Game Physics

Euler's Method

CS 415: Game Development

Professor Eric Shaffer

Source: Millington, Ian. Game Physics Engine Development, Second Edition.

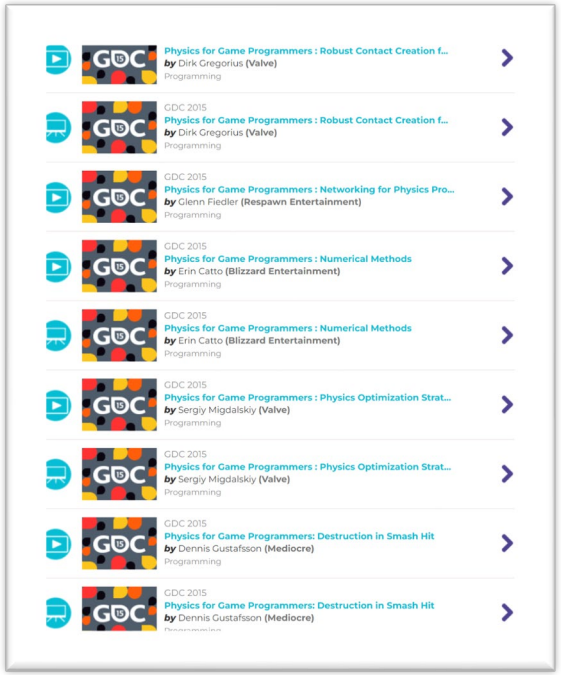
Applied Scientific Computing…aka Games

Game engines use numerical methods

- Look at all those GDC talks from 2015….

We'll take a quick look at some concepts from numerical methods

And re-visit Euler Integration



ILLINOIS

Dynamics

Games often simulate motion of objects in response to forces

We want to find the position of an object at a certain time

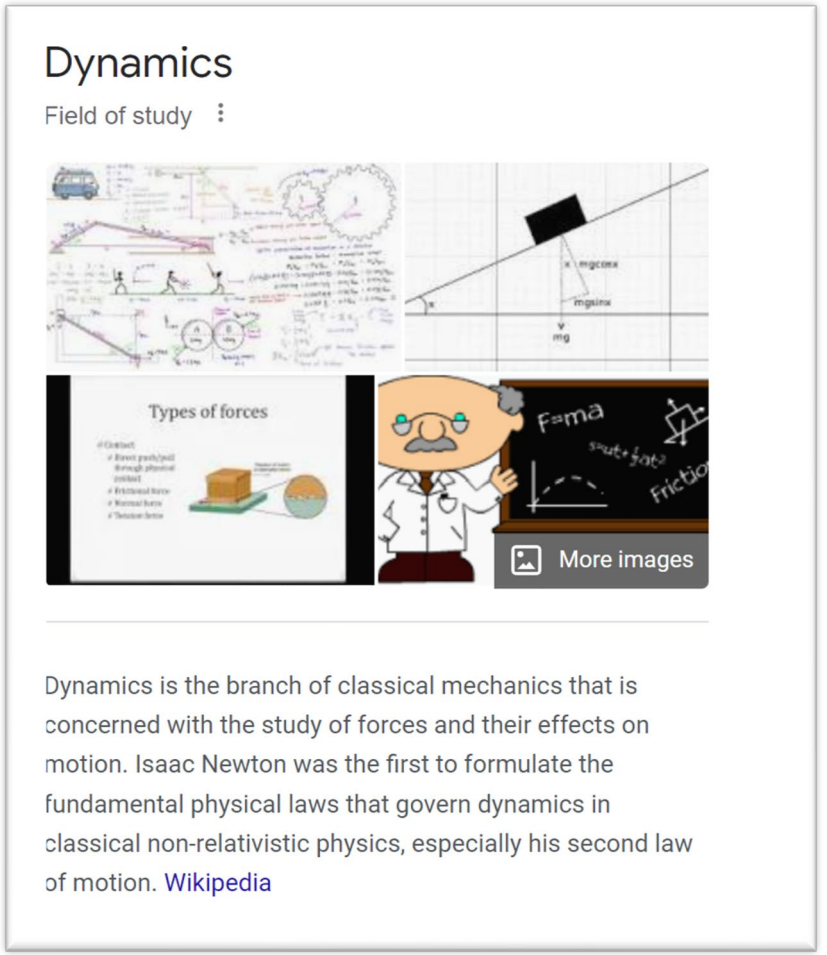
Forces generate acceleration that alters velocity that alters position

So…we're solving a differential equation $\ddot{\mathbf{p}} = \frac{1}{m}\mathbf{f}$

Computing the second integral tells us the new position after t units of time
Assuming acceleration is constant….

$$p' = p + \dot{p}t + \ddot{p}\frac{t^2}{2}$$

Why does that assumption typically break for games?



