CS294-73 Final Project: Multiphase Material Solid Mechanics

Santiago Miret, Max Poschmann, Ji Wei Yoon

December 16, 2015

1 Problem Definition

The project proposed for this class involves the calculation of the mechanical behavior of a multiphase material. The main equation being solved will be the balance of linear momentum in a static setting. The constitutive model for the mechanical behavior is based on the a paper by T.I. Zohdi and A.J. Szeri on a multiphase model of kidney stone fatigue[1]. The detailed mathematics of the project will be outlined further below, yet it is important to note that the proposed problem will be solved on an unstructured grid in a 2-dimensional setting.

The differential equation being solved for this study is the balance of linear momentum

$$\nabla_x \boldsymbol{\sigma} + \mathbf{f} = \rho \frac{d^2\mathbf{u}}{dt^2}$$

in a time-independent setting, meaning $\frac{d^2\mathbf{u}}{dt^2} = 0$, resulting in

$$\nabla_x \boldsymbol{\sigma} + \mathbf{f} = 0$$

where σ represents the Cauchy stress, $\mathbf{f}$ represents body forces. When considering only infinitesimal deformations, we can approximate $\frac{d()}{dt} \approx \frac{\partial()}{\partial t}\,|X$ and $\nabla_{\mathbf{x}} = \nabla_{\mathbf{X}}$ where $\mathbf{x}$ is the current configuration and $\mathbf{X}$ is the reference configuration of the system. Now using a multiphase, elastic, isotropic constitutive law we can write

$$\boldsymbol{\sigma} = \mathbb{E}^* : (\boldsymbol{\epsilon})$$

where $\mathbb{E}^*$ represents the effective elastic tensor of the material. The effective elastic tensor for a multiphase material can be calculated using property derived from variational principles[2]. In the case of two-phase materials, the following bounds can be written

- The bulk modulus κ, the lower and upper are respectively

$$\kappa^*_{(-)} = \kappa_1 + v_2\left(\frac{1}{\kappa_2 - \kappa_1} + \frac{3(1-v_2)}{3\kappa_1 + 4\mu_1}\right)^{-1}$$

$$\kappa^*_{(+)} = \kappa_2 + (1-v_2)\left(\frac{1}{\kappa_1 - \kappa_2} + \frac{3v_2}{3\kappa_2 + 4\mu_2}\right)^{-1}$$

- The shear modulus μ

$$\mu^*_{(-)} = \mu_1 + v_2\left(\frac{1}{\mu_2 - \mu_1} + \frac{6v_2(\kappa_2 + 2\mu_2)}{5\mu_1(3\kappa_1 + 4\mu_1)}\right)^{-1}$$

$$\mu^*_{(+)} = \mu_2 + (1-v_2)\left(\frac{1}{\mu_1 - \mu_2} + \frac{6v_2(\kappa_2 + 2\mu_2)}{5\mu_2(3\kappa_2 + 4\mu_2)}\right)^{-1}$$

[1] Zohdi, T. I. and Szeri, A. J. Fatigue of Kidney Stones with Heterogeneous Microstructure. Wiley InterScience. 2005.

[2] Hashin, Z. Analysis of Composite Materials - A Survey. Journal of Applied Mathemetics. Vol. 50/481. September 1983.

The constitutive law can be written more compactly using Voigt notation[3]

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{Bmatrix} = \begin{bmatrix} \mathbb{E}_{11} & \mathbb{E}_{12} & \mathbb{E}_{13} & \mathbb{E}_{14} & \mathbb{E}_{15} & \mathbb{E}_{16} \\ \mathbb{E}_{21} & \mathbb{E}_{22} & \mathbb{E}_{23} & \mathbb{E}_{24} & \mathbb{E}_{25} & \mathbb{E}_{26} \\ \mathbb{E}_{31} & \mathbb{E}_{32} & \mathbb{E}_{33} & \mathbb{E}_{34} & \mathbb{E}_{35} & \mathbb{E}_{36} \\ \mathbb{E}_{41} & \mathbb{E}_{42} & \mathbb{E}_{43} & \mathbb{E}_{44} & \mathbb{E}_{45} & \mathbb{E}_{46} \\ \mathbb{E}_{51} & \mathbb{E}_{52} & \mathbb{E}_{53} & \mathbb{E}_{54} & \mathbb{E}_{55} & \mathbb{E}_{56} \\ \mathbb{E}_{61} & \mathbb{E}_{62} & \mathbb{E}_{63} & \mathbb{E}_{64} & \mathbb{E}_{65} & \mathbb{E}_{66} \end{bmatrix} \begin{Bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ 2\epsilon_{12} \\ 2\epsilon_{23} \\ 2\epsilon_{31} \end{Bmatrix}$$

which can be rewritten in terms of the effective bulk modulus $\kappa*$ and the effective shear modulus $\mu*$

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{Bmatrix} = \begin{bmatrix} \kappa^* + \frac{4}{3}\mu^* & \kappa^* - \frac{2}{3}\mu^* & \kappa^* - \frac{2}{3}\mu^* & 0 & 0 & 0 \\ \kappa^* - \frac{2}{3}\mu^* & \kappa^* + \frac{4}{3}\mu^* & \kappa^* - \frac{2}{3}\mu^* & 0 & 0 & 0 \\ \kappa^* - \frac{2}{3}\mu^* & \kappa^* - \frac{2}{3}\mu^* & \kappa^* + \frac{4}{3}\mu^* & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu^* & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu^* & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu^* \end{bmatrix} \begin{Bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ 2\epsilon_{12} \\ 2\epsilon_{23} \\ 2\epsilon_{31} \end{Bmatrix}$$

Furthermore, the strain tensor $\{\epsilon\}$ can be written in terms of spatial derivatives

$$\{\epsilon\} = \begin{Bmatrix} \frac{\partial u_1}{\partial x_1} \\ \frac{\partial u_2}{\partial x_2} \\ \frac{\partial u_3}{\partial x_3} \\ \frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \\ \frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2} \\ \frac{\partial u_1}{\partial x_3} + \frac{\partial u_3}{\partial x_1} \end{Bmatrix}$$

which can be expressed in term of the finite element discretization

$$\frac{\partial u_i}{\partial x_j} = \sum_{A=1}^{N} a_i^A \frac{\partial \phi^A}{\partial x_j}$$

which makes it useful to define the following matrix $[T]$

$$[T] = \begin{Bmatrix} \frac{\partial}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial}{\partial x_3} \\ \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} & 0 \\ 0 & \frac{\partial}{\partial x_3} & \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_3} & 0 & \frac{\partial}{\partial x_1} \end{Bmatrix}$$

which allows the strain tensor to rewritten as

$$\{\epsilon\} = [T] \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix}$$

2 Discretization Approach

Weak Form of Time-Dependent Balance of Linear Momentum

The first step in the finite element method[4] is to derive the weak form of the equation by integrating the equation over the domain of the problem and multiplying the equation by a test function v:

$$\nabla_X \boldsymbol{\sigma} + \mathbf{f} = 0$$

[3]Zohdi, T. I. and Wriggers, P. Introduction to computational micromechanics. Springer-Verlag. 2008.
[4]Zohdi, T.I. A Finite Element Primer For Beginners. Springer-Verlag. 2014.

