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: let's consider the equation of motion

$$v = \frac{dx}{dt} \implies \Delta x = v \times \Delta t$$

$$F = m \frac{dv}{dt} \implies \Delta v = \frac{m (F \times \Delta t)}{m}$$
Solving ODEs numerically

We will use the following equations to update the value of \( x \) over time.

\[
\Delta x = v \times \Delta t \\
\Delta v = \frac{\text{mass}(F \times \Delta t)}{m}
\]

Case study

Given \( m = 1, F = 1, v(0) = 0, x(0) = 0 \), and \( \Delta t = 1 \). What is the value of \( x \) at time \( t = 3 \)?

- \( x(1) = \)
  - \( v(1) = \)
- \( x(2) = \)
  - \( v(2) = \)
- \( x(3) = \)
  - \( v(3) = \)
Solving ODEs numerically: practical considerations

- Have to choose $\Delta t$ carefully
  - $\Delta t$ too small; the simulation can become very slow
  - $\Delta t$ too large; the simulation can become very inaccurate
  - Advanced techniques can change $\Delta t$ when solving equations to maintain acceptable accuracy and speed
- $\Delta t$ 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 $\Delta t$ used for solving ODEs is much smaller than the one used to update the display
Choice of $\Delta T$

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

Solve the ODE
Displaying results

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

Hooke’s law (1676) states, “the extension is proportional to the force”

Mathematically, \( F = -kx \), where \( k \) is the spring constant and \( x \) is the displacement of the spring from rest position under the application of force \( F \).
Robert Hooke (1635–1703)

- Royal Society’s Curator of Experiments (*Nullus in Verba*)
- Scientific contributions
  - 1660, Discovered Hooke’s Law
  - 1665, Published *Micrographia*
  - Discovered plant cells

*By the means of Telescopes, there is nothing so far distant but may be represented to our view; and by the help of Microscopes, there is nothing so small as to escape our inquiry; hence there is a new visible World discovered to the understanding.*

(From Micrographia)
Mass spring system

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

Hook’s law: \( F = -kx \)

Newton’s Second Law of Motion: \( F = ma \)

Combining the two we get

\[ ma = -kx \]

\[ \Rightarrow m \frac{dx^2}{dt^2} = -kx \]
Mass spring system

Step2: find a way to solve the model numerically

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

\[
\frac{dx}{dt} = v \\
\frac{dv}{dt} = -kx
\]

And make the update rules

\[
x(t + \Delta t) = x(t) + v(t) \Delta t \\
v(t + \Delta t) = -\frac{k}{m} x(t) \Delta t + v(t)
\]
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 rules
  3. Display results or save them to file for plotting
  4. Repeat steps 1 to 4

We just simulated a *Simple Harmonic Oscillator*
Mass spring system

```python
# 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
```

Position of the mass or spring deformation

Velocity of the mass
Mass Spring Damper

Exercise: simulate a mass spring damper system

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

Mathematically

\[ F = -kx - cv \]

where \( c \) is the damping constant
Sir Isaac Newton (1642–1726)

- Classical mechanics (*Mathematical Principles of Natural Philosophy*)
  - Laws of motion
  - Universal gravitation
- Optics
- Calculus (in parallel with Gottfried Wilhelm Leibniz)
- Theoretical calculation of the speed of sound

*I do not know what I may appear to the world, but to myself I seem to have been only like a boy playing on the sea-shore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me.*

(Memoirs)
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

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

\[ F = ma \]

\[ \Rightarrow -mg = xa \]

\[ \Rightarrow \frac{d^2x}{dt^2} = -g \]
Bouncing ball

Data collection

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

Did you notice something peculiar with this plot?

\[ \text{k.e.} = \frac{1}{2} mv^2 \]

\[ \text{p.e.} = mgh \]
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 *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 $= mgH$

This behavior is due to incorrect assumptions of Euler method.
Bouncing Ball

Total energy is conserved when using Runga-Kutta or RK4 solver.
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 - \[ G(y(t + \Delta t), y(t)) = 0 \]
Euler method

- Numerically unstable
  - Adversally affects accuracy
  - Exhibits error growth over time
    - Error is proportional to $\Delta t$
- 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 $\Delta t^4$
  - Makes a huge difference for small values of $\Delta t$

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
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?

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.
Bouncing ball

Exercise: we need a better way to handle 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

**Scheme 2**

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

Collision detection
1. Approximate time to collide \( t_c = \frac{x}{v} \)
2. Set \( x = 0 \) and \( v = -v(t + t_c) \)
   - Flip \( v \) to indicate that the ball is now going back up again

Problem
\( v \) is larger than had we calculated \( t_c \) exactly right (that’s because the particle is under constant acceleration). Consequently energy is not conserved.
Bouncing ball

Use the Law of Conservation of Energy to compute the velocity of the ball when it touches ground.

The ball was released at height \( h \). We know the total energy of the system, which is \( mgh \). At the start the kinetic energy is 0.

When the ball touches the ground, its potential energy reduces to 0. Since the total energy remains the same, all of its energy is now kinetic energy.

\[
\frac{1}{2}mv^2 = mgh
\]

\[
v = \sqrt{2gh}
\]

i.e., set \( x = 0 \) and \( v = -\sqrt{2gh} \) at collision time.
Bouncing ball

Ball doesn’t enter the floor

Energy is conserved