Continuous Systems

Simulation and Modeling (CSCI 3010U)

Faisal Z. Qureshi

http://vclab.science.ontariotechu.ca

Continuous systems simulation

- Time is treated as a continuous variable that drives the simulation
- Model is based upon differential equations, which describe how systems evolves over time, and how it responds to changes in input variables
- In this course, we will mostly deal with Ordinary Differential Equations (ODE), though Partial Differential Equations are used in some cases

Continuous system simulation: Variables

- Three set of variables:
  - State variables;
  - Input variables; and
  - Output variables.
- There is *no* overlap between input and state variables

Continuous system simulation: Variables

- Three set of variables:
  - State variables;
  - Input variables; and
  - Output variables.
- There is *no* overlap between input and state variables
- **Input variable** are not controlled by the simulation
  - The angle of the steering wheel in the car simulator example

Continuous system simulation: Variables

- Three set of variables:
  - State variables;
  - Input variables; and
  - Output variables.
- There is *no* overlap between input and state variables
- **Input variable** are not controlled by the simulation
  - The angle of the steering wheel in the car simulator example
- **Output variable** are things that we observe
  - The speed of the vehicle in the car simulator example

Continuous system simulation: Variables

- Three set of variables:
  - State variables;
  - Input variables; and
  - Output variables.
- There is *no* overlap between input and state variables
- **Input variable** are not controlled by the simulation
  - The angle of the steering wheel in the car simulator example
- **Output variable** are things that we observe
  - The speed of the vehicle in the car simulator example
- **State Variables** are controlled by the differential equations
  - The speed of the vehicle in the car simulator example
  - The location of the vehicle in the car simulator example

Ordinary Differential Equations (ODEs)

An ODE consists of the following ingredients:

Ordinary Differential Equations (ODEs)

An ODE consists of the following ingredients:

- An independent variable (usually “time” t that derivatives are taken with respect to

Ordinary Differential Equations (ODEs)

An ODE consists of the following ingredients:

- An independent variable (usually “time” t that derivatives are taken with respect to
- A dependent variable, i.e. function of the independent variable, e.g. $x = x(t)$ “the variable x which is a function of t”.

Ordinary Differential Equations (ODEs)

An ODE consists of the following ingredients:

- An independent variable (usually “time” t that derivatives are taken with respect to
- A dependent variable, i.e. function of the independent variable, e.g. $x = x(t)$ “the variable x which is a function of t”.
- A multi-variable function F that describes a relationship between the derivatives of the dependent variable (taken with respect to the independent variable)

Ordinary Differential Equations (ODEs)

An ODE consists of the following ingredients:

- An independent variable (usually "time" t that derivatives are taken with respect to
- A dependent variable, i.e. function of the independent variable, e.g. $x = x(t)$ "the variable x which is a function of t".
- A multi-variable function F that describes a relationship between the derivatives of the dependent variable (taken with respect to the independent variable)

Putting it all together, we get

$$F\left(\underset{\text{independent variable}}{\uparrow t}, \overset{\text{dependent}}{\downarrow x}, \frac{dx}{dt}, \frac{d^2x}{dt^2}, \cdots, \frac{d^{(n-1)}x}{dt^{(n-1)}}\right) = \frac{d^nx}{dt^n}$$

ODE: comments

- $\frac{dx}{dt}$ denotes derivative of x w.r.t. t

ODE: comments

- $\frac{dx}{dt}$ denotes derivative of x w.r.t. t
- $\frac{d^n x}{dt^n}$ denotes n-th derivative of x w.r.t. t

ODE: comments

- $\frac{dx}{dt}$ denotes derivative of x w.r.t. t
- $\frac{d^n x}{dt^n}$ denotes n-th derivative of x w.r.t. t
- $x' = \frac{dx}{dt}$

ODE: comments

- $\frac{dx}{dt}$ denotes derivative of x w.r.t. t
- $\frac{d^n x}{dt^n}$ denotes n-th derivative of x w.r.t. t
- $x' = \frac{dx}{dt}$
- $Dx = \frac{dx}{dt}$

Example: $\frac{d^3 x}{dt^3} = x''' = x^{(3)}$

ODE: comments

- $\frac{dx}{dt}$ denotes derivative of x w.r.t. t
- $\frac{d^n x}{dt^n}$ denotes n-th derivative of x w.r.t. t
- $x' = \frac{dx}{dt}$
- $Dx = \frac{dx}{dt}$
- Order of a differential equation is the highest order of derivative in that equation

ODE: comments

- $\frac{dx}{dt}$ denotes derivative of x w.r.t. t
- $\frac{d^n x}{dt^n}$ denotes n-th derivative of x w.r.t. t
- $x' = \frac{dx}{dt}$
- $Dx = \frac{dx}{dt}$
- Order of a differential equation is the highest order of derivative in that equation

Examples

- $mx'' = F$
- $x' + 32x'' + x''' = 0$
- $x' + 34x = 32$

ODE: comments

- $\frac{dx}{dt}$ denotes derivative of x w.r.t. t
- $\frac{d^n x}{dt^n}$ denotes n-th derivative of x w.r.t. t
- $x' = \frac{dx}{dt}$
- $Dx = \frac{dx}{dt}$
- Order of a differential equation is the highest order of derivative in that equation

Examples

- $mx'' = F$ (order is 2)
- $x' + 32x'' + x''' = 0$
- $x' + 34x = 32$

ODE: comments

- $\frac{dx}{dt}$ denotes derivative of x w.r.t. t
- $\frac{d^n x}{dt^n}$ denotes n-th derivative of x w.r.t. t
- $x' = \frac{dx}{dt}$
- $Dx = \frac{dx}{dt}$
- Order of a differential equation is the highest order of derivative in that equation

Examples

- $mx'' = F$ (order is 2)
- $x' + 32x'' + x''' = 0$ (order is 3)
- $x' + 34x = 32$

ODE: comments

- $\frac{dx}{dt}$ denotes derivative of x w.r.t. t
- $\frac{d^n x}{dt^n}$ denotes n-th derivative of x w.r.t. t
- $x' = \frac{dx}{dt}$
- $Dx = \frac{dx}{dt}$
- Order of a differential equation is the highest order of derivative in that equation

Examples

- $mx'' = F$ (order is 2)
- $x' + 32x'' + x''' = 0$ (order is 3)
- $x' + 34x = 32$ (order is 1)

Solving differential equations

- Solution is the dependent variable $x = x(t)$ that satisfies the equation
- Key idea: integration

$$\int x'' \, dt = \int 2 \, dt$$

Example: $x'' = 2$

Solving differential equations

- Solution is the dependent variable $x = x(t)$ that satisfies the equation
- Key idea: integration

Example: $x'' = 2$

1. Integrate once: $x' = 2t + C_1$

Do not forget the constant of integration.

Solving differential equations

- Solution is the dependent variable $x = x(t)$ that satisfies the equation
- Key idea: integration

Example: $x'' = 2$

1. Integrate once: $x' = 2t + C_1$
2. Integrate again: $x = \underline{t^2 + tC_1 + C_2}$ (Solution)

? ?

Solving differential equations

- Solution is the dependent variable $x = x(t)$ that satisfies the equation
- Key idea: integration

Example: $x'' = 2$

1. Integrate once: $x' = 2t + C_1$
2. Integrate again: $x = t^2 + tC_1 + C_2$ (Solution)

How to solve for C_1 and C_2, also called, constants of integration?