$$\int_\Omega v \cdot (\nabla_X \boldsymbol{\sigma} + \mathbf{f})\, d\Omega = 0 \quad \forall v \in H_{1(\Omega)}$$

This equation can be modified using the product rule of differentiation:

$$\int_\Omega \nabla \cdot (\boldsymbol{\sigma} : v)\, d\Omega = \int_\Omega v \cdot (\nabla \boldsymbol{\sigma})\, d\Omega + \int_\Omega \nabla v : \boldsymbol{\sigma}\, d\Omega \quad \forall v \in H_{1(\Omega)}$$

which leads to the following equation

$$\int_\Omega \nabla_X \cdot (\boldsymbol{\sigma} : v)\, d\Omega - \int_\Omega \nabla_X v : \boldsymbol{\sigma}\, d\Omega + \int_\Omega v \cdot \mathbf{f}\, d\Omega = 0 \quad \forall v \in H_{1(\Omega)}$$

Next the divergence theorem is applied

$$\int_\Omega \nabla_X \cdot (\boldsymbol{\sigma} : v)\, d\Omega = \int_{\partial\Omega} \boldsymbol{\sigma} \cdot \boldsymbol{n} \cdot v\, dS$$

yielding

$$\int_\Omega \nabla_X v : \boldsymbol{\sigma}\, d\Omega = \int_{\partial\Omega} \boldsymbol{\sigma} \cdot \boldsymbol{n} \cdot v\, dS + \int_\Omega v \cdot \mathbf{f}\, d\Omega \quad \forall v \in H_{1(\Omega)}$$

which applying the boundary condition becomes

$$\int_{\partial\Omega} \boldsymbol{\sigma} \cdot \boldsymbol{n} \cdot v\, dS = \int_{\Gamma_t} \boldsymbol{t} \cdot v\, dS \qquad \forall v \; \textit{such that} \; v = 0 \; \textit{on} \; \Gamma_d$$

Now we can apply the constitutive law for the Cauchy stress

$$\boldsymbol{\sigma} = \mathbb{E}^* : (\boldsymbol{\epsilon})$$

to obtain the equation

$$\int_\Omega \nabla_X v : (\mathcal{D}\, \mathbb{E}_{\mathbf{0}} : (\boldsymbol{\epsilon} - \boldsymbol{\epsilon_\theta} - \boldsymbol{\epsilon_p}))\, d\Omega = \int_{\partial\Omega} \boldsymbol{\sigma} \cdot \boldsymbol{n} \cdot v\, dS + \int_\Omega v \cdot \mathbf{f}\, d\Omega$$

which can be broken into the different strain integrals

$$\int_\Omega \nabla_X v : (\mathcal{D}\, \mathbb{E}^* : \boldsymbol{\epsilon})\, d\Omega = \int_{\partial\Omega} \boldsymbol{\sigma} \cdot \boldsymbol{n} \cdot v\, dS + \int_\Omega v \cdot \mathbf{f}\, d\Omega$$

Now the finite element discretization can be applied

$$u(x,y,z) = \sum_{A=1}^{N} a_A \phi_A(x,y,z) \qquad v(x,y,z) = \sum_{A=1}^{N} b_A \phi_A(x,y,z)$$

or

$$u_i(x,y,z) = \sum_{A=1}^{N} a_i^A \phi^A(x,y,z) \qquad v_i(x,y,z) = \sum_{A=1}^{N} b_i^A \phi^A(x,y,z) \quad i = 1,2,3$$

Now we define the matrix 3x24 $[\phi]$ as follows

$$[\phi] = \begin{bmatrix} \phi_1 & \phi_2 & \phi_3 & \phi_4 & \phi_5 & \phi_6 & \phi_7 & \phi_8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

as the first block of eight columns

$$[\phi] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \phi_1 & \phi_2 & \phi_3 & \phi_4 & \phi_5 & \phi_6 & \phi_7 & \phi_8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

as the second block of eight columns

$$[\phi] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \phi_1 & \phi_2 & \phi_3 & \phi_4 & \phi_5 & \phi_6 & \phi_7 & \phi_8 \end{bmatrix}$$

as the third block of eight columns. All three blocks are put together to form the full matrix. This is written implicitly here due to space limitations of the text processor. A similar approach will be used to express other large matrices. Now if we define the solution vector $\{a\}$ to be a 24x1 vector of the following form

$$\{a\} = \begin{Bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \\ a_1^2 \\ a_2^2 \\ a_3^2 \\ \vdots \\ a_3^8 \end{Bmatrix}$$

we can define the solution vector as

$$u(x) = [\phi]\{a\}$$

and the strain tensor as

$$\{\epsilon\} = [T][\phi]\{a\}$$

where $[T][\phi]$ is

$$[T][\phi] = \begin{bmatrix} \frac{\partial \phi^1}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial \phi^1}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial \phi^1}{\partial x_3} \\ \frac{\partial \phi^1}{\partial x_2} & \frac{\partial \phi^1}{\partial x_1} & 0 \\ 0 & \frac{\partial \phi^1}{\partial x_3} & \frac{\partial \phi^1}{\partial x_2} \\ \frac{\partial \phi^1}{\partial x_1} & 0 & \frac{\partial \phi^1}{\partial x_3} \end{bmatrix}$$

for the first block

$$[T][\phi] = \begin{bmatrix} \frac{\partial \phi^2}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial \phi^2}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial \phi^2}{\partial x_3} \\ \frac{\partial \phi^2}{\partial x_2} & \frac{\partial \phi^2}{\partial x_1} & 0 \\ 0 & \frac{\partial \phi^2}{\partial x_3} & \frac{\partial \phi^2}{\partial x_2} \\ \frac{\partial \phi^2}{\partial x_1} & 0 & \frac{\partial \phi^2}{\partial x_3} \end{bmatrix}$$

for the second block

$$[T][\phi] = \begin{bmatrix} \frac{\partial \phi^3}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial \phi^3}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial \phi^3}{\partial x_3} \\ \frac{\partial \phi^3}{\partial x_2} & \frac{\partial \phi^3}{\partial x_1} & 0 \\ 0 & \frac{\partial \phi^3}{\partial x_3} & \frac{\partial \phi^3}{\partial x_2} \\ \frac{\partial \phi^3}{\partial x_1} & 0 & \frac{\partial \phi^3}{\partial x_3} \end{bmatrix}$$

for the third block, all of which assembled yield the full $[T][\phi]$.

