# Point Masses and Force Fields

Donald Fussell

Computer Science Department
The University of Texas at Austin

October 28, 2014

## Particle Dynamics

- Simple classical dynamics - point masses moved by forces
- Point masses can model particles
- They can be grouped to model macroscopic objects
- Forces can move particles
- Forces can constrain point masses to be part of larger objects
- Used in particle systems, cloth, rigid bodies, etc.

## Particles

At time $t$, a particle has

- A position

$$\vec{\mathbf{x}}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$$

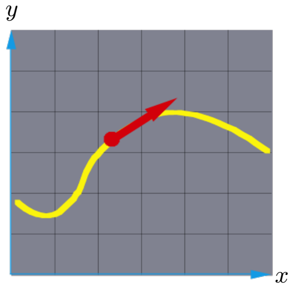
- A velocity

$$\vec{\mathbf{v}}(t) = \dot{\mathbf{x}}(t) = \begin{bmatrix} \frac{dx(t)}{dt} \\ \frac{dy(t)}{dt} \end{bmatrix}$$

- An acceleration

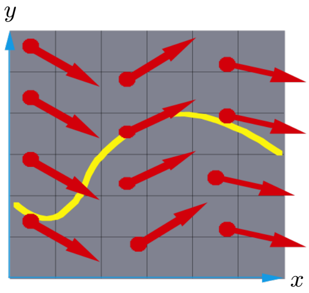
$$\vec{\mathbf{a}}(t) = \ddot{\mathbf{x}}(t) = \frac{d\vec{\mathbf{v}}(t)}{dt} = \frac{d^2\vec{\mathbf{x}}(t)}{dt^2} = \begin{bmatrix} \frac{d^2x(t)}{dt^2} \\ \frac{d^2y(t)}{dt^2} \end{bmatrix}$$

- And a mass $m$



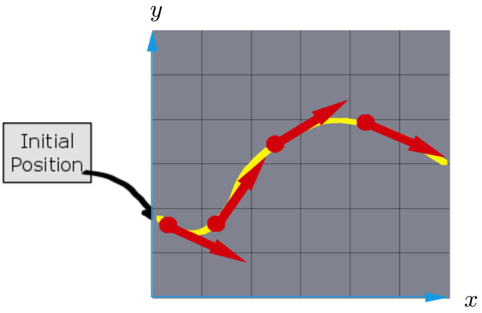
## Vector Fields

- Ignore acceleration and mass for now and let the velocity be dictated by a function $g$ of position and time, so $\dot{\mathbf{x}}(t) = g(\vec{\mathbf{x}}(t), t)$.
- At any time $t$, the function $g$ defines a **vector field** over $\vec{\mathbf{x}}$.
- This vector field specifies the velocity of a particle located at position $\vec{\mathbf{x}}(t)$ at time $t$.



## 1st-order ODEs

- The equation $\dot{\mathbf{x}}(t) = g(\vec{\mathbf{x}}(t), t)$ is a **first-order ordinary differential equation (ODE)**.
- We need to use it to compute the path that a particle takes through the vector field if it starts at a given initial position.
    - This path is called an **integral curve**.
    - The problem is called an **initial value problem**, a special case of a **boundary value problem**.



## Moving Particles

Given $\dot{\mathbf{x}}(t) = g(\vec{\mathbf{x}}(t), t)$, we want to solve for $x(t)$, the particle's trajectory.

We can do this numerically.

### Euler's Method

- Choose a discrete timestep $\Delta t$.
- Iterate, taking linear steps in the direction of the velocity at the beginning of each step.

$$\vec{\mathbf{x}}(t + \Delta t) = \vec{\mathbf{x}}(t) + \dot{\mathbf{x}}(t) \cdot \Delta t = \vec{\mathbf{x}}(t) + g(\vec{\mathbf{x}}(t), t) \cdot \Delta t$$

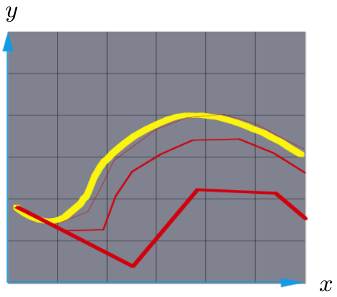
- In simpler timestep iteration notation

$$\vec{\mathbf{x}}_{i+1} = \vec{\mathbf{x}}_i + \vec{\mathbf{v}}_i \cdot \Delta t$$

This is the simplest (but not the best) way to numerically integrate a first-order ODE.