Solving differential equations

- Solution is the dependent variable $x = x(t)$ that satisfies the equation
- Key idea: integration

Example: $x'' = 2$

1. Integrate once: $x' = 2t + C_1$
2. Integrate again: $x = t^2 + tC_1 + C_2$ (Solution)

How to solve for C_1 and C_2, also called, constants of integration?

- Use initial or boundary conditions.

This is the information that we are missing.

Solving differential equations

- Solution is the dependent variable $x = x(t)$ that satisfies the equation
- Key idea: integration

Example: $x'' = 2$

1. Integrate once: $\underline{\underline{x'}} = 2t + C_1$
2. Integrate again: $\underline{\underline{x}} = t^2 + tC_1 + C_2$ (Solution)

How to solve for C_1 and C_2, also called, constants of integration?

- Use initial or boundary conditions.
- Using initial conditions $\underline{x'(0)} = 3$ and $\underline{x(0)} = 2$, we get C_1=3 and $C_2 = 2$. The solution is $x(t) = \underline{t^2 + 3t + 2}$.

complete solution.

Aside: Constant of Integration

Notice that

$$\frac{df(x)}{dx} = 3x^2$$

for both when $f(x) = x^3$ or $f(x)x^3 + C$.

This suggests constant C disappears in the process of differentiation.

Therefore, when we integrate we add the constant C for the sake of completeness.

$$\int 3x^2 dx = 3\left(\frac{x^3}{3}\right) + C = x^3 + C$$

Furthermore, we will need other information to find the true value of C. Note also that there is nothing preventing $C = 0$.

Nth order ODEs

- General solution to an nth order ODE will contain n constants of integration
- We need n more equations
  - Use initial or boundary conditions to get these equations to solve for the constants of integration

Nth order ODEs

- General solution to an nth order ODE will contain n constants of integration
- We need n more equations
  - Use initial or boundary conditions to get these equations to solve for the constants of integration
- Initial conditions
  - The values of $x(t)$ and its first $n-1$ derivatives for a particular value of t
  - If such values are only available at the end, run time backwards to convert problem to initial conditions

Nth order ODEs

- General solution to an nth order ODE will contain n constants of integration
- We need n more equations
  - Use initial or boundary conditions to get these equations to solve for the constants of integration
- Initial conditions
  - The values of $x(t)$ and its first $n-1$ derivatives for a particular value of t
  - If such values are only available at the end, run time backwards to convert problem to initial conditions
- Boundary conditions
  - The values of $x(t)$ and its derivatives for two different values of t

↑

$x(1) = 3$

Eg. $x'' = 2$ $\qquad x(7) = 4$

Nth order ODE reducibility

1 n^{th}-order $\rightarrow$ n 1^{st}-order

Any explicit differential equation of order n

$$\frac{d^n x}{dt^n} \longrightarrow x^{(n)} = F\left(t, x, x', \cdots, x^{(n-1)}\right)$$

can be written as a system of n first-order differential equations by defining a new family of unknown functions

$$x^{(i-1)} = x_i$$

Notice the abuse of notation. Here $x^{(k)}$ denote the k-th derivative of x w.r.t. t.

Nth order ODE reducibility

We can then represent the following n-th order ODE

$$x^{(n)} = F\left(t, x, x', \cdots, x^{(n-1)}\right)$$

with n first-order ODEs as follows

$$\begin{aligned}
x_1' &= x_2 \\
x_2' &= x_3 \\
&\vdots \\
x_{(n-1)}' &= x_n \\
x_n' &= F\left(t, x_1, x_2, \cdots, x_n\right)
\end{aligned}$$

Nth order ODE reducibility

Example

Acceleration due to gravity.

The following 2nd order ODE

$$\frac{d^2x}{dt^2} = -g$$

9.8 m/s²

can be reduced to

$$\frac{dx}{dt} = v$$

introduced a new variable.

$$\frac{dv}{dt} = -g$$

Nth order ODE reducibility

Example

The following 2nd order ODE

$$\underline{\frac{d^2x}{dt^2} = -g}$$

can be reduced to

$$\frac{dx}{dt} = v$$
$$\frac{dv}{dt} = -g$$

We introduced a new variable v. ✓

Introduce a variable:

$$\boxed{h = \frac{dx}{dt}}$$ 1st-order ODE

$$\boxed{\frac{dh}{dt} = -g}$$ 1st-order ODE

solve for h

solve for x, given h

First order ODEs

- All of our simulations only involve first order ODEs

First order ODEs

- All of our simulations only involve first order ODEs
- What about models that involve higher order ODEs?
  - E.g., the equation of motion for particle is modelled by a second order differential equation ✓
  - Applies reducibility, i.e., replace a higher order differential equation by a system of first order differential equations
  - We can replace an nth order ODE with n first order ODE

First order ODEs

- All of our simulations only involve first order ODEs
- What about models that involve higher order ODEs?
  - E.g., the equation of motion for particle is modelled by a second order differential equation
  - Applies reducibility, i.e., replace a higher order differential equation by a system of first order differential equations
  - We can replace an nth order ODE with n first order ODE
- Advantages of first order ODEs

First order ODEs

- All of our simulations only involve first order ODEs
- What about models that involve higher order ODEs?
  - E.g., the equation of motion for particle is modelled by a second order differential equation
  - Applies reducibility, i.e., replace a higher order differential equation by a system of first order differential equations
  - We can replace an nth order ODE with n first order ODE
- Advantages of first order ODEs
  - First order equations are much easier to solve numerically ✓

First order ODEs

- All of our simulations only involve first order ODEs
- What about models that involve higher order ODEs?
  - E.g., the equation of motion for particle is modelled by a second order differential equation
  - Applies reducibility, i.e., replace a higher order differential equation by a system of first order differential equations
  - We can replace an nth order ODE with n first order ODE
- Advantages of first order ODEs
  - First order equations are much easier to solve numerically
  - Very few numerical solvers available for higher order equations

Equation of motion

Newton's second law of motion:

> *"The acceleration of an object as produced by a net force is directly proportional to the magnitude of the net force, in the same direction as the net force, and inversely proportional to the mass of the object."*

Mathematically:

$\underline{a \propto F}$ and $\underline{a \propto \frac{1}{m}}$, and combining the two we get $\boxed{F = ma.}$ Recall $a = \frac{d^2x}{dt^2}$, so $F = ma$ is a second order equation.

$$F = ma$$

$$\implies F = m\frac{d^2x}{dt^2}$$

2nd-order ODE

Equation of motion

We introduce a new variable velocity $v = \frac{dx}{dt}$, and get the following first order system of equations

$$F = m\,d^2x/dt^2$$

$$= m\frac{d}{dt}\underset{\overset{\big\downarrow}{v}}{\left(\frac{dx}{dt}\right)}$$

$$\boxed{v = \frac{dx}{dt}}$$

$$\boxed{F = m\frac{dv}{dt}}$$

introduce a variable.
(velocity)

↑

2 first-order ODE.

Solving ODEs numerically

General idea: given a solution $x(t)$ at time $t = t_0$, incrementally step forward in time to find $x(t + \Delta t)$

Example: lets consider the equation of motion

$$F = m\frac{d^2x}{dt^2} \longrightarrow \left|\begin{array}{ll} \textcircled{1}\ F = m\dfrac{dv}{dt} \implies & \textcircled{3}\ \Delta v = \left(\dfrac{F}{m}\right)(\Delta t) \\ \textcircled{2}\ v = \dfrac{dx}{dt} \implies & \textcircled{4}\ \Delta x = (v)(\Delta t) \end{array}\right.$$

We can use Δv and Δx to update the **current** values of v and x, respectively.

