Physically Based Modeling
Implicit Methods for Differential Equations

David Baraff
Pixar Animation Studios

Implicit Methods for Differential Equations

David Baraff
Pixar Animation Studios

1 Implicit Methods

The methods we have looked at for solving differential equations in the first section of these notes (Euler's method, the midpoint method) are all called "explicit" methods for solving ODE's. However, sometimes an ODE can become "stiff," in which case explicit methods don't do a very good job of solving them. Whenever possible, it is desirable to change your problem formulation so that you don't have to solve a stiff ODE. Sometimes however that's not possible, and you have just have to be able to solve stiff ODE's. If that's the case, you'll usually have to use an ODE solution method which is "implicit."

2 Example Stiff ODE

First, what is the meaning and cause of stiff equations? Lets consider an example that arises frequently in dynamics. Suppose that we have a particle, with position $(x(t), y(t))$, and suppose that we want the y-coordinate to always be zero. One way of doing this is to add a component $-ky(t)$ to $\dot{y}(t)$ where k is a large positive constant. If k is large enough, then the particle will never move too far away from $y(t) = 0$, since the $-ky(t)$ term always brings $y(t)$ back towards zero. However, lets assume that there is no restriction on the x-coordinate, and that we want a user to be able to pull the particle arbitrarily along the x-axis. So lets assume that over some time interval our differential equation is simply

$$\dot{\mathbf{X}}(t) = \frac{d}{dt}\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} -x(t) \\ -ky(t) \end{pmatrix}. \tag{2–1}$$

(We'll also assume that the particle doesn't start exactly with $y_0 = 0$.) What's happening here is that the particle is strongly attracted to the line $y = 0$, and less strongly towards $x = 0$. If we solve the ODE far enough forward in time, we expect the particle's location to converge towards $(0, 0)$ and then stay there once it arrives.

Now suppose we use Euler's method to solve the equation. If we take a step of size h, we get

$$\mathbf{X}_{new} = \mathbf{X}_0 + h\dot{\mathbf{X}}(t_0) = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + h\begin{pmatrix} -x_0 \\ -ky_0 \end{pmatrix}.$$

This yields

$$\mathbf{X}_{new} = \begin{pmatrix} x_0 - hx_0 \\ y_0 - hky_0 \end{pmatrix} = \begin{pmatrix} (1-h)x_0 \\ (1-hk)y_0 \end{pmatrix}.$$

If we look at the y component of this equation, we see that if $|1-hk|>1$ then the y_{new} we compute will have an absolute value which is larger than $|y_0|$. In other words, if $|1-hk|>1$, Euler's method will not converge to an answer: each step will result in a value of y_{new} which is larger than the last. Technically, Euler's method is unstable for $|1-hk|>1$. Thus, we better have $1-hk>-1$ or $hk<2$ if we hope to converge. The largest step we can hope to take is less than $2/k$.

Now, if k is a large number, we'll have to take very small steps. This means that the particle slides towards $(0,0)$ excruciatingly slowly. Even though the particle may nearly satisfy $y_0=0$, we have to take such small steps that the particles' progress along the x-axis is pretty much nonexistent. That's the embodiment of a stiff ODE. In this case, the stiffness arises from making k very large in order to keep the particle close to the line $y=0$. Later on, when we connect particles with second-order dynamics together with springs, we'll experience exactly the same effect: stiff ODE's. Even if we use a more sophisticated explicit method such as fourth-order Runge-Kutta, we may do a little better in the size of our steps, but we'll still have major problems.

Now as we said above, the name of the game is to pose your dynamics problems so that you don't experience stiff ODE's. However, when you can't, you'll have to turn towards an implicit solution method. The method we'll show below is the simplest of the implicit methods, and its based on taking an Euler step "backwards."