ILLINOIS

A Practical Approach

- Since forces and acceleration change…simpler to numerically integrate
- The position update using Euler's Method:

$$\mathbf{p}_{new} = \mathbf{p} + \dot{\mathbf{p}}t$$

- The velocity update using Euler's Method:

$$\dot{\mathbf{p}}_{new} = \dot{\mathbf{p}}d^t + \ddot{\mathbf{p}}t$$

Is Euler Good Enough?

It seems reasonable…but it is an approximation…

It discretely updates position…real physics does it continually

Even in the case of constant acceleration Euler typically gives different answers than the analytical solution

So..is it good enough for games? Or will #26 float up to the sky?

ILLINOIS

Let's Look at Euler in a Simpler Setting….

A vector field can be thought of as a field of velocities

Each vector describes velocity at a point…velocity is first derivative of position

Flow of a vector field tells us where a particle ends up

- after a certain time
- given a particular starting point

Requires the solution of an ordinary differential equation (ODE)

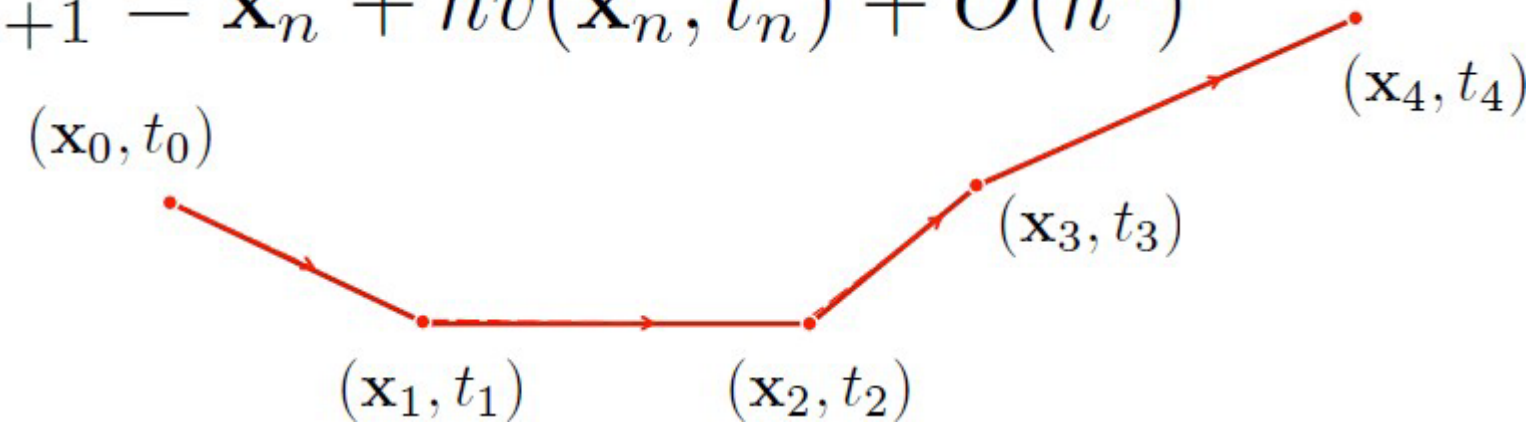
$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \vec{v}(\mathbf{x}(u))du$$

Analytical solution generally does not exist…need to use a numerical method

ILLINOIS

Euler's Method

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\vec{v}(\mathbf{x}_n, t_n) + O(h^2)$$



- Simple
- Fast
- Inaccurate
- Unstable

But beloved in computer graphics where things just need to look good…

ILLINOIS

Understanding Error

- Two sources of error

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\vec{v}(\mathbf{x}_n, t_n) + O(h^2)$$

1. Rounding error
    - due to the finite precision of floating-point arithmetic
2. Truncation error (or discretization error)
    - e.g. approximating an infinite process with a finite number of steps

