ME 280a: HW 5

April Novak

November 16, 2016

1 Introduction and Objectives

The purpose of this study is to describe in great detail the process to solve a 3-D elastics problem, and also to generate a 3-D mesh with the appropriate connectivity matrix to be used for placing the local matrices and vectors into the global matrices and vectors.

2 Procedure

This section details the problem statement and mathematical method used for solving the problem.

2.1 Theoretical Problem Statement

The theory of elasticity is based on the continuum approximation, which assumes that the length scale on which solid particles exchange momentum is much smaller than the length scale characterizing the problem. An Eulerian perspective is adopted in the derivation of the governing equations, since control-volume-based approaches are much more amenable to solution than Lagrangian approaches (at least for small-deformation problems). Before deriving the governing equation, some preliminaries are necessary. The deformation gradient tensor $\mathbf{F}$ is used to map between two different coordinate systems. One of these coordinates frames is defined by the coordinates x_i (the present coordinates), and the other by X_i (the reference coordinates):

$$\bar{dx} = \mathbf{F}\bar{d\xi} \tag{1}$$

The Jacobian $\mathcal{J}$ is defined as the determinant of the deformation gradient tensor, and is required to transform integrals over the physical domain to the master element for application of quadrature rules:

$$\mathcal{J} \equiv \det \mathbf{F} \tag{2}$$

Finally, to have a physically meaningful transformation between two coordinate frames, the Jacobian must be positive. A balance of linear momentum in an arbitrary continuum is:

$$\frac{d}{dt}\int_\Omega \rho\dot{\mathbf{u}}dV = \int_{\partial\Omega}\mathbf{t}dS + \int_\Omega \rho\mathbf{b}dV \tag{3}$$

where Ω is the domain of the continuum (a volume), ρ is the density, $\mathbf{u}$ is the displacement vector, $\partial\Omega$ represents the boundary of the volume domain (a surface S), $\mathbf{t}$ represents the traction (loading on a surface), and $\mathbf{b}$ represents a body force such as gravity. Eq. (3) is no more than the Cauchy balance of linear momentum. By the Cauchy theorem, where surface forces are balanced on a tetrahedron, the traction is equivalent to:

$$\mathbf{t} \equiv \bar{\bar{\sigma}}^T \cdot \hat{n} \tag{4}$$

where $\hat{n}$ is a unit normal and $\bar{\bar{\sigma}}$ is the stress tensor, with components σ_{ji}, where j refers to the face on which the stress acts, and i the direction in which the stress points. The transpose appears in the above equation because, in fluids scenarios, the stress is usually defined in the opposite manner such that the first

index refers to the direction in which the stress points, and the second index the face on which the stress acts. The stress tensor is symmetric in virtually in applications for which there are no "micro-stresses" in the body. In other words, from a balance of angular momentum, and by inserting the Cauchy balance of momentum in Eq. (3), the stress tensor is symmetric so long as the only moments on the body are due to the stress and body forces. Inserting Eq. (4) into Eq. (3):

$$\frac{d}{dt}\int_\Omega \rho\dot{\mathbf{u}}dV = \int_{\partial\Omega} \bar{\bar{\sigma}}^T \cdot \hat{n}dS + \int_\Omega \rho\mathbf{b}dV \tag{5}$$

If mass is conserved in this continuum (it is assumed so), then the material derivative on the LHS of the above equation can be moved inside the integration to act only on $\dot{\mathbf{u}}$. This can conceptually be understood if the integral is divided into a finite sum over material elements. In a material element, mass is assumed conserved such that the time rate of change of ρdV is zero, which is why the time derivative acts only on $\dot{\mathbf{u}}$. This also assumes that the coordinate system is not moving in time.

$$\int_\Omega \rho\ddot{\mathbf{u}}dV = \int_{\partial\Omega} \bar{\bar{\sigma}}^T \cdot \hat{n}dS + \int_\Omega \rho\mathbf{b}dV \tag{6}$$

Applying Gauss's divergence theorem, which implicitly assumes that $\bar{\bar{\sigma}}$ is sufficiently smooth, and dropping the transpose on the stress tensor with the assumption that it is symmetric:

$$\int_\Omega \rho\ddot{\mathbf{u}}dV = \int_\Omega \nabla \cdot \bar{\bar{\sigma}}dV + \int_\Omega \rho\mathbf{b}dV \tag{7}$$

Then, by rearranging:

$$\int_\Omega \left(\rho\ddot{\mathbf{u}} - \nabla \cdot \bar{\bar{\sigma}} - \rho\mathbf{b}\right) dV = 0 \tag{8}$$

Because the selection of the control volume was arbitrary, the integrand must equal zero:

$$\rho\ddot{\mathbf{u}} = \nabla \cdot \bar{\bar{\sigma}} + \rho\mathbf{b} \tag{9}$$

Eq. (9) represents a balance of linear momentum, and is very general. It applies equally to fluids and solids, until a constitutive relationship is introduced for $\bar{\bar{\sigma}}$. The theory of linear elasticity is now introduced in order to provide this constitutive relationship. From the definition of the deformation gradient tensor $\mathbf{F}$ in Eq. (1), the Lagrangian strain tensor $\mathbf{E}$ is defined as:

$$\mathbf{E} \equiv \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T + \underbrace{(\nabla\mathbf{u})^T \cdot \nabla\mathbf{u}}_{\text{higher-order term}}\right) \tag{10}$$

With the linear theory of elasticity, deformations are assumed to be small such that the higher order term above can be neglected. The infinitesimal strain $\mathbf{e}$ is defined as:

$$\mathbf{e} \equiv \frac{1}{2}\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right) \tag{11}$$

and for small deformations, $\mathbf{E} \approx \mathbf{e}$. (Note: I couldn't figure out how to get the "epsilon" variable to work out, so epsilon is replaced by e in this document). A similar derivation of a conservation of energy equation gives:

$$\rho\dot{w} = \bar{\bar{\sigma}} : \nabla\dot{\mathbf{u}} - \nabla \cdot \mathbf{q} + \rho z \tag{12}$$

where w is the internal energy per unit mass, $\mathbf{q}$ the heat flux vector, and z the volumetric heating source. If thermal effects are neglected, then the above equation simplifies to:

$$\rho\dot{w} = \bar{\bar{\sigma}} : \nabla\dot{\mathbf{u}} \tag{13}$$

Applying the chain rule to the LHS above:

$$\rho \frac{\partial w}{\partial \mathbf{e}} : \frac{d\mathbf{e}}{dt} = \bar{\bar{\sigma}} : \nabla \dot{\mathbf{u}} \tag{14}$$

from which it is clear that:

$$\bar{\bar{\sigma}} = \rho \frac{\partial w}{\partial \mathbf{e}} \tag{15}$$

In order to develop a constitutive relationship for the theory of linear elasticity, it is assumed that a stored elastic energy function exists that depends only on the deformation. For small strains beginning from an undeformed state, energy is stored in the material. The stored elastic energy function W is by definition:

$$W \equiv \rho w \tag{16}$$

The simplest function that satisfies Eq. (15) is:

$$W = \frac{1}{2} \mathbf{e} : \mathbf{IE} : \mathbf{e} \tag{17}$$

where **IE** is the fourth rank elasticity tensor. With these assumptions, the constitutive relationship for the stress tensor becomes:

$$\bar{\bar{\sigma}} = \mathbf{IE} : \mathbf{e} \tag{18}$$

Due to the linear relationship between stress and strain, this constitutive relationship is referred to as the "theory of linear elasticity." Because **IE** is a fourth-order tensor, it has 81 unique entries ($3^4 = 81$). However, because the stress tensor is symmetric, there are actually only 36 unique entries. With all 36 constants, the relationship in Eq. (18) is:

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{bmatrix} = \begin{bmatrix} E_{1111} & E_{1122} & E_{1133} & E_{1112} & E_{1123} & E_{1113} \\ E_{2211} & E_{2222} & E_{2233} & E_{2212} & E_{2223} & E_{2213} \\ E_{3311} & E_{3322} & E_{3333} & E_{3312} & E_{3323} & E_{3313} \\ E_{1211} & E_{1222} & E_{1233} & E_{1212} & E_{1223} & E_{1213} \\ E_{2311} & E_{2322} & E_{2333} & E_{2312} & E_{2323} & E_{2313} \\ E_{1311} & E_{1322} & E_{1333} & E_{1312} & E_{1323} & E_{1313} \end{bmatrix} \begin{bmatrix} e_{11} \\ e_{22} \\ e_{33} \\ 2e_{12} \\ 2e_{23} \\ 2e_{31} \end{bmatrix} \tag{19}$$

where the strain vector shows that $e_{12} = e_{21}$. For isotropic materials, **IE** simplifies by defining two parameters, κ and μ:

$$\mathbf{IE} = \begin{bmatrix} \kappa + \frac{4}{3}\mu & \kappa - \frac{2}{3}\mu & \kappa - \frac{2}{3}\mu & 0 & 0 & 0 \\ \kappa - \frac{2}{3}\mu & \kappa + \frac{4}{3}\mu & \kappa - \frac{2}{3}\mu & 0 & 0 & 0 \\ \kappa - \frac{2}{3}\mu & \kappa - \frac{2}{3}\mu & \kappa + \frac{4}{3}\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{20}$$

The eigenvalues of **IE** are directly proportional to μ and κ, and hence both μ and κ must be positive in order to retain the positive definiteness of **IE**. Combining the above results with the conservation of momentum statement in Eq. (9), the balance of momentum for a linear elastic system is:

$$\rho \ddot{\mathbf{u}} = \nabla \cdot (\mathbf{IE} : \nabla \mathbf{u}) + \rho \mathbf{b} \tag{21}$$

The finite element implementation, beginning with the weak form, is discussed next.

