Assignment 7

CASPER BRESDAHL, whs715, University of Copenhagen, Denmark

1 INTRODUCTION

This week we will look at how we can build a simulator for elastic objects. We will begin by applying the Finite Volume Method to the Cauchy equation and derive update rules for our simulation. We will then look at two experiments, the first looking into how we can compute an upper bound for the time steps we can take for different materials, and the second looking into how our a simulation changes based on grid resolution.

2 THEORY

2.1 Derivation of Cauchy equation

This week we are dealing with the Cauchy equation, also known as the motion equation. We will define it in terms of material coordinates which gives us $\rho_0 \ddot{\mathbf{x}} = \mathbf{b}_0 + \nabla_{\mathbf{X}} \cdot \mathbf{P}$ where ρ_0 is the mass density in material coordinates, $\ddot{\mathbf{x}}$ is the spatial acceleration, $\mathbf{b}_0$ is the body force densities and $\nabla_{\mathbf{X}} \cdot \mathbf{P}$ is the divergence with respect to the material coordinates of the stress tensor. We can now apply the Finite Volume Method to our partial differential equation. The first step is to take the volume integral over our equation. We let A_i denote the control volume of the *i'th* vertex and get:

$$\int_{A_i} \rho_0 \ddot{\mathbf{x}}_i dA = \int_{A_i} \mathbf{b}_0 dA + \int_{A_i} \nabla_{\mathbf{X}} \cdot \mathbf{P} dA$$

We would now like to simplify the term $\int_{A_i} \rho_0 \ddot{\mathbf{x}}_i dA$ by applying a midpoint approximation, but because $\ddot{\mathbf{x}}_i$ depends on time we first need to apply Leibniz' rule to move the time derivative outside the integral. We can then apply the midpoint approximation by simply removing the integral and multiplying with A_i. As we now have both the area and density we can combine these terms into a single term of mass, m_i. As the mass does not depend on time we are left with:

$$m_i \ddot{\mathbf{x}}_i = \int_{A_i} \mathbf{b}_0 dA + \int_{A_i} \nabla_{\mathbf{X}} \cdot \mathbf{P} dA$$

For the term $\int_{A_i} \mathbf{b}_0 dA$ we can directly apply the midpoint approximation and get:

$$m_i \ddot{\mathbf{x}}_i = \mathbf{b}_0 A_i + \int_{A_i} \nabla_{\mathbf{X}} \cdot \mathbf{P} dA$$

For the last term $\int_{A_i} \nabla_{\mathbf{X}} \cdot \mathbf{P} dA$ we apply Gauss-divergence theorem to convert the volume integral to a surface integral:

$$m_i \ddot{\mathbf{x}}_i = \mathbf{b}_0 A_i + \int_S \mathbf{P}\mathbf{N} dS$$

We can now exploit that our surfaces are piecewise continuous and integrate over each edge piece:

$$m_i \ddot{\mathbf{x}}_i = \mathbf{b}_0 A_i + \sum_e \sum_{\gamma \in \{\alpha, \beta\}} \int_{S_\gamma^e} \mathbf{P}\mathbf{N} dS$$

Author's address: Casper Bresdahl, whs715, University of Copenhagen, Copenhagen, Denmark, whs715@alumni.ku.dk.

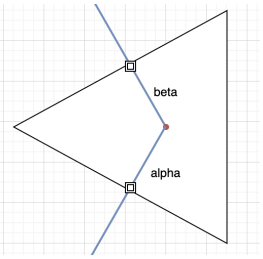

Fig. 1. A single triangle in the mesh (in black) with the control volume (in blue) going through it. We note the control volume goes from the midpoints on the triangle mesh to the center of triangles. Because of this we can split the control volume surface in two pieces, α and β.

Where e denotes our edges in the control volumes. Due to the choice of using median dual vertex centered control volumes, we can split the control volume edges into two, α and β. This is illustrated in Figure 1.