Using these definitions, we can rewrite the weak form equation in matrix form

$$\underbrace{\int_\Omega ([T]\{v\})^T[\mathbb{E}^*]([T]\{u\})d\Omega}_{[S]_1} = \underbrace{\underbrace{\int_\Omega \{v\}^T \cdot \mathbf{f}\, d\Omega}_{\{R\}_f} + \underbrace{\int_{\Gamma_t} v\{\bar{t}\} \cdot \boldsymbol{t}\, d\Gamma}_{\{R\}_t}}_{\{R\}}$$

Using the same discretization scheme as above,

$$u(x,y,z) = \sum_{A=1}^{N} a_A\phi_A(x,y,z) \qquad v(x,y,z) = \sum_{A=1}^{N} b_A\phi_A(x,y,z) \qquad \dot{u}(x,y,z) = \sum_{A=1}^{N} \dot{a}_A\phi_A(x,y,z)$$

leads to the following discretized weak form equation. For the left-hand-side strain terms:

$$\int_\Omega ([T]\{\sum_{A=1}^{N} b_A\phi_A\})^T[\,\mathbb{E}^*]([T]\{\sum_{A=1}^{N} a_A\phi_A\})d\Omega$$

and the right-hand-side

$$\int_\Omega \sum_{A=1}^{N} b_A\phi_A \cdot \mathbf{f}\, d\Omega + \int_{\Gamma_t} \sum_{A=1}^{N} b_A\phi_A \cdot \boldsymbol{t}\, d\Gamma$$

The b_A's are arbitrary and can be canceled out of the equation leading to the following stiffness matrix terms on the left-hand side

$$\sum_{A=1}^{N}\sum_{B=1}^{N} \underbrace{\int_\Omega ([T]\{\phi_A\})^T[\mathbb{E}^*]([T]\{\phi_B\})d\Omega}_{[S_1]} \underbrace{a_B}_{\{a\}}$$

and the following right-hand side

$$\underbrace{\sum_{A=1}^{N} \underbrace{\int_\Omega \phi_A \cdot \mathbf{f}\, d\Omega}_{\{R\}_f} + \underbrace{\int_{\Gamma_t} \phi_A \cdot \boldsymbol{t}\, d\Gamma}_{\{R\}_t}}_{\{R\}}$$

The weak then becomes the following matrix system of equations

$$([S_1])\{a(t)\} = \{R\}$$

Evaluation of the Weak Form

The mapping from the real space to the master element in three dimensions is defined as follows

$$x_1 = \sum_{i=1}^{8} X_{1i}\hat{\phi}_i = Map_{x_1}(\zeta_1,\zeta_2,\zeta_3)$$

$$x_2 = \sum_{i=1}^{8} X_{2i}\hat{\phi}_i = Map_{x_2}(\zeta_1,\zeta_2,\zeta_3)$$

$$x_3 = \sum_{i=1}^{8} X_{3i}\hat{\phi}_i = Map_{x_3}(\zeta_1,\zeta_2,\zeta_3)$$

In 3-D the shape function of the master element take on a different form. First, there are 8 distinct linear shape functions that define the isoparametric space

$$\hat{\phi}_1(\zeta_1,\zeta_2,\zeta_3) = \frac{1}{8}(1-\zeta_1)(1-\zeta_2)(1-\zeta_3) \qquad for\, \zeta_1,\zeta_2,\zeta_3 \in [-1,1]$$

$$\hat{\phi}_2(\zeta_1,\zeta_2,\zeta_3) = \frac{1}{8}(1+\zeta_1)(1-\zeta_2)(1-\zeta_3) \qquad for\, \zeta_1,\zeta_2,\zeta_3 \in [-1,1]$$

$$\hat{\phi}_3(\zeta_1,\zeta_2,\zeta_3) = \frac{1}{8}(1+\zeta_1)(1+\zeta_2)(1-\zeta_3) \qquad for\, \zeta_1,\zeta_2,\zeta_3 \in [-1,1]$$

$$\hat{\phi}_4(\zeta_1,\zeta_2,\zeta_3) = \frac{1}{8}(1-\zeta_1)(1+\zeta)(1-\zeta_3) \qquad for\, \zeta_1,\zeta_2,\zeta_3 \in [-1,1]$$

$$\hat{\phi}_5(\zeta_1,\zeta_2,\zeta_3) = \frac{1}{8}(1-\zeta_1)(1-\zeta_2)(1+\zeta_3) \qquad for\, \zeta_1,\zeta_2,\zeta_3 \in [-1,1]$$

$$\hat{\phi}_6(\zeta_1,\zeta_2,\zeta_3) = \frac{1}{8}(1+\zeta_1)(1-\zeta_2)(1+\zeta_3) \qquad for\, \zeta_1,\zeta_2,\zeta_3 \in [-1,1]$$

$$\hat{\phi}_7(\zeta_1,\zeta_2,\zeta_3) = \frac{1}{8}(1+\zeta_1)(1+\zeta_2)(1+\zeta_3) \qquad for\, \zeta_1,\zeta_2,\zeta_3 \in [-1,1]$$

$$\hat{\phi}_8(\zeta_1,\zeta_2,\zeta_3) = \frac{1}{8}(1-\zeta_1)(1+\zeta_2)(1+\zeta_3) \qquad for\, \zeta_1,\zeta_2,\zeta_3 \in [-1,1]$$

Now we define the the $[F]$ matrix

$$[F] = \begin{bmatrix} \frac{\partial x_1}{\partial \zeta_1} & \frac{\partial x_1}{\partial \zeta_2} & \frac{\partial x_1}{\partial \zeta_3} \\ \frac{\partial x_2}{\partial \zeta_1} & \frac{\partial x_2}{\partial \zeta_2} & \frac{\partial x_2}{\partial \zeta_3} \\ \frac{\partial x_3}{\partial \zeta_1} & \frac{\partial x_3}{\partial \zeta_2} & \frac{\partial x_3}{\partial \zeta_3} \end{bmatrix}$$

