1 Hexahedral Finite Elements

Figure 1 shows a hexahedral element in its rest shape. We use $\mathbf{X}_i$ to denote its vertex positions in its reference space. $\mathbf{X}_i$ is specified before the simulation and remains constant. Following the notation conventions, we use $\mathbf{x}_i$ to denote the deformed state of the cube as shown in Figure 2.

The elastic behavior of the cube is governed by a non-negative scalar function called the strain energy function $\Psi(\mathbf{x})$. At rest shape $\Psi(\mathbf{x})$ takes the minimum value 0. When forces are applied on vertecies of the cube, the cube would deform to minimize the total potential energy defined as

$$V = \Psi(\mathbf{x}) - \sum_{i=1}^{8} \mathbf{x}_i \cdot \mathbf{f}_i, \tag{1}$$

where $\mathbf{f}_i$ is a force vector applied to vertex i.

1.1 Material Model

Instead of directly using $\mathbf{x}$, we define the strain energy functions in terms of deformation gradients $\mathbf{F}_j$ sampled at a small number of quadrature points $\mathbf{X}_j$. The deformation gradients depend on $\mathbf{x}$.

$$\Psi(\mathbf{x}) = \sum_j \Psi(\mathbf{F}_j(\mathbf{x})).$$

For example, using 3-dimensional 2-point Gaussian Quadrature rule, we have 8 sample points in the reference space of the cube: $\mathbf{X}_j = (\pm\sqrt{\frac{1}{3}}, \pm\sqrt{\frac{1}{3}}, \pm\sqrt{\frac{1}{3}})$. Subscript j is used denote a quadrature point and i is used to denote a cube vertex.

To compute deformation gradient $\mathbf{F}_j$ at $\mathbf{p}_j$, we need to know how the volume has deformed near $\mathbf{p}_j$. Let $\mathbf{u}_i = \mathbf{x}_i - \mathbf{X}_i$ be the displacement for vertex i, and $\mathbf{u}_j$ be the displacement for $\mathbf{X}_j$. We use trilinear interpolation to define displacement $\mathbf{u}_j$ by

$$\mathbf{u}_j = \frac{1}{8}\sum_{i=1}^{8}(1 + w_{i,1}\mathbf{X}_j^x)(1 + w_{i,2}\mathbf{X}_j^y)(1 + w_{i,3}\mathbf{X}_j^z),$$

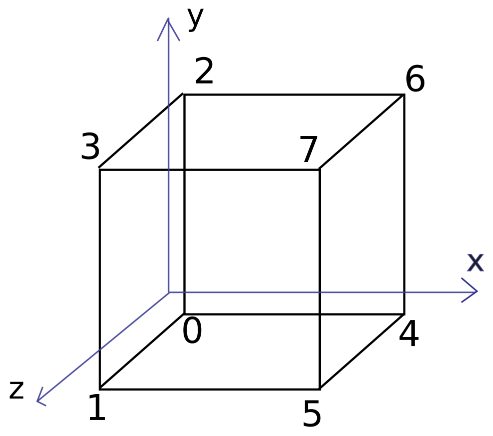

Figure 1: A hexahedral element with its vertex ordering used in this document and our code.

$$w = \begin{pmatrix} -1 & -1 & -1 \\ -1 & -1 & 1 \\ -1 & 1 & -1 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \\ 1 & 1 & 1 \end{pmatrix}.$$

$$F_j = \frac{d\mathbf{u}_j}{d\mathbf{X}_j} = \mathbf{I} + \sum_{i=1}^{8} \nabla_{\mathbf{X}} N_i(\mathbf{X}_j)\mathbf{u}_i,$$

$$\nabla_{\mathbf{X}} N_i(\mathbf{X}_j) = \frac{1}{4} \begin{pmatrix} \frac{w_{i,0}}{\mathbf{X}_7^x - \mathbf{X}_1^x}(1 + w_{i,1}\mathbf{X}_j^y)(1 + w_{i,2}\mathbf{X}_j^z) \\ \frac{w_{i,1}}{\mathbf{X}_7^y - \mathbf{X}_1^y}(1 + w_{i,0}\mathbf{X}_j^x)(1 + w_{i,2}\mathbf{X}_j^z) \\ \frac{w_{i,2}}{\mathbf{X}_7^z - \mathbf{X}_1^z}(1 + w_{i,0}\mathbf{X}_j^x)(1 + w_{i,1}\mathbf{X}_j^y) \end{pmatrix}^T.$$