We can now apply boundary conditions. Here we split the term $\int_{S_\gamma^e} \mathbf{P}\mathbf{N} dS$ into three boundaries terms. The first one being the free boundaries where no traction is applied, and is given by $\mathbf{t} = \mathbf{P}\mathbf{N} = 0$ and thus giving us:

$$\int_{S_\gamma^e} \mathbf{P}\mathbf{N} dS = 0$$

The second type being the non-free boundaries, where we apply some known amount of traction giving us:

$$\int_{S_\gamma^e} \mathbf{P}\mathbf{N} dS = \mathbf{t} l_\gamma^e$$

where l_γ^e is the length of the boundary piece. And the last one being the internal boundaries which we 'pull out'. We thus split the summation into two (the term for the free boundaries simply disappear):

$$m_i \ddot{\mathbf{x}}_i = \mathbf{b}_0 A_i + \sum_{e_{traction}} \mathbf{t} l_\gamma^{e_{traction}} + \sum_{e_{inner}} \sum_{\gamma \in \{\alpha, \beta\}} \int_{S_\gamma^{e_{inner}}} \mathbf{P}\mathbf{N} dS$$

To simplify a little, we can define $\mathbf{f}_i^{ext} = \mathbf{b}_0 A_i$ as this models the external forces and $\mathbf{f}_i^t = \sum_{e_{traction}} \mathbf{t} l_\gamma^{e_{traction}}$ as this models the traction forces. We then have:

$$m_i \ddot{\mathbf{x}}_i = \mathbf{f}_i^{ext} + \mathbf{f}_i^t + \sum_{e_{inner}} \sum_{\gamma \in \{\alpha, \beta\}} \int_{S_\gamma^{e_{inner}}} \mathbf{P}\mathbf{N} dS$$

We can now look at the deformation gradient. To go from spatial coordinates to material coordinates we have the relation $\mathbf{x} = \phi(\mathbf{X})$. The deformation gradient is the derivative of the spatial coordinates with respect to the material coordinates, and is thus defined as:

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \frac{\partial \phi}{\partial \mathbf{X}}$$

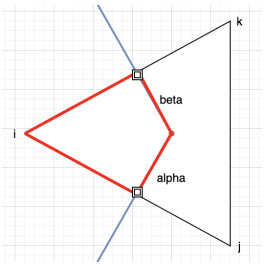

Fig. 2. The cleverly constructed closed surface, where the edge pieces from vertex i to vertex j and k are used along with the edge pieces α and β.

We can also think of the deformation gradient as a differentiable, and we then have:

$$d\mathbf{x} = \mathbf{F}d\mathbf{X}$$

If we assume the coordinates changes linearly over our triangles, that means the deformation gradient will be constant for all vectors on the triangles. To compute the deformation gradient we can thus take two points on our triangle in spatial coordinates, a and b, and their corresponding location in material coordinates, A and B, and then we have the following relations:

$$a = \mathbf{F}A$$
$$b = \mathbf{F}B$$

This can be combined and written as a linear system:

$$\begin{bmatrix} a & b \end{bmatrix} = \mathbf{F}^e \begin{bmatrix} A & B \end{bmatrix}$$

If we define $D^e = \begin{bmatrix} a & b \end{bmatrix}$ and $D_0^e = \begin{bmatrix} A & B \end{bmatrix}$ we have:

$$\mathbf{F}^e = D^e (D_0^e)^{-1}$$

And as long our triangles does not collapse, we can always find the inverse. With $\mathbf{F}^e$ we can now compute the Green strain tensor $\mathbf{E}^e$, the second Piola-Kirchhoff tensor, $\mathbf{S}^e$ and the first Piola-Kirchhoff tensor, $\mathbf{P}^e$:

$$\mathbf{E}^e = \frac{1}{2}\left(\mathbf{F}^{eT}\mathbf{F}^e - \mathbf{I}\right)$$
$$\mathbf{S}^e = \lambda tr\left(\mathbf{E}^e\right)\mathbf{I} + 2\mu\mathbf{E}^e$$
$$\mathbf{P}^e = \mathbf{F}^e\mathbf{S}^e$$

Where $\mathbf{I}$ is the identity tensor, λ first lamé coefficient and mu is the second lamé coefficient.

Following the usual Finite Volume Method we would apply the midpoint approximation rule to the term $\int_{S_\gamma^{e_{inner}}} \mathbf{P}\mathbf{N}dS$ and get $\left[\mathbf{P}^e\mathbf{N}_\gamma^e\right]_c l_\gamma^e$. However, due to our choice of control volumes we can evaluate this a bit smarter. A closed surface integral over a constant tensor field is always 0, that is $\int_S \mathbf{P}\mathbf{N}dS = 0$. We can now construct a clever closed surface integral:

$$\int_{S_{ij}^e} \mathbf{P}^e\mathbf{N}dS + \int_{S_{ik}^e} \mathbf{P}^e\mathbf{N}dS + \int_{S_\alpha^e} \mathbf{P}^e\mathbf{N}dS + \int_{S_\beta^e} \mathbf{P}^e\mathbf{N}dS = 0$$

