Part A: Shape Functions for Linear Tetrahedral Finite Elements

Problem 1

Start with the continuous definition of the potential energy for a linearly elastic object given by:

$$E = \frac{1}{2}\int_{\Omega} \epsilon : C : \epsilon d\Omega$$

For linear elasticity we can rewrite this using the

$$C = \frac{Y}{(1+\mu)\cdot(1-2\mu)}\begin{pmatrix} 1-\mu & \mu & \mu & 0 & 0 & 0 \\ \mu & 1-\mu & \mu & 0 & 0 & 0 \\ \mu & \mu & 1-\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}(1-2\mu) & 0 & 0 \\ 0 & & 0 & 0 & \frac{1}{2}(1-2\mu) & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}(1-2\mu) \end{pmatrix}$$

$$\epsilon = \begin{pmatrix} \epsilon_{xx} & \epsilon_{yy} & \epsilon_{zz} & \epsilon_{yz} & \epsilon_{xy} & \epsilon_{xx} \end{pmatrix}^T \qquad \text{and} \qquad \epsilon_{ij} = \frac{1}{2}\left(\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}_j} + \frac{\partial \mathbf{u}_j}{\partial \mathbf{x}_i}\right)$$

Here $\mathbf{u} \in \mathbb{R}^3$ is the displacement at a point $\mathbf{x}$ in space. Y is the stiffness of the material (Young's modulus) and μ is the poisson's ratio which determines how incompressible a material is. Start by rewriting E using the matrix C and the vector definition of ϵ.

$$E = \frac{1}{2}\int_{\Omega} \epsilon^t C \epsilon d\Omega$$

Problem 2

Next you will discretize E using linear finite element shape functions on a tetrahedron. The shape functions for a linear, tetrahedral element are the barycentric coordinates of each tetrahedron the formula here Let's use barycentric coordinates to represent the displacement of all points inside a tetrahedron: $\mathbf{u}(\mathbf{x}) = \mathbf{u}_1\lambda_1 + \mathbf{u}_2\lambda_2 + \mathbf{u}_3\lambda_3 + \mathbf{u}_4\lambda_4$, where $\mathbf{u}_i$ is the displacement of the i^{th} vertex and λ_i is the i^{th} barycentric coordinate. You should be able to build a discrete version of ϵ using this formula. Try to express your formula in the form $B\mathbf{u}$, where B is a matrix and $\mathbf{u}$ is the 12×1 vector of stacked nodal displacements: $\begin{pmatrix} u_{1x} & u_{1y} & u_{1z} & \cdots & u_{4x} & u_{4y} & u_{4z} \end{pmatrix}^T$
Consider a linear system represent coordinate transformation from barycentric coordinate $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$ to cartesian coordinate (x, y, z) for a point $\mathbf{x}$ inside an element

$$\begin{pmatrix} 1 \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{pmatrix}\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{pmatrix}$$

We can derive the inverse transformation by inverting the matrix

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{pmatrix} = \frac{1}{6V}\begin{pmatrix} 6V_1 & a_1 & b_1 & c_1 \\ 6V_2 & a_2 & b_2 & c_2 \\ 6V_3 & a_3 & b_3 & c_3 \\ 6V_4 & a_4 & b_4 & c_4 \end{pmatrix}\begin{pmatrix} 1 \\ x \\ y \\ z \end{pmatrix}$$

From these relationships, we can derive partial derivatives with respect to the coordinates

$$\frac{\partial \lambda_i}{\partial x} = \frac{1}{6V}a_i \qquad \frac{\partial \lambda_i}{\partial y} = \frac{1}{6V}b_i \qquad \frac{\partial \lambda_i}{\partial z} = \frac{1}{6V}c_i$$

