An Introduction to Simulation

Lagrangian simulation of spring-mass systems

Niels Joubert

2009-04-30

Abstract

Simulation of physical phenomenon - so-called Physically Based Animation - is of crucial interest in the Computer Graphics community, since it enables the creation of effects beyond the ability of artists to fake, and interactions beyond scripted or baked behavior. The believability and realism of environments and interactions are hugely affected by robust, accurate physical simulation. Thanks to Nuttapong Chentanez, Sebastian Burke, James Andrews, Daniel Ritchie and Dr. Carlo Sequin for their help in preparing these notes.

Contents

1 Problem Setup & Intuitive Explanation

Consider a snapshot of a ball being thrown into the air. At any point in time, the movement, or **dynamics**, of the ball is characterized by what we call it's **state**. If we could freeze-frame the world and look at the dynamics of the ball, we could identify its state as its mass, its position in space and the velocity it is moving at. Given this snapshot in time, Newton's Three Laws of Motion explain how the object moves.

1. Items remain in their current state of motion unless a force is applied to it.
2. The relationship between the force on an object and its acceleration is given by $F = ma$.
3. For every force on an object, the object exerts an equal and opposite force back.

So, if we un-freeze the world, Newton's first law states that we expect the ball to keep moving in the direction it's currently moving. Of course we know that gravity is a constant force between objects and the earth, so the ball feels this force. Newton's second law relates this constant force to how the ball accelerates. Specifically:

$$\vec{a} = \frac{\vec{F}}{m} \tag{1}$$

Thus, the acceleration of the ball depends on the force, scaled down by the mass. Intuitively, a large boulder is a lot harder to push around than a soccerball.

Let's consider the ball. **At any point in time**, the ball's state consists of its velocity, position and mass. Our environment (the world) also applies a force to it - in our case, this force is just gravity. Given all this, we want to calculate how the ball moves.

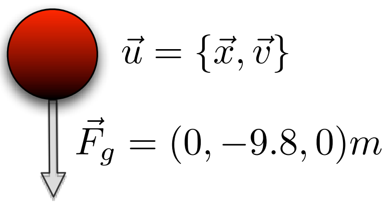

Figure 1: The state $\vec{u}$ of and the force $\vec{F}_g$ on a falling ball.

We also consider another interesting setup. Rather than a ball just falling under gravity, let's consider a ball being dropped, but the ball is connected to the ceiling with a spring.

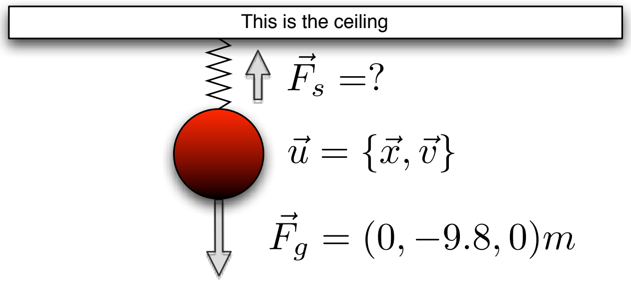

Figure 2: The state $\vec{x}$ of and the forces $\vec{F}_g$ and $\vec{F}_s$ on a falling ball.

How does this object move? We need to know how to calculate the net force on the objects at the current point in time to set up $F = ma$, which we will discuss in Modeling the System. We also want to know how to compute the state at the next point in time, which we will discuss in Solving the System.

2 Modeling the System - Calculating Forces

We model our system as a vector of positions and velocities: $\vec{u} = \{\vec{x}, \vec{v}\}$. The first step in simulation is to find $\vec{F}$ so that we can relate the forces on an object to how it's movement is changing. We want to set up the equation $\vec{a} = \frac{\vec{F}}{m}$. As a reminder, the $\vec{F}$ we want to calculate is the **total force** on the object, so we sum all the forces together.

2.1 Gravity

In the simple case of gravity, this is really easy. Gravity is a constant force, always equal to $\vec{F_g} = (0, -9.8, 0)m$ where $-y$ points towards the earth and m is the mass of the object.