## Timestep Errors

- Problem: discrete steps in instantaneously valid directions
- Bigger steps, bigger errors
- Timestep error $\sim O(\Delta t^2)$, global error $\sim O(\Delta t)$
- Need to take pretty small steps, so maybe not efficient
- Step size needed depends on rate of velocity change
- Better methods allow bigger timesteps, we'll see some later



## Force Fields

For particle dynamics, we won't have a $g(\vec{\mathbf{x}}(t), t)$ to specify velocities directly, rather we'll compute velocities from the forces acting on particles

Since particles obey Newton's Law: $\vec{\mathbf{f}} = m\vec{\mathbf{a}} = m\ddot{\mathbf{x}}$

And since in general force (and thus acceleration) can depend on the position and velocity of a particle

We have

$$\vec{\mathbf{a}}(t) = \ddot{\mathbf{x}}(t) = \frac{\vec{\mathbf{f}}(\vec{\mathbf{x}}(t), \dot{\mathbf{x}}(t), t)}{m}$$

## 2nd-order ODEs

Equations with a second derivative in them (like $\ddot{\mathbf{x}}$) are called **second-order ODE**s.
Euler's Method and related techniques handle first-order ODEs.

### Turning higher-order ODEs into first-order ODEs

We have

$$\ddot{\mathbf{x}}(t) = \frac{\vec{\mathbf{f}}(\vec{\mathbf{x}}(t), \dot{\mathbf{x}}(t), t)}{m}$$

If we introduce the velocity variable $\vec{\mathbf{v}}(t)$, we can rewrite this as a pair of coupled first-order ODEs

$$\begin{bmatrix} \dot{\mathbf{x}}(t) = \vec{\mathbf{v}}(t) \\ \dot{\mathbf{v}}(t) = \frac{\vec{\mathbf{f}}(\vec{\mathbf{x}}(t), \vec{\mathbf{v}}(t), t)}{m} \end{bmatrix}$$

We can treat this as a first-order ODE in **phase space**

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{v}}(t) \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{v}}(t) \\ \frac{\vec{\mathbf{f}}(\vec{\mathbf{x}}(t), \vec{\mathbf{v}}(t), t)}{m} \end{bmatrix}$$

## Phase Space

We can concatenate vectors to form vectors in higher-dimensional space, called **phase space**. If $\vec{\mathbf{x}}$ and $\vec{\mathbf{v}}$ are each 3-dimensional, we get a 6-dimensional phase space.

$$\begin{bmatrix} \vec{\mathbf{x}}(t) \\ \vec{\mathbf{v}}(t) \end{bmatrix}$$

## Phase Space

We can concatenate vectors to form vectors in higher-dimensional space, called **phase space**. If $\vec{\mathbf{x}}$ and $\vec{\mathbf{v}}$ are each 3-dimensional, we get a 6-dimensional phase space.

$$\begin{bmatrix} \vec{\mathbf{x}}(t) \\ \vec{\mathbf{v}}(t) \end{bmatrix}$$

We can take time derivatives in phase space.

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{v}}(t) \end{bmatrix}$$

## Phase Space

We can concatenate vectors to form vectors in higher-dimensional space, called **phase space**. If $\vec{\mathbf{x}}$ and $\vec{\mathbf{v}}$ are each 3-dimensional, we get a 6-dimensional phase space.

$$\begin{bmatrix} \vec{\mathbf{x}}(t) \\ \vec{\mathbf{v}}(t) \end{bmatrix}$$

We can take time derivatives in phase space.