A simple and robust material model is called the neo-hookean material model and it defines the strain energy function as

$$\Psi(\mathbf{F}) = \frac{\mu}{2}(I_1 - 3) - \mu J + \frac{\lambda}{2} J^2,$$

$$I_1 = tr(\mathbf{F}^T\mathbf{F}),\ J = \log(\det(\mathbf{F})).$$

μ and λ are shear modulus and first Lamé constant of the material.

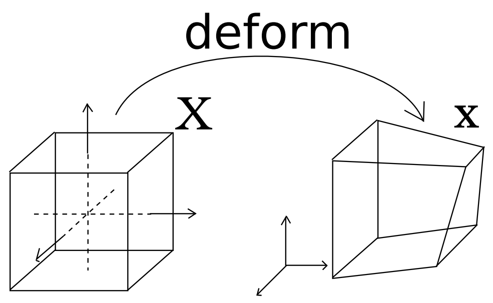

Figure 2: A hexhedral element in its reference state and deformed state. The reference state is drawn in its natural coordinate system.

2 Solving for Statics

We want to minize the potential energy function in Equation 1. An easy first step is to take the derivative of the function and walk in the opposite direction of the gradient:

$$\frac{\partial V}{\partial \mathbf{x}} = \frac{\partial \Psi}{\partial \mathbf{x}} - \mathbf{f}. \tag{2}$$

The second term of the gradient tells us the moving vertices along the directions of $\mathbf{f}$ can reduce the potential energy caused by these forces. The first term can be expanded using chain rule

$$\frac{\partial \Psi}{\partial \mathbf{x}_i} = \sum_j \frac{\partial \Psi}{\partial \mathbf{F}_j} \frac{\partial \mathbf{F}_j}{\partial \mathbf{x}_i} \tag{3}$$

$$= \mathbf{P}(\mathbf{F}) \nabla N_i(\mathbf{X}_j). \tag{4}$$

$\mathbf{P}(\mathbf{F})$ is a material specific function known as the first Piola-Kirchoff stress. For Neo-hookean model,

$$\mathbf{P}(\mathbf{F}) = \mu(\mathbf{F} - \mathbf{F}^{-T}) + \lambda J F^{-T}.$$

To improve convergence, we use newton's method to minize the potential energy V. The second derivative of V is called the stiffness matrix $\mathbf{K}$. $\mathbf{K}$ only depends on $\Psi(\mathbf{F})$. Taking derivative of $\mathbf{P}$ with respect to $\mathbf{F}$ results in a third order tensor. In a particular direction of $\mathbf{F}$ denoted as $\delta\mathbf{F}$, the amount of change in $\mathbf{P}$ is

$$\delta\mathbf{P}(\mathbf{F};\delta\mathbf{F}) = \mu\delta\mathbf{F} + (\mu - \lambda J)\mathbf{F}^{-T}\delta\mathbf{F}^{T}\mathbf{F}^{-T} + \lambda tr(\mathbf{F}^{-1}\delta\mathbf{F})\mathbf{F}^{-T}.$$