2.2 The Weak Form

The weak form of Eq. (21) is obtained by multiplying through by a test function $\mathbf{v}$, where, like $\mathbf{u}$, $\mathbf{v}$ is a vector-valued function. Then, integrating over the body:

$$\int_\Omega \rho\ddot{\mathbf{u}} \cdot \mathbf{v} d\Omega = \int_\Omega (\nabla \cdot \bar{\bar{\sigma}}) \cdot \mathbf{v} d\Omega + \int_\Omega \rho\mathbf{b} \cdot \mathbf{v} d\Omega \tag{22}$$

To apply the product rule, use the following identity:

$$\nabla \cdot (\bar{\bar{\sigma}} \cdot \mathbf{v}) = (\nabla \cdot \bar{\bar{\sigma}}) \cdot \mathbf{v} + \nabla\mathbf{v} : \bar{\bar{\sigma}} \tag{23}$$

Using this identity in Eq. (22):

$$\int_\Omega \rho\ddot{\mathbf{u}} \cdot \mathbf{v} d\Omega = \int_\Omega \left(\nabla \cdot (\bar{\bar{\sigma}} \cdot \mathbf{v}) - \nabla\mathbf{v} : \bar{\bar{\sigma}}\right) d\Omega + \int_\Omega \rho\mathbf{b} \cdot \mathbf{v} d\Omega \tag{24}$$

The divergence theorem can be applied to the first term on the RHS to give:

$$\int_\Omega \rho\ddot{\mathbf{u}} \cdot \mathbf{v} d\Omega = \int_{\partial\Omega} \bar{\bar{\sigma}} \cdot \mathbf{v} \cdot \hat{n} dS - \int_\Omega \nabla\mathbf{v} : \bar{\bar{\sigma}} d\Omega + \int_\Omega \rho\mathbf{b} \cdot \mathbf{v} d\Omega \tag{25}$$

where S is the boundary area of the volume Ω with unit normal vector $\hat{n}$. The traction is defined from Eq. (4), and hence the above simplifies to:

$$\begin{aligned} \int_\Omega \rho\ddot{\mathbf{u}} \cdot \mathbf{v} d\Omega &= \int_{\partial\Omega} \mathbf{t} \cdot \mathbf{v} dA - \int_\Omega \nabla\mathbf{v} : \bar{\bar{\sigma}} d\Omega &&+ \int_\Omega \rho\mathbf{b} \cdot \mathbf{v} d\Omega \\ \int_\Omega \rho\ddot{\mathbf{u}} \cdot \mathbf{v} d\Omega &= \int_{\partial\Omega} \mathbf{t} \cdot \mathbf{v} dA - \int_\Omega \nabla\mathbf{v} : \mathbf{IE} : \nabla\mathbf{u}\ d\Omega &&+ \int_\Omega \rho\mathbf{b} \cdot \mathbf{v} d\Omega \end{aligned} \tag{26}$$

For convenience, it is assumed that the weight function $\mathbf{v}$ goes to zero on the displacement boundaries such that the area integral that appears in the weak form above applies strictly to boundaries on which the traction is specified. Hence, the weak form can be stated as:

$$\begin{aligned} &\text{Find } \mathbf{u} \in \mathbf{H}^u(\Omega) \subset \mathbf{H}^1(\Omega) \text{ so that } \mathbf{u}|_{\Gamma_u} = \mathbf{d} \text{ and so that } \forall\ \mathbf{v} \in \mathbf{H}^v(\Omega) \subset \mathbf{H}^1(\Omega), \mathbf{v}|_{\Gamma_u} = \mathbf{0}, \\ &\qquad\qquad \text{and for } \mathbf{t} \in \mathbf{L}^2(\Gamma_t), \mathbf{t} = \mathbf{t}^*|_{\Gamma_t} \text{ and } \mathbf{b} \in \mathbf{L}^2(\Omega) \\ &\int_\Omega \rho\ddot{\mathbf{u}} \cdot \mathbf{v} d\Omega = \int_{\partial\Omega} \mathbf{t} \cdot \mathbf{v} dA - \int_\Omega \nabla\mathbf{v} : \mathbf{IE} : \nabla\mathbf{u} d\Omega + \int_\Omega \rho\mathbf{b} \cdot \mathbf{v} d\Omega \end{aligned} \tag{27}$$

where $\mathbf{d}$ is the vector of known displacements on the displacement boundary, $\mathbf{t}^*$ is a vector of known tractions on the traction boundary, and Γ_u and Γ_t indicate the portions of the boundary for which there are known displacements and tractions, respectively. This weak form is more general than the strong form because it does not assume differentiability of the stress. The particular weighted residual method to be applied is the Bubnov-Galerkin method, where both the solution and the weight function are expanded in the same basis functions. Hence, both the displacement and the weight function are in $\mathbf{H}^1(\Omega)$, the space necessary to ensure finite integrals in the weak form above. For other methods such as the Petrov-Galerkin method, where different shape functions are used to expand the solution and weight function, $\mathbf{u}$ and $\mathbf{v}$ do not necessarily need to come from the same space, but they must be in at least $\mathbf{H}^1(\Omega)$.

The space $\mathbf{H}^1(\Omega)$ is a Hilbert-space norm, where the 1 superscript indicates that it contains all functions whose highest finite derivative is the first derivative. This is the space from which the shape and weight functions most come because at most a first derivative is required in the weak form. For other applications, where for example, the highest derivative present in the weak form is a second derivative, then the weight and shape functions would need to be in $\mathbf{H}^2(\Omega)$ in order for all integrals to remain finite. In other words, $\mathbf{u}$ is in $\mathbf{H}^1(\Omega)$ if the following statement is true:

$$\|\mathbf{u}\|^2_{\mathbf{H}^1(\Omega)} = \int_\Omega \left(\frac{\partial u_i}{\partial x_j}\right)^2 d\Omega + \int_\Omega u_i u_i d\Omega < \infty \tag{28}$$

where suffix notation is implied. Because $\mathbf{u}$ is vector-valued, the space of approximation functions must also be vector-valued. This simple extension then requires that each component of $\mathbf{u}$ be in $H^1(\Omega)$. This is indicated by boldface in Eq. (27). Because the traction and body load also appear in the integrals in Eq. (27), there is a requirement on the space of functions from which they can inhabit. From the weak form, no differentiation of these functions is required, so they must be within $\mathbf{H}^0(\Omega)$, sometimes referred to as $\mathbf{L}^2(\Omega)$. The weak form in Eq. (27) is equivalent to the strong form in Eq. (21) provided that the solution is sufficiently differentiable that the higher derivatives required in the strong form are defined. Next, the specifics of the finite element implementation are given in order to specify the above to the finite element method.

2.2.1 The Finite Element Weak Form

This section covers the details regarding finite element implementation of Eq. (27). To implement this weak form, first the displacement, body force, traction, and weight function must be defined:

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} \rho b_1 \\ \rho b_2 \\ \rho b_3 \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{29}$$

The governing conservation of momentum equation is a vector-valued equation, so each component of $\mathbf{u}$, and the weight function $\mathbf{v}$, are expanded in a series of shape functions:

$$\mathbf{u}^h = \begin{bmatrix} \sum_{i=1}^{n_{en}} a_i \phi_i \\ \sum_{i=1}^{n_{en}} a_{i+n_{en}} \phi_i \\ \sum_{i=1}^{n_{en}} a_{i+2n_{en}} \phi_i \end{bmatrix} \tag{30}$$

where the above expansion applies over a single element with n_{en} nodes per element. $3n_{en}$ unknowns are solved for in each element for a 3-D problem. A similar expansion is performed for the weight function:

$$\mathbf{v}^h = \begin{bmatrix} \sum_{i=1}^{n_{en}} b_i \phi_i \\ \sum_{i=1}^{n_{en}} b_{i+n_{en}} \phi_i \\ \sum_{i=1}^{n_{en}} b_{i+2n_{en}} \phi_i \end{bmatrix} \tag{31}$$

The number and order of these shape functions determines the order of the finite element approximation. The weak form in Eq. (27) can be written in matrix form as follows, noting that the dot product of two vectors can be written as $\mathbf{a} \cdot \mathbf{b} = \mathbf{b}^T \mathbf{a}$:

$$\int_\Omega \rho \mathbf{v}^T \ddot{\mathbf{u}} d\Omega = \int_{\Gamma_t} \mathbf{v}^T \mathbf{t} dA - \int_\Omega \nabla \mathbf{v} : \mathbf{IE} : \nabla \mathbf{u} d\Omega + \int_\Omega \mathbf{v}^T \rho \mathbf{b} d\Omega + P^* \int_{\Gamma_u} \mathbf{v}^T (\mathbf{d} - \mathbf{u}) dA \tag{32}$$

where the penalty term has been included for completeness (though it could be neglected if static condensation were to be performed on the matrix system prior to solving). To transform the second term on the RHS, use the identity of the second order inner product:

$$A : B = A_{ij} B_{ij} = \mathrm{Tr}(A^T B) \tag{33}$$

Then, the transpose of a product is:

$$(AB)^T = A^T B^T \tag{34}$$

Then, Eq. (32) becomes:

$$\int_\Omega \rho \mathbf{v}^T \ddot{\mathbf{u}} d\Omega = \int_{\Gamma_t} \mathbf{v}^T \mathbf{t} dA - \int_\Omega (\mathbf{D}\mathbf{v})^T \mathbf{IE} (\mathbf{D}\mathbf{u}) d\Omega + \int_\Omega \mathbf{v}^T \rho \mathbf{b} d\Omega + P^* \int_{\Gamma_u} \mathbf{v}^T (\mathbf{d} - \mathbf{u}) dA \tag{35}$$

where $\mathbf{D}$ is the deformation tensor:

$$\mathbf{D} = \begin{bmatrix} \frac{\partial}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial}{\partial x_3} \\ \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} & 0 \\ 0 & \frac{\partial}{\partial x_3} & \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_3} & 0 & \frac{\partial}{\partial x_1} \end{bmatrix} \tag{36}$$

The goal is to solve for the expansion coefficients on $\mathbf{u}$. This requires decomposing the expansion in Eq. (30) into a matrix containing just the shape functions and a vector containing the unknowns. This can be done in two different ways - the first way is to organize all the expansion coefficients pertaining to u_1 in a vector $\mathbf{a}$, following by all the coefficients for u_2, and followed by all the coefficients for u_3. This is the approach used in this class. An alternative method is to organize the vector of unknowns such that all unknowns for a node appear together, i.e. the first three entries would be the coefficients pertaining to u_1, then u_2, then u_3, for the first node. This second approach is used in ME 180, but is not used here. So, the displacement is organized as:

$$\mathbf{u}^h = \bar{\bar{\phi}}\mathbf{a} \tag{37}$$

where $\mathbf{a}$ is a vector of unknowns containing all the unknowns pertaining to the x-displacement first, following by all the unknowns pertaining to the y-displacement second, and for the z-displacement third. The matrix $\bar{\bar{\phi}}$ is defined in a different manner depending on the particular value of the node number:

$$\bar{\bar{\phi}} = \begin{cases} (\phi_i, 0, 0)^T & i < N \\ (0, \phi_i, 0)^T & N+1 \leq i \leq 2N \\ (0, 0, \phi_i)^T & 2N+1 \leq i \leq 3N \end{cases} \tag{38}$$

