Lecture 15

Differential Equations in Python

Workshop 5 Review

Differential equations (DiffEQs)...

- Any equation with a derivative can be described as a differential equation
- You have already seen some and you will see more in the future

$$\frac{dx}{dt} = v(t)$$

ODEs

$$\frac{d^2\mathbf{r}}{dt^2} = \frac{\mathbf{F}(\mathbf{r})}{m}$$

$$\frac{\partial^2 f(x,t)}{\partial t^2} = c^2 \frac{\partial^2 f(x,t)}{\partial x^2}$$

PDE

ODE = EASYish

What we will
cover today

PDE = HARD

Save these for an
upper division
calculus class

Solving ordinary differential equations (ODEs) Analytically

- When these equations are simple enough, we can actually just solve them
- By using some math trickery and the calculus we learned last week, we can turn this into an easy to solve problem
- Unfortunately, not every differential equation can be solved like this, so we need some approximation methods

$dy/dx = x$

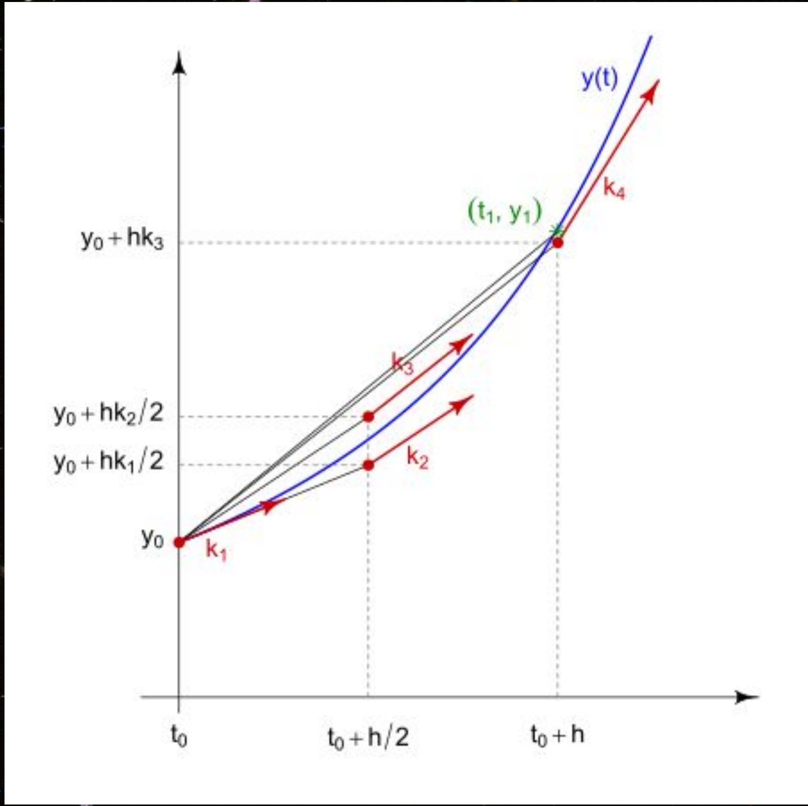
$dy = x*dx$

$\int dy = \int x\, dx$

$y = \frac{1}{2} x^2$

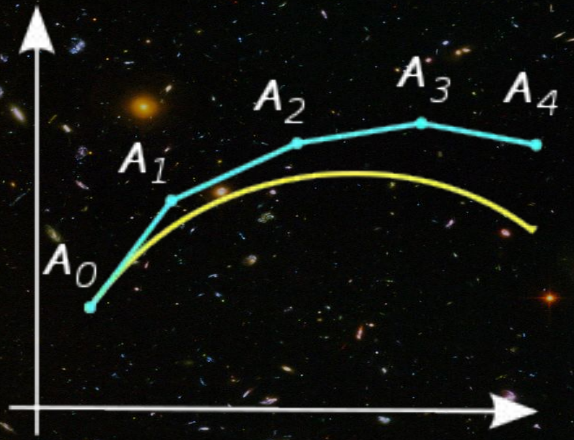
Solving ODEs: Runge Kutta Methods

- A family of different iterative methods (including the Euler Method) used for solving nonlinear equations
- We have some initial conditions (y_o) that we then iterate bit by bit to solve the equation
- Similar idea to the integration approximation methods we studied last week



Solving ODEs by Euler

- **Euler's method** (very similar to numerical integration)
  - Break down to segments
  - Specifying initial condition
    - You need to start from somewhere
    - Order of diff eq. = # of initial conditions
  - Need to specify your step size
- Assign! Not Append!



Solving ODEs...

$$\frac{dy}{dx} = xy$$

$$(x_0, y_0) = (2, 3)$$

```
dx = 0.01
x0, y0 = 2, 3

def derivative(y,x):
    return x*y

x_arr = np.linspace(x0,5,301)
y_arr = np.zeros(301)
```

```
y_arr[0] = y0

for i in range(1, 301):
    dy = derivative(y_arr[i-1],
    x_arr[i-1]) * dx
    y = y_arr[i-1]+dy
    y_arr[i] = y
```

Scipy does it better...

https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html

```
from scipy.integrate import odeint
```

```
odeint(func, y0, t)
```

```
def deriv(y,t):
    return …
```

Your function, make sure y is the first variable

Initial condition on y

Time point at which you would like to evaluate the function for

Equivalent to x_arr

Final Coding Demo