Where the subscript denotes an edge piece in our mesh, and is illustrated in Figure 2. We can now rearrange the terms and get:

$$\int_{S_\alpha^e} \mathbf{P}^e\mathbf{N}dS + \int_{S_\beta^e} \mathbf{P}^e\mathbf{N}dS = -\int_{S_{ij}^e} \mathbf{P}^e\mathbf{N}dS - \int_{S_{ik}^e} \mathbf{P}^e\mathbf{N}dS$$

And we can now apply the midpoint approximation rule and define the internal forces:

$$\mathbf{f}_i^e = -\int_{S_{ij}^e} \mathbf{P}^e\mathbf{N}dS - \int_{S_{ik}^e} \mathbf{P}^e\mathbf{N}dS$$
$$\iff -\frac{1}{2}\mathbf{P}^e\mathbf{N}_{ij}^e l_{ij}^e - \frac{1}{2}\mathbf{P}^e\mathbf{N}_{ik}^e l_{ik}^e$$

Where the half comes from the fact we only need half of the edge lengths, and l is the length of the edge. We now have:

$$m_i\ddot{\mathbf{x}}_i = \mathbf{f}_i^{ext} + \mathbf{f}_i^t + \sum_{e_{inner}} \mathbf{f}_i^{e_{inner}}$$

We can simplify this by defining $f_i^{total} = \mathbf{f}_i^{ext} + \mathbf{f}_i^t + \sum_{e_{inner}} \mathbf{f}_i^{e_{inner}}$ and we then have:

$$m_i\ddot{\mathbf{x}}_i = \mathbf{f}_i^{total}$$

The last part is the time discretization. Here we want to discretize the velocity updates and positional updates. We know $\ddot{\mathbf{x}} = \dot{\mathbf{v}}$ and $\mathbf{v} = \dot{\mathbf{x}}$, thus we have:

$$\dot{\mathbf{v}}_i = \frac{1}{m_i}\mathbf{f}_i^{total} \tag{1}$$
$$\dot{\mathbf{x}}_i = \mathbf{v}_i \tag{2}$$

We can now use a forward difference approximation to approximate $\dot{\mathbf{v}}_i$:

$$\dot{\mathbf{v}}_i \approx \frac{\mathbf{v}_i^{t+\Delta t} - \mathbf{v}_i^t}{\Delta t}$$

Substituting this in at Equation 1, we get:

$$\mathbf{v}_i^{t+\Delta t} = \mathbf{v}_i^t + \frac{\Delta t}{m_i}\mathbf{f}_i^{total}$$

We now have an explicit velocity update as the total force is updated at time t.
We can now use a forward difference approximation for the $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}}_i \approx \frac{\mathbf{x}_i^{t+\Delta t} - \mathbf{x}_i^t}{\Delta t}$$

Substituting this into Equation 2 we get:

$$\mathbf{x}_i^{t+\Delta t} = \mathbf{x}_i^t + \Delta t\mathbf{v}_i^{t+\Delta t}$$

Because we just updated the velocities this becomes an implicit positional update.

We can now set up a simulation loop consisting of 9 steps:

(1) Compute deformation gradient $\mathbf{F}^e$ for all e.
(2) Compute Green strain tensor $\mathbf{E}^e$ for all e.
(3) Compute second Piola-Kirchhoff stress tensor $\mathbf{S}^e$ for all e.
(4) Compute first Piola-Kirchhoff stress tensor $\mathbf{P}^e$ for all e.
(5) Compute elastic forces $\sum_{e_{inner}} \mathbf{f}_i^{e_{inner}}$ for all i.
(6) Compute total forces $\mathbf{f}_i^{total}$ for all i.
(7) Compute velocity update $\mathbf{v}_i^{t+\Delta t}$ for all i.
(8) Compute position updates $\mathbf{x}_i^{t+\Delta t}$ for all i.
(9) Increment time $t = t + \Delta t$, and begin anew from step 1.

2.2 Central difference approximation

In the derivation of the Cauchy equation we used the forward difference approximation to derive the velocity and positional updates. This along with the backwards and central difference approximations can be derived from a Taylor expansion. We will here see the derivation for a second order central difference approximation. The other approximations are very similar.
First we write up the forward and backwards terms:

$$f(x_i + \Delta x_i) = f(x_i) + f^{(1)}\Delta x + \frac{1}{2}f^{(2)}(\Delta x)^2 + \mathbf{o}((\Delta x)^2)$$
$$f(x_i - \Delta x_i) = f(x_i) - f^{(1)}\Delta x + \frac{1}{2}f^{(2)}(-\Delta x)^2 + \mathbf{o}((-\Delta x)^2)$$

