Numerical Analysis in Python

Overview

- Briefly cover linear algebra basics/functions that you should know how to use
- Cover numerical methods for solving ODEs

Linear Algebra

- Will generally use the numpy linalg package for most linear algebra computations (or scipy's linalg package, they have many of the same functions)
- Commonly used tools:
  - Matrix/matrix, matrix/vector, vector,vector products
  - Solving linear equations
  - Eigenvectors/values
  - Decompositions (SVD, Cholesky, LU, etc.)
- Let's look at some examples!

Numerically Solving an ODE

- First discretize the ODE, note that you already have experience with discretized functions evertime you use a vector in numpy!

Numerically Solving an ODE

- First discretize the ODE, note that you already have experience with discretized functions evertime you use a vector in numpy!
- Taylor expand a function about a point x_{i+1}, then can solve for the derivative:

$$
\begin{aligned}
f\left(x_{i+1}\right) &= f\left(x_i\right) + \frac{df}{dx}\Delta x + \cdots \\
\frac{df}{dx} &= \frac{f\left(x_{i+1}\right) - f\left(x_i\right)}{\Delta x}
\end{aligned}
$$

Numerically Solving an ODE

- First discretize the ODE, note that you already have experience with discretized functions evertime you use a vector in numpy!
- Taylor expand a function about a point x_{i+1}, then can solve for the derivative:

$$
\begin{aligned}
f(x_{i+1}) &= f(x_i) + \frac{df}{dx}\Delta x + \cdots \\
\frac{df}{dx} &= \frac{f(x_{i+1}) - f(x_i)}{\Delta x}
\end{aligned}
$$

- Consider an ode: $\frac{dy}{dt} = f(t, y)$, a possible formula (Forward Euler scheme) for solving this ode for each timestep is:

$$y_{i+1} = y_i + \Delta t \cdot f(t_i, y_i)$$

Numerically Solving an ODE

- First discretize the ODE, note that you already have experience with discretized functions evertime you use a vector in numpy!
- Taylor expand a function about a point x_{i+1}, then can solve for the derivative:

$$\begin{aligned} f\left(x_{i+1}\right) &= f\left(x_i\right) + \frac{df}{dx}\Delta x + \cdots \\ \frac{df}{dx} &= \frac{f\left(x_{i+1}\right) - f\left(x_i\right)}{\Delta x} \end{aligned}$$

- Consider an ode: $\frac{dy}{dt} = f\left(t, y\right)$, a possible formula (Forward Euler scheme) for solving this ode for each timestep is:

$$y_{i+1} = y_i + \Delta t \cdot f\left(t_i, y_i\right)$$

- Can just iterate through this process until you reach the final time, need an initial condition to start

Numerically Solving an ODE

- First discretize the ODE, note that you already have experience with discretized functions evertime you use a vector in numpy!
- Taylor expand a function about a point x_{i+1}, then can solve for the derivative:

$$
\begin{aligned}
f\left(x_{i+1}\right) &= f\left(x_i\right)+\frac{df}{dx}\Delta x+\cdots \\
\frac{df}{dx} &= \frac{f\left(x_{i+1}\right)-f\left(x_i\right)}{\Delta x}
\end{aligned}
$$

- Consider an ode: $\frac{dy}{dt}=f\left(t,y\right)$, a possible formula (Forward Euler scheme) for solving this ode for each timestep is:

$$y_{i+1}=y_i+\Delta t\cdot f\left(t_i,y_i\right)$$

- Can just iterate through this process until you reach the final time, need an initial condition to start
- Many other algorithms to solve this problem, some work better than others depending on what equation needs to be solved

Numerically Solving an ODE

- First discretize the ODE, note that you already have experience with discretized functions evertime you use a vector in numpy!
- Taylor expand a function about a point x_{i+1}, then can solve for the derivative:

$$
\begin{aligned}
f\left(x_{i+1}\right) &= f\left(x_i\right) + \frac{df}{dx}\Delta x + \cdots \\
\frac{df}{dx} &= \frac{f\left(x_{i+1}\right) - f\left(x_i\right)}{\Delta x}
\end{aligned}
$$