Reducibility property

$x(0)$
$v(0)$
↑
initial

$x(0)$
$x(1)$
↑
boundary

Solving ODEs numerically: an example

What is the value of x at time $t = 3$?

Model

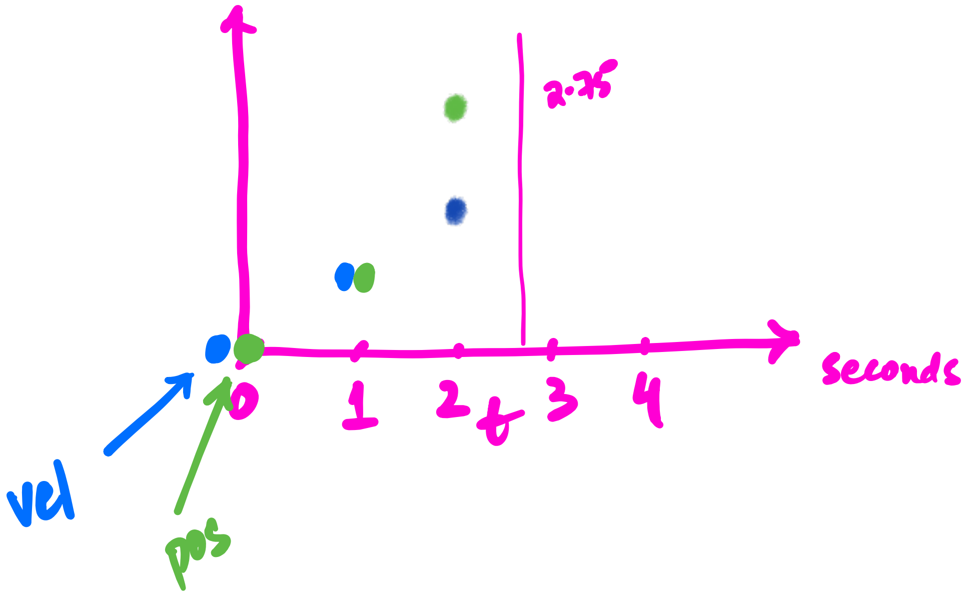
$$v(t + \Delta t) = v(t) + \Delta v = v(t) + (F/m)(\Delta t)$$
$$x(t + \Delta t) = x(t) + \Delta x = x(t) + (v)(\Delta t)$$

Setup

- $m = 1,\ F = 1$ (other quantities used in the simulation)
- $\underline{v(0) =}\ 0,\ x(0) = 0$ (initial state), $\underline{\Delta t = 1}$ (time step)

Solution

- $v(1) = 1$
  - $x(1) = 1$
- $v(2) = 2$
  - $x(2) = 3$
- $v(3) = 3$
  - $x(3) = 6$



Solving ODEs numerically: an example

What is the value of x at time $t = 3$?

Model

$$v(t + \Delta t) = v(t) + \Delta v = v(t) + (F/m)(\Delta t)$$
$$x(t + \Delta t) = x(t) + \Delta x = x(t) + (v)(\Delta t)$$

Setup

- $m = 1$, $F = 1$ (other quantities used in the simulation)
- $v(0) = 0$, $x(0) = 0$ (initial state), $\Delta t = 1$ (time step)

Solution

- $v(1) = 1$
  - $x(1) =$
- $v(2) =$
  - $x(2) =$
- $v(3) =$
  - $x(3) =$

Solving ODEs numerically: an example

What is the value of x at time $t = 3$?

Model

$$v(t + \Delta t) = v(t) + \Delta v = v(t) + (F/m)(\Delta t)$$
$$x(t + \Delta t) = x(t) + \Delta x = x(t) + (v)(\Delta t)$$

Setup

- $m = 1$, $F = 1$ (other quantities used in the simulation)
- $v(0) = 0$, $x(0) = 0$ (initial state), $\Delta t = 1$ (time step)

Solution

- $v(1) = 1$
  - $x(1) = 1$
- $v(2) =$
  - $x(2) =$
- $v(3) =$
  - $x(3) =$

Solving ODEs numerically: an example

What is the value of x at time $t = 3$?

Model

$$v(t + \Delta t) = v(t) + \Delta v = v(t) + (F/m)(\Delta t)$$
$$x(t + \Delta t) = x(t) + \Delta x = x(t) + (v)(\Delta t)$$

Setup

- $m = 1$, $F = 1$ (other quantities used in the simulation)
- $v(0) = 0$, $x(0) = 0$ (initial state), $\Delta t = 1$ (time step)

Solution

- $v(1) = 1$
  - $x(1) = 1$
- $v(2) = 2$
  - $x(2) =$
- $v(3) =$
  - $x(3) =$

Solving ODEs numerically: an example

What is the value of x at time $t = 3$?

Model

$$v(t + \Delta t) = v(t) + \Delta v = v(t) + (F/m)(\Delta t)$$

$$x(t + \Delta t) = x(t) + \Delta x = x(t) + (v)(\Delta t)$$

Setup

- $m = 1$, $F = 1$ (other quantities used in the simulation)
- $v(0) = 0$, $x(0) = 0$ (initial state), $\Delta t = 1$ (time step)

Solution

- $v(1) = 1$
  - $x(1) = 1$
- $v(2) = 2$
  - $x(2) = 3$
- $v(3) =$
  - $x(3) =$

Solving ODEs numerically: an example

What is the value of x at time $t = 3$?

Model

$$v(t + \Delta t) = v(t) + \Delta v = v(t) + (F/m)(\Delta t)$$
$$x(t + \Delta t) = x(t) + \Delta x = x(t) + (v)(\Delta t)$$

Setup

- $m = 1$, $F = 1$ (other quantities used in the simulation)
- $v(0) = 0$, $x(0) = 0$ (initial state), $\Delta t = 1$ (time step)

Solution

- $v(1) = 1$
  - $x(1) = 1$
- $v(2) = 2$
  - $x(2) = 3$
- $v(3) = 3$
  - $x(3) =$

Solving ODEs numerically: an example

What is the value of x at time $t = 3$?

Model

$$v(t + \Delta t) = v(t) + \Delta v = v(t) + (F/m)(\Delta t)$$
$$x(t + \Delta t) = x(t) + \Delta x = x(t) + (v)(\Delta t)$$

Setup

- $m = 1$, $F = 1$ (other quantities used in the simulation)
- $v(0) = 0$, $x(0) = 0$ (initial state), $\Delta t = 1$ (time step)

Solution

- $v(1) = 1$
  - $x(1) = 1$
- $v(2) = 2$
  - $x(2) = 3$
- $v(3) = 3$
  - $x(3) = 6$

State vars: pos & time

Input vars: M, f

Output: pos

Solving ODEs numerically: practical considerations

- Have to choose Δt carefully
  - Δt too small; the simulation can become very slow
  - Δt too large; the simulation can become very inaccurate
  - Advanced techniques can change Δt when solving equations to maintain acceptable accuracy and speed

Solving ODEs numerically: practical considerations

- Have to choose Δt carefully
  - Δt too small; the simulation can become very slow
  - Δt too large; the simulation can become very inaccurate
  - Advanced techniques can change Δt when solving equations to maintain acceptable accuracy and speed

- Δt determines the exact points in time for which we have the solution
  - What if we want solution at other points in time?
  - This places constraints on how we solve the equations