The Jacobian of the transformation to the isoparametric space is

$$J = det\,|F| = \frac{\partial x_1}{\partial \zeta_1}(\frac{\partial x_2}{\partial \zeta_2}\frac{\partial x_3}{\partial \zeta_3} - \frac{\partial x_3}{\partial \zeta_2}\frac{\partial x_2}{\partial \zeta_3}) - \frac{\partial x_1}{\partial \zeta_2}(\frac{\partial x_2}{\partial \zeta_1}\frac{\partial x_3}{\partial \zeta_3} - \frac{\partial x_2}{\partial \zeta_1}\frac{\partial x_3}{\partial \zeta_3}) + \frac{\partial x_1}{\partial \zeta_3}(\frac{\partial x_2}{\partial \zeta_1}\frac{\partial x_2}{\partial \zeta_2} - \frac{\partial x_3}{\partial \zeta_1}\frac{\partial x_2}{\partial \zeta_2})$$

The differential relationships for the isoparametric mapping are

$$\frac{\partial}{\partial \zeta_1} = \frac{\partial}{\partial x_1}\frac{\partial x_1}{\partial \zeta_1} + \frac{\partial}{\partial x_2}\frac{\partial x_2}{\partial \zeta_1} + \frac{\partial}{\partial x_3}\frac{\partial x_3}{\partial \zeta_1}$$

$$\frac{\partial}{\partial \zeta_2} = \frac{\partial}{\partial x_1}\frac{\partial x_1}{\partial \zeta_2} + \frac{\partial}{\partial x_2}\frac{\partial x_2}{\partial \zeta_2} + \frac{\partial}{\partial x_3}\frac{\partial x_3}{\partial \zeta_2}$$

$$\frac{\partial}{\partial \zeta_3} = \frac{\partial}{\partial x_1}\frac{\partial x_1}{\partial \zeta_3} + \frac{\partial}{\partial x_2}\frac{\partial x_2}{\partial \zeta_3} + \frac{\partial}{\partial x_3}\frac{\partial x_3}{\partial \zeta_3}$$

and the inverse differential relationships are

$$\frac{\partial}{\partial x_1} = \frac{\partial}{\partial \zeta_1}\frac{\partial \zeta_1}{\partial x_1} + \frac{\partial}{\partial \zeta_2}\frac{\partial \zeta_2}{\partial x_1} + \frac{\partial}{\partial \zeta_3}\frac{\partial \zeta_3}{\partial x_1}$$

$$\frac{\partial}{\partial x_2} = \frac{\partial}{\partial \zeta_1}\frac{\partial \zeta_1}{\partial x_2} + \frac{\partial}{\partial \zeta_2}\frac{\partial \zeta_2}{\partial x_2} + \frac{\partial}{\partial \zeta_3}\frac{\partial \zeta_3}{\partial x_2}$$

$$\frac{\partial}{\partial x_3} = \frac{\partial}{\partial \zeta_1}\frac{\partial \zeta_1}{\partial x_3} + \frac{\partial}{\partial \zeta_2}\frac{\partial \zeta_2}{\partial x_3} + \frac{\partial}{\partial \zeta_3}\frac{\partial \zeta_3}{\partial x_3}$$

Therefore

$$\begin{Bmatrix} dx_1 \\ dx_2 \\ dx_3 \end{Bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial x_1}{\partial \zeta_1} & \frac{\partial x_1}{\partial \zeta_2} & \frac{\partial x_1}{\partial \zeta_3} \\ \frac{\partial x_2}{\partial \zeta_1} & \frac{\partial x_2}{\partial \zeta_2} & \frac{\partial x_2}{\partial \zeta_3} \\ \frac{\partial x_3}{\partial \zeta_1} & \frac{\partial x_3}{\partial \zeta_2} & \frac{\partial x_3}{\partial \zeta_3} \end{bmatrix}}_{F} \begin{Bmatrix} d\zeta_1 \\ d\zeta_2 \\ d\zeta_3 \end{Bmatrix}$$

and

$$\begin{Bmatrix} d\zeta_1 \\ d\zeta_2 \\ d\zeta_3 \end{Bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial \zeta_1}{\partial x_1} & \frac{\partial \zeta_1}{\partial x_2} & \frac{\partial \zeta_1}{\partial x_3} \\ \frac{\partial \zeta_2}{\partial x_1} & \frac{\partial \zeta_2}{\partial x_2} & \frac{\partial \zeta_2}{\partial x_3} \\ \frac{\partial \zeta_3}{\partial x_1} & \frac{\partial \zeta_3}{\partial x_2} & \frac{\partial \zeta_3}{\partial x_3} \end{bmatrix}}_{F^{-1}} \begin{Bmatrix} dx_1 \\ dx_2 \\ dx_3 \end{Bmatrix}$$

Now the matrices in the weak form need to be expressed in terms of the master domain coordinates $\zeta_1, \zeta_2, \zeta_3$

$$[T(\phi(x_1, x_2, x_3)] = [\hat{T}(\hat{\phi}(Map_{x_1}(\zeta_1, \zeta_2, \zeta_3), Map_{x_2}(\zeta_1, \zeta_2\zeta_3), Map_{x_3}(\zeta_1, \zeta_2, \zeta_3)))]$$

Now we define $[\hat{T}]$ to be

$$[\hat{T}] = \begin{bmatrix} \frac{\partial}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} \\ 0 \\ 0 \\ \frac{\partial}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} \\ 0 \\ \frac{\partial}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} \end{bmatrix}$$

for the first column

$$[\hat{T}] = \begin{bmatrix} 0 \\ \frac{\partial}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} \\ 0 \\ \frac{\partial}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} \\ \frac{\partial}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} \\ 0 \end{bmatrix}$$

for the second column

$$[\hat{T}] = \begin{bmatrix} 0 \\ 0 \\ \frac{\partial}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} \\ 0 \\ \frac{\partial}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} \\ \frac{\partial}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} \end{bmatrix}$$

The elemental shape function matrix $[\hat{\phi}]$ has the following form

$$[\hat{\phi}] = \begin{bmatrix} \hat{\phi}_1 & \hat{\phi}_2 & \hat{\phi}_3 & \hat{\phi}_4 & \hat{\phi}_5 & \hat{\phi}_6 & \hat{\phi}_7 & \hat{\phi}_8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

for the first block of eight columns

$$[\hat{\phi}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hat{\phi}_1 & \hat{\phi}_2 & \hat{\phi}_3 & \hat{\phi}_4 & \hat{\phi}_5 & \hat{\phi}_6 & \hat{\phi}_7 & \hat{\phi}_8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

for the second block of eight columns

$$[\hat{\phi}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hat{\phi}_1 & \hat{\phi}_2 & \hat{\phi}_3 & \hat{\phi}_4 & \hat{\phi}_5 & \hat{\phi}_6 & \hat{\phi}_7 & \hat{\phi}_8 \end{bmatrix}$$

for the third block of eight columns, resulting in a 3x24 matrix. The product $[\hat{T}][\hat{\phi}]$ then becomes a 6x24 matrix