2.2 Springs & Hook's Law

Springs do not exert a constant force. Consider a rubber band. If you hold it between your hands without stretching it, you feel no force. As you move your hands apart to stretch out the band, you feel increasing resistance from the rubber band. We want to capture this behavior, so we need to know both how the force scales with extension of the spring, and what the rest length of the spring is:

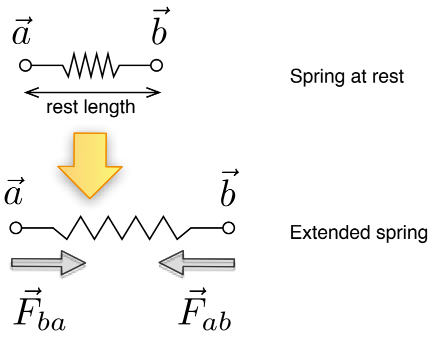

Figure 3: A spring at rest length, and the same spring extended beyond rest length, causing forces attempting to pull its endpoints back to rest length.

The direction of the force is along the spring itself - that is, it is along the vector between the two endpoints. The magnitude of the force is given by hook's law, which states that the force is proportional to the linear extention from the rest length rl:

$$\boxed{-\vec{F}_{ab} = \vec{F}_{ba} = \vec{F}_s = k_s((\vec{b} - \vec{a}) - rl)} \tag{2}$$

So we have now calculated the net force on a ball hanging from the ceiling by a spring. Gravity is pulling down by $(0, -9.8, 0)$ and the spring is pulling back up by $k_s(\vec{b} - \vec{a})$ where $\vec{b}$ is the ball's position and $\vec{a}$ is the attachment point to the ceiling.

As an additional note, we usually model springs to have not only an **elastic** force as we've already described, but to also have a **damping** force that resists changes in velocity. This models such phenomenon as shock absorbers, and in fact most of the spring-like objects around you also experience some damping effects. This damping is **local** to the two particles connected by a spring. The full spring equation (for a zero-length spring) becomes:

$$\vec{l} = \vec{b} - \vec{a} \tag{3}$$

$$\vec{F_s} = \left[k_s(\vec{l} - rl) + k_s \frac{\dot{l} \cdot \vec{l}}{|\vec{l}|}\right] \tag{4}$$

3 Solving the System

We know where the ball currently is (it's state) and we know the forces on it. Let's say the current time is t and we want to know where the ball will be some h amount of time later. We call this value h the **timestep**, and we use the terminology "taking a timestep" to mean moving forward in time by h amount.

3.1 Moving forward in time

We know the current velocity and position of the ball, and we find the acceleration through Newton's Second Law. How do we find the velocity and position of the ball after h time? Given the system's state u_t how do we find u_{t+h} First, let me remind you of the relationship between position, velocity and acceleration:

$$\text{velocity } v = \dot{\mathbf{x}} = \frac{dx}{dt} = \text{instantaneous change in of position} \tag{5}$$

$$\text{acceleration } a = \ddot{\mathbf{x}} = \frac{dv}{dt} = \frac{d^2x}{dt^2} = \text{instantaneous change in velocity} \tag{6}$$

Let's make an assumption to help us solve this problem. We know that the forces on the object is causing it to accelerate. Let's assume all the acceleration is applied immediately, at the current time. For the duration of the timestep, then, the velocity remains constant:

Assumption 1: *Assume the acceleration is constant over the timestep, so we can apply the total acceleration immediately, then use this new constant velocity over the course of the timestep.*

What is the new velocity after we apply the acceleration?

$$\Delta v = ha = h\frac{dv}{dt} \tag{7}$$

$$v_{t+h} = v_t + \Delta v \tag{8}$$

$$\therefore \boxed{v_{t+h} = v_t + ha} \tag{9}$$

We would like to take the current velocity, and add the change in velocity caused by acceleration. Well, acceleration does tell us how velocity changes, and acceleration is in units of "change in velocity" per time unit. Since the acceleration is *per time unit* we need to somehow account for the fact that we might be taking a timestep that is not the length of just one time unit. So we scale acceleration by the length of our timestep h, and we get something in units of "change in velocity" for the length of this timestep. The units match, and we can add this change in velocity to the current velocity to get the new velocity.