- For ODEs truncation error is usually dominant
- The two types error are not independent
- Reducing the step-size will typically reduce truncation error but may increase rounding error

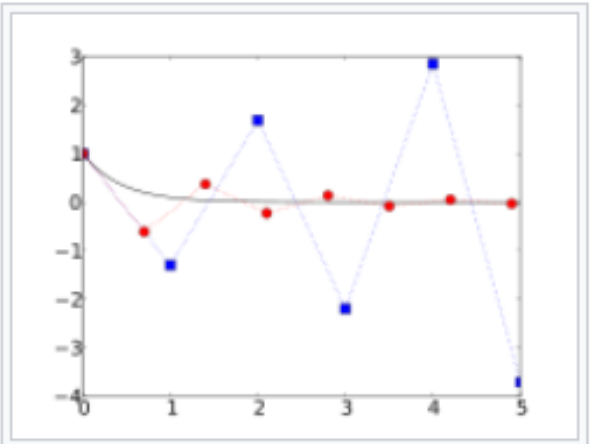
ILLINOIS

Stability

A numerically stable algorithm does not magnify approximation errors

With an unstable algorithm, a small amount of error in the input can produce a wildly incorrect result.

Euler is unstable



(Figure 4) Solution of $y' = -2.3y$ computed with the Euler method with step size $h = 1$ (blue squares) and $h = 0.7$ (red circles). The black curve shows the exact solution.

