = v_{iJ} N_{J,k}$

$$K^{ext}_{ikIJ} v_{kJ} = -\int_{-1}^{1}\int_{-1}^{1} pN_I \left(v_{nJ} N_{J,n} \delta_{ik} - v_{kJ} N_{J,i} \right) e_{klm} x_{l,\xi} x_{m,\eta} d\xi d\eta$$

Noting that $N_{J,p} = \frac{\partial N_J}{\partial \xi_q}\frac{\partial \xi_q}{\partial x_p}$ and $\frac{\partial \xi_q}{\partial x_p}\frac{\partial x_r}{\partial \xi_q} = \delta_{pr}$, and evaluating the first term of the integral separately:

$$v_{nJ} N_{J,n} \delta_{ik} e_{klm} x_{l,\xi} x_{m,\eta} = v_{nJ} \frac{\partial N_J}{\partial \xi_q}\frac{\partial \xi_q}{\partial x_n} e_{ilm} \frac{\partial x_l}{\partial \xi}\frac{\partial x_m}{\partial \eta}$$

$$= e_{ilm} v_{nJ} \frac{\partial N_J}{\partial \xi}\frac{\partial \xi}{\partial x_n}\frac{\partial x_l}{\partial \xi}\frac{\partial x_m}{\partial \eta} + e_{ilm} v_{nJ} \frac{\partial N_J}{\partial \eta}\frac{\partial \eta}{\partial x_n}\frac{\partial x_l}{\partial \xi}\frac{\partial x_m}{\partial \eta}$$

$$= v_{nJ} e_{ilm} \frac{\partial N_J}{\partial \xi}\delta_{nl}\frac{\partial x_m}{\partial \eta} + v_{nJ} e_{ilm} \frac{\partial N_J}{\partial \eta}\frac{\partial x_l}{\partial \xi}\delta_{nm}$$

$$= e_{inm} v_{nJ} N_{J,\xi} x_{m,\eta} + e_{i\ln} v_{nJ} N_{J,\eta} x_{l,\xi}$$

$$= e_{inm} v_{n,\xi} x_{m,\eta} + e_{iln} x_{l,\xi} v_{n,\eta}$$

Now, evaluating the second term using the same procedure:

$$v_{kJ} N^T_{J,i} e_{klm} x_{l,\xi} x_{m,\eta} = v_{kJ} \frac{\partial N^T_J}{\partial \xi_q}\frac{\partial \xi_q}{\partial x_i} e_{klm} \frac{\partial x_l}{\partial \xi}\frac{\partial x_m}{\partial \eta}$$

$$= e_{klm} v_{kJ} \frac{\partial N_J}{\partial \xi}\frac{\partial \xi}{\partial x_i}\frac{\partial x_l}{\partial \xi}\frac{\partial x_m}{\partial \eta} + e_{klm} v_{kJ} \frac{\partial N_J}{\partial \eta}\frac{\partial \eta}{\partial x_i}\frac{\partial x_l}{\partial \xi}\frac{\partial x_m}{\partial \eta}$$

$$= v_{kJ} e_{klm} \frac{\partial N_J}{\partial \xi}\delta_{il}\frac{\partial x_m}{\partial \eta} + v_{kJ} e_{klm} \frac{\partial N_J}{\partial \eta}\frac{\partial x_l}{\partial \xi}\delta_{im}$$

$$= e_{kim} v_{kJ} N_{J,\xi} x_{m,\eta} + e_{kli} v_{kJ} N_{J,\eta} x_{l,\xi} = -e_{ikm} v_{k,\xi} x_{m,\eta} - e_{ilk} x_{l,\xi} v_{k,\eta}$$

Therefore, we conclude that

$$K^{ext}_{ikIJ} v_{kJ} = -2\int_{-1}^{1}\int_{-1}^{1} pN_I \left(e_{inm} v_{n,\xi} x_{m,\eta} + e_{iln} x_{l,\xi} v_{n,\eta} \right) d\xi d\eta$$

Which is the same result as equation (6.4.30) in the textbook, leading to equation (6.4.33) as shown there.

6.2. Show that (6.3.60) corresponds to the stationary points of (6.3.59).

Solution to 6.2.

Equation (6.3.60) is:

$$W_{PL}(\mathbf{d},\boldsymbol{\lambda},\varepsilon) = W(\mathbf{d}) + \boldsymbol{\lambda}^T g(\mathbf{d}) - \frac{1}{2}\varepsilon\boldsymbol{\lambda}^T\boldsymbol{\lambda}$$

To examine the stationary points we will differentiate the above with respect to **d** and **λ** separately and equate them to zero about an equilibrium point d_a as below:

$$\frac{\partial W_{PL}}{\partial d_a} = \frac{\partial W}{\partial d_a} + \lambda_I \frac{\partial g_I}{\partial d_a} = 0$$

$$\frac{\partial W_{PL}}{\partial \lambda_I} = g_I - \varepsilon\lambda_I = 0$$

Rewriting the above two equations in terms of the residual **r** and the gradient of the matrix **G** yields:

$$\mathbf{r} + \boldsymbol{\lambda}^T\mathbf{G} = \mathbf{0}$$

$$\mathbf{g} - \varepsilon\boldsymbol{\lambda} = \mathbf{0}$$

6.3. Show that (6.3.61), the linearized perturbed Lagrangian equations, can be converted to the linearized penalty equations by eliminating the Lagrange multipliers.

Solution to 6.3.

The linearized perturbed Lagrangian equations are:

$$\begin{bmatrix} \mathbf{A}+\lambda_I \mathbf{H}_I & \mathbf{G}^T \\ \mathbf{G} & -\varepsilon \mathbf{I} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{d} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\left(\mathbf{r}+\boldsymbol{\lambda}^T \mathbf{G}\right) \\ -\left(\mathbf{g}-\varepsilon \boldsymbol{\lambda}\right) \end{bmatrix}$$

Writing the equations separately:

$$\left(\mathbf{A}+\lambda_I \mathbf{H}_I\right)\Delta \mathbf{d} + \mathbf{G}^T \Delta \boldsymbol{\lambda} = -\mathbf{r} - \boldsymbol{\lambda}^T \mathbf{G} \tag{1}$$

$$\mathbf{G}\Delta \mathbf{d} + \varepsilon \mathbf{I}\Delta \boldsymbol{\lambda} = -\left(\mathbf{g}-\varepsilon \boldsymbol{\lambda}\right) \tag{2}$$

Equation (6.3.47) derived for the penalty method is:

$$\mathbf{r} + \beta \mathbf{g}^T \mathbf{G} = \mathbf{0} \tag{3}$$

Using the results obtained in exercise 2:

$$\mathbf{r} + \boldsymbol{\lambda}^T \mathbf{G} = \mathbf{0} \tag{4}$$

$$\mathbf{g} - \varepsilon \boldsymbol{\lambda} = \mathbf{0} \tag{5}$$

From equations (3) and (4):

$$\boldsymbol{\lambda}^T \mathbf{G} = \beta \mathbf{g}^T \mathbf{G} \tag{6}$$

Substituting equation (6) into (1):

$$\left(\mathbf{A} + \beta g_I \mathbf{H}_I \right) \Delta \mathbf{d} + \mathbf{G}^T \Delta \boldsymbol{\lambda} = -\mathbf{r} - \beta \mathbf{g}^T \mathbf{G}$$

Also, from equation (5) we can write equation (2) as:

$$\mathbf{G} \Delta \mathbf{d} - \varepsilon \mathbf{I} \Delta \boldsymbol{\lambda} = \mathbf{0}$$

From which we obtain

$$\Delta \boldsymbol{\lambda} = \left(\varepsilon \mathbf{I} \right)^{-1} \mathbf{G} \Delta \mathbf{d} = \frac{1}{\varepsilon} \mathbf{G} \Delta \mathbf{d} \tag{7}$$

Substituting equation (7) into (6)

$$\left(\mathbf{A} + \beta g_I \mathbf{H}_I + \frac{1}{\varepsilon} \mathbf{G}^T \mathbf{G} \right) \Delta \mathbf{d} = -\mathbf{r} - \beta \mathbf{g}^T \mathbf{G}$$

Which is identical to the linearized penalty equation (6.3.49) since ε is a very small number and β is a very large number.

6.4. Obtain (6.4.20) by letting the reference configuration in (6.4.4) be the current configuration.

Solution to 6.4.

Equation (6.4.4) is:

$$\dot{f}_{iI}^{\text{int}} = \int_{\Omega_0} \frac{\partial N_I}{\partial X_j}\left(\dot{S}_{jr}F_{ir} + S_{jr}\dot{F}_{ir}\right)d\Omega_0$$

In order to obtain (6.4.20) let's evaluate each term of the above equation separately. Starting with the first term, $\int_{\Omega_0} \frac{\partial N_I}{\partial X_j}\dot{S}_{jr}F_{ir}d\Omega_0$, if we let the reference configuration be the current configuration (i.e. if $\mathbf{x}$ coincides with $\mathbf{X}$) then the following equalities hold:

$$\frac{\partial N_I}{\partial X_j} \to \frac{\partial N_I}{\partial x_j} \;;\; \dot{S}_{jr} \to \sigma_{jr}^{\Delta\tau} \;;\; F_{ir} \to \delta_{ir} \;;\; \Omega_0 \to \Omega$$

Where the second relation in the above is expressed in equation (3.7.19). Therefore, we have obtained the first term of equation (6.4.20):

$$\int_{\Omega_0} \frac{\partial N_I}{\partial X_j}\dot{S}_{jr}F_{ir}d\Omega_0 = \int_{\Omega} \frac{\partial N_I}{\partial x_j}\sigma_{ji}^{\Delta\tau}d\Omega$$

To obtain the second term of equation (6.4.20), $\int_{\Omega_0} \frac{\partial N_I}{\partial X_j}S_{jr}\dot{F}_{ir}d\Omega_0$, we first convert the integral to the current domain by writing:

$$\int_{\Omega_0} \frac{\partial N_I}{\partial X_j}S_{jr}\dot{F}_{ir}d\Omega_0 = \int_{\Omega} \frac{\partial N_I}{\partial X_j}S_{jr}\dot{F}_{ir}J^{-1}d\Omega \equiv \int_{\Omega} \frac{\partial N_I}{\partial \mathbf{X}}\mathbf{S}\dot{\mathbf{F}}^T J^{-1}d\Omega$$

Where in the last equality we have converted to tensor notation for convenience. Then we can substitute in the above expression the definition of PK2 stress in terms of the Cauchy stress, $\mathbf{S} = J^{-1}\mathbf{F}^{-1}\cdot\boldsymbol{\sigma}\cdot\mathbf{F}^{-T}$, and the relation $\dot{\mathbf{F}} = \mathbf{L}\mathbf{F}$, resulting in:

$$\int_\Omega \frac{\partial N_I}{\partial \mathbf{X}} \cdot \mathbf{S} \cdot \dot{\mathbf{F}}^T J^{-1} d\Omega = \int_\Omega \frac{\partial N_I}{\partial \mathbf{X}} \cdot \left(J\mathbf{F}^{-1} \cdot \boldsymbol{\sigma} \cdot \mathbf{F}^{-T} \right) \cdot \mathbf{F}^T \cdot \mathbf{L}^T J^{-1} d\Omega$$

$$= \int_\Omega \frac{\partial N_I}{\partial \mathbf{x}} \cdot \boldsymbol{\sigma} \cdot \mathbf{L}^T d\Omega \equiv \int_\Omega \frac{\partial N_I}{\partial x_j} \sigma_{jr} L_{ir} d\Omega$$

Hence, we have obtained equation (6.4.20):

$$\dot{f}_{iI}^{\text{int}} = \int_\Omega \frac{\partial N_I}{\partial x_j} \left(\sigma_{jr}{}^{\Delta\tau} + \sigma_{jr} L_{ir} \right) d\Omega$$

6.5. Show that the critical time steps given by the updated and total Lagrangian formulations in (6.6.61) and (6.6.63) are identical. Use the relations between tangent moduli in Example 5.1 for uniaxial strain.

Solution to 6.5.

This can be proven by showing that the relations in equations (6.6.59) and (6.6.62) are equivalent.

The relation in equation (6.6.59) is

$$\frac{A\left(C^{\sigma\tau} + \sigma_{xx}\right)}{l} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \frac{\lambda \rho A l}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

and the relation in equation (6.6.62) is

$$\frac{A_0\left(F^2 C^{SE} + S_{11}\right)}{l_0} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \frac{\lambda \rho_0 A_0 l_0}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

The relationship between the tangent moduli for uniaxial strain is:

$$C^{SE} = J\lambda^{-4}C^{\sigma\tau}$$

Since $J = \frac{Al}{A_0 l_0}$ and $\lambda = \frac{l}{l_0}$, then:

$$C^{SE} = \frac{Al_0^3}{A_0 l^3} C^{\sigma\tau}$$

Also, using the definition of PK2 stress, $S_{11} = JF^{-1}\sigma_{xx}F^{-T}$, and since a uniaxial strain problem is being considered $F = \frac{l}{l_0}$:

$$S_{11} = \frac{Al_0}{A_0 l}\sigma_{xx}$$

Therefore, equation (6.6.62) can be re-written as:

$$\frac{A_0\left(F^2 C^{SE} + S_{11}\right)}{l_0}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \frac{\lambda\rho_0 A_0 l_0}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

$$\frac{A_0\left(\frac{Al_0}{A_0 l}C^{\sigma\tau} + \frac{Al_0}{A_0 l}\sigma_{xx}\right)}{l_0}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \frac{\lambda\rho_0 A_0 l_0}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

$$\frac{A\left(C^{\sigma\tau} + \sigma_{xx}\right)}{l}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \frac{\lambda\rho_0 A_0 l_0}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix}$$

Assuming conservation of mass i.e., $\rho Al = \rho_0 A_0 l_0$, we reach to the desired equivalence:

$$\frac{A\left(C^{\sigma\tau}+\sigma_{xx}\right)}{l}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\begin{Bmatrix}y_1\\ y_2\end{Bmatrix}=\frac{\lambda\rho Al}{2}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{Bmatrix}y_1\\ y_2\end{Bmatrix}$$

6.6. Develop the tangent stiffness for an axisymmetric 2-node membrane element.

Solution to 6.6.

Assuming the material is elastic isotropy with Young's modulus E and Poisson's ratio ν

$$\begin{Bmatrix}\sigma_r\\ \sigma_z\\ \sigma_\theta\\ \tau_{rz}\end{Bmatrix}=\frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix}1-\nu & \nu & \nu & 0\\ \nu & 1-\nu & \nu & 0\\ \nu & \nu & 1-\nu & 0\\ 0 & 0 & 0 & (1-2\nu)/2\end{bmatrix}\begin{Bmatrix}\varepsilon_r\\ \varepsilon_z\\ \varepsilon_\theta\\ \gamma_{rz}\end{Bmatrix}$$

which can be represented as $\boldsymbol{\sigma}=\mathbf{D}:\boldsymbol{\varepsilon}$.

$$\mathbf{K}^{mat}=\int_\Omega \mathbf{B}_I^T\mathbf{D}\mathbf{B}_J\,d\Omega$$

$$\mathbf{K}^{geo}=\int_\Omega \hat{\mathbf{B}}_I^T\boldsymbol{\sigma}\hat{\mathbf{B}}_J\,d\Omega$$

Following the example 2.6, we have the 2D membrane with thickness a. Since the radius of membrane is much larger than its thickness, we only consider the plane stress condition as $\sigma_z=0$ and shear stress free $\tau_{rz}=0$. From example 2.6, we have $\mathbf{N}=\begin{bmatrix}N_1 & N_2\end{bmatrix}=\begin{bmatrix}1-\xi & \xi\end{bmatrix}$

$$\mathbf{B}_1 = \begin{bmatrix} \dfrac{\partial N_1}{\partial r} & 0 \\ 0 & \dfrac{\partial N_1}{\partial z} \\ \dfrac{N_1}{r} & 0 \\ \dfrac{\partial N_1}{\partial z} & \dfrac{\partial N_1}{\partial r} \end{bmatrix} = \begin{bmatrix} -\dfrac{1}{r_{21}} & 0 \\ 0 & 0 \\ \dfrac{1-\xi}{r} & 0 \\ 0 & -\dfrac{1}{r_{21}} \end{bmatrix} \text{ and } \mathbf{B}_2 = \begin{bmatrix} \dfrac{\partial N_2}{\partial r} & 0 \\ 0 & \dfrac{\partial N_2}{\partial z} \\ \dfrac{N_2}{r} & 0 \\ \dfrac{\partial N_2}{\partial z} & \dfrac{\partial N_2}{\partial r} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{r_{21}} & 0 \\ 0 & 0 \\ \dfrac{\xi}{r} & 0 \\ 0 & \dfrac{1}{r_{21}} \end{bmatrix}$$

Thus, $$\mathbf{B} = [\mathbf{B}_1 \quad \mathbf{B}_2] = \begin{bmatrix} -\dfrac{1}{r_{21}} & 0 & \dfrac{1}{r_{21}} & 0 \\ 0 & 0 & 0 & 0 \\ \dfrac{1-\xi}{r} & 0 & \dfrac{\xi}{r} & 0 \\ 0 & -\dfrac{1}{r_{21}} & 0 & \dfrac{1}{r_{21}} \end{bmatrix}$$

where $r_{21} = r_2 - r_1$, $\xi = \dfrac{r - r_1}{r_{21}}$ and ar is the area of one radian segment.

$$K^{mat} = \int_{\Omega} \mathbf{B}^T \mathbf{D} \mathbf{B} ar dr$$

From Eq. (E4.6.4), we have $\hat{\mathbf{B}} = \mathbf{B}$, thus

$$K^{geo} = \int_{\Omega} \mathbf{B}^T \boldsymbol{\sigma} \mathbf{B} ar dr$$

Here the integrals are taken from r_2 to r_1 and $K^{\text{int}} = K^{mat} + K^{geo}$.

6.7. Examine the stability of a solution of the two-dimensional heat conduction equation $(k_{ij}\theta_{,j})_{,i} = 0$ in the following way. Consider an infinite slab under a uniform temperature and apply the perturbation $\tilde{\theta} = e^{\omega t + i\kappa \mathbf{n}\cdot\mathbf{x}}$ where κ is real. Using the transient equations of heat conduction, determine the conditions under which the solution is stable if k_{ij} is symmetric.

Solution to 6.7.

The heat conduction equation in 2D can be written as

$$f(\theta) = \frac{\partial}{\partial x_i}\left(k_{ij}\frac{\partial \theta}{\partial x_j}\right) = 0$$

The Taylor series expansion about the uniform temperature θ_0 gives

$$f(\theta_0 + \tilde{\theta}) = f(\theta_0) + \frac{\partial f(\tilde{\theta})}{\partial \theta}\tilde{\theta} + \text{higher order terms}$$

The first term vanishes because the temperature is uniform, and if we neglect the higher order terms we can write:

$$f(\theta_0 + \tilde{\theta}) = \frac{\partial f(\tilde{\theta})}{\partial \theta}\tilde{\theta} = \frac{\partial}{\partial \theta}\left[\frac{\partial}{\partial x_i}\left(k_{ij}\frac{\partial \theta}{\partial x_j}\right)\right]\tilde{\theta} = \frac{\partial}{\partial x_i}\left[\frac{\partial}{\partial \theta}\left(k_{ij}\frac{\partial \theta}{\partial x_j}\right)\right]\tilde{\theta}$$

$$= \frac{\partial}{\partial x_i}\left(\frac{\partial k_{ij}}{\partial \theta}\frac{\partial \theta}{\partial x_j} + k_{ij}\frac{\partial}{\partial \theta}\frac{\partial \theta}{\partial x_j}\right)\tilde{\theta}$$

The second term vanishes, therefore we obtain:

$$f(\theta_0 + \tilde{\theta}) = \frac{\partial^2 k_{ij}}{\partial x_i \partial x_j}\tilde{\theta}$$

Since the transient heat conduction equation is:

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial x_i}\left(k_{ij}\frac{\partial \theta}{\partial x_j}\right) = f(\theta)$$

Then,

$$\frac{\partial \tilde{\theta}}{\partial t} - \frac{\partial^2 k_{ij}}{\partial x_i \partial x_j} \tilde{\theta} = 0$$

Now considering the perturbation given in the problem statement, $\tilde{\theta} = e^{\omega t + i\kappa \mathbf{n}\cdot\mathbf{x}}$ where κ is real,

$$\left(\omega - \frac{\partial^2 k_{ij}}{\partial x_i \partial x_j} \right) e^{\omega t + i\kappa \mathbf{n}\cdot\mathbf{x}} = 0$$

From which we obtain:

$$\omega = \frac{\partial^2 k_{ij}}{\partial x_i \partial x_j}$$

Therefore, we conclude that

$$\omega = \frac{\partial^2 k_{ij}}{\partial x_i \partial x_j} \leq 0, \text{ the solution is stable}$$

$$\omega = \frac{\partial^2 k_{ij}}{\partial x_i \partial x_j} > 0, \text{ the solution is unstable}$$

CHAPTER 7: Arbitrary Lagrangian Eulerian formulations

7.1. Develop your own code (Matlab, FORTRAN, C, Maple and etc.) to solve the 1D advection-diffusion equation

$$u\frac{d\phi}{dx} = \upsilon\frac{d^2\phi}{dx^2}$$

with Galerkin and SUPG method separately. [See Example 7.2]

The BCs and parameters are assigned to a real world problem to determine the particles distribution at the steady state. Consider a 1m length segment in a long tube filled with a solution. At steady state, the particle concentration at end A **is** 5% **and that at end** B **is** 20%**, i.e.** $\phi(x=0)=0.05$, $\phi(x=1)=0.2$**. The solvent flows in the tube from end** A **to** B **under a constant velocity** u = 2 m/s**. The particles diffusion coefficient in the solvent is** υ = 0.025 m^2/s**. Please provide the distribution of particles concentration distribution along the tube segment.**

Simulate and discuss the following situations:

(a) Mesh the domain with 10**,** 20**,** 50**,** 100 **and** 200 **uniform size elements. What is the element Peclet number** P_e **for each mesh? Compare the analytical solution, Galerkin and SUPG prediction. In SUPG method, select** $\gamma = \frac{\Delta x}{2}\left(\coth(P_e) - \frac{1}{P_e}\right)$ **for each case. Discuss the stability and accuracy of the results.**

(b) In the mesh with 20 **uniform size elements, conduct the SUPG with** $\gamma = 10\gamma_0, 2\gamma_0, 0.5\gamma_0$ **and** $0.1\gamma_0$**, where** $\gamma_0 = \frac{\Delta x}{2}\left(\coth(P_e) - \frac{1}{P_e}\right)$**. Discuss the influence of** γ.

(c) Mesh the domain with a nonuniform mesh. Discuss the following:

c.1) Where should the finer mesh be?

c.2) How to select a proper γ?

Solution to 7.1.

Analytical solution:

The steady state linear advection-diffusion equation in one dimension is

$$u\frac{d\phi}{dx} - \nu\frac{d^2\phi}{dx^2} = 0, \qquad x \in [0, L] \tag{1}$$

where ϕ is the dependent variable, ν is the kinematic viscosity (diffusion coefficient) and u is a given velocity of the system. The analytical solution of equation (1) is straightforward to obtain, see for example Kreyszig[1],

$$\phi = C_1 e^{\frac{u}{\nu}x} + C_2$$

Substituting the given boundary conditions of $\phi(x = 0) = 0.05$, $\phi(x = 1) = 0.20$, and the given flow velocity of $u = 2\, m/s$ and the particles diffusion coefficient in the solvent of $\nu = 0.025\, m^2/s$

$$\phi = 2.707 \cdot 10^{-36} e^{\frac{u}{\nu}x} + 0.05$$

Galerkin method:

The development of the Galerkin discretization of the advection-diffusion equation using linear shape functions is given in the book. We need to multiply equation (1) by a test function, w, and integrate over the domain

$$\int_\Omega w\left(u\frac{d\phi}{dx} - \nu\frac{d^2\phi}{dx^2}\right)dx = 0, \qquad u > 0 \tag{2}$$

Integrating by parts and using the divergence theorem, one obtains the weak form of the advection-diffusion equation

[1] Kreyszig, E.. *Advanced Engineering Mathematics*. Wiley, 2011.

$$\int_{\Omega} wu\frac{d\phi}{dx}dx + \int_{\Omega} \nu\frac{dw}{dx}\frac{d\phi}{dx}dx = 0, \qquad \forall w \in U_0 \tag{3}$$

Performing the usual discretization using finite elements, Ω_e, equation (3) for each element can be rewritten as

$$\left(\int_{\Omega_e} uN_a\frac{dN_b}{dx}dx\right)\phi_b + \left(\int_{\Omega_e} \nu\frac{dN_a}{dx}\frac{dN_b}{dx}dx\right)\phi_b = 0, \qquad a,b = 1,2$$

where $N_a$and N_b are the finite element shape functions that discretize the test and trial functions, w and ϕ, respectively. Rewriting this equation in indicial form

$$L_{ab}\phi_b + K_{ab}\phi_b = 0, \qquad a,b = 1,2$$

where the convective matrix and the diffusion matrix are

$$L_{ab} = \int_{x_e}^{x_e+1} uN_a\frac{dN_b}{dx}dx\,; \quad K_{ab} = \int_{x_e}^{x_e+1} \nu\frac{dN_a}{dx}\frac{dN_b}{dx}dx\,, \qquad a,b = 1,2$$

Therefore, as it is shown in the book, using linear shape functions for elements of length l_e

$$\boldsymbol{L} = \frac{u}{2}\begin{bmatrix}-1 & 1\\ -1 & 1\end{bmatrix}; \quad \boldsymbol{K} = \frac{\nu}{l_e}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}$$

Finally, after assembly, the FEM matrix equations become

$$(\boldsymbol{L} + \boldsymbol{K})\boldsymbol{\phi} = 0 \tag{4}$$

SUPG method:

The Streamline Upwind Petrov-Galerkin method is obtained using the same reasoning explained previously but replacing the test function w in equation (2) by $\widetilde{w}$, which is defined as

$$\widetilde{w} = w + \gamma\frac{dw}{dx}$$

where $\gamma = \alpha\frac{l_e}{2}$, with α being defined differently according to the problem. The idea is quite simple: without changing the method of discretizing the advection-diffusion equation, one just

inserts a stabilization parameter that can be seen as an artificial viscosity. This way, the strong form for the SUPG method is obtained as

$$\int_{\Omega} w\left(u\frac{d\phi}{dx}-\nu\frac{d^2\phi}{dx^2}\right)dx+\sum_{e=1}^{N_e}\int_{\Omega_e}\gamma\frac{dw}{dx}\left(u\frac{d\phi}{dx}-\nu\frac{d^2\phi}{dx^2}\right)dx=0 \qquad (5)$$

The first integral corresponds to the strong form of the Galerkin method, while the second term corresponds to the upwind Petrov-Galerkin method. It is important to refer that this second term has been subdivided into element integrals due to the fact that $\frac{dw}{dx}$ and $\frac{d\phi}{dx}$ are both discontinuous in their derivatives, i.e., C^{-1}.

Integrating by parts the second term of Galerkin integral in equation (5)

$$\int_{\Omega} wu\frac{d\phi}{dx}dx+\int_{\Omega}\nu\frac{dw}{dx}\frac{d\phi}{dx}dx+\sum_{e=1}^{N_e}\int_{\Omega_e}u\gamma\frac{dw}{dx}\frac{d\phi}{dx}dx-\sum_{e=1}^{N_e}\int_{\Omega_e}\nu\gamma\frac{dw}{dx}\frac{d^2\phi}{dx^2}dx=0 \qquad (6)$$

Now, if one applies the same discretization procedure shown for the Galerkin method and using linear shape functions, one sees that the last term in the above equation is zero,

$$\sum_{e=1}^{N_e}\int_{\Omega_e}\left(wu\frac{d\phi}{dx}+\nu^*\frac{dw}{dx}\frac{d\phi}{dx}\right)dx=0$$

with $\nu^*=\nu+u\gamma$. Therefore, the discretized form of equation (6) is

$$(\boldsymbol{L}+\boldsymbol{K}^*)\boldsymbol{\phi}=0$$

where the only difference when compared to equation (4) is the diffusion matrix, $\boldsymbol{K}^*$, that is calculated replacing ν for ν^*. This way, the convective matrix and the diffusion matrix are

$$L_{ab}=\int_{x_e}^{x_e+1} uN_a\frac{dN_b}{dx}dx\,;\quad K_{ab}=\int_{x_e}^{x_e+1}\nu^*\frac{dN_a}{dx}\frac{dN_b}{dx}dx\,,\qquad a,b=1,2$$

Finally, it is shown in the book how the parameter γ can be determined for the one dimensional case of the advection-diffusion equation that is being evaluated in this work, leading to the following expression

$$\gamma = \frac{l_e}{2}\left(\coth(Pe) - \frac{1}{Pe}\right)$$

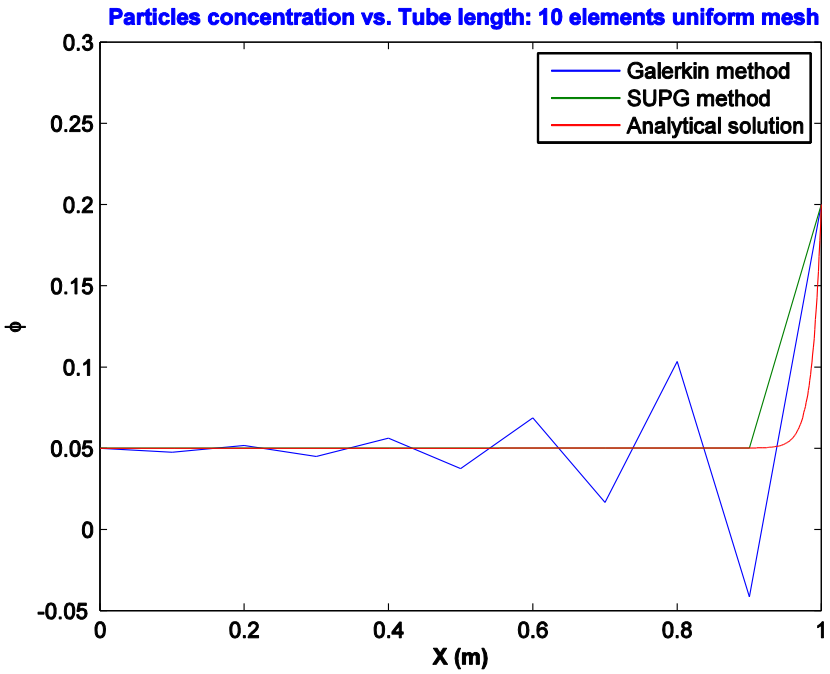
Solution to 7.1a).

Simulation and discussion of the results.

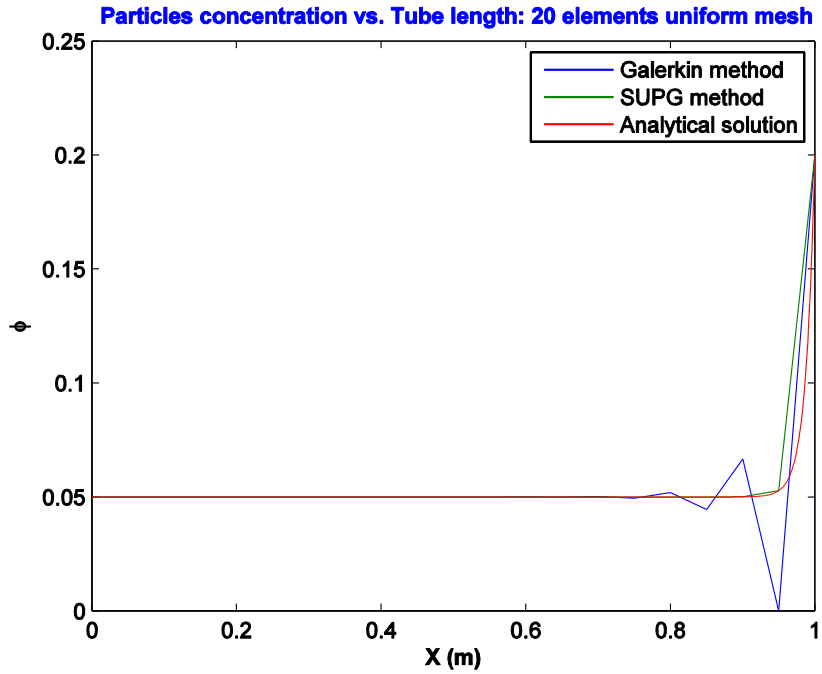
The analytical solution and the results obtained from the Galerkin method and the SUPG method will be presented for 5 different mesh sizes with 10, 20, 50, 100 and 200 uniform size elements. The element Peclet number for each mesh is

Table 1. Element Peclet number for each mesh.