The Bubnov-Galerkin method is used so that the approximation for $\mathbf{v}$ is performed in the same manner as for $\mathbf{u}$, so that:

$$\mathbf{v}^h = \bar{\bar{\phi}}\mathbf{b} \tag{39}$$

Inserting these matrix forms into the weak form gives:

$$\int_\Omega \rho(\bar{\bar{\phi}}\mathbf{b})^T\bar{\bar{\phi}}\ddot{\mathbf{a}}d\Omega = \int_{\Gamma_t}(\bar{\bar{\phi}}\mathbf{b})^T\mathbf{t}dA - \int_\Omega(\mathbf{D}\bar{\bar{\phi}}\mathbf{b})^T\mathbf{I}\mathbf{E}(\mathbf{D}\bar{\bar{\phi}}\mathbf{a})d\Omega + \int_\Omega(\bar{\bar{\phi}}\mathbf{b})^T\mathbf{f}d\Omega + P^*\int_{\Gamma_u}(\bar{\bar{\phi}}\mathbf{b})^T(\mathbf{d} - \bar{\bar{\phi}}\mathbf{a})dA \tag{40}$$

where the fact that the shape functions are not functions of time has been included in the first term and the body force $\rho\mathbf{b}$ has been replaced by $\mathbf{f}$ to avoid confusion with the expansion coefficients of the weight function. Because $\mathbf{b}$ is arbitrary, the above can be rearranged such that $\mathbf{b}$ acts on a single term. Then, because that term, multiplied by an arbitrary term, must be zero, it can be concluded that the term equals zero. In other words, $\mathbf{b}$ can be "canceled" from each term, though this isn't exactly what is happening.

$$\int_\Omega \rho(\bar{\bar{\phi}})^T\bar{\bar{\phi}}\ddot{\mathbf{a}}d\Omega = \int_{\Gamma_t}(\bar{\bar{\phi}})^T\mathbf{t}dA - \int_\Omega(\mathbf{D}\bar{\bar{\phi}})^T\mathbf{I}\mathbf{E}(\mathbf{D}\bar{\bar{\phi}}\mathbf{a})d\Omega + \int_\Omega(\bar{\bar{\phi}})^T\mathbf{f}d\Omega + P^*\int_{\Gamma_u}(\bar{\bar{\phi}})^T(\mathbf{d} - \bar{\bar{\phi}}\mathbf{a})dA \tag{41}$$

At this point, it is assumed that the system is in steady state. Then, the above reduces to the following, where the penalty term is split up into the portion that is a function of $\mathbf{a}$ and the portion that is not:

$$\int_\Omega(\mathbf{D}\bar{\bar{\phi}})^T\mathbf{I}\mathbf{E}(\mathbf{D}\bar{\bar{\phi}}\mathbf{a})d\Omega + P^*\int_{\Gamma_u}(\bar{\bar{\phi}})^T(\bar{\bar{\phi}}\mathbf{a})dA = \int_{\partial\Omega}(\bar{\bar{\phi}})^T\mathbf{t}dA + \int_\Omega(\bar{\bar{\phi}})^T\mathbf{f}d\Omega + P^*\int_{\Gamma_u}(\bar{\bar{\phi}})^T\mathbf{d}dA \tag{42}$$

For simplicity, the above terms can be defined as matrices:

$$\begin{aligned}\mathbf{K} &\equiv \int_\Omega (\mathbf{D}\bar{\bar{\phi}})^T \mathbf{I}\mathbf{E}(\mathbf{D}\bar{\bar{\phi}}) d\Omega + P^* \int_{\Gamma_u} \bar{\bar{\phi}}^T \bar{\bar{\phi}} dA \\ \mathbf{R} &\equiv \int_{\Gamma_t} (\bar{\bar{\phi}})^T \mathbf{t} dA + \int_\Omega (\bar{\bar{\phi}})^T \mathbf{f} d\Omega + P^* \int_{\Gamma_u} (\bar{\bar{\phi}})^T \mathbf{d} dA\end{aligned} \tag{43}$$

to give the matrix system:

$$\mathbf{K}\mathbf{a} = \mathbf{R} \tag{44}$$

So, the weak form for the FE method is:

$$\begin{aligned}&\text{Find } \mathbf{u} \in \mathbf{H}^u(\Omega) \subset \mathbf{H}^1(\Omega) \text{ so that } \mathbf{u}|_{\Gamma_u} = \mathbf{d} \text{ and so that } \forall\ \mathbf{v} \in \mathbf{H}^v(\Omega) \subset \mathbf{H}^1(\Omega), \mathbf{v}|_{\Gamma_u} = \mathbf{0}, \\ &\qquad \text{and for } \mathbf{t} \in \mathbf{L}^2(\Gamma_t), \mathbf{t} = \mathbf{t}^*|_{\Gamma_t} \text{ and } \mathbf{b} \in \mathbf{L}^2(\Omega) \\ &\int_\Omega (\mathbf{D}\bar{\bar{\phi}})^T \mathbf{I}\mathbf{E}(\mathbf{D}\bar{\bar{\phi}}\mathbf{a}) d\Omega = \int_{\partial\Omega} (\bar{\bar{\phi}})^T \mathbf{t} dA + \int_\Omega (\bar{\bar{\phi}})^T \mathbf{f} d\Omega\end{aligned} \tag{45}$$

where the penalty term has been removed because it is asked for explicitly in the next portion of this assignment. The definitions for all the terms that appear above have been given previously. Section 2.2.2 will show the weak form with the penalty method, while Section 2.3 will show how the above is applied element-by-element.

2.2.2 The Finite Element Weak Form - Penalty Method

The penalty method is a means by which to apply Dirichlet boundary conditions without the tedious need to separate rows and columns from the stiffness and loading vectors. From Eq. (45), the weak form for a finite element implementation requires that the weight functions are zero on essential boundaries. The penalty method relaxes this requirement, and adds a term to the weak form to account for violation of a Dirichlet boundary condition on a Dirichlet boundary. This method is widely-used, but is not strictly required to apply Dirichlet boundary conditions - the alternative of separating rows and columns containing known quantities, and subtracting from the load vector, can always be performed. The penalty method adds a term to Eq. (45):

$$\begin{aligned}&\text{Find } \mathbf{u} \in \mathbf{H}^u(\Omega) \subset \mathbf{H}^1(\Omega) \text{ so that } \mathbf{u}|_{\Gamma_u} = \mathbf{d} \text{ and so that } \forall\ \mathbf{v} \in \mathbf{H}^v(\Omega) \subset \mathbf{H}^1(\Omega), \\ &\qquad \text{and for } \mathbf{t} \in \mathbf{L}^2(\Gamma_t), \mathbf{t} = \mathbf{t}^*|_{\Gamma_t} \text{ and } \mathbf{b} \in \mathbf{L}^2(\Omega) \\ &\int_\Omega (\mathbf{D}\bar{\bar{\phi}})^T \mathbf{I}\mathbf{E}(\mathbf{D}\bar{\bar{\phi}}\mathbf{a}) d\Omega + P^* \int_{\Gamma_u} (\bar{\bar{\phi}})^T (\bar{\bar{\phi}}\mathbf{a}) dA = \int_{\partial\Omega} (\bar{\bar{\phi}})^T \mathbf{t} dA + \int_\Omega (\bar{\bar{\phi}})^T \mathbf{f} d\Omega + P^* \int_{\Gamma_u} (\bar{\bar{\phi}})^T \mathbf{d} dA\end{aligned} \tag{46}$$

where P^* represents something like a spring constant, and is a large, positive number. A high value of this artificial spring constant will apply a traction to "force" the displacement boundary to be satisfied on Γ_u, the displacement boundary. In other words, $P^*(\mathbf{d} - \mathbf{u}) \approx \mathbf{t}_{penalty}$, so the penalty term represents a traction that enforces the Dirichlet boundary condition. This term, however, would never be applied if we still required $\mathbf{v}|_{\Gamma_u} = \mathbf{0}$, and so the kinematic restrictions on the weight functions are dropped, and they do not need to be zero on the Dirichlet boundaries.

The penalty method is motivated by the fact that, for symmetric systems for which a potential can be defined, an augmented potential can be defined, whose variation is the weak from in Eq. (46):

$$\mathcal{T}(u) = \mathcal{T}(u) + P^* \int_{\Gamma_u} (\mathbf{d} - \mathbf{u})(\mathbf{d} - \mathbf{u}) dA \tag{47}$$

Hence, the penalty method with this interpretation is a quadratic addition to the potential energy.

2.3 Finite Element Implementation

2.3.1 Element-by-Element Matrices and Vectors

This section provides in explicit detail the forms of the element stiffness matrices and load vectors using the penalty method (the penalty terms could simply be dropped if needed, and hence the penalty method is discussed here to be as complete as possible). While Eq. (46) holds over the entire domain, the strength of the finite element method is that the integrals in Eq. (46) can be performed over each element, since the shape functions are a nodal basis such that they are only nonzero at a single node. Once assembling into the global stiffness matrix, this gives a sparse system. So, for an element e, the element stiffness matrix and load vector are:

$$\begin{aligned} \mathbf{K}^e &\equiv \int_{\Omega_e} (\mathbf{D}\bar{\bar{\phi}})^T \mathbf{I}\mathbf{E} (\mathbf{D}\bar{\bar{\phi}}) d\Omega_e + P^* \int_{\Gamma_{u,e}} \bar{\bar{\phi}}^T \bar{\bar{\phi}} dA_e \\ \mathbf{R}^e &\equiv \int_{\Gamma_{t,e}} (\bar{\bar{\phi}})^T \mathbf{t} dA_e + \int_{\Omega_e} (\bar{\bar{\phi}})^T \mathbf{f} d\Omega_e + P^* \int_{\Gamma_{u,e}} (\bar{\bar{\phi}})^T \mathbf{d} dA_e \end{aligned} \tag{48}$$

where $\Gamma_{u,e}$ is the intersection of the boundary of element e with the displacement boundary and $\Gamma_{t,e}$ is the intersection of the boundary of element e with the traction boundary. All of these integrals are performed in the master domain using quadrature rules. This master domain is a cube defined over $-1 \leq \xi_1 \leq 1, -1 \leq \xi_2 \leq 1, -1 \leq \xi_3 \leq 1$. In order to transform between the physical and master domain, a transformation rule is needed to map between x, y, z and ξ_1, ξ_2, ξ_3. This transformation rule can take many forms, but a convenient one is to simply use the shape function expansion:

$$x_i = \sum_{j=1}^{n_{en}} X_{i,j} \phi_j(\xi_1, \xi_2, \xi_3) \tag{49}$$

where the shape functions are defined over the master element, $X_{i,j}$ are the physical (real) coordinates, and i refers to the fact that the above expansion is assumed to apply equally for x_1, x_2 and x_3. This mapping, which uses the shape functions as the basis for the mapping, is called a parametric map.