ILLINOIS

Truncation Error

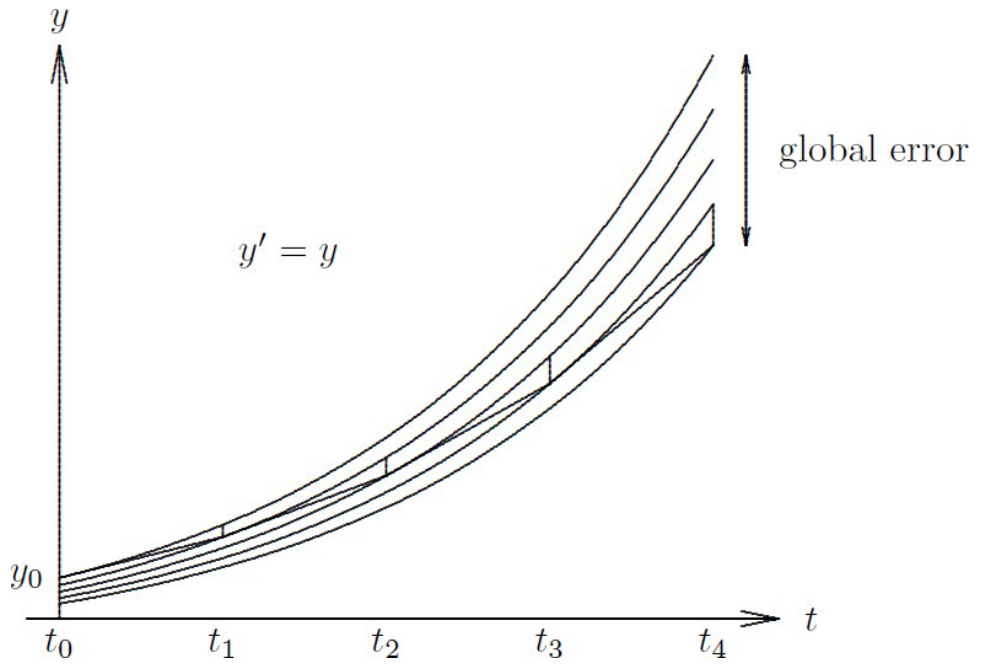
At kth step

- Global error is the cumulative overall error

$$e_k = y_k - y(t_k)$$

- Local error is the error in one step

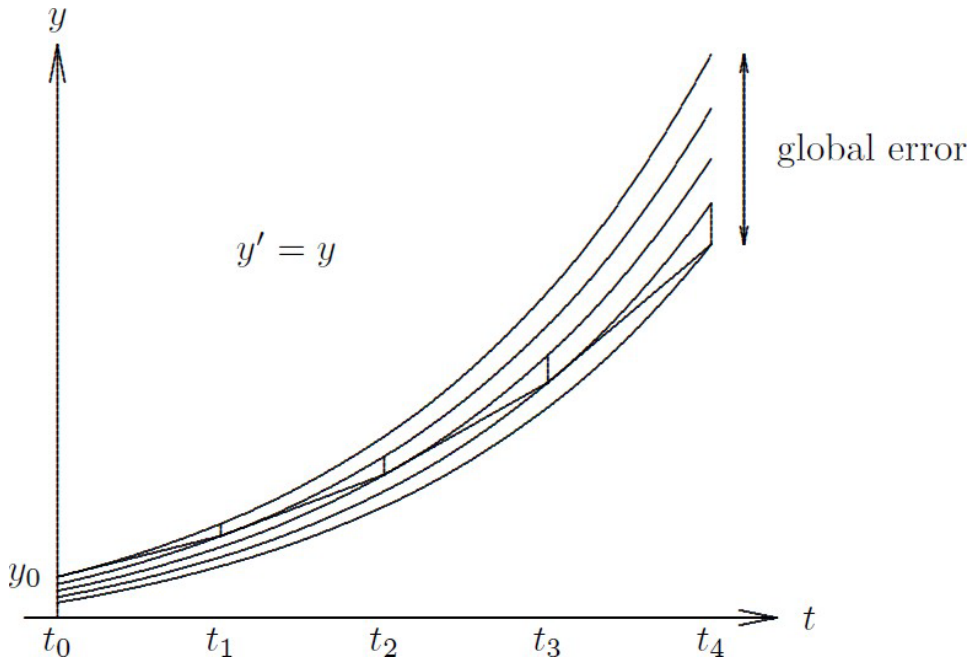
$$\ell_k = y_k - u_{k-1}(t_k)$$



ILLINOIS

Truncation Error for Euler's method

- Global error $O(h)$
- Local error is $O(h^2)$

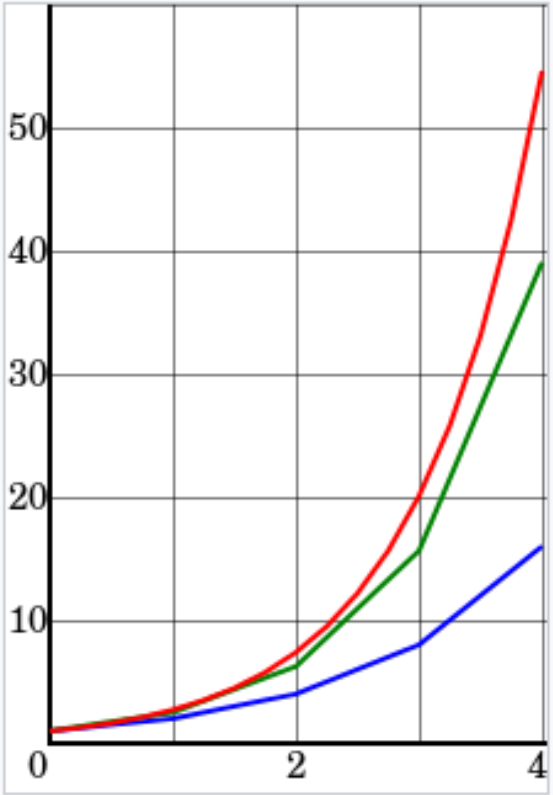


This is called *first order accurate*

*p*th order accurate when $\ell_k = \mathcal{O}(h_k^{p+1})$

ILLINOIS

1D Example: Euler's Method

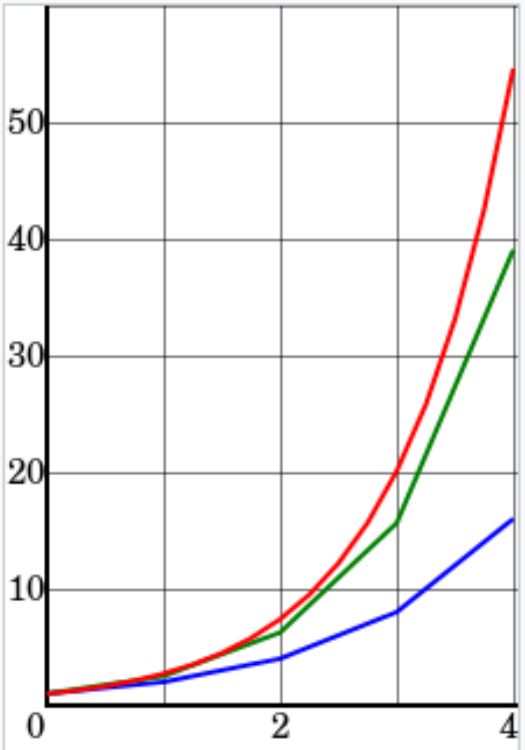


(Figure 2) Illustration of numerical integration for the equation $y' = y, y(0) = 1$. Blue is the Euler method; green, the midpoint method; red, the exact solution, $y = e^t$. The step size is $h = 1.0$.