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{v}}(t) \end{bmatrix}$$

We can have vector fields in phase space, like our force field. (Yes, I know it sounds like Star Trek.)

$$\vec{\mathbf{f}}(\vec{\mathbf{x}}(t), \vec{\mathbf{v}}(t), t)$$

## Phase Space

We can concatenate vectors to form vectors in higher-dimensional space, called **phase space**. If $\vec{\mathbf{x}}$ and $\vec{\mathbf{v}}$ are each 3-dimensional, we get a 6-dimensional phase space.

$$\begin{bmatrix} \vec{\mathbf{x}}(t) \\ \vec{\mathbf{v}}(t) \end{bmatrix}$$

We can take time derivatives in phase space.

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{v}}(t) \end{bmatrix}$$

We can have vector fields in phase space, like our force field. (Yes, I know it sounds like Star Trek.)

$$\vec{\mathbf{f}}(\vec{\mathbf{x}}(t), \vec{\mathbf{v}}(t), t)$$

We can have ODEs in phase space too.

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{v}}(t) \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{v}}(t) \\ \frac{\vec{\mathbf{f}}(\vec{\mathbf{x}}(t), \vec{\mathbf{v}}(t), t)}{m} \end{bmatrix}$$

## Euler, with Force

We can solve $\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{v}}(t) \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{v}}(t) \\ \frac{\vec{\mathbf{f}}(\vec{\mathbf{x}}(t), \vec{\mathbf{v}}(t), t)}{m} \end{bmatrix}$ using Euler's Method.

### Euler's Method in Phase Space

- Choose a discrete timestep $\Delta t$ and iterate, as before.

$$\vec{\mathbf{x}}(t+\Delta t) = \vec{\mathbf{x}}(t) + \vec{\mathbf{v}}(t) \cdot \Delta t$$
$$\vec{\mathbf{v}}(t+\Delta t) = \vec{\mathbf{v}}(t) + \frac{\vec{\mathbf{f}}(\vec{\mathbf{x}}(t), \vec{\mathbf{v}}(t), t)}{m} \cdot \Delta t$$

- With timestep indices

$$\vec{\mathbf{x}}_{i+1} = \vec{\mathbf{x}}_i + \vec{\mathbf{v}}_i \cdot \Delta t$$
$$\vec{\mathbf{v}}_{i+1} = \vec{\mathbf{v}}_i + \frac{\vec{\mathbf{f}}_i(\vec{\mathbf{x}}_i, \vec{\mathbf{v}}_i)}{m} \cdot \Delta t$$

This is just what we did before except we're updating $\vec{\mathbf{v}}$ iteratively along with $\vec{\mathbf{x}}$.

## Better Methods

- Euler's Method is first-order, so
  - The error per timestep (local error) is proportional to $\Delta t^2$.
  - The cumulative (global) error is proportional to $\Delta t$.
- This isn't very good, small timesteps are often needed to keep the computation from diverging.
- This is a particular problem for **stiff systems**.
  - For our purposes, these are roughly systems with really stiff springs in them, or something exerting very strong forces that can change rapidly.
  - That's not a definition, which is much harder.
- A related problem is that Euler's Method is not **symplectic**. It does not conserve energy (or come close to doing so) as the simulation proceeds.

## Error Analysis

We said Euler's Method is local order $O(\Delta t^2)$ and global order $O(\Delta t)$. What do we mean'?

We simulate from $t_0$ to $T$ by taking $n$ timesteps of size $\Delta t$. The total error at time $T$, $e(\Delta t, T) = |\vec{\mathbf{x}}(T) - \vec{\mathbf{x}}_n|$, the difference between the correct position at $T$ and the simulated position after the $n$th timestep.

$e(\Delta t, T)$ is a function of both the timestep size $\Delta t$ and the total simulation time $T$. The order of an integration method describes how $e(\Delta t, T)$ changes as a function of $\Delta t$, holding $T$ constant.