So, all the integrals in $\mathbf{K}^e$ and $\mathbf{R}^e$ are performed over a single element by transforming the integrals to the master domain. The integrals in the physical domain are with respect to $\bar{dx}$, and to transform them to the master domain, the deformation gradient tensor $\mathbf{F}$ defined by Eq. (1) is used. Using the chain rule reveals the form of $\mathbf{F}$:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_2} & \frac{\partial x_1}{\partial \xi_3} \\ \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_2} & \frac{\partial x_2}{\partial \xi_3} \\ \frac{\partial x_3}{\partial \xi_1} & \frac{\partial x_3}{\partial \xi_2} & \frac{\partial x_3}{\partial \xi_3} \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} \tag{50}$$

Or, in shorthand notation:

$$dx_i = F_{ij} \xi_j \tag{51}$$

where suffix notation is implied. The inverse relationship is:

$$d\xi_j = F_{ji} dx_i = F^{-1}_{ij} dx_i \tag{52}$$

The above transformation rule holds for the volume integral Jacobian - Nanson's formula must be used to transform the area integrals appearing in the element stiffness matrix and load vector to area integrals in the master domain. There is no reason that the Jacobian for the volume transformation be the same as that for the surface transformation, which is why more care must be taken here to use the correct transformation. In this case, area integrals are transformed by Nanson's rule:

$$dA_e = (\mathcal{J}\ \mathbf{F}^{-T} \cdot \hat{N}) \cdot \hat{n} d\hat{A}_e \tag{53}$$

where $\hat{n}$ is a unit normal to the master element surface (must be determined for each element on the surface), $\hat{N}$ the normal to the physical element surface (must be determined), $\mathcal{J}$ is the Jacobian defined in Eq. (2), and $\mathbf{F}$ is the deformation gradient tensor defined in Eq. (50). Then, all the integrals over areas must set one of ξ_i to ± 1 to be consistent with the fact that on a surface, one of the master coordinates is held constant. So, in order to perform computations over each element, Eq. (54) becomes:

$$\begin{aligned}\mathbf{K}^e &\equiv \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}(\mathbf{D}\bar{\bar{\phi}})^T\mathbf{I}\mathbf{E}(\mathbf{D}\bar{\bar{\phi}})|\mathbf{F}|d\xi_1 d\xi_2 d\xi_3 + P^*\int_{\Gamma_{u,e}}(\bar{\bar{\phi}})^T\bar{\bar{\phi}}(\mathcal{J}\ \mathbf{F}^{-T}\cdot\hat{N})\cdot\hat{n}d\hat{A}_e \\ \mathbf{R}^e &\equiv \int_{\Gamma_{t,e}}(\bar{\bar{\phi}})^T\mathbf{t}(\mathcal{J}\ \mathbf{F}^{-T}\cdot\hat{N})\cdot\hat{n}d\hat{A}_e + \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}(\bar{\bar{\phi}})^T\mathbf{f}\,|\mathbf{F}|d\xi_1 d\xi_2 d\xi_3 + P^*\int_{\Gamma_{u,e}}(\bar{\bar{\phi}})^T\mathbf{d}(\mathcal{J}\ \mathbf{F}^{-T}\cdot\hat{N})\cdot\hat{n}d\hat{A}_e\end{aligned} \tag{54}$$

where it is implied in the area integrals that one of the spatial coordinates is set to ± 1 according to the orientation of the surface. To accurately perform the integration in the master element, everything within the integrand must also be converted to the master coordinate frame. Hence, the deformation gradient becomes:

$$\mathbf{D} = \begin{bmatrix} \frac{\partial}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_1} + \frac{\partial}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_1} + \frac{\partial}{\partial\xi_3}\frac{\partial\xi_3}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_2} + \frac{\partial}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_2} + \frac{\partial}{\partial\xi_3}\frac{\partial\xi_3}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_3} + \frac{\partial}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_3} + \frac{\partial}{\partial\xi_3}\frac{\partial\xi_3}{\partial x_3} \\ \frac{\partial}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_2} + \frac{\partial}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_2} + \frac{\partial}{\partial\xi_3}\frac{\partial\xi_3}{\partial x_2} & \frac{\partial}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_1} + \frac{\partial}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_1} + \frac{\partial}{\partial\xi_3}\frac{\partial\xi_3}{\partial x_1} & 0 \\ 0 & \frac{\partial}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_3} + \frac{\partial}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_3} + \frac{\partial}{\partial\xi_3}\frac{\partial\xi_3}{\partial x_3} & \frac{\partial}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_2} + \frac{\partial}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_2} + \frac{\partial}{\partial\xi_3}\frac{\partial\xi_3}{\partial x_2} \\ \frac{\partial}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_3} + \frac{\partial}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_3} + \frac{\partial}{\partial\xi_3}\frac{\partial\xi_3}{\partial x_3} & 0 & \frac{\partial}{\partial\xi_1}\frac{\partial\xi_1}{\partial x_1} + \frac{\partial}{\partial\xi_2}\frac{\partial\xi_2}{\partial x_1} + \frac{\partial}{\partial\xi_3}\frac{\partial\xi_3}{\partial x_1} \end{bmatrix} \tag{55}$$

In addition, the matrix $\bar{\bar{\phi}}$ is defined in terms of the shape functions over the master domain (indicated with hats below):

$$\bar{\bar{\phi}} = \begin{cases} (\hat{\phi}_i, 0, 0)^T & i < N \\ (0, \hat{\phi}_i, 0)^T & N+1 \le i \le 2N \\ (0, 0, \hat{\phi}_i)^T & 2N+1 \le i \le 3N \end{cases} \tag{56}$$

So, for trilinear bricks, the element stiffness and load vectors become:

$$\begin{aligned}\mathbf{K}^e &\equiv \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}(\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T\underbrace{\mathbf{I}\mathbf{E}}_{6\times 6}(\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24})\underbrace{|\mathbf{F}|}_{\text{scalar}}d\xi_1 d\xi_2 d\xi_3 + \\ &\qquad \underbrace{P^*}_{\text{scalar}}\int_{\Gamma_{u,e}}(\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T\underbrace{\bar{\bar{\phi}}}_{3\times 24}(\underbrace{\mathcal{J}}_{\text{scalar}}(\underbrace{\mathbf{F}}_{3\times 3})^{-T}\cdot\hat{N})\cdot\hat{n}d\hat{A}_e \\ \mathbf{R}^e &\equiv \int_{\Gamma_{t,e}}(\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T\underbrace{\mathbf{t}}_{3\times 1}(\underbrace{\mathcal{J}}_{\text{scalar}}(\underbrace{\mathbf{F}}_{3\times 3})^{-T}\cdot\hat{N})\cdot\hat{n}d\hat{A}_e + \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}(\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T\underbrace{\mathbf{f}}_{3\times 1}\underbrace{|\mathbf{F}|}_{\text{scalar}}d\xi_1 d\xi_2 d\xi_3 + \\ &\qquad \underbrace{P^*}_{\text{scalar}}\int_{\Gamma_{u,e}}(\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T\mathbf{d}(\underbrace{\mathcal{J}}_{\text{scalar}}(\underbrace{\mathbf{F}}_{3\times 3})^{-T}\cdot\hat{N})\cdot\hat{n}d\hat{A}_e\end{aligned} \tag{57}$$

where 3 is the number of spatial dimensions (3 for 3-D) and 24 is the number of unknowns per element (8 for 3-D). The net result is that $\mathbf{K}^e$ is a 24×24 matrix, and $\mathbf{R}^e$ is a 24×1 column vector. The trilinear

brick shape functions (in the master domain) are defined in Eq. (65). The unit normals are of length 3, and are oriented such that the correct dimensions are obtained in the matrix and load integrals above.

All the integrals above are performed using quadrature. To integrate in higher than a single dimension using a Gaussian quadrature rule, simply apply the rule in each direction. So, in 1-D, where a single loop sums over all the quadrature points, three loops are needed to sum over the ξ_1, ξ_2, ξ_3 directions. For instance, the integrals above, in quadrature form, are:

$$\begin{aligned}\mathbf{K}^e = \sum_{q=1}^{g}\sum_{r=1}^{g}\sum_{s=1}^{g} w_q w_g w_s \left\{ (\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{IE}}_{6\times 6}(\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24})\underbrace{|\mathbf{F}|}_{\text{scalar}} \right\} + \\ \sum_{r=1}^{g}\sum_{s=1}^{g} w_g w_s \left\{ \underbrace{P^*}_{\text{scalar}} \int_{\Gamma_{u,e}} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\bar{\bar{\phi}}}_{3\times 24} (\mathcal{J}\ \mathbf{F}^{-T}\cdot\hat{N})\cdot\hat{n} d\hat{A}_e \right\}\end{aligned} \tag{58}$$

$$\mathbf{R}^e \equiv \sum_{q=1}^{g}\sum_{r=1}^{g}\sum_{s=1}^{g} w_q w_g w_s (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{f}}_{3\times 1}\ \underbrace{|\mathbf{F}|}_{\text{scalar}} +$$

$$\sum_{r=1}^{g}\sum_{s=1}^{g} w_g w_s \left\{ \int_{\Gamma_{t,e}} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{t}}_{3\times 1} (\mathcal{J}\ \mathbf{F}^{-T}\cdot\hat{N})\cdot\hat{n} d\hat{A}_e + \underbrace{P^*}_{\text{scalar}} \int_{\Gamma_{u,e}} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \mathbf{d} (\mathcal{J}\ \mathbf{F}^{-T}\cdot\hat{N})\cdot\hat{n} d\hat{A}_e \right\}$$

where g is the number of quadrature points, where the same quadrature rule has been assumed to be applied in each spatial dimension. It is implied that in the area integrals, $d\hat{A}_e$ refers to one of $d\xi_1 d\xi_2$, $d\xi_2 d\xi_3, d\xi_3 d\xi_1$, depending on the particular surface.

To be as explicit as possible, there are essentially four types of elements. The element stiffness matrix and load vector for each of these possible combinations is shown below for completeness.