3 Solving Stiff ODE's

Given a differential equation

$$\frac{d}{dt}\mathbf{X}(t)=\boldsymbol{f}(\mathbf{X}(t)),$$

the explicit Euler update would be $\mathbf{X}_{new}=\mathbf{X}_0+h\boldsymbol{f}(\mathbf{X}(t_0))$, to advance the system forward h in time. For a stiff problem though, we change the update to instead be

$$\mathbf{X}_{new}=\mathbf{X}_0+h\boldsymbol{f}(\mathbf{X}_{new}) \tag{3–1}$$

That is, we're going to evaluate $\boldsymbol{f}$ at the point we're aiming at, rather than where we came from. (If you think about reversing the world and running everything backwards, the above equation makes perfect sense. Then the equation says "if you were at $\mathbf{X}_{new}$, and took a step $-h\boldsymbol{f}(\mathbf{X}_{new})$, you'd end up at $\mathbf{X}_0$." So if your differential equation represents a system that is reversible in time, this step makes sense. It's just finding a point $\mathbf{X}_{new}$ such that if you ran time backwards, you'd end up at $\mathbf{X}_0$.) So, we're looking for a $\mathbf{X}_{new}$ such that f, evaluated there, times h, points directly back at where we came from. Unfortunately, we can't in general solve for $\mathbf{X}_{new}$, unless f happens to be a linear function.

To cope with this, we'll replace $\boldsymbol{f}(\mathbf{X}_{new})$ with a linear approximation, again based on f's Taylor series. Lets define $\Delta\mathbf{X}$ by $\Delta\mathbf{X}=\mathbf{X}_{new}-\mathbf{X}_0$. Using this, we rewrite equation (3–1) as

$$\mathbf{X}_0+\Delta\mathbf{X}=\mathbf{X}_0+h\boldsymbol{f}(\mathbf{X}_0+\Delta\mathbf{X}).$$

or just

$$\Delta\mathbf{X}=h\boldsymbol{f}(\mathbf{X}_0+\Delta\mathbf{X}).$$

Next, lets approximate $\boldsymbol{f}(\mathbf{X}_0+\Delta\mathbf{X})$ by

$$\boldsymbol{f}(\mathbf{X}_0)+f'(\mathbf{X}_0)\Delta\mathbf{X}.$$

(Note that since $\boldsymbol{f}(\mathbf{X}_0)$ is a vector, the derivative $f'(\mathbf{X}_0)$ is a matrix.) Using this approximation, we can approximate $\Delta\mathbf{X}$ with

$$\Delta\mathbf{X} = h\left(\boldsymbol{f}(\mathbf{X}_0) + f'(\mathbf{X}_0)\Delta\mathbf{X}\right).$$

or

$$\Delta\mathbf{X} - hf'(\mathbf{X}_0)\Delta\mathbf{X} = h\boldsymbol{f}(\mathbf{X}_0)$$

Rewriting this as

$$\left(\frac{1}{h}\mathbf{I} - f'(\mathbf{X}_0)\right)\Delta\mathbf{X} = \boldsymbol{f}(\mathbf{X}_0),$$

where $\mathbf{I}$ is the identity matrix, we can solve for $\Delta\mathbf{X}$ as

$$\Delta\mathbf{X} = \left(\frac{1}{h}\mathbf{I} - f'(\mathbf{X}_0)\right)^{-1}\boldsymbol{f}(\mathbf{X}_0) \tag{3–2}$$

Computing $\mathbf{X}_{new} = \mathbf{X}_0 + \Delta\mathbf{X}$ is clearly more work than using an explicit method, since we have to solve a linear system at each step. While this would seem a serious weakness, computationally speaking, don't despair (yet). For many types of problems, the matrix f' will be sparse—for example, if we are simulating a spring-lattice, f' will have a structure which matches the connectivity of the particles. (For a discussion of sparsity and solution techniques, see Baraff and Witkin [1]. Basic material in Press *et al*. [2] will also prove useful.) As a result, it is usually possible to solve equation (3–2) in linear time (i.e. time proportional to the dimension of $\mathbf{X}$). In such cases, the payoff is dramatic: we can usually take considerably large timesteps without losing stability (i.e. without divergence, as happens with the explicit case if the stepsize is too large). The time taken in solving each linear system is thus more than offset by the fact that our timesteps are often orders of magnitude bigger than we could manage using an explicit method. (Of course, the code needed to do all this is much more complicated than in the explicit case; like we said, make your problems un-stiff if you can, and if not, pay the price.)