Adding these equations together gives us:

$$\begin{aligned} f(x_i + \Delta x_i) + f(x_i - \Delta x_i) = f(x_i) + f^{(1)}\Delta x + \frac{1}{2}f^{(2)}(x_i)(\Delta x)^2 + \\ \mathbf{o}((\Delta x)^2) + f(x_i) - f^{(1)}\Delta x + \frac{1}{2}f^{(2)}(x_i)(-\Delta x)^2 + \\ \mathbf{o}((-\Delta x)^2) \iff \end{aligned}$$
$$f(x_i + \Delta x_i) + f(x_i - \Delta x_i) = 2f(x_i) + f^{(2)}(x_i)(\Delta x)^2 + \mathbf{o}((\Delta x)^4) \iff$$
$$\frac{f(x_i + \Delta x_i) + f(x_i - \Delta x_i)}{(\Delta x)^2} = \frac{2f(x_i)}{(\Delta x)^2} + f^{(2)}(x_i) + \frac{\mathbf{o}((\Delta x)^4)}{(\Delta x)^2} \iff$$
$$\begin{aligned} f^{(2)}(x_i) = \frac{f(x_i + \Delta x_i) + f(x_i - \Delta x_i) - 2f(x_i)}{(\Delta x)^2} - \\ \mathbf{o}((\Delta x)^2) \end{aligned}$$

We thus find:

$$\frac{\partial^2 f(x_i)}{\partial x^2} \approx \frac{f(x_i + \Delta x_i) + f(x_i - \Delta x_i) - 2f(x_i)}{(\Delta x)^2}$$

The $\mathbf{o}((\Delta x)^2)$ means the approximation is second order accurate.

2.3 Matrix assembly versus update schemes

When solving our partial differential equations for space and they are not time dependent, we can usually apply two methods to find solutions. The first one being to derive an update scheme which describes which other nodes in our mesh another node depends on. This can be combined into a stencil which can be moved over the entire mesh iteratively until no nodes in our mesh changes values. This requires us to make an initial guess about their values which, if far from the correct solution, can cause slow convergence.
Another way to find a solution is to note that a stencil form a linear system of equations which can be assembled into a matrix system of the form $\mathbf{Ax} = \mathbf{b}$ where we would be able to solve for $\mathbf{x}$.
The first approach is known as a *update scheme* approach, and the second is known as a *matrix assembly*. The advantage of using an update scheme approach is we do not need storage for a matrix which can end up being quite big as it need to hold values for all nodes in our mesh. However, it might take many iterations before an update scheme converges. At the same time, if we do matrix assembling we can apply different solvers to the matrix system like LU, QR or Cholesky factorization which might speed up process quite significantly.

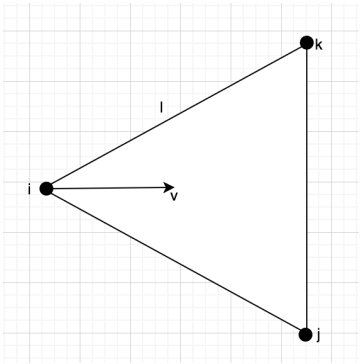

Fig. 3. A vertex *i* with some velocity. If the time step we take gets too big, the vertex will be moved 'outside' the triangle, and the mesh will collapse.

3 EXPERIMENT 1

In this experiment we would like to experimentally verify how big time steps we can take before the simulation breaks down. As the simulation breaks down if the triangles collapse we need to ensure our time steps are small enough that the nodal velocities will not move the vertices 'through' an edge in the mesh. That is, in a single time step, no vertex should because of its velocity be moved beyond another vertex and collapse a triangle in the mesh. In Figure 3 we see a vertex with a velocity, and if the time step is big enough the vertex will be moved 'to the outside' of the triangle. As it can be hard to measure the length the vertex has to travel before the triangle collapses, and we want this to hold for all vertices, we can approximate the length with the smallest edge length in the mesh. We can thus write the following relation:

$$l_{min} > \mathbf{v}_{max} \cdot \Delta t \iff$$
$$\Delta t < \frac{l_{min}}{\mathbf{v}_{max}}$$

We can now look at our velocity update given by the Finite Volume Method derivation. The forces acting are primarily the elastic forces, and we simplify and only look at these:

$$\mathbf{v}_i^{t+\Delta t} = \mathbf{v}_i^t + \frac{\Delta t}{m_i}\mathbf{f}_i^{elastic}$$