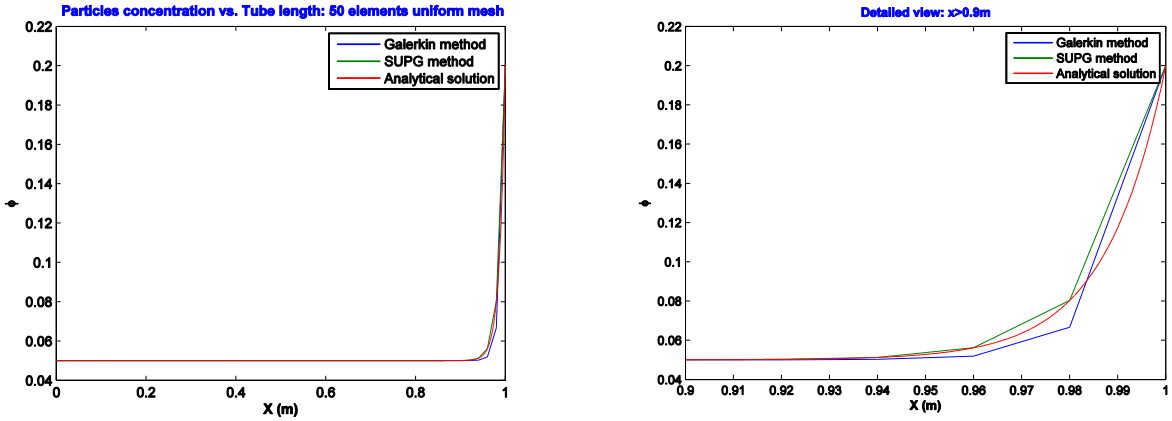
n_{el}	10	20	50	100	200
Pe	4	2	0.8	0.4	0.2



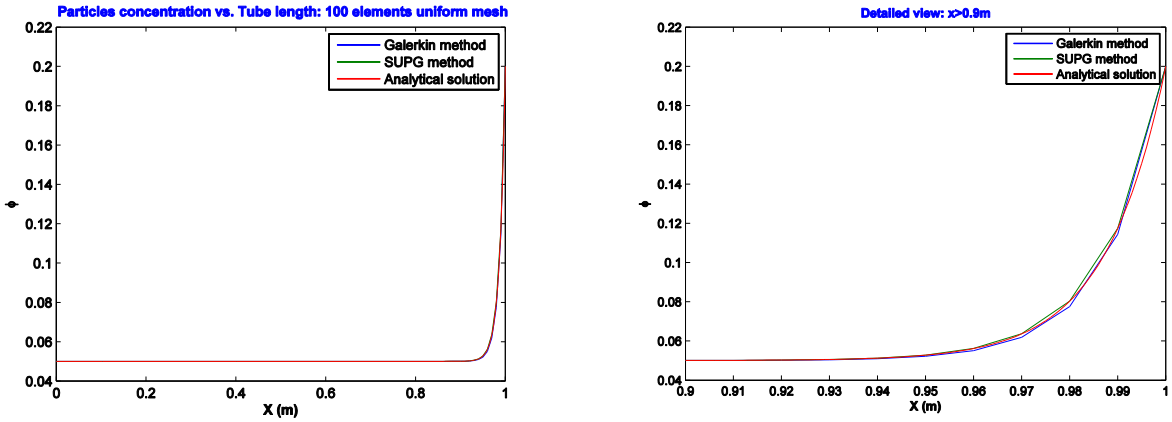
Comparison of Analytical solution and results from the Galerkin and SUPG methods: <u>10 uniform elements</u>.



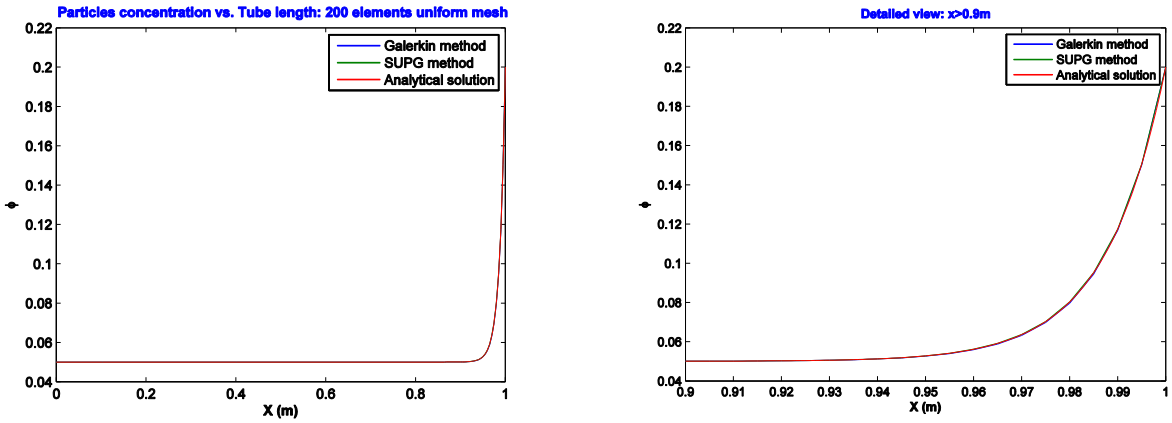
Comparison of Analytical solution and results from the Galerkin and SUPG methods: <u>20 uniform elements</u>.



Comparison of Analytical solution and results from the Galerkin and SUPG methods: 50 uniform elements.



Comparison of Analytical solution and results from the Galerkin and SUPG methods: 100 uniform elements.



Comparison of Analytical solution and results from the Galerkin and SUPG methods: 200 uniform elements.

All of the previous figures show the results obtained for different mesh sizes using uniform length elements for the Galerkin and SUPG methods by comparing them with the analytical solution. The results are consistent to Example 7.2 of the book. Indeed, when the Peclet number is larger than 1, the Galerkin method shows an unstable solution. On the contrary the SUPG method is stable for any Peclet number due to the stabilization parameter previously discussed.

Therefore, the Galerkin method returns spurious oscillatory concentration values, being the oscillation higher for higher values of the Peclet number. Not surprisingly, the accuracy of the predictions for both Galerkin and SUPG methods improves with increasing number of elements. Due to the fact that the variation of concentration is very abrupt very close to end B, only when a sufficient number of elements are used close to this end (roughly the last 8% of the tube) the results are very accurate.

Finally, when comparing the accuracy of the Galerkin method and the SUPG method it is important to understand that for the problem that is being solved there are two different physical processes that are being modeled: 1) the transport of particles caused by the flow within the tube of a solvent at a certain speed (in this case from end A to end B), i.e., the advective (or convective) term; 2) the diffusion of particles from end B to end A due to the fact that the concentration at end B is higher than at end A. This is important to keep in mind because the SUPG method is based on introducing a parameter on the diffusion term that can be thought of as an artificial viscosity. Hence, adding viscosity (i.e., increasing the diffusion coefficient) increases the contribution of the diffusion term, which leads to an earlier increase of the concentration of the particles. In conclusion, the SUPG method tends to the analytical solution with an overestimation of the dependent variable, ϕ, when compared to the Galerkin method. This will be clear with the analysis performed next, where we will assess the contribution of γ to the solution obtained by the SUPG method.

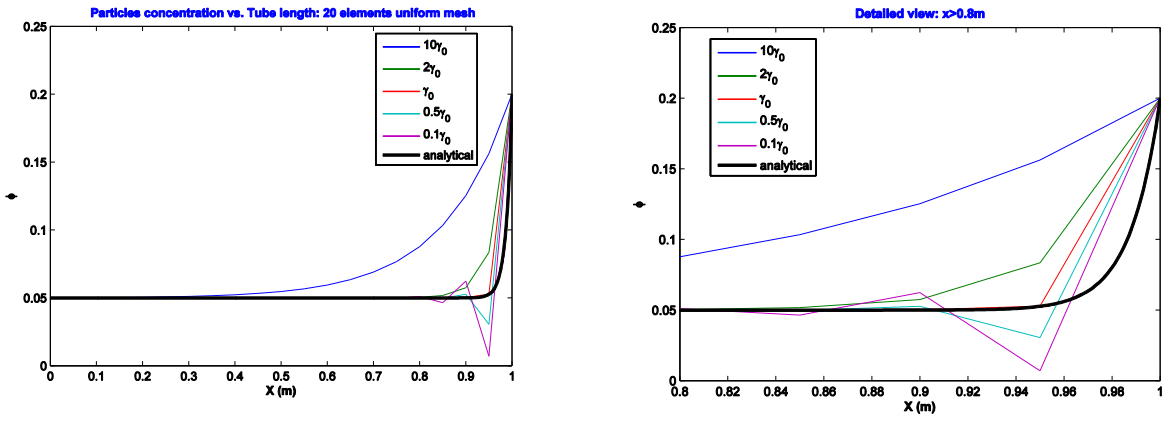
Solution to 7.1b).

Influence of the γ parameter.

The figure below shows the predictions using different values for the γ parameter: $\gamma = 10\gamma_0, 2\gamma_0, 0.5\gamma_0$ and $0.1\gamma_0$, where $\gamma_0 = \frac{l_e}{2}\left(\coth(Pe) - \frac{1}{Pe}\right)$. As was predicted in the previous section by comparing the Galerkin method with the SUPG method, it is clear that increasing the γ parameter causes an overestimation of the distribution of the concentration in the tube because it increases unrealistically the weight of the diffusion term in the advection-diffusion equation; the system becomes over damped.

On the other hand, if one uses γ lower than $\gamma_0 = \frac{l_e}{2}\left(\coth(Pe) - \frac{1}{Pe}\right)$ then the response starts being oscillatory due to the fact that not enough damping was introduced in the system. The lower γ is, the more oscillatory the response is; in the limit, if $\gamma = 0$ the same response given by the Galerkin method will be achieved.

This parametric study is very important to keep in mind once a three-dimensional system is being analyzed, because the correct γ parameter is not known a priori. Therefore, one might be over damping the system without knowing it because the response is smooth. When γ is too low and the system is underdamped it is easier to detect due to the instability of the response.



Influence of the γ parameter in the predictions using the SUPG method.

Solution to 7.3c).

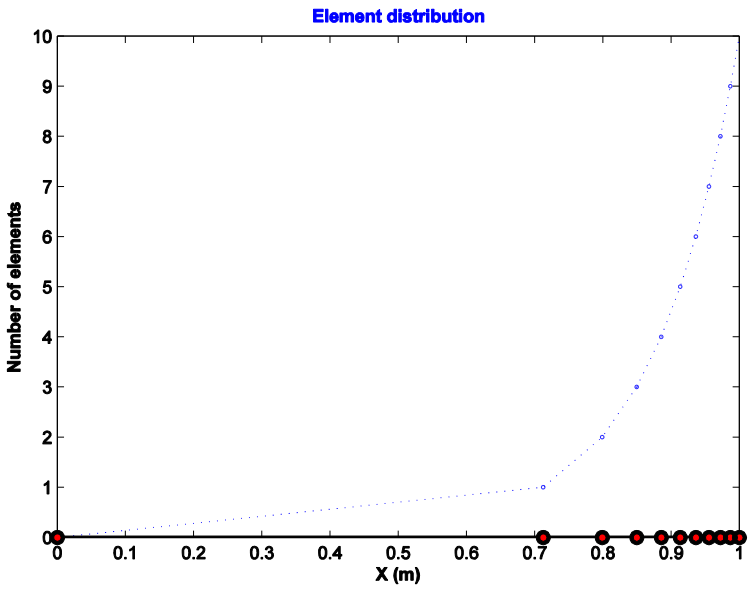
Nonuniform mesh.

As referred previously, increasing the number of elements increases the accuracy of the solution. However, it was also mentioned that it is predictable that using smaller elements at end B and larger elements at end A will produce better results, even without changing the total number of elements, because the change in concentration is very localized at end B, being very close to zero until the last 8% of the tube.

Several different meshes can be used to prove this hypothesis. One of the most effective meshes for this problem can be obtained considering an exponential variation of the number of elements along the tube towards the end B:

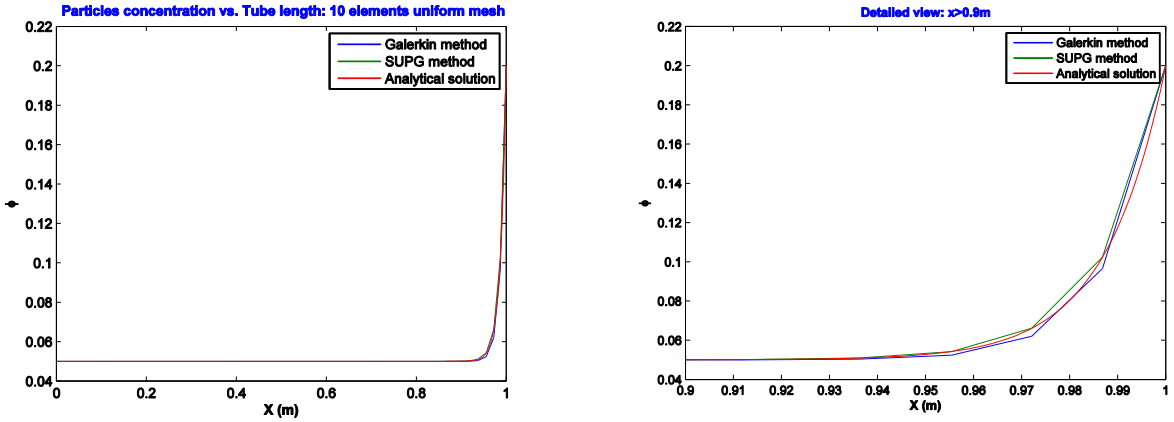
$$ne(x) = A \cdot \left(e^{\frac{\beta u}{\nu}x} - 1\right) \tag{S7.1.17}$$

where $A = \frac{ne_{total}}{\left(e^{\frac{\beta u}{\nu}L} - 1\right)}$, with ne_{total} being the total number of elements of the mesh, and β is a parameter to avoid a resolution that is too low at end A (the code uses $\beta = 0.1$). This distribution generates a mesh that is very coarse at the beginning of the tube and that is very refined at the end, just like it is needed. A plot of the mesh and the distribution is shown in the figure below for 10 elements.



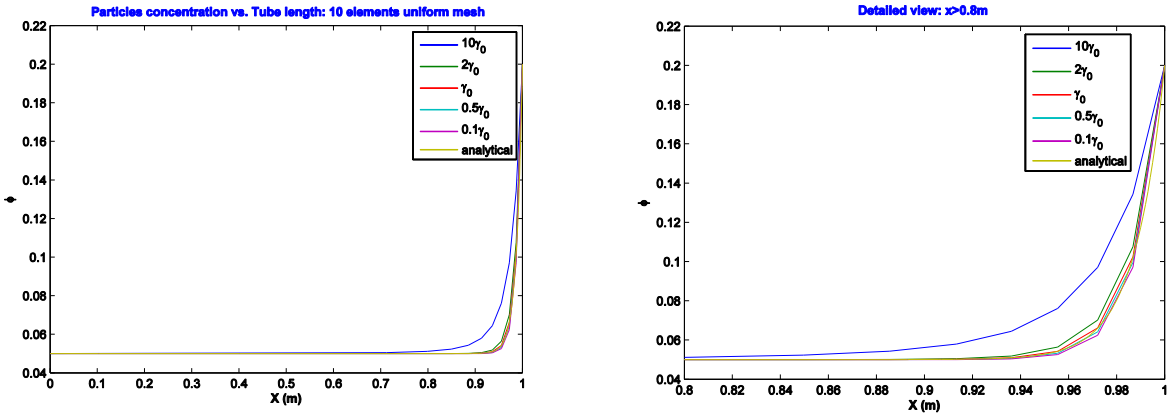
Element distribution and nonuniform mesh obtained for 10 elements.

The results obtained with this nonuniform mesh of 10 elements are also plotted in the next figure. As predicted, even for such a low number of elements, the simulations lead to an excellent result for both methods: Galerkin and SUPG. This was possible for the Galerkin method because the Peclet number was calculated for each element and the elements that lead to a variation of the particle concentration had a Peclet number lower than 1 (the first element is $0.7126\,m$ long, whether the second element is only $0.0864\,m$, with the remaining elements getting even smaller). This way, after the sixth element ($le_6 = 0.0228\,m$, thus $Pe = 0.912$) which has its first node at $X = 0.9134\,m$, the $Pe < 1$ eliminating the oscillation because it is at approximately that length that the concentration starts to increase.



Comparison of Analytical solution and results from the Galerkin and SUPG methods: <u>10 nonuniform elements</u>.

As a final comment, the variation of the γ parameter was also studied using the nonuniform mesh. As can be seen in the next figure, the influence of γ on the response is less pronounced when compared with the influence on the response obtained for a uniform mesh with the double of elements. This is reassuring because it means that if the mesh resolution is fine enough, even if one does not know the exact value of the γ parameter for a tridimensional case, the response may be accurate enough.



Influence of the γ parameter in the predictions using the SUPG method for a nonuniform mesh with 10 elements.

MATLAB code for Problem 7.1

```
%%%%%%%%%%%%%%%%%%%% Advection-Diffusion problem %%%%%%%%%%%%%%%%%%%%%%%%%
%
%    M. A. Bessa  (mbessa@u.northwestern.edu)
%    Northwestern University, Mechanical Engineering
```

```
%
%   April 15, 2012
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [] = Project1_AdvectionDiffusion()
    close all; clear all; clc;
    %% Input parameters.
    uni_mesh    = 1;     % Select uniform [1] or nonuniform [0] mesh
    ne          = 10;    % Number of elements
    nn          = ne+1;  % Number of nodes.
    tube_length = 1.0;   % Length of the tube. [m]
    phi_BC = [0.05,0.2]; % Particle concentration at end A and B
    BC_dof      = [1,nn];% Degrees-of-freedom of Boundary Conditions
    u           = 2;     % Velocity of the solvent flow in the tube from...
                         % end A to end B [m/s]
    nu          = 0.025; % Particles diffusion coef. in the solvent [m^2/s]
    beta        = 0.10;  % Parameter for nonuniform mesh

    %% Preprocessing.
    switch uni_mesh
        case 1
            % Nodal coordinates for uniform mesh.
            X = linspace(0, tube_length, nn)';
        case 0
            % Nodal coordinates for nonuniform mesh.
            A = ne/(exp(beta*u/nu*tube_length)-1);
            for el = 0:ne
                X(el+1) = 1/beta*nu/u*log(el/A+1);
            end
    end

    % Initialize nodal variables to zero.
    phi = zeros(nn,1);

    % Connectivity matrix (each row gives nodes in an element.)
    connect = [ 1:length(X)-1; 2:length(X) ]';

    % Calculate the length of each element
    le = zeros(ne,1); % length of each element [m]
    count = 0;
    for conn = connect'
        count = count+1;
        le(count) = X(conn(2))-X(conn(1));
    end

    %%
    % For this simple problem, using linear shape functions it is easy...
    % obtain the Convective matrix and Diffusion matrix for each element:
    L_Gal = zeros(nn,nn);
    K_Gal = zeros(nn,nn);
    L_SUPG = zeros(nn,nn);
    K_SUPG = zeros(nn,nn);
    K_SUPG_gam1 = zeros(nn,nn);
    K_SUPG_gam2 = zeros(nn,nn);
    K_SUPG_gam3 = zeros(nn,nn);
    K_SUPG_gam4 = zeros(nn,nn);
    for conn = connect'
        % Element Convective matrix
        L_Gal_e = u/2*[ -1, 1; ...
                        -1, 1 ];
        % Element Diffusion matrix
        K_Gal_e = nu/le(conn(1))*[ 1, -1; ...
```

```
                            -1,  1 ];
        L_Gal(conn,conn) = L_Gal(conn,conn) + L_Gal_e;
        K_Gal(conn,conn) = K_Gal(conn,conn) + K_Gal_e;

        Pe = u*le(conn(1))/(2*nu);
        alpha = coth(Pe)-1/Pe;
        gamma = le(conn(1))/2*alpha;
        nu_bar = u*gamma;
        % Element Convective matrix
        L_SUPG_e = u/2*[ -1, 1; ...
                    -1, 1 ];
        % Element Diffusion matrix
        K_SUPG_e = (nu+nu_bar)/le(conn(1))*[ 1, -1; ...
                            -1,  1 ];
        L_SUPG(conn,conn) = L_SUPG(conn,conn) + L_SUPG_e;
        K_SUPG(conn,conn) = K_SUPG(conn,conn) + K_SUPG_e;

        % PARAMETRIC STUDY OF GAMMA
        nu_bar1 = 10*nu_bar;  % 10 times gamma
        K_SUPG_gam1_e = (nu+nu_bar1)/le(conn(1))*[ 1, -1; ...
                            -1,  1 ];
        K_SUPG_gam1(conn,conn) = K_SUPG_gam1(conn,conn) + K_SUPG_gam1_e;

        nu_bar2 = 2*nu_bar;  % 2 times gamma
        K_SUPG_gam2_e = (nu+nu_bar2)/le(conn(1))*[ 1, -1; ...
                            -1,  1 ];
        K_SUPG_gam2(conn,conn) = K_SUPG_gam2(conn,conn) + K_SUPG_gam2_e;

        nu_bar3 = 0.5*nu_bar;  % 0.5 times gamma
        K_SUPG_gam3_e = (nu+nu_bar3)/le(conn(1))*[ 1, -1; ...
                            -1,  1 ];
        K_SUPG_gam3(conn,conn) = K_SUPG_gam3(conn,conn) + K_SUPG_gam3_e;

        nu_bar4 = 0.1*nu_bar;  % 0.1 times gamma
        K_SUPG_gam4_e = (nu+nu_bar4)/le(conn(1))*[ 1, -1; ...
                            -1,  1 ];
        K_SUPG_gam4(conn,conn) = K_SUPG_gam4(conn,conn) + K_SUPG_gam4_e;
    end

    % Since both matrices operate on phi, they can be added:
    M_Gal = L_Gal + K_Gal;
    M_SUPG = L_SUPG + K_SUPG;

    M_SUPG_gam1 = L_SUPG + K_SUPG_gam1;
    M_SUPG_gam2 = L_SUPG + K_SUPG_gam2;
    M_SUPG_gam3 = L_SUPG + K_SUPG_gam3;
    M_SUPG_gam4 = L_SUPG + K_SUPG_gam4;

    F = zeros(nn,1);
    BigNumber = 10^9;
    for i=1:length(BC_dof)
    M_Gal(BC_dof(i),BC_dof(i)) = BigNumber;
    M_SUPG(BC_dof(i),BC_dof(i)) = BigNumber;

    M_SUPG_gam1(BC_dof(i),BC_dof(i)) = BigNumber;
    M_SUPG_gam2(BC_dof(i),BC_dof(i)) = BigNumber;
    M_SUPG_gam3(BC_dof(i),BC_dof(i)) = BigNumber;
    M_SUPG_gam4(BC_dof(i),BC_dof(i)) = BigNumber;

    % RHS term ("force")
```

```
    F(BC_dof) = phi_BC*BigNumber;
    end

    % Solution
    phi_Gal = M_Gal \ F;
    phi_SUPG = M_SUPG \ F;

    phi_SUPG_gam1 = M_SUPG_gam1 \ F;
    phi_SUPG_gam2 = M_SUPG_gam2 \ F;
    phi_SUPG_gam3 = M_SUPG_gam3 \ F;
    phi_SUPG_gam4 = M_SUPG_gam4 \ F;

    %% Analytical solution
    phi_A = phi_BC(1);
    X_A   = X(1);
    phi_B = phi_BC(end);
    X_B   = X(end);

    C1 = (phi_B-phi_A)/exp(u/nu*(X_B-X_A));
    C2 = phi_A-C1*exp(u/nu*X_A);

    X_ana   = X_A:(0.001*abs(X_B-X_A)):X_B;
    phi_ana = zeros(length(X_ana),1);
    phi_ana = C1*exp(u/nu*X_ana)+C2;

    figure();
    plot(X, phi_Gal, X, phi_SUPG, X_ana, phi_ana);
    title('Particles concentration vs. Tube length: 10 elements uniform mesh',...
        'FontWeight', 'bold', 'Color', [0,0,1]);
    xlabel('X (m)', 'FontWeight', 'bold');
    ylabel('\phi', 'FontWeight', 'bold');
    legend('Galerkin method', 'SUPG method', 'Analytical solution');

    figure();
    plot(X, phi_SUPG_gam1, X, phi_SUPG_gam2, X, phi_SUPG, X,...
        phi_SUPG_gam3, X, phi_SUPG_gam4, X_ana, phi_ana);
    title('Particles concentration vs. Tube length: 10 elements uniform mesh',...
        'FontWeight', 'bold', 'Color', [0,0,1]);
    xlabel('X (m)', 'FontWeight', 'bold');
    ylabel('\phi', 'FontWeight', 'bold');
    legend('10\gamma_0', '2\gamma_0', '\gamma_0', '0.5\gamma_0', '0.1\gamma_0',...
        'analytical solution');

    %% Nonuniform mesh plot

    if uni_mesh == 0
        elem_dist = A*(exp(beta*u/nu*X)-1);

        plot_nodes = 0.0.*elem_dist;

        figure();
        plot(X, elem_dist, 'bo:', X, plot_nodes, 'ro-');
        title('Element distribution', 'FontWeight', 'bold', 'Color',...
            [0,0,1]);
        xlabel('X (m)', 'FontWeight', 'bold');
        ylabel('Number of elements', 'FontWeight', 'bold');
    end

end
```

CHAPTER 8: Element technology

8.1. Show that when $X_2 \neq \frac{1}{2}(X_1 + X_3)$, then the 3-node element in Example 2.5 does not reproduce the quadratic displacement field. *Hint*: set the node displacements by a quadratic field in X and examine the resulting field.

Solution to 8.1.

The map between the reference configuration and the parent element is given by

$$X(\xi) = N_I(\xi)X_I$$

With the shape functions:

$$\boldsymbol{N}^T = \left[\frac{1}{2}(\xi^2 - \xi) \quad 1 - \xi^2 \quad \frac{1}{2}(\xi^2 + \xi)\right], \qquad \xi = \frac{2X - X_1 - X_3}{X_1 + X_3}$$

And the reference coordinates:

$$\boldsymbol{X} = \begin{Bmatrix} X_1 \\ (1-\beta)X_1 + \beta X_\xi \\ X_3 \end{Bmatrix}, \qquad \beta \in]0,1[\setminus \{1/2\}$$

For an isoparametric element, the dependent variable u is interpolated by the same shape functions, so

$$u(\xi) = N_I(\xi)u_I$$

Let the dependent variable be a linear function of the spatial coordinates, so

$$u = \alpha_0 + \alpha_1 X + \alpha_2 X^2$$

Where α_i are arbitrary parameters. If the nodal values of the field are given by the above, then

$$u_I = \alpha_0 + \alpha_1 X_I + \alpha_2 X_I^2$$

Substituting the nodal values of the dependent variable given by the above expression into the displacement field obtained with the discretization:

$$u = \alpha_0 \, 1_I N_I(\xi) + \alpha_1 (N_I X_I) + \alpha_2 (N_I X_I^2)$$

After some algebra, we can reach the following expression:

$$u = \alpha_0 + \alpha_1 \left\{ (1-2\beta) \left[\frac{2(X-X_1)^2}{X_3 - X_1} + 2X_1 \right] + (4\beta - 1)X \right\}$$
$$+ \alpha_2 \left\{ X^2 - (2\beta - 1) \left[\frac{(X-X_1)(X-X_3)[X_1(3-2\beta) + X_3(1+2\beta)]}{X_1 - X_3} \right] \right\}$$

Reaching the conclusion that for $\beta = \frac{1}{2}$:

$$\beta = \frac{1}{2} \Rightarrow u = \alpha_0 + \alpha_1 X + \alpha_2 X^2$$

The quadratic displacement field is reproduced. However, for $\beta \in \,]0,1[\, \backslash \{1/2\}$ the quadratic displacement field is not reproduced:

$$\beta \in \,]0,1[\, \backslash \{1/2\} \, \Rightarrow u \, \neq \alpha_0 + \alpha_1 X + \alpha_2 X^2 \qquad q.e.d.$$

8.2. Show that the weak form (8.5.12) leads to the following strong form:

$$\bar{p} = p, \; \bar{D}_{ii} = D_{ii}, \quad \sigma_{ij,j}^{\text{dev}} - \bar{p}_{,i} + \rho b_i = \rho \dot{v}_i$$

$$n_i \left(\sigma_{ij}^{\text{dev}} - \bar{p} \delta_{ij} \right) = \bar{t}_j \text{ on } \Gamma_t, \quad [n_i \left(\sigma_{ij}^{\text{dev}} - \bar{p} \delta_{ij} \right)] = 0 \text{ on } \Gamma_{\text{int}}$$

Solution to 8.2.

Due to the symmetry of σ_{ij}^{dev}, the internal power (8.5.12) can be written as:

$$\delta P^{int} = \int_{\Omega} \left(\delta v_{i,j}\, \sigma_{ij}^{dev} - \delta \bar{D}_{ii}\, p\, \right) d\Omega - \int_{\Omega} \delta \left[\bar{p}\, (v_{i,i} - \bar{D}_{ii}\right] d\Omega$$

With the external power and kinetic power given by:

$$\delta P^{ext} = \int_{\Gamma_t} \delta v_i\, \bar{t}_i d\Gamma + \int_{\Omega} \delta v_i\, \rho b_i d\Omega$$

$$\delta P^{kin} = \int_{\Omega} \delta v_i\, \rho \dot{v}_i d\Omega$$

Taking the variations of the second term in the internal power yields

$$\int_{\Omega} \delta \left[\bar{p}\, (v_{i,i} - \bar{D}_{ii}\right] d\Omega = \int_{\Omega} \left[\delta \bar{p}\, \left(v_{i,i} - \bar{D}_{ii}\right) + \bar{p}\, (\delta v_{i,i} - \delta \bar{D}_{ii})\, \right] d\Omega$$

Considering the third term on the RHS of the above:

$$\int_{\Omega} \delta v_{i,j} \delta_{ji} \bar{p}\, d\Omega = \int_{\Omega} \left[\left(\delta v_i \delta_{ji} \bar{p}\right)_{,j} - \delta v_i \delta_{ji} \bar{p}_{,j}\right] d\Omega =$$

$$= \int_{\Gamma_t} \delta v_i\ \bar{p}\, \delta_{ji} n_j d\Gamma + \int_{\Gamma_{int}} \delta v_i [\![\bar{p}\, \delta_{ji} n_j]\!] d\Gamma - \int_{\Omega} \delta v_i \delta_{ji} \bar{p}_{,j} d\Omega$$

Where the Gauss divergence theorem was used to obtain the first two terms of the RHS, noting that Γ was changed to Γ_t because $\delta v_i = 0$ on $\Gamma_v = \Gamma - \Gamma_t$.

Evaluating the 1st term of the internal power:

$$\int_{\Omega} \delta v_{i,j} \sigma_{ij}^{dev}\, d\Omega = \int_{\Omega} \left[\left(\delta v_i \sigma_{ij}^{dev}\right)_{,j} - \delta v_i \sigma_{ij,j}^{dev}\right] d\Omega =$$

$$= \int_{\Gamma_t} \delta v_i\, \sigma_{ij}^{dev}\, n_j\, d\Gamma + \int_{\Gamma_{int}} \delta v_i\, [\![\sigma_{ij}^{dev}\, n_j]\!]\, d\Gamma - \int_{\Omega} \delta v_i \sigma_{ij,j}^{dev}\, d\Omega$$

Substituting each of the results in δP^{int} we obtain:

$$\delta P^{int} - \delta P^{ext} + \delta P^{kin} = 0$$

$$\int_{\Omega}\left[\delta v_i\left(-\sigma_{ij,j} + \delta_{ji}\,\bar{p},j\right) + \delta\bar{D}_{ii}\,(\bar{p} - p\,) + \delta\bar{p}\,(\,\bar{D}_{ii} - v_{i,i})\right]d\Omega + \int_{\Gamma_t}\delta v_i(\,\sigma_{ij}^{dev}\,n_j$$

$$-\,\bar{p}\,\delta_{ji}n_j)\,d\Gamma + \int_{\Gamma_{int}}\delta v_i\left(\left[\!\left[\sigma_{ij}^{dev}\,n_j\right]\!\right] - \left[\!\left[\bar{p}\,\delta_{ij}n_j\right]\!\right]\right)d\Gamma - \int_{\Gamma_t}\delta v_i\bar{t}_i d\Gamma$$

$$-\int_{\Omega}\delta v_i\rho b_i d\Omega + \int_{\Omega}\delta v_i\rho\dot{v}_\iota d\Omega = 0$$

Using the arbitrariness of the test functions then gives the strong form:

$$\sigma_{ij,j}^{dev} - \bar{p}_{,i} + \rho b_i = \rho\dot{v}_\iota$$

$$\bar{p} = p$$

$$\bar{D}_{ii} = D_{ii}$$

$$n_i\left(\sigma_{ij}^{dev} - \bar{p}\,\delta_{ij}\right) = \bar{t}_j \quad \text{on} \quad \Gamma_t$$

$$\left[\!\left[n_i(\sigma_{ij}^{dev} - \bar{p}\,\delta_{ij}\right]\!\right] = 0 \quad \text{on} \quad \Gamma_{int} \qquad q.e.d$$

8.3. Show that the weak form (8.5.13) leads to the following strong form:

$$\sigma_{ij,j}^{\text{dev}} - \bar{p}_{,i} + \rho b_i = \rho\dot{v}_i,\ \bar{p} = p,\ \bar{D}_{ii} = 0$$

$$n_i\left(\sigma_{ij}^{\text{dev}} - \bar{p}\delta_{ij}\right) = \bar{t}_j \text{ on } \Gamma_t,\ \ \left[n_i\left(\sigma_{ij}^{\text{dev}} - \bar{p}\delta_{ij}\right)\right] = 0 \text{ on } \Gamma_{\text{int}}$$

Solution to 8.3.