1. an element on the interior (no displacement or traction boundaries)

$$\begin{aligned}\mathbf{K}^e &\equiv \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{IE}}_{6\times 6}(\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24})\underbrace{|\mathbf{F}|}_{\text{scalar}}\, d\xi_1 d\xi_2 d\xi_3 \\ \mathbf{R}^e &\equiv \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{f}}_{3\times 1}\ \underbrace{|\mathbf{F}|}_{\text{scalar}}\, d\xi_1 d\xi_2 d\xi_3\end{aligned} \tag{59}$$

2. an element on the boundary with only displacement boundaries

$$\begin{aligned}\mathbf{K}^e \equiv \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{IE}}_{6\times 6}(\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24})\underbrace{|\mathbf{F}|}_{\text{scalar}}\, d\xi_1 d\xi_2 d\xi_3 + \\ \underbrace{P^*}_{\text{scalar}} \int_{\Gamma_{u,e}} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\bar{\bar{\phi}}}_{3\times 24} (\underbrace{\mathcal{J}}_{\text{scalar}}\ (\underbrace{\mathbf{F}}_{3\times 3})^{-T}\cdot\hat{N})\cdot\hat{n} d\hat{A}_e\end{aligned} \tag{60}$$

$$\mathbf{R}^e \equiv \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{f}}_{3\times 1}\ \underbrace{|\mathbf{F}|}_{\text{scalar}}\, d\xi_1 d\xi_2 d\xi_3 + \underbrace{P^*}_{\text{scalar}} \int_{\Gamma_{u,e}} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \mathbf{d} (\underbrace{\mathcal{J}}_{\text{scalar}}\ (\underbrace{\mathbf{F}}_{3\times 3})^{-T}\cdot\hat{N})\cdot\hat{n} d\hat{A}_e$$

3. an element on the boundary with only traction boundaries

$$\mathbf{K}^e \equiv \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{IE}}_{6\times 6}(\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24}) \underbrace{|\mathbf{F}|}_{\text{scalar}} d\xi_1 d\xi_2 d\xi_3$$

$$\mathbf{R}^e \equiv \int_{\Gamma_{t,e}} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{t}}_{3\times 1}(\underbrace{\mathcal{J}}_{\text{scalar}} (\underbrace{\mathbf{F}}_{3\times 3})^{-T}\cdot \hat{N})\cdot \hat{n} d\hat{A}_e + \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{f}}_{3\times 1} \underbrace{|\mathbf{F}|}_{\text{scalar}} d\xi_1 d\xi_2 d\xi_3 \tag{61}$$

4. an element on the boundary with both displacement and traction boundaries

$$\mathbf{K}^e \equiv \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{IE}}_{6\times 6}(\underbrace{\mathbf{D}}_{6\times 3}\underbrace{\bar{\bar{\phi}}}_{3\times 24}) \underbrace{|\mathbf{F}|}_{\text{scalar}} d\xi_1 d\xi_2 d\xi_3 + \underbrace{P^*}_{\text{scalar}} \int_{\Gamma_{u,e}} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\bar{\bar{\phi}}}_{3\times 24}(\underbrace{\mathcal{J}}_{\text{scalar}} (\underbrace{\mathbf{F}}_{3\times 3})^{-T}\cdot \hat{N})\cdot \hat{n} d\hat{A}_e$$

$$\mathbf{R}^e \equiv \int_{\Gamma_{t,e}} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{t}}_{3\times 1}(\underbrace{\mathcal{J}}_{\text{scalar}} (\underbrace{\mathbf{F}}_{3\times 3})^{-T}\cdot \hat{N})\cdot \hat{n} d\hat{A}_e + \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \underbrace{\mathbf{f}}_{3\times 1} \underbrace{|\mathbf{F}|}_{\text{scalar}} d\xi_1 d\xi_2 d\xi_3 + \underbrace{P^*}_{\text{scalar}} \int_{\Gamma_{u,e}} (\underbrace{\bar{\bar{\phi}}}_{3\times 24})^T \mathbf{d}(\underbrace{\mathcal{J}}_{\text{scalar}} (\underbrace{\mathbf{F}}_{3\times 3})^{-T}\cdot \hat{N})\cdot \hat{n} d\hat{A}_e \tag{62}$$

Care must be taken for determining if an element is on a Dirichlet boundary. If it is, then penalty terms must be added, and if not, the penalty terms are absent. So, whether or not the penalty method is used, there is some amount of bookkeeping required to record which nodes correspond to Dirichlet boundaries.

2.3.2 Global-Local Transformation

After all computations over the elements are complete, all the local stiffness matrices and load vectors must be organized into the global stiffness matrix and load vector. The placement of local matrices into the global matrix is performed using a connectivity matrix that relates the local node numbers to the global node numbers. A connectivity matrix is usually organized so that each row corresponds to a single element. Then, each column in that row refers to each local node to that element, and the information held in the connectivity matrix are the global node numbers relating to the local node numbers. For example, for a 2-D domain with four elements, and global node numbering beginning in the bottom left corner and moving up to the top right corner, the connectivity matrix (also called the location matrix in this document) has the following form:

$$\mathbf{LM} = \begin{bmatrix} 1 & 2 & 5 & 4 \\ 2 & 3 & 6 & 5 \\ 4 & 5 & 8 & 7 \\ 5 & 6 & 9 & 8 \end{bmatrix} \tag{63}$$

where the local nodes are numbered in a counterclockwise manner beginning from the bottom left node. Then, for example, the second row in the global stiffness matrix would be assembled as:

$$\mathbf{K}(2,:) = \begin{bmatrix} k_{2,1}^{e=1}, & k_{2,2}^{e=1} + k_{1,1}^{e=2}, & k_{1,2}^{e=2}, & k_{2,4}^{e=1}, & k_{1,4}^{e=2} + k_{2,3}^{e=1}, & k_{1,3}^{e=2}, & 0, & 0, & 0 \end{bmatrix} \tag{64}$$

2.4 Shape Functions

The shape functions for 3-D finite elements are a natural extension of the shape functions used in lower dimensions. Trilinear elements, or 3-D linear elements, are used for the remainder of this assignment. This choice of shape functions represents a nodal basis, since the shape functions go to zero at all nodes except for the node for which they are defined. The trilinear shape functions are:

$$\begin{aligned}
\phi_1(\xi_1,\xi_2,\xi_3) &= \frac{1}{8}(1-\xi_1)(1-\xi_2)(1-\xi_3) \\
\phi_2(\xi_1,\xi_2,\xi_3) &= \frac{1}{8}(1+\xi_1)(1-\xi_2)(1-\xi_3) \\
\phi_3(\xi_1,\xi_2,\xi_3) &= \frac{1}{8}(1+\xi_1)(1+\xi_2)(1-\xi_3) \\
\phi_4(\xi_1,\xi_2,\xi_3) &= \frac{1}{8}(1-\xi_1)(1+\xi_2)(1-\xi_3) \\
\phi_5(\xi_1,\xi_2,\xi_3) &= \frac{1}{8}(1-\xi_1)(1-\xi_2)(1+\xi_3) \\
\phi_6(\xi_1,\xi_2,\xi_3) &= \frac{1}{8}(1+\xi_1)(1-\xi_2)(1+\xi_3) \\
\phi_7(\xi_1,\xi_2,\xi_3) &= \frac{1}{8}(1+\xi_1)(1+\xi_2)(1+\xi_3) \\
\phi_8(\xi_1,\xi_2,\xi_3) &= \frac{1}{8}(1-\xi_1)(1+\xi_2)(1+\xi_3)
\end{aligned} \tag{65}$$

This gives 24 total unknowns per element (since the unknown is vector-valued with three components).

3 Mesh Generator

This section discusses the mesh generator used to mesh the tubular "S" structure given in the assignment. The meshing begins in each θ-chunk. For the circular cross-section structure, the x and y coordinates are related to each other by:

$$x^2 + y^2 = r \tag{66}$$

where r is the inner radius of the tubular structure. Each piece in the circumferential direction is defined according to θ, where $0 \leq \theta \leq 2\pi$ defines the "slice" parallel to $\vec{e}_r$. The generation of the mesh is based on determining the coordinates of each node. The first node is assigned to the first "slice" for $\theta = 0$. In the discussion of the mesh generator, "slice" refers to each plane for which there exists a hollowed annulus (that is meshed according to the number of layers and circumferential points). Θ is the angle referring to the circumferential angle, while θ refers to the angle in each slice. The overall algorithm for generating the coordinates for the mesh is as follows:

1. Begin with slice for $\Theta = 0$. Beginning then for $\theta = 0$, move counterclockwise around the first layer (inner surface of the tube). Increment θ in units of $2\pi/N_c$, and for each θ, assign the x and y coordinates according to Eq. (66).

2. After finishing the inner layer, increase r by dt, where dt is the thickness of each layer, to move to the next layer, then repeat step 1.

3. Repeat steps 1 and 2 until all layers in each slice have been meshed, where for each layer, dt is added.

4. Now that a slice has been meshed, repeat for all slices. This requires determining the x, y, z centers for each new slice. This is performed by sweeping through Θ. The y coordinate is $y = 0$ for all points, while the x coordinate continually increases moving from slice to slice, and the z coordinate is positive for the left half of the tube, and negative for the right half of the tube.

5. Now that all slices have been meshed, the mesh looks like as follows for $N_c = 8, N_\theta = 8, N_t = 3$.

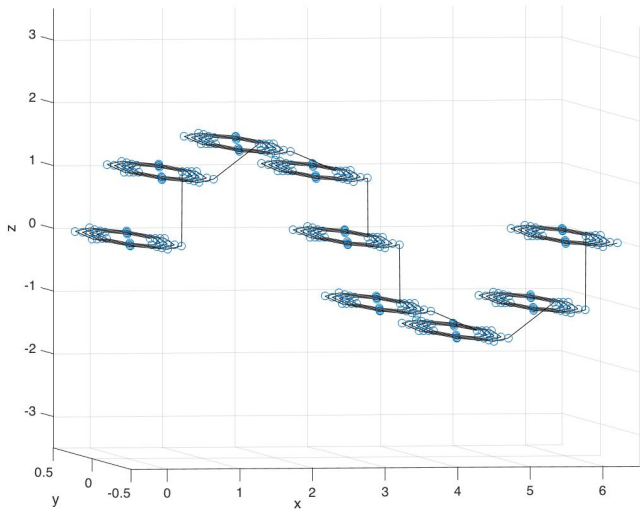

Figure 1. Mesh for $N_c = 8, N_\theta = 8, N_t = 3$ with no tilt to the slices. Lines connect each coordinate for better visuality.

The next step is to rotate the slices appropriately through the angle Θ for each slice so that a tubular structure is formed. Only the x and z coordinates must be modified. To perform the tilt, the following quantities are computed using trigonometry:

$$\begin{aligned} w &= \sin(\pi/2 - \Theta)(r + dt)\cos(\theta)/\sin(pi/2) \\ h &= w\sin(\Theta) \\ p &= w\cos(\Theta) \end{aligned} \tag{67}$$

To tilt the z-coordinate, for points in the first and fourth quadrant of each slice:

$$z_{new} = z - (-1)^{tube} h \tag{68}$$

And for points in the second and third quadrant of each slice:

$$z_{new} = z + (-1)^{tube} h \tag{69}$$

where *tube* is a variable indicating whether or not the slice is in the left or right half of the tube. For the left half of the tube, $tube = 1$, and in the right half, $tube = 2$.