Lets apply the implicit method to equation (2–1). We have that $\boldsymbol{f}(\mathbf{X}(t))$ is

$$\boldsymbol{f}(\mathbf{X}(t)) = \begin{pmatrix} -x(t) \\ -ky(t) \end{pmatrix}.$$

Differentiating with respect to $\mathbf{X}$ yields

$$f'(\mathbf{X}(t)) = \frac{\partial}{\partial\mathbf{X}}\boldsymbol{f}(\mathbf{X}(t)) = \begin{pmatrix} -1 & 0 \\ 0 & -k \end{pmatrix}$$

Then the matrix $\frac{1}{h}\mathbf{I} - f'(\mathbf{X}_0)$ is

$$\begin{pmatrix} \frac{1}{h}+1 & 0 \\ 0 & \frac{1}{h}+k \end{pmatrix} = \begin{pmatrix} \frac{1+h}{h} & 0 \\ 0 & \frac{1+kh}{h} \end{pmatrix}$$

Inverting this matrix, and multiplying by $\boldsymbol{f}(\mathbf{X}_0)$ yields

$$\begin{aligned}\Delta\mathbf{X} &= \begin{pmatrix} \frac{1+h}{h} & 0 \\ 0 & \frac{1+kh}{h} \end{pmatrix}^{-1} \begin{pmatrix} -x_0 \\ -ky_0 \end{pmatrix} \\ &= \begin{pmatrix} \frac{h}{h+1} & 0 \\ 0 & \frac{h}{1+kh} \end{pmatrix} \begin{pmatrix} -x_0 \\ -ky_0 \end{pmatrix} \\ &= -\begin{pmatrix} \dfrac{h}{h+1}x_0 \\ \dfrac{h}{1+kh}ky_0 \end{pmatrix}\end{aligned}$$

What is the limit on the stepsize in this case? The answer is: there is no limit! In this case, if we let h grow to infinity, we get

$$\lim_{h\to\infty} \Delta\mathbf{X} = \lim_{h\to\infty} -\begin{pmatrix} \dfrac{h}{h+1}x_0 \\ \dfrac{h}{1+kh}ky_0 \end{pmatrix} = -\begin{pmatrix} x_0 \\ \dfrac{1}{k}ky_0 \end{pmatrix} = -\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}.$$

This means that we achieve $\mathbf{X}_{new} = \mathbf{X}_0 + (-\mathbf{X}_0) = \mathbf{0}$ in a single step! For a general stiff ODE, we won't be able to take steps of arbitrary size, but we will be able to take much larger steps using an implicit method than using an explicit method. The extra cost of solving a linear equation is more than made up by the time saved by taking large timesteps.

4 Solving Second-Order Equations

Most dynamics problems are written in terms of a second-order differential equation:

$$\ddot{\mathbf{x}}(t) = \boldsymbol{f}(\mathbf{x}(t), \dot{\mathbf{x}}(t)). \tag{4–1}$$

This equation is easily converted to a first-order differential equation by adding new variables. If we define $\mathbf{v} = \dot{\mathbf{x}}$, then we can rewrite equation (4–1) as

$$\frac{d}{dt}\begin{pmatrix} \mathbf{x}(t) \\ \mathbf{v}(t) \end{pmatrix} = \begin{pmatrix} \mathbf{v}(t) \\ \boldsymbol{f}(\mathbf{x}(t), \mathbf{v}(t)) \end{pmatrix} \tag{4–2}$$

which is a first-order system. However, applying the backward Euler method to equation (4–2) results in a linear system of size $2n \times 2n$ where n is the dimension of $\mathbf{x}$. A fairly simple transformation allows us to reduce the size of the problem to solving an $n \times n$ linear system instead. It is important to note that both the $2n \times 2n$ and $n \times n$ systems will have the same degree of sparsity, so solving the smaller system will be faster.