- Consider an ode: $\frac{dy}{dt} = f\left(t, y\right)$, a possible formula (Forward Euler scheme) for solving this ode for each timestep is:

$$y_{i+1} = y_i + \Delta t \cdot f\left(t_i, y_i\right)$$

- Can just iterate through this process until you reach the final time, need an initial condition to start
- Many other algorithms to solve this problem, some work better than others depending on what equation needs to be solved

Solving an ODE with scipy

- The scipy.integrate module has the function odesolve() that works very well for numerical ode solving
- This function will be able to solve a single or a system of **first** order differential equations

Solving an ODE with scipy

- The scipy.integrate module has the function odesolve() that works very well for numerical ode solving
- This function will be able to solve a single or a system of **first** order differential equations
- Necessary arguments in order:

Solving an ODE with scipy

- The scipy.integrate module has the function odesolve() that works very well for numerical ode solving
- This function will be able to solve a single or a system of **first** order differential equations
- Necessary arguments in order:
  - A function indicating the equation that you wish to solve takes arguments (y,t)

Solving an ODE with scipy

- The scipy.integrate module has the function odesolve() that works very well for numerical ode solving
- This function will be able to solve a single or a system of **first** order differential equations
- Necessary arguments in order:
  - A function indicating the equation that you wish to solve takes arguments (y,t)
  - Initial conditions (can be a vector if you have a system of equations)

Solving an ODE with scipy

- The scipy.integrate module has the function odesolve() that works very well for numerical ode solving
- This function will be able to solve a single or a system of **first** order differential equations
- Necessary arguments in order:
    - A function indicating the equation that you wish to solve takes arguments (y,t)
    - Initial conditions (can be a vector if you have a system of equations)
    - Interval to solve over (should be a range of values)

Solving an ODE with scipy

- The scipy.integrate module has the function odesolve() that works very well for numerical ode solving
- This function will be able to solve a single or a system of **first** order differential equations
- Necessary arguments in order:
  - A function indicating the equation that you wish to solve takes arguments (y,t)
  - Initial conditions (can be a vector if you have a system of equations)
  - Interval to solve over (should be a range of values)
  - Advanced options also available such as selecting a specific solver algorithm, step sizes for solving, etc.

Examples

- Lets start by solving basic a basic differential equation:

$$\frac{dx}{dt} = x\left(t\right)$$

Examples

- Lets start by solving basic a basic differential equation:

$$\frac{dx}{dt} = x\left(t\right)$$

- Equation of motion for mass-spring-damper system:

$$m\frac{d^2x}{dt^2} = -kx + c\frac{dx}{dt}$$

- Wait, but how do we solve a 2nd order ODE with a function that only solves first order equations?

Examples

- Lets start by solving basic a basic differential equation:

$$\frac{dx}{dt} = x\left(t\right)$$

- Equation of motion for mass-spring-damper system:

$$m\frac{d^2x}{dt^2} = -kx + c\frac{dx}{dt}$$

- Wait, but how do we solve a 2nd order ODE with a function that only solves first order equations?
- Introduce a new variables: $x_1 = x$, $x_2 = \frac{dx}{dt}$, then equation becomes:

Examples

- Lets start by solving basic a basic differential equation:

$$\frac{dx}{dt} = x\left(t\right)$$

- Equation of motion for mass-spring-damper system:

$$m\frac{d^2x}{dt^2} = -kx + c\frac{dx}{dt}$$

- Wait, but how do we solve a 2nd order ODE with a function that only solves first order equations?
- Introduce a new variables: $x_1 = x$, $x_2 = \frac{dx}{dt}$, then equation becomes:

$$\begin{aligned} \frac{dx_1}{dt} &= x_2 \\ m\frac{dx_2}{dt} &= -kx_1 + cx_2 \end{aligned}$$

Lab

- You will be solving equations of motion for the orbit of the earth around the sun
- The driving ode for this problem is Newton's universal law of gravitation:

$$m\frac{d^2r}{dt^2} = \frac{GMm}{r_{distance}^2}\hat{r}_{distance}$$

  - Where G is the gravitational universal constant
  - It is common to represent the value $GM = \mu$ for each planet or object involved.
- Need to solve for states of all planets together as need both planets states to calculate the next state of either!