Solving ODEs numerically: practical considerations

- Have to choose Δt carefully
  - Δt too small; the simulation can become very slow
  - Δt too large; the simulation can become very inaccurate
  - Advanced techniques can change Δt when solving equations to maintain acceptable accuracy and speed

- Δt determines the exact points in time for which we have the solution
  - What if we want solution at other points in time?
  - This places constraints on how we solve the equations

- Often times Δt used for solving ODEs is much smaller than the one used to update the display

Choice of ΔT

- Molecular activity
- Evolution of an ecosystem
- Galaxy formation

Choice of ΔT

- Molecular activity (fraction of a millisecond)
- Evolution of an ecosystem
- Galaxy formation

Choice of ΔT

- Molecular activity (fraction of a millisecond)
- Evolution of an ecosystem (months or years)
- Galaxy formation

Choice of ΔT

- Molecular activity (fraction of a millisecond)
- Evolution of an ecosystem (months or years)
- Galaxy formation (millions or billions of years)

Simulation loop

- Advance the simulation

Simulation loop

- Advance the simulation
- Display current results

Simulation loop

- Advance the simulation
- Display current results
- Get the user response

Simulation loop

- Advance the simulation
- Display current results
- Get the user response
- Repeat

Displaying results

- Real-time or not?

Displaying results

- Real-time or not?
- Timescale

Displaying results

- Real-time or not?
- Timescale
- Response and interactivity

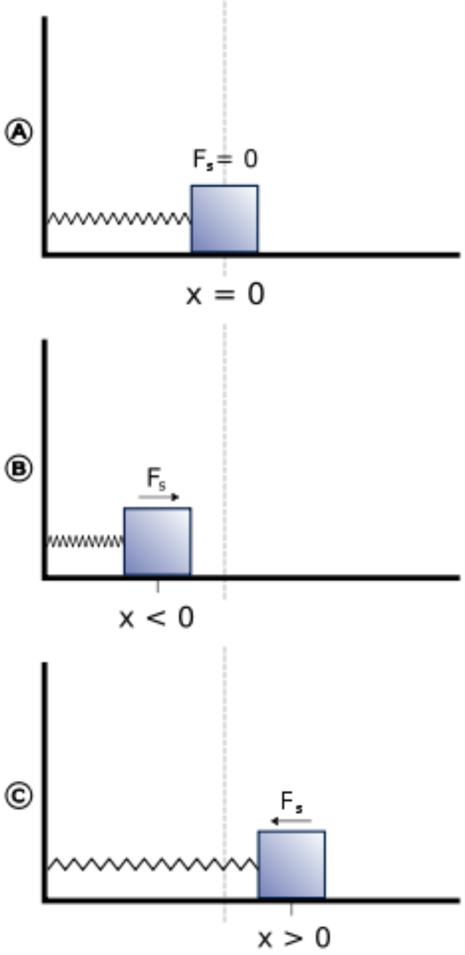
Displaying results

- Real-time or not?
- Timescale
- Response and interactivity
- Refresh rates

Mass spring system

Hook's law (1676) states, "the extension is proportional to the force."

Mathematically, $\underbrace{F = -k\overset{\text{deformation}}{x}}_{\text{linear model}}$, where k is the spring constant and x is the displacement of the spring from rest position under the application of force F.



Mass spring system

Step 1: construct a model that will describe the motion of the mass over time

Hook's law: $\underline{F = -kx}$ Force

Newton's Second Law of Motion: $\underline{F = ma}$ — How things move under force.

Combining the two we get

① $v = \dfrac{dx}{dt}$

② $m\dfrac{dv}{dt} = -kx$

$$\underline{ma = -kx}$$

$$\implies \boxed{m\frac{dx^2}{dt^2} = -kx}$$ 2nd - order ODE

② $\Delta v = -\dfrac{k}{m} x \Delta t$

① $\Delta x = v \Delta t$

Mass spring system

Step2: find a way to solve the model numerically

Mass spring system

Step2: find a way to solve the model numerically

Convert the second order $m\frac{dx^2}{dt^2} = -kx$ to a system of first order equations

$$\frac{dx}{dt} = v$$

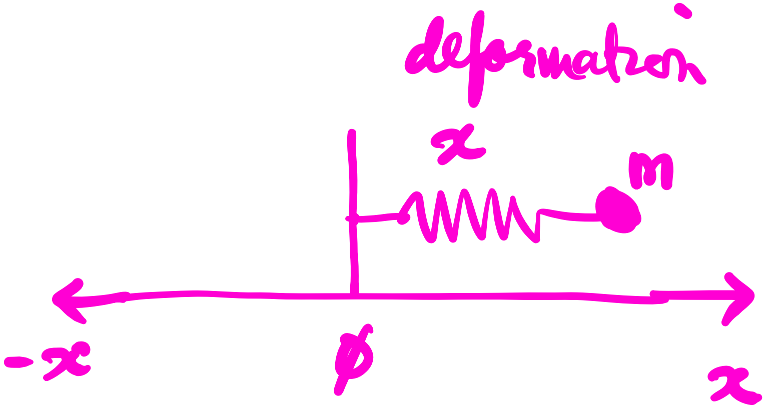
$$m\frac{dv}{dt} = -kx$$

Mass spring system

Step2: find a way to solve the model numerically

Convert the second order $m\frac{dx^2}{dt^2} = -kx$ to a system of first order equations

$$\frac{dx}{dt} = v$$
$$m\frac{dv}{dt} = -kx$$



And make the update rules

$$x(t+\Delta t) = x(t) + v(t)\Delta t$$
$$v(t+\Delta t) = v(t) - \frac{k}{m}x(t)\Delta t$$

Mass spring system

- Set values for mass m and spring constant k
- Set the initial conditions
  - Values for x and v at start time t_0
- Run the simulation loop
  1. Update t to $t + \Delta t$
  2. Update values for x and v using the update ruls
  3. Display results or save them to file for plotting
  4. Repeat steps 1 to 4

We just simulated a *Simple Harmonic Oscillator*

Code: `1d-mass-spring`

Mass spring system

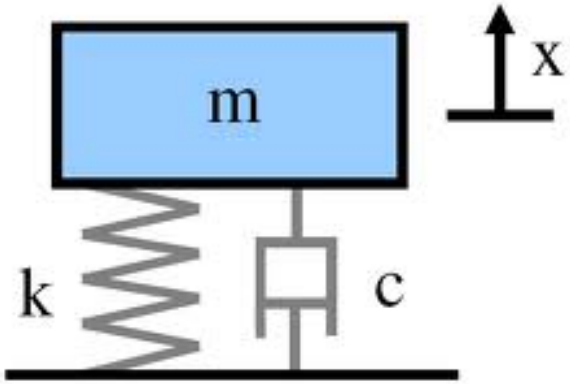
```
# Mass-Spring system
class Mass:
    def __init__(self):
        self.x = 5
        self.vx = 0
        self.k = 1
        self.dt = 0.1
        self.t = 0
        self.m = 1.0

    def update(self):
        self.x += (self.vx * self.dt)
        self.vx += (- self.k * self.x * self.dt / self.m)
        self.t += self.dt
```

damping force .