This proof is obvious because the only difference is that $D_{ii} = 0$, so only one equation in the strong form is changed

$$\bar{D}_{ii} = D_{ii} = 0$$

8.4. By using the transformation for stresses and letting $\delta\mathbf{D} = \delta\mathbf{F}$, show that (8.5.1) can be transformed to (8.5.14).

Solution to 8.4.

Equation (8.5.1) is:

$$0 = \delta\Pi^{HW}(\boldsymbol{v}, \bar{\boldsymbol{D}}, \bar{\boldsymbol{\sigma}}) = \int_{\Omega} \delta\bar{\boldsymbol{D}} : \boldsymbol{\sigma}\, \boldsymbol{B}(\bar{\boldsymbol{D}}) d\Omega + \int_{\Omega} \delta\left[\bar{\boldsymbol{\sigma}} : (\boldsymbol{D}(\boldsymbol{v}) - \bar{\boldsymbol{D}})\right] d\Omega - \delta P^{ext} - \delta P^{kin}$$

Let's start by converting the integrals to the reference configuration:

$$\int_{\Omega_0} \delta\bar{\boldsymbol{D}} : \sigma(\bar{\boldsymbol{D}})\, J\, d\Omega_0 + \int_{\Omega_0} \delta\left[\bar{\boldsymbol{\sigma}} : (\boldsymbol{D}(\boldsymbol{v}) - \bar{\boldsymbol{D}})\right] J d\Omega_0 - \delta P^{ext} - \delta P^{kin} = 0$$

Now, noting that by definition of $\boldsymbol{D}$ and symmetry of stress $\boldsymbol{\sigma}$,

$$D_{ij}\, \sigma_{ij}\, J = \frac{\delta v_i}{\delta x_j}\, \sigma_{ij} J$$

Applying the chain rule,

$$D_{ij}\, \sigma_{ij}\, J = \frac{\delta v_i}{\delta X_K} \frac{\delta X_k}{\delta x_j} \sigma_{ij} J$$

Using the definition of $\boldsymbol{F}$,

$$D_{ij}\, \sigma_{ij}\, J = \dot{F_{ik}} \frac{\delta X_k}{\delta x_j}\, \sigma_{ij} J$$

And finally using the appropriate stress transformation:

$$D_{ij}\, \sigma_{ij}\, J = \dot{F}_{ik}\, P_{ki}$$

Now, using this relation we can write:

$$\delta\bar{\boldsymbol{D}} : \boldsymbol{\sigma}(\bar{\boldsymbol{D}}) J = \dot{\bar{\boldsymbol{F}}}^T : \boldsymbol{P}(\dot{\bar{\boldsymbol{F}}})$$

$$\bar{\boldsymbol{\sigma}} : \boldsymbol{D}(\boldsymbol{v}) = \boldsymbol{D}(\boldsymbol{v}) : \bar{\boldsymbol{\sigma}} = \dot{\boldsymbol{F}}^T(\dot{\boldsymbol{u}}) : \bar{\boldsymbol{P}}$$

$$\bar{\boldsymbol{\sigma}} : \bar{\boldsymbol{D}} = \bar{\boldsymbol{P}} : \dot{\bar{\boldsymbol{F}}}$$

Hence, the three-field weak form becomes

$$0 = \delta\Pi\left(\dot{\boldsymbol{u}}, \dot{\bar{\boldsymbol{F}}}, \bar{P}\right)$$

$$= \int_{\Omega_0} \delta\dot{\bar{\boldsymbol{F}}}^T : \boldsymbol{P}\left(\dot{\bar{\boldsymbol{F}}}\right) d\Omega_0 + \int_{\Omega_0} \delta\left[\bar{\boldsymbol{P}} : \left(\dot{\bar{\boldsymbol{F}}} - \dot{\bar{\boldsymbol{F}}}^T\right)\right] d\Omega_0 - \delta P^{ext} + \delta P^{kin}$$

Finally, noting that equation (8.5.14) is a virtual work instead of a virtual power, the above expression needs to be converted accordingly:

$$0 = \delta\Pi^{HW}(\boldsymbol{u}, \bar{\boldsymbol{F}}, \bar{P})$$

$$= \int_{\Omega_0} \delta\dot{\bar{\boldsymbol{F}}}^T : \boldsymbol{P}\,(\bar{\boldsymbol{F}}) d\Omega_0 + \int_{\Omega_0} \delta\,[\bar{\boldsymbol{P}} : (\boldsymbol{F}^T(\boldsymbol{u}) - \boldsymbol{F}^{-T}]\, d\Omega_0 - \delta W^{ext}$$

$$+ \delta W^{kin} \qquad\qquad q.e.d$$

This can be expressed in index notation, leading to equation (8.5.14).

CHAPTER 9: Beams and shells

9.1. Consider the three-node CB element shown in Figure 9.16. The shape functions are quadratic in ξ. Develop the velocity field and the rate-of-deformation in the corotational coordinates. Give an expression for the nodal forces. Develop an expression for the angle between the pseudonormal $\boldsymbol{p}$ and the true normal to the midline.

Solution to 9.1.

The motion of the 6-node continuum element is:

$$\boldsymbol{x} = \boldsymbol{x}_{I^*} N_{I^*}(\xi, \eta)$$

With the following shape functions and respective derivatives,

$$N_{1^*} = \frac{1}{4}\xi(\xi-1)(1-\eta) \quad \therefore \quad \begin{cases} N_{1^*,\xi} = \frac{1}{2}\left(\xi - \frac{1}{2}\right)(1-\eta) \\ N_{1^*,\eta} = -\frac{1}{4}\xi(\xi-1) \end{cases}$$

$$N_{2^*} = \frac{1}{2}(1-\xi^2)(1-\eta) \quad \therefore \quad \begin{cases} N_{2^*,\xi} = -\xi(1-\eta) \\ N_{2^*,\eta} = -\frac{1}{2}(1-\xi^2) \end{cases}$$

$$N_{3^*} = \frac{1}{4}\xi(\xi+1)(1-\eta) \quad \therefore \quad \begin{cases} N_{3^*,\xi} = \frac{1}{2}\left(\xi + \frac{1}{2}\right)(1-\eta) \\ N_{3^*,\eta} = -\frac{1}{4}\xi(\xi+1) \end{cases}$$

$$N_{4^*} = \frac{1}{4}\xi(\xi+1)(1+\eta) \quad \therefore \quad \begin{cases} N_{4^*,\xi} = \frac{1}{2}\left(\xi + \frac{1}{2}\right)(1+\eta) \\ N_{4^*,\eta} = \frac{1}{4}\xi(\xi+1) \end{cases}$$

$$N_{5^*} = \frac{1}{2}(1-\xi^2)(1+\eta) \quad \therefore \quad \begin{cases} N_{5^*,\xi} = -\xi(1+\eta) \\ N_{5^*,\eta} = \frac{1}{2}(1-\xi^2) \end{cases}$$

$$N_{6^*} = \frac{1}{4}\xi(\xi-1)(1+\eta) \quad \therefore \quad \begin{cases} N_{6^*,\xi} = \frac{1}{2}\left(\xi - \frac{1}{2}\right)(1+\eta) \\ N_{6^*,\eta} = \frac{1}{4}\xi(\xi-1) \end{cases}$$

Replacing the shape functions in the motion equation and rearranging the terms:

$$\boldsymbol{x} = \frac{1}{4}(\boldsymbol{x}_{1^*} + \boldsymbol{x}_{6^*})\xi(\xi-1) + \frac{1}{4}(\boldsymbol{x}_{6^*} - \boldsymbol{x}_{1^*})\xi(\xi-1)\eta + \frac{1}{2}(\boldsymbol{x}_{2^*} + \boldsymbol{x}_{5^*})(1-\xi^2) + \frac{1}{2}(\boldsymbol{x}_{5^*} - \boldsymbol{x}_{2^*})(1-\xi^2)\eta + \frac{1}{4}(\boldsymbol{x}_{3^*} + \boldsymbol{x}_{4^*})\xi(\xi+1) + \frac{1}{4}(\boldsymbol{x}_{4^*} - \boldsymbol{x}_{3^*})\xi(\xi+1)\eta$$

Considering,

$$\boldsymbol{x}_1 = \frac{1}{2}(\boldsymbol{x}_{1^*} + \boldsymbol{x}_{6^*}) \equiv \frac{1}{2}(\boldsymbol{x}_{1^-} + \boldsymbol{x}_{1^+}) \quad , \quad \boldsymbol{x}_2 = \frac{1}{2}(\boldsymbol{x}_{2^*} + \boldsymbol{x}_{5^*}) \equiv \frac{1}{2}(\boldsymbol{x}_{2^-} + \boldsymbol{x}_{2^+})$$

$$\boldsymbol{x}_3 = \frac{1}{2}(\boldsymbol{x}_{3^*} + \boldsymbol{x}_{4^*}) \equiv \frac{1}{2}(\boldsymbol{x}_{3^-} + \boldsymbol{x}_{3^+}) \quad , \quad \|\boldsymbol{x}_{6^*} - \boldsymbol{x}_{1^*}\| = h_1^0$$

$$\|\boldsymbol{x}_{5^*} - \boldsymbol{x}_{2^*}\| = h_2^0 \quad , \quad \|\boldsymbol{x}_{4^*} - \boldsymbol{x}_{3^*}\| = h_3^0 \quad , \quad \boldsymbol{p}_1 = \frac{\boldsymbol{x}_{6^*} - \boldsymbol{x}_{1^*}}{h_1^0}$$

$$\boldsymbol{p}_2 = \frac{\boldsymbol{x}_{5^*} - \boldsymbol{x}_{2^*}}{h_2^0} \quad , \quad \boldsymbol{p}_3 = \frac{\boldsymbol{x}_{4^*} - \boldsymbol{x}_{3^*}}{h_3^0}$$

We can then rewrite the equation of motion as:

$$\boldsymbol{x} = \frac{1}{2}\boldsymbol{x}_1\xi(\xi-1) + \boldsymbol{x}_2(1-\xi^2) + \frac{1}{2}\boldsymbol{x}_3\xi(\xi+1) + \eta\frac{h_1^0}{4}\boldsymbol{p}_1\xi(\xi-1) + \eta\frac{h_2^0}{2}\boldsymbol{p}_2(1-\xi^2) + \eta\frac{h_3^0}{4}\boldsymbol{p}_3\xi(\xi+1)$$

The velocity field can now be determined to be:

$$\boldsymbol{v} = \frac{1}{2}\boldsymbol{v}_1\xi(\xi-1) + \boldsymbol{v}_2(1-\xi^2) + \frac{1}{2}\boldsymbol{v}_3\xi(\xi+1) + \eta\frac{h_1^0}{4}\xi(\xi-1)\boldsymbol{\omega}_1 \times \boldsymbol{p}_1 + \eta\frac{h_2^0}{2}(1-\xi^2)\boldsymbol{\omega}_2 \times \boldsymbol{p}_2 + \eta\frac{h_3^0}{4}\xi(\xi+1)\boldsymbol{\omega}_3 \times \boldsymbol{p}_3$$

with $\boldsymbol{\omega}_I = \dot{\theta}_I \boldsymbol{e}_y = \omega_I \boldsymbol{e}_y$, and $\boldsymbol{v}_I = v_{Ix}\boldsymbol{e}_x + v_{Iz}\boldsymbol{e}_z$, and $\boldsymbol{p}_I = \cos\theta_I\, \boldsymbol{e}_x + \sin\theta_I\, \boldsymbol{e}_z$.

To develop the velocity field in corotational coordinates, we need to determine the laminar base vectors:

$$\hat{\boldsymbol{e}}_x = \frac{x_{,\xi}\boldsymbol{e}_x + z_{,\xi}\boldsymbol{e}_z}{\left(x_{,\xi}^2 + z_{,\xi}^2\right)^{1/2}} \quad , \quad \hat{\boldsymbol{e}}_z = \frac{-z_{,\xi}\boldsymbol{e}_x + x_{,\xi}\boldsymbol{e}_z}{\left(x_{,\xi}^2 + z_{,\xi}^2\right)^{1/2}}$$

Where,

$$x_{,\xi} = \sum_{I^*} x_{I^*} N_{I^*,\xi}(\xi,\eta) \quad , \quad z_{,\xi} = \sum_{I^*} z_{I^*} N_{I^*,\xi}(\xi,\eta)$$

Substituting the derivatives of the shape functions with respect to ξ:

$$\boldsymbol{x}_{,\xi} = \boldsymbol{x}_1\left(\xi - \frac{1}{2}\right) + 2\boldsymbol{x}_2\xi + \boldsymbol{x}_3\left(\xi + \frac{1}{2}\right) + \eta\frac{h_1^0}{2}\boldsymbol{p}_1\left(\xi - \frac{1}{2}\right) - \eta\frac{h_2^0}{2}\boldsymbol{p}_2\xi + \eta\frac{h_3^0}{2}\boldsymbol{p}_3\left(\xi + \frac{1}{2}\right)$$

with $\boldsymbol{x}_I = x_I\boldsymbol{e}_x + +y_I\boldsymbol{e}_y + z_I\boldsymbol{e}_z$.

From which we can calculate the rotation tensor:

$$\boldsymbol{R}_{lam} = \begin{bmatrix} \boldsymbol{e}_x\cdot\hat{\boldsymbol{e}}_x & 0 & \boldsymbol{e}_x\cdot\hat{\boldsymbol{e}}_z \\ 0 & 1 & 0 \\ \boldsymbol{e}_z\cdot\hat{\boldsymbol{e}}_x & 0 & \boldsymbol{e}_z\cdot\hat{\boldsymbol{e}}_z \end{bmatrix} = \frac{1}{\left(x_{,\xi}^2 + z_{,\xi}^2\right)^{1/2}}\begin{bmatrix} x_{,\xi} & 0 & -z_{,\xi} \\ 0 & 1 & 0 \\ z_{,\xi} & 0 & x_{,\xi} \end{bmatrix}$$

The velocity field in corotational coordinates is obtained from the nodal velocities transformed by the above rotation tensor:

$$\begin{aligned}\hat{\boldsymbol{v}} = {} & \frac{1}{2}\boldsymbol{R}_{lam}^T\cdot\boldsymbol{v}_1\xi(\xi-1) + \boldsymbol{R}_{lam}^T\cdot\boldsymbol{v_2}(1-\xi^2) + \frac{1}{2}\boldsymbol{R}_{lam}^T\cdot\boldsymbol{v}_3\xi(\xi+1) \\ & + \eta\frac{h_1^0}{4}\xi(\xi-1)\boldsymbol{R}_{lam}^T\cdot\boldsymbol{\omega}_1\times\boldsymbol{p}_1 + \eta\frac{h_2^0}{2}(1-\xi^2)\boldsymbol{R}_{lam}^T\cdot\boldsymbol{\omega}_2\times\boldsymbol{p}_2 \\ & + \eta\frac{h_3^0}{4}\xi(\xi+1)\boldsymbol{R}_{lam}^T\cdot\boldsymbol{\omega}_3\times\boldsymbol{p}_3\end{aligned}$$

The rate-of-deformation can be determined once all the previous quantities are defined. First, we determine:

$$N_{I,\hat{x}}^T = N_{I,\xi}^T\hat{\boldsymbol{x}}_{,\xi}^{-1}$$

where $\hat{\boldsymbol{x}}_{,\xi}$ is determined following the same procedure used for $\hat{\boldsymbol{v}}$:

$$\begin{aligned}\boldsymbol{x}_{,\xi} = {} & \boldsymbol{R}_{lam}^T\cdot\boldsymbol{x}_1\left(\xi - \frac{1}{2}\right) + 2\boldsymbol{R}_{lam}^T\cdot\boldsymbol{x}_2\xi + \boldsymbol{R}_{lam}^T\cdot\boldsymbol{x}_3\left(\xi + \frac{1}{2}\right) + \eta\frac{h_1^0}{2}\boldsymbol{R}_{lam}^T\cdot\boldsymbol{p}_1\left(\xi - \frac{1}{2}\right) \\ & - \eta\frac{h_2^0}{2}\boldsymbol{R}_{lam}^T\cdot\boldsymbol{p}_2\xi + \eta\frac{h_3^0}{2}\boldsymbol{R}_{lam}^T\cdot\boldsymbol{p}_3\left(\xi + \frac{1}{2}\right)\end{aligned}$$

We can then calculate the velocity gradient and the rate-of-deformation in corotational coordinates:

$$\hat{\boldsymbol{L}} = \hat{\boldsymbol{v}}_I N_{I,\hat{x}} \;\Rightarrow\; \hat{\boldsymbol{D}} = \frac{\boldsymbol{1}}{\boldsymbol{2}}\left(\hat{\boldsymbol{L}} + \hat{\boldsymbol{L}}^T\right)$$

The nodal forces are given by:

$$f_I^{mast} = \begin{Bmatrix} f_{xI} \\ f_{yI} \\ m_I \end{Bmatrix}^{mast} = \boldsymbol{T}_I^T \begin{Bmatrix} \boldsymbol{f}_I^- \\ \boldsymbol{f}_I^+ \end{Bmatrix}^{slave} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ y_I - y_{I^-} & x_{I^-} - x_I & y_I - y_{I^+} & x_{I^+} - x_I \end{bmatrix} \begin{Bmatrix} f_{xI^-} \\ f_{yI^-} \\ f_{xI^+} \\ f_{yI^+} \end{Bmatrix}$$

Leading to:

$$f_{xI} = f_{xI^-} + f_{xI^+} \quad , \quad f_{yI} = f_{yI^-} + f_{yI^+}$$

$$m_I = (y_I - y_{I^-}) f_{xI^-} + (x_{I^-} - x_I) f_{yI^-} + (y_I - y_{I^+}) f_{xI^+} + (x_{I^+} - x_I) f_{yI^+}$$

Finally, we need to evaluate the angle between the pseudonormal and the true normal to the midline. The pseudonormal to the midline ($\eta = 0$) is:

$$\boldsymbol{p} = \frac{1}{2}\boldsymbol{p}_1 \xi(\xi - 1) + \boldsymbol{p}_2(1 - \xi^2) + \frac{1}{2}\boldsymbol{p}_3 \xi(\xi + 1)$$

with $\boldsymbol{p}_I = \cos\theta_I \, \boldsymbol{e}_x + \sin\theta_I \, \boldsymbol{e}_z$.

The true normal to the midline is:

$$\boldsymbol{n} = \hat{\boldsymbol{e}}_z = \frac{-z_{,\xi}\boldsymbol{e}_x + x_{,\xi}\boldsymbol{e}_z}{\left(x_{,\xi}^2 + z_{,\xi}^2\right)^{1/2}}$$

Therefore, the angle $\bar{\theta}$ between $\boldsymbol{p}$ and $\boldsymbol{n}$ is then given by:

$$\cos\bar{\theta} = \boldsymbol{p} \cdot \boldsymbol{n} \;\Rightarrow\; \bar{\theta} = \cos^{-1}\left[\frac{-z_{,\xi}\cos\theta_I + x_{,\xi}\sin\theta_I}{\left(x_{,\xi}^2 + z_{,\xi}^2\right)^{1/2}}\right]$$

9.2. Consider a plate (a flat shell) in the *x*–*y* plane governed by the Mindlin–Reissner theory. Show that the rate-of-deformation is given by

$$D_{xx} = \frac{\partial v_x^M}{\partial x} + z\frac{\partial \omega_y}{\partial x}, \quad D_{yy} = \frac{\partial v_x^M}{\partial y} - z\frac{\partial \omega_x}{\partial y}, \quad D_{xy} = \frac{1}{2}\left(\frac{\partial v_x^M}{\partial y} + \frac{\partial v_x^M}{\partial x}\right) + \frac{z}{2}\left(\frac{\partial \omega_y}{\partial y} - \frac{\partial \omega_x}{\partial x}\right)$$

$$D_{xz} = \frac{1}{2}\left(\omega_y + \frac{\partial v_z^M}{\partial x}\right), \quad D_{yz} = \frac{1}{2}\left(-\omega_x + \frac{\partial v_z^M}{\partial y}\right)$$

Solution to 9.2.

The Mindlin-Reissner theory for a flat plate leads to a simple velocity field:

$$\boldsymbol{v} = \boldsymbol{v}^M + z\boldsymbol{\omega} \times \boldsymbol{p} \;\Rightarrow\; \boldsymbol{v} = \boldsymbol{v}^M + z\begin{Bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{Bmatrix} \times \begin{Bmatrix}0\\ 0\\ 1\end{Bmatrix}$$

$$\boldsymbol{v} = \begin{Bmatrix}v_x\\ v_y\\ v_z\end{Bmatrix} = \begin{Bmatrix}v_x^M\\ v_y^M\\ v_z^M\end{Bmatrix} + z\begin{Bmatrix}\omega_y\\ -\omega_x\\ 0\end{Bmatrix}$$

Therefore, obtaining each component of the rate-of-deformation is straightforward:

$$D_{xx} = \frac{\partial v_x}{\partial x} = \frac{\partial v_x^M}{\partial x} + z\frac{\partial \omega_y}{\partial x}$$

$$D_{yy} = \frac{\partial v_y}{\partial y} = \frac{\partial v_y^M}{\partial y} - z\frac{\partial \omega_x}{\partial y}$$

$$D_{xy} = \frac{1}{2}\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right) = \frac{1}{2}\left(\frac{\partial v_x^M}{\partial y} + \frac{\partial v_y^M}{\partial x}\right) + \frac{z}{2}\left(\frac{\partial \omega_y}{\partial y} - \frac{\partial \omega_x}{\partial x}\right)$$

$$D_{xz} = \frac{1}{2}\left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}\right) = \frac{1}{2}\left(\omega_y + \frac{\partial v_z^M}{\partial x}\right)$$

$$D_{yz} = \frac{1}{2}\left(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y}\right) = \frac{1}{2}\left(-\omega_x + \frac{\partial v_z^M}{\partial y}\right)$$

9.3. Consider the lumped mass for a rectangular CB-beam element (Figure 9.17), $\hat{\mathrm{M}} = \frac{1}{8}m\mathrm{I}$, $m = \rho_0 a_0 b_0 h_0$ **where** ρ_0, a_0, b_0, **and** h_0 **are the initial density and dimensions of the rectangular continuum element underlying the beam element. Using the transformation (9.3.24), develop a mass matrix for the 2-node CB element and diagonalize the result with the row-sum technique.**

Solution to 9.3.

The transformation matrix for the 2-node CB element is:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & y_1 - y_{1-} & 0 & 0 & 0 \\ 0 & 1 & x_{1-} - x_1 & 0 & 0 & 0 \\ 1 & 0 & y_1 - y_{1+} & 0 & 0 & 0 \\ 0 & 1 & x_{1+} - x_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & y_2 - y_{2-} \\ 0 & 0 & 0 & 0 & 1 & x_{2-} - x_2 \\ 0 & 0 & 0 & 1 & 0 & y_2 - y_{2+} \\ 0 & 0 & 0 & 0 & 1 & x_{2+} - x_2 \end{bmatrix}$$

Where

$$x_{1+} = x_1 + \frac{h}{2} p_{x1}, \quad y_{1+} = y_1 + \frac{h}{2} p_{y1}$$

$$x_{1-} = x_1 - \frac{h}{2} p_{x1}, \quad y_{1-} = y_1 - \frac{h}{2} p_{y1}$$

$$x_{2+} = x_2 + \frac{h}{2} p_{x2}, \quad y_{2+} = y_2 + \frac{h}{2} p_{y2}$$

$$x_{2-} = x_2 - \frac{h}{2} p_{x2}, \quad y_{2-} = y_2 - \frac{h}{2} p_{y2}$$

With $p_{xI} = \cos\theta_I$ and $p_{yI} = \sin\theta_I$. The transformation matrix is then written as:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & \frac{h}{2} p_{y1} & 0 & 0 & 0 \\ 0 & 1 & -\frac{h}{2} p_{x1} & 0 & 0 & 0 \\ 1 & 0 & -\frac{h}{2} p_{y1} & 0 & 0 & 0 \\ 0 & 1 & \frac{h}{2} p_{x1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \frac{h}{2} p_{y2} \\ 0 & 0 & 0 & 0 & 1 & -\frac{h}{2} p_{x2} \\ 0 & 0 & 0 & 1 & 0 & -\frac{h}{2} p_{y2} \\ 0 & 0 & 0 & 0 & 1 & \frac{h}{2} p_{x2} \end{bmatrix}$$

The mass matrix for the rectangular continuum element underlying the beam element (in the expanded 8x8 form) is:

$$\widehat{\boldsymbol{M}} = \frac{\rho_0 a_0 b_0 h_0}{8} \boldsymbol{I}$$

Therefore, the mass matrix for the 2-node CB element is obtained using the transformation (9.3.24):

$$\boldsymbol{M} = \boldsymbol{T}^T\widehat{\boldsymbol{M}}\boldsymbol{T} = \frac{\rho_0 a_0 b_0 h_0}{4}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{h^2}{4}(p_{x1}^2 + p_{y1}^2) & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{h^2}{4}(p_{x2}^2 + p_{y2}^2) \end{bmatrix}$$

9.4. Develop the consistent mass matrix for a rectangular continuum element.

(a) develop a consistent mass for the CB beam using (9.3.24), i.e. $\mathbf{M} = \mathbf{T}^T\hat{\mathbf{M}}\mathbf{T}$ for a beam element lying along the *x*-axis as shown in Figure 9.17;

(b) develop the complete inertia term including the time-dependent term in (9.3.26).

Solution to 9.4a).

From example 4.2 in the book, equation (E4.2.17), we can calculate the mass matrix for a quadrilateral element and expand it to an 8x8 matrix, resulting in the following matrix:

$$\widehat{\boldsymbol{M}} = \frac{\rho_0 a_0 b_0 h_0}{72}\begin{bmatrix} 4 & 0 & 2 & 0 & 1 & 0 & 2 & 0 \\ 0 & 4 & 0 & 2 & 0 & 1 & 0 & 2 \\ 2 & 0 & 4 & 0 & 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 4 & 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 & 4 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 & 0 & 4 & 0 & 2 \\ 2 & 0 & 1 & 0 & 2 & 0 & 4 & 0 \\ 0 & 2 & 0 & 1 & 0 & 2 & 0 & 4 \end{bmatrix}$$

The transformation matrix is the same as in the previous problem, hence:

$$\boldsymbol{M} = \boldsymbol{T}^{\boldsymbol{T}}\widehat{\boldsymbol{M}}\boldsymbol{T}$$

$$= \frac{\rho_0 a_0 b_0 h_0}{72}\begin{bmatrix} 12 & 0 & 0 & 6 & 0 & 0 \\ 0 & 12 & 0 & 0 & 6 & 0 \\ 0 & 0 & h^2(p_{x1}^2 + p_{y1}^2) & 0 & 0 & -\frac{h^2}{2}(p_{x1}p_{x2} + p_{y1}p_{y2}) \\ 6 & 0 & 0 & 12 & 0 & 0 \\ 0 & 6 & 0 & 0 & 12 & 0 \\ 0 & 0 & -\frac{h^2}{2}(p_{x1}p_{x2} + p_{y1}p_{y2}) & 0 & 0 & h^2(p_{x2}^2 + p_{y2}^2) \end{bmatrix}$$

Solution to 9.4b).

From example 4.2 in the book, equation (E4.2.17), we can calculate the mass matrix for a quadrilateral element and expand it to an 8x8 matrix, resulting in the following matrix:

$$\widehat{\boldsymbol{M}} = \frac{\rho_0 a_0 b_0 h_0}{72}\begin{bmatrix} 4 & 0 & 2 & 0 & 1 & 0 & 2 & 0 \\ 0 & 4 & 0 & 2 & 0 & 1 & 0 & 2 \\ 2 & 0 & 4 & 0 & 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 4 & 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 & 4 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 & 0 & 4 & 0 & 2 \\ 2 & 0 & 1 & 0 & 2 & 0 & 4 & 0 \\ 0 & 2 & 0 & 1 & 0 & 2 & 0 & 4 \end{bmatrix}$$

The transformation matrix is the same as in the previous problem, and the time derivative of the transformation matrix is given by:

$$\dot{\boldsymbol{T}} = \begin{bmatrix} 0 & 0 & \omega_1(x_1 - x_{1-}) & 0 & 0 & 0 \\ 0 & 0 & \omega_1(y_1 - y_{1-}) & 0 & 0 & 0 \\ 0 & 0 & \omega_1(x_1 - x_{1+}) & 0 & 0 & 0 \\ 0 & 0 & \omega_1(y_1 - y_{1+}) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \omega_2(x_2 - x_{2-}) \\ 0 & 0 & 0 & 0 & 0 & \omega_2(y_2 - y_{2-}) \\ 0 & 0 & 0 & 0 & 0 & \omega_2(x_2 - x_{2+}) \\ 0 & 0 & 0 & 0 & 0 & \omega_2(y_2 - y_{2+}) \end{bmatrix}$$

$$\boldsymbol{T^T \widehat{M} \dot{T}} = \frac{\rho_0 a_0 b_0 h_0}{72} \begin{bmatrix} 6 & 0 & 6\omega_1 p_{x1} h & 6 & 0 & 3\omega_2 p_{x2} h \\ 0 & 6 & 6\omega_1 p_{y1} h & 0 & 6 & 3\omega_2 p_{x2} h \\ -p_{y1} h & p_{x1} h & 0 & 0 & 0 & 0 \\ 3 & 0 & 3\omega_1 p_{x1} h & 12 & 0 & 6\omega_2 p_{x2} h \\ 0 & 3 & 3\omega_1 p_{y1} h & 0 & 12 & 6\omega_2 p_{y2} h \\ \frac{h}{2} p_{y2} & -\frac{h}{2} p_{x2} & 0 & 0 & 0 & 0 \end{bmatrix}$$

CHAPTER 11: Extended Finite Element Method

11.1. Consider a 1D bar that has a total length of 20mm **with its left end at** -10mm **and right end at** 10mm **and a discontinuity surface at** $x_c = 0$ mm**. The bar is stretched by an external load (P=1MPa) on the left end and fixed on the right. Assuming that the bar is made of linear elastic material with material properties** $E = 200\,GPa, \rho$ = 7.83 g/cm^3**,** $\nu = 0.3$**.**

Write a 1D code to solve this problem using explicit formulation with standard Verlet time integrator. Implement XFEM to model the crack surface. To simplify the solution, no cohesive force on the crack surface needs to be considered. One of two ways of implementing XFEM can be used: the original XFEM or phantom node method. The hints below are given for implementing original XFEM.

Hints:

(a) Calculate the level set values (signed distance function to the crack surface) for each node. Note that only one element cut by the crack will be enriched in this case. It will have both positive and negative nodal level set values.

(b) Derive the B matrix for the enriched element and use the shifted enrichment function introduced in this chapter for strong discontinuity. Keep in mind that the dimension for B matrix has changed, since it now has double the degrees of freedom as before. However, the dimension of the stress shouldn't change. Keep the standard form for the B matrix for all the other unenriched elements.

(c) Integrate Bσ over the enriched element carefully to get the internal force. Since B is now a discontinuous function over the enriched domain, using the same number of gauss integration points as the unenriched elements will give poor results. There are two ways around: a) Use a lot of integration points. b) Integrate the two parts of the enriched element formed by the crack separately.