What about the new position?

$$\Delta x = hv_t \tag{10}$$

$$x_{t+h} = x_t + \Delta x \tag{11}$$

$$\therefore \boxed{x_{t+h} = x_t + hv_t} \tag{12}$$

Well, since velocity is now constant throughout the duration of the timestep, you should be able to guess how to find the new position. For example, a ball at position 0 moving at 5 meters per second, will be at position 5 after the first second, position 10 after the second second. Why? Because it moved at 5 meters per second for 2 seconds, it will be 10 meters from the original position. Do you use the velocity at the beginning or the end of the timestep? It doesn't really matter, it turns out, since assumption 1 is an approximation of what is really going on in both cases.

3.2 Stability Issues

What does assumption 1 do to our simulation? Consider the approximation we made when we assumed that *acceleration is applied instantly* - we assumed that any change in both the magnitude and the direction of the velocity happens instantly, and then we *traveled at a constant speed in a straight line* for the rest of the timestep! Why does this not work? The ball dropping might not be the most obvious example, so let's consider a ball attached to a spring, swinging around a fixed point:[1]

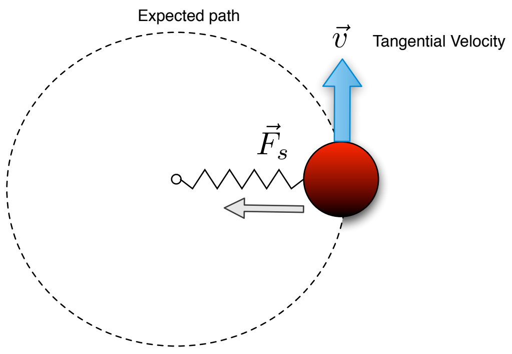

Figure 4: A ball rotating around a point. The spring force $\vec{F}_s$ pulls inward while velocity moved tangential to the path (perpendicular to the radius)

If we ask where the ball is after a timestep of length h, using our previous method, we'll calculate that the net force changes the velocity to point more towards the center, and we'll move the ball in a straight line defined by the direction of the velocity vector. But since we're moving along a straight line, you won't end up on the expected path! Even worse, the velocity will now be larger! Over time, the distance from the center to the ball will increase, so the spring would have extended, and the spring force $\vec{F}_s$ will be larger. We say that we've made an *error* in our approximation of the ball's position. This error will compound as we keep moving, every time getting larger and larger, until the position is completely wrong.

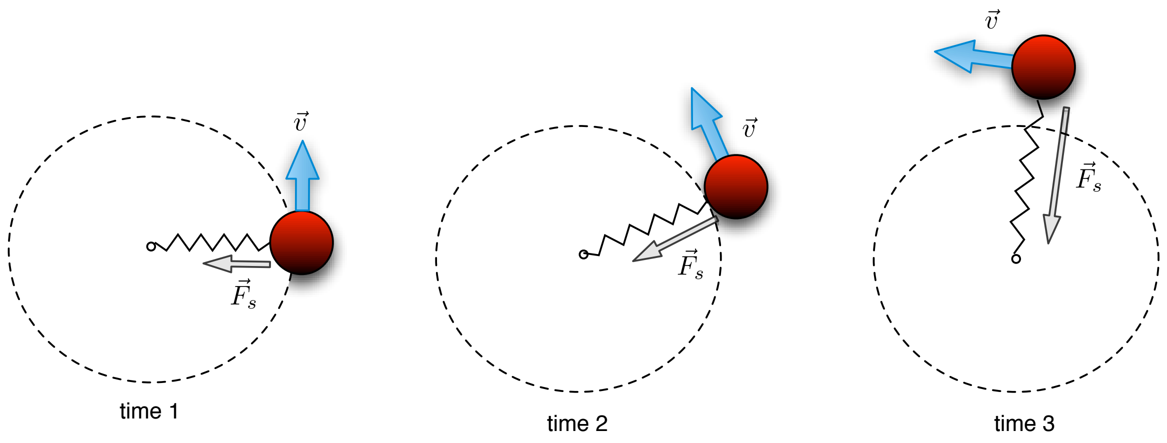

Figure 5: As the ball rotates, the position is moved along a straight line and the velocity is not correctly updated, causing our approximation of the ball to spiral out of control rather than follow the circular path.

[1]If this example seems contrived, consider a satellite moving around the earth - it works exactly like this.