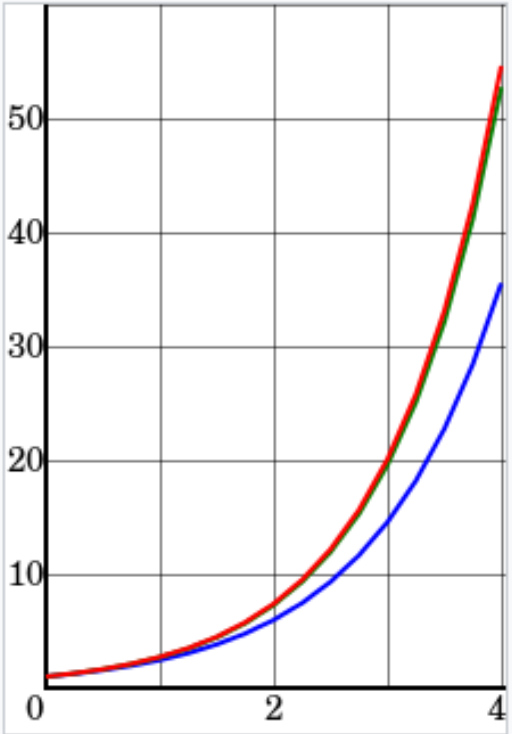
$$y_{n+1} = y_n + hf(t_n, y_n)$$

n	y_n	t_n	$f(t_n, y_n)$	h	Δy	y_{n+1}
0	1	0	1	1	1	2
1	2	1	2	1	2	4
2	4	2	4	1	4	8
3	8	3	8	1	8	16

Changing the Step Size



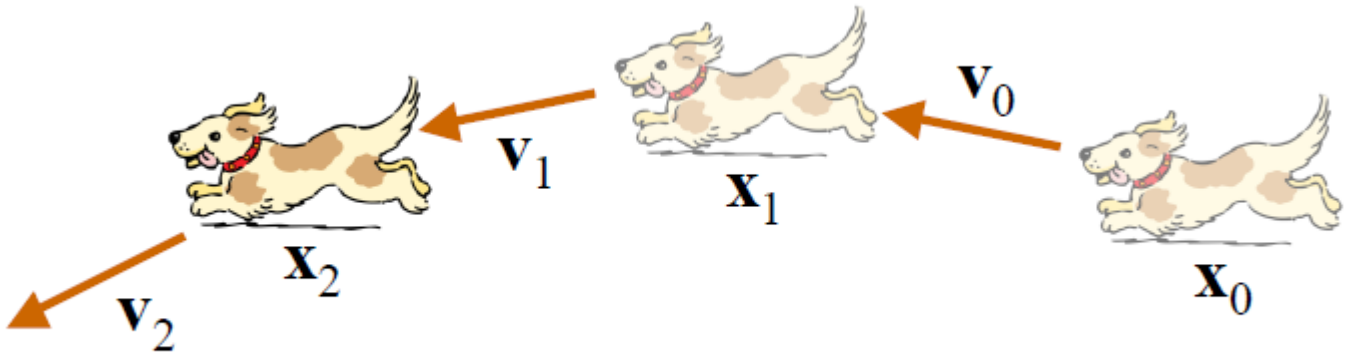
(Figure 2) Illustration of numerical integration for the equation $y' = y, y(0) = 1$. Blue is the Euler method; green, the midpoint method; red, the exact solution, $y = e^t$. The step size is $h = 1.0$.



(Figure 3) The same illustration for $h = 0.25$.