The $n \times n$ system that needs to be solved is derived as follows. Let us simplify notation by writing $\mathbf{x_0} = \mathbf{x}(t_0)$ and $\mathbf{v_0} = \mathbf{v}(t_0)$. We also define $\Delta\mathbf{x} = \mathbf{x}(t_0 + h) - \mathbf{x}(t_0)$ and $\Delta\mathbf{v} = \mathbf{v}(t_0 + h) - \mathbf{v}(t_0)$. The backward Euler update, applied to equation (4–2), yields

$$\begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{v} \end{pmatrix} = h\begin{pmatrix} \mathbf{v_0} + \Delta\mathbf{v} \\ \boldsymbol{f}(\mathbf{x_0} + \Delta\mathbf{x}, \mathbf{v_0} + \Delta\mathbf{v}) \end{pmatrix}. \tag{4–3}$$

Applying a Taylor series expansion to $\boldsymbol{f}$—which in this context is a function of both $\mathbf{x}$ and $\mathbf{v}$—yields the first-order approximation

$$\boldsymbol{f}(\mathbf{x_0} + \Delta\mathbf{x}, \mathbf{v_0} + \Delta\mathbf{v}) = \mathbf{f_0} + \frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}\Delta\mathbf{x} + \frac{\partial \boldsymbol{f}}{\partial \mathbf{v}}\Delta\mathbf{v}.$$

In this equation, the derivative $\partial\boldsymbol{f}/\partial\mathbf{x}$ is evaluated for the state $(\mathbf{x_0}, \mathbf{v_0})$ and similarly for $\partial\boldsymbol{f}/\partial\mathbf{v}$. Substituting this approximation into equation (4–3) yields the linear system

$$\begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{v} \end{pmatrix} = h \begin{pmatrix} \mathbf{v_0} + \Delta\mathbf{v} \\ (\mathbf{f_0} + \frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}\Delta\mathbf{x} + \frac{\partial \boldsymbol{f}}{\partial \mathbf{v}}\Delta\mathbf{v}) \end{pmatrix}. \tag{4–4}$$

Taking the bottom row of equation (4–4) and substituting $\Delta\mathbf{x} = h(\mathbf{v_0} + \Delta\mathbf{v})$ yields

$$\Delta\mathbf{v} = h\left(\mathbf{f_0} + \frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}h(\mathbf{v_0} + \Delta\mathbf{v}) + \frac{\partial \boldsymbol{f}}{\partial \mathbf{v}}\Delta\mathbf{v}\right).$$

Letting $\mathbf{I}$ denote the identity matrix, and regrouping, we obtain

$$\left(\mathbf{I} - h\frac{\partial \boldsymbol{f}}{\partial \mathbf{v}} - h^2\frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}\right)\Delta\mathbf{v} = h\left(\mathbf{f_0} + h\frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}\mathbf{v_0}\right) \tag{4–5}$$

which we then solve for $\Delta\mathbf{v}$. Given $\Delta\mathbf{v}$, we trivially compute $\Delta\mathbf{x} = h(\mathbf{v_0} + \Delta\mathbf{v})$.

The above assumes that the function $\boldsymbol{f}$ has no direct dependence on time; in the case that $\boldsymbol{f}$ varies directly with time (for example, if $\boldsymbol{f}$ describes time-varying external forces, or references moving points or coordinate frames that are not variables of $\mathbf{x}$) then equation (4–5) needs an additional term to account for this dependence:

$$\left(\mathbf{I} - h\frac{\partial \boldsymbol{f}}{\partial \mathbf{v}} - h^2\frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}\right)\Delta\mathbf{v} = h\left(\mathbf{f_0} + h\frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}\mathbf{v_0} + \frac{\partial \boldsymbol{f}}{\partial t}\right) \tag{4–6}$$

References

[1] D. Baraff and A. Witkin. Large steps in cloth simulation. *Computer Graphics* (*Proc. SIGGRAPH*), 1998.

[2] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes*. Cambridge University Press, 1986.