Solution of 11.1. (solved by P. D. Lea)

MATLAB code for Problem 11.1

```
%%%%%%%%%%%%%%% 1D XFEM code using a consistent mass matrix %%%%%%%%%%%%%%%%
%
%   P. D. Lea  (Patrick.Lea@u.northwestern.edu)
%   Northwestern University, Theoretical and Applied Mechanics
%
%   May 30, 2013
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [] = XFEM1D()
clc
close all
clear all
    %% Input parameters.
    nn         = 100;      % Number of nodes.
    barstart   = -10;      % Location fo the bar starting point
    barend     =  10;      % Location fo the bar ending point
    runtime    = 4.0;      % Number of times for wave to cross domain.
    youngs     = 2e5;      % Modulus of elasticity (for linear).
    rho        = 7.85e-6;  % Density per unit length.

    %%XFEM Parameters
    xc         = 0;      % Location of discontinuity

    %% Calculated parameters.
    bar_length = barend-barstart;  % Determines overall length of bar
    c      = sqrt(youngs/rho);     % Wave speed.
    t_crit = bar_length/(nn-1)/c;  % Critical timestep for lumped mass matrix of non-
cut bar.
    dt     = t_crit * .2;       % Timestep size (Needs to be reduced for XFEM and
consistent mass matrix.
    num_timesteps = ceil(runtime*bar_length/c/dt); % Number of time steps.

    %Accounting for XFEM
    ndof = nn+2;

    %% Preprocessing.
    % Nodal coordinates in reference configuration.
    X = linspace(barstart, barend, nn)';

    % Initialize nodal variables to zero.
    u  = zeros(ndof,1);
    v  = zeros(ndof,1);
    a1 = zeros(ndof,1);

    % Connectivity matrix (each row gives nodes in an element.)
    connect = [ 1:length(X)-1; 2:length(X) ]';

    % Defines the element shape functions.
    % See MATLAB help for "anonymous function" if you don't understand.
    N = @(x) ( 0.5*[1-x, 1+x] );
    % Defines the stress as a function of strain.
    stressfunction = @(eps) ( youngs * eps );
    % Defines the applied load as a function of time.
    applied_load = @(t) (-10e3 * (t<Inf).*(t>=0));
    %Level Set
```

```
    LV = @(x) (x-xc);
    %Heaviside function 1 for x>0 and 0 otherwise
    Hx = @(x) floor(heaviside(x));
    %Shifted enrichment for strong discontinuity
    Psi = @(x,xI) (Hx(LV(x)) - Hx(LV(xI)));

    %Determines value of xi for real point xp inside of element bounded by XI(1) and
XI(2)
    Getxi = @(xp,XI) ((xp-.5*(XI(1) + XI(2)))/((XI(2)-XI(1))/2));
    %Determines the value of X given xi
    GetX = @(xi,XI) ((xi*(XI(2)-XI(1)))/2 + (XI(1)+XI(2))/(2));
    %Heaviside * Shape functions
    NPsi = @(xp,XI) (0.5*[(1-xp) * Psi(GetX(xp,XI),XI(1)) ,
(xp+1)*Psi(GetX(xp,XI),XI(2))]);

    %Determine what element has a crack in it
    CrackElem = DetermineCrackedElement(X,xc,nn);
    %% Compute mass matrix.
    M = sparse(ndof,ndof);
    connCrack = (CrackElem:CrackElem+1);
    for conn = connect'
        xi = sqrt(3)/3;
        Me = 0.5*(N(xi)'*N(xi) + N(-xi)'*N(-xi));
        Me = Me * range(X(conn)) * rho;
        M(conn,conn) = M(conn,conn) + Me;
    end
    conn = (CrackElem:CrackElem+1); %Connectivity of cracked element
    qconn = (nn+1:nn+2); %connectivity of extra degrees of freedom
    lce = range(X(conn));  %length of cracked element
    XI = X(conn); %locations of nodes of element
    l1 = xc - X(CrackElem); %length from first node to crack
    l2 = lce -l1; %length from crack to second node
    xic = Getxi(xc,X(conn)); %Determine location of crack in parent coordinates
    %Calculate location values for cracked element Gauss quadrature
    xi1 = (-sqrt(3)/3)*(xic+1)/(2) + ((xic-1)/2);
    xi2 = (sqrt(3)/3)*(xic+1)/(2) + ((xic-1)/2);
    xi3 = (-sqrt(3)/3)*(1-xic)/(2) + ((xic+1)/2);
    xi4 = (sqrt(3)/3)*(1-xic)/(2) + ((xic+1)/2);
    %Numerical integration of Mass matrix
    Meqq1 = .5*l1*rho*(NPsi(xi1,XI)'*NPsi(xi1,XI) + NPsi(xi2,XI)'*NPsi(xi2,XI));
    Meqq2 = .5*l2*rho*(NPsi(xi3,XI)'*NPsi(xi3,XI) + NPsi(xi4,XI)'*NPsi(xi4,XI));
    Meqq = Meqq1 + Meqq2;
    M(qconn,qconn) = M(qconn,qconn) + Meqq;
    Meuq1 = .5*l1*rho*(N(xi1)'*NPsi(xi1,XI) + N(xi2)'*NPsi(xi2,XI));
    Meuq2 = .5*l2*rho*(N(xi3)'*NPsi(xi3,XI) + N(xi4)'*NPsi(xi4,XI));
    Meuq = Meuq1+Meuq2;
    M(conn,qconn) = M(conn,qconn) + Meuq;
    Mequ1 = .5*l1*rho*(NPsi(xi1,XI)'*N(xi1) + NPsi(xi2,XI)'*N(xi2));
    Mequ2 = .5*l2*rho*(NPsi(xi3,XI)'*N(xi3) + NPsi(xi4,XI)'*N(xi4));
    Mequ = Mequ1+Mequ2;
    M(qconn,conn) = M(qconn,conn) + Mequ;

    %% Allocate arrays for output.
    figure(1);
    u_mid = zeros(num_timesteps,1);
    P_mid = zeros(num_timesteps,1);
    u_left = zeros(num_timesteps,1);
    u_right = zeros(num_timesteps,1);
    %% Loop over time steps.
    for ts = 1:num_timesteps
        %% Displacement update.
        % Current simulation time, t.
```

```
        t = (ts-1) * dt;
        % Update displacements to time t + dt.
        u = u + dt*(v + 0.5*dt*a1);
        %% Apply boundary conditions.
        % Fix right most node.
        u(end-2) = 0;
        % Initialize nodal forces to zero.
        f = zeros(ndof,1);
        % Apply load at left end.
        f(1) = applied_load(t);
        %% Element nodal force loop.
        for conn = connect'
            lce = range(X(conn));    % Compute element reference length.
            Be = [-1, 1] / lce;      % Element B matrix (dN/dX).
            eps = Be * u(conn);
            % Evaluate constitutive law.
            P = stressfunction(eps);
            % Compute element forces with 1pt integration.
            fe = Be' * P * lce;
            f(conn) = f(conn) - fe;  % Scatter element internal forces.
        end
        % Integrating fractured element internal stress
        conn = [CrackElem:CrackElem+1];
        qconn = [nn+1:ndof];
        X1 = X(CrackElem);
        X2 = X(CrackElem+1);
        lce = range(X(conn));    % Compute element reference length.
        l1 = abs(LV(X(CrackElem))); %length from first node to crack
        l2 = lce-l1; %length from crack to second node
        % B matrices from N and N Psi
        Be = [-1,1]/lce;
        Bp1 = [-(Psi((X1+l1/2),X1)), (Psi((X1+l1/2),X2))]/lce;
        Bp2 = [-(Psi((X2-l2/2),X1)), (Psi((X2-l2/2),X2))]/lce;
        % Strains for u and q displacements
        epsq1 = Bp1*u(qconn);
        epsq2 = Bp2*u(qconn);
        eps = Be * u(conn);
        P = stressfunction(eps); %Stress from real displacements
        %Stress from q degrees of freedom
        Pq1 = stressfunction(epsq1);
        Pq2 = stressfunction(epsq2);
        %Internal stress calculation associated with XFEM degrees of freedom
        feuq = (Be' * Pq1 * l1) + (Be' * Pq2 * l2);
        fequ = (Bp1' * P * l1) + (Bp2' * P * l2);
        f(conn) = f(conn) - feuq;
        f(qconn) = f(qconn) - fequ;
        feqq = (Bp1' * Pq1 * l1) + (Bp2' * Pq2 * l2);
        f(qconn) = f(qconn) - feqq;
        %
        %% Velocity update.
        % Consistent.
        a2 = M\f;

        % Update velocities to time t + dt.
        v = v + 0.5*dt*(a1 + a2);
        % Push back a2 to a1.
        a1 = a2;
        %% Post processing
        if (mod(ts,5) == 0)
            Xe = 0.5*(X(2:end)+X(1:end-1));
            eps = (u(2:end-2)-u(1:end-3)) / (bar_length/(nn-1));
            eps(CrackElem) = 0; %zeros stress in the cut element (for post processing
only)
```

```
            P = stressfunction(eps);
            plot(Xe, P);
            pause(0.001);
        end
        eps = (u(ceil(end/4))-u(ceil(end/4)-1)) / (bar_length/(nn-1));
        P_mid(ts) = stressfunction(eps);
        u_mid(ts) = u(ceil(end/4));
        u_left(ts) = u(CrackElem);
        psin = N(1) * u(nn+1:ndof);
        u_right(ts) = u(CrackElem+1) +  Psi(X(CrackElem+1),X(CrackElem+1)) *  N(1) *
u(nn+1:ndof);

    end
    figure(2);
    t = (1:num_timesteps)'*dt;
    plot(t, u_mid);
    title('Displacement Quarter Point')
    figure(3);
    plot(t, P_mid);
    title('Quarter Point Stress')
    figure(4);
    plot(t,u_left);
    title('Displacement Left')
    figure(5);
    plot(t,u_right);
    title('Displacement Right')

end



function  [BrokenElem] = DetermineCrackedElement(X,xc,nn)
% Determines which element are cut
    for i =1:nn-1
        if(X(i) < xc && X(i+1) > xc)
            BrokenElem = i;
        end
    end
end
```

CHAPTER 12: Introduction to Multiresolution Theory

12.1 In section 12.5.3 the kinetic power is defined for MCT theory. Beginning with the definition of kinetic energy density Eq. 12.5.6, show that Eq. 12.5.7 results for the kinetic power. (*Hint*: See derivation in [38]).

Solution to 12.1.

Beginning with the expression of kinetic energy density,

$$\delta e_{kin} = \bar{\rho}_0 \mathbf{v}_0 \cdot \delta \mathbf{v}_0 + \sum\nolimits_n \tfrac{1}{V_n^a} \int_{V_n^a} \left(\bar{\rho}_n - \bar{\rho}_{n-1}\right)\left(\mathbf{v}_0 + \boldsymbol{v}_n\right) \cdot \delta\left(\mathbf{v}_0 + \boldsymbol{v}_n\right) dV$$

Take the material time derivative

$$\frac{D}{Dt}\left(\int_\Omega \delta e_{kin} d\Omega\right) = \frac{D}{Dt}\left(\int_\Omega \bar{\rho}_0 \mathbf{v}_0 \cdot \delta \mathbf{v}_0 d\Omega\right) + \frac{D}{Dt}\left(\int_\Omega \sum\nolimits_n \tfrac{1}{V_n^a} \int_{V_n^a} \left(\bar{\rho}_n - \bar{\rho}_{n-1}\right)\left(\mathbf{v}_0 + \boldsymbol{v}_n\right) \cdot \delta\left(\mathbf{v}_0 + \boldsymbol{v}_n\right) dV d\Omega\right)$$

We now focus on each term from the RHS separately:

A- Focusing on the first term of the right hand side,

$$\begin{aligned}
&\frac{D}{Dt}\left(\int_\Omega \bar{\rho}_0 \mathbf{v}_0 \cdot \delta \mathbf{v}_0 d\Omega\right) = \\
&= \int_\Omega \left(\dot{\bar{\rho}}_0 + \nabla \cdot \mathbf{v}_0 \bar{\rho}_0\right)(\mathbf{v}_0 \cdot \delta \mathbf{v}_0) d\Omega + \int_\Omega \bar{\rho}_0 \left(\frac{\partial}{\partial t}\left(\mathbf{v}_0 \cdot \delta \mathbf{v}_0\right) + \nabla \cdot \mathbf{v}_0 (\mathbf{v}_0 \cdot \delta \mathbf{v}_0)\right) d\Omega \\
&= 0 + \int_\Omega \bar{\rho}_0 \left(\dot{\mathbf{v}}_0 \cdot \delta \mathbf{v}_0 + \mathbf{v}_0 \cdot \delta \dot{\mathbf{v}}_0 + \left(\nabla \cdot \mathbf{v}_0\right)(\mathbf{v}_0 \cdot \delta \mathbf{v}_0)\right) d\Omega \\
&= 0 + \int_\Omega \bar{\rho}_0 \left(\dot{\mathbf{v}}_0 \cdot \delta \mathbf{v}_0 + \mathbf{v}_0 \cdot \left(\delta \dot{\mathbf{v}}_0 + \delta \mathbf{v}_0 \left(\nabla \cdot \mathbf{v}_0\right)\right)\right) d\Omega
\end{aligned}$$

We wish to drop the term containing $\delta\dot{\mathbf{v}}_0$ to simplify the resulting expressions. To do so, we notice that for small increments of time τ, using integration by parts

$$\int_t^{t+\tau}\int_\Omega \bar{\rho}_0\mathbf{v}_0\cdot\left(\delta\dot{\mathbf{v}}_0(\mathbf{x})\right)d\Omega dt = \int_t^{t+\tau}\int_\Omega \frac{1}{\tau}\left[\bar{\rho}_0\mathbf{v}_0\cdot\delta\mathbf{v}_0(\mathbf{x})\right]_t^{t+\tau}d\Omega dt$$
$$-\int_t^{t+\tau}\int_\Omega\left(\frac{d}{dt}\left(\bar{\rho}_0\mathbf{v}_0\right)\right)\cdot\delta\mathbf{v}_0(\mathbf{x})d\Omega dt$$

The variation $\delta\mathbf{v}_0$ can be assumed roughly constant over interval τ. We will thus use its mid-interval value $\delta\mathbf{v}_0^{t+t/2}$ to approximate the time integral,

$$\int_t^{t+\tau}\int_\Omega \bar{\rho}_0\mathbf{v}_0\cdot\left(\delta\dot{\mathbf{v}}_0(\mathbf{x})\right)d\Omega dt \cong \int_t^{t+\tau}\int_\Omega \frac{1}{\tau}\left[\bar{\rho}_0\mathbf{v}_0\right]_t^{t+\tau}\cdot\delta\mathbf{v}_0^{t+\tau/2}(\mathbf{x})d\Omega dt$$
$$-\int_t^{t+\tau}\int_\Omega\left(\frac{d}{dt}\left(\bar{\rho}_0\mathbf{v}_0\right)\right)\cdot\delta\mathbf{v}_0^{t+\tau/2}(\mathbf{x})d\Omega dt$$
$$=\int_t^{t+\tau}\int_\Omega\left(\frac{1}{\tau}\left[\bar{\rho}_0\mathbf{v}_0\right]_t^{t+\tau}-\frac{d}{dt}\left(\bar{\rho}_0\mathbf{v}_0\right)\right)\cdot\delta\mathbf{v}_0^{t+\tau/2}(\mathbf{x})d\Omega$$

Since τ was assumed to be small and arbirtrary, if we exclude sudden jumps in momentum, we find the quantity in the parentheses is zero to the first order, thus the volume integral must be zero, so that the term may be dropped as desired. Thus, the material rate becomes,

$$\frac{D}{Dt}\left(\int_\Omega\bar{\rho}_0\mathbf{v}_0\cdot\delta\mathbf{v}_0d\Omega\right)=\int_\Omega\bar{\rho}_0\left(\dot{\mathbf{v}}_0\cdot\delta\mathbf{v}_0+\left(\mathbf{v}_0\cdot\delta\mathbf{v}_0\right)\left(\nabla\cdot\mathbf{v}_0\right)\right)d\Omega$$
$$=\int_\Omega\bar{\rho}_0\left(\dot{\mathbf{v}}_0+\mathbf{v}_0\left(\nabla\cdot\mathbf{v}_0\right)\right)\cdot\delta\mathbf{v}_0d\Omega$$
$$\frac{D}{Dt}\left(\int_\Omega\bar{\rho}_0\mathbf{v}_0\cdot\delta\mathbf{v}_0d\Omega\right)=\int_\Omega\bar{\rho}_0\frac{D\mathbf{v}_0}{Dt}\cdot\delta\mathbf{v}_0d\Omega$$

In an updated Lagrangian formulation, $\frac{D\mathbf{v}_0(\mathbf{x})}{Dt}=\dot{\mathbf{v}}_0(f(\mathbf{X},t))$, so

$$\frac{D}{Dt}\left(\int_\Omega \bar{\rho}_0 \mathbf{v}_0 \cdot \delta\mathbf{v}_0 d\Omega\right) = \int_\Omega \bar{\rho}_0 \dot{\mathbf{v}}_0 \cdot \delta\mathbf{v}_0 d\Omega$$

B- Focusing on the second term from the RHS,

We define $\boldsymbol{v}_n = \mathbf{l}_n \times \mathbf{y}_n$. The material time derivative is thus,

$$\begin{aligned}
&\frac{D}{Dt}\left(\int_\Omega \sum_n \tfrac{1}{V_n^a}\int_{V_n^a}\left(\bar{\rho}_n - \bar{\rho}_{n-1}\right)\left(\mathbf{v}_0 + \boldsymbol{v}_n\right)\cdot\delta\left(\mathbf{v}_0 + \boldsymbol{v}_n\right) dV d\Omega\right. \\
&= \int_\Omega \sum_n \tfrac{1}{V_n^a}\int_{V_n^a}\left(\bar{\rho}_n - \bar{\rho}_{n-1}\right)\left(\dot{\mathbf{v}}_0 + \left(\dot{\mathbf{l}}_n + \mathbf{l}_n\cdot\mathbf{l}_n\right)\cdot\mathbf{y}_n\right)\cdot\delta\left(\mathbf{v}_0 + \mathbf{l}_n\cdot\mathbf{y}_n\right) dV d\Omega
\end{aligned}$$

$$\begin{aligned}
&= \int_\Omega \sum_n \tfrac{1}{V_n^a}\int_{V_n^a}\left(\bar{\rho}_n - \bar{\rho}_{n-1}\right)\left(\dot{\mathbf{v}}_0\right) dV\cdot\left(\delta\mathbf{v}_0\right) d\Omega \\
&+ \int_\Omega \sum_n \tfrac{1}{V_n^a}\int_{V_n^a}\left(\bar{\rho}_n - \bar{\rho}_{n-1}\right)\left(\dot{\mathbf{l}}_n + \mathbf{l}_n\cdot\mathbf{l}_n\right)\cdot\mathbf{y}_n dV\cdot\left(\delta\mathbf{v}_0\right) d\Omega \\
&+ \int_\Omega \sum_n \tfrac{1}{V_n^a}\int_{V_n^a}\left(\bar{\rho}_n - \bar{\rho}_{n-1}\right)\mathbf{y}_n dV\cdot\left(\dot{\mathbf{v}}_0\right)\cdot\left(\delta\mathbf{l}_n\right) d\Omega \\
&+ \int_\Omega \sum_n \tfrac{1}{V_n^a}\int_{V_n^a}\left(\dot{\mathbf{l}}_n + \mathbf{l}_n\cdot\mathbf{l}_n\right)\cdot\left(\bar{\rho}_n - \bar{\rho}_{n-1}\right)\left(\mathbf{y}_n\otimes\mathbf{y}_n\right) dV : \left(\delta\mathbf{l}_n\right) d\Omega
\end{aligned}$$

Noting that $\frac{1}{V_n^a}\int_{V_n^a}\left(\bar{r}_n - \bar{r}_{n-1}\right)\mathbf{y}_n\, dV = 0,$ since the integrand is an odd function. Hence, the material derivative

$$\begin{aligned}
&= \int_\Omega \sum_n \tfrac{1}{V_n^a}\int_{V_n^a}\left(\bar{\rho}_n - \bar{\rho}_{n-1}\right)\left(\dot{\mathbf{v}}_0\right) dV\cdot\left(\delta\mathbf{v}_0\right) d\Omega \\
&+ \int_\Omega \sum_n \tfrac{1}{V_n^a}\int_{V_n^a}\left(\dot{\mathbf{l}}_n + \mathbf{l}_n\cdot\mathbf{l}_n\right)\cdot\left(\bar{\rho}_n - \bar{\rho}_{n-1}\right)\left(\mathbf{y}_n\otimes\mathbf{y}_n\right) dV : \left(\delta\mathbf{l}_n\right) d\Omega
\end{aligned}$$

$$= \int_\Omega \sum_n \left(\bar{\rho}_n - \bar{\rho}_{n-1}\right)\left(\dot{\mathbf{v}}_0\right)\cdot\left(\delta\mathbf{v}_0\right) d\Omega + \int_\Omega \sum_n \alpha_n \cdot \mathrm{P}_n : \left(\delta\mathbf{l}_n\right) d\Omega$$

Where the last equation follows from the definitions of α and P given in chapter 12. Combining the results of (A) and (B) yields Eq. 12.5.7.

12.2 Using the assumptions outlined in Section 12.5.4, derive the MCT strong form Eq. 12.5.8 and Eq. 12.5.9. (*Hint*: See derivation in [38]).

Solution to 12.2.

The main effort lies in re-writing the internal power expression in terms of variations of the degrees of freedom only, by getting rid of the variations in their gradients using integration by parts and the divergence theorem. Thus,

$$\delta P_{\text{int}} = \int_{\Omega} \boldsymbol{\sigma}_0 : \delta \mathbf{L}_0 \mathrm{d}\Omega + \int_{\Omega} \sum_{n=1}^{N} \left(\mathbf{s}_n : \delta \mathbf{l}_n + \mathbf{ss}_n \vdots \delta \nabla_{\mathbf{x}} \mathbf{L}_n \right) \mathrm{d}\Omega$$

Focusing on the first term on the RHS,

$$\int_{\Omega} \boldsymbol{\sigma}_0 : \delta \mathbf{L}_0 \mathrm{d}\Omega = \int_{\Omega} \boldsymbol{\sigma}_0 : \delta \nabla \mathbf{v}_0 \mathrm{d}\Omega = \int_{\Gamma} \left(\boldsymbol{\sigma}_0 \cdot \mathbf{n} \right) \cdot \delta \mathbf{v}_0 d\Gamma - \int_{\Omega} \left(\boldsymbol{\sigma}_0 \cdot \nabla_{\mathbf{x}} \right) \cdot \delta \mathbf{v}_0 \mathrm{d}\Omega$$

Focusing on the second term,

$$\begin{aligned}
&\int_{\Omega} \sum_{n=1}^{N} \left(\mathbf{s}_n : \delta \mathbf{l}_n + \mathbf{ss}_n \vdots \delta \nabla_{\mathbf{x}} \mathbf{L}_n \right) \mathrm{d}\Omega \\
&\quad = \sum_{n=1}^{N} \left(\int_{\Omega} \mathbf{s}_n : \delta \left(\mathbf{L}_n - \mathbf{L}_0 \right) \mathrm{d}\Omega + \int_{\Omega} \mathbf{ss}_n \vdots \delta \nabla_{\mathbf{x}} \mathbf{L}_n \mathrm{d}\Omega \right) \\
&\quad = \sum_{n=1}^{N} \left(\begin{array}{l} \int_{\Omega} \mathbf{s}_n : \delta \mathbf{L}_n \mathrm{d}\Omega - \left(\int_{\Gamma} \left(\mathbf{s}_n \cdot \mathbf{n} \right) \cdot \delta \mathbf{v}_0 \mathrm{d}\Gamma - \int_{\Omega} \left(\mathbf{s}_n \cdot \nabla_{\mathbf{x}} \right) \cdot \delta \mathbf{v}_0 \mathrm{d}\Omega \right) \\ + \left(\int_{\Gamma} \left(\mathbf{ss}_n \cdot \mathbf{n} \right) : \delta \mathbf{L}_n \mathrm{d}\Gamma - \int_{\Omega} \left(\mathbf{ss}_n \cdot \nabla_{\mathbf{x}} \right) : \delta \mathbf{L}_n \mathrm{d}\Omega \right) \end{array} \right)
\end{aligned}$$

Thus the variation of internal power is given by,

$$\delta P_{\text{int}} = \int_\Gamma \left(\boldsymbol{\sigma}_0 \cdot \mathbf{n}\right) \cdot \delta \mathbf{v}_0 d\Gamma - \int_\Omega \left(\boldsymbol{\sigma}_0 \cdot \nabla_{\mathbf{x}}\right) \cdot \delta \mathbf{v}_0 \mathrm{d}\Omega$$
$$+ \sum_{n=1}^{N} \left(\begin{array}{l} \int_\Omega \mathbf{s}_n : \delta \mathbf{L}_n \mathrm{d}\Omega - \left(\int_\Gamma \left(\mathbf{s}_n \cdot \mathbf{n}\right) \cdot \delta \mathbf{v}_0 \mathrm{d}\Gamma - \int_\Omega \left(\mathbf{s}_n \cdot \nabla_{\mathbf{x}}\right) \cdot \delta \mathbf{v}_0 \mathrm{d}\Omega \right) \\ + \left(\int_\Gamma \left(\mathbf{s}\mathbf{s}_n \cdot \mathbf{n}\right) : \delta \mathbf{L}_n \mathrm{d}\Gamma - \int_\Omega \left(\mathbf{s}\mathbf{s}_n \cdot \nabla_{\mathbf{x}}\right) : \delta \mathbf{L}_n \mathrm{d}\Omega \right) \end{array} \right)$$

We now turn our attention to the variations of kinetic and external powers, and use the simplifying assumption $\mathbf{l}^n = \mathbf{L}^n$ and $\delta\mathbf{l}^n = \delta\mathbf{L}^n$ for the case of large deformations, then

$$\delta P_{\mathbf{ext}} \cong \int_\Omega \left(\mathbf{b} \cdot \delta \mathbf{v}_0 + \sum_{n=1}^{N} \mathbf{B}_n : \delta \mathbf{L}_n \right) \mathrm{d}\Omega + \int_\Gamma \left(\mathbf{t} \cdot \delta \mathbf{v}_0 + \sum_{n=1}^{N} \mathbf{T}_n : \delta \mathbf{L}_n \right) \mathrm{d}\Gamma$$

$$\delta P_{\mathbf{kin}} \cong \int_\Omega \left(\rho \dot{\mathbf{v}}_0 \cdot \delta \mathbf{v}_0 + \sum_{n=1}^{N} \boldsymbol{\alpha}_n \cdot \mathbf{P}_n : \delta \mathbf{L}_n \right) \mathrm{d}\Omega$$

Finally, we set

$$\delta P_{\mathbf{ext}} - \delta P_{\text{int}} = \delta P_{\mathbf{kin}}$$

Noting that all variations are arbitrary, and must vanish at Dirichlet boundaries, the strong forms 12.5.8 and 12.5.9 result.

12.3 Show that the stress rate $\overset{\nabla}{\mathbf{s}}_n$ and double stress rate $\overset{\nabla}{\mathbf{ss}}_n$ in Eq. 12.6.4 are objective.

Solution to 12.3

(A) Focusing on $\overset{\nabla}{\mathbf{s}}_n$.

We begin by relating $\mathbf{s}_n$ to $\mathbf{s}^*_n$ in a starred coordinate system,

$$\mathbf{s}^*_n = \mathbf{Q}\mathbf{s}_n\mathbf{Q}^T$$

Where **Q** is the rotation matrix between the starred and un-starred coordinates. Taking the material derivative

$$\dot{\mathbf{s}}_n^* = \dot{\mathbf{Q}}\mathbf{s}_n\mathbf{Q}^T + \mathbf{Q}\dot{\mathbf{s}}_n\mathbf{Q}^T + \mathbf{Q}\mathbf{s}_n\dot{\mathbf{Q}}^T$$

Recall that

$$\dot{\mathbf{Q}} = \mathbf{W}^*\mathbf{Q} - \mathbf{Q}\mathbf{W}$$
$$\dot{\mathbf{Q}}^T = -\mathbf{Q}^T\mathbf{W}^* + \mathbf{W}\mathbf{Q}^T$$

Then,

$$\dot{\mathbf{s}}_n^* = \left(\mathbf{W}^*\mathbf{Q} - \mathbf{Q}\mathbf{W}\right)\mathbf{s}_n\mathbf{Q}^T + \mathbf{Q}\dot{\mathbf{s}}_n\mathbf{Q}^T + \mathbf{Q}\mathbf{s}_n\left(-\mathbf{Q}^T\mathbf{W}^* + \mathbf{W}\mathbf{Q}^T\right)$$
$$\dot{\mathbf{s}}_n^* = \left(\mathbf{W}^*\mathbf{Q}\mathbf{s}_n\mathbf{Q}^T - \mathbf{Q}\mathbf{W}\mathbf{s}_n\mathbf{Q}^T\right) + \mathbf{Q}\dot{\mathbf{s}}_n\mathbf{Q}^T + \left(-\mathbf{Q}\mathbf{s}_n\mathbf{Q}^T\mathbf{W}^* + \mathbf{Q}\mathbf{s}_n\mathbf{W}\mathbf{Q}^T\right)$$
$$\dot{\mathbf{s}}_n^* = \left(\mathbf{W}^*\mathbf{s}_n^* - \mathbf{Q}\mathbf{W}\mathbf{s}_n\mathbf{Q}^T\right) + \mathbf{Q}\dot{\mathbf{s}}_n\mathbf{Q}^T + \left(-\mathbf{s}_n^*\mathbf{W}^* + \mathbf{Q}\mathbf{s}_n\mathbf{W}\mathbf{Q}^T\right)$$
$$\dot{\mathbf{s}}_n^* - \mathbf{W}^*\mathbf{s}_n^* + \mathbf{s}_n^*\mathbf{W}^* = \mathbf{Q}\left(\dot{\mathbf{s}}_n - \mathbf{W}\mathbf{s}_n + \mathbf{s}_n\mathbf{W}\right)\mathbf{Q}^T$$

Thus, stress rate $\overset{\nabla}{\mathbf{s}}_n = \left(\dot{\mathbf{s}}_n - \mathbf{W}\mathbf{s}_n + \mathbf{s}_n\mathbf{W}\right)$ transforms as a second rank Eulerian tensor and is objective.

(B) Focusing on $\overset{\nabla}{\mathbf{ss}}_n$. We use index notation for clarity. Relating starred to un-starred components we find,

$$\left(ss^*{}_n\right)_{ijk} = Q_{il}Q_{jm}Q_{kn}\left(ss{}_n\right)_{lmn}$$

Taking the material derivative,

$$\left(\dot{ss}^{*}_{n}\right)_{ijk} = \dot{Q}_{il}Q_{jm}Q_{kn}\left(ss_n\right)_{lmn} + Q_{il}\dot{Q}_{jm}Q_{kn}\left(ss_n\right)_{lmn} + Q_{il}Q_{jm}\dot{Q}_{kn}\left(ss_n\right)_{lmn} + Q_{il}Q_{jm}Q_{kn}\left(\dot{ss}_n\right)_{lmn}$$

Given that the rotation rates can be written as,

$$\begin{aligned}\dot{Q}_{il} &= W^{*}_{ip}Q_{pl} - Q_{ip}W_{pl}\\ \dot{Q}_{jm} &= W^{*}_{jp}Q_{pm} - Q_{jp}W_{pm}\\ \dot{Q}_{kn} &= W^{*}_{kp}Q_{pn} - Q_{kp}W_{pn}\end{aligned}$$

We find the double stress rate is,

$$\begin{aligned}\left(\dot{ss}^{*}_{N}\right)_{ijk} &= \left(W^{*}_{ip}Q_{pl} - Q_{ip}W_{pl}\right)Q_{jm}Q_{kn}\left(ss_{N}\right)_{lmn}\\ &+ Q_{il}\left(W^{*}_{jp}Q_{pm} - Q_{jp}W_{pm}\right)Q_{kn}\left(ss_{N}\right)_{lmn}\\ &+ Q_{il}Q_{jm}\left(W^{*}_{kp}Q_{pn} - Q_{kp}W_{pn}\right)\left(ss_{N}\right)_{lmn}\\ &+ Q_{il}Q_{jm}Q_{kn}\left(\dot{ss}_n\right)_{lmn}\end{aligned}$$

Plugging in the for $\left(ss^{*}_{n}\right)_{ijk} = Q_{il}Q_{jm}Q_{kn}\left(ss_{n}\right)_{lmn}$ we find,

$$\begin{aligned}\left(\dot{ss}^{*}_{N}\right)_{ijk} &= \left(W^{*}_{ip}\left(ss^{*}_{n}\right)_{pjk} - Q_{ip}W_{pl}Q_{jm}Q_{kn}\left(ss_{N}\right)_{lmn}\right)\\ &+ \left(W^{*}_{jp}\left(ss^{*}_{n}\right)_{ipk} - Q_{il}Q_{jp}W_{pm}Q_{kn}\left(ss_{N}\right)_{lmn}\right)\\ &+ \left(W^{*}_{kp}\left(ss^{*}_{n}\right)_{ijp} - Q_{il}Q_{jm}Q_{kp}W_{pn}\left(ss_{N}\right)_{lmn}\right)\\ &+ Q_{il}Q_{jm}Q_{kn}\left(\dot{ss}_n\right)_{lmn}\end{aligned}$$

Relabeling repeated indices as appropriate,

$$\left(\overset{*}{\dot{ss}}_N\right)_{ijk} - W^*_{ip}\left(ss^*_n\right)_{pjk} - W^*_{jp}\left(ss^*_n\right)_{ipk} - W^*_{kp}\left(ss^*_n\right)_{ijp}$$
$$= \left(-Q_{il}Q_{jm}Q_{kn}W_{lo}\left(ss_N\right)_{omn} - Q_{il}Q_{jm}Q_{kn}W_{mo}\left(ss_N\right)_{lon} - Q_{il}Q_{jm}Q_{kn}W_{no}\left(ss_N\right)_{lmo}\right)$$
$$= Q_{il}Q_{jm}Q_{kn}\left(\left(\dot{ss}_n\right)_{lmn} - W_{lo}\left(ss_N\right)_{omn} - W_{mo}\left(ss_N\right)_{lon} - W_{no}\left(ss_N\right)_{lmo}\right)$$

Thus, stress rate $\overset{\nabla}{\mathbf{ss}}_n$ transforms as a third rank Eulerian tensor and is objective.