6. Then, tilt the x-coordinates. For points in the first and fourth quadrants of each slice:

$$x_{new} = x + p \tag{70}$$

And for points in the second and third quadrants of each slice:

$$x_{new} = x - p \tag{71}$$

7. Finally, tilt the slices that exactly align with the peak and valley of the tube (for odd numbers of N_θ, this would not be performed). For the slices that align with the peaks and for nodes in the first and fourth quadrants, adjust the z coordinates according to:

$$z_{new} = z - (r + dt)\cos(\theta) \tag{72}$$

And for points in the second and third quadrants:

$$z_{new} = z + (r + dt)\cos(\theta) \tag{73}$$

The x-coordinates are adjusted by simply setting all of them to the centroid coordinate for that slice.

This process is fairly complicated, and reveals why meshing software is so valuable. The process here is left fairly general that it can apply for any values of N_θ, N_c, N_t, but any slight change in the geometry completely invalidates the program. The final mesh for $N_c = 8, N_\theta = 8, N_t = 3$ is shown below. This mesh is shown because the requested mesh with only $N_c = 4$ is relatively difficult to perceive in a 3-D plot in Matlab, so the following plot better reveals the mesh. Lines are drawn between each coordinate.

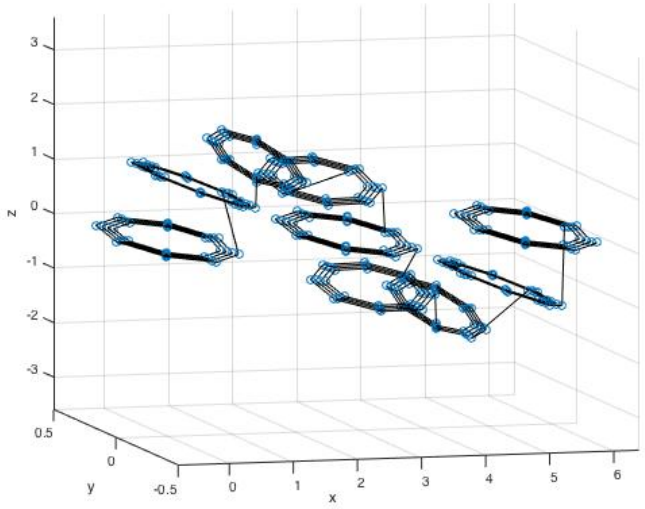

Figure 2. Mesh for $N_c = 8, N_\theta = 8, N_t = 3$. Lines connect each coordinate for better visuality.

The coarser mesh, for $N_c = 4, N_\theta = 8, N_t = 3$ is shown below, again with lines connecting each coordinate for better visibility.

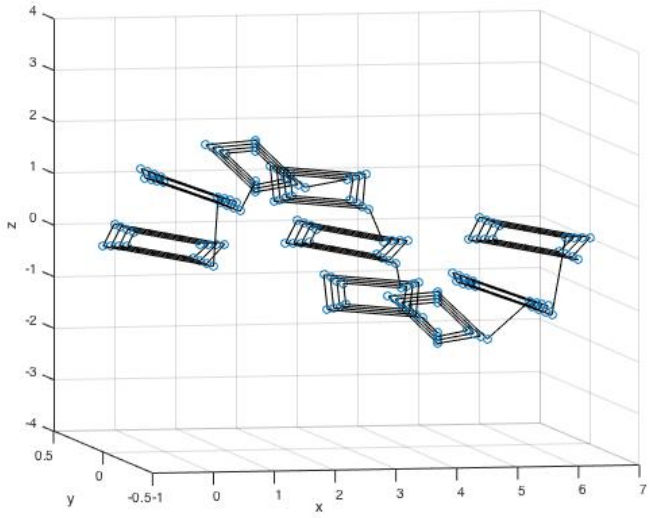

Figure 3. Mesh for $N_c = 4, N_\theta = 8, N_t = 3$. Lines connect each coordinate for better visuality.

The mesh in Fig. 3 is shown below without the lines connecting each coordinate.

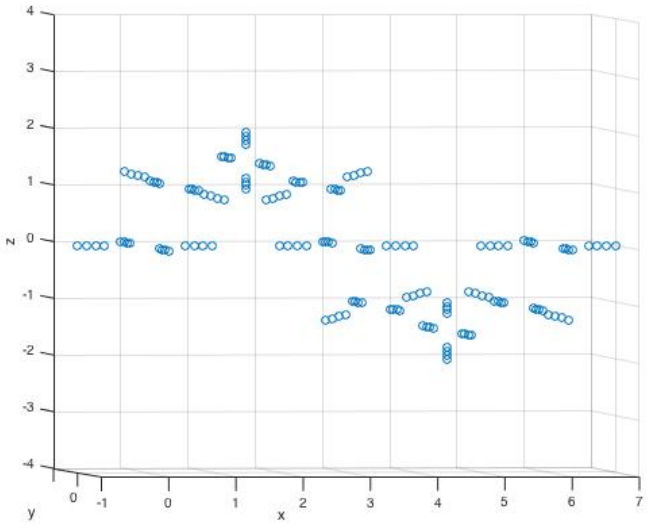

Figure 4. Mesh for $N_c = 4, N_\theta = 8, N_t = 3$.

Note that this assignment did not provide the dimensions of the tube, so I assumed that the inner radius of each arch was 1, the radius of the inner hole of the tube 0.3, and the thickness of the tube 0.2.

3.1 The Connectivity Matrix

In order for this mesh to be useful for finite element implementation, a connectivity function must be defined to relate the local node numbering to the global node numbering. The mesh generated numbers the global nodes according to the order in which they were generated. For instance, the first 8 nodes are in the inner layer of the first slice, the next 8 are in the second layer of the first slice, and so on for the first slice. Then, moving to the next Θ slice, the node numbering again begins on the inside of the tube and moves

counterclockwise in layers until reaching the outside of the tube. This is shown schematically in the figures above by the black lines connecting the coordinates *in the order in which the coordinates are generated.*

The connectivity matrix is an $N \times 8$ matrix, where N is the total number of elements and 8 is the number of local nodes per element (linear elements are assumed). The local node numbering is performed according to a clockwise fashion. The following schematic shows the node numbering, where the left portion shows the front face, and the right portion shows the back face, all while looking at the front face (i.e. the back face is not written with the perspective of looking at the outward-facing portion of the back face).

$$\begin{array}{ll} 4 - -3 & 8 - -7 \\ 2 - -1 & 6 - -5 \end{array} \tag{74}$$

So, for each slice, the nodes on the face of each element can be determined using a numbering scheme that follows the order in which the nodes were defined. Beginning with $\theta = 0$, and moving counterclockwise, the local nodes are numbered, moving progressively outwards in the layers until reaching the last node for a particular slice.

There are $N_t \cdot N_c \cdot N_\theta$ total elements. For each slice, the nodes are numbered moving counterclockwise, beginning at the same node that is meshed first. After each layer is complete, the numbering moves to the next layer in the same fashion. Once an entire slice is complete, the next slice is also meshed. This defines only the *frontal* node numberings shown in the above equation. For example, for $N_\theta = 2, N_t = 3, N_c = 4$, the connectivity matrix **LM** looks like the following *before* the nodes on the backs of the first 12 elements are related to the nodes on the fronts of the next 12 elements.

$$\mathbf{LM} = \begin{bmatrix}
1 & 2 & 5 & 6 & 0 & 0 & 0 & 0 \\
2 & 3 & 6 & 7 & 0 & 0 & 0 & 0 \\
3 & 4 & 7 & 8 & 0 & 0 & 0 & 0 \\
4 & 1 & 8 & 5 & 0 & 0 & 0 & 0 \\
5 & 6 & 9 & 10 & 0 & 0 & 0 & 0 \\
6 & 7 & 10 & 11 & 0 & 0 & 0 & 0 \\
7 & 8 & 11 & 12 & 0 & 0 & 0 & 0 \\
8 & 5 & 12 & 9 & 0 & 0 & 0 & 0 \\
9 & 10 & 13 & 14 & 0 & 0 & 0 & 0 \\
10 & 11 & 14 & 15 & 0 & 0 & 0 & 0 \\
11 & 12 & 15 & 16 & 0 & 0 & 0 & 0 \\
12 & 9 & 16 & 13 & 0 & 0 & 0 & 0 \\
17 & 18 & 21 & 22 & 0 & 0 & 0 & 0 \\
18 & 19 & 22 & 23 & 0 & 0 & 0 & 0 \\
19 & 20 & 23 & 24 & 0 & 0 & 0 & 0 \\
20 & 17 & 24 & 21 & 0 & 0 & 0 & 0 \\
21 & 22 & 25 & 26 & 0 & 0 & 0 & 0 \\
22 & 23 & 26 & 27 & 0 & 0 & 0 & 0 \\
23 & 24 & 27 & 28 & 0 & 0 & 0 & 0 \\
24 & 21 & 28 & 25 & 0 & 0 & 0 & 0 \\
25 & 26 & 29 & 30 & 0 & 0 & 0 & 0 \\
26 & 27 & 30 & 31 & 0 & 0 & 0 & 0 \\
27 & 28 & 31 & 32 & 0 & 0 & 0 & 0 \\
28 & 25 & 32 & 29 & 0 & 0 & 0 & 0 \\
33 & 34 & 37 & 38 & 0 & 0 & 0 & 0 \\
34 & 35 & 38 & 39 & 0 & 0 & 0 & 0 \\
35 & 36 & 39 & 40 & 0 & 0 & 0 & 0 \\
36 & 33 & 40 & 37 & 0 & 0 & 0 & 0 \\
37 & 38 & 41 & 42 & 0 & 0 & 0 & 0 \\
38 & 39 & 42 & 43 & 0 & 0 & 0 & 0 \\
39 & 40 & 43 & 44 & 0 & 0 & 0 & 0 \\
40 & 37 & 44 & 41 & 0 & 0 & 0 & 0 \\
41 & 42 & 45 & 46 & 0 & 0 & 0 & 0 \\
42 & 43 & 46 & 47 & 0 & 0 & 0 & 0 \\
43 & 44 & 47 & 48 & 0 & 0 & 0 & 0 \\
44 & 41 & 48 & 45 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{75}$$

