ODE solvers recursive equations

Aurelie Degletagne

June 11, 2015

Contents

1 Notations **2**

2 Euler solver **2**

2.1 EulerExplicit solver 2

2.2 EulerImplicit solver 2

3 Newmark solver **3**

4 VariationalSymplectic solver **3**

4.1 VariationalSymplecticExplicit solver 3

4.2 VariationalSymplecticImplicit solver 4

5 RungeKutta **5**

5.1 Runge Kutta second order 6

5.2 Runge Kutta fourth order 6

6 CentralDifference solver **7**

1 Notations

In each following equation h is the time step, x_k,v_k,a_k the position, velocity and acceleration at the time step k, K the stiffness, M the mass, r_m the rayleigh mass, r_k the rayleigh stiffness and F_{ext} the external forces.

2 Euler solver

2.1 EulerExplicit solver

It is the simplest time integration solver.

The integration scheme is based on the following equations:

$v_{k+1} = v_k + ha_k$

$x_{k+1} = x_k + hv_k$

2.2 EulerImplicit solver

It is a semi-implicit time integrator using backward Euler scheme for first and second degree Ordinary Differential Equations (ODE). This is based on [Baraff and Witkin, Large Steps in Cloth Simulation, SIGGRAPH 1998]

The integration scheme is based on the following equations:

$v_{k+1} = v_k + h\frac{Fext_k + v_k(-(r_k+h)K - r_m M)}{h(h+r_k)K + (1+hr_m)M}$

$x_{k+1} = x_k + hv_k$

3 Newmark solver

It is an implicit time integrator using Newmark scheme. The implementation computes a_k directly then solves the first equation to compute a_{k+1}, and finally computes the new velocity v_{k+1} and the new position x_{k+1}. In Sofa one uses for Newmark coefficients $\gamma = 0.5$ and $\beta = 0.25$.

$$a_{k+1} = \frac{Fext_k + h*cstAcc*a_k + cstVel*v_k}{(h^2\beta + h\gamma r_k)K + (1 + h\gamma r_m)M} \begin{cases} cstAcc &= (-r_m(1-\gamma)M - h(0.5-\beta)K - rk(1-\gamma)K) \\ cstVel &= (-r_m M - (h + r_k)K) \end{cases}$$

$$v_{k+1} = v_k + h(1-\gamma)a_k + h\gamma a_{k+1}$$

$$x_{k+1} = x_k + hv_k + 0.5h^2(1-2\beta)a_k + h^2\beta a_{k+1}$$

4 VariationalSymplectic solver

It is explicit and implicit time integrator using the Variational Symplectic Integrator as defined in: Kharevych, L et al. Geometric, Variational Integrators for Computer Animation. ACM SIGGRAPH Symposium on Computer Animation 4 (2006): 4351. p is the momentum.

4.1 VariationalSymplecticExplicit solver

$$v_{k+1} = \frac{2(Fext_k + p_k - hr_m v_k M)}{M}$$

$$p_{k+1} = Mv_{k+1}$$

$$x_{k+1} = x_k + hv_{k+1}$$

4.2 VariationalSymplecticImplicit solver

The current implementation for implicit integration assume $alpha = 0.5$ (quadratic accuracy) and uses several Newton steps to estimate velocity. We have 3 state variables: position q_t, the velocity v_t and the momentum p_t initialized such that $p_0 = Mv_0$.

2 steps are required for this solver:

Step 1: **Find the new velocity v_{k+1} by minimizing Lylian energy.**

The Lilyan L energy is the sum of kinetic energy, elastic energy and damping:

$L(v_{k+1}) = \frac{h}{2} v_{k+1}^T M v_{k+1} + hW(q_k + \frac{h}{2} v_{k+1}) - hp_k v_{k+1}$

To minimize this energy we solve the non-linear system of equations:

$\partial L(v_{k+1}) = 0 = hMv_{k+1} - \frac{h^2}{2} F(q_k + \frac{h}{2} v_{k+1}) - hp_k$

where F() is the force associated with the potential energy (opposite gradient)

Finding the roots of the non-linear equations can be done performing several newton steps by linearizing the force around $q_k + \frac{h}{2} v_{k+1} = q^*$

This is an iterative process:

- Define q_i^* as the current estimate of q^*. The objective is to find q^* such that: $2 * M(q^* - q_k) - \frac{h^2}{2} F(q^*) - hp_k = 0$
- Search for $\partial q_i = q_{i+1}^* - q_k$ by linearizing the force around q_i^* At the end we have: $(\frac{4}{h^2} M - K)\partial q_i = F(q_i^*) - K()\partial q_{i-1} + \frac{2}{h} p_k$

This iterative process stops when the maximal number of newton iterations is reached or when the required accuracy of the estimated position q^* is reached. At the end we obtain the velocity for the next time step: $v_{k+1} = \frac{2(q_{i+1}^* - q_k)}{h}$

Step 2: **Update the new position q and the new momentum p**

$q_{k+1} = q_k + hv_{k+1}$

$p_{k+1} = Mvk + 1 + \frac{h}{2} q^*_{i+1}$

Energy conservation:

This solver enables to conserve the Hamiltonian energy.

$E_{Hamiltonian} = \frac{p^t_{k+1} M p_{k+1}}{2} + E_{potential}$

Some tests in modules/SofaImplicitOdeSolver/SofaImplicitOdeSolver_test show the energy conservation for a mass spring-system but also for a planet-sun system.

For example you can have a look at the example scene:

examples/Components/solver/VariationalSymplecticSolver.scn

It is an embedded beam with FEM force field. This scene uses the implicit variational solver and as you can see on the generated file energy.txt the Hamiltonian energy is constant.

5 RungeKutta

It is a popular explicit time integration method, much more precise than euler explicit solver.

5.1 Runge Kutta second order

It is the Runge-Kutta method with order 2 or the middle point rule. Functions are evaluated two times at each step.

At time t + h/2

$x_{k+\frac{1}{2}} = x_k + \frac{1}{2}hv_k$

$v_{k+\frac{1}{2}} = v_k + \frac{1}{2}ha_k$

$a_{k+\frac{1}{2}} = \frac{Fext_{k+\frac{1}{2}}}{m}$

At time t+h

$x_{k+1} = x_k + h * v_{k+\frac{1}{2}}$

$v_{k+1} = v_k + h * a_{k+\frac{1}{2}}$

$a_{k+1} = \frac{Fext_{k+1}}{m}$

5.2 Runge Kutta fourth order

It is the Runge-Kutta method with order four. Functions are evaluated four times at each step.

Variables

$stepBy2 = \frac{h}{2}$ $stepBy3 = \frac{h}{3}$ $stepBy6 = \frac{h}{6}$

At time t :

- **Step 1**

  $k1v = v_k$

  $k1a = a_k$

- **Step 2**

$k2x = x_k + k1v * stepBy2$

$k2v = v_k + k1a * stepBy2$

$k2a = (Fext_{stepBy2}(k2x, k2v))/m$

- **Step 3**

  $k3x = x_k + k2v * stepBy2$

  $k3v = v_k + k2a * stepBy2$

  $k3a = (Fext_{stepBy2}(k3x, k3v))/m$

- **Step 4**

  $k4x = x_k + k3v * h$

  $k4v = v_k + k3a * h$

  $k4a = (Fext_{k+1}(k4x, k4v))/m$

At time t + h :

$x_{k+1} = x_k + k1v * stepBy6 + k2v * stepBy3 + k3v * stepBy3 + k4v * stepBy6$

$v_{k+1} = v_k + k1a * stepBy6 + k2a * stepBy3 + k3a * stepBy3 + k4a * stepBy6$

$a_{k+1} = (Fextk + 1)/m$

6 CentralDifference solver

It is an explicit time integrator using central difference (also known as Verlet or Leap-frog). The equations are separated in two cases: either the rayleigh mass is null or not.

- if $rm = 0$

$a_{k+1} = \frac{Fext_{k+1}}{m}$

$v_{k+1} = v_k + ha_{k+1}$

$x_{k+1} = x_k + hv_{k+1}$

- if $rm! = 0$

$a_{k+1} = \frac{Fext_{k+1}}{m}$

$$v_{k+1} = cstVel * v_k + cstAcc * a_k \begin{cases} cstVel = \dfrac{\frac{1}{h} - \frac{r_m}{2}}{\frac{1}{h} + \frac{r_m}{2}} \\ cstAcc = \dfrac{1}{\frac{1}{h} + \frac{r_m}{2}} \end{cases}$$

$x_{k+1} = x_k + hv_{k+1}$