1 One Particle Central Force

1.1 Acceleration Vector

Suppose a single particle of mass $m$ is at position $\vec{x}$, and is attracted gravitationally to a single fixed particle of mass $M$ at position $\vec{x}_0$. There is a force vector on $\vec{x}$, denoted $\vec{a}$. The magnitude of acceleration is

$$k = \frac{GM}{\|\vec{x}_0 - \vec{x}\|^2}$$

The direction of acceleration is from $\vec{x}$, to $\vec{x}_0$, so it points in the direction of $\vec{x}_0 - \vec{x}$. One way to remember the sign is to think of putting $\vec{x}$ at the origin. We have the magnitude and direction of the acceleration, so we can find the vector itself:

$$\vec{a} = GM \frac{\vec{x}_0 - \vec{x}}{\|\vec{x}_0 - \vec{x}\|^3}$$

The quantity $GM$ has units of length cubed per time squared, and it’s the quantity that is easily experimentally measurable. $G$ is known to four or five digits, but $GM$ (G times the mass of the sun) is known to ten or eleven digits! $GM$ is called the standard gravitational parameter for a given body.

As an aside, how do you do the same process if instead of one free body you have $N$ free bodies interacting with each other? Sum the forces!

$$\vec{a}_i = \sum_{j=1, j\neq i}^{N} GM_j \frac{\vec{x}_j - \vec{x}_i}{\|\vec{x}_j - \vec{x}_i\|^3}$$

The distance from particle $i$ to particle $j$ is the same as the distance from $j$ to $i$, and we ignore the $j = i$ case, so there are actually only $(N \choose 2) = N(N - 1)/2$ calculations. Try plugging in $N = 10^{10}$, the particle number in some modern simulations. Not even modern supercomputers can run the brute force method!!

1 A back of the envelope estimate tells me, with a few tens of teraflops of computing power, it would take months to half a year to do finish all of these distance calculations = one timestep. The thing is you need thousands of timesteps to get a useful result.
1.2 Numerics

Let’s run a simulation of our one body orbiting around a sun using Euler’s method. I’m going to omit vector arrows, but keep in mind that \(x, v,\) and \(a(x)\) are all vectors here. We want to solve the second order ODE \(\ddot{x} = a(x).\) This is equivalent to solving the two first order ODEs \(\dot{x} = v, \dot{v} = a(x).\) We have initial conditions \(v_0\) and \(x_0\) at \(t = 0.\) Let’s use a fixed timestep \(\delta t,\) and let \(x_n\) denote the position of the particle at time \(n\delta t.\)

One way to approximate the solution is using the Euler scheme:

\[
\dot{x} \approx \frac{x_{n+1} - x_n}{\delta t} = v_n \quad \dot{v} \approx \frac{v_{n+1} - v_n}{\delta t} = a(x_n)
\]

Then...

\[
x_{n+1} = x_n + \delta t \cdot v_n \quad v_{n+1} = v_n + \delta t \cdot a(x_n)
\]

As we saw in the javascript applet in class, this is a pretty bad method. We should get an elliptical orbit, but the result isn’t elliptical at all!!

For a function \(f(t),\) the quantity \(\frac{1}{\delta t}(f(t + \delta t) - f(t))\) is a pretty bad approximation to the derivative at \(t,\) and the error is proportional to \(\delta t.\) You can prove this by using Taylor expansions:

\[
\frac{1}{\delta t}(f(t + \delta t) - f(t)) = f'(t) + O(\delta t)
\]

There has to be a better way, and the better way is the leapfrog method. We can get something accurate to second order in \(\delta t\) if we use a bit of intuition: when we find the slope of a line segment, we’re really finding a good approximation to the slope at the midpoint of the line segment.
You can prove the following formula if you Taylor expand both sides about $t$:

$$f(t + \delta t) - f(t) \approx f'(t + \frac{\delta t}{2})\delta t + O(\delta t^2)$$

Then we have the following approximations:

$$\dot{x} \approx \frac{x_{n+1} - x_n}{\delta t} = v_{n+1/2}$$

$$\dot{v} \approx \frac{v_{n+1/2} - v_{n-1/2}}{\delta t} = a(x_n)$$

which we turn into the Leapfrog numerical scheme:

$$v_{n+1/2} = v_{n-1/2} + \delta t \cdot a(x_n)$$

$$x_{n+1} = x_n + \delta t \cdot v_{n+1/2}$$

Leapfrog is a great algorithm for four reasons:

1. It’s second-order (whereas Euler is only first-order),
2. It only uses one force evaluation per timestep (force evaluations are expensive!),
3. It is time-reversible (unlike Euler),
4. It’s symplectic, meaning it conserves phase space volume.

The power of the method is demonstrated by the Javascript implementation we went over class.