ILLINOIS

Problems with Euler's Method

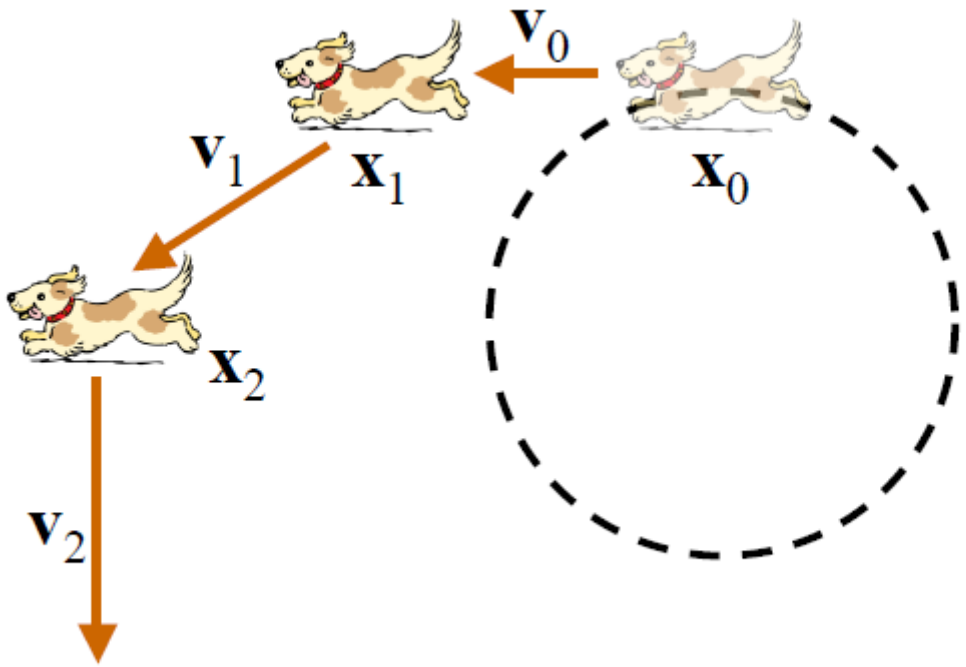


Relatively good solution

Actual velocity close to constant each time step

ILLINOIS

Problems with Euler's Method



Suppose dog is tied to a tree

Should spin in a circle

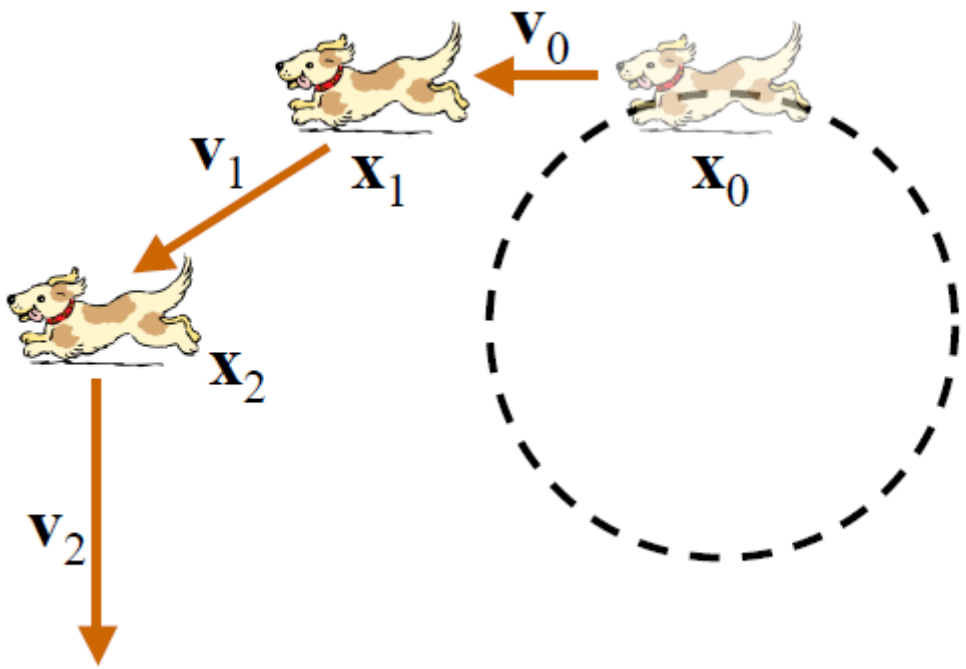
Velocity changes significantly during timestep

Spiral away from actual solution

Errors keep getting larger

ILLINOIS

Problems with Euler's Method



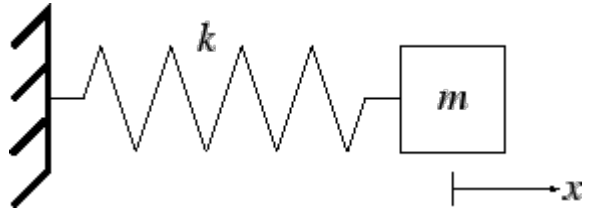
In the case of using Euler's method with oscillating systems like orbit, and springs, error accumulates in a way that adds energy.

