Assignment 4: The Finite Element Method

Due: October 29th, 11:59pm

Introduction

This assignment will give you the opportunity to implement the important pieces of the finite element method (FEM) discussed in lectures. The starter code contains two important files. The first, `{SRC_DIRECTORY}/A4/include/ShapeFunctionTetLinearA4.h`, is an incomplete implementation of the barycentric shape functions for a tetrahedaral finite element. The second, `{SRC_DIRECTORY}/A4/include/QuadratureLinearElasticity.h`, is an incomplete implementation of the discrete linearly elastic energy (the integrated energy derived in the previous lecture). You will be responsible for the following:

1. The derivation of the internal forces and stiffness matrix for a linearly elastic constitutive model discretized using a tetrahedral element (as shown in lecture)
2. An implementation of the shape functions, gradients and the deformation gradient for a linear tetrahedral element.
3. An implementation of the per-element, exact quadrure rule for this element.

Setup Assignment Code Base

This should be done in a similar manner to all previous assignments

1. Run 'cd {SRC_DIRECTORY}/A4'
2. Run 'mkdir ./build'
3. Run 'cd ./build'
4. Run 'cmake .. -DCMAKE_BUILD_TYPE=RELEASE' to build the project Makefile.
5. Run 'make -j8'
6. This should build an execubtable named **./bin/CompFabA4**.

Part A: Shape Functions for Linear Tetrahedral Finite Elements. [6 marks]

1. Start with the continuous definition of the potential energy for a linearly elastic object given by: $E=\frac{1}{2}\int_\Omega \epsilon\colon C \colon\epsilon d\Omega$. For linear elasticity we can rewrite this using the

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

   $\epsilon= \begin{pmatrix} \epsilon_{xx} & \epsilon_{yy} & \epsilon_{zz} &\epsilon_{yz} & \epsilon_{xy} & \epsilon_{xx} \end{pmatrix}^T$ and $\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 $\mathbf{u} \in \mathcal{}^3$ is the displacement at a point $\mathbf{x}$ in space. Y is the stiffness of the material 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. [2 marks]

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 for which is shown 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: $[u1_x u1_y u1_z ... u4_x u4_y u4_z]^T$. [2 marks]

3. Taking Ω to be the volume of single tetrahedron, right the discrete formula for E. **Include this derivation in the assignment write up**. [2 marks]

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 [7 marks]

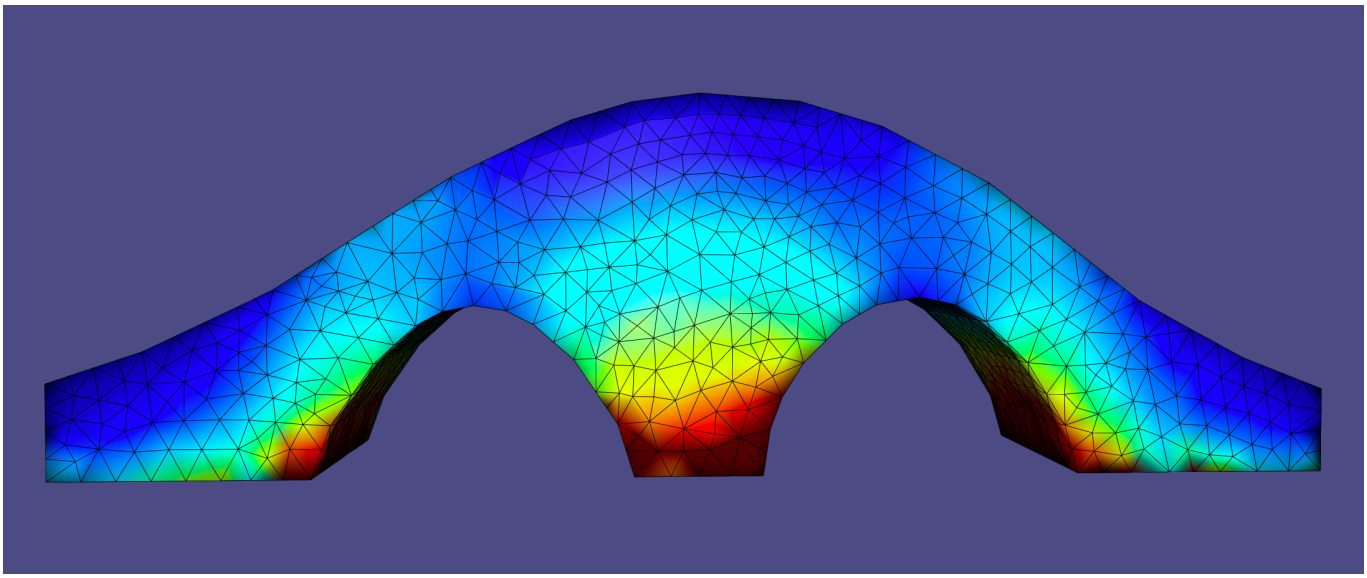

Fig. 1: Expected output for the assignment code once the shape function and quadrature rule have been implemented correctly.

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.

Hand In [1 mark]

Collect all required images into a PDF report which must include your full name and student number. Submit this PDF and a zip file containt your A4 source code via **email** to **diwlevin@cs.toronto.edu**. The subject of the email must be **CompFabA4_LASTNAME_STUDENTNUMBER**.