$$[\hat{T}][\hat{\phi}] = \begin{bmatrix} \frac{\partial\hat{\phi}_1}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial\hat{\phi}_1}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial\hat{\phi}_1}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} & \frac{\partial\hat{\phi}_2}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial\hat{\phi}_2}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial\hat{\phi}_2}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} & \cdots & \cdots & \cdots & \cdots & \cdots & \frac{\partial\hat{\phi}_8}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial\hat{\phi}_8}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial\hat{\phi}_8}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial\hat{\phi}_1}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial\hat{\phi}_1}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial\hat{\phi}_1}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} & \frac{\partial\hat{\phi}_2}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial\hat{\phi}_2}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial\hat{\phi}_2}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} & \cdots & \cdots & \cdots & \cdots & \cdots & \frac{\partial\hat{\phi}_8}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial\hat{\phi}_8}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial\hat{\phi}_8}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial\hat{\phi}_1}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial\hat{\phi}_1}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial\hat{\phi}_1}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} & \frac{\partial\hat{\phi}_2}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial\hat{\phi}_2}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial\hat{\phi}_2}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} & \cdots & \cdots & \cdots & \cdots & \cdots & \frac{\partial\hat{\phi}_8}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial\hat{\phi}_8}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial\hat{\phi}_8}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} \end{bmatrix}$$

for the first block of eight columns

$$[\hat{T}][\hat{\phi}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial\hat{\phi}_1}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial\hat{\phi}_1}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial\hat{\phi}_1}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} & \frac{\partial\hat{\phi}_2}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial\hat{\phi}_2}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial\hat{\phi}_2}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} & \dots & \dots & \dots & \dots & \dots & \frac{\partial\hat{\phi}_8}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial\hat{\phi}_8}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial\hat{\phi}_8}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial\hat{\phi}_1}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial\hat{\phi}_1}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial\hat{\phi}_1}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} & \frac{\partial\hat{\phi}_2}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial\hat{\phi}_2}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial\hat{\phi}_2}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} & \dots & \dots & \dots & \dots & \dots & \frac{\partial\hat{\phi}_8}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial\hat{\phi}_8}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial\hat{\phi}_8}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} \\ \frac{\partial\hat{\phi}_1}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial\hat{\phi}_1}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial\hat{\phi}_1}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} & \frac{\partial\hat{\phi}_2}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial\hat{\phi}_2}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial\hat{\phi}_2}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} & \dots & \dots & \dots & \dots & \dots & \frac{\partial\hat{\phi}_8}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial\hat{\phi}_8}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial\hat{\phi}_8}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

for the second block of eight columns

$$[\hat{T}][\hat{\phi}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial\hat{\phi}_1}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial\hat{\phi}_1}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial\hat{\phi}_1}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} & \frac{\partial\hat{\phi}_2}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial\hat{\phi}_2}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial\hat{\phi}_2}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} & \dots & \dots & \dots & \dots & \dots & \frac{\partial\hat{\phi}_8}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_3} + \frac{\partial\hat{\phi}_8}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_3} + \frac{\partial\hat{\phi}_8}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial\hat{\phi}_1}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial\hat{\phi}_1}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial\hat{\phi}_1}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} & \frac{\partial\hat{\phi}_2}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial\hat{\phi}_2}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial\hat{\phi}_2}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} & \dots & \dots & \dots & \dots & \dots & \frac{\partial\hat{\phi}_8}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_2} + \frac{\partial\hat{\phi}_8}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_2} + \frac{\partial\hat{\phi}_8}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_2} \\ \frac{\partial\hat{\phi}_1}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial\hat{\phi}_1}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial\hat{\phi}_1}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} & \frac{\partial\hat{\phi}_2}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial\hat{\phi}_2}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial\hat{\phi}_2}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} & \dots & \dots & \dots & \dots & \dots & \frac{\partial\hat{\phi}_8}{\partial\zeta_1}\frac{\partial\zeta_1}{\partial x_1} + \frac{\partial\hat{\phi}_8}{\partial\zeta_2}\frac{\partial\zeta_2}{\partial x_1} + \frac{\partial\hat{\phi}_8}{\partial\zeta_3}\frac{\partial\zeta_3}{\partial x_1} \end{bmatrix}$$

Applying Gaussian Quadrature

Now that all the matrices are defined, quadrature is applied to evaluate the expressions numerically. This leads to the following expressions

$$S^e_{ij} = \underbrace{\sum_{q=1}^{g}\sum_{r=1}^{g}\sum_{s=1}^{g} w_q w_r w_s ([\hat{T}]\{\hat{\phi}_i\})^T [\mathbb{E}^*] ([\hat{T}]\{\phi_j\}) \, |F|}_{standard\,[S]\,term}$$

for the stiffness matrix, where g is the number of Gauss Points and $|F_s|$ is the Jacobian of the surface integral transformation. Elements that are not subject to the Dirichlet boundary conditions will only have the standard contribution to the stiffness matrix and elements with Dirichlet boundary conditions will be implemented by modifying the corresponding stiffness matrix accordingly. The load vector then becomes

$$R^e_i = \underbrace{\sum_{q=1}^{g}\sum_{r=1}^{g}\sum_{s=1}^{g} w_q w_r w_s [\hat{\phi}_i]^T \{f\} \, |F|}_{standard\,\{R\}\,term}$$

$$+ \underbrace{\sum_{q=1}^{g}\sum_{r=1}^{g} w_q w_r [\hat{\phi}_i]^T \{t\} \, |F_s|}_{for\,\Gamma_t \cap \Omega_e \neq 0}$$

Elements that are not subject to the Dirichlet or the traction boundary conditions, i.e are not on the surface of Γ_t or Γ_d, are only subject to the standard contribution to the load vector. Elements that are subject to the traction boundary condition, i.e. are on the surface defined by Γ_t will have a contribution from the second sum. Elements that are subject to the Dirichlet boundary conditions, i.e. are on the surface defined by Γ_d will be modified to account for the Dirichlet boundary terms.

The transformation factor for the surface integral can be found using Nanson's formula

$$\mathbf{n} d\mathbf{\Gamma} = J F^{-T} \mathbf{N} \, d\Gamma_\zeta$$

which for an expression inside a surface integral yields

$$\mathbf{n} \cdot \mathbf{n} \, d\Gamma = J \sqrt{\mathbf{N} \cdot F^{-1} F^{-T} \cdot \mathbf{N}} d\Gamma_\zeta$$

as the transformation factor for surface integrals. The finite element approximation of the weak form is defined in the space of $H_1(\Omega)$.

