Final Project Report

Jiexun Xu

1. Introduction

Variational integrators are a class of numerical integrators for physical simulations. This project seeks to implement one such integrator proposed in [Kharevych et al 2006]. The integrator proposed in that paper is based on the Hamilton-Pontryagin Principle. The resultnig integrator should have exact momentum preservation and correct energy behavior. Furthermore, external forces and holonomic constraints can be easily applied to the system. The major component of this project is the proposed numerical integrator and a GUI that allows the user to control most of the variables (step size, material property etc) in the simulation.

2. Related Work

There has been a lot of work done in variational integrators. [Marsden and West 2001] gives a review of integration algorithms based on discrete variational principles. Other work, such as [Kane et al 1999] and [Taeyoung et al 2005], also proposes different integration schemes. Variational integrators are interesting to the community because these integrators give exact linear and angular momentum preservations, their energy behavior isreasonble, and they're simple to implement

3. Technical & Implementation Details

The core of the system is the discretized Hamilton-Pontryagin Principle, which can be expressed as

$$q_{k+1} - q_k = h_k v_{k+1} \qquad (1)$$

$$p_{k+1} - p_k = \frac{\partial L}{\partial q_k} \qquad (2)$$

$$h_k p_{k+1} = \frac{\partial L}{\partial v_{k+1}} \qquad (3)$$

where p is the momentum, q is the position, v is the velocity, and L is the Lagrangian of the system.

Holonomic constraints and external forces can be easily added to equation (2) above, yielding

$$p_{k+1} - p_k = \frac{\partial L}{\partial q_k} + F_{ext} + h_k \gamma_k \nabla \mathrm{g}(q_k) \quad (4)$$

where F_{ext} is external force, h_k is time step size, γ_k is the Lagrange multiplier to impose the constraint $\mathrm{g}(q_k) = 0$, and $\nabla \mathrm{g}(q_k)$ is the gradient of $\mathrm{g}(q_k)$.

To solve equations (1) ~ (3), simply plug (2) into (3) and solve for v_{k+1}, and update q_{k+1} and p_{k+1}. Both explicit and implicit methods can be used to solve for v_{k+1}. Explicit methods are easy to implement and faster to solve, but are limited in step size by the Courant-Friedrichs-Lewy condition, as explained in [Jeltsch 1996]. Implicit methods are not limited in step size, but need to solve an optimization problem, so they are slower. This project implemented the explicit integration method.

This project applied equations (1) ~ (4) in non-linear elastic material simulation. In particular, the discretized version of equations (1), (3) and (4) are

$$v_{k+1} = M^{-1}[p_k - h_k \nabla \mathrm{W} - k_D \nabla W_{damp} + h_k F_{ext} + h_k \gamma_k \nabla \mathrm{g}(q_k) \qquad (5)$$

$$p_{k+1} = M v_{k+1} \qquad (6)$$

$$q_{k+1} = q_k + h_k v_{k+1} \qquad (7)$$

The project implemented three different material energy models: Neo-hookean model, Mooney-Rivlin model and Saint Venant–Kirchhoff model. All three models require

computation of the deformation gradient and the Cauchy-strain tensor. I followed the same method used in [Sumner and Popovic 2004] to compute the deformation gradient at each step.

Before computing the energy density function, an estimation of the displacement fields of each vertex with respect to their rest positions is necessary. Since the system allows the user to change holonomic constraints, momentums, external forces and environments at any time, it's difficult to keep track of the rigid motion of the object at any time, making the estimation of the displacement fields difficult. To simplify this problem, I used the shape matching technique described in [Muller et al 2005].

There're two different ways to compute the gradient of the energy density function. One way is to use finite differences to approximately compute the gradient, and the other way is to symbolically find the gradient of the energy density function. I calculated the gradient of the Saint Venant-Kirchhoff model symbolically and compared the result to the finite difference model. The relative error is very small (usually less than 0.1%), and since the auto generated code is very lengthy, so I used the finite difference in the final code for efficiency considerations. In the GUI, the user can change the two parameters for the three models at any time in the simulation to experiment with different parameters.

The damping force I used is very simple: its magnitude is proportional to the magnitude of the velocity in each direction, and its direction is opposite to the direction of the velocity. In the GUI, the user can easily control the proportionality constant in the damping force. If the constant is set to zero, there is no damping force.

There're two types of external forces. One is set by the user. The user can set the external force on any vertex at any time during the simulation. The other is set according to the environment. Currently three types of environments are implemented. The first one has no external forces, the second one has uniform gravity that points in negative y direction, and its magnitude is proportional to the mass of each vertex. The third one will attract each vertex to the origin with a force that is proportional to the inverse square of the distance of each vertex to the center (when the vertex is already at the center, there is no attractive force). This environment is similar to an environment in which each vertex can be thought of as charged particles, and the origin can be thought of as an oppositely charged monople. The user can easily switch environments and change the proportionality constant anytime during the simulation.

