CMIS Project: Simulating hyper elastic materials

Nikolai Plambech Nielsen
lpk331@alumni.ku.dk
Niels Bohr Institute, University of Copenhagen

1 FINITE VOLUME METHOD FOR HYPERELASTIC MATERIALS

In this assignment we look at simulating the dynamics of hyper elastic materials using the Finite Volume Method (FVM). Where we in hand-in 5 looked at the Cauchy momentum equation with the assumption $\ddot{x} = 0$, we will in this assignment not neglect this term, and instead employ a simple time evolution of the system to find the solution.

The governing equation is

$$\rho_0 \ddot{\mathbf{x}} = \mathbf{b}_0 + \nabla_0 \cdot \mathbf{P} \tag{1}$$

where ρ is the mass density, $\mathbf{b}$ is the body force density (gravity, viscosity), and $\mathbf{P}$ is the first Piola-Kirchhoff stress tensor. A subscript of zero means the function is dependent on material, or undeformed, coordinates, meaning the last term is the divergence of the first Piola-Kirchhoff stress tensor, in material coordinates.

We impose boundary conditions on the system in the form of a nodal traction field, defined by

$$\mathbf{P}\hat{\mathbf{N}} = \mathbf{t} \tag{2}$$

where $\hat{\mathbf{N}}$ is the outward unit normal of the edge, in material coordinates, and $\mathbf{t}$ is a known traction field.

In our discretization we use a triangular mesh as the computational domain, with median dual vertex centred control volumes. We further assume that the deformation gradient is constant over each triangular element, meaning we get a constant 2×2 matrix deformation gradient for each element. Of course, this approximation is not entirely physical, since it potentially introduces discontinuities at every element border, but it simplifies calculations immensely, and in any case, it approaches the correct solution as the density of elements approaches infinity.

We further assume the mass density of the system to be constant, as we work with homogeneous materials, and only one at a time.

1.1 Notation, nomenclature and the control volumes

As with many problems, a good notation is half the battle. As mentioned before, a subscript of 0 (or capital letters for stuff other than stress tensors) denotes material coordinates, whilst lower letters and no subscript denotes spatial coordinates. We denote a element edge on the boundary of the domain as free, if there is no traction field applied to it, and non-free if there is.

The control volume we employ is slightly different than the centroid dual vertex centred control volume from last week. For each vertex i, we join the incenters of each element e the vertex is a part of, to the midpoint between the vertex in question (i), and the other vertices in the element j, k (see figure **ref** for a sample control volume). Using this control volume turns out to be beneficial for this governing equation with our approximations, as we shall soon see.

We denote the line segments between edge midpoints and incenters of element e as S^e_α and S^e_β, and the line segments between the i'th vertex and the midpoint between the i'th and k'th vertex as S^e_{ik}. As such, the union of these four line segments constitutes a closed curve on the domain, going from vertex i to midpoint, to incenter, to midpoint and back to vertex i.

1.2 Discretization

Next we integrate the governing equation over the i'th control volume in material coordinates, giving

$$\int_{A_i} \rho_0 \ddot{\mathbf{x}} \, \mathrm{d}A = \int_{A_i} \mathbf{b}_0 \, \mathrm{d}A + \int_{A_i} \nabla_0 \cdot \mathbf{P} \, \mathrm{d}A \tag{3}$$

On the left hand side we exchange the order of integration and differentiation, which we can do due to the control volume having a fixed area (in material coordinates, that is. It will deform and change in spatial coordinates). Further, with the assumption that ρ_0 is a constant we get

$$\int_{A_i} \rho_0 \ddot{\mathbf{x}} \, \mathrm{d}A = \rho_0 \frac{\mathrm{d}^2}{\mathrm{d}t^2} \int_{A_i} \mathbf{x} \, \mathrm{d}A \tag{4}$$