3.2.1 Proving Instability

To prove to you that the new position and velocity makes no sense given that the ball should be moving in a circle, I'll sketch out the calculations for the new velocity and position and show that the velocity is increasing. To do this we'll work with figure 4 and the x and y components of position p and velocity v:

Prove instability. Start with position $[p_x, p_y]$ and velocity $[v_x, v_y]$ where, in our case, $v_x = 0$. Take timestep h:

1. $v' = [v_x + h * F_{sx}/m,\ v_y]$ where v_y stays the same since the force is along the x axis.
2. $p' = [p_x + hv_x,\ p_y + hv_y] = [p_x,\ p_y + hv_y]$

Notice the length of v' is greater than the length of v, and that p' does not fulfill the constraint that $\parallel p' \parallel =$ radius. The explicit integrator thus adds energy to the system, which violates the first law of thermodynamics. $\square$

3.3 Integration Methods

3.3.1 Explicit Euler

We've been concerned with finding the new position and velocity of an object given its current position, velocity and acceleration. We also know that acceleration and velocity is the first and second derivative of position. So to find a new position and velocity, what we really want to do is **integrate** velocity and acceleration:

$$\vec{v}_{t+h} = \vec{v}_t + \int_t^{t+h} \dot{v} \qquad \vec{x}_{t+h} = \vec{x}_t + \int_t^{t+h} \dot{x} \tag{13}$$

What we've done so far is to say that the integral is equal to the timestep duration h times the (constant) value of the function we're integrating at the current point in time. We can visually graph that as approximating the graph with a rectangle of width h and height taken from the value of the function we're integrating:

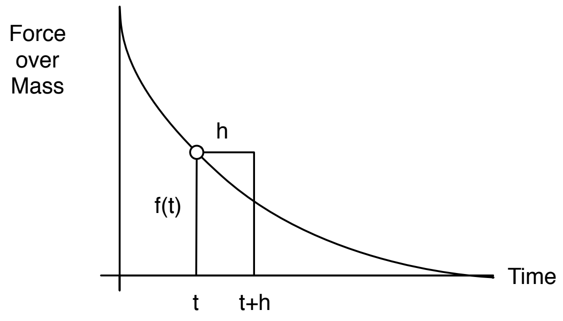

Figure 6: Explicit Euler integration. Notice the area difference between the curve and the rectangle.

This kind of integration is called **Explicit Euler integration**, and is the simplest form of numerical integration you'll find. It transforms an integral into a multiplication, using the value of the function at the beginning of the timestep to approximate the area.[2]

$$\boxed{\vec{v}_t + \int_t^{t+h} \dot{v} \quad \longrightarrow \quad \vec{v}_t + ((t+h) - t)\dot{v}} \tag{14}$$

[2]Notice that this method is the same as those Reimann Sums you calculated back in calculus class.

3.3.2 Implicit Euler

We would like to do better than our previous method by not overestimating the area as badly. Here I will suggest that we use the force at the *end* of the timestep. Notice that this underestimates the area, and thus I don't add energy. Let's be precise in the force we're looking for: Let's find a *future position* such that the force on the object at that position is such that this force will take me from where I am now to this new position. (See the appendix for a mathematical explanation of this statement.)

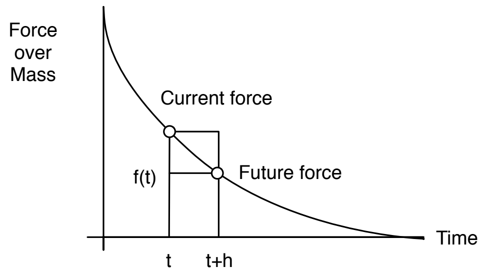

Figure 7: Explicit Euler integration versus Implicit Euler integration. Notice that Implicit uses the force at the future position while Explicit uses the force at the current position.

To do this, what we want to know is the force in the future. In figure 7, we want to calculate the "Future force" from the current force and the current state. Calculating this future force seems like circular logic - to get the future force, don't I need the future position, and if we had that, why are we bothering with integration? Luckily, we *can* accurately find *or* approximate the future force!