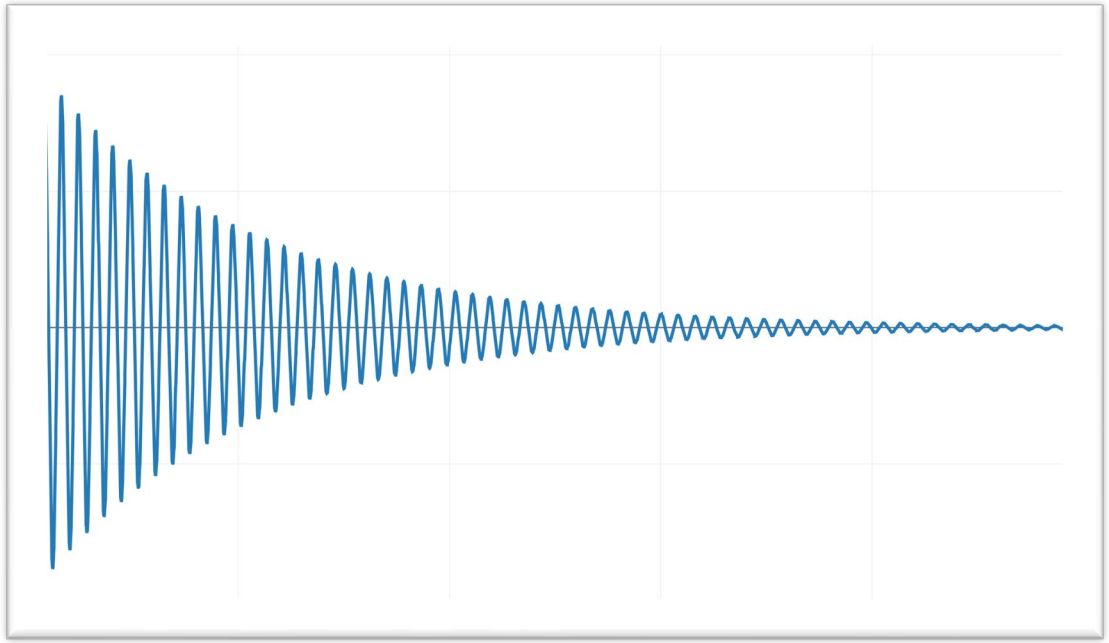
Decreasing how far we step will decrease the error, but ultimately we will still have problems with energy gain.