12.4 (a) What condition on the stresses and double stresses would permit rewriting the variation of MCT internal power as,

$$\delta P_{\text{int}} = \int_\Omega \left(\boldsymbol{\sigma}_0 : \delta\mathbf{D}_0 + \sum\nolimits_{n=1}^{N}\left(\mathbf{s}_n : \delta\mathbf{d}_n + \mathbf{ss}_n \vdots \delta\nabla_{\mathbf{x}}\mathbf{D}_n\right)\right)\mathrm{d}\Omega,$$

where $\mathbf{d}_n = \mathbf{D}_n - \mathbf{D}_0$**.**

(b) What savings in degrees of freedom per node could be achieved as a result?

Solution to 12.4a).

First we re-write the internal power as,

$$\delta P_{\text{int}} = \int_\Omega \left(\boldsymbol{\sigma}_0 : \delta(\mathbf{D}_0 + \mathbf{W}_0) + \sum\nolimits_{n=1}^{N}\left(\mathbf{s}_n : \delta\left(\mathbf{d}_n + \mathbf{w}_n\right) + \mathbf{ss}_n \vdots \delta\nabla_{\mathbf{x}}(\mathbf{D}_n + \mathbf{W}_n)\right)\right)\mathrm{d}\Omega$$

If $\boldsymbol{\sigma}_0$, $\mathbf{s}_n$ are symmetric, and $\mathbf{ss}_n$ is symmetric in its first two indices, then

for the term,

$$\boldsymbol{\sigma}_0 : \delta \mathbf{W}_0 = (\sigma_0)_{ij} (W_0)_{ij} = (\sigma_0)_{11} (W_0)_{11} + (\sigma_0)_{22} (W_0)_{22} + (\sigma_0)_{33} (W_0)_{33}$$
$$(\sigma_0)_{12} (W_0)_{12} + (\sigma_0)_{23} (W_0)_{23} + (\sigma_0)_{31} (W_0)_{31}$$
$$(\sigma_0)_{12} (W_0)_{21} + (\sigma_0)_{23} (W_0)_{32} + (\sigma_0)_{31} (W_0)_{13}$$
$$= 0$$

Same for $\mathbf{s}_n : \delta \mathbf{w}_n = 0$,

Now $\mathbf{ss}_n \vdots \delta \nabla_{\mathbf{x}} \mathbf{W}_n = (ss_n)_{ijk} (W_n)_{ij,k}$,

For k = 1,2,3

$$= (ss_n)_{11k} (W_n)_{11,k} + (ss_n)_{22,k} (W_n)_{22,k} + (ss_n)_{33k} (W_n)_{33,k}$$
$$(ss_n)_{12k} (W_n)_{12,k} + (ss_n)_{23,k} (W_n)_{23,k} + (ss_n)_{31k} (W_n)_{31,k}$$
$$(ss_n)_{12k} (W_n)_{21,k} + (ss_n)_{23,k} (W_n)_{32,k} + (ss_n)_{31k} (W_n)_{13,k}$$
$$= 0$$

Finally, the internal power could be written as,

$$\delta P_{\text{int}} = \int_{\Omega} \left(\boldsymbol{\sigma}_0 : \delta(\mathbf{D}_0) + \sum_{n=1}^{N} \left(\mathbf{s}_n : \delta(\mathbf{d}_n) + \mathbf{ss}_n \vdots \delta \nabla_{\mathbf{x}} (\mathbf{D}_n) \right) \right) \mathrm{d}\Omega$$

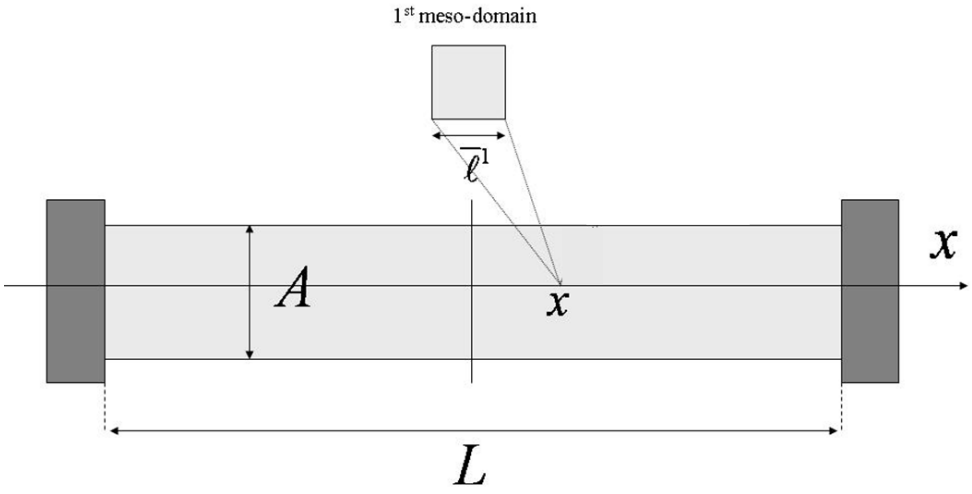
Remark: This assumption ignores Micropolar effects.

Solution to 12.4b).

The savings in degrees of freedom per node would be in three dimensions computed from:

Saving = 1 - (6N+3)/(9N+3)

12.5 Computer project: Write a code for the finite element implementation of a two-scale MCT model (i.e., a micromorphic continuum with a macro- and micro-scale), applied to a one-dimensional rod, using explicit time stepping.



The 1D bar is subjected to uniaxial tension by applying on both ends a constant velocity of $5mm/ms$**. Consider a constant cross sectional area of** $A = 30\,mm^2$ **and initial length** $L = 100\,mm$**, as shown in the figure above. The origin of the coordinate system lies in the center of the bar. The body forces and body couples are considered negligible for this problem. The macro-scale is defined as elastic and perfectly plastic with a yield stress of** 250 MPa**, while the micro-scale is considered linear elastic. To induce localized deformation, set the yield strength in the middle element of your mesh to have a yield stress of** 237.5 MPa **(imperfection of** 5%**). The micro-scale is considered to be a cubic cell with length** $l^{(1)} = 4.04mm$**. The remaining properties that need to be considered are:**

	Density (ρ) (g/mm^3)	Young's modulus (E) (GPa)
Macro-scale	7.85e-3	200
Micro-scale	7.065e-3	2

In order to write the FEM code to solve this problem consider doing the following:

(a) Write the equations of motion (the strong form) and explain the meaning of each variable.

(b) State the constitutive equations used in the code for each set of degrees-of-freedom.

(c) Write the weak form of the problem and discretize it such that it can be solved in a finite element procedure.

(d) Write the FEM code and an outline of the algorithm.

(e) Plot the different stress and strain measures over the entire bar for the parameters given in this problem statement. Compare the results with the classic continuum case.

(f) Evaluate the influence on the result of different input parameters.

Solution to 12.5a).

Strong form:

The equations of motion are derived from the conservation of linear momentum and the conservation of angular momentum. The multiresolution continuum equations of motion for this problem coincide with the equations of a micromorphic continuum since only one extra scale is considered. The strong form can then be written as follows, noting that no body forces or double body forces are present:

$$\frac{d}{dx^{(0)}}\left(\sigma_x^{(0)} - \sigma_x^{(1)}\right) = \rho^{(0)}\dot{v}_x^{(0)}, \qquad in\ -\frac{L}{2} < x^{(0)} < \frac{L}{2}$$

$$\frac{d}{dx^{(0)}}\left(\sigma\sigma_x^{(1)}\right) + \sigma_x^{(1)} = I_{xx}^{(1)}\gamma_x^{(1)}, \qquad in\ -\frac{L}{2} < x^{(0)} < \frac{L}{2}$$

where $\sigma_x^{(0)}$, $\sigma_x^{(1)}$ and $\sigma\sigma_x^{(1)}$ are the macro-stress, moicro-stress and double micro-stress, respectively. $\rho^{(0)}$ and $\rho^{(1)}$ are the macro-density and micro-density, $I_{xx}^{(1)} = \rho^{(1)}\frac{\left(l^{(1)}\right)^2}{6}$ is the moment of inertia along x per unit micro-volume (cube with length $l^{(1)}$). $\dot{v}_x^{(0)}$ is the macro-acceleration, and $\gamma_x^{(1)}$ is the micro-acceleration defined by:

$$\gamma_x^{(1)} = \dot{L}_x^{(1)} + L_x^{(1)}L_x^{(1)} = \dot{D}_x^{(1)} + D_x^{(1)}D_x^{(1)}$$

Note that since this is a 1D problem there are no couple stresses (moments per unit area) and there is no spin tensor, leaving just the rate-of-deformation to be equal to the velocity gradient. Also note that the double stress $\sigma\sigma_x^{(1)}$ is a couple stress **without moment**, i.e. a pair of stresses acting along the same line. This measure of stress is characteristic of the micromorphic

continuum and does not exist in the classic continuum neither in the Cosserat continuum. Rather, it is a higher order stress that is often used in strain gradient plasticity formulations. Therefore, its existence in this formulation is expected to regularize the localization of the deformation at the center of the bar (as will be verified with the results).

The boundary conditions are:

$$v_x^{(0)} = -5, \quad D_x^{(1)} = 0 \qquad for\ x^{(0)} = -\frac{L}{2}$$

$$v_x^{(0)} = 5, \quad D_x^{(1)} = 0 \qquad for\ x^{(0)} = \frac{L}{2}$$

Solution to 12.5b).

Constitutive equations:

The generalized stresses and strain measures are gathered into a single vector as follows

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma^{(0)} & \sigma^{(1)} & \sigma\sigma^{(1)} \end{bmatrix}$$

$$\mathbf{\Delta} = \begin{bmatrix} D_x^{(0)} & D_x^{(1)} & \dfrac{\partial D_x^{(1)}}{\partial x^{(0)}} \end{bmatrix}$$

Since the problem of interest is 1-dimensional, there is no need to define a rotationally invariant stress measure. The generalized constitutive law in rate form becomes:

$$\frac{d}{dt}\begin{bmatrix} \sigma^{(0)} \\ \sigma^{(1)} \\ \sigma\sigma^{(1)} \end{bmatrix} = \begin{bmatrix} \bar{C}^{(0)} & 0 & 0 \\ 0 & \bar{C}^{(1)} & \bar{B}^{(1)} \\ 0 & \bar{B}^{(1)} & \bar{\bar{C}}^{(1)} \end{bmatrix}\begin{bmatrix} D^{(0)} \\ D^{(1)} \\ D_{,x}^{(1)} \end{bmatrix}$$

The constitutive constants for the micro-scale are obtained by averaging procedures (recall that the micro-cell is considered as a cube of length $l^{(1)}$ in this problem):

$$\bar{C}^{(1)} = \frac{1}{(l^{(1)})^3}\int_{\Omega^{(1)}} E^{(1)}\, d\Omega^{(1)} = E^{(1)}$$

$$\bar{B}^{(1)} = \frac{1}{(l^{(1)})^3}\int_{-\frac{l^{(1)}}{2}}^{\frac{l^{(1)}}{2}} E^{(1)}x^{(1)}\, d\mathrm{x}^{(1)}\left(l^{(1)}\right)^2 = 0$$

$$\bar{\bar{C}}^{(1)} = \frac{1}{(l^{(1)})^3}\int_{-\frac{l^{(1)}}{2}}^{\frac{l^{(1)}}{2}} E^{(1)}x^{(1)}x^{(1)}\, d\mathrm{x}^{(1)}\left(l^{(1)}\right)^2 = E^{(1)}\frac{\left(l^{(1)}\right)^2}{6}$$

A simple perfect plasticity law defined by the following yield surface is defined

$$\Phi(\sigma^0) = |\sigma^0| - \sigma_y\,,$$

where σ_y is the yield stress.

<u>*Solution to 12.5c).*</u>

Weak form and FEM discretization:

Obtaining the weak form from the above written strong form is very similar to what we did for the classic continuum. For the conservation of linear momentum, we multiply both sides of the equation for the virtual velocity $\delta v_x^{(0)}$, and after using the chain rule and the divergence theorem we obtain:

$$\int_{\Omega^{(0)}}\left(\sigma + \sigma^{(1)}\right)\frac{d\left(\delta v_x^{(0)}\right)}{dx^{(0)}}d\Omega^{(0)} + \int_{\Omega^{(0)}}[\rho^{(0)}\dot{v}_x^{(0)}\delta v_x^{(0)}\frac{d\left(\delta v_x^{(0)}\right)}{dx^{(0)}}d\Omega^{(0)} = 0$$

For the conservation of angular momentum the procedure is similar in every way, with the difference that the power conjugate is now the relative micro velocity gradient (note that the relative micro velocity gradient is equal to the total micro velocity gradient subtracted with the macro velocity gradient). Therefore, we multiply both sides of the equation by a virtual relative micro velocity gradient $\delta L_x^{rel^{(1)}} = \delta D_x^{rel^{(1)}}$, with "rel" standing for relative. Then using the chain rule and the divergence theorem we get:

$$\int_{\Omega^{(0)}} [\sigma\sigma^{(1)} \frac{d\left(\delta D_x^{rel^{(1)}}\right)}{dx^{(0)}} d\Omega^{(0)} + \int_{\Omega^{(0)}} \sigma^{(1)} \delta D_x^{rel^{(1)}} d\Omega^{(0)} + \int_{\Omega^{(0)}} I^{(1)} \gamma_x^{(1)} \delta D_x^{rel^{(1)}} d\Omega^{(0)} = 0$$

Writing the relative micro velocity gradient as: $D_x^{rel^{(1)}} = D_x^{(1)} - D_x^{(0)}$, and assuming that $\frac{d\left(\delta D_x^{(0)}\right)}{dx^{(0)}} \approx 0$, we can write the above equation as:

$$\int_{\Omega^{(0)}} [\sigma\sigma^{(1)} \frac{d\left(\delta D_x^{(1)}\right)}{dx^{(0)}} d\Omega^{(0)} + \int_{\Omega^{(0)}} \sigma^{(1)} \left(\delta D_x^{(1)} - \delta D_x^{(0)}\right) d\Omega^{(0)} + \int_{\Omega^{(0)}} I^{(1)} \gamma_x^{(1)} \left(\delta D_x^{(1)} - \delta D_x^{(0)}\right) d\Omega^{(0)} = 0$$

Then, we can discretize these two equations in a similar way to the regular FEM. First define a generalized stress vector $\sum$ and generalized rates of deformation as follows (we omit the subscript x from now on):

$$\sum = [\sigma \quad \sigma^{(1)} \quad \sigma\sigma^{(1)}]$$

$$\Delta = \left[\mathrm{D}^{(0)} \quad \mathrm{D}^{(1)} - \mathrm{D}^{(0)} \quad \frac{\partial \mathrm{D}^{(1)}}{\partial \mathrm{x}^{(0)}}\right]$$

Then define a vector d_α, which contains 2 degrees of freedom at node α:

$$\mathbf{d}_\alpha = \begin{bmatrix} v_\alpha \\ D_\alpha^{(1)} \end{bmatrix}$$

Let $\mathbf{d}^e$ be the assembly of all of the $\mathbf{d}_\alpha$ in an element. For a one-dimensional two-node rod element at node α,

$$\mathbf{d}^e = \begin{bmatrix} \boldsymbol{d}_1^T \\ \boldsymbol{d}_2^T \end{bmatrix}$$

Defining the matrix $\mathbf{N}^e$ as the shape function matrix, we interpolate the nodal values as follows:

$$\begin{bmatrix} v_\alpha \\ D^{(1)} \end{bmatrix} = \mathbf{N}^e \mathbf{d}^e$$

Similarly, it can be defined the matrix $\mathcal{Q}$ containing spatial derivatives of shape functions (the generalized $\boldsymbol{B}$ matrix) so that one writes the generalized strain rate measure in the form:

$$\Delta = \mathbf{Q}\mathbf{d}^{e}$$

The generalized mass matrix is defined as follows:

$$\mathbf{M}^{\mathrm{e}} = \begin{bmatrix} \rho^{(0)} & 0 \\ 0 & I^{(1)} \end{bmatrix}$$

Using these definitions, the finite element discretization is written as follows:

$$\delta P_{tot} = \delta P_{kin} + \delta P_{int}$$

$$\delta P_{int} = \sum_{e=1}^{n_e} \int_{\Omega^{(0)}} (\delta \boldsymbol{d}^{e})^{T} \boldsymbol{Q}^{T} \boldsymbol{\Sigma} d\Omega_{\mathrm{e}}^{(0)} = (\delta \boldsymbol{d}^{e}) \boldsymbol{f}^{\boldsymbol{int}}$$

$$\delta P_{kin} = \sum_{e=1}^{n_e} \int_{\Omega_{\mathrm{e}}^{(0)}} (\delta \boldsymbol{d}^{e})^{T} (\boldsymbol{N}^{e})^{T} \boldsymbol{M}^{e} \boldsymbol{N}^{e} \ddot{\boldsymbol{d}}^{e} d\Omega_{\mathrm{e}}^{(0)} = (\delta \boldsymbol{d}^{e})^{T} \boldsymbol{M}^{e} \ddot{\boldsymbol{d}}^{e}$$

where the number of elements is given by n_e. The mass of the system is given by the matrix **M**, the internal force by the vector $\mathbf{f}^{int}$ and the external force by the vector $\mathbf{f}^{ext}$. Defining $\mathbf{L}_e$ as the connectivity matrix of the system, then the discretization could be:

$$\boldsymbol{f}_{int} = \sum_{e=1}^{n_e} \boldsymbol{L}_e^{T} \left\{ \int \boldsymbol{Q}^{T} \boldsymbol{\Sigma} \, d\Omega_e \right\}$$

$$\boldsymbol{M} = \sum_{e=1}^{n_e} \boldsymbol{L}_e^{T} \left\{ \int (\boldsymbol{N}^{e})^{T} \boldsymbol{M}^{e} \boldsymbol{N}^{e} d\Omega_e \right\} \boldsymbol{L}_e$$

Finally, the semi-discrete equation of motion is written in a similar form as the standard finite element method:

$$\mathbf{M}\mathbf{d} = \mathbf{f}^{int}$$

This equation is then integrated explicitly following the algorithm described below.

Solution to 12.5d).

Algorithm for 1D problem (with n scales):

---

1. Initial conditions and initialization:
   Define generalized displacement **d**, generalized velocity **v**;
   set generalized stress $\sum = 0$, generalized strain $\Delta = 0$, **d**=**0**, **v**=**0**, energy variables=0, t=0, n=1
2. *getmass* and compute the lumped mass matrix and its inverse.
3. *getforce*
4. Compute the accelerations for every scale:
   $\mathbf{a}_{\mathrm{n}}^{(0)} = (\mathbf{M}^{(0)})^{-1}\mathbf{f}_{\mathrm{n}}^{(0)}$, $\mathbf{a}_{\mathrm{n}}^{(\mathrm{I})} = (\mathbf{M}^{(I)})^{-1}\mathbf{f}_{\mathrm{n}}^{(\mathrm{I})} - (\boldsymbol{D}^{(I)})^2$
5. Time update: $t^{n+1} = t^n + \Delta t^{n+1/2}$, $t^{n+1/2} = \frac{1}{2}(t^n + t^{n+1})$
6. First partial update of nodal generalized velocities: $\mathbf{v}^{n+1/2} = \mathbf{v}^n + (t^{n+1/2} - t^n)\mathbf{a}^n$
7. Enforce velocity boundary conditions:
   If node I on Γ_{v_i}: $v_{il}^{n+1/2} = \overline{v_i}(\mathbf{x}_{\boldsymbol{I}}, t^{n+\frac{1}{2}})$
8. Update nodal displacements: $\mathbf{d}^{n+1} = \mathbf{d}^n + \Delta t^{n+1/2}\mathbf{v}^{n+1/2}$
9. *getforce*
10. compute $\mathbf{a}^{n+1}$
11. Second partial update nodal velocities: $\mathbf{v}^{n+1} = \mathbf{v}^{n+1/2} + (t^{n+1} - t^n + 1/2)\mathbf{a}^{n+1}$
12. Check energy balance at time step n+1:

$$W_{int}^{n+1} = W_{int}^n + \frac{\Delta t^{n+\frac{1}{2}}}{2}\left(\mathbf{v}^{n+\frac{1}{2}}\right)^T(\mathbf{f}_{int}^n + \mathbf{f}_{int}^{n+1}) = W_{int}^n + \frac{1}{2}\Delta t^{n+1/2}\mathbf{v}^{n+1/2}(\mathbf{f}_{int}^n + \mathbf{f}_{int}^{n+1})$$

$$W_{ext}^{n+1} = W_{ext}^n + \frac{\Delta t^{n+\frac{1}{2}}}{2}\left(\mathbf{v}^{n+\frac{1}{2}}\right)^T(\mathbf{f}_{int}^n + \mathbf{f}_{int}^{n+1}) = W_{ext}^n + \frac{1}{2}\Delta t^{n+1/2}\mathbf{v}^{n+1/2}(\mathbf{f}_{int}^n + \mathbf{f}_{int}^{n+1})$$

$$W_{kin}^{n+1} = W_{kin}^{n+1} + \frac{1}{2}\mathbf{v}^{n+1/2}\mathbf{M}\mathbf{v}^{n+1/2}$$

$$|W_{kin} + W_{int} - W_{ext}| \leq \varepsilon \max(W_{ext}, W_{int}, W_{ext})$$

13. Update counter: $n \leftarrow n + 1$
14. Output; if simulation not complete, go to 5.

Subroutine *getmass*

0. Initialization: $\mathbf{M} = 0$, Global Gauss point $\xi = 1$
1. Loop over elements *e*
   - i. Set $e \leftarrow e + 1$, initialization of mass in all scales
   - ii. Compute quadrature Gauss point and weight
   - iii. Loop over quadrature points ξ_Q
     1. Compute shape function, Jacobian, and partial derivatives
     2. Compute element mass matrix for macro-scale:
        $M_e^{(0)} \leftarrow M_e^{(0)} + \rho^{(0)} \boldsymbol{N}\boldsymbol{N}^T J A \bar{w}_Q$
     3. Loop over microscale I
        - i. $M_e^{(I)} \leftarrow M_e^{(I)} + i^{(I)} \boldsymbol{N}\boldsymbol{N}^T J A^{(I)} \bar{w}_Q$

        END loop over microscale

     4. Update global Gauss points $\xi \leftarrow \xi + 1$

     END loop over quadrature points

   - iii. SCATTER $\mathbf{M}_e$ to $\mathbf{M}$, put microscale mass in the end

   END loop over elements

Subroutine *getforce*

0. Initialization: generalized stress $\sum$, generalized strain Δ,and $f_{int} = 0$, global Gauss point $\xi = 1$
1. Compute global external nodal generalized forces $\mathbf{f}_{ext}^{n}$
2. Loop over elements e
   - i. Set $e \leftarrow e + 1$, and compute quadrature Gauss point and weight
   - ii. $\mathbf{f}_{e}^{int,n}$=0
   - iii. Loop over quadrature points ξ_Q
     1. Compute shape function N, Jacobian, B matrix and relative derivatives
     2. Compute rate of deformation of macro-scale: $D^{(0),\ n-1/2} = \boldsymbol{v}^{(0),\ n-1/2^T}\mathbf{B}$
     3. Loop over microscale I
        - i. Compute $\mathrm{D}^{(I),\ n-1/2} = \boldsymbol{v}^{(I),\ n-1/2^T}\mathrm{N}$, $\frac{\partial \mathrm{D}^{(I),\ n-1/2}}{\partial x} = \boldsymbol{v}^{(I),\ n-1/2^T}\mathbf{B}$

        END loop over microscale

     4. SCATTER to generalized strain measure Δ_e,
        $\Delta_1^e = D^{(0),\ n-1/2}, \Delta_2^e = \mathrm{D}^{(1),\ n-1/2} - D^{(0),\ n-\frac{1}{2}}, \Delta_3^e = \frac{\partial \mathrm{D}^{(1),\ n-1/2}}{\partial x}$…..
     5. Update $\Delta \leftarrow \Delta + \Delta_e * \Delta t^{n+1/2}$
     6. Compute generalized stress at the Gauss point from the constitutive law
     7. Reassign the stress in the generalized stress for this element to macro stress, micro stress, micro stress couple
     8. Update the internal force in macro scale: $\mathbf{f}_{e}^{int,n} \leftarrow \mathbf{f}_{e}^{int,n} + \mathbf{B}\left(\sigma^{(0)} - \sigma^{(I)}\right) JA\overline{w}_Q$
     9. Update the internal force in microscale: $\mathbf{f}_{e}^{int,n,(I)} \leftarrow \mathbf{f}_{e}^{int,n,(I)} + (\mathbf{N}\sigma^{(I)} + \mathbf{B}\sigma\sigma^{(\boldsymbol{I})}) J^{(I)}A^{(I)}\overline{w}_Q$
     10. SCATTER internal force from all scales
     11. Update global Gauss points $\xi \leftarrow \xi + 1$

     END loop over quadrature points

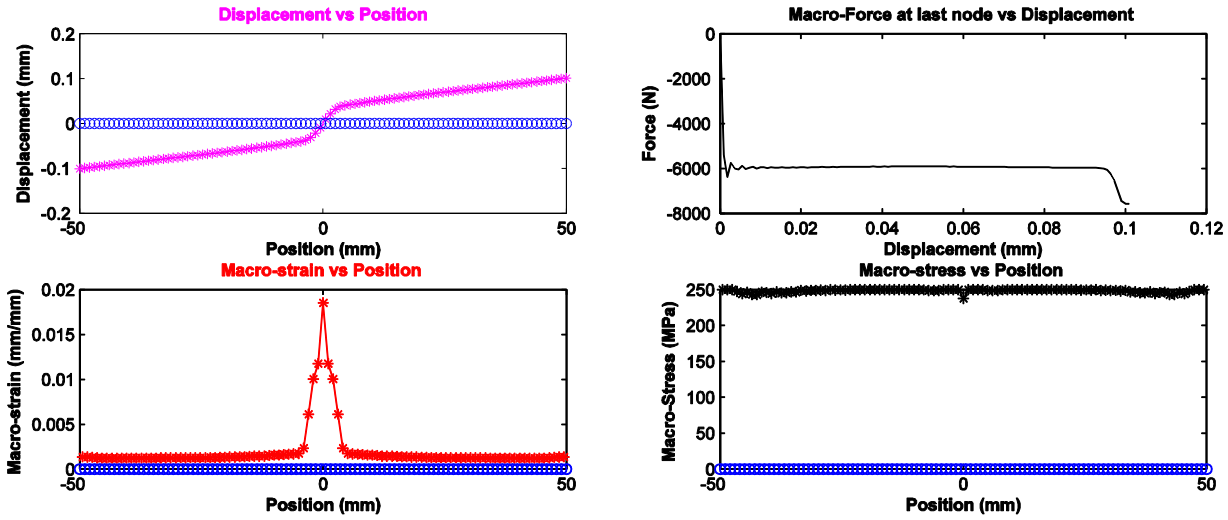
   - iv. SCATTER $\mathbf{f}_{e}^{int,n}$ to global $\mathbf{f}^{int,n}$

   END loop over elements

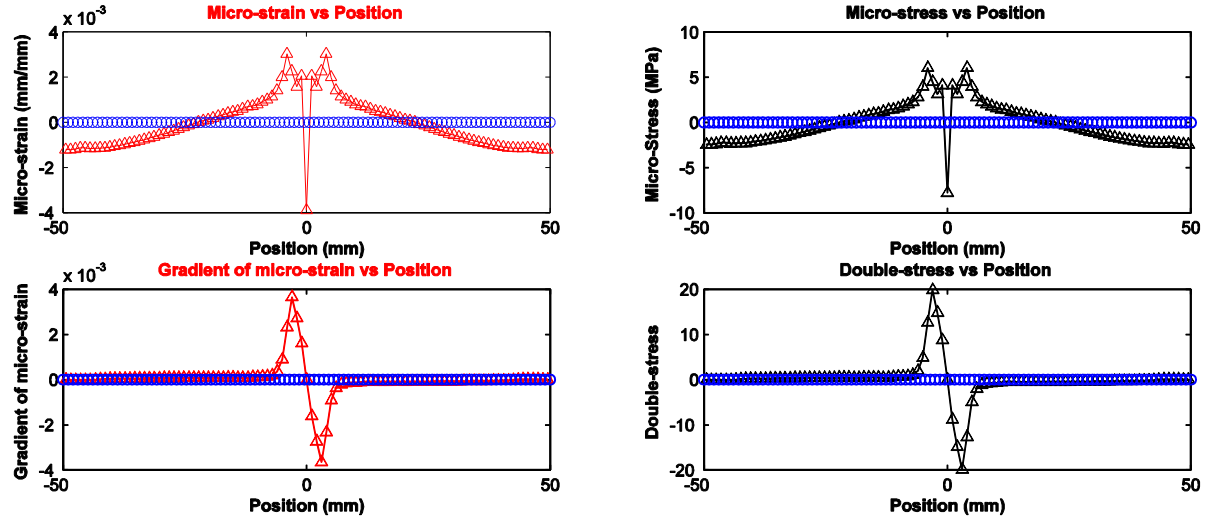
Compute nodal force: $\mathbf{f} = \mathbf{f}_{ext}^{n} - \mathbf{f}_{int}^{n}$

Solution to 12.5e).

Discussion of the results of the code:



Code outputs for the macro-scale.



Code outputs for the micro-scale.

The above figures show the results for the input parameters defined in the problem statement. Observing first the outputs for the macro-scale it is clear that there is a strain localization at the center of the bar over several elements. Readers familiar with strain gradient plasticity immediately identify the macro-strain plot versus position to have the same shape as a plot for a

one-scale strain gradient plasticity continuum. In fact, identifying this similarity is very important. Since there are no forces on the boundary conditions of the micro-scale, the second equation of the strong form,

$$\int_{\Omega^{(0)}} [\sigma\sigma^{(1)} \frac{d\left(\delta D_x^{(1)}\right)}{dx^{(0)}} d\Omega^{(0)} + \int_{\Omega^{(0)}} \sigma^{(1)}\left(\delta D_x^{(1)} - \delta D_x^{(0)}\right) d\Omega^{(0)} + \int_{\Omega^{(0)}} I^{(1)}\gamma_x^{(1)}\left(\delta D_x^{(1)} - \delta D_x^{(0)}\right) d\Omega^{(0)} = 0$$

only leads to an increase of the micro-stress due to the macro rate-of-deformation $D_x^{(0)}$. Therefore, the micro-stresses and micro double stresses influence the macro-scale right from the beginning of the simulation. This influence can be interpreted as the effect of the microstructure of the material that is being averaged for each macro-deformation state. This averaging procedure does **not** occur concurrently, but it is the result of previous analyses such that the constitutive laws for the micro-scale are calibrated. This may be viewed as a drawback of the method, since it is difficult to assign meaningful values for those extra laws. The most successful solution is to calibrate those laws by performing representative volume element (RVE) simulations of the microstructure for different load states and averaging the stresses at that scale such that approximate constitutive laws are obtained. Nevertheless, in this problem we are trying to understand the numerical details of the implementation of the method as well as understanding the influence of the extra stress measures without considering the development of the constitutive laws.

Once the macro-stress reaches the yield value at the center element (with the imperfection), this causes the macro-strain to localize in a sharp peak that leads to a sharp peak of the micro-stress and subsequentlly to a sharp peak of the micro rate-of-deformation. Once the micro rate-of-deformation develops that peak then the gradient of the micro rate-of-deformation presents an abrupt change with a different sign on each side of the peak (slope of the peak of $D_x^{(1)}$), as can be seen in the second figure above. This has the effect of diffusing the deformation to the neighboring elements, similar to what is observed in strain gradient plasticity. Note that in strain gradient plasticity there is no micro-stress, only the double stress is present (usually called higher order stress).