Currently, only one type of holonomic constraint is implemented. The constraint is of the form

$$\sum_{\boldsymbol{v}_i \in V} (\boldsymbol{q}_{ik} - \boldsymbol{c}_i)^2 = 0$$

This constraint fixes every vertex $\boldsymbol{v}_i$ to a fixed point $\boldsymbol{c}_i$. There's also a boolean variable for each vertex to indicate whether this vertex is fixed or not. The gradient of this constraint is simple:

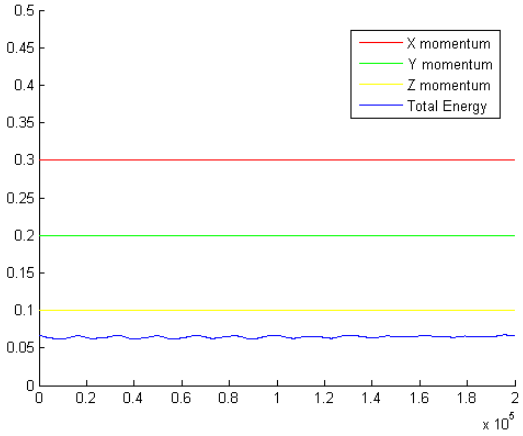
$$\nabla g = \sum_{\boldsymbol{v}_i \in V} 2(\boldsymbol{q}_{ik} - \boldsymbol{c}_i) = 0$$

The GUI allows the user to fix or unfix a given vertex to any position.

Other than the functionalities described above, the GUI also allows the user to change the step size, set the momentum of each vertex at any step to experiment with different parameters. The user can also change rendering and viewing options such as render the mesh as wireframes or point clouds, draw or hide external forces, momentums and fixed points, rotate, move and zoom the camera or rotate the lighting source. The goal of the GUI is to allow me to easily experiment with many different parameters (there're so many parameters to experiment with) and gain a real experience of many of the things learned in this class.

4. Results and Observations

For all the simulations I tried including a single tetrahedron, a cube, and a tetrahedron, the momentums are conserved exactly as long as there're no external forces or holonomic constraints. The energy oscillates around the initial energy of the system. Depending on different parameter settings, the oscillation can go from tiny to significant (which usually lead a blowup soon afterwards). I'm unable to get a simulation where the energy behavior is as nice as the one described in [Kharevych et al 2006]. The following is a plot of total energy and momentum. It's also one of the best energy behaviors I obtained:



When damping is added, as anticipated, the energy decreases slowly, but the momentum is still preserved.

When external forces are added, nothing is conserved, but the mesh moves and rotates correctly as indicated by the direction and magnitude of the external forces.

When holonomic constraints are added, there's an additional parameter, γ_k, to tune. Ideally, setting this parameter to a very large number will strongly impose the holonomic constraint. Unfortunately, that would also easily blow up the system, unless extremely small time step size is used.

Being able to control so many variables at any time during the simulation is both fun and pain. It's fun because it's possible to try all kinds of weird combinations of parameters, and try them at any state of the simulation. It's a pain because there're so many parameters to tune, and unlike machine learning where some tuning parameters can be automated, the only way to tune parameters here is through trials and errors. Among the experiments I played with, I encountered much more blowups than stable simulations due to incorrect parameter combinations.

5. Conclusion and Futhre Work

The variational integrators proposed in the paper indeed give exact momentum conservation, and as propsed in the paper, it's easy to incoorperate external forces and holonomic constraints into the system. The energy behavior I obtained is not as good as the experiment proposed in the original paper, but the behavior is still reasonable. For future work, I'd like to implicit methods and see if indeed the system won't blowup for large time step sizes. Even the authors of the paper haven't tried implicit methods.

References

[Kharevych et al 2006] L. Kharevych, Weiwei, Y. Tong, E. Kanso, J. Marsden, P. Schroder, M. Desbrun, Geometric,Variational Integrators for Computer Animation

[Kane et al 1999] C. Kane, J. Marsden, M. Ortiz, Symplectic-energy-momentum Preserving Variational Integrators

[Taeyoung et al 2005] L. Taeyoung, N. McClamroch, M. Leok, A Lie Group Variational Integrator for the Attitude Dynamics of a Rigid Body with Applications to the 3D Pendulum

[Marsden and West 2001] J. Marsden, M. West, Discrete Mechanics and Variational Integrators

[Jeltsch 1996] R. Jeltsch, On the Courant-Friedrichs-Lewy Condition Equipped with Order for Hyperbolic Differential Equations
[Sumner and Popovic 2004] Deformation Transfer for Triangle Meshes
[Muller et al 2005] M. Muller, B. Heidelberger, M. Teschner, M. Gross, Meshless Deformations Based on Shape Matching