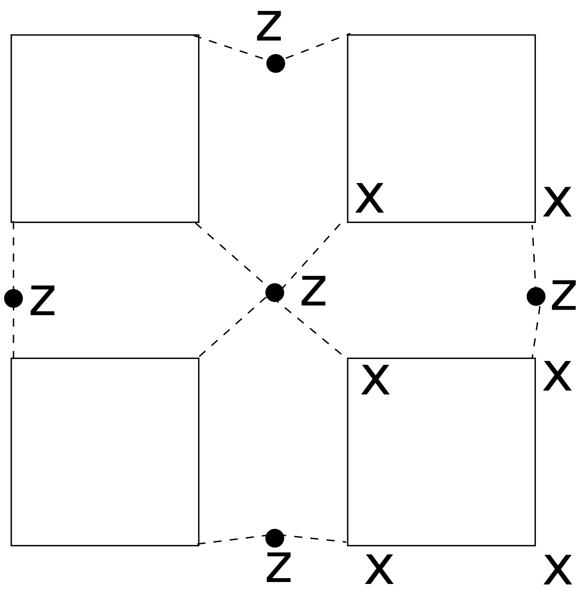

Figure 3: A schematic drawing of ADMM variables. The squares represent elements, whose corners correspond to x. The dots represent consensus variables that tries to bring together the x variables connected to it.

3 ADMM for Statics

Alternating direction of multipliers method (ADMM) allows each element to keep a separate copy of its vertex positions x. It introduces additional variables z called the consensus variables and lagrangian multipliers λ. Suppose there are n_e elements and n_v vertices, then there are $3 \times 8 \times n_e$ x variables, $3 \times n_v$ z variables and the same number of multipliers as the number of x variables. A 2D schematic is shown in Figure 3.

Let V_e be the potential energy for element e. Let x_e be the vertices for element e and z_e for the corresponding consensus variables. ADMM iterates between the following three steps:

$$x_i = \arg\min V_i(x) + \frac{\rho}{2}\|x - z + \lambda\|^2 \tag{5}$$

$$z_i = \frac{1}{n_i} \sum_{x_j \text{ connected to } z_i} (x_j + \lambda_j) \tag{6}$$

$$\lambda_j = \lambda_j + x_j - z_j \tag{7}$$

x updates are the slowest because it solves a statics simulation for each element. However, this step is highly parallizable because the statics equilibrium can be solved independently for each element.

4 Heirachical Simulation(Under construction)

$\mathbf{q}_c$ and $\mathbf{q}_d$ are coarse and detail vertex displacements. $\mathbf{q}_d$ is written in the reference frame of the coarse element. In this case, displacements of cube vertices. $\mathbf{X}$ material coordinates. $\mathbf{u}$ displacement at $\mathbf{X}$.

$$\mathbf{u} = \mathcal{J}_c(\mathbf{q}_c, \mathbf{X} + \mathcal{J}_d(\mathbf{q}_d, \mathbf{X})) \approx \mathcal{J}_c(\mathbf{q}_c, \mathbf{X}) + \mathbf{F}_c \mathcal{J}_d(\mathbf{q}_d, \mathbf{X}).$$

$$\mathcal{J}(\mathbf{q}, \mathbf{X}) = \sum_{i=1}^{8} N(\mathbf{X})\mathbf{q}_i.$$

The deformation gradient is

$$\mathbf{F}(\mathbf{q}, \mathbf{X}) = \mathbf{I} + \frac{\partial \mathcal{J}(\mathbf{q}, \mathbf{X})}{\partial \mathbf{X}} = \mathbf{I} + \sum_i \mathbf{q}_i \nabla N^T(\mathbf{X}).$$

$$\begin{aligned}\mathbf{F}_e = \mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}} &= \mathbf{I} + \frac{\partial \mathcal{J}_c}{\partial \mathbf{X}} + \frac{\partial \mathcal{J}_d}{\partial \mathbf{X}} + \frac{\partial \mathcal{J}_c}{\partial \mathbf{X}}\frac{\partial \mathcal{J}_d}{\partial \mathbf{X}} + \frac{\partial^2 \mathcal{J}_c}{\partial \mathbf{X}^2}\mathcal{J}_d \\ &= \mathbf{F}_c\mathbf{F}_d + \frac{\partial^2 \mathcal{J}_c}{\partial \mathbf{X}^2}\mathcal{J}_d \approx \mathbf{F}_c\mathbf{F}_d.\end{aligned}$$

The strain energy can be defined in the same way in terms of $\mathbf{F}$. The problem is that there isn't a unique minimizer given forces and constraints.