## Error Analysis

Local order $O(\Delta t^2)$ means changing $\Delta t$ changes the worst-case truncation error proportionally to $\Delta t^2$. Global order $O(\Delta t)$ means a change in $\Delta t$ changes the worst case cumulative truncation error of the entire simulation proportionally to $\Delta t$.

This doesn't tell you much about how the simulation error grows as the overall simulation time $T$ grows. This depends on many factors, what ODEs you're integrating, the initial conditions, etc.

A method with a higher global order is better than a lower-order method because it gets worse more slowly as you increase the timestep size, not because you have any idea how far off the right answer it will get in the actual simulation.

## Symplectic Euler

Assume that $\vec{\mathbf{f}}$ is only a function of position and not of velocity, i.e. it is $\vec{\mathbf{f}}(\vec{\mathbf{x}}(t), t)$. We don't model wind resistance, for example, but we can still handle a lot of interesting forces.

Now we can easily fix Euler's Method to make it symplectic.

### Semi-implicit Euler Integration

- We can use $\vec{\mathbf{v}}_{i+1}$ instead of $\vec{\mathbf{v}}_i$ to compute $\vec{\mathbf{x}}_{i+1}$

$$\vec{\mathbf{v}}_{i+1} = \vec{\mathbf{v}}_i + \frac{\vec{\mathbf{f}}_i(\vec{\mathbf{x}}_i)}{m} \cdot \Delta t$$
$$\vec{\mathbf{x}}_{i+1} = \vec{\mathbf{x}}_i + \vec{\mathbf{v}}_{i+1} \cdot \Delta t$$

- Alternatively, we can use $\vec{\mathbf{x}}_{i+1}$ instead of $\vec{\mathbf{x}}_i$ to compute $\vec{\mathbf{v}}_{i+1}$

$$\vec{\mathbf{x}}_{i+1} = \vec{\mathbf{x}}_i + \vec{\mathbf{v}}_i \cdot \Delta t$$
$$\vec{\mathbf{v}}_{i+1} = \vec{\mathbf{v}}_i + \frac{\vec{\mathbf{f}}_i(\vec{\mathbf{x}}_{i+1})}{m} \cdot \Delta t$$

## Störmer-Verlet

Euler's Method can be obtained from the Taylor series for $\vec{\mathbf{x}}(t + \Delta t)$ and $\vec{\mathbf{v}}(t + \Delta t)$.

$$\begin{aligned}\vec{\mathbf{x}}(t + \Delta t) &= \vec{\mathbf{x}}(t) + \dot{\mathbf{x}}(t)\Delta t + O(\Delta t^2) \\ \vec{\mathbf{v}}(t + \Delta t) &= \vec{\mathbf{v}}(t) + \dot{\mathbf{v}}(t)\Delta t + O(\Delta t^2)\end{aligned}$$

We just throw away the quadratic and higher order terms, which is why the local error is proportional to $\Delta t^2$.

We can use similar derivations to get better methods.

For instance, the techniques we've seen so far aren't time-reversible, which proper physics should be. Let's try for that property.

## Störmer-Verlet

We can do Taylor series in both time directions

$$\vec{\mathbf{x}}(t+\Delta t) = \vec{\mathbf{x}}(t) + \dot{\mathbf{x}}(t)\Delta t + 1/2\ddot{\mathbf{x}}(t)\Delta t^2 + 1/6\dddot{\mathbf{x}}(t)\Delta t^3 + O(\Delta t^4)$$
$$\vec{\mathbf{x}}(t-\Delta t) = \vec{\mathbf{x}}(t) - \dot{\mathbf{x}}(t)\Delta t + 1/2\ddot{\mathbf{x}}(t)\Delta t^2 - 1/6\dddot{\mathbf{x}}(t)\Delta t^3 + O(\Delta t^4)$$

### Störmer-Verlet Integration

Add them and apply some algebra to get

$$\vec{\mathbf{x}}(t+\Delta t) = 2\vec{\mathbf{x}}(t) - \vec{\mathbf{x}}(t-\Delta t) + \ddot{\mathbf{x}}(t)\Delta t^2 + O(\Delta t^4)$$

