

Tweaking ODE for variable time steps

Something that's been haunting ODE (Open Dynamics Engine) users is the inability to keep the physics simulation in synch with the rendering frame rate while keeping the system stable. This article presents a simple yet effective method in providing consistent simulation behavior over different time step lengths.

The main tools ODE provides to affect the overall stability of the simulation is the constraint force mixing (CFM) and error reduction parameter (ERP). Most users simply set a fixed time step and assign CFM and ERP to (more or less arbitrary) fixed numbers. The key to solving the problem with keeping the physics simulation in synchronization with the rendering is to change the CFM and ERP values whenever the frame rate (time step) changes.

The spring and dampener representation

As a first step; forget all about CFM and ERP. They aren't really what we want to handle since their behavior is greatly affected by the length of our time step. Instead I present to you the spring constant k and damping coefficient c . As you know, a spring exerts a force proportional to its extension and a dampener exerts a force proportional to the velocity of its extension:

$$\begin{aligned} F_k &= -kx \\ F_c &= -c\dot{x} \quad , \\ F &= F_k + F_c \end{aligned} \quad (1)$$

where x is the extension.

The Taylor expansion of the second order can be written as:

$$x(t + \Delta t) \approx x(t) + \dot{x}(t) \cdot \Delta t + \frac{\ddot{x}(t)}{2} \cdot \Delta t^2, \quad (2)$$

where Δt is the length of the time step.

In a simplified model without any other forces affecting a body the following holds true:

$$F = m\ddot{x}, \quad (3)$$

where m is the body's mass.

Equation 1, 2 and 3 gives:

$$x(t + \Delta t) \approx x(t) \left(1 - \frac{k}{2m} \cdot \Delta t^2 \right) + \dot{x}(t) \left(1 - \frac{c}{2m} \cdot \Delta t \right) \cdot \Delta t. \quad (4)$$

We also have:

$$\ddot{x}(t) \approx \frac{\dot{x}(t + \Delta t) - \dot{x}(t)}{\Delta t} \Leftrightarrow \dot{x}(t + \Delta t) \approx \dot{x}(t) + \ddot{x}(t) \cdot \Delta t,$$

and therefore it holds true that:

$$\dot{x}(t + \Delta t) \approx \dot{x}(t) \left(1 - \frac{c}{m} \cdot \Delta t \right) - \frac{k}{m} x \cdot \Delta t. \quad (5)$$

The numerical approximation of the solution consists of Equation 4 and 5. Please note that even though this approximation isn't optimal, it serves its purpose in providing a way to compare the behavior for different lengths of time steps.

The exact solution (for weak damping) is¹:

$$x(t) = Ce^{-\frac{c}{2m}t} \sin(\omega t + \alpha) \quad , \quad \omega = \sqrt{\frac{k}{m} - \left(\frac{c}{2m}\right)^2} \quad , \quad \frac{c}{2\sqrt{mk}} < 1,$$

where C and α determines the initial position and velocity.

The time derivative of the exact solution is:

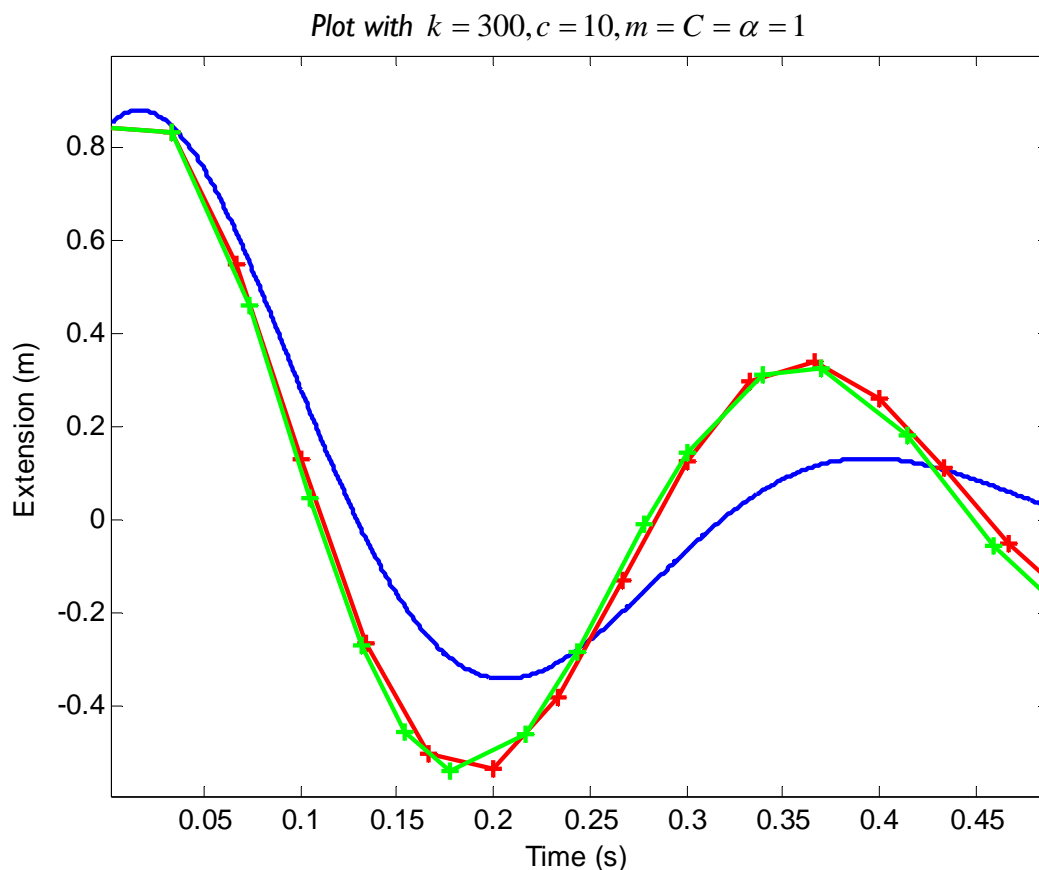
$$\dot{x}(t) = Ce^{-\frac{c}{2m}t} \left(\omega \cos(\omega t + \alpha) - \frac{c}{2m} \sin(\omega t + \alpha) \right) \quad , \quad \omega = \sqrt{\frac{k}{m} - \left(\frac{c}{2m}\right)^2} \quad , \quad \frac{c}{2\sqrt{mk}} < 1,$$