Mass Spring Damper

The mass experiences a damping force that is proportional to its current velocity



Mathematically

$$F = -kx - cv$$

where c is the damping constant

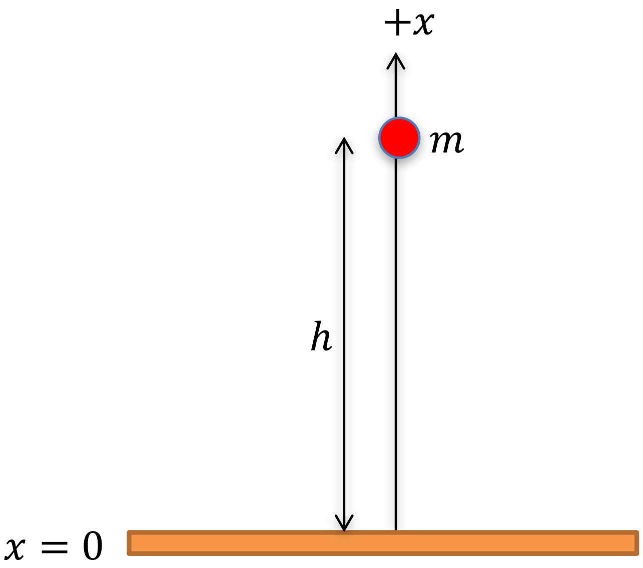
Code: modify `1d-mass-spring` to add damping effect

Bouncing ball

Assumption 1: We simplify the problem by treating the ball as a particle
From Newton's Second Law of Motion

$$F = m\frac{dx^2}{dt^2}$$

where x is the height of the ball from the ground and m is the mass of the ball.



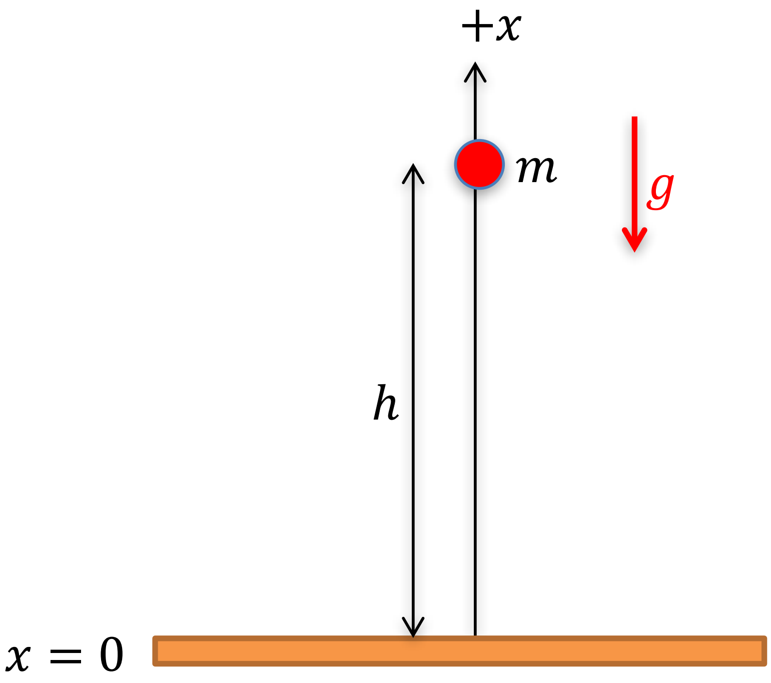
Bouncing ball

Assumption 2 Gravity is the only force acting upon this ball then

$$F = -mg$$

Putting it together we get

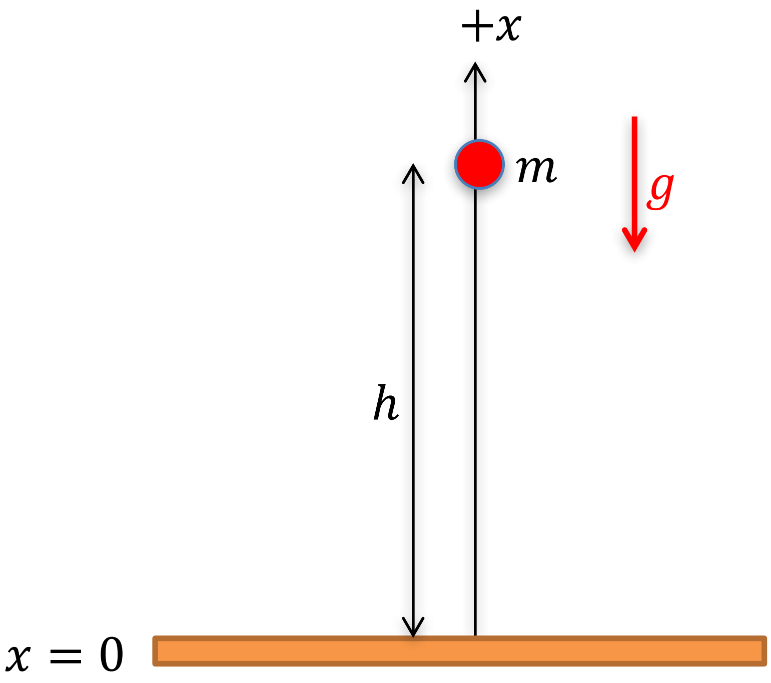
$$\frac{\overset{2}{d}x^{\cancel{2}1}}{dt^2} = -g$$



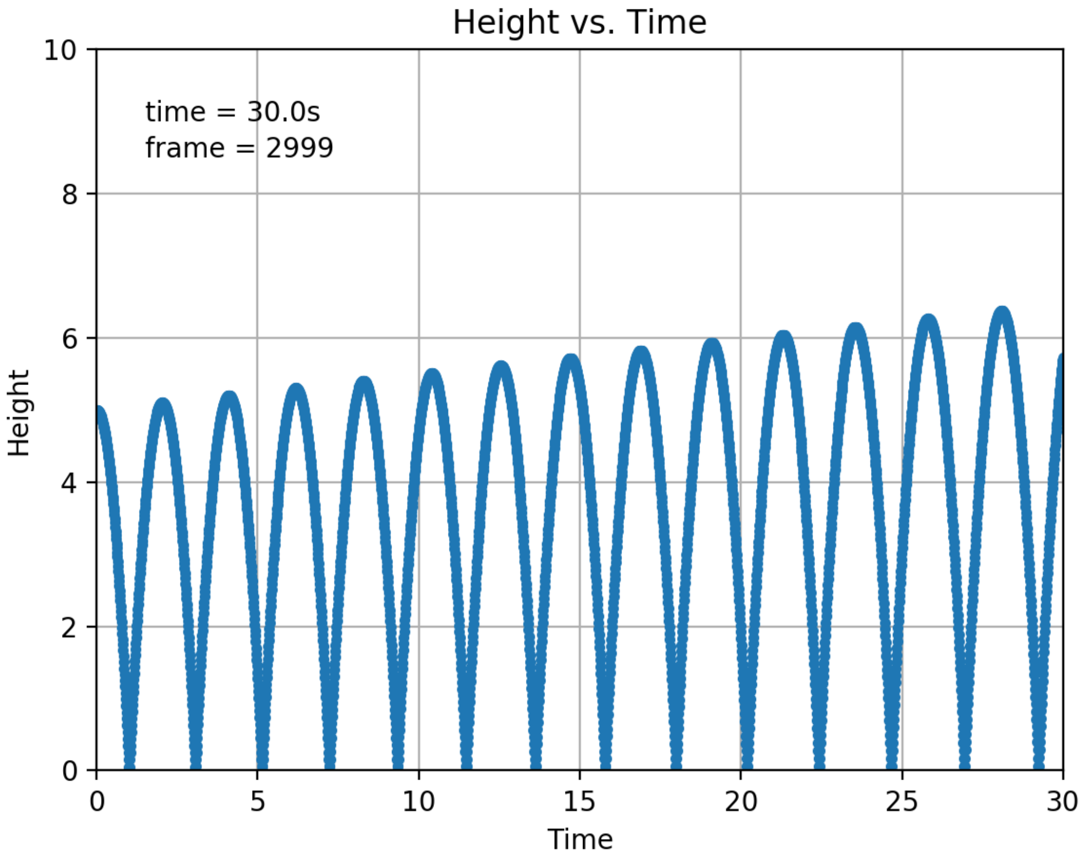
Bouncing ball

Data collection

- We need to know the value of g. For our purposes, we use $g = 9.8m/s^2$
- By using different g we can simulate bouncing ball on different planets



Bouncing ball



Problem.