We then use the midpoint approximation rule and take $\mathbf{x}$ to be the value in the center of the control volume, ie the position of the i'th vertex, giving

$$\int_{A_i} \rho_0 \ddot{\mathbf{x}} \, \mathrm{d}A = \rho_0 A_i \ddot{\mathbf{x}}_i = m_i \ddot{\mathbf{x}}_i \tag{5}$$

where m_i is the "nodal mass", ie the mass attributed to each vertex/control volume. We give a similar treatment to the first term on the right hand side and use the midpoint rule with the value of $\mathbf{b}_0$ at the i'th vertex as the integrand. For the second term we just use the Gauss-divergence theorem to change the surface integral over the divergence, to a closed line integral, and then split up this line integral into piecewise continuous parts. This all gives us

$$m_i \ddot{\mathbf{x}} = f_i^{\text{body}} + \sum_\gamma \int_{S^e_\gamma} \mathbf{P}\hat{\mathbf{N}} \, \mathrm{d}S \tag{6}$$

where $f_i^{\text{body}} = A_i \mathbf{b}_0$ is the total body force on the control volume, and γ runs over all edges of the control volume (α, β for each element).

If S^e_γ is on the boundary of the domain, then it is either free or non-free. In the free case $\mathbf{P}\hat{\mathbf{N}} = \mathbf{t} = 0$ and the integral vanishes. In the non-free case the integral is just $\mathbf{t} l^e_\gamma$, where l^e_γ is the length of S^e_γ ($\mathbf{t}$ is constant over the edge, so goes outside the integral, and the integral of 1 is just the edge length).

With this we can split up the piecewise integral further into boundary edges and inner edges. The integral over inner edges can be calculated without even using the α, β edges if we utilize the closed contour mentioned before. Since we assume that the tensor $\mathbf{P}$ is constant over each element, the integral of a closed contour

on the element is zero (like that of a gradient or any field with a vanishing curl):

$$\oint_S \mathbf{P}^e \hat{\mathbf{N}} \, \mathrm{d}S = 0 \tag{7}$$

Now we let $S = S^e_{ji} \cup S^e_{ik} \cup S^e_\alpha \cup S^e_\beta$, giving

$$\mathbf{f}^e_i = \int_{S^e_\alpha \cup S^e_\beta} \mathbf{P}^e \hat{\mathbf{N}} \, \mathrm{d}S = -\int_{S^e_{ji} \cup S^e_{ik}} \mathbf{P}^e \hat{\mathbf{N}} \, \mathrm{d}S \tag{8}$$

Next we note that $\mathbf{P}^e$ is constant by assumption (the deformation gradient is constant, giving a constant stress tensor), and $\hat{\mathbf{N}}$ is constant along each straight edge, so $\mathbf{f}^e_i$ is given by

$$\mathbf{f}^e_i = -\frac{1}{2}\mathbf{P}^e \hat{\mathbf{N}}^e_{ji} l_{ji} - \frac{1}{2}\mathbf{P}^e \hat{\mathbf{N}}^e_{ik} l_{ik} \tag{9}$$

with $l_{ik} = |\mathbf{x}_k - \mathbf{x}_i|$. Computations can be saved by noting that $\hat{\mathbf{N}}^e_{ji} l_{ji}$ is just the vector $\mathbf{x}_i - \mathbf{x}_j$ rotated by 90 degrees, such that it points outward of the element (counter clockwise). Likewise with $\hat{\mathbf{N}}^e_{ik} l_{ik}$ being $\mathbf{x}_k - \mathbf{x}_i$ rotated 90 degrees counter clockwise (note the order of the indices, first index is starting point, last index is ending point).

Putting all of this together, we get the final equation of motion for the i'th vertex:

$$m_i \ddot{\mathbf{x}}_i = \mathbf{f}^{\text{body}}_i + \mathbf{f}^t_i + \mathbf{f}^E_i, \quad \mathbf{f}^t_i = \sum_\gamma \mathbf{t} l^e_\gamma, \quad \mathbf{f}^E_i = \sum_e \mathbf{f}^e_i \tag{10}$$

where $\mathbf{f}^t_i$ is the total traction force, made up of contributions from each non-free boundary edge, and $\mathbf{f}^E_i$ is the total elastic force, made up of contributions from each element of the i'th control volume (ie, all elements the i'th vertex is a part of).