and can be used to calculate $\dot{x}(0)$ (used by the approximation) for the specified C and α .

$$x(0) = C \sin(\alpha) \quad , \quad \frac{c}{2\sqrt{mk}} < 1.$$

$$\dot{x}(0) = C \left(\omega \cos(\alpha) - \frac{c}{2m} \sin(\alpha) \right) \quad , \quad \omega = \sqrt{\frac{k}{m} - \left(\frac{c}{2m}\right)^2} \quad , \quad \frac{c}{2\sqrt{mk}} < 1.$$

We will now plot the extension in time for an arbitrary case using the above approximation for both fixed and variable time steps, with the actual solution as reference.



Blue exact solution

Red fixed time step, $\Delta t = 1/30$ (30 fps)

Green variable time step, $1/15 \leq \Delta t \leq 1/45$ (15-45 fps), uniform distribution

The approximations are a very close match to each other and thus gives us an indication that the spring and dampener representation yields consistent result for variable time steps (even though the ODE integrator most likely differs from this simple approximation). Since the approximation isn't a particularly good one, and the time step is relatively large, it doesn't match the exact solution very closely.

Implementation in ODE

As it turns out, we can convert our spring constant and damping coefficient to ODE's CFM and ERP counterparts using the following formulas²:

$$erp = \frac{\Delta t \cdot k}{\Delta t \cdot k + c}$$

$$cfm = \frac{1}{\Delta t \cdot k + c}$$

If CFM and ERP are updated using these formulas every frame we can expect consistent behavior with sufficiently small time steps.

How do you pick the k and c values? The answer is an educated guess, like when picking suitable CFM and ERP values for your simulation. If you got some good values for a fixed time step (a certain Δt) you can convert them to k and c values by solving the equations:

$$k = \frac{erp}{\Delta t \cdot cfm}$$

$$c = \frac{1 - erp}{cfm}$$

Don't forget to handle all your CFM and ERP values in this time step independent manner (some joints also allows for these values to be set) and make sure to specify a largest time step the simulation can handle. If the time step gets too large the simulation might "blow up" (as usual). No lower limit is needed since it will only make the simulation more accurate.

For your convenience the pseudo code for a partial physics update, supporting variable time steps, is provided below.

```
foreach joint in world {
    foreach cfmErpParameter in joint {
        k = cfmErpParameter.getSpringConstant()
        c = cfmErpParameter.getDampingCoefficient()

        cfmErpParameter.setErp(timeStep*k / (timeStep*k + c))
        cfmErpParameter.setCfm(1.0 / (timeStep*k + c))
    }
}

k = world.getSpringConstant()
c = world.getDampingCoefficient()

erp = timeStep*k / (timeStep*k + c)
cfm = 1.0 / (timeStep*k + c)

dWorldSetCFM(world.getOdeWorld(), cfm)
dWorldSetERP(world.getOdeWorld(), erp)

dWorldQuickStep(world.getOdeWorld(), timeStep)
```

Disclaimer

This method was originally developed by the author for the game demo Burning Tires which finished 2nd place in Intel Game Demo Contest 2007, category of Best Threaded Game³. The demo can be downloaded from the contest website. This method, wholly or in part, might be used for any purpose without any explicit crediting of the author.

The information contained in this article is provided "as is". Rasmus Barringer does not warrant the accuracy, adequacy or completeness of this information and expressly disclaims liability for errors or omissions in this information. No warranty of any kind, implied, express or statutory, including but not limited to the warranties of non-infringement of third-party rights, title, merchantability, and fitness for a particular purpose, is given in conjunction with the information.

References

1. Nyberg, Christer (2003). *Mekanik Grundkurs*. ISBN 91-47-05167-1.
2. http://opende.sourceforge.net/wiki/index.php/Manual_%28Concepts%29. *ODE Manual – Concepts*. Accessed October 30, 2007.
3. <http://softwarecommunity.intel.com/articles/eng/1409.htm>. *Intel Game Demo Contest Winners*. Accessed November 1, 2007.