Did you notice something peculiar with this plot?

Code: `ball-floor` turn off RK4

Bouncing ball

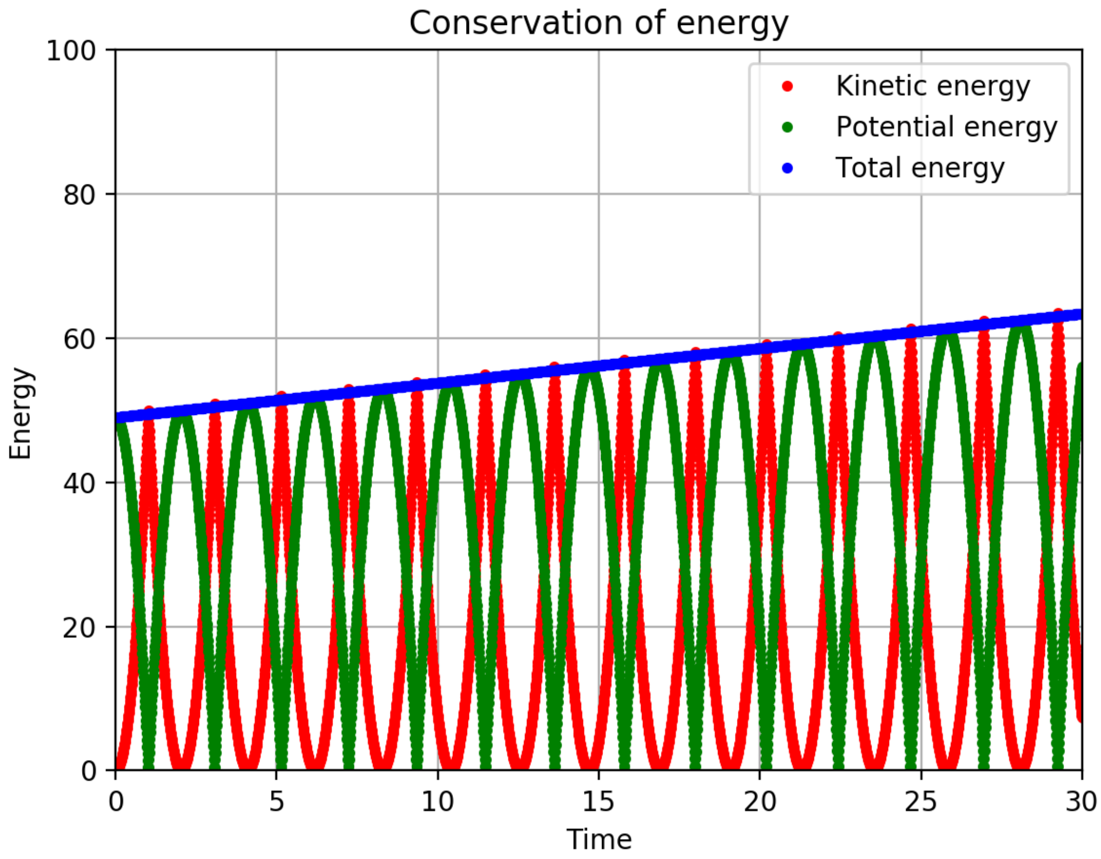
- The ball goes higher with each bounce, which is unexpected.
- The error doesn't go away even if we make timestep really small. It does, however, minimizes the effect.
- It seems we are imparting energy to the ball with each bounce. This breaks the *the law of conservation of energy*, which states that "the total energy of an isolated system remains constant."

Bouncing ball total energy

Total energy of the ball is the sum of its *kinetic* and *potential* energies.

Kinetic energy $= \frac{1}{2}mv^2$
Potential energy $= mgy$

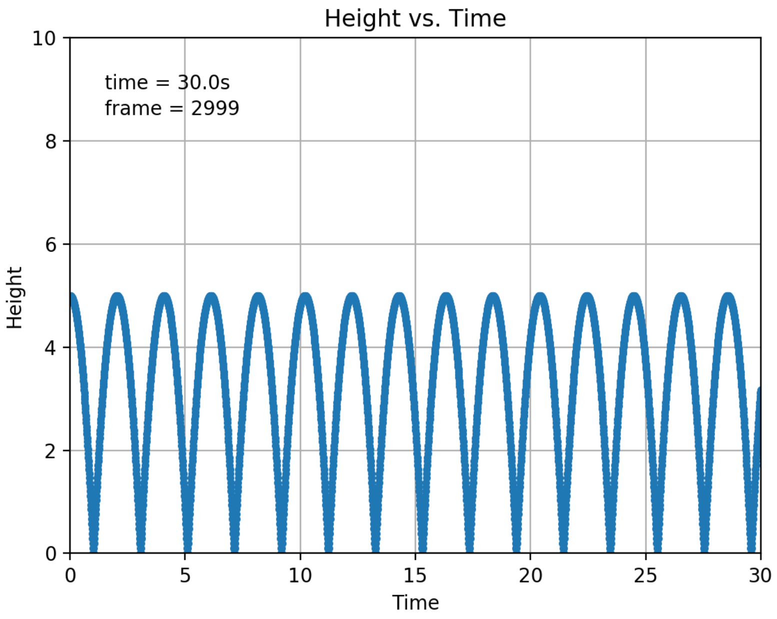
height

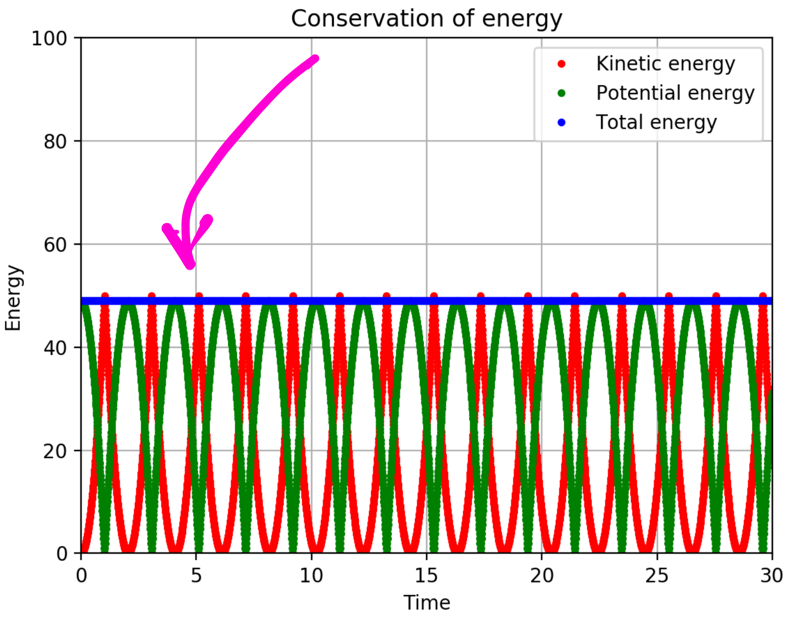


This behavior is due to incorrect assumptions of Euler method.