Solution to 12.5f).

The reader is invited to run the code and to study the influence of the various parameters in the result. In particular, it is interesting to observe that the micro-scale length paramater has a much smaller influence on the localization zone than the elastic constants of the constitutive laws of the micro-scale. Although, the expected effect of increasing the size of the micro-cell causing an increase of the localization zone happens, this increase is small.

As a final note, the code is prepared to use a different number of integration points, any number of elements, different input parameters, and any number of extra scales.

MATLAB code for Problem 12.5

```
%%%%%%%%%%%%%%%%%%% Multiresolution Continuum 1D code %%%%%%%%%%%%%%%%%%%%%%
%
%   M. A. Bessa  (mbessa@u.northwestern.edu)
%   Northwestern University, Mechanical Engineering
%
%   June 14, 2013
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [] = multiresolution_in_1D()
    close all; clear all; clc; tic
    %% CONSISTENT UNITS:
    % MASS  LENGTH  TIME  FORCE  STRESS    ENERGY
    %  g      mm     ms     N      MPa    N-mm (mJ)
    %
    %% Input parameters.
    % Termination time
    endtime = 2.0e-2;      % total time of simulation [ms]
    prescribed_v_0 = 5.0;  % Prescribed velocity [mm/ms]
    output_frames = 'all'; % Number of times that every plot will be shown
                           %(for output_frames = 'all' every time
                           %step will have a plot)
    % Mesh definition
    nel = 99;          % Number of elements
    ngp = 1;           % Number of Gauss points per element
    % Geometric properties
    mesh.L    = 100.0;   % Length of the bar [mm]
    Xmin      = -mesh.L/2;  % Position of the left end of the bar [mm]
    mesh.area = 30;  % Cross-section Area of the bar [mm^2]
    % Material properties
    youngsC_0 = 2.0e5;    % Modulus of elasticity (for linear). [MPa]
    rho_0     = 7.85e-3;  % Density per unit length. [g/mm^3]
    sigy_0    = 250.0;    % Macro-yield stress [MPa]
    imperfection = 0.95;  % Imperfection at the middle element (0: no ele-
                          %ment; 1: no imperfection)
    %
    % Number of extra sets of degrees of freedom ('scales' in lack of a
    %better term)
    nscales = 1;       % Number of extra 'scales' (NOT including the
```

```
                    %macro-scale). Use nscales = 0 for a regular FEM
                    %analysis
    %
    % Extra dof input parameters
    % Initilize variables
    l_n = zeros(nscales,1);
    youngsC_n = zeros(nscales,1);
    youngsB_n = zeros(nscales,1);
    rho_n     = zeros(nscales,1);
    %
    % (Extra) Scale 1:
    l_n(1)    = mesh.L/nel * 4; % Horizontal size of domain in scale 1 (in
                                %the x direction [mm]
    h_n(1)    = l_n(1); % Vertical size of domain in scale 1 (in the y
                        %direction) [mm]
    thick_n(1)= l_n(1); % Width of domain in scale 1 (z direction or
                        %out-of-plane direction) [mm]
    %
    youngsC_n(1) = youngsC_0*0.01; % Constant for scale 1 constitutive law
    youngsB_n(1) = 0.0;            % Constant for scale 1 constitutive law
                                   %(gradient term)
    rho_n(1)     = rho_0*0.9;      % Density for scale 1
    %
    %
    % Scale 2:
%     l_n(2)    = mesh.L/nel*0.5; % Horizontal size of domain in scale 2 (in
%                                 %the x direction [mm]
%     h_n(2)    = l_n(2); % Vertical size of domain in scale 2 (in the y
%                         %direction) [mm]
%     thick_n(2)= l_n(2); % Width of domain in scale 2 (z direction or
%                         %out-of-plane direction) [mm]
%     %
%     youngsC_n(2) = youngsC_0*0.002;  % Constant for scale 2 constitutive law
%     youngsB_n(2) = 0.0;              % Constant for scale 2 constitutive law
%                                      %(gradient term)
%     rho_n(2)     = rho_0*1.5;        % Density for scale 2
    %% Calculated parameters.
    mesh.nn = nel+1;            % Number of nodes
    l_0     = mesh.L/nel;       % Element reference length
    %
    area_n = zeros(nscales,1);
    I_n    = zeros(nscales,1);
    for iscale=1:nscales
        % Compute the cross-section of the domain of scale i
        area_n(iscale) = h_n(iscale)*thick_n(iscale);
        %
        % Compute the I term of the mass matrix for scale i. This is the
        %moment of inertia DIVIDED by the 'volume' of that domain.
        I_n(iscale) = rho_n(iscale)*(h_n(iscale)^2+thick_n(iscale)^2)/12;

    end
    %
    % Calculate the total number of Gauss points
    ngp_tot = ngp*nel;
    %
    sigygp_0 = sigy_0*ones(ngp_tot,1); % Macro-yield stress in each Gauss
                                       %point
    %
    % Include imperfection at middle element
    num_imp = floor(nel/2)+1;          %imperfection element
    %
    for igp = 1:ngp
        sigygp_0((num_imp-1)*ngp+igp)= imperfection*sigy_0;
```

```
%         epsygp_0((num_imp-1)*ngp+igp) = imperfection*epsy_0;
    end
    %
    c_0     = sqrt(youngsC_0/rho_0);  % Macro-wave speed [mm/ms]
    tcrit_0 = l_0/c_0;           % Critical timestep (macro scale) [ms].
    %
    c_n     = zeros(nscales,1);
    tcrit_n = zeros(nscales,1);
    for i = 1:nscales
        c_n(i) = sqrt(youngsC_n(i)/I_n(i)); % Wave speed in each scale
        %
        if c_n(i) ~= 0
            tcrit_n(i) = l_0/c_n(i);     % Critical time step in each scale
            tcrit = min(tcrit_0, tcrit_n(i));
            cmax = max(c_0, c_n(i));
        else
            tcrit = tcrit_0;
            cmax = c_0;
        end
    end
    %
    if nscales == 0
        dt = tcrit_0 * 0.9;
    else
        dt = tcrit * 0.9;       % Time step of simulation.
    end
    num_ts = round(endtime/dt); % Total number of time steps.
    %
    % Output some information about the simulation
    fprintf('INFORMATION:\n');
    if nscales == 0
        fprintf('-> Critical time step: %d miliseconds\n',tcrit_0);
    elseif nscales > 0
        fprintf('-> Critical time step: %d miliseconds\n',tcrit);
    end
    fprintf('-> Total number of time steps: %d \n',num_ts);
    endtime = num_ts*dt;    % Termination time
    if output_frames == 'all'
        output_frames = num_ts;
    end
    %
    % Save material properties for all scales in one variable:
    matprops = zeros((1+nscales),3);
    matprops(1,:) = [youngsC_0 , 0 , rho_0];
    for iscale=1:nscales
        matprops(1+iscale,:) = [youngsC_n(iscale) , youngsB_n(iscale) ,...
                                rho_n(iscale)];
    end
    %% Preprocessing.
    % Nodal coordinates in reference configuration.
    Xmax = Xmin + mesh.L;
    mesh.X = linspace(Xmin, Xmax, mesh.nn)';
    % Connectivity matrix (each row gives nodes in an element.)
    mesh.conn = [ 1:length(mesh.X)-1; 2:length(mesh.X) ]';
    %
    %%
    % ..::: Now following the flowchart for explicit time integration :::..
    %                 Belytschko et al book (box 6.1 - p.313)
    %
    %% 1. Initial conditions and initialization
    % Nodal Array definition
    gen_u = zeros((1+nscales)*mesh.nn,1);      % generalized displacement
    gen_v = zeros((1+nscales)*mesh.nn,1);      % generalized velocity
```

```
    % Initialize variables with information from the previous time step
    old.gen_sigma = zeros((1+2*nscales),ngp_tot); % old gen. stresses
    old.gen_truestrain = zeros((1+2*nscales),ngp_tot); % old generalized
                                                       %true strains
    % Initialize energy variables and others
    Wint = zeros(num_ts,1);
    Wext = zeros(num_ts,1);
    Wkin = zeros(num_ts,1);
    Wtot = zeros(num_ts,1);
    displacement = zeros(num_ts,1);
    reaction = zeros(num_ts,1);
    time = zeros(num_ts,1);
    fint_old = zeros(mesh.nn*(nscales+1),1);
    fext_old = zeros(mesh.nn*(nscales+1),1);
    Wint_old = 0;
    Wext_old = 0;
    Wkin_old = 0;
    %
    %
    t  = 0; % time
    ts = 1; % n + 1 (where 'n' is the counter used in box 6.1, point 1)
    %
    % Compute the consistent mass matrix
    [Mc,Xgp] = getMass(mesh,nscales,ngp,rho_0,I_n,area_n);
    %
    % Compute lumped mass matrix
    [M,invM] = getLumpedMass(Mc,mesh,nscales);
    %
    %
    %% 2. getforce
    % No residual stresses considered:
    fint = zeros(mesh.nn*(nscales+1),1);
    % Get external nodal forces
    [fext] = getFext(t,mesh,nscales);
    % Compute nodal forces
    force = fext - fint;
    %
    %
    %% 3. Compute accelerations a^n
    gen_a = invM*force;
    % NOTE: for the extra DOF what we obtain is GAMMA, which is NOT the
    %time derivative of D, that is used in the explicit time integration
    % Therefore, dotD^(n) = gamma^(n) - D^(n)*D^(n)
    gen_a((mesh.nn+1):end) = gen_a((mesh.nn+1):end)...
                             - gen_v((mesh.nn+1):end).^2;
    %
    %
    while(t <= endtime)
        %% 4. Time update
        tnew = t + dt;          % tnew = t^(n+1); t = t^n
        thalf = 1/2*(t + tnew); % thalf = t^(n+1/2)
        %
        %% 5. First partial update nodal velocities
        gen_vhalf = gen_v + (thalf-t)*gen_a;
        %
        %% 6. Enforce velocity boundary conditions
%         gen_vhalf(mesh.nn) = 0.0;
        gen_vhalf(1) = -prescribed_v_0;
        gen_vhalf(mesh.nn) = prescribed_v_0;
        %
        %% 7. Update nodal displacements
        gen_u = gen_u+dt*gen_vhalf;
        %
```

```
        % Determine the current position of the nodes
        mesh.x = gen_u(1:mesh.nn) + mesh.X;
        %% 8. getforce
        % Get internal nodal forces
        [fint,gen_sigma,gen_truestrain] = getFint(dt,mesh,nscales,ngp,...
                                          matprops,old,l_n,h_n,thick_n,...
                                          gen_vhalf,sigygp_0);
        % Get external nodal forces
        [fext] = getFext(t,mesh,nscales);
        % Compute nodal forces
        force = fext - fint;
        %
        %% 9. Compute a^(n+1)
        gen_a = invM*force;
        % NOTE: for the extra DOF what we obtain for gen_a (generalized
        %acceleration) is GAMMA, which is NOT the time derivative of D,
        %that is used in the explicit time integration. Therefore,
        %dotD^(n) = gamma^(n) - D^(n)*D^(n)
        gen_a((mesh.nn+1):end) = gen_a((mesh.nn+1):end)...
                                 - gen_v((mesh.nn+1):end).^2;
        %
        %% 10. Second partial update nodal velocities
        gen_vnew = gen_vhalf + (tnew-thalf)*gen_a;
        %% 11. Check energy balance at time step n+1
        Wint_inc = 0;
        Wext_inc = 0;
        Wkin_inc = 0;
        Wint_inc = Wint_inc + 1/2*dt*(gen_vhalf(1:mesh.nn)')...
                   *(fint_old(1:mesh.nn)+fint(1:mesh.nn));
        Wext_inc = Wext_inc + 1/2*dt*(gen_vhalf(1:mesh.nn)')...
                   *(fext_old(1:mesh.nn)+fext(1:mesh.nn));
        Wkin_inc = Wkin_inc + 1/2*(gen_vhalf(1:mesh.nn)')...
                   *M(1:mesh.nn,1:mesh.nn)*gen_vhalf(1:mesh.nn);
        Wint(ts) = Wint_old + Wint_inc;
        Wext(ts) = Wext_old + Wext_inc;
        Wkin(ts) = Wkin_old + Wkin_inc;
        %
        Wtot(ts) = Wint(ts) + Wext(ts) + Wkin(ts);
        time(ts) = t;
        %% 12. Update counter
        ts = ts + 1;
        gen_v = gen_vnew;
        t = tnew;
        fint_old = fint;
        fext_old = fext;
        old.gen_sigma = gen_sigma;
        old.gen_truestrain = gen_truestrain;
        %% 13. Output
        displacement(ts) = gen_u(mesh.nn);
        reaction(ts) = force(mesh.nn);
        if mod(ts,round(num_ts/output_frames)) == 0
            figure(1)
            % Displacement vs Position
            subplot(2,2,1);
            plot(mesh.X,gen_u(1:mesh.nn),'m*-',...
                 mesh.X,zeros(1,size(mesh.X,2)),'bo-');
            title('Displacement vs Position', 'FontWeight', 'bold',...
                  'Color', 'm');
            xlabel('Position (mm)', 'FontWeight', 'bold');
            ylabel('Displacement (mm)', 'FontWeight', 'bold');
            %
            % Macro-force at 1st node vs Displacement
            subplot(2,2,2);
```

```
plot(displacement,reaction,'k-');
title('Macro-Force at last node vs Displacement',...
      'FontWeight', 'bold', 'Color', 'k');
xlabel('Displacement (mm)', 'FontWeight', 'bold');
ylabel('Force (N)', 'FontWeight', 'bold');
%
% Macro-strain (true strain) vs Position
subplot(2,2,3);
plot(Xgp,gen_truestrain(1:(2*nscales+1):end)','r*-',...
     mesh.X,zeros(1,size(mesh.X,2)),'bo-');
title('Macro-strain vs Position', 'FontWeight', 'bold',...
      'Color', 'r');
xlabel('Position (mm)', 'FontWeight', 'bold');
ylabel('Macro-strain (mm/mm)', 'FontWeight', 'bold');
%
% Macro-stress vs Position
subplot(2,2,4);
plot(Xgp,gen_sigma(1:(2*nscales+1):end)','k*-',...
     mesh.X,zeros(1,size(mesh.X,2)),'bo-');
title('Macro-stress vs Position', 'FontWeight', 'bold',...
    'Color', 'k');
%     axis([xminplot xmaxplot yminplot ymaxplot])
xlabel('Position (mm)', 'FontWeight', 'bold');
ylabel('Macro-Stress (MPa)', 'FontWeight', 'bold');
hold off;
%
%
if nscales > 0
    % Plots of the first micro-scale
    figure(2)
    %
    % Micro-strain vs Position
    subplot(2,2,1);
    plot(Xgp,gen_truestrain(2:(2*nscales+1):end)','r^-',...
         mesh.X,zeros(1,size(mesh.X,2)),'bo-');
    title('Micro-strain vs Position', 'FontWeight', 'bold',...
          'Color', 'r');
    xlabel('Position (mm)', 'FontWeight', 'bold');
    ylabel('Micro-strain (mm/mm)', 'FontWeight', 'bold');
    %
    % Micro-stress vs Position
    subplot(2,2,2);
    plot(Xgp,gen_sigma(2:(2*nscales+1):end)','k^-',...
         mesh.X,zeros(1,size(mesh.X,2)),'bo-');
    title('Micro-stress vs Position', 'FontWeight', 'bold',...
        'Color', 'k');
    %     axis([xminplot xmaxplot yminplot ymaxplot])
    xlabel('Position (mm)', 'FontWeight', 'bold');
    ylabel('Micro-Stress (MPa)', 'FontWeight', 'bold');
    %
    % Gradient of micro-strain vs Position
    subplot(2,2,3);
    plot(Xgp,gen_truestrain(3:(2*nscales+1):end)','r^-',...
         mesh.X,zeros(1,size(mesh.X,2)),'bo-');
    title('Gradient of micro-strain vs Position', ...
        'FontWeight','bold', 'Color', 'r');
    xlabel('Position (mm)', 'FontWeight', 'bold');
    ylabel('Gradient of micro-strain', 'FontWeight', 'bold');
    %
    % Double-stress vs Position
    subplot(2,2,4);
    plot(Xgp,gen_sigma(3:(2*nscales+1):end)','k^-',...
         mesh.X,zeros(1,size(mesh.X,2)),'bo-');
```

```
                title('Double-stress vs Position', 'FontWeight',...
                    'bold', 'Color', 'k');
                %     axis([xminplot xmaxplot yminplot ymaxplot])
                xlabel('Position (mm)', 'FontWeight', 'bold');
                ylabel('Double-stress', 'FontWeight', 'bold');
%
%                 figure(3)
%                 %
%                 % Micro-strain vs Position
%                 subplot(2,2,1);
%                 plot(Xgp,gen_truestrain(4:(2*nscales+1):end)','r^-',...
%                      mesh.X,zeros(1,size(mesh.X,2)),'bo-');
%                 title('Micro-strain vs Position', 'FontWeight', 'bold', 'Color',
'r');
%                 xlabel('Position (mm)', 'FontWeight', 'bold');
%                 ylabel('Micro-strain (2) (mm/mm)', 'FontWeight', 'bold');
%                 %
%                 % Micro-stress vs Position
%                 subplot(2,2,2);
%                 plot(Xgp,gen_sigma(4:(2*nscales+1):end)','k^-',...
%                      mesh.X,zeros(1,size(mesh.X,2)),'bo-');
%                 title('Micro-stress vs Position', 'FontWeight', 'bold',...
%                     'Color', 'k');
%                 %     axis([xminplot xmaxplot yminplot ymaxplot])
%                 xlabel('Position (mm)', 'FontWeight', 'bold');
%                 ylabel('Micro-Stress (2) (MPa)', 'FontWeight', 'bold');
%                 %
%                 % Gradient of micro-strain vs Position
%                 subplot(2,2,3);
%                 plot(Xgp,gen_truestrain(5:(2*nscales+1):end)','r^-',...
%                      mesh.X,zeros(1,size(mesh.X,2)),'bo-');
%                 title('Gradient of micro-strain vs Position', 'FontWeight',...
%                     'bold', 'Color', 'r');
%                 xlabel('Position (mm)', 'FontWeight', 'bold');
%                 ylabel('Gradient of micro-strain (2)', 'FontWeight', 'bold');
%                 %
%                 % Double-stress vs Position
%                 subplot(2,2,4);
%                 plot(Xgp,gen_sigma(5:(2*nscales+1):end)','k^-',...
%                      mesh.X,zeros(1,size(mesh.X,2)),'bo-');
%                 title('Double-stress vs Position', 'FontWeight',...
%                     'bold', 'Color', 'k');
%                 %     axis([xminplot xmaxplot yminplot ymaxplot])
%                 xlabel('Position (mm)', 'FontWeight', 'bold');
%                 ylabel('Gradient of micro-stress (2)', 'FontWeight', 'bold');
%
                hold off;
            end % end if nscales
        end % end if mod
        %
    end % end while loop
    toc
end
%% Returns the consistent mass matrix
function [Mc,Xgp] = getMass(mesh,nscales,ngp,rho_0,I_n,area_n)
    % Allocate space for mass matrix.
    Mc = zeros(mesh.nn*(nscales+1),mesh.nn*(nscales+1));
    nel = 0;
    ngp_global = 1;
    Xgp = zeros((mesh.nn-1)*ngp,1);
    for conn = mesh.conn'
        nel = nel+1; % Element number inside this loop
        nne = length(conn); % Number of nodes of this element
```

```
        %
        Me_0 = zeros(nne,nne);  % element mass matrix for macro-scale
        Me_n = zeros(nne,nne,nscales); % element mass
                                                    %matrix for
                                                    %extra scales
        %
        [qpt, qwt] = quadrature(ngp); % Quadrature points and weights
        for i=1:ngp
            xigp = qpt(i); % Gauss point in parent coordinates
            % Get the shape functions:
            N = getN(xigp);
            % Gauss point in ref. coordinates (not needed):
            Xgp(ngp_global) = (mesh.X(conn)')*N;
            %
            dN = getGradN(xigp);    % dN/dxi (wrt parent coords)
            dXdxi  = (mesh.X(conn)')*dN; % dx/dxi (Jacobian of the ref.
            %config. wrt parent coords)
            Jac0xi = det(dXdxi); % Jacobian determinant ref./parent
            % Compute element mass matrix for macro-scale
            Me_0 = Me_0 + rho_0*N*(N')*Jac0xi*mesh.area*qwt(i);
            %
            % Compute element mass matrix for each extra scale
            for iscale=1:nscales
                Me_n(:,:,iscale) = Me_n(:,:,iscale)+I_n(iscale)*N*(N')...
                                   *Jac0xi*area_n(iscale)*qwt(i);
            end
            %
            ngp_global = ngp_global + 1;
        end
        %
        % Mass matrix assembly
        %Indices of rows and columns for each type of dof:
        i_0  = conn;
        Mc(i_0,i_0) = Mc(i_0,i_0) + Me_0;
        for iscale=1:nscales
            i_n = mesh.nn*iscale+conn;
            Mc(i_n,i_n) = Mc(i_n,i_n) + Me_n(:,:,iscale);
        end
        %
    end
end
%% Returns quadrature points and quadrature weights.
function [qpt, qwt] = quadrature(ngp)
    if ngp == 1
        x = zeros(ngp,1);
        wt = zeros(ngp,1);
        x(1) = 0;
        wt(1) = 2;
    elseif ngp == 2
        x = zeros(ngp,1);
        wt = zeros(ngp,1);
        x(1) = -sqrt(1/3);
        x(2) = -x(1);
        wt(1) = 1;
        wt(2) = 1;
    elseif ngp == 3
        x = zeros(ngp,1);
        wt = zeros(ngp,1);
        x(1) = -sqrt(3/5);
        x(2) = 0;
        x(3) = -x(1);
        wt(1) = 5/9;
        wt(2) = 8/9;
```

```
        wt(3) = wt(1);
    elseif ngp == 4
        x = zeros(ngp,1);
        wt = zeros(ngp,1);
        x(1) = -sqrt( (3+2*sqrt(6/5))/7 );
        x(2) = -sqrt( (3-2*sqrt(6/5))/7 );
        x(3) = -x(2);
        x(4) = -x(1);
        wt(1) = ( 18-sqrt(30) )/36;
        wt(2) = ( 18+sqrt(30) )/36;
        wt(3) = wt(2);
        wt(4) = wt(1);
    else error('More than 4 point quadrature is not implemented');
    end
    %
    qpt = x;
    qwt = wt;
end
%% Returns a 2x1 matrix that gives the shape functions
function [N] = getN(xi)
    % xi is the parent coordinate
    %
    N = 1/2*[(1-xi) , (1+xi)]';
    % NOTE: N is defined as a COLUMN vector: 2x1 (instead of a row vector
    %as in the book)
end
%% Returns the lumped mass matrix and its inverse
function [M,invM] = getLumpedMass(Mc,mesh,nscales)
    M = spalloc(mesh.nn*(nscales+1),mesh.nn*(nscales+1),mesh.nn);
    for i=1:(mesh.nn*(nscales+1))
        for j=1:(mesh.nn*(nscales+1))
            M(i,i) = M(i,i)+Mc(i,j);
        end
    end
    %
    % Compute the inverse of the lumped mass matrix
    invM = spalloc(mesh.nn*(nscales+1),mesh.nn*(nscales+1),mesh.nn);
    %
    for i=1:(mesh.nn*(nscales+1))
        for j=1:(mesh.nn*(nscales+1))
            if M(i,i) == 0;
                invM(i,i) = 0;
            else
                invM(i,i) = 1/M(i,i);
            end
        end
    end
end
%% Computes the force.
function [fint,gen_sigma,gen_truestrain] = getFint(dt,mesh,nscales,ngp,...
                                            matprops,old,l_n,h_n,...
                                            thick_n,gen_v,sigygp_0)
    % Allocate space for generalized stresses and strain measures
    gen_sigma = zeros(size(old.gen_sigma)); % generalized stresses
    gen_truestrain = zeros(size(old.gen_sigma)); % generalized strains
    % Allocate space for the nodal forces matrix
    fint = zeros(mesh.nn*(nscales+1),1);
    nel = 0;
    ngp_global = 1; % Counter for the total number of Gauss points
    % Loop over element nel
    for conn = mesh.conn'
        nel = nel+1; % Element number inside this loop
        nne = length(conn); % Number of nodes of this element
```

```
        %
        [qpt, qwt] = quadrature(ngp); % Quadrature points and weights
        fe_int = zeros(nne*(nscales+1),1);
        for i=1:ngp
            xigp = qpt(i); % Gauss point in parent coordinates
            % Get the shape functions:
            N = getN(xigp);
%             % Gauss point in current coordinates (not needed):
%             xgp = (mesh.x(conn)')*N;
            % dN/dxi (wrt parent coords):
            dN = getGradN(qpt(i));
            % dx/dxi (Jacobian of the current config. wrt parent coords):
            dxdxi  = (mesh.x(conn)')*dN;
            % Jacobian determinant
            Jac_xi = det(dxdxi);
            % Calculate the scripted B matrix in current configuration
            %for the regular dofs:
            Bscript = dN/dxdxi; % Scripted B matrix
            %
            %
            % Compute rate-of-deformation
            D_0 = (gen_v(conn)')*Bscript;
            %
            D_n = zeros(nscales,1);
            gradD_n = zeros(nscales,1);
            %
            for is=1:nscales
                D_n(is) = (gen_v(mesh.nn*(is)+conn)')*N;
                gradD_n(is) = (gen_v(mesh.nn*(is)+conn)')*Bscript;
            end
            %
            % Generalized strain measures (assemble previous rate-of-
            %-deformation measures in one vector):
            gen_D = zeros((1+2*nscales),1);
            gen_D(1) = D_0;
            for is=1:nscales
                gen_D(2*is) = D_n(is)-D_0;
                gen_D(2*is+1) = gradD_n(is);
            end
            %
            % Determine the true strain (logarithmic strain) for this Gauss
            %point
            gen_truestrainGP = old.gen_truestrain(:,ngp_global) +...
                               gen_D * dt;
            %
            % Get the stress from the constitutive law:
            [gen_sigmaGP] = getGenStress(ngp_global,dt,matprops,l_n,h_n,...
                                         thick_n,nscales,old,gen_D,sigygp_0);
            % Note: "gen_sigmaGP" is the vector with the generalized
            %stresses for this element
            %
            % The generalized stress vector containing all the information
            %for every Gauss point is:
            gen_sigma(:,ngp_global) = gen_sigmaGP;
            % The generalized true strain vector containing all the
            %information for every Gauss point is:
            gen_truestrain(:,ngp_global) = gen_truestrainGP;
            % Rename the stresses in the generalized stress vector for this
            %element:
            sigma_0 = gen_sigmaGP(1);
            sigma_n = zeros(nscales,1);
            gradsigma_n = zeros(nscales,1);
            for is=1:nscales
```

```
                sigma_n(is) = gen_sigmaGP(2*is); % micro-stresses
                gradsigma_n(is) = gen_sigmaGP(2*is+1); % double-stresses
            end
            %
            auxsigma = sigma_0;
            for is=1:nscales
                auxsigma = auxsigma - sigma_n(is);
            end
            % Calculate the nodal internal forces
            fe_int(1:nne) = fe_int(1:nne) + Bscript*auxsigma*Jac_xi...
                            *mesh.area*qwt(i);
            %
            for is=1:nscales
                ind = 1+nne*is;
                fe_int(ind:(ind+nne-1)) = fe_int(ind:(ind+nne-1)) + ...
                    ( N*sigma_n(is) + Bscript*gradsigma_n(is) ) ...
                    *Jac_xi*h_n(is)*thick_n(is)*qwt(i);
            end
            %
            ngp_global = ngp_global + 1;
        end % end element mass matrix for enriched element
        %
        % Scatter the force
        % Indices of rows and columns for each type of dof:
        i_0  = conn;
        fint(i_0) = fint(i_0) + fe_int(1:nne);
        for is=1:nscales
            i_n = mesh.nn*is+conn;
            ind = 1+nne*is;
            fint(i_n) = fint(i_n) + fe_int(ind:(ind+nne-1));
        end
        %
    end
end
%% Returns a 2x1 matrix that gives the derivative of the shape functions.
function [dN] = getGradN(xi)
    % xi is not necessary for linear shape functions.
    dN = 1/2*[-1 , 1]';
    % Note: dN is a COLUMN vector: 2x1 matrix (instead of 1x2 as in the
    %book)
end
%% Returns the generalized stress according to chosen constitutive law
function [gen_sigmaGP] = getGenStress(ngp_global,dt,matprops,l_n,h_n,...
                                      thick_n,nscales,old,gen_D,sigygp_0)
    % The generalized stresses for this ELEMENT are given by the vector:
    %
    %              |      sigma^(0)        |   <--- macro stress
    %              |      sigma^(1)        |   <--- micro stress (1)
    %              |    sigma-sigma^(1)    |   <--- double-stress (1)
    %              |      sigma^(2)        |   <--- micro stress (2)
    %gen_sigmaGP = |    sigma-sigma^(2)    |   <--- double-stress (2)
    %              |           .           |
    %              |           .           |
    %              |           .           |
    %              |      sigma^(n)        |   <--- micro stress (n)
    %              |    sigma-sigma^(n)    |   <--- double-stress (n)
    %
    %% Read material properties for every scale:
    youngsC_0 = matprops(1,1);
    %
    youngsC_n = zeros(nscales,1);
    youngsB_n = zeros(nscales,1);
    rho_n = zeros(nscales,1);
```

```
    for iscale=1:nscales
        youngsC_n(iscale) = matprops(1+iscale,1);
        youngsB_n(iscale)= matprops(1+iscale,2);
        rho_n(iscale) = matprops(1+iscale,3);
    end
    %
    Cmat = zeros((1+2*nscales),(1+2*nscales));
    Cmat(1,1) = youngsC_0;
    for i=1:nscales
        Cmat(2*i,2*i:(1+2*i)) = [youngsC_n(i) , youngsB_n(i)];
        Cmat(2*i+1,2*i:(1+2*i))= [youngsB_n(i) , youngsC_n(i)...
            *(h_n(iscale)^2+thick_n(iscale)^2)/12]];
    end
    %
    %% Apply the constitutive law
    % Get the generalized stresses for this Gauss Point before yielding
    gen_sigmaGP = old.gen_sigma(:,ngp_global) + Cmat * gen_D * dt;
    %
    % Get the generalized stresses after yielding
    if abs(gen_sigmaGP(1)) > sigygp_0(ngp_global)
        gen_sigmaGP(1) = sign(gen_sigmaGP(1))*sigygp_0(ngp_global);
    end
    %
end
%% Returns external nodal forces
function [fext] = getFext(t,mesh,nscales)
    fext = zeros(mesh.nn*(nscales+1),1);
%     fext(1) = 1e3 * mesh.area;
end
```

CHAPTER 13: Single Crystal Plasticity

13.1 Show that for cubic crystals the unit normal $\mathbf{n}^{hkl} = \left(h^2 + k^2 + l^2\right)^{-1/2} (h\mathbf{e}_1 + k\mathbf{e}_2 + l\mathbf{e}_3)$.