Now, this is all well and good, but we still have not calculated the first Piola-Kirchhoff stress tensor $\mathbf{P}$. The tensor is given by $\mathbf{P} = \mathbf{FS}$, with $\mathbf{F}$ being the deformation gradient and $\mathbf{S}$ being the second Piola-Kirchhoff sress tensor. This is given by

$$\mathbf{S} = \lambda \operatorname{tr}(\mathbf{E})\mathbf{I} + 2\mu\mathbf{E}, \quad \mathbf{E} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}) \tag{11}$$

where $\mathbf{E}$ is the Green strain tensor and λ, μ are the first and second Lame coefficient, which can be calculated from the Young Modulus E and Poisson Ratio ν by

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\mu)}. \tag{12}$$

The can has been kicked down the road long enough now, we just need the deformation gradient for each element, then we are golden. The deformation gradient is what takes the material coordinates to the spatial coordinates, and since we assume the deformation gradient to be constant, we can find it quite simply. For the e'th element we define $\mathbf{g}^e_{ij} = \mathbf{x}_j - \mathbf{X}_i, \mathbf{G}^e_{ij} = \mathbf{X}_j - \mathbf{X}_i$. These vectors are related by the deformation gradient with

$$\mathbf{g}^e_{ij} = \mathbf{F}^e \mathbf{G}^e_{ij} \tag{13}$$

So if we let $\mathbf{D}^e = [\mathbf{g}^e_{ij} \; \mathbf{g}^e_{ik}], \mathbf{D}^e_0 = [\mathbf{G}^e_{ij} \; \mathbf{G}^e_{ik}]$ we get

$$\mathbf{D}^e = \mathbf{F}^e \mathbf{D}^e_0, \quad \mathbf{F}^e = \mathbf{D}^e (\mathbf{D}^e_0)^{-1} \tag{14}$$

This has the advantage that the matrix inverse is only dependent on the material coordinates, which by definition are constant - so these matrices can be precomputed!

With all of this done we are now in a position where we can calculate the total forces on each vertex for a given time step. All we then need is to update the positions of the vertices to evolve the system in time. For this we employ a semi-implicit first order finite difference approximation.

We let $\mathbf{v}_i = \dot{\mathbf{x}}_i$, giving us a pair of coupled first order ODEs by

$$\dot{\mathbf{v}}_i = \frac{1}{m_i}\mathbf{f}^{\text{total}}_i = \frac{1}{m_i}(\mathbf{f}^{\text{body}}_i + \mathbf{f}^t_i + \mathbf{f}^E_i), \tag{15}$$

$$\dot{\mathbf{x}}_i = \mathbf{v}_i. \tag{16}$$

With the FDM approximation giving

$$\mathbf{v}^{t+\Delta t}_i = \mathbf{v}^t_i + \frac{\Delta t}{m_i}\mathbf{f}^{\text{total}}_i, \tag{17}$$

$$\mathbf{x}^{t+\Delta t}_i = \mathbf{x}^t_i + \Delta t \mathbf{v}^{t+\Delta t}_i. \tag{18}$$

Note that we use an explicit updating scheme for the velocity, but an implicit updating scheme for the position, hence the name "semi-implicit time integration".

To evolve the system in time by one time step, we need to calculate the total forces, which entails computing all deformation gradients, to calculate all element stress tensors. With the total force on the i'th vertex calculated, we can then update the position and velocity of this vertex.

Note that this that all coupling between elements occur in computing the total forces (the elastic forces, actually), and we do not need to perform any matrix assembly and solving, in contrast to all the other weeks assignments (except for semi-lagrangian explicit time integration).