This is not the final form for the connectivity matrix (a.k.a. location matrix). Because the slices lay exactly on top of one another, the front nodes of the second slice are exactly the back nodes on the previous slice. With this knowledge, the back nodes for each element can be assigned based on the frontal nodes of the following slice. Then, the last $N_c N_t$ rows in the location matrix above can be deleted, since they refer to the frontal nodes of a slice that does not technically exist (there are only 2 slices, but 3 planes defining those slices). With this information, the final form of the location matrix becomes, for $N_\theta = 2, N_t = 3, N_c = 4$ for example:

$$\mathbf{LM} = \begin{bmatrix} 1 & 2 & 5 & 6 & 17 & 18 & 21 & 22 \\ 2 & 3 & 6 & 7 & 18 & 19 & 22 & 23 \\ 3 & 4 & 7 & 8 & 19 & 20 & 23 & 24 \\ 4 & 1 & 8 & 5 & 20 & 17 & 24 & 21 \\ 5 & 6 & 9 & 10 & 21 & 22 & 25 & 26 \\ 6 & 7 & 10 & 11 & 22 & 23 & 26 & 27 \\ 7 & 8 & 11 & 12 & 23 & 24 & 27 & 28 \\ 8 & 5 & 12 & 9 & 24 & 21 & 28 & 25 \\ 9 & 10 & 13 & 14 & 25 & 26 & 29 & 30 \\ 10 & 11 & 14 & 15 & 26 & 27 & 30 & 31 \\ 11 & 12 & 15 & 16 & 27 & 28 & 31 & 32 \\ 12 & 9 & 16 & 13 & 28 & 25 & 32 & 29 \\ 17 & 18 & 21 & 22 & 33 & 34 & 37 & 38 \\ 18 & 19 & 22 & 23 & 34 & 35 & 38 & 39 \\ 19 & 20 & 23 & 24 & 35 & 36 & 39 & 40 \\ 20 & 17 & 24 & 21 & 36 & 33 & 40 & 37 \\ 21 & 22 & 25 & 26 & 37 & 38 & 41 & 42 \\ 22 & 23 & 26 & 27 & 38 & 39 & 42 & 43 \\ 23 & 24 & 27 & 28 & 39 & 40 & 43 & 44 \\ 24 & 21 & 28 & 25 & 40 & 37 & 44 & 41 \\ 25 & 26 & 29 & 30 & 41 & 42 & 45 & 46 \\ 26 & 27 & 30 & 31 & 42 & 43 & 46 & 47 \\ 27 & 28 & 31 & 32 & 43 & 44 & 47 & 48 \\ 28 & 25 & 32 & 29 & 44 & 41 & 48 & 45 \end{bmatrix} \tag{76}$$

Each row in the location matrix corresponds to an elements, and each column to a local node number, so that $LM(1,4)$ indicates the global node number of local node number 4 in element 1. This method is extended to the case for $N_\theta = 8, N_c = 4, N_t = 3$, where the purpose of the previous discussion for a fewer number of circumferential elements was simply to illustrate the process by which the location matrix is generated. So, for the problem statement in this homework assignment ($N_\theta = 8, N_c = 4, N_t = 3$):

$$\mathbf{LM} = \begin{bmatrix} LM_1 \\ LM_2 \end{bmatrix} \tag{77}$$

where, in order to be able to print the matrix, the following components are defined to simply be stacked on top of each other as in Eq. (77).

$$LM_1 = \begin{bmatrix}
1 & 2 & 5 & 6 & 17 & 18 & 21 & 22 \\
2 & 3 & 6 & 7 & 18 & 19 & 22 & 23 \\
3 & 4 & 7 & 8 & 19 & 20 & 23 & 24 \\
4 & 1 & 8 & 5 & 20 & 17 & 24 & 21 \\
5 & 6 & 9 & 10 & 21 & 22 & 25 & 26 \\
6 & 7 & 10 & 11 & 22 & 23 & 26 & 27 \\
7 & 8 & 11 & 12 & 23 & 24 & 27 & 28 \\
8 & 5 & 12 & 9 & 24 & 21 & 28 & 25 \\
9 & 10 & 13 & 14 & 25 & 26 & 29 & 30 \\
10 & 11 & 14 & 15 & 26 & 27 & 30 & 31 \\
11 & 12 & 15 & 16 & 27 & 28 & 31 & 32 \\
12 & 9 & 16 & 13 & 28 & 25 & 32 & 29 \\
17 & 18 & 21 & 22 & 33 & 34 & 37 & 38 \\
18 & 19 & 22 & 23 & 34 & 35 & 38 & 39 \\
19 & 20 & 23 & 24 & 35 & 36 & 39 & 40 \\
20 & 17 & 24 & 21 & 36 & 33 & 40 & 37 \\
21 & 22 & 25 & 26 & 37 & 38 & 41 & 42 \\
22 & 23 & 26 & 27 & 38 & 39 & 42 & 43 \\
23 & 24 & 27 & 28 & 39 & 40 & 43 & 44 \\
24 & 21 & 28 & 25 & 40 & 37 & 44 & 41 \\
25 & 26 & 29 & 30 & 41 & 42 & 45 & 46 \\
26 & 27 & 30 & 31 & 42 & 43 & 46 & 47 \\
27 & 28 & 31 & 32 & 43 & 44 & 47 & 48 \\
28 & 25 & 32 & 29 & 44 & 41 & 48 & 45 \\
33 & 34 & 37 & 38 & 49 & 50 & 53 & 54 \\
34 & 35 & 38 & 39 & 50 & 51 & 54 & 55 \\
35 & 36 & 39 & 40 & 51 & 52 & 55 & 56 \\
36 & 33 & 40 & 37 & 52 & 49 & 56 & 53 \\
37 & 38 & 41 & 42 & 53 & 54 & 57 & 58 \\
38 & 39 & 42 & 43 & 54 & 55 & 58 & 59 \\
39 & 40 & 43 & 44 & 55 & 56 & 59 & 60 \\
40 & 37 & 44 & 41 & 56 & 53 & 60 & 57 \\
41 & 42 & 45 & 46 & 57 & 58 & 61 & 62 \\
42 & 43 & 46 & 47 & 58 & 59 & 62 & 63 \\
43 & 44 & 47 & 48 & 59 & 60 & 63 & 64 \\
44 & 41 & 48 & 45 & 60 & 57 & 64 & 61 \\
49 & 50 & 53 & 54 & 65 & 66 & 69 & 70 \\
50 & 51 & 54 & 55 & 66 & 67 & 70 & 71 \\
51 & 52 & 55 & 56 & 67 & 68 & 71 & 72 \\
52 & 49 & 56 & 53 & 68 & 65 & 72 & 69 \\
53 & 54 & 57 & 58 & 69 & 70 & 73 & 74 \\
54 & 55 & 58 & 59 & 70 & 71 & 74 & 75 \\
55 & 56 & 59 & 60 & 71 & 72 & 75 & 76 \\
56 & 53 & 60 & 57 & 72 & 69 & 76 & 73 \\
57 & 58 & 61 & 62 & 73 & 74 & 77 & 78 \\
58 & 59 & 62 & 63 & 74 & 75 & 78 & 79 \\
59 & 60 & 63 & 64 & 75 & 76 & 79 & 80 \\
60 & 57 & 64 & 61 & 76 & 73 & 80 & 77 \\
65 & 66 & 69 & 70 & 81 & 82 & 85 & 86
\end{bmatrix} \tag{78}$$

$$LM_2 = \begin{bmatrix} 66 & 67 & 70 & 71 & 82 & 83 & 86 & 87 \\ 67 & 68 & 71 & 72 & 83 & 84 & 87 & 88 \\ 68 & 65 & 72 & 69 & 84 & 81 & 88 & 85 \\ 69 & 70 & 73 & 74 & 85 & 86 & 89 & 90 \\ 70 & 71 & 74 & 75 & 86 & 87 & 90 & 91 \\ 71 & 72 & 75 & 76 & 87 & 88 & 91 & 92 \\ 72 & 69 & 76 & 73 & 88 & 85 & 92 & 89 \\ 73 & 74 & 77 & 78 & 89 & 90 & 93 & 94 \\ 74 & 75 & 78 & 79 & 90 & 91 & 94 & 95 \\ 75 & 76 & 79 & 80 & 91 & 92 & 95 & 96 \\ 76 & 73 & 80 & 77 & 92 & 89 & 96 & 93 \\ 81 & 82 & 85 & 86 & 97 & 98 & 101 & 102 \\ 82 & 83 & 86 & 87 & 98 & 99 & 102 & 103 \\ 83 & 84 & 87 & 88 & 99 & 100 & 103 & 104 \\ 84 & 81 & 88 & 85 & 100 & 97 & 104 & 101 \\ 85 & 86 & 89 & 90 & 101 & 102 & 105 & 106 \\ 86 & 87 & 90 & 91 & 102 & 103 & 106 & 107 \\ 87 & 88 & 91 & 92 & 103 & 104 & 107 & 108 \\ 88 & 85 & 92 & 89 & 104 & 101 & 108 & 105 \\ 89 & 90 & 93 & 94 & 105 & 106 & 109 & 110 \\ 90 & 91 & 94 & 95 & 106 & 107 & 110 & 111 \\ 91 & 92 & 95 & 96 & 107 & 108 & 111 & 112 \\ 92 & 89 & 96 & 93 & 108 & 105 & 112 & 109 \\ 97 & 98 & 101 & 102 & 113 & 114 & 117 & 118 \\ 98 & 99 & 102 & 103 & 114 & 115 & 118 & 119 \\ 99 & 100 & 103 & 104 & 115 & 116 & 119 & 120 \\ 100 & 97 & 104 & 101 & 116 & 113 & 120 & 117 \\ 101 & 102 & 105 & 106 & 117 & 118 & 121 & 122 \\ 102 & 103 & 106 & 107 & 118 & 119 & 122 & 123 \\ 103 & 104 & 107 & 108 & 119 & 120 & 123 & 124 \\ 104 & 101 & 108 & 105 & 120 & 117 & 124 & 121 \\ 105 & 106 & 109 & 110 & 121 & 122 & 125 & 126 \\ 106 & 107 & 110 & 111 & 122 & 123 & 126 & 127 \\ 107 & 108 & 111 & 112 & 123 & 124 & 127 & 128 \\ 108 & 105 & 112 & 109 & 124 & 121 & 128 & 125 \\ 113 & 114 & 117 & 118 & 129 & 130 & 133 & 134 \\ 114 & 115 & 118 & 119 & 130 & 131 & 134 & 135 \\ 115 & 116 & 119 & 120 & 131 & 132 & 135 & 136 \\ 116 & 113 & 120 & 117 & 132 & 129 & 136 & 133 \\ 117 & 118 & 121 & 122 & 133 & 134 & 137 & 138 \\ 118 & 119 & 122 & 123 & 134 & 135 & 138 & 139 \\ 119 & 120 & 123 & 124 & 135 & 136 & 139 & 140 \\ 120 & 117 & 124 & 121 & 136 & 133 & 140 & 137 \\ 121 & 122 & 125 & 126 & 137 & 138 & 141 & 142 \\ 122 & 123 & 126 & 127 & 138 & 139 & 142 & 143 \\ 123 & 124 & 127 & 128 & 139 & 140 & 143 & 144 \\ 124 & 121 & 128 & 125 & 140 & 137 & 144 & 141 \end{bmatrix} \tag{79}$$

3.2 Mesh Quality

In order for the integrals in the weak form to remain finite when transformed to the master element, the Jacobian of the transformation must be positive. For a 3-D implementation, this means that $\mathcal{J} \equiv \det(\mathbf{F}) > 0$. A mesh has poor quality if it has elements with negative or zero Jacobians. This can occur for nonconvex

elements, where some region of the master domain is mapped outside of the element when transforming back to the physical domain, or when the node numbering is performed incorrectly within an element such that the element is mapped inside-out. The Jacobian is not necessarily constant over an element, and it is required that the Jacobian be positive everywhere in the element - if the Jacobian were allowed to be negative over some regions of the element, but positive over others, this could cause singularities in the stiffness matrix.