We can here define the mass as $m_i = \rho l$. We also note from the derivation of the first Piola-Kirchhoff tensor, the second Piola-Kirchhoff tensor and the Green strain tensor that the Green strain tensor depend on the deformation gradient, and from the definition of the second Piola-Kirchhoff tensor we see $\lambda, \mu \propto \mathbf{E}$. As the lamé coefficients are proportional to Young's modulus, E, we have $\mathbf{f}_i^{elastic}$ is proportional to Young's modulus, and thus we can substitute Young's modulus into our previous equation and find:

$$\mathbf{v}_{max} \propto \frac{\Delta t}{\delta l}E$$

Materials:	Rubber	Glass	Steel	concrete
Δt used:	0.00026	0.0000159	0.00001305	0.0000205
Upper bound:	0.00082	0.000017	0.000016	0.000022
$\frac{\Delta t \text{used}}{\text{Upper bound}}$	0.317	0.935	0.816	0.931

Table 1. Results of experiment 1.

Substituting this into our original relation we find:

$$\Delta t < \frac{l_{min}}{\mathbf{v}_{max}} \iff$$
$$\Delta t < \frac{l_{min}}{\frac{\Delta t}{\rho l_{min}} E} \iff$$
$$\Delta t < \frac{\rho l_{min}^2}{\Delta t E} \iff$$
$$\Delta t < \sqrt{\frac{\rho l_{min}^2}{E}}$$

And we thus have an approximation to an upper bound on Δt. We can now run a series of experiments where we for different material find the largest Δt where the simulation does not break down, and compare this with out analytically computed upper bound. Here we denote a successful simulation (a 'non-breakdown') as a simulation which runs for 1 second without showing unexpected behaviour. We use the same ball for the free fall for all materials, and the shortest edge in the ball is 0.080. The results can be seen in Table 1.

3.1 Discussion of results

From our results we see that our upper bound does indeed hold for a wide arrange of materials. However, it does vary a little how tight of an upper bound it is. We here especially note how the upper bound for rubber is a lot higher than the Δt actually used to make the simulation work. This is probably due to how much smaller Young's modulus is for rubber compared to the other materials used in the experiment, where the upper bound is a lot tighter. We thus conclude we have experimentally verified that we can find an upper bound to Δt with the relation $\Delta t < C\sqrt{\frac{\rho l_{min}^2}{E}}$ where C is a constant somewhere between 0.3 and 0.9.

4 EXPERIMENT 2

In this experiment we would like to experimentally verify that our simulation gets more accurate as we increase the mesh resolution. We will do this by applying light gravitational force to a cantilever beam fixed to a wall. We then measure the position of a lowest hanging vertex as we increase the mesh resolution. In the real world gravity would be applied to all molecules, and so our expectation is the measured position converge towards some fixed value as the simulation gets more realistic with more nodes. Our initial grid resolution will be 12x4, and we will then increase the mesh resolutions in multiples of this initial resolution. The results can be seen in Figure 4.

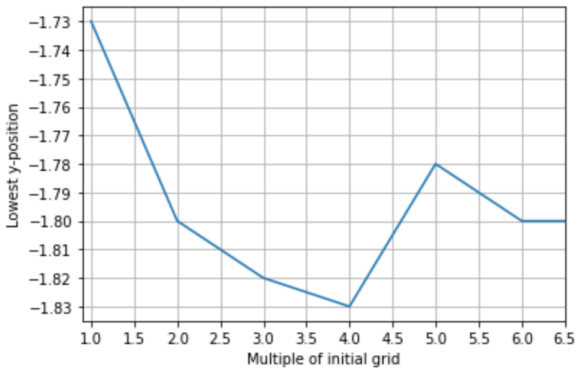

Fig. 4. Lowest vertex position of the simulation measured for different mesh resolutions.

4.1 Discussion of results

In the beginning we see the lowest positions changes quite a lot as we make the grid finer, but towards the end it seems the lowest position converges towards -1.8. This matches our expectation about fine mesh resolutions would converge towards a fixed value. We thus conclude our discretization of how the forces are applied is correct.

5 CONCLUSION

In conclusion we have derived the application of the Finite Volume Method on the Cauchy equation and seen how we can set up a simulation loop for elastic objects. We have also seen in experiment one how we can derive an upper bound on Δt and experimentally verified this upper bounds holds. From this experiment we concluded the upper bound is more tight for higher Young's modulus values. In experiment 2 we looked at how the lowest vertex position in the grid changed as the grid resolution increased, and here we experimentally verified that we would converge towards some fixed position as the grid got more fine. We thus end by concluding we apply our force correctly as our results from experiment 2 shows our simulation gets more accurate at we increase the mesh resolution.