Why does this formulation work any better than our previous integrator? Let's consider our simple ball-spring system and attempt to move it 100 years into the future, then move back in time to our current position. Explicit Euler will apply 100 years worth of acceleration and move off in a straight line at some ridiculous speed, ending up at a completely nonphysical place. Our spring force is now absurdly large, and things only get worse from here. Implicit Euler, on the other hand, finds the future position such that the force there can move me between my current and this future position. This is equivalent to saying "find me a position so that, if we move back in time by applying the negated acceleration, we get back to the starting positon". (Take some time to absorb that statement). What position would this be? The further we move away, the larger the force becomes on the mass, thus no far away point (such as the position produces by Explicit Euler) can satisfy this statement. A position close to the rest length of the spring can satisfy this statement since it will have a small force on it. So even though we're moving 100 years into the future in a single timestep, we have such a small acceleration that, if we move back from that position using the acceleration at that position, we can get back to where we were initially. This reasoning shows you that implicit Euler is **unconditionally stable**.

3.3.3 Semi-Implicit Euler

What's the big problem here? **We don't know what the force in the future is!** But we solved the same problem when we discussed Inverse Kinematics. We have to make an approximation, and to do so we use the first order Taylor expansion to **linearize** the force equation. This linearized version of the implicit integrator is known as a **Semi-Implicit Euler Integrator**. Our approximation states that you can fit a straight line to the force curve, and move along this line to approximate the force in the future. The slope of this line is defined by the derivative of the force function, as shown in figure 8.

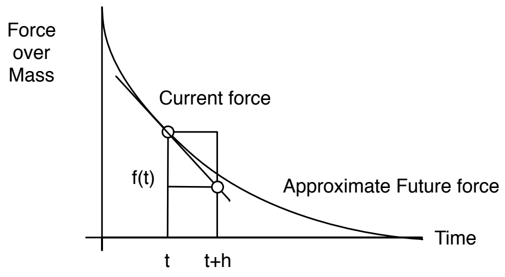

Figure 8: Using the Taylor expansion to approximate the force in the future.

4 Appendix

4.1 Explicit and (Semi) Implicit Euler Integration Derivation

We want to integrate an ordinary differential equation of the form:

$$\dot{x} = f(x) \tag{15}$$

I present here one possible derivation of the implicit and explicit Euler integration schemes, and make the approximation of implicit integration to find semi-implicit integration. Both of these schemes start with the following observation:

Observation 1: *The derivative of a function can be discretized by looking at the future or the past.*

In mathematical terms, using subscripts to indicate the discretized time value, we can write the derivative at a point as being either of the following:

$$\dot{x_i} \approx \frac{x_{i+1} - x_i}{h} \qquad \text{Forward Euler} \tag{16}$$

$$\dot{x_i} \approx \frac{x_i - x_{i-1}}{h} \qquad \text{Backwards Euler} \tag{17}$$

But let's say that we **have** the derivative, and we're interested in these discretized points, x_i, x_{i-1} and x_{i+1}. We will now derive the formula to get the "next" x value from the "current" x value for both the Forward and Backward cases. As a note, Forward Euler and Explicit Euler is considered to be the same, as is Backward Euler and Implicit Euler.

4.1.1 Explicit Euler / Forward Euler

For this case, we are at x_i and we want to move to x_{i+1}, so we rewrite equation 16:

$$\dot{x_i} \approx \frac{x_{i+1} - x_i}{h} \tag{18}$$

$$\therefore x_{i+1} \approx x_i + h\dot{x_i} \tag{19}$$

Notice how equation 19 is the same as equation 12 and 9 we came up with when we were originally attempting to calculate new velocities and positions given the acceleration (recall acceleration is the derivative of velocity, thus we were solving the differential equation $\dot{v} = \frac{\vec{F}}{m}$ given by Newton's Second Law). Thus we need to do no more work to show that forward Euler approximates the next position based on all the information we already have at the current position.

4.1.2 Implicit & Semi-Implicit Euler / Backward Euler

For the case of equation 17, we are currently at x_{i-1} and want to move to x_i. We start by rewriting equation 17:

$$\dot{x_i} \approx \frac{x_i - x_{i-1}}{h} \tag{20}$$

$$\therefore x_i \approx x_{i-1} + h\dot{x_i} \tag{21}$$

At first glance, equation 21 looks just like the forward Euler step in equation 19. The critical difference is that the derivative of the function, $\dot{x_i}$, is at time i but we're still at time $i-1$ in our simulation. Calculating the derivative of, say, velocity at the current time is easy - that's what Newton's Second Law gives us. But how do we calculate the derivative of a function in the future?