Integrals for Balance of Linear Momentum

Given the procedure outlined, the balance of linear differential equation requires the following integrals:

Integral 1

$$[S_1] = \int_\Omega ([T]\{\phi_A\})^T [\mathbb{E}^*]([T]\{\phi_B\}) d\Omega = \sum_{e=1}^{N_e} \int_{\Omega_e} ([T]\{\phi_A\})^T [\mathbb{E}^*]([T]\{\phi_B\}) d\Omega_e$$

$$[S_1]^e_{AB} = \iiint_{-1}^{1} \underbrace{([\hat{T}]\{\hat{\phi}_i\})^T [\,\mathbb{E}_0]([\hat{T}]\{\phi_j\}) \cdot |F| \; d\zeta_1 \, d\zeta_2 \, d\zeta_3}_{f_1(\zeta_1, \zeta_2, \zeta_3)}$$

which can using Gaussian Quadrature becomes

$$[S_1]^e_{AB} = \sum_{I=1}^{NP_1} \sum_{J=1}^{NP_2} \sum_{K=1}^{NP_3} w_I w_J w_K f_1(\zeta_1, \zeta_2, \zeta_3)$$

If the material properties change with time or temperature, then this integral has to be computed at each time-step. If they remain constant, then it has to be computed only once.

Integral 2

$$\{R\}_f = \int_\Omega \phi_A \cdot \mathbf{f} \, d\Omega = \sum_{e=1}^{N_e} \int_{\Omega_e} \phi_A \cdot \mathbf{f} \, d\Omega_e$$

$$\{R_f\}^e_A = \iiint_{-1}^{1} \underbrace{(\{\hat{\phi}_i\}^T \mathbf{f}) \cdot |F| \; d\zeta_1 \, d\zeta_2 \, d\zeta_3}_{f_5(\zeta_1, \zeta_2, \zeta_3)}$$

which can using Gaussian Quadrature becomes

$$\{R_f\}^e_A = \sum_{I=1}^{NP_1} \sum_{J=1}^{NP_2} \sum_{K=1}^{NP_3} w_I w_J w_K f_5(\zeta_1, \zeta_2, \zeta_3)$$

This integral has to be computed only once.

Integral 3

$$\{R\}_t = \int_\Omega \phi_A \cdot \boldsymbol{t} \, d\Omega = \sum_{e=1}^{N_e} \int_{\Omega_e} \phi_A \cdot \boldsymbol{t} \, d\Omega_e$$

$$\{R_t\}^e_A = \iint_{-1}^{1} \underbrace{(\{\hat{\phi}_i\}^T \boldsymbol{t}) \cdot |F| \; \sqrt{\mathbf{N} \cdot F^{-1} F^{-T} \cdot \mathbf{N}} \, |_{\zeta_i = -1,1} \; d\zeta_2 \, d\zeta_3}_{f_6(a, \zeta_2, \zeta_3)}$$

which can using Gaussian Quadrature becomes

$$\{R_t\}^e_A = \sum_{I=1}^{NP_1} \sum_{J=1}^{NP_2} w_I w_J f_6(a, \zeta_2, \zeta_3)$$

where a is a constant that represents the surface normal. The surface normal can be placed on any ζ_i depending on the specific boundary conditions. This integral has to be computed only once.

Applying the Boundary Conditions

Since this problem is a linear problem, the boundary conditions by forcing $v = 0 \quad on\, \Gamma_d$, which was applied as follows in the program:

Dirichlet Boundary Conditions

1. Find the rows of $[S]$ corresponding to the boundary nodes using the connectivity table

(a) Make the rows of the boundary nodes in $[S]$ equal to 0

(b) Make the diagonal element corresponding to that row equal to 1.

2. Find the rows of $\{R\}$ corresponding to the same boundary nodes using the connectivity table

   (a) Make the rows of the interior nodes in $\{R\}$ equal to $\bar{u}$.

This procedure ensures that each of the interior nodes always has the solution $u = \bar{u}$.

Neumann Boundary Conditions

The flux condition was applied by using the surface integral of the flux term and adding it to the load vector.

Conjugate Gradient Solver

The conjugate gradient solver calculates the solution of the finite element system by taking perpendicular steps for each iteration. The perpendicular steps make the solver efficent, as each component of the solution is tackled independently with minimal effort wasted. The mechanism of the conjugate gradient solver is based on the minimization of the following cost function

$$\pi = \frac{1}{2}\{a\}^T\mathbf{K}\{a\} - \{a\}^T\{R\}$$

where

$$\frac{\partial \pi}{\partial a} = \mathbf{K}\{a\} - \{R\} \Longrightarrow \min \pi \Rightarrow \mathbf{K}\{a\} = \{R\}$$

The general procedure of the conjugate gradient solver is shown below

$$\{a_{i+1}\} = \{a_i\} + \lambda\{z_i\}$$

$$\pi(a_{i+1}) = \frac{1}{2}(\{a_i\} + \lambda\{z_i\})^T\mathbf{K}(\{a_i\} + \lambda\{z_i\}) - (\{a_i\} + \lambda\{z_i\})^T\{R\}$$

$$\pi(a_{i+1}) = \frac{1}{2}(\{a_i\}^T\mathbf{K}\{a_i\} + 2\lambda\{z_i\}^T\mathbf{K}\{a_i\} + \lambda^2\{z_i\}^T\mathbf{K}\{z_i\}) - (\{a_i\} + \lambda\{z_i\})^T\{R\}$$

$$\frac{\partial \pi}{\partial \lambda} = \{z_i\}^T\mathbf{K}\{a_i\} + \lambda\{z_i\}^T\mathbf{K}\{z_i\} - \{z\}^T\{R\} = 0$$

$$\rightarrow \lambda_i = \frac{-\{z_i\}^T\mathbf{K}\{a_i\} + \{z_i\}^T\{R\}}{\{z_i\}^T\mathbf{K}\{z_i\}}$$

$$\{r_i\} = -\nabla\pi = \{R\} - \mathbf{K}\{a_i\}$$

$$\rightarrow \lambda_i = \frac{\{z_i\}^T\{r_i\}}{\{z_i\}^T\mathbf{K}\{z_i\}}$$

$$\{r_{i+1}\} = \{R\} - \mathbf{K}\{a_{i+1}\}$$

$$\{z_{i+1}\}\mathbf{K}\{z_i\} = 0 \Rightarrow \{z_{i+1}\} = \{r_{i+1}\} + \theta\{z_i\}$$

$$\rightarrow (\{r_{i+1}\} + \theta\{z_i\})^T\mathbf{K}\{z_i\} = 0$$

$$\rightarrow \theta_i = \frac{\{-r_{i+1}\}^T\mathbf{K}\{z_i\}}{\{z_i\}^T\mathbf{K}\{z_i\}}$$