Throw away the $O(\Delta t^4)$ local error term, write as an iteration, put in the forcing function for the acceleration, and we have

$$\vec{\mathbf{x}}_{i+1} = 2\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_{i-1} + \frac{\vec{\mathbf{f}}_i(\vec{\mathbf{x}}_i)}{m}\Delta t^2$$

We're given initial $\vec{\mathbf{x}}_0$ and $\vec{\mathbf{v}}_0$, but we also need $\vec{\mathbf{x}}_1$, so

$$\vec{\mathbf{x}}_1 = \vec{\mathbf{x}}_0 + \vec{\mathbf{v}}_0\Delta t + \frac{1}{2}\frac{\vec{\mathbf{f}}_i(\vec{\mathbf{x}}_0)}{m}\Delta t^2$$

## Störmer-Verlet

- Störmer-Verlet advantages
  - It is time-reversible, symplectic, local $O(t^4)$ and global $O(t^2)$, or second-order.
  - Easy to use for constraints between particles (e.g. infinitely stiff springs), especially for rigid bodies, cloth, etc. See Jakobsen's famous GamaSutra article http://www.gamasutra.com/resource_guide/20030121/jacobson_pfv.htm
- Störmer-Verlet disadvantages
  - Multistep method, needs adjustment for non-constant $\Delta t$.

$$\vec{\mathbf{x}}_{i+1} = \vec{\mathbf{x}}_i + (\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_{i-1})\frac{\Delta t_i}{\Delta t_{i-1}} + \frac{\vec{\mathbf{f}}_i(\vec{\mathbf{x}}_i)}{m}\frac{\Delta t_i + \Delta t_{i-1}}{2}\Delta t_i$$

  - Doesn't calculate velocities.

## Velocity Verlet

If we need velocities, we can just calculate them from the Störmer-Verlet point positions.

$$\vec{\mathbf{v}}_i = \frac{\vec{\mathbf{x}}_{i+1} - \vec{\mathbf{x}}_{i-1}}{\Delta t_i + \Delta t_{i-1}} + O(\Delta t^2) \qquad \text{or} \qquad \vec{\mathbf{v}}_{i+1} = \frac{\vec{\mathbf{x}}_{i+1} - \vec{\mathbf{x}}_i}{\Delta t_i} + O(\Delta t)$$

More commonly, they are incorporated into the iteration as the

### Velocity Verlet Method

Expand $\vec{\mathbf{x}}(t+\Delta t) = \vec{\mathbf{x}}(t) + \vec{\mathbf{v}}(t)\Delta t + \frac{1}{2}\vec{\mathbf{a}}\Delta t^2 + O(\Delta t^3)$

Expand forward $\vec{\mathbf{v}}(t+\Delta t) = \vec{\mathbf{v}}(t) + \vec{\mathbf{a}}(t)\Delta t + O(\Delta t^2)$
and reverse $\vec{\mathbf{v}}(t) = \vec{\mathbf{v}}(t+\Delta t) - \vec{\mathbf{a}}(t+\Delta t)\Delta t + O(\Delta t^2)$

Add the latter two to get $\vec{\mathbf{v}}(t+\Delta t) = \vec{\mathbf{v}}(t) + \frac{1}{2}(\vec{\mathbf{a}}(t) + \vec{\mathbf{a}}(t+\Delta t))\Delta t$

So we have

$$\vec{\mathbf{x}}_{i+1} = \vec{\mathbf{x}}_i + \vec{\mathbf{v}}_i\Delta t + \frac{1}{2}\frac{\vec{\mathbf{f}}_i(\vec{\mathbf{x}}_i)}{m}\Delta t^2$$

$$\vec{\mathbf{v}}_{i+1} = \vec{\mathbf{v}}_i + \frac{\vec{\mathbf{f}}_i(\vec{\mathbf{x}}_i) + \vec{\mathbf{f}}_{i+1}(\vec{\mathbf{x}}_{i+1})}{2m}\Delta t$$