Unstable

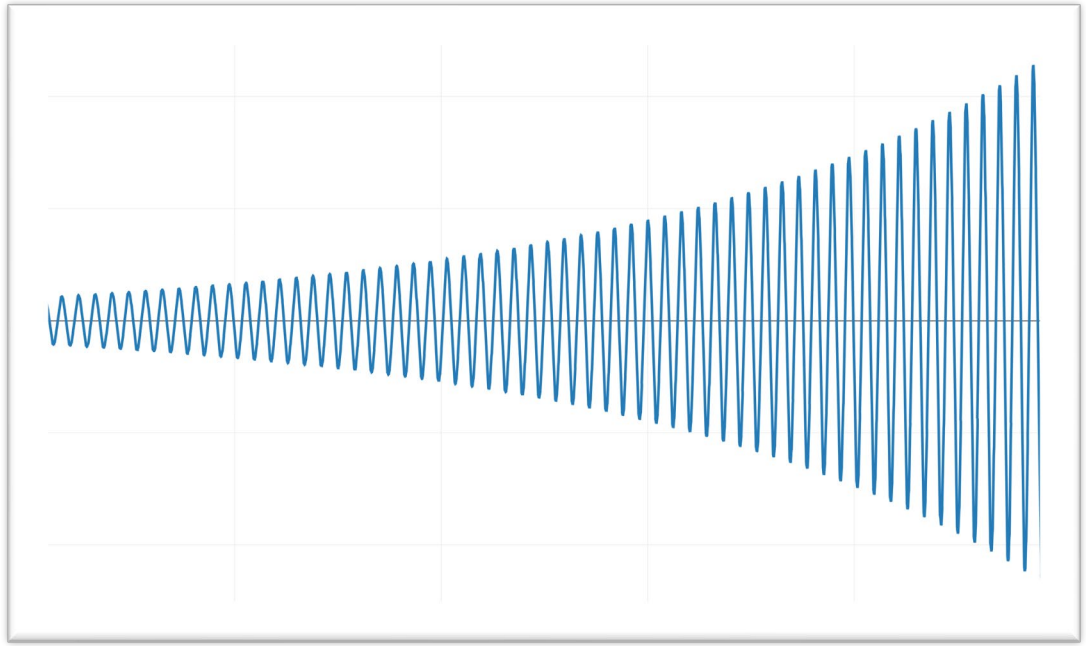
ILLINOIS

Damped Harmonic Oscillator: Euler's Method





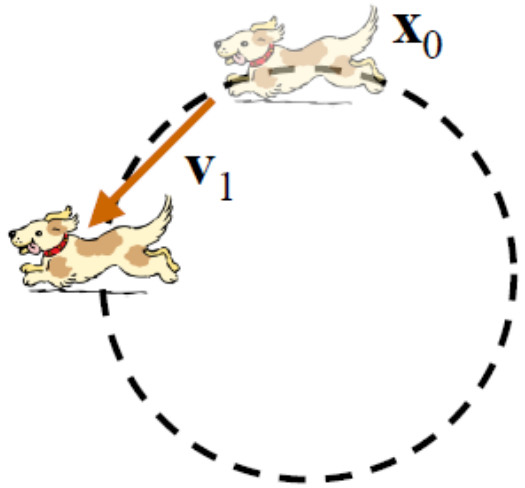
Actual Solution



Euler

ILLINOIS

Semi-Implicit Euler's Method



$$\mathbf{v}_{i+1} = \mathbf{v}_i + \mathbf{a}_i \Delta t$$

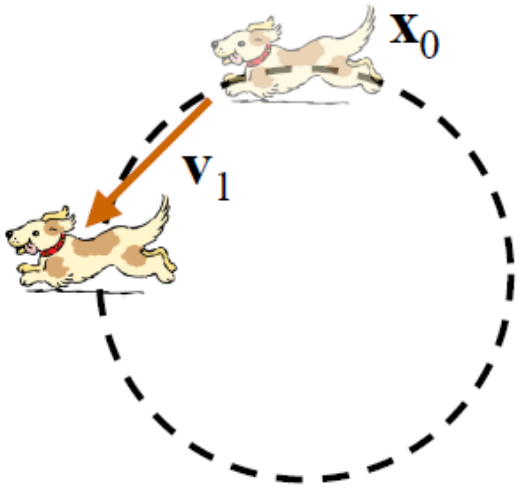
$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_{i+1} \Delta t$$

Update velocity first and use the new velocity for position update

Used in most game engines

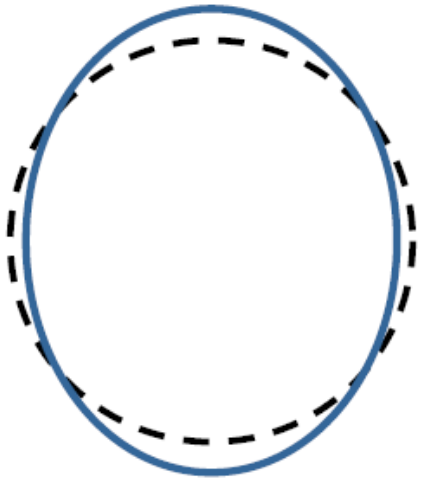
ILLINOIS

Semi-Implicit Euler's Method



$$\mathbf{v}_{i+1} = \mathbf{v}_i + \mathbf{a}_i \Delta t$$

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



Fast to compute

Stable

Can still diverge for large time steps and/or stiff ODEs

ILLINOIS

Euler's Method in History

Euler's Method scene in Hidden Figures (2016)
https://youtu.be/v-pbGAts_Fg

Used to calculate trajectories by NASA for Project Mercury…apparently

ILLINOIS