A fully implicit integrator uses a scheme such as Newton's Method to find the value of the derivative of $\dot{x_i}$ from the value that we can calculate, $\dot{x_{i-1}}$. More specifically, we have a function that gives us the derivative of our variable: $\dot{x} = \vec{F}(x)$, where this function depends on the current x.[3]. This function tends to be nonlinear, and we can use several rootfinding techniques to find the value of $\vec{F}(x)$ at some future x.

If you look at what we've just done, you'll notice that we've chosen the next position in such a way that the force on that point brings you there from the previous position. This is where you see why it's called backwards Euler - the force is such that if you run the simulation *backwards*, you can use the force on x_i to move directly to x_{i-1}. Intuitively, this presents a reasoning for why we're interested in this integration scheme. There is no way that x_i can spiral off to some crazy value, it has to go to such a position that the force on it "makes sense" - that the force allows you to get back to where you were. Specifically, Implicit Euler is **unconditionally stable** on continuous, smooth functions.

Unfortunately, finding $\vec{F}(x_i)$ is computationally prohibitive. Since we usually don't want to spend the extra time in calculating a precise value for $\dot{x_i}$, the future value of the derivative of our variable, we use the **Taylor Series** to make an approximation of our function. Making this approximation will give us the equations for a **Semi-Implicit Integrator**, which we will now derive from equation 21:

$$x_i \approx x_{i-1} + h\dot{x_i} \tag{22}$$

$$\therefore x_i \approx x_{i-1} + h\vec{F}(x) \tag{23}$$

We now apply a first order Taylor expansion to $\vec{F}(x_{i-1})$ to find (x_i), keeping in mind that $\vec{F}$ is a vector, thus it's partial is a **Jacobian Matrix**:

$$\vec{F}(x_i) \approx \vec{F}(x_{i-1}) + \frac{\partial \vec{F}(x_{i-1})}{\partial x}(x_i - x_{i-1}) \tag{24}$$

[3]This is the definition of an ordinary differential equation

And substitute this expansion into the backwards Euler equation and solve for x_i:

$$\begin{aligned}
x_i &\approx x_{i-1} + h\left[\vec{F}(x_{i-1}) + \frac{\partial \vec{F}(x_{i-1})}{\partial x}(x_i - x_{i-1})\right] && (25)\\
x_i &\approx x_{i-1} + h\vec{F}(x_{i-1}) + hx_i\frac{\partial \vec{F}(x_{i-1})}{\partial x} - hx_{i-1}\frac{\partial \vec{F}(x_{i-1})}{\partial x} && (26)\\
x_i - hx_i\frac{\partial \vec{F}(x_{i-1})}{\partial x} &\approx x_{i-1} - hx_{i-1}\frac{\partial \vec{F}(x_{i-1})}{\partial x} + h\vec{F}(x_{i-1}) && (27)\\
x_i\left[I - h\frac{\partial \vec{F}(x_{i-1})}{\partial x}\right] &\approx x_{i-1}\left[I - h\frac{\partial \vec{F}(x_{i-1})}{\partial x}\right] + h\vec{F}(x_{i-1}) && (28)\\
x_i &\approx x_{i-1} + h\vec{F}(x_{i-1})\left[I - h\frac{\partial \vec{F}(x_{i-1})}{\partial x}\right]^{-1} && (29)
\end{aligned}$$

What we've done is derived a formula for x at time i using the values we can find at time $i-1$. This is, naturally, just an approximation, so we've lost the property of being **unconditionally stable**, but it will still be far better than our initial Explicit Euler equation, allowing us to take bigger timesteps.

Notice the derivative we take in equation 29: $\frac{\partial \vec{F}(x_{i-1})}{\partial x}$. This is a **Jacobian**, since it contains **all the first order partial derivatives of this function with respect to its independent variables**. This comes from the Taylor expansion's first order term, which we can intuitively explain. We currently have all the information at time $i-1$, and we find the derivative of the function at this time, and move along it in a straight line to get to to the approximate value of the function at a time in the future. To create a straight line, we need a slope. A slope is simply a derivative, and the Jacobian contains all the necessary derivatives to create this "straight line" along which we move to find the future value of the function $\vec{F}$. The Jacobian forms a matrix that encodes the change in force on each coordinate of each mass with respect to the change in position of each coordinate of each other mass. As you can imagine, for most of the masses, this value is 0 - the force on most masses is <u>not</u> affected by the movement of other points. A spring is an example of a coupling between points that causes nonzero entries in the Jacobian.[4]

4.2 Implicit versus Explicit

Here we present two other ways of analyzing the advantages of a implicit integrator over an explicit integrator.