The following pseudocode was implemented for the conjugate gradient solver used for the calculations:

1.

$$\textit{Initialize by guessing } \{a_0\}. \textit{ This gives } \{z_0\} = \{r_0\} = \{R\} - \mathbf{K}\{a_0\}$$

2.

$$\lambda_i = \frac{\{z_i\}^T\{r_i\}}{\{z_i\}^T\mathbf{K}\{z_i\}}$$

3.
$$\{a_{i+1}\} = \{a_i\} + \lambda\{z_i\}$$

4.
$$Calculate\ the\ error:\ error = \frac{\|\{a_{i+1}\} - \{a_i\}\|_K}{\|\{a_i\}\|_K} \quad where\ \|\{a\}\|_K = \sqrt{\{a\}^T\mathbf{K}\{a\}}$$

5.
$$If\ error \leq Tolerance \rightarrow Quit$$

6.
$$Else\ \{r_{i+1}\} = \{R\} - \mathbf{K}\{a_{i+1}\}$$

7.
$$\theta_i = \frac{\{-r_{i+1}\}^T\mathbf{K}\{z_i\}}{\{z_i\}^T\mathbf{K}\{z_i\}}$$

8.
$$\{z_{i+1}\} = \{r_{i+1}\} + \theta\{z_i\}$$

Reapeat the process from step 2 until $error \leq Tolerance$

3 Description of Software Design

Our software is divided into the following sections: a mesher to create the connectivity and node tables, a number of mathematical functions to perform integrations that create the load vector and stiffness matrix, and finally solver functions to solve the balance of linear momentum. Our code is written such that all non-integer numbers will be of type double, as this level of accuracy is generally required for our application. Thus, functions are not templated on float/double.

The mesher is contained in mesher.cpp and mesher.H, and does not rely on any other code. As we have limited our approach to cubic elements all that is required to build our tables is the number of nodes in each direction. The constructor takes this and initializes these tables to the correct size. The function createmesh takes no arguments and builds the actual tables using the member data. These tables can then be accessed using member functions. This approach was used mostly to utilize object-oriented concepts taught in class, and is not necessarily the most sensible for our program as a whole. One obvious disadvantage is that we have an object of class Mesher that we only use at the beginning of our program, but do not require later.

Some parameter interpolation is required, as described above, to treat our composite material. This is performed in hsbounds.cpp and hsbounds.H. This class consists of a single function that calculates the effective material properties.

Our integration functions are contained in blmintegrals.cpp and blmintegrals.H. While these technically form a class, their usage follows more of a procedural pattern. It is a fact of our problem that a number of different integrals on volumes and surfaces must be calculated only once, but require different input and output designs as well as different content. It is much easier to perform this in a procedural manner than object-oriented, so this was used.

A variety of dense linear algebra math functions are required to perform these integrals, and those are contained in femfunctions.cpp and femfunctions.H. Again this is technically a class, but acts in the manner of a group of functions. This could have been written by creating a class "matrix" that owns a vector of vectors of doubles, and performs these operations, but this provided no clear advantage to us. Usability-wise, it seemed more intuitive to enter arguments (e.g. matrix times vector to get vector) to a function in the order in which multiplication would be written on paper.

We have included in our project two solvers: the point Jacobi solver written in class (JacobiSolver.cpp and JacobiSolver.H) and a conjugate gradient solver (CGSolver.cpp and CGSolver.H) added for this project. Both act on a vector and a sparse matrix to create a solution vector. The Jacobi solver was used for debugging and as a comparison. Our conjugate gradient solver produces effectively identical results in far less time. For example, for a system with 10 nodes in each direction and 10^{-6} tolerance, the Jacobi solver

requires 1095 iterations taking 674 seconds, while the conjugate gradient solver takes 37 iterations taking 24 seconds on the same platform. Clearly the advantage to conjugate gradient is enormous for our project.

Finally, GridWrite.cpp and GridWrite.H contain the GridWrite class, which is a single that takes the node table, connectivity table and solution vector to produce a file for our visual output. The GridWrite function borrows from FEGrid and VisitWriter (both provided in class), but produces a .vtu file that uses the updated VisIt standard rather than a .vtk file. A number of other functions are present in our src directory for use in debugging and unit tests. Unit tests where created for our mesher, dense linear algebra, solver, and writer functions.

4 Results

In order to ensure that the program was working properly, we first conducted a couple of test studies. In the following cases, we applied a Dirichlet boundary condition on the bottom face of the cube and a traction force at the top face. All other face had zero Neumann boundary conditions applied. The expected result is a linear solution along the z axis, which is shown below.

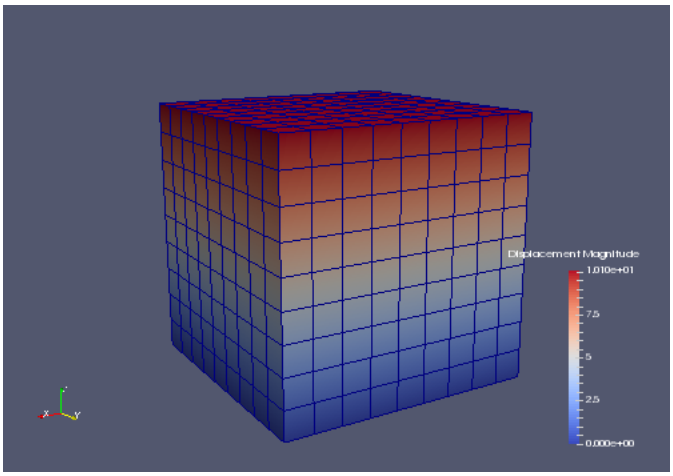
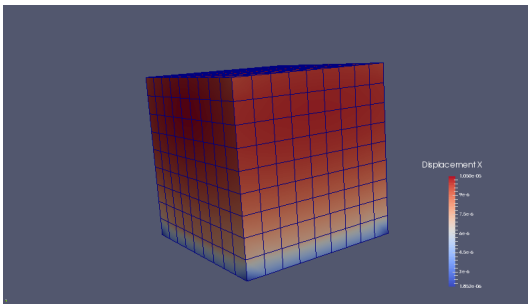


Figure 1: Linear Test Case

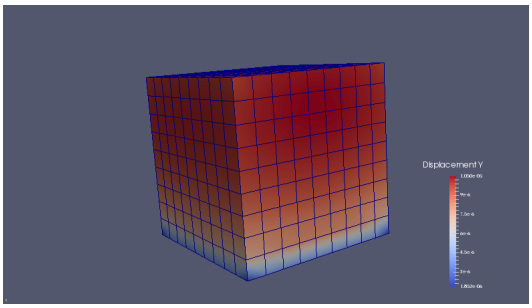
The figure above shows the expected solution, indicating that the program is solving the problem correctly. Following this benchmark, along with a couple of other tests (all of which can be found in "Unit Tests"), we conducted studies with a stiff multiphase material and a soft multiphase material.