Bouncing Ball

Total energy is conserved when using Runga-Kutta or RK4 solver.


Height vs. Time


Conservation of energy

Code: `ball-floor` turn on RK4

Euler method

- A numerical solver for first order ODEs
- First order numerical procedure for solving ODEs (initial value problems)
- It is an *explicit method*
  - Calculates the state of the system at a later time given its current state by using the *update equations*
  - $y(t + \Delta t) = F(y(t))$

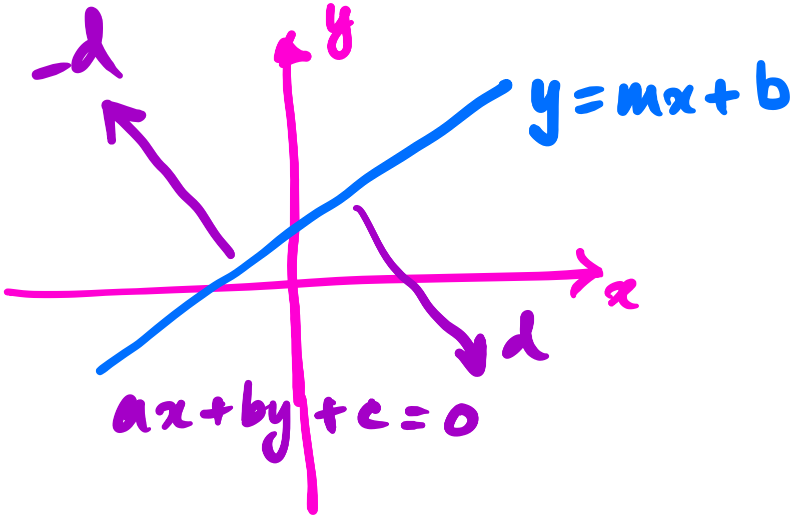
Euler method

- A numerical solver for first order ODEs
- First order numerical procedure for solving ODEs (initial value problems)
- It is an *explicit method*
  - Calculates the state of the system at a later time given its current state by using the *update equations*
  - $y(t + \Delta t) = F(y(t))$

Aside: implicit methods

- Calculates the state of the system at a later time given by solving an equation that includes both the future state and the current state
- $\underline{G(y(t + \Delta t), y(t))} = 0$

cool, but difficult.

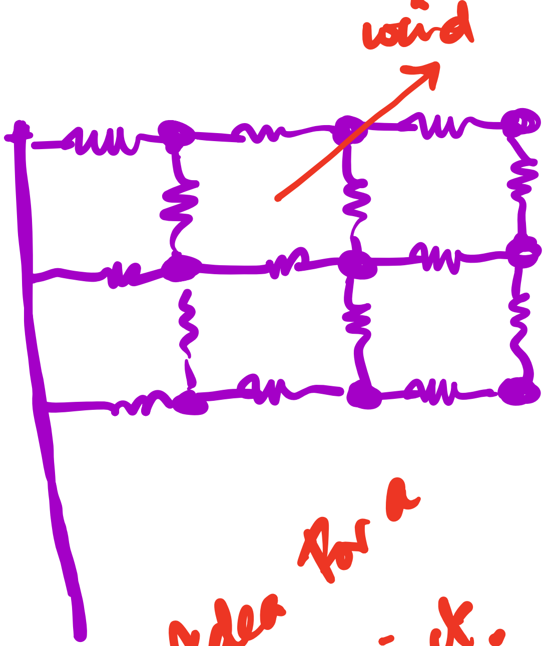


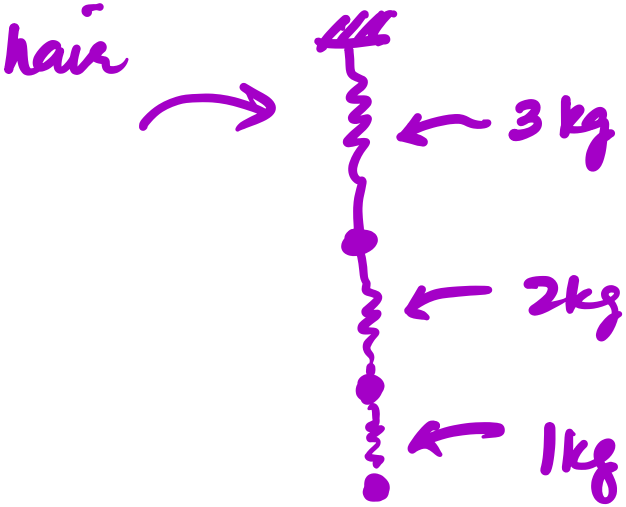
Euler method

- Numerically unstable
  - Adversally effects accuracy ↓
  - Exhibits error growth over time
    - Error is proportional to Δt

Euler method

- Numerically unstable
  - Adversally effects accuracy
  - Exhibits error growth over time
    - Error is proportional to Δt
- Particularly unsuited for stiff equations
  - Equations containing terms that lead to rapid changes
  - E.g., a mass spring system with large spring constant





Idea for a Project.

Euler method

- Numerically unstable
  - Adversally effects accuracy
  - Exhibits error growth over time
    - Error is proportional to Δt
- Particularly unsuited for stiff equations
  - Equations containing terms that lead to rapid changes
  - E.g., a mass spring system with large spring constant
- Use *extremely* small time steps
  - Infeasible in practice

Runga-Kutta method

- An other numerical solver for first order ODEs
- An alternate to Euler method
- A family of explicit and implicit methods
- Often RK4 is used
  - Error is proportional to Δt^4
  - Makes a huge difference for small values of Δt

Takeaway: whenever possible use RK4 method

Numerical solvers in Python

```
from scipy.integrate import ode
def f(self, t, y, arg1):
    """Solves y' = f(t, y)
      Arguments:
      - y is the state of the system.  In our case
        y[0] is the position and y[1] is the velocity.
      - arg1 is 9.8, as set by set_f_params() method.

      Returns vector dy/dt.  In our case, dx/dt = v and
      dv/dt = -g.
      """
    return [y[1], -arg1]

r = ode(f).set_integrator('dop853')
r.set_initial_value([y0, vy0], t0)
r.set_f_params(9.8)
r.integrate(dt)
print r.t, r.y
```

→ 1st order ODE

Bouncing ball: takeaways

- Exploit your knowledge of physics to determine if simulation is behaving as expected
- Use several strategies
- Compare outputs of several strategies
  - If outputs differ, you must have a way to explain the differences
  - If outputs are the same, the simulation *may* be correct