4.2.1 Energy-level Reasoning

We can approach the fully implicit integrator on an energy level to understand why it's a better approach for simulation than the explicit integrator. The second law of thermodynamics state that "entropy increases over time" - that is, kinetic and potential energy is dissipated through friction and collisions into heat, sound and so forth. Thus we *expect* our system to lose energy. If we integrate the energy curve (and energy is related to force) our implicit integrator will always underestimate the energy in the system, and we will dissipate energy. This is *more* physically correct than adding energy, since we expect the second law of thermodynamics to hold. By dissipating energy we introduce some global **damping**, which is preferred over adding energy since we expect energy to dissipate in our system anyway, and having a simulation damp out is much better than having a simulation become unstable and blow up. This shows why the implicit integrator's solution is plausible.

[4]For more explanation of the Jacobian, see my course notes on Inverse Kinematics, http://njoubert.com/teaching/

4.2.2 Low-Pass Filter Reasoning

We will not cover this approach in detail in these notes, but it is interesting to note that you can show that Semi-Implicit Euler is equal to a low pass filtered version if Explicit Euler by writing the change in position as a sum of the explicit change in position and a Jacobian times the implicit change in position, and factoring the Jacobian matrix into a filter term:

$$\vec{F}(x_{t+h}) \approx \vec{F}(x_t) + h\frac{\partial \vec{F}(x_t)}{\partial x} \tag{30}$$

$$\therefore \Delta x \approx \frac{h}{M}\vec{F}(x_t) + \frac{h}{M}\frac{\partial \vec{F}(x_t)}{\partial x} \tag{31}$$

$$\text{Let } J = \frac{\partial \vec{F}(x_t)}{\partial x} \tag{32}$$

$$\therefore \Delta x = \Delta x_{\text{explicit}} + J\Delta x_{\text{implicit}} \tag{33}$$

$$\therefore \Delta x_{\text{implicit}} = (I - J)^{-1}\,\Delta x_{\text{explicit}} \tag{34}$$

If you look at the change in position as a vibration, then $(I - J)^{-1}$ acts as a low-pass filter, removing high frequency changes in position.

4.3 Error in Explicit Euler Integration

We initially said that our simple forward Euler integration scheme will give us an incorrect answer to the future position. How big is this error? We can use the Taylor expansion to quantify the error. The full Taylor Expansion states that:

$$\vec{F}(x) = \sum_{0}^{\infty} \frac{\vec{F}^{(n)}(a)}{n!}(x-a)^n \tag{35}$$

We truncate this equation at $n = 1$ to get the forward Euler equation:

$$\vec{F}(x_{i+1}) \approx \vec{F}(x_i) + \dot{\vec{F}}(x_i)(x_{i+1} - x_i) \tag{36}$$

Thus the error is completely contained in all the higher order terms of the Taylor expansion:

$$\text{error}(\vec{F}(x)) = \sum_{2}^{\infty} \frac{\vec{F}^{(n)}(a)}{n!}(x-a)^n \tag{37}$$

Since the error is dominated by the first term, there's plenty of integration schemes that also attempt to provide for the second term. It is important to note that implicit Euler integration's error magnitude is *the same* ad the explicit Euler integrator, but it <u>underestimates</u> rather than *overestimates* the integral. More sophisticated integration schemes are available, each with their own properties. Remember, always apply the right tool for the job, regardless of how the tool is viewed!

5 References

Much of this material was adapted from the following sources:

- Physically Based Modeling, Online SIGGRAPH 2001 course notes, http://www.pixar.com/companyinfo/research/pbm2
- Implicit Integration - the linear case. http://www.mech.gla.ac.uk/ peterg/software/MTT/examples/Simulation_rep/node89.html
- Large Steps in Cloth Simulation, Baraff Witkin, Siggraph 1998