Stiff Multiphase Material

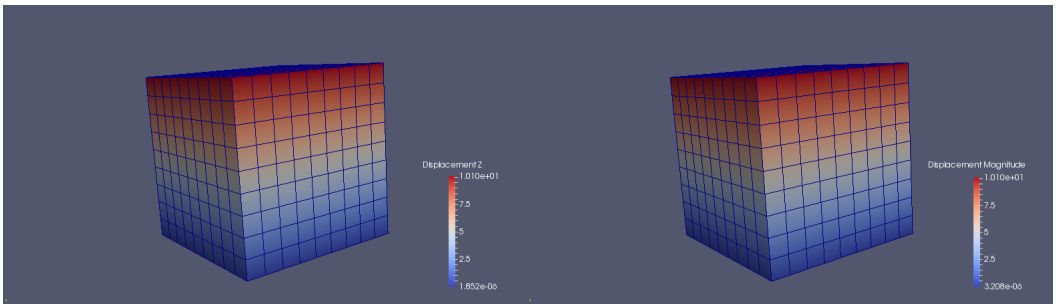
For this study we had a very stiff material that took up a significant volume fraction of the material, 0.9, and therefore accounted for the most of the mechanical behavior. The boundary conditions for this study for the same as the boundary conditions in the previously outlined test case. In addition to the above described boundary conditions, we also applied body forces in each direction. The images below show the displacement in each direction, as well as the overall magnitude of the displacement



(a) Displacement in the X-Direction



(b) Displacement in the Y-Direction



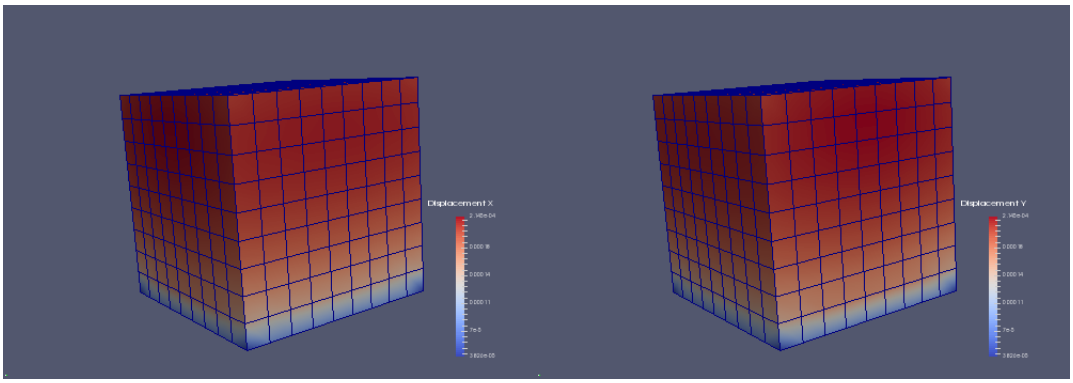
(c) Displacement in the Z-Direction

(d) Displacement Magnitude

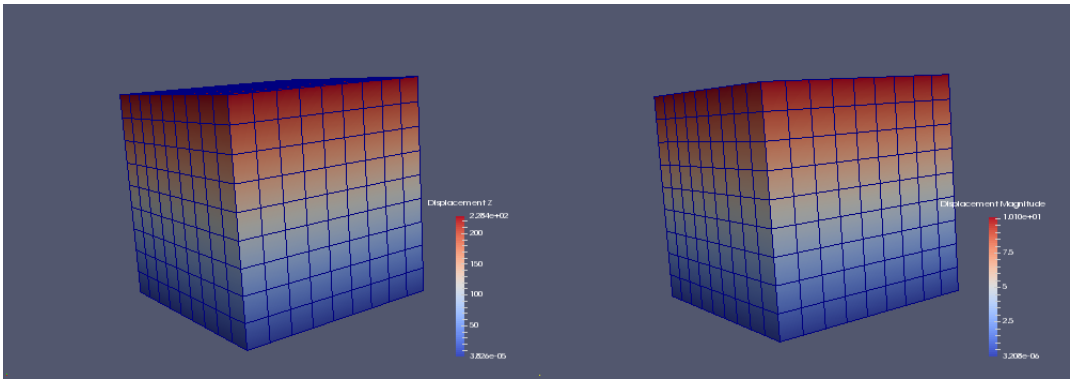
Figure 2: Stiff Multiphase Material

Soft Multiphase Material

For this study we had a soft material that took up a most of the volume fraction. We applied the same boundary conditions and body forces as in the case of the stiff material. We expected a similar behavior, given the similar setup of the problem. The different material property inputs, however, should yield different magnitudes for the displacements. The images below show the displacement in each direction, as well as the overall magnitude of the displacement:



(a) Displacement in the X-Direction (b) Displacement in the Y-Direction



(c) Displacement in the Z-Direction (d) Displacement Magnitude

Figure 3: Soft Multiphase Material

The images above show that the general elastic behavior of the material is similar to the stiff multiphase material, but with a larger magnitude of the displacement. This is the behavior that we expected.

5 Instructions to Run Test Example

Our program is very simple to run. Simply:

- enter Code/exec
- make all (provided you use g++, otherwise GNUmakefile requires change)
- ./groupF.exe <input file name>

We have provided an input file (input.txt) that may be used for testing. This file contains input data as well as comments on what that data is. The data input here are commonly changed parameters when running finite element codes. This includes the number of nodes in each direction on the grid, a force vector, and a traction vector. Also included are materials parameters for the composite material of interest (described above). Users may change any of these values, or add comments at the end of the file. The code is not currently equipped to handle changes to the existing comments lines, so please leave these as-is. This is not ideal, but users of other commonly-used codes are likely already familiar with picky input files.

When the code is run, a file named projectSolution.vtu will be output. This file can be imported to VisIt. To visualize, add a mesh using nodeData and a pseudocolor using DisplacementMagnitude.

6 References

[1] Zohdi, T. I. and Szeri, A. J. Fatigue of Kidney Stones with Heterogeneous Microstructure. Wiley Inter-Science. 2005.

[2] Hashin, Z. Analysis of Composite Materials - A Survey. Journal of Applied Mathemetics. Vol. 50/481. September 1983.

[3] Zohdi, T. I. and Wriggers, P. Introduction to computational micromechanics. Springer-Verlag. 2008.

[4] Zohdi, T.I. A Finite Element Primer For Beginners. Springer-Verlag. 2014.