Discussion

Q. Why does Euler method performing so poorly for our bouncing ball example?

Discussion

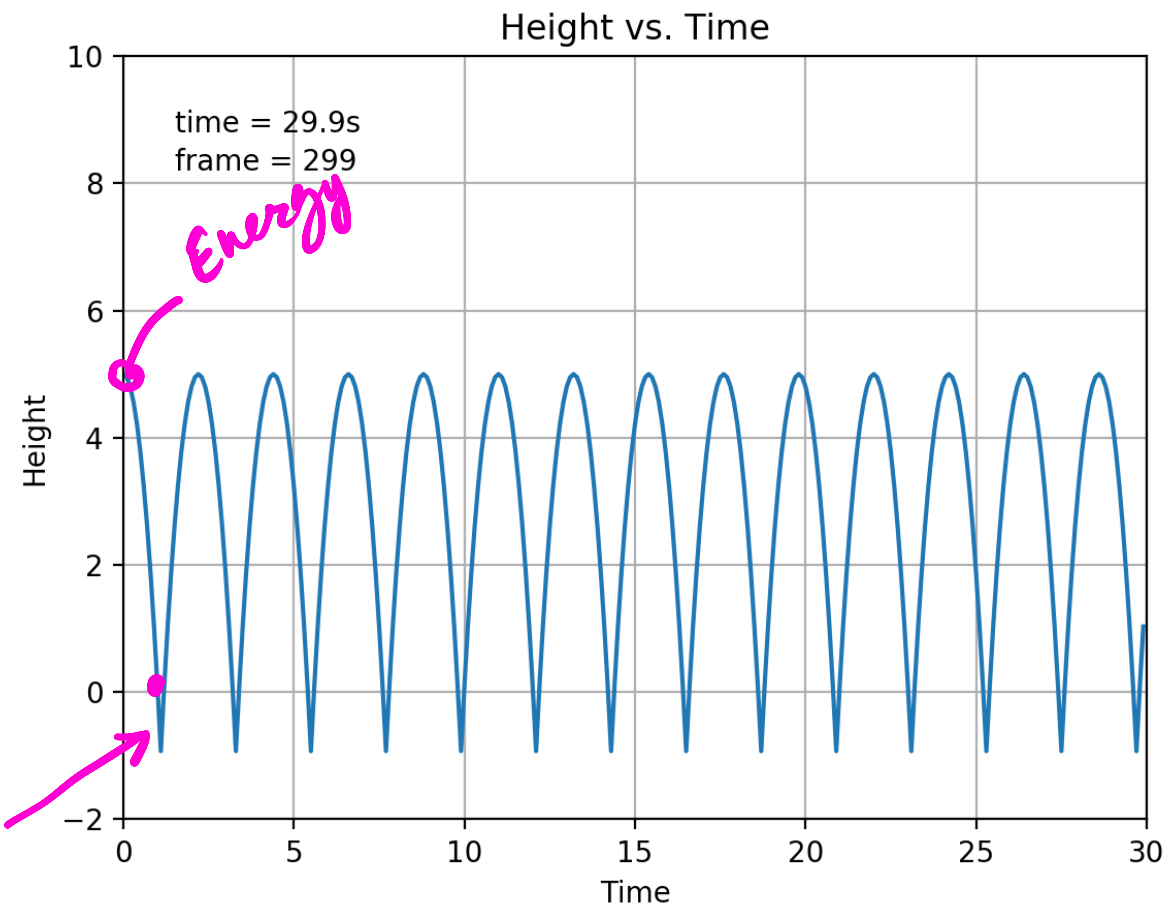
Q. Why does Euler method performing so poorly for our bouncing ball example?

A. Euler method assumes that the acceleration remains constant between two time steps. Notice that this assumption is generally false, but especially so when the ball "hits" the ground at $x = 0$. The velocity is flipped, changing the sign of the derivative and causing a discontinuity.

RK4 method is much better at handling discontinuities (as long as there aren't too many of these).

This is why RK4 is able to get good results even for large time steps.

Bouncing ball



For this simulation, the floor sits at height 0. The ball pierces through the floor, which is incorrect.

Code: `ball-floor` increase timestep to see the ball penetrating the floor

Bouncing ball

Need a better way to detect collisions with the floor

Bouncing ball

Need a better way to detect collisions with the floor

Scheme 1

- Use smaller time steps
- The ball will travel less distance between two time steps, and there is a greater chance of catching the collision instant
- In any case, the ball will penetrate less into the floor

Bouncing ball

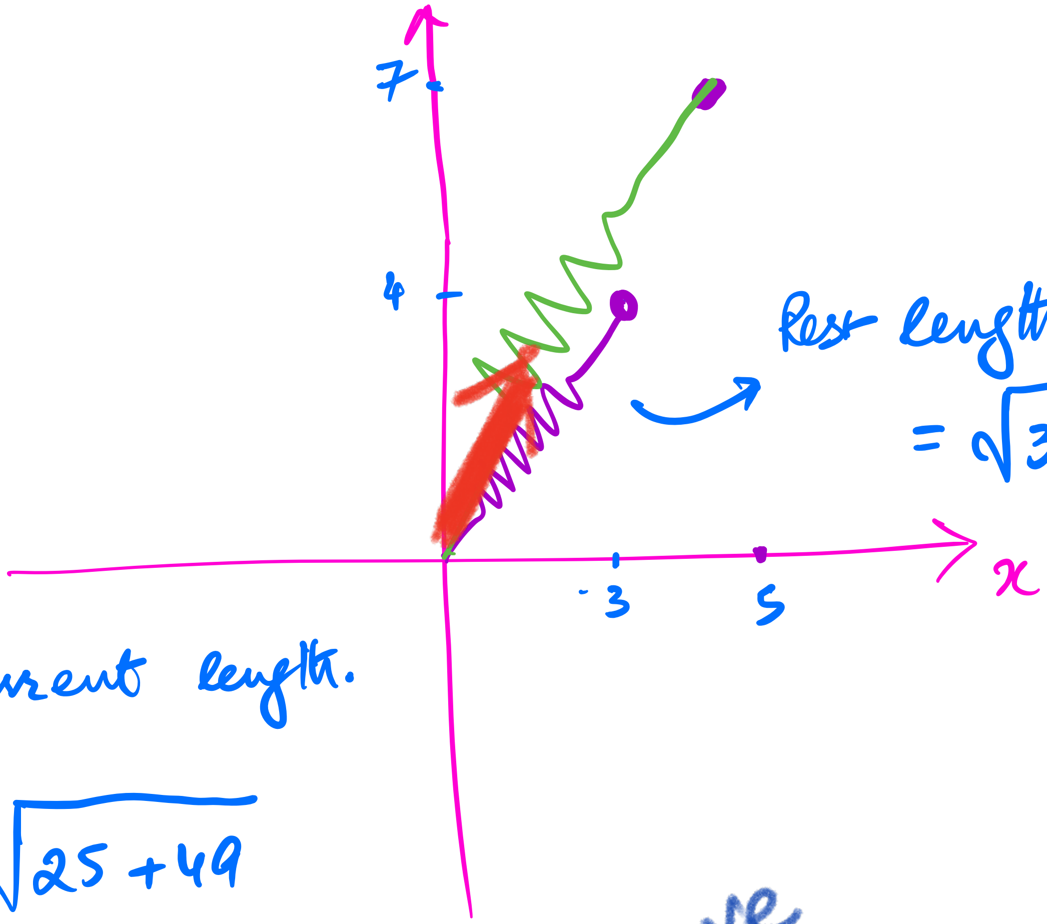
Need a better way to detect collisions with the floor

Scheme 1

- Use smaller time steps
- The ball will travel less distance between two time steps, and there is a greater chance of catching the collision instant
- In any case, the ball will penetrate less into the floor

Scheme 2

- Try to find the exact time of collision using $x = vt$ relationship
- Adjust time step accordingly



Rest length = 5

$= \sqrt{3^2+4^2}$

Current length.

$\sqrt{25+49}$

+ve

deformation : $k\left(\sqrt{74} - 5\right) = f$

$$\text{axis} = \left(\frac{5}{\sqrt{74}}, \frac{7}{\sqrt{74}}\right) = \vec{a}$$

$$\text{Force} = -\boxed{f\vec{a}}$$

-ve +ve