Solution to 13.1

Beginning with the definition of the reciprocal lattice vector,

$$\mathbf{g}^{hkl} = h\mathbf{b}^1 + k\mathbf{b}^2 + l\mathbf{b}^3$$

The reciprocal bases are given by,

$$\mathbf{b}^1 = \frac{\mathbf{a}_2 \times \mathbf{a}_3}{\mathbf{a}_1 \cdot \left(\mathbf{a}_2 \times \mathbf{a}_3\right)} = \frac{\mathbf{a}_1}{\mathbf{a}_1 \cdot \left(\mathbf{a}_2 \times \mathbf{a}_3\right)}$$
$$\mathbf{b}^2 = \frac{\mathbf{a}_3 \times \mathbf{a}_1}{\mathbf{a}_1 \cdot \left(\mathbf{a}_2 \times \mathbf{a}_3\right)} = \frac{\mathbf{a}_2}{\mathbf{a}_1 \cdot \left(\mathbf{a}_2 \times \mathbf{a}_3\right)}$$
$$\mathbf{b}^3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{\mathbf{a}_1 \cdot \left(\mathbf{a}_2 \times \mathbf{a}_3\right)} = \frac{\mathbf{a}_3}{\mathbf{a}_1 \cdot \left(\mathbf{a}_2 \times \mathbf{a}_3\right)}$$

Thus, the reciprocal lattice vector is given by,

$$\mathbf{g}^{hkl} = \frac{1}{\mathbf{a}_1 \cdot \left(\mathbf{a}_2 \times \mathbf{a}_3\right)} \left(h\mathbf{a}_1 + k\mathbf{a}_2 + l\mathbf{a}_3\right)$$

In a cubic crystal, all lattice vectors are of equal length, say *a*, therefore,

$$\mathbf{g}^{hkl} = \frac{1}{a^2} \left(h\mathbf{e}_1 + k\mathbf{e}_2 + l\mathbf{e}_3\right)$$

The inverse of its norm is given by,

$$\left|\mathbf{g}^{hkl}\right|^{-1} = \frac{a^2}{\sqrt{h^2 + k^2 + l^2}}$$

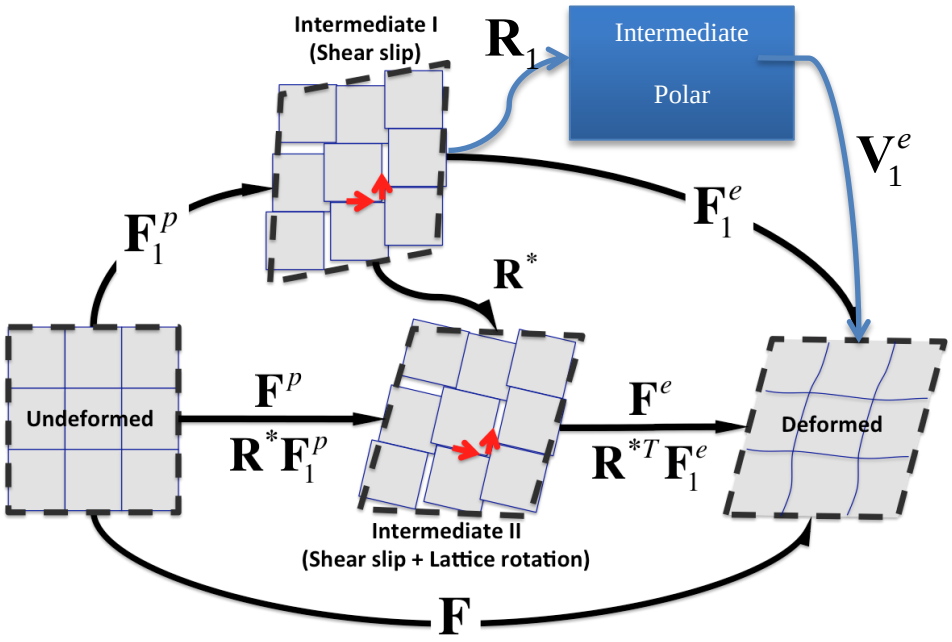
The unit normal is therefore,

$$\mathbf{n}^{hkl} = \left|\mathbf{g}^{hkl}\right|^{-1} \mathbf{g}^{hkl} = \frac{1}{\sqrt{h^2 + k^2 + l^2}} \left(h\mathbf{e}_1 + k\mathbf{e}_2 + l\mathbf{e}_3\right)$$

Which proves that the normal vector is in the direction [*hkl*].

13.2 Show that according to Fig. 13.10 $\mathbf{F}^e = \mathbf{V}_1^e$ and $\mathbf{R}_1 = \mathbf{R}^*$, where $\mathbf{F}_1^e = \mathbf{V}_1^e \mathbf{R}_1$.

Solution to 13.2



Two definitions are of interest,

$$\mathbf{F} = \mathbf{F}_1^e \mathbf{F}_1^p, \text{ so } \mathbf{F} = \left(\mathbf{V}_1^e \mathbf{R}_1\right)\mathbf{F}_1^p, \text{ and}$$

$$\mathbf{F} = \mathbf{F}^e \mathbf{F}^p, \text{ with } \mathbf{F}^e \text{ symmetric}$$

Now, configurations Intermediate II and Intermediate-Polar both follow from Intermediate I by rigid rotations only, **R*** and $\mathbf{R}_1$ respectively. Going from Intermediate II and Intermediate-Polar to the deformed configuration only symmetric tensors are to be used, $\mathbf{F}^e$ and $\mathbf{V}_1^e$ respectively. Since Intermediate II and Intermediate-Polar follow from intermediate I by rigid rotation, *symmetric* $\mathbf{F}^e$ and $\mathbf{V}_1^e$ must describe the *same* total elastic *stretching* of the crystal that yields the final configuration. That is, $\mathbf{F}^e = \mathbf{V}_1^e$. But then $\mathbf{V}_1^{e\text{-}1}$ maps into intermediate II from the deformed configuration, just as $\mathbf{F}^{e-1}$

does. In this manner, intermediate II and intermediate-Polar must be the same configuration, since $\mathbf{V}_1^{e\text{-}1}a$, a 1-to-1 function, maps to both, and $\mathbf{R}_1 = \mathbf{R}^*$.

13.3 Show that crystalline plasticity is incompressible (*Hint*: use Eq. 13.5.7).

Solution to 13.3

Beginning with the relation

$$\tilde{\mathbf{C}}^{e-1} : \tilde{\mathbf{D}}^p = \sum\nolimits_\alpha \tilde{\mathbf{C}}^{e-1} : \tilde{\mathbf{P}}^\alpha \dot{\gamma}^\alpha$$

Noting that $\tilde{\mathbf{C}}^e$ serves as a metric tensor **g** on intermediate configuration I, we can write

$$\sum\nolimits_\alpha \tilde{\mathbf{C}}^{e-1} : \tilde{\mathbf{P}}^\alpha \dot{\gamma}^\alpha = \sum\nolimits_\alpha \left(\mathbf{G} : \tilde{\mathbf{P}}^\alpha\right)\dot{\gamma}^\alpha = \sum\nolimits_\alpha trace\left(\mathbf{G}\cdot\tilde{\mathbf{P}}^\alpha\right)\dot{\gamma}^\alpha$$

where $\mathbf{G} = \mathbf{g}^{-1}$. Substituting the definition of $\tilde{\mathbf{P}}^a$ in its covariant form we find,

$$\tilde{\mathbf{P}}^\alpha = \frac{1}{2}\left(\mathbf{g}\cdot\tilde{\mathbf{s}}^{(\alpha)} \otimes \tilde{\mathbf{n}}^{(\alpha)} + \tilde{\mathbf{n}}^{(\alpha)} \otimes \tilde{\mathbf{s}}^{(\alpha)} \cdot \mathbf{g}\right), \text{ where}$$

$\tilde{\mathbf{s}}^{(\alpha)}$ is defined with contravariant components, and $\tilde{\mathbf{n}}^{(\alpha)}$ with covariant components

Thus,

$$\sum\nolimits_\alpha trace\left(\mathbf{G}\cdot\tilde{\mathbf{P}}^\alpha\right)\dot{\gamma}^\alpha = \frac{1}{2}\sum\nolimits_\alpha trace\left(\mathbf{G}\cdot\left(\mathbf{g}\cdot\tilde{\mathbf{s}}^{(\alpha)} \otimes \tilde{\mathbf{n}}^{(\alpha)} + \tilde{\mathbf{n}}^{(\alpha)} \otimes \tilde{\mathbf{s}}^{(\alpha)} \cdot \mathbf{g}\right)\right)\dot{\gamma}^\alpha$$

$$\sum\nolimits_\alpha trace\left(\mathbf{G}\cdot\tilde{\mathbf{P}}^\alpha\right)\dot{\gamma}^\alpha = \frac{1}{2}\sum\nolimits_\alpha trace\left(\left(\tilde{\mathbf{s}}^{(\alpha)} \otimes \tilde{\mathbf{n}}^{(\alpha)} + \mathbf{G}\cdot\tilde{\mathbf{n}}^{(\alpha)} \otimes \tilde{\mathbf{s}}^{(\alpha)} \cdot \mathbf{g}\right)\right)\dot{\gamma}^\alpha$$

In index notation, and noting that the **g** lowers an index and **G** raises an index,

$$\sum\nolimits_\alpha trace\left(\mathbf{G}\cdot\tilde{\mathbf{P}}^\alpha\right)\dot{\gamma}^\alpha = \frac{1}{2}\sum\nolimits_\alpha trace\left(\tilde{s}^{\,i(\alpha)}\tilde{n}_j^{\,(\alpha)} + \tilde{n}^{\,i(\alpha)}\tilde{s}_j^{\,(\alpha)}\right)\dot{\gamma}^\alpha$$

$$= \frac{1}{2}\sum\nolimits_\alpha \left(\tilde{s}^{\,i(\alpha)}\tilde{n}_i^{(\alpha)} + \tilde{n}^{\,i(\alpha)}\tilde{s}_i^{(\alpha)}\right)\dot{\gamma}^\alpha = 0$$

By the orthogonality of $\tilde{\mathbf{n}}^{(\alpha)}$ and $\tilde{\mathbf{s}}^{(\alpha)}$.

13.4 Show that the second of Eq. 13.8.5 results from the skew symmetry of $\boldsymbol{\Omega}^*$.

Solution to 13.4

Beginning with the relation,

$$\dot{\tau}^{(\alpha)} = \left(\overset{\nabla}{\boldsymbol{\sigma}}^{dev} + \boldsymbol{\Omega}^* \boldsymbol{\sigma}^{dev} - \boldsymbol{\sigma}^{dev} \boldsymbol{\Omega}^* \right) : \mathbf{P}^{\alpha} + \boldsymbol{\sigma}^{dev} : \left(\boldsymbol{\Omega}^* \mathbf{P}^{\alpha} - \mathbf{P}^{\alpha} \boldsymbol{\Omega}^* \right)$$

In index notation,

$$\dot{\tau}^{(\alpha)} = \overset{\nabla}{\sigma}{}^{dev}_{ij} P^{\alpha}_{ij} + \left(\sigma_{kj}{}^{dev} \Omega^*_{ik} P^{\alpha}_{ij} - \sigma_{ik}{}^{dev} \Omega^*_{kj} P_{ij}{}^{\alpha} + \sigma_{ij}{}^{dev} \Omega^*_{ik} P^{\alpha}_{kj} - \sigma_{ij}{}^{dev} \Omega^*_{kj} P_{ik}{}^{\alpha} \right)$$

Substituting $\Omega = -\Omega^T$ for the third and fourth terms, and rearranging terms we find,

$$\dot{\tau}^{(\alpha)} = \overset{\nabla}{\sigma}{}^{dev}_{ij} P^{\alpha}_{ij} + \left(\sigma_{kj}{}^{dev} \Omega^*_{ik} P^{\alpha}_{ij} - \sigma_{ij}{}^{dev} \Omega^*_{ki} P^{\alpha}_{kj} \quad - \sigma_{ik}{}^{dev} \Omega^*_{kj} P_{ij}{}^{\alpha} + \sigma_{ij}{}^{dev} \Omega^*_{jk} P_{ik}{}^{\alpha} \right)$$

Swapping *i* and *k* indices in the second term, and *j* and *k* in the third term,

$$\dot{\tau}^{(\alpha)} = \overset{\nabla}{\sigma}{}^{dev}_{ij} P^{\alpha}_{ij} + \left(\sigma_{kj}{}^{dev} \Omega^*_{ik} P^{\alpha}_{ij} - \sigma_{kj}{}^{dev} \Omega^*_{ik} P^{\alpha}_{ij} \quad - \sigma_{ij}{}^{dev} \Omega^*_{jk} P_{ik}{}^{\alpha} + \sigma_{ij}{}^{dev} \Omega^*_{jk} P_{ik}{}^{\alpha} \right)$$

Thus,

$$\dot{\tau}^{(\alpha)} = \overset{\nabla}{\boldsymbol{\sigma}}{}^{dev} : \mathbf{P}^{\alpha}$$

13.5 Find the explicit expression for $\mathcal{W}$, in Eq. 13.5.4, in component form. (*Hint*: See derivation in [9].)

Solution to 13.5

Beginning from the definition

$$\mathbf{L} = \dot{\mathbf{F}}^e \mathbf{F}^{e-1} + \mathbf{F}^e \dot{\mathbf{F}}^p \mathbf{F}^{p-1} \mathbf{F}^{e-1}$$

where $\bar{\mathbf{D}}^p + \bar{\mathbf{W}}^p = \dot{\mathbf{F}}^p \mathbf{F}^{p-1}$.

Post multiplying by $\mathbf{F}^e$,

$$\mathbf{L}\mathbf{F}^e = \dot{\mathbf{F}}^e + \mathbf{F}^e\dot{\mathbf{F}}^p\mathbf{F}^{p-1}$$

Solving for $\dot{\mathbf{F}}^e$

$$\dot{\mathbf{F}}^e = \left(\mathbf{D}+\mathbf{W}\right)\mathbf{F}^e - \mathbf{F}^e\left(\bar{\mathbf{D}}^p + \bar{\mathbf{W}}^p\right)$$

Using the symmetry of $\dot{\mathbf{F}}^e$ we find,

$$\dot{\mathbf{F}}^e = \dot{\mathbf{F}}^{eT} = \mathbf{F}^e\left(\mathbf{D}-\mathbf{W}\right) - \left(\bar{\mathbf{D}}^p - \bar{\mathbf{W}}^p\right)\mathbf{F}^e$$

Subtracting the two expressions, for $\dot{\mathbf{F}}^e$

$$\mathbf{F}^e\left(\mathbf{D}-\mathbf{W}\right) - \left(\bar{\mathbf{D}}^p - \bar{\mathbf{W}}^p\right)\mathbf{F}^e = \left(\mathbf{D}+\mathbf{W}\right)\mathbf{F}^e - \mathbf{F}^e\left(\bar{\mathbf{D}}^p + \bar{\mathbf{W}}^p\right)$$

Regrouping terms

$$\left(\mathbf{D}+\bar{\mathbf{D}}^p\right)\mathbf{F}^e - \mathbf{F}^e\left(\bar{\mathbf{D}}^p + \mathbf{D}\right) = \left(\bar{\mathbf{W}}^p - \mathbf{W}\right)\mathbf{F}^e + \mathbf{F}^e\left(\bar{\mathbf{W}}^p - \mathbf{W}\right)$$

Now

$$\bar{\mathbf{F}}^e\left(\bar{\mathbf{W}}^p - \mathbf{W}\right) + \left(\bar{\mathbf{W}}^p - \mathbf{W}\right)\bar{\mathbf{F}}^e = \bar{W} : \left(\bar{\mathbf{W}}^p - \mathbf{W}\right)$$

where $\bar{W}_{ijkl} = \left(\bar{F}_{ik}{}^e\delta_{jl} + \delta_{ik}\bar{F}_{jl}{}^e\right)$

Thus

$$\bar{\mathbf{W}}^p = \mathbf{W} + \bar{W}^{-1}\left(\left(\mathbf{D}+\bar{\mathbf{D}}^p\right)\bar{\mathbf{F}}^e - \bar{\mathbf{F}}^e\left(\mathbf{D}+\bar{\mathbf{D}}^p\right)\right)$$

Also,

$$\left(\mathbf{D}+\bar{\mathbf{D}}^p\right)\bar{\mathbf{F}}^e - \bar{\mathbf{F}}^e\left(\mathbf{D}+\bar{\mathbf{D}}^p\right) = \bar{V} : \left(\mathbf{D}+\bar{\mathbf{D}}^p\right)$$

where $\bar{V}_{ijkl} = \left(\delta_{ik}\bar{F}_{jl}{}^e - \bar{F}_{ik}{}^e\delta_{jl}\right)$.

Therefore,

$$\bar{\mathbf{W}}_{ij}{}^p = \mathbf{W}_{ij} + \bar{W}^{-1}_{ijmn}\bar{V}_{mnkl}\left(\mathbf{D}+\bar{\mathbf{D}}^p\right)_{kl}$$

Hence $\mathcal{W}_{ijkl} = \bar{W}^{-1}_{ijmn}\bar{V}_{mnkl}$.

13.6 Computer problem.

(a) Write a poly-slip, rate-dependent, single crystal plasticity subroutine, based on Box 13.2 and Table 13.3, to reproduce the results in Fig. 13.11 to Fig. 13.13.

(b) Change the Euler angles to (0°, 60°, 0°) and compare the deformation pattern at 30% nominal strain. Use Box 13.1 to help re-define your slip directions and normals correctly, or use Table E13.1.

Solution to 13.6 (a)

Here we list a FORTRAN material subroutine, VUMAT to be used in ABAQUS\Explicit. The results of this code reproduce Fig. 13.11 to Fig. 13.13 based on Box 13.2 and Table 13.3.

The code, however, does not present the details of the ODE solver (ODEINT.inc), since different available solvers could be chosen from. Nonetheless, the parameters to be passed in and out of the ODE solver are clearly indicated and setup.

Solution to 13.6 (b)

The same code will be used for this part. Only, Table E13.1 should be used to define the initial values for slip directions and normals.

% FORTRAN CODE FOR PROBLEM 13.6 %

```
C #=====================================================================
C                                                          _______
C                 \    /  |    |   |\  /|    /\        |
C                  \  /   |    |   | \/ |   /__\       |
C                   \/    |____|   |    |  /    \      |
C
C     This subroutine is the MAIN MATERIAL SUBROUTINE:
C     It updates: stresses, internal variables (SDV array), and energies
C
C     Khalil El Khodary <khalile@aucegypt.edu>
C     Assistant Professor at the American University in Cairo
C =====================================================================
      subroutine vumat (
C Read only -
     1     nblock, ndir, nshr, nstatev, nfieldv, nprops, lanneal,
     2     stepTime, totalTime, dt, cmname, coordMp, charLength,
     3     props, density, strainInc, relSpinInc,
     4     tempOld, stretchOld, defgradOld, fieldOld,
     5     stressOld, stateOld, enerInternOld, enerInelasOld,
     6     tempNew, stretchNew, defgradNew, fieldNew,
C Write only -
     7     stressNew, stateNew, enerInternNew, enerInelasNew )
      include 'vaba_param.inc'
      include 'DataInitialize.inc'
      include 'ElasticStarter.inc'
C
C #=====================================================================
C            #   LOADING MATERIAL DATA FROM INPUT FILE  #
C           (independent of Element), (nprops = 7*nslip+27)
C         Note: Not all data in props is loaded here.
C          Some will be loaded when needed
C #=====================================================================
        nslip        = int(props(1))
        e            = props(2)
        xnu           = props(3)
        yield         = props(4)
        g_source     = props(10)
        g_immob      = props(11)
        g_minter     = props(12)
        g_recov      = props(13)
        Cp           = props(14)
        tempR        = props(15)

        H_K           = props(16)
        xi           = props(17)
        Chi          = props(18)
        EM_factor    = props(20)
        b_v(1:nslip) = props(21:20+nslip)
C
```

```
C Derived material constants
        g         = e/(two*(one + xnu))
        twomu     = two*g
        bulk      = e/three/( one - two * xnu )

C
C Note H/K is assumed constant, so temp must stay at tempR
        thermal_coef = H_K/tempR
C
C Derived Pointers/Flags
        ibeginN   = 20 + nslip
        ibeginS   = nslip*3 + ibeginN

C #====================================================================
C                                       # WORKING WITH ELEMENT "k" #
C #====================================================================
         do k = 1,nblock
C Load Internal variables independent of slip system
          Psi21    = stateOld( k,1)*pi/180.d0
          Psi32    = stateOld( k,2)*pi/180.d0
          Psi13    = stateOld( k,3)*pi/180.d0
          gamma    = stateOld( k,4)
          temp     = stateOld( k,6) + tempR
C Load Internal variables dependent on slip system
          do j = 1, nslip

                  tau(j)        = stateOld( k,8  + (j-1)*10 )
                  gdot(j)       = stateOld( k,9  + (j-1)*10 )
                  den_im(j)     = stateOld( k,10 + (j-1)*10 )
                  den_m(j)      = stateOld( k,11 + (j-1)*10 )
                  slip_n(j,1)   = stateOld( k,12 + (j-1)*10 )
                  slip_n(j,2)   = stateOld( k,13 + (j-1)*10 )
                  slip_n(j,3)   = stateOld( k,14 + (j-1)*10 )
                  slip_s(j,1)   = stateOld( k,15 + (j-1)*10 )
                  slip_s(j,2)   = stateOld( k,16 + (j-1)*10 )
                  slip_s(j,3)   = stateOld( k,17 + (j-1)*10 )

          end do
C
C Compute derived slip-specific mechanical constants
          do j = 1, nslip
             recip_m(j)  = one / props(6)
             ref_gdot(j) = props(7)
          end do
C

C    ------------------------------------------------------------------
C       # IF APPLICABLE: REPLACE ZEROED VARIABLES WITH INITIAL DATA #
C    ------------------------------------------------------------------
          if (dt .eq. totalTime) then
             do i = 1, nslip

                 den_im(i) = props(8)
                 den_m (i) = props(9)
```

```
                  do j =  1, 3
                     slip_n(i,j) = props( ibeginN + 3*(i-1) + j )
                     slip_s(i,j) = props( ibeginS + 3*(i-1) + j )
                  enddo
               enddo
            endif
C                        ------------------------------------------------
C                        #  COMPUTE TOTAL SPIN and RATE OF DEFORMATION  #
C                        #       COMPUTE ROTATION FROM F = RU           #
C                        #       (Rnew is the Transpose of R)           #
C                        ------------------------------------------------
C Compute Spin, Dij and Rotation tensors
         call getSpinDij(dt,defgradNew(k,:),defgradOld(k,:),Spin,Dij)
         call getRot(defgradNew(k,:),stretchNew(k,:),Rnew)
C
C Deviatoric Dij tensor
         traceDij = Dij(1) + Dij(2) + Dij(3)
         Dij_dev(1) = Dij(1) - traceDij/3.d0
         Dij_dev(2) = Dij(2) - traceDij/3.d0
         Dij_dev(3) = Dij(3) - traceDij/3.d0
         Dij_dev(4) = Dij(4)
         Dij_dev(5) = Dij(5)
         Dij_dev(6) = Dij(6)
C
C                        ------------------------------------------------
C                         #               COMPUTE Pij & Wij              #
C                        ------------------------------------------------

C SYMMETRIC SCHMID TENSOR (EXPRESSED W.R.T GLOBAL AXES)
C This is for the sake of Dij tensor compuations.
         do j = 1, nslip
               p(j,1) = 0.5d0*(slip_s(j,1)*slip_n(j,1) +
     1                         slip_s(j,1)*slip_n(j,1))
               p(j,2) = 0.5d0*(slip_s(j,2)*slip_n(j,2) +
     1                         slip_s(j,2)*slip_n(j,2))
               p(j,3) = 0.5d0*(slip_s(j,3)*slip_n(j,3) +
     1                         slip_s(j,3)*slip_n(j,3))
               p(j,4) = 0.5d0*(slip_s(j,1)*slip_n(j,2) +
     1                         slip_s(j,2)*slip_n(j,1))
               p(j,5) = 0.5d0*(slip_s(j,2)*slip_n(j,3) +
     1                         slip_s(j,3)*slip_n(j,2))
               p(j,6) = 0.5d0*(slip_s(j,3)*slip_n(j,1) +
     1                         slip_s(j,1)*slip_n(j,3))
         end do
C
C ANTI-SYMMETRIC SCHMID TENSOR (EXPRESSED W.R.T GLOBAL AXES)
C This is for the sake of spin tensor compuations, which are
C also computed and expressed w.r.t. Initial (global) Axes
         do j = 1, nslip
               w12(j) = 0.5d0 * (slip_s(j,1)*slip_n(j,2) -

     1                                slip_s(j,2)*slip_n(j,1))
               w23(j) = 0.5d0 * (slip_s(j,2)*slip_n(j,3) -
```

```
     1                                slip_s(j,3)*slip_n(j,2))
                w31(j) = 0.5d0 * (slip_s(j,3)*slip_n(j,1) -

     1                                slip_s(j,1)*slip_n(j,3))
          end do
C
C
C                       ------------------------------------------------
C                        #                    PijDij                   #
C                                    For the Computation of Tau_dot
C                       ------------------------------------------------
          do j = 1, nslip
               PijDijDev(j)= p(j,1)*Dij_dev(1) +      p(j,2)*Dij_dev(2)
     1                   + p(j,3)*Dij_dev(3) + 2.d0*p(j,4)*Dij_dev(4)
     2                 +2.d0*p(j,5)*Dij_dev(5) + 2.d0*p(j,6)*Dij_dev(6)
          end do
C
C                       ------------------------------------------------
C                        #  UPDATE Resolved Shear stress: TAU (alpha)  #
C                       ------------------------------------------------
C
C Thermal Multiplier
          thermal_factor = (tempR/temp)**xi
          tauR(1:maxSS) = 0.d0
C
C Compute Tau Reference (crystal strength)
          do j = 1, nslip
             do kk = 1,nslip
                alphaInt = abs(    p(j,1)*p(kk,1) +    p(j,2)*p(kk,2)
     1                         +     p(j,3)*p(kk,3) + 2.*p(j,4)*p(kk,4)
     2                         +  2.*p(j,5)*p(kk,5) + 2.*p(j,6)*p(kk,6) )/0.5d0
                tauR(j) = tauR(j) + alphaInt*g*b_v(kk)*sqrt(den_im(kk))
             enddo
          enddo
          tauR(1:nslip) = (tauR(1:nslip) +yield/EM_factor)*thermal_factor
C
C Define Integration Parameters to pass to ODEINT (initial value ODE problem)
          tauStart(1:nslip) = tau(1:nslip)
         NVAR = nslip
         Time = totalTime - dt
         eps  = 0.01d0
         hmin = dt/24.d0
C PACK Parameters into Array PAR to send to tauDot and tauResidual
C These are all required input to solve for the resolved shear stress
         PAR (1)                        = dble(nslip)
         PAR (         2 :    NVAR+1 ) = recip_m(1:nslip)
         PAR (   NVAR+2 :  2*NVAR+1 ) = ref_gdot(1:nslip)
         PAR ( 2*NVAR+2 :  3*NVAR+1 ) = PijDijDev(1:nslip)
         PAR ( 3*NVAR+2 :  4*NVAR+1 ) = p(1:nslip,1)
         PAR ( 4*NVAR+2 :  5*NVAR+1 ) = p(1:nslip,2)
         PAR ( 5*NVAR+2 :  6*NVAR+1 ) = p(1:nslip,3)
         PAR ( 6*NVAR+2 :  7*NVAR+1 ) = p(1:nslip,4)
         PAR ( 7*NVAR+2 :  8*NVAR+1 ) = p(1:nslip,5)
         PAR ( 8*NVAR+2 :  9*NVAR+1 ) = p(1:nslip,6)
```

```
        PAR ( 9*NVAR+2 : 10*NVAR+1 ) = tau(1:nslip)
        PAR (10*NVAR+2 : 11*NVAR+1 ) = tauR(1:nslip)
        PAR (11*NVAR+2)              = twomu
        PAR (size(PAR,1)) = 0.d0 !This takes the value of time step increment
'h'
C Integrate tauDot using an ODE solver subroutine
        call odeint(tauDot, tauResidual, tauStart, NVAR,
     1               Time, Time + dt, eps, dt, hmin, PAR)
C Update Tau
        tau(1:nslip) = tauStart(1:nslip)
C
C                    ------------------------------------------------
C                    #  COMPUTE CRYSTALLOGRAPHIC SHEAR STRAIN RATE   #
C                    ------------------------------------------------

         do j = 1, nslip
C Power-law
            gdot(j) = ref_gdot(j)*(tau(j)/tauR(j))*
     1                (abs(tau(j)/tauR(j)))**(recip_m(j)-1.)
         end do
C                   ------------------------------------------------
C                   #  COMPUTE PLASTIC RATES OF DEFORMATION & SPIN  #
C                   ------------------------------------------------

C Initialize Values
         D11_p    = 0.d0
         D22_p    = 0.d0
         D33_p    = 0.d0
         D12_p    = 0.d0
         D23_p    = 0.d0
         D31_p    = 0.d0
         spin_p12 = 0.d0
         spin_p23 = 0.d0
         spin_p31 = 0.d0
         spin_21  = Spin(1)
         spin_32  = Spin(2)
         spin_13  = Spin(3)
         spin_12  =-Spin(1)
         spin_23  =-Spin(2)
         spin_31  =-Spin(3)
         do j = 1 , nslip
C Plastic Deformation Rate,
            if (abs(gdot(j)).gt.abs(ref_gdot(j))) then
                  D11_p  = D11_p    + p(j,1)*gdot(j)
                  D22_p  = D22_p    + p(j,2)*gdot(j)
                  D33_p  = D33_p    + p(j,3)*gdot(j)
                  D12_p  = D12_p    + p(j,4)*gdot(j)
                  D23_p  = D23_p    + p(j,5)*gdot(j)
                  D31_p  = D31_p    + p(j,6)*gdot(j)
                  spin_p12 = spin_p12 + w12(j)*gdot(j)
                  spin_p23 = spin_p23 + w23(j)*gdot(j)
                        spin_p31 = spin_p31 + w31(j)*gdot(j)
            endif
         end do
```

```
C
C
C
C                       ------------------------------------------------
C                       #                COMPUTE ELASTIC SPIN          #
C                       ------------------------------------------------
          spin_p21 = -spin_p12
          spin_p13 = -spin_p31
          spin_p32 = -spin_p23
          spin_e21 =  spin_21 - spin_p21
          spin_e12 = -spin_e21
          spin_e32 =  spin_32 - spin_p32
          spin_e23 = -spin_e32
          spin_e13 =  spin_13 - spin_p13
          spin_e31 = -spin_e13
C
C                       ------------------------------------------------
C                       # UPDATE/NORMALIZE SLIP NORMALS AND DIRECTIONS #
C                           NOTE: STILL REFERENCED TO GLOBAL AXES
C                       ------------------------------------------------

          do j = 1 , nslip
C n_dot and s_dot,
               an_dot1 = spin_e12*slip_n(j,2) + spin_e13*slip_n(j,3)
               an_dot2 = spin_e21*slip_n(j,1) + spin_e23*slip_n(j,3)
               an_dot3 = spin_e31*slip_n(j,1) + spin_e32*slip_n(j,2)
               as_dot1 = spin_e12*slip_s(j,2) + spin_e13*slip_s(j,3)
               as_dot2 = spin_e21*slip_s(j,1) + spin_e23*slip_s(j,3)
               as_dot3 = spin_e31*slip_s(j,1) + spin_e32*slip_s(j,2)
C n and s updated
               slip_n(j,1) = slip_n(j,1) + an_dot1*dt
               slip_n(j,2) = slip_n(j,2) + an_dot2*dt
               slip_n(j,3) = slip_n(j,3) + an_dot3*dt
               slip_s(j,1) = slip_s(j,1) + as_dot1*dt
               slip_s(j,2) = slip_s(j,2) + as_dot2*dt
               slip_s(j,3) = slip_s(j,3) + as_dot3*dt
          end do
C Normalize slip vectors to unit magnitude
          call unit_vector( 3,nslip,slip_n,slip_s)
C
C              ---------------------------------------------------------
C                # EXPRESS THE ROTATED STRESS AT BEGINNING OF INCREMENT  #
C                # AS CAUCHY STRESS:: S = R*S(corot)*R^T ,for R = FU^-1  #
C                # NOTE: RNEW is the Transpose of that in F=R*U          #
C              ---------------------------------------------------------

C Cauchy Stress
          Rnew = Transpose(Rnew)
          call ROTATE(stressNew(k,:),Rnew,stressOld(k,1),stressOld(k,2),
     1               stressOld(k,3),stressOld(k,4),stressOld(k,5),
     2               stressOld(k,6) )
          sigGLB1 = stressNew(k,1)
          sigGLB2 = stressNew(k,2)
          sigGLB3 = stressNew(k,3)
```

```
          sigGLB4 = stressNew(k,4)
          sigGLB5 = stressNew(k,5)
          sigGLB6 = stressNew(k,6)
          Rnew = Transpose(Rnew)

C COMPUTE THE DEVIATORIC PARTS
          press =  ( sigGLB1 + sigGLB2 + sigGLB3 )/3.d0
          sigGLB1 = sigGLB1  - press
          sigGLB2 = sigGLB2  - press
          sigGLB3 = sigGLB3  - press
C
C                        ------------------------------------------------
C                        #            UPDATE CAUCHY STRESSES             #
C                        ------------------------------------------------
C COMPUTE THE NON-CONSTITUTIVE PART OF STRESS RATE: "SIGMA.W* - W*.SIGMA"

          o11    = - 2.d0*( spin_e12*sigGLB4 + spin_e13*sigGLB6 )
          o22    = - 2.d0*( spin_e21*sigGLB4 + spin_e23*sigGLB5 )
          o33    = - 2.d0*( spin_e31*sigGLB6 + spin_e32*sigGLB5 )
          o12    =   spin_e12*sigGLB1 - spin_e12*sigGLB2 -
     1              spin_e13*sigGLB5 - spin_e23*sigGLB6

          o23    =   spin_e23*sigGLB2 - spin_e23*sigGLB3 -
     1              spin_e21*sigGLB6 - spin_e31*sigGLB4

          o31    =   spin_e13*sigGLB1 - spin_e13*sigGLB3 -
     1              spin_e32*sigGLB4 - spin_e12*sigGLB5
C
C UPDATE THE CAUCHY STRESSES, BUT EXPRESSED W.R.T. GLOBAL AXES
          sigGLB1 = sigGLB1 + dt*o11 + twomu*dt*(Dij_dev(1)-D11_p)
          sigGLB2 = sigGLB2 + dt*o22 + twomu*dt*(Dij_dev(2)-D22_p)
          sigGLB3 = sigGLB3 + dt*o33 + twomu*dt*(Dij_dev(3)-D33_p)
          sigGLB4 = sigGLB4 + dt*o12 + twomu*dt*(Dij_dev(4)-D12_p)
          sigGLB5 = sigGLB5 + dt*o23 + twomu*dt*(Dij_dev(5)-D23_p)
          sigGLB6 = sigGLB6 + dt*o31 + twomu*dt*(Dij_dev(6)-D31_p)
C
C              ------------------------------------------------
C               # UPDATE TEMPERATURE AND PLASTIC WORK INC#
C              ------------------------------------------------
C (Deviatoric) StressPower,
          DijSij    =       sigGLB1*D11_p  +       sigGLB2*D22_p +
     1                      sigGLB3*D33_p  +  2.d0*sigGLB4*D12_p +
     2                 2.d0*sigGLB5*D23_p  +  2.d0*sigGLB6*D31_p
C Update plastic work from stress power
          plasticWorkInc = dt*abs(DijSij)

          eta            = density(k)*Cp
C Adiabatic Temp Update,
          tempInc        = plasticWorkInc/eta* Chi
C Update normal stress components with volumetric part
          press  = press  + bulk*dt*(traceDij)
          sigGLB1 = sigGLB1 + press
          sigGLB2 = sigGLB2 + press
          sigGLB3 = sigGLB3 + press
```

```
C                              ------------------------------------------------
C                              #   STORE UPDATED COROTATIONAL STRESS :: R^T*S*R
#
C                              ------------------------------------------------
          call ROTATE(stressNew(k,:),Rnew,sigGLB1,sigGLB2,sigGLB3,
     1                                    sigGLB4,sigGLB5,sigGLB6)
C                         ------------------------------------------------
C                          # UPDATE INTERNAL VARIABLES: RHO_M & RHO_IM #
C                         ------------------------------------------------

C Define Integration Parameters to pass to ODEINT (initial value ODE problem)
          eps  = 0.05d0
          hmin = dt/24.d0
          Time = totalTime - dt
          NVAR = 2
C
C Reset PAR
          PAR( 1: size(PAR,1) )  = 0.d0
C PACK Parameters into Array PAR to send to rhoDot and rhoResidual
          PAR (       1        ) = dble(nslip)
          PAR (       2        ) = g_source
          PAR (       3        ) = g_immob
          PAR (       4        ) = g_minter
          PAR (       5        ) = g_recov
          PAR (       6        ) = thermal_coef
          PAR (      11    :   nslip+10 ) = gdot(1:nslip)
          PAR (  nslip+11 : 2*nslip+10 ) = b_v(1:nslip)
          PAR (   size(PAR,1)  ) = 0.d0!This later takes on a value'h'
C
C Call rho_dot integrator 'nslip' times
          do j = 1, nslip
             rhoStart(1) = den_im(j)!Rho_im Starting Value
             rhoStart(2) = den_m(j) !Rho_ m Starting Value
             PAR (     2*nslip+11    ) = dble(j) !assigns jslip
             PAR (2*nslip+12:2*nslip+13) = rhoStart(1:2)
C
C Call Nested Subroutines for Adaptive RK5 Integration or Back Euler
             call odeint(rhoDot,rhoResidual,rhoStart, NVAR,
     1                   Time, Time + dt, eps, dt, hmin, PAR )
             den_im(j) = rhoStart(1)      !Rho_im Updated Value
             den_m(j)  = rhoStart(2)      !Rho_ m Updated Value
          end do
C
C Reset PAR
         PAR( 1: size(PAR,1) ) = 0.d0
C
C                         ------------------------------------------------
C                             # UPDATE ALL OTHER INTERNAL VARIABLES #
C                         ------------------------------------------------

C Increment in shear slip:: Using an 'effective' measure;
C dgamma = dt*sqrt([D_p]:[D_p])
         dgamma = dt*sqrt(( D11_p**2.d0 +     D22_p**2.d0 +     D33_p**2.d0+
     1               2.d0*D12_p**2.d0 +2.d0*D23_p**2.d0 +2.d0*D31_p**2.d0 ))
```

```
C
C
C
C Internal variables not dependent on slip system
        Psi21 = Psi21 + spin_e21*dt
        Psi32 = Psi32 + spin_e32*dt
        Psi13 = Psi13 + spin_e13*dt
       stateNew( k,1) = Psi21*(180.d0/pi)!Angle21 that Slip systems rotated,
        stateNew( k,2) = Psi32*(180.d0/pi)!Angle32 that Slip systems rotated,
        stateNew( k,3) = Psi13*(180.d0/pi)!Angle13 that Slip systems rotated,
         stateNew( k,4) = stateOld( k,4) + dgamma     !New Shear Slip,
        stateNew( k,5) = sum(tauR(1:nslip))/nslip     !New Reference Tau
         stateNew( k,6) = stateOld( k,6) + tempInc    !New Temperature
C
C Internal variables dependent on slip system
        do j = 1, nslip                   !Loop over slip-system alpha:
           stateNew( k,8  + (j-1)*10 ) = tau(j) !shear stress
           stateNew( k,9  + (j-1)*10 ) = gdot(j)!shear strain-rate
           stateNew( k,10 + (j-1)*10 ) = den_im(j) !rho mobile
           stateNew( k,11 + (j-1)*10 ) = den_m(j)!rho immobile
           stateNew( k,12 + (j-1)*10 ) = slip_n(j,1)!slip plane normal (x)
          stateNew( k,13 + (j-1)*10 ) = slip_n(j,2)!slip plane normal (y)
           stateNew( k,14 + (j-1)*10 ) = slip_n(j,3)!slip plane normal (z)
           stateNew( k,15 + (j-1)*10 ) = slip_s(j,1)!slip direction    (x)
          stateNew( k,16 + (j-1)*10 ) = slip_s(j,2)!slip direction    (y)
          stateNew( k,17 + (j-1)*10 ) = slip_s(j,3)!slip direction    (z)
        end do

C                      ------------------------------------------------
C                                        # UPDATE THE ENERGIES #
C                      ------------------------------------------------

C Update the dissipated inelastic specific energy -
       enerInelasNew( k) = enerInelasOld( k) +
     1                     plasticWorkInc / density( k)
C Update the specific internal energy -
        stressPower = half * (
     1      ( stressOld( k,1)+stressNew( k,1) ) * (Dij(1) - D11_p)*dt +
     2      ( stressOld( k,2)+stressNew( k,2) ) * (Dij(2) - D22_p)*dt +
     3      ( stressOld( k,3)+stressNew( k,3) ) * (Dij(3) - D33_p)*dt ) +
     4      ( stressOld( k,4)+stressNew( k,4) ) * (Dij(4) - D12_p)*dt +
     5      ( stressOld( k,5)+stressNew( k,5) ) * (Dij(5) - D23_p)*dt +
     6      ( stressOld( k,6)+stressNew( k,6) ) * (Dij(6) - D31_p)*dt
C
        enerInternNew( k) = enerInternOld( k)
     1                      + stressPower / density( k)
      enddo
       return
      end
C #====================================================================
      include 'TensorSubs.inc'
      include 'UpdateRhos.inc'
      include 'UpdateTaus.inc'
      include 'ODEINT.inc'
```

```
C ======================================================================
```