Embed quadrature points $\mathbf{X}_i$ in fine cubes. Define strain energy density Ψ at each quadrature point as usual in terms of $\mathbf{F}_e(\mathbf{q}, \mathbf{X}_i)$. Minimize potential energy

$$\arg\min_{\mathbf{q}} V = \sum_i \Psi(\mathbf{F}_e(\mathbf{q}, \mathbf{X}_i)) - \mathbf{f}_{ext} \cdot \mathbf{q}.$$

Need to distribute stress to $\mathbf{q}_c$ and $\mathbf{q}_d$ so that we don't double count forces. There are 24 extra degrees of freedom from the coarse cube.

Here are some desirable constraints for the distribution of forces and displacements specific to our arrangement of elements.

- Project internal force to the coarse degrees of freedom. The coarse displacements handle as much stress as possible. Using chain rule, force on a coarse vertex $\mathbf{q}_i$ is

$$\mathbf{f}_{ci} = \frac{\partial \Psi(\mathbf{F})}{\partial \mathbf{q}_i} = \sum_{j=1}^{64} \mathbf{P}(\mathbf{F}_j) w_j \mathbf{F}_d^T \nabla N_i(\mathbf{X}_j),$$

where $\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}}$ is the first Piola-Kirchhoff stress and w_j are quadrature weights. Similarly for a fine vertex,

$$\mathbf{f}_{di} = \sum_{j=1}^{8} \mathbf{P}(\mathbf{F}_j)\mathbf{F}_c \nabla N_i(\mathbf{X}_j).$$

In gradient descent algorithm, forces are used as displacements for the next step.

We can use $\mathbf{f}_c$ as it is and then compute $\mathbf{P}$ after removing the contribution from $\mathbf{f}_c$.

We can project the fine forces into the row space of $\mathbf{J}_c$ to get f_{dc}, the amount of detail displacements handled by the coarse vertices.

$$\mathbf{J}_c = \frac{\partial \mathcal{J}}{\partial \mathbf{q}} = \begin{pmatrix} N_1(\mathbf{X}) \\ N_2(\mathbf{X}) \\ \dots \\ N_8(\mathbf{X}) \end{pmatrix}.$$

$$\mathbf{f}_c = (\mathbf{J}_c^T \mathbf{J}_c)^{-1} \mathbf{J}_c^T \mathbf{f}_d.$$

$$\mathbf{f}_{dc} = \mathbf{J}\mathbf{f}_c.$$

The displacement left for the fine vertices are

$$\mathbf{f}'_d = \mathbf{f}_d - \mathbf{f}_{dc}.$$

We can use these forces to update the position during each step. For a newton's solver, we can apply the same idea to split the stiffness matrix for coarse and fine displacements. Given the stiffness matrix for the fine vertices $\mathbf{K}_d$,

$$d\mathbf{q}_d = \mathbf{K}_d \mathbf{f}_d.$$

$\mathbf{K}_c = \mathbf{J}_c^T \mathbf{K}_d \mathbf{J}_c$. We can project the displacement to the coarse level to minimize the difference

$$\|\mathbf{K}_c d\mathbf{q}_c - \mathbf{K}_d \mathbf{f}_d\|^2.$$

$$\mathbf{q}_c = \mathbf{K}_c^{-1} \mathbf{J}_c^T \mathbf{K}_d \mathbf{f}_d.$$

The amount of detail displacement removed by coarse displacement is

$$\mathbf{q}_{dc} = \mathbf{J}_c \mathbf{K}_c^{-1} \mathbf{J}_c^T \mathbf{K}_d \mathbf{f}_d.$$

To implement ADMM under this formulation, we can make consensus variables all coarse and fine vertices.

1. Compute $\mathbf{f}_d, \mathbf{K}_d$, $\mathbf{q}_c$,$\mathbf{q}_d$.
2. Update consensus variables.
3. update multipliers.

- A different and more straightforward constraint may be $\mathbf{q}_d = 0$ for fine vertices that overlap with the coarse vertices. The coarse vertices always correspond to the correct fine vertex positions.

  This constaint and the strain energy function can be directly coded into ADMM. We can add consensus variables for coarse vertices and fine vertices that does not overlap with coarse vertices. For fine vertices that does overlap with coarse vertex, we enforce the constraint that that it agrees with the consensus variable for the corresponding coarse vertex.