4 Computational Cost

The cost of a 3-D finite element simulation of a vector-valued equation such as the equation of linear elasticity results in element stiffness matrices of size 24×24 when using trilinear shape functions. For a cubical mesh, with $M \times M$ elements, there are $3(M+1)^3$ total unknowns in the mesh, so the global stiffness matrix is of size $(3(M+1)^3) \times (3(M+1)^3)$. If the symmetry of the local stiffness matrices is taken into account (they will always be symmetric for linear differential equations so long as the Bubnov-Galerkin approach is used), then the required storage per element drops from 576 to 300.

If the Conjugate Gradient (CG) method is used to solve the matrix system $\mathbf{Ka} = \mathbf{R}$, then this represents repeated application of a 24×24 matrix and a 24×1 vector, which is an $\mathcal{O}(N)$ operation. This is performed for each iteration, so the cost for the CG method scales as $\mathcal{O}(IN)$, where I is the number of iterations needed to reach a particular tolerance in the solve.

The cost of a solve refers to both the storage required and the number of floating point operations required. There are three ways to perform the storage for a mesh consisting of $M \times M \times M$ linear elements:

1. Direct storage - all zeros are stored, and no shortcuts are made by saving element-by-element. This requires $(3(M+1)^3) \times (3(M+1)^3) \approx 9M^6$
2. Element-by-element storage: no zeros are stored. This requires $24 \times 24 \times M^3 = 576M^3$
3. Element-by-element storage, taking advantage of the symmetry of the element stiffness matrices. This requires $300M^3$

So, the storage scales cubically when using element-by-element storage as opposed to direct storage. The number of floating point operations is significantly reduced when using an iterative solver. The total number of unknowns is $3(M+1)^3$, so the ratio of the direct solver (Gaussian elimination) to the CG method is:

$$\frac{\mathcal{O}(N^3)}{I\mathcal{O}(N)} = \frac{\left(3(M+1)^3\right)^2}{I} = \frac{9(M+1)^6}{I} \tag{80}$$

where $N = 3(M+1)^3$ is the total number of unknowns. For $N_t \times N_c \times N_\theta$ elements, there are $(N_t + 1)N_c(N_\theta + 1)$ total nodes. So, if there are 3 unknowns per node, and if a CG solver takes $I\mathcal{O}(N)$ operations, then the total number of operations required is:

$$\text{Operations with CG method} = 3I(N_t+1)N_c(N_\theta+1) \tag{81}$$

where I is the number of iterations.

5 Appendix

This section contains the complete code used in this assignment.

5.1 `MeshGenerator.m`

This program generates the mesh and connectivity matrix.

```
% Mesh generator, ME 280a HW 5
clear all
```

```
Nt = 3;                                 % number of layers
No = 8;                                 % number of elements in theta
Nc = 4;                                 % number of elements in circum

if mod(No, 2) ~= 0
    disp('No must be even!')
end

num_nodes_per_elem = 8;                 % linear elements

R = 1;                                  % radius of each arch
r = 0.3;                                % radius of inner hole
t = 0.2;                                % thickness of the tube wall

layer_thickness = t / (Nt);             % thickness of each ring
angle = (2*pi) / Nc;                    % angle in horizontal plane



% global node numbering begins on the inner surface, and moves clockwise
% until reaching the outer surface

% each row represents one coordinate of a global node
coordinates = zeros(Nt * Nc, 3);
Angle = pi / (No / 2);

k = 1; % index for coordinate row
x = 0;
y = 0;
z = 0;

Theta = 0;  % angle in each plane

% find the x-coordinates of the No + 1 slices
x_centers = zeros(1, No + 1);
x_centers(1) = x;

theta_inc = pi / (No / 2);
theta = theta_inc;

% in the first half of the tube
for i = 2:((No/2) + 1)
    x_centers(i) = x_centers(1) + (R + t + r) * (1 - cos(theta));
    theta = theta + theta_inc;
end

% in the second half of the tube
j = i + 1;
second_part_start = x_centers(i);
for l = 2:((No/2) + 1)
    x_centers(j) = second_part_start + x_centers(l);
    j = j + 1;
end
```

```
% create a vector of the y-coordinates
z_centers = zeros(1, No + 1);
z_centers(1) = z;
theta = theta_inc;

% in the first half of the tube
for i = 2:((No/2) + 1)
    z_centers(i) = (R + t + r) * sin(theta);
    theta = theta + theta_inc;
end

% in the second half of the tube
j = i + 1;
for l = 2:((No/2) + 1)
    z_centers(j) = - z_centers(l);
    j = j + 1;
end



% mesh in the theta direction
for l = 1:(No + 1)

    % for each plane
    theta = 0;
    dt = 0;

    % meshes in a plane perpendicular to tube axis
    for j = 1:(Nt + 1) % create all layers of rings
        for i = 1:Nc % create a single ring

            % x-coordinate
            coordinates(k, 1) = x_centers(l) + (r + dt) * cos(theta);

            % compute tilting parameters
            w = sin(pi/2 - Theta) * ((r + dt) * cos(theta)) / sin(pi/2);
            h = w * sin(Theta);
            p = w * cos(Theta);

            % y-coordinate
            coordinates(k, 2) = y + (r + dt) * sin(theta);

            % z-coordinate
            coordinates(k,3) = z_centers(l);

            % tilt for each half of the tube
            if l > No/2
                sn = -1;
            else
                sn = 1;
            end
```

```
            % tilt the z-coordinate for off-symmetric planes
            if find([1, 2, 8], i)
                coordinates(k,3) = coordinates(k,3) - sn*h;
            else
                coordinates(k,3) = coordinates(k,3) + sn*h;
            end

            % tilt the symmetric planes (the peaks)
            if (l == 3) || (l == 7)
                if find([1, 2, 8], i)
                    coordinates(k,3) = coordinates(k,3) - ((r + dt) * cos(
                        ↪ theta));
                else
                    coordinates(k,3) = coordinates(k,3) + ((r + dt) * cos(
                        ↪ theta));
                end
                coordinates(k,1) = x_centers(l);
            end

            % tilt the x-coordinate
            if find([1, 2, 8], i)
                coordinates(k,1) = coordinates(k,1) + p;
            else
                coordinates(k,1) = coordinates(k,1) - p;
            end

            k = k + 1;
            theta = theta + angle;
        end
        dt = dt + layer_thickness; % reset radius
        theta = 0; % reset angle
    end

    % move the angle (along length) and centers for the next plane
    Theta = Theta + Angle;
    x = x + (R + t + r) * (1 - cos(Theta));
    y = y;
    z = z + (R + t + r) * sin(Theta);
end



X = coordinates(:,1);
Y = coordinates(:,2);
Z = coordinates(:,3);

% scatter3(X, Y, Z)
% span = (max(X)-min(X))/2;
% xlim([min(X), max(X)])
% zlim([-span, span])
% xlabel('x')
% ylabel('y')
```

```
% zlabel('z')

% hold on
% plot3(X, Y, Z, 'k-')
% span = (max(X)-min(X))/2;
% xlim([min(X), max(X)])
% zlim([-span, span])
% xlabel('x')
% ylabel('y')
% zlabel('z')


% generate the connectivity matrix
num_elem = No * Nc * Nt;
LM = zeros(num_elem, num_nodes_per_elem);

% apply in a single slice
j = 1;
k = 1;
e = 1;
for l = 1:(No + 1) % for each slice, assign the front node values
    for elem = e:(e + num_elem / No - 1) % for each element in the slice
        LM(elem, 1) = j;
        LM(elem, 3) = j + Nc;

        if (mod(elem, Nc) == 0)
            LM(elem, 2) = k;
            LM(elem, 4) = k + Nc;
            k = k + Nc;
        else
            LM(elem, 2) = j + 1;
            LM(elem, 4) = j + Nc + 1;
        end
        j = j + 1;
    end

    % update for next slice frontal values
    k = k + Nc;
    j = j + Nc;
    e = e + Nc * Nt;
end

% print the LM for the case before relating front to back nodes
%for e = 1:length(LM)
%    fprintf('%i & %i & %i & %i & %i & %i & %i & %i\\\\\n', LM(e, 1), LM(e,2),
    ↪ LM(e,3), LM(e,4), LM(e,5), LM(e,6), LM(e,7), LM(e,8))
%end

% assign the back face values - front values for second slice are
% back values for first slice, etc.
i = 1;
for j = 1:No
    for elem = i:(i + Nc*Nt - 1)
        LM(elem, (Nc + 1):end) = LM(elem + Nc * Nt, 1:Nc);
```

```
    end
    i = i + Nc*Nt;
end

% delete the unecessary last "chunk" in the LM, since the last slice
% has the back values determined as if they were the frontal values of
% another slice
LM = LM(1:num_elem, :);

% print the final LM
% for e = 1:length(LM)
%     fprintf('%i & %i & %i & %i & %i & %i & %i & %i\\\\\n', LM(e, 1), LM(e,2),
    ↪ LM(e,3), LM(e,4), LM(e,5), LM(e,6), LM(e,7), LM(e,8))
% end
```