We have the following formula for computing partial derivatives of displace with respect to nodal displacement at the vertices. As an example, we can compute $\frac{\partial \mathbf{u}_x}{\partial x}$

$$\begin{aligned} \frac{\partial \mathbf{u}_x}{\partial x} &= u_{1x}\frac{\partial \lambda_1}{\partial x} + u_{2x}\frac{\partial \lambda_2}{\partial x} + u_{3x}\frac{\partial \lambda_3}{\partial x} + u_{4x}\frac{\partial \lambda_4}{\partial x} \\ &= \frac{1}{6V}\left(a_1u_{1x} + a_2u_{2x} + a_3u_{3x} + a_4u_{4x}\right) \end{aligned}$$

We can find B such that $\epsilon = B\mathbf{u}$ where

$$B = \frac{1}{6V}\begin{pmatrix} a_1 & 0 & 0 & a_2 & 0 & 0 & a_3 & 0 & 0 & a_4 & 0 & 0 \\ 0 & b_1 & 0 & 0 & b_2 & 0 & 0 & b_3 & 0 & 0 & b_4 & 0 \\ 0 & 0 & c_1 & 0 & 0 & c_2 & 0 & 0 & c_3 & 0 & 0 & c_4 \\ 0 & \frac{1}{2}b_1 & \frac{1}{2}c_1 & 0 & \frac{1}{2}b_2 & \frac{1}{2}c_2 & 0 & \frac{1}{2}b_3 & \frac{1}{2}c_3 & 0 & \frac{1}{2}b_4 & \frac{1}{2}c_4 \\ \frac{1}{2}a_1 & 0 & \frac{1}{2}c_1 & \frac{1}{2}a_2 & 0 & \frac{1}{2}c_2 & \frac{1}{2}a_3 & 0 & \frac{1}{2}c_3 & \frac{1}{2}a_4 & 0 & \frac{1}{2}c_4 \\ \frac{1}{2}a_1 & \frac{1}{2}b_1 & 0 & \frac{1}{2}a_2 & \frac{1}{2}b_2 & 0 & \frac{1}{2}a_3 & \frac{1}{2}b_3 & 0 & \frac{1}{2}a_4 & \frac{1}{2}b_4 & 0 \end{pmatrix}$$

Therefore we can express potential energy for one element as follows

$$\begin{aligned} E &= \frac{1}{2}\int_\Omega \epsilon^t C \epsilon d\Omega \\ &= \frac{1}{2}\int_\Omega \mathbf{u}^t B^t C B \mathbf{u} d\Omega \\ &= \frac{1}{2}\mathbf{u} V B^t C B \mathbf{u} \end{aligned}$$

where we assume material dependent variables, i.e. Y and μ, are constant inside an element in last step

Part B: Implement Linear Shape Functions

1. Open 'SRC_DIRECTORY/A4/include/ShapeFunctionTetLinearA4.h'.
2. The function `*double phi(double x) { ... }` returns the value of the linear shape function λ_{VERTEX}, evaluated at $\mathbf{x}$. Implement this function.
3. The function `*std::array<DataType,3> dphi(double x) { ... }` returns the gradient linear shape function $\nabla\lambda_{VERTEX}$, evaluated at $\mathbf{x}$. Implement this function.
4. The function `*Eigen::Matrix<DataType, 3,3> F(double x, const State &state)` returns the 3×3 deformation gradient for a tetrahedral element where $F_{ij} = \frac{\partial \mathbf{u}_i}{\partial \mathbf{X}_j}$. Use **dphi** to implement this method.

Part C: Implement Quadrature and Simulate

1. Open 'SRC_DIRECTORY/A4/include/QuadratureLinearElasticity.h'
2. Complete the **getEnergy** function to return the value of E from Part A.
3. Compute the stiffness matrix and the forces from E. The forces are the negative gradient of E wrt to the degrees of freedom, $\mathbf{u}$ and the stiffness matrix is the hessian matrix, i.e $K_{ij} = -\frac{\partial^2 E}{\partial \mathbf{u}_i \partial \mathbf{u}_j}$. Derive these terms and complete the **getGradient** and **getHessian** functions.
4. Run the assignment code which will compute the stress on a test object and plot the result.