C ElasticStarter.inc

```
C     ****************************************************************
C           THIS LOOP IS FOR ABAQUS TO COMPUTE INITIAL ELASTIC WAVE SPEED
C     ****************************************************************
      if ( totalTime .eq. zero ) then
            do k = 1,nblock
C     ------------------------------------------------
C                 #  MATERIAL DATA  #
C     ------------------------------------------------
C     material properties not dependent on slip system
              e         = props(2)
              xnu       = props(3)
C     derived mechanical constants
              g         = e/(two*(one + xnu))
              twomu     = two*g
              bulk      = e/three/( one - two * xnu )

C     ------------------------------------------------
C             # COMPUTING VUMATS INITIAL STRESSES #
C     ------------------------------------------------
C     Trial/Elastic stress
          trace = (strainInc(k,1) + strainInc(k,2) + strainInc(k,3))
          stressNew(k,1) = stressOld(k,1)
     1         + twomu * strainInc(k,1) + bulk * trace
          stressNew(k,2) = stressOld(k,2)
     1         + twomu * strainInc(k,2) + bulk * trace
          stressNew(k,3) = stressOld(k,3)
     1         + twomu * strainInc(k,3) + bulk * trace
          stressNew(k,4) = stressOld(k,4)  +
     1                     twomu * strainInc(k,4)
          stressNew(k,5) = stressOld(k,5)  +
     1                        twomu * strainInc(k,5)
          stressNew(k,6) = stressOld(k,6)  +
     1                        twomu * strainInc(k,6)
            enddo
            return
         endif
C      ****************************************************************
C           THIS LOOP IS FOR ABAQUS TO COMPUTE INITIAL ELASTIC WAVE SPEED
C     ****************************************************************
```

C Datainitialize.inc

```
C #====================================================================
C #                  ABAQUS-SPECIFIC ARRAYS/VARIABLES
C #
      dimension coordMp(nblock,*), charLength(nblock),
     1     props(nprops),density(nblock), strainInc(nblock,ndir+nshr),
     2     relSpinInc(nblock,nshr), tempOld(nblock),
     3     stretchOld(nblock,ndir+nshr),
     4     defgradOld(nblock,ndir+nshr+nshr),
     5     fieldOld(nblock,nfieldv), stressOld(nblock,ndir+nshr),
     6     stateOld(nblock,nstatev), enerInternOld(nblock),
     7     enerInelasOld(nblock), tempNew(nblock),
     8     stretchNew(nblock,ndir+nshr),
     9     defgradNew(nblock,ndir+nshr+nshr),
     1     fieldNew(nblock,nfieldv),
     2     stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev),
     3     enerInternNew(nblock), enerInelasNew(nblock)
      character*80 cmname

C #  ========#    CRYSTAL PLASTICITY ARRAYS/VARIABLES #==============
        real*8, PARAMETER:: zero = 0.d0,one = 1.d0,two = 2.d0,
     1            three = 3.d0,half = 0.5d0,pi = 3.141592654d0
C Integer Scalars
        integer, parameter:: maxSS = 14
        integer  i,j,k,kk, nslip, NVAR,IBEGINN, IBEGINS, IER
C
C Real Scalars
        real*8 g_source, g_immob, g_minter, g_recov, thermal_coef
        real*8 tauOld,g,e,xnu,trace,yield
        real*8 Cp, tempR, H_k, xi, Chi, alphaInt, EM_factor
       real*8 TWOMU,BULK, PSI21
      real*8 PSI32, PSI13, GAMMA, TEMP
       real*8 TRACEDIJ, THERMAL_FACTOR,STRESSPOWER
       real*8 TIME, EPS, HMIN, PRESS, DIJSIJ
       real*8 D11_P, D22_P, D33_P, D12_P, D23_P, D31_P
       real*8 SPIN_21, SPIN_32, SPIN_13,SPIN_12, SPIN_23, SPIN_31
       real*8 SPIN_P12, SPIN_P23, SPIN_P31, SPIN_P21, SPIN_P32, SPIN_P13
       real*8 SPIN_E12, SPIN_E23, SPIN_E31, SPIN_E21, SPIN_E32, SPIN_E13
       real*8 AN_DOT1, AN_DOT2, AN_DOT3, AS_DOT1, AS_DOT2, AS_DOT3
       real*8 sigGLB1, sigGLB2, sigGLB3, sigGLB4, sigGLB5, sigGLB6
       real*8 O11, O22, O33, O12, O23, O31
      real*8 PLASTICWORKINC, ETA, TEMPINC, DGAMMA,SNRM2
C
C Vectorial/Matrix Definitions
         real*8 Dij_dev(6),w12(maxSS),w23(maxSS), w31(maxSS),Dij(6)
         real*8 slip_n(maxSS,3),slip_s(maxSS,3),p(maxSS,6)
        real*8 Spin(3),PijDijDev(maxSS),Rnew(3,3),tauStart(maxSS)
        real*8 rhoStart(2), tau(maxSS), recip_m(maxSS),TAUR(maxSS)
        real*8 den_im(maxSS),den_m(maxSS),PAR(10*maxSS+4)
        real*8 ref_gdot(maxSS),gdot(maxSS),b_v(maxSS),PijDij(maxSS)

C #====================================================================
        external tauDot, tauResidual, rhoDot, rhoResidual
```

```
C #=====================================================================
```

```
C #====================================================================
C                --------   /\                 _____    _____   --------
C                   |      /  \    |      | |     \  |     |    |
C                   |     /____\   |      | |      | |     |    |
C                   |    /      \  |      | |      | |     |    |
C                   |   /        \ |------| |_____/  |_____|    |
C                              Computes tau dot
c  ...  Define the system of nonlinear differential equations for each
c  ...  active slip-system (k)
c            tau_dot(k)   = 2*mu*Pij(k)*[Dij_dev - Dij_p_dev]
c            Dij_p_dev    = Pij(n)*gamma_dot(n)
c            gamma_dot(n) = ref_gamma_dot(n)*[tau(n)/tau_r(n)]**(1/m)
C #====================================================================
      subroutine tauDot ( y, yprime, PAR)
      include 'vaba_param.inc'
      integer, parameter:: maxSS = 14
      integer nslip, i, j
      real*8 TWOMU, DUMMY
      real*8 PAR(10*maxSS+4)
      real*8 p(maxSS,6), TAUR(maxSS)
      real*8 recip_m(maxSS), ref_gdot(maxSS), PijDijDev(maxSS)
      real*8 y(maxSS), yprime(maxSS), r(maxSS), g(maxSS)
C UN-PACK Parameters from Array PAR
       nslip              = int( PAR (1)  )
       recip_m(1:nslip)   = PAR (         2 :   nslip+1 )
       ref_gdot(1:nslip)  = PAR (   nslip+2 : 2*nslip+1 )
       PijDijDev(1:nslip) = PAR ( 2*nslip+2 : 3*nslip+1 )
       p(1:nslip,1)       = PAR ( 3*nslip+2 : 4*nslip+1 )
       p(1:nslip,2)       = PAR ( 4*nslip+2 : 5*nslip+1 )
       p(1:nslip,3)       = PAR ( 5*nslip+2 : 6*nslip+1 )
       p(1:nslip,4)       = PAR ( 6*nslip+2 : 7*nslip+1 )
       p(1:nslip,5)       = PAR ( 7*nslip+2 : 8*nslip+1 )
       p(1:nslip,6)       = PAR ( 8*nslip+2 : 9*nslip+1 )
       tauR(1:nslip)      = PAR (10*nslip+2 :11*nslip+1 )
       twomu              = PAR (11*nslip+2)
      dummy = 0.0
      do i = 1, nslip
         r(i) = ((abs(y(i)/tauR(i)))**(recip_m(i)-1.0)) *
     1          (y(i)/tauR(i))
         g(i) = 0.0
       end do
       do i = 1, nslip
          do j = 1, nslip
             dummy = r(j)*ref_gdot(j) *  ( p(i,1)*p(j,1) +
     1                                     p(i,2)*p(j,2) +
     2                                     p(i,3)*p(j,3) +
     3                                   2.*p(i,4)*p(j,4) +
     4                                   2.*p(i,5)*p(j,5) +
     5                                   2.*p(i,6)*p(j,6) )
             g(i) = g(i) + dummy
          end do
       end do
       do i = 1, nslip
          yprime(i) = twomu*( PijDijDev(i) - g(i) )
```

```
      end do
      return
      end
C #=====================================================================
C #=====================================================================
C           --------   /\               -----   -----   -----
C              |      /  \    |      | |     | |       |
C              |     /____\   |      | |_____| |_____  |_____
C              |    /      \  |      | |   \   |             |
C              |   /        \ |______| |    \  |_____   _____|
C #=====================================================================

      subroutine tauResidual(y,f,n,PAR)
      include 'vaba_param.inc'
      integer, parameter:: maxSS=14
      integer n, nslip, i
      real*8 dt
      real*8 y(n), yold(n), yprime(n), f(n), PAR(10*maxSS+4)
C UN-PACK Parameters  from Array PAR
      nslip      = int( PAR(1) )
      yold(1:n)  = PAR ( 9*nslip+2 :10*nslip+1 )
      dt         = PAR ( size(PAR,1) )
      call tauDot( y, yprime, PAR)
      do i = 1, n
        f(i) = y(i)-yold(i)-dt*yprime(i)
      end do

      return
      end
C #=====================================================================
```

```
#=======================================================================
C                ____          _____      _____    _____  _______
C              |     | |     | |     |    |     \  |     |    |
C              |_____| |_____| |     |    |      | |     |    |
C              |   \   |     | |     |    |      | |     |    |
C              |    \  |     | |_____|    |_____/  |_____|    |
C         Computes dRho_m(alpha)/dt and dRho_im(alpha)/dt
#=======================================================================
      subroutine rhoDot (y,yprime,PAR)
      include 'vaba_param.inc'
C Integer Scalar
      integer,parameter:: maxSS = 14
      integer NSLIP, JSLIP
C Real Scalar
      real*8 G_SOURCE, G_IMMOB, G_MINTER, G_RECOV, THERMAL_COEF
      real*8 ABSGDOT, C1, C2, C3, C4, C5
C Vectorial/Matrix Definitions
      real*8 y(2),yprime(2), gdot(maxSS), b_v(maxSS)
      real*8 PAR(10*maxSS+4)
C Un-PACK Parameters  from Array PAR
      nslip         = int ( PAR(1) )
      g_source      = PAR ( 2 )
      g_immob       = PAR ( 3 )
      g_minter      = PAR ( 4 )
      g_recov       = PAR ( 5 )
      thermal_coef  = PAR ( 6 )
      gdot(1:nslip) = PAR (    11    :   nslip+10 )
      b_v( 1:nslip) = PAR ( nslip+11 : 2*nslip+10 )
      jslip         = int (   PAR ( 2*nslip+11 )  )
C Compute Local Parameters
      absGdot = abs(gdot(jslip))
      c1 = g_immob/b_v(jslip)
      c2 = g_recov
      c3 = g_minter*y(2)
      c4 = g_source/(b_v(jslip)*b_v(jslip))
      c5 = - thermal_coef
      yprime(1) = absGdot* ( -c2*exp(c5)*y(1) + c3*exp(c5)
     1                                        + c1*sqrt(y(1)) ) !rho_im_dot
      yprime(2) = absGdot* (   c4*y(1)/y(2)   - c3*exp(c5)
     1                                        - c1*sqrt(y(1)) ) !rho_m_dot
      return
      end
 =======================================================================
```

```
C #======================================================================
C                    ____          _____      _____    _____   ____
C                   |     | |     | |     |    |     |  |       |
C                   |_____| |____| |     |    |_____|  |____   |____
C                   |   \   |     | |     |    |   \    |            |
C                   |    \  |     | |_____|    |    \   |_____  ____|
C #======================================================================
      subroutine rhoResidual(y,f,n,PAR)
      include 'vaba_param.inc'
      integer, parameter:: maxSS=14
      integer n, nslip
      real*8 dt
      real*8 y(n),yold(n),yprime(n),f(n),PAR(10*maxSS+4)
C Un-PACK relevant parameters  from Array PAR
      nslip     = int ( PAR(1) )
      yold(1:n) = PAR (2*nslip+12:2*nslip+13)
      dt        = PAR ( size(PAR,1) )
C
      call rhoDot(y,yprime,PAR)
      f( 1) = y( 1) - yold( 1) - dt*yprime( 1)
      f( 2) = y( 2) - yold( 2) - dt*yprime( 2)

      return
      end
C ======================================================================
```

```
C #====================================================================
C                                  _______          __    ___
C                 |    |   |\  |   |    |        \    /  |      |
C                 |    |   | \ |   |    |         \  /   | --   |
C                 |____|   |  \|   |    |          \/    | __   |___
C

C
C    This subroutine rescales all slips directions and normals to unity
C #====================================================================
      subroutine unit_vector(m,n,slipn,slips)
      include 'vaba_param.inc'
      integer n, m, i, k, j
      real*8,dimension(n,m):: slipn, slips
      real*8 anorm_n, anorm_s, sum_n, sum_s

      sum_n = 0.0
      sum_s = 0.0

      do i = 1, n
         do k = 1, m
            sum_n = sum_n + (abs(slipn(i,k)))**2.
            sum_s = sum_s + (abs(slips(i,k)))**2.
         end do
         anorm_n = sqrt(sum_n)
         anorm_s = sqrt(sum_s)
         do j = 1, m
            slipn(i,j) = slipn(i,j)/anorm_n
            slips(i,j) = slips(i,j)/anorm_s
         end do
         sum_n = 0.0
         sum_s = 0.0
      end do
      return
      end
C ====================================================================
```

```
C #====================================================================
C
C                  ____   ____  _____      ______   ______
C                 |    | |    |   |       /\     |    |
C                 |___/  |    |   |      /  \    |    |____
C                 |   \  |    |   |     / -- \   |    |
C                 |    | |____|   |    /      \  |    |______
C
C #====================================================================
      subroutine rotate(stressNew,Rnew,sigGLB1,sigGLB2,sigGLB3,
     1                                  sigGLB4,sigGLB5,sigGLB6)
      include 'vaba_param.inc'
C Real Scalar
      real*8 stressNew(6), Rnew(3,3),sigGLB1,sigGLB2,sigGLB3
      real*8                         sigGLB4,sigGLB5,sigGLB6




         stressNew(1) =
     1         Rnew(1,1)**2.d0*sigGLB1+2.d0*Rnew(1,1)*Rnew(1,2)*sigGLB4+
     2         2.d0*Rnew(1,1)*Rnew(1,3)*sigGLB6+Rnew(1,2)**2.d0*sigGLB2+
     3         2.d0*Rnew(1,2)*Rnew(1,3)*sigGLB5+Rnew(1,3)**2.d0*sigGLB3
         stressNew(2) =
     1         Rnew(2,1)**2.d0*sigGLB1+2.d0*Rnew(2,1)*Rnew(2,2)*sigGLB4+
     2         2.d0*Rnew(2,1)*Rnew(2,3)*sigGLB6+Rnew(2,2)**2.d0*sigGLB2+
     3         2.d0*Rnew(2,2)*Rnew(2,3)*sigGLB5+Rnew(2,3)**2.d0*sigGLB3
         stressNew(3) =
     1         Rnew(3,1)**2.d0*sigGLB1+2.*Rnew(3,1)*Rnew(3,2)*sigGLB4+
     2         2.d0*Rnew(3,1)*Rnew(3,3)*sigGLB6+Rnew(3,2)**2.d0*sigGLB2+
     3         2.d0*Rnew(3,2)*Rnew(3,3)*sigGLB5+Rnew(3,3)**2.d0*sigGLB3
         stressNew(4) =
     1         Rnew(1,1)*Rnew(2,1)*sigGLB1+Rnew(1,1)*Rnew(2,2)*sigGLB4+
     2         Rnew(1,1)*Rnew(2,3)*sigGLB6+Rnew(1,2)*Rnew(2,1)*sigGLB4+
     3         Rnew(1,2)*Rnew(2,2)*sigGLB2+Rnew(1,2)*Rnew(2,3)*sigGLB5+
     4         Rnew(1,3)*Rnew(2,1)*sigGLB6+Rnew(1,3)*Rnew(2,2)*sigGLB5+
     5         Rnew(1,3)*Rnew(2,3)*sigGLB3
          stressNew(5) =
     1         Rnew(2,1)*Rnew(3,1)*sigGLB1+Rnew(2,1)*Rnew(3,2)*sigGLB4+
     2         Rnew(2,1)*Rnew(3,3)*sigGLB6+Rnew(2,2)*Rnew(3,1)*sigGLB4+
     3         Rnew(2,2)*Rnew(3,2)*sigGLB2+Rnew(2,2)*Rnew(3,3)*sigGLB5+
     4         Rnew(2,3)*Rnew(3,1)*sigGLB6+Rnew(2,3)*Rnew(3,2)*sigGLB5+
     5         Rnew(2,3)*Rnew(3,3)*sigGLB3
          stressNew(6) =
     1         Rnew(1,1)*Rnew(3,1)*sigGLB1+Rnew(1,1)*Rnew(3,2)*sigGLB4+
     2         Rnew(1,1)*Rnew(3,3)*sigGLB6+Rnew(1,2)*Rnew(3,1)*sigGLB4+
     3         Rnew(1,2)*Rnew(3,2)*sigGLB2+Rnew(1,2)*Rnew(3,3)*sigGLB5+
     4         Rnew(1,3)*Rnew(3,1)*sigGLB6+Rnew(1,3)*Rnew(3,2)*sigGLB5+
     5         Rnew(1,3)*Rnew(3,3)*sigGLB3
      return
      end
C ====================================================================
```

```
C #======================================================================
C
C                  _____    ____ _______  ____  ____
C                 |        |        |    |     |    | |  |\   |
C                 |   --   |---     |    |___  |___ | |  | \  |
C                 |    |   |        |        | |     |  |  \ |
C                 |____|   |____    |    ____| |     |  |   \|
C
C           This subroutine computes the spin and deformation rate
C #======================================================================

      subroutine getSpinDij(dt,fNew,fOld,Spin,Dij)
      include 'vaba_param.inc'
C Real Scalar
      real*8 F11DD,F22DD,F33DD,F12DD,F23DD,F31DD,F21DD,F32DD,F13DD
      real*8 F11DI,F22DI,F33DI,F12DI,F23DI,F31DI,F21DI,F32DI,F13DI
      real*8 F11DV,F22DV,F33DV,F12DV,F23DV,F31DV,F21DV,F32DV,F13DV
      real*8 VG11, VG22, VG33, VG12, VG23, VG31, VG21, VG32, VG13
      real*8 fdet, dt
C Vectorial/ Matrix Definitions
      real*8 fNew(9),fOld(9),Spin(3),Dij(6)


         Spin(1) = 0.d0
        Spin(2) = 0.d0
        Spin(3) = 0.d0
        Dij(1)  = 0.d0
        Dij(2)  = 0.d0
        Dij(3)  = 0.d0
        Dij(4)  = 0.d0
        Dij(5)  = 0.d0
        Dij(6)  = 0.d0
C F_ij and F_dot_ij:
        f11dd = fNew( 1)
        f22dd = fNew( 2)
        f33dd = fNew( 3)
        f12dd = fNew( 4)
       f23dd = fNew( 5)
       f31dd = fNew( 6)
       f21dd = fNew( 7)
       f32dd = fNew( 8)
       f13dd = fNew( 9)
      f11dv = ( fNew( 1)  - fOld( 1) )/dt
      f22dv = ( fNew( 2)  - fOld( 2) )/dt
      f33dv = ( fNew( 3)  - fOld( 3) )/dt
      f12dv = ( fNew( 4)  - fOld( 4) )/dt
       f23dv = ( fNew( 5)  - fOld( 5) )/dt
       f31dv = ( fNew( 6)  - fOld( 6) )/dt
       f21dv = ( fNew( 7)  - fOld( 7) )/dt
       f32dv = ( fNew( 8)  - fOld( 8) )/dt
       f13dv = ( fNew( 9)  - fOld( 9) )/dt

       fdet  = (f11dd*f22dd*f33dd - f11dd*f23dd*f32dd-
     1           f21dd*f12dd*f33dd + f21dd*f13dd*f32dd+
     2           f31dd*f12dd*f23dd - f31dd*f13dd*f22dd)
```

```
C inv(F_ij)
               f11di =  (  f22dd*f33dd - f23dd*f32dd )/ fdet
               f22di =  (  f11dd*f33dd - f13dd*f31dd )/ fdet
               f33di =  (  f11dd*f22dd - f12dd*f21dd )/ fdet
               f12di =  ( -f12dd*f33dd + f13dd*f32dd )/ fdet
               f23di =  ( -f11dd*f23dd + f13dd*f21dd )/ fdet
               f31di =  (  f21dd*f32dd - f22dd*f31dd )/ fdet
               f21di =  ( -f21dd*f33dd + f23dd*f31dd )/ fdet
               f32di =  ( -f11dd*f32dd + f12dd*f31dd )/ fdet
               f13di =  (  f12dd*f23dd - f13dd*f22dd )/ fdet
C L_ij = F_dot_ik * inv(F_kj)
               vg11 = f11dv*f11di+f12dv*f21di+f13dv*f31di
               vg22 = f21dv*f12di+f22dv*f22di+f23dv*f32di
               vg33 = f31dv*f13di+f32dv*f23di+f33dv*f33di
               vg12 = f11dv*f12di+f12dv*f22di+f13dv*f32di
               vg23 = f21dv*f13di+f22dv*f23di+f23dv*f33di
               vg31 = f31dv*f11di+f32dv*f21di+f33dv*f31di
               vg21 = f21dv*f11di+f22dv*f21di+f23dv*f31di
               vg32 = f31dv*f12di+f32dv*f22di+f33dv*f32di
               vg13 = f11dv*f13di+f12dv*f23di+f13dv*f33di
C Wij = asym(L_ij)
               Spin(1) = 0.5d0*(vg21 - vg12)
               Spin(2) = 0.5d0*(vg32 - vg23)
               Spin(3) = 0.5d0*(vg13 - vg31)


C Dij = sym(L_ij)
              Dij(1) = vg11
              Dij(2) = vg22
              Dij(3) = vg33
              Dij(4) = 0.5d0*(vg21 + vg12)
              Dij(5) = 0.5d0*(vg32 + vg23)
              Dij(6) = 0.5d0*(vg13 + vg31)
         return
         end
C ======================================================================
```

```
C #====================================================================
C
C                  _____    ____  _____   ____   ____  _____
C                 |        |        |    |    | |    |   |
C                 |   --   |---     |    |___ | |    |   |
C                 |    |   |        |    |   \  |    |   |
C                 |____|   |____    |    |    | |____|   |
C
C             This subroutine computes the Element Rotation Matrix
C #====================================================================
      subroutine getRot(fNew,strtchNew,R)
        include 'vaba_param.inc'
         real*8 fnew(9), strtchNew(9), R(3,3)
         real*8 detFU, U11,U12,U22,U23,U31,U33
         real*8 F11,F12,F13,F21,F22,F23,F31,F32,F33
C     Define Deformation gradient components
         F11 = fNew(1)
         F22 = fNew(2)
         F33 = fNew(3)
         F12 = fNew(4)
          F23 = fNew(5)
          F31 = fNew(6)
          F21 = fNew(7)
          F32 = fNew(8)
          F13 = fNew(9)
C     Define Stretch Matrix components
         U11 = strtchNew(1)
         U22 = strtchNew(2)
         U33 = strtchNew(3)
         U12 = strtchNew(4)
          U23 = strtchNew(5)
          U31 = strtchNew(6)
C     Compute rotation matrix using R = F*inv(U){Polar decomposition of F}
C     Actually, store R_transpose
          detFU =      U11*U22*U33-U11*U23**2.d0-U12**2.d0*U33+
     1           2.d0*U31*U12*U23-U22*U31**2.d0
          R(1,1) = ( F11*U22*U33-F11*U23**2.d0-F12*U12*U33+
     1              F12*U23*U31+F13*U12*U23-F13*U22*U31  )/detFU
         R(2,1) = (-F11*U12*U33+F11*U23*U31+F12*U11*U33-
     1              F12*U31**2.d0-F13*U11*U23+F13*U12*U31)/detFU
         R(3,1) = ( F11*U12*U23-F11*U22*U31-F12*U11*U23+
     1            F12*U12*U31+F13*U11*U22-F13*U12**2.d0)/detFU
         R(1,2) = ( F21*U22*U33-F21*U23**2.d0-F22*U12*U33+
     1            F22*U23*U31+F23*U12*U23-F23*U22*U31  )/detFU
         R(2,2) = (-F21*U12*U33+F21*U23*U31+F22*U11*U33-
     1            F22*U31**2.d0-F23*U11*U23+F23*U12*U31)/detFU
         R(3,2) = ( F21*U12*U23-F21*U22*U31-F22*U11*U23+
     1            F22*U12*U31+F23*U11*U22-F23*U12**2.d0)/detFU
         R(1,3) = ( F31*U22*U33-F31*U23**2.d0-F32*U12*U33+
     1            F32*U23*U31+F33*U12*U23-F33*U22*U31  )/detFU
         R(2,3) = (-F31*U12*U33+F31*U23*U31+F32*U11*U33-
     1            F32*U31**2.d0-F33*U11*U23+F33*U12*U31)/detFU
         R(3,3) = ( F31*U12*U23-F31*U22*U31-F32*U11*U23+
     1            F32*U12*U31+F33*U11*U22-F33*U12**2.d0)/detFU
         return
```

```
          end
C ======================================================================
```