Mini-Project for Phd-Python-1 Documentation

Release 0.0.1

Jilin Hu

Jul 28, 2016

CONTENTS

CHAPTER

ONE

PROBLEM STATEMENT

This mini-project relates to the Lorenz attractor. In this process, we need to solve the Ordinary Differential Equations (ODEs), write and read the data, and plot the data. By looking this problem, we can divide this problem into following steps.

1.1 ODE Solver

In this part, we construct a class named "ODE_solvers". In this class, there's a initial function which is used to specify sigma, beta, rho of the ODE and these three parameters are the attributes of this class. Moreover, we also define a function named "Euler_solve" which adopts Euler approach to solve this ODE by giving the initial conditions corresponding to values for (x[0], y[0], z[0]), time interval and number of steps. Of course, you can added some more advanced and accurate solve methods into this class, but here we just implement the Euler one to show how it works.

1.2 Filehandling

In this part, we implement a module named "filehandling" to take over the task of writing(reading)the trajectory from "Euler_solve" into(out) file.

1.3 Plotting

In this part, we implement a module named "plot" which can plot a 3D and 2D of the trajectory

1.4 Running

This is a integrated module, which integrate the solver, file writing and reading, and plotting. There are two methods in this module, "run_save_plot" and "run_load_plot". When you call "run_save_plot", it will automatically solving the ODE, saving the data into file and plot the trajectory. Otherwise, you can call "run_load_plot", it will load the data from the file you specified and plot the trajectory.

1.5 Testing

Here we conduct unit test for the functions mentioned above. All of these functions passed the unit test.

CHAPTER

TWO

LORENZ

lorenz

2.1 lorenz package

2.1.1 Submodules

2.1.2 lorenz.filehandling module

This file can contain functionalities for saving/loading data

Functions

write_x_y_z_tofile(var, filename)	Write the arrays of x, y, z to a file
read_x_y_z_fromfile(filename)	read x y z from file: filename

lorenz.filehandling.**write_x_y_z_tofile**(*var*, *filename*)
Write the arrays of x, y, z to a file

Parameters

- **var** – x, y, z
- **filename** – the target filename

Returns None

Return type None

lorenz.filehandling.**read_x_y_z_fromfile**(*filename*)
read x y z from file: filename

Parameters **filename** – The file which we want to read from

Returns three arrays: x, y, z

Return type numpy.arrays

2.1.3 lorenz.plot module

This file may contain functionalities for plotting

Functions

plot3Dpdf([x, y, z, sigma, beta, rho])	Plot the 3D scatter of lorenz
plot2Dpdf([x, y, z, sigma, beta, rho])	Plot 3*1 subplots of 2D plot for lorenz

lorenz.plot.**plot3Dpdf**(*x=None*, *y=None*, *z=None*, *sigma=10*, *beta=2.67*, *rho=6*)

Plot the 3D scatter of lorenz

Parameters

- **x** – x-axis data
- **y** – y-axis data
- **z** – z-axis data
- **sigma** – param sigma for lorenz
- **beta** – param beta for lorenz
- **rho** – param rho for lorenz

Returns bool

lorenz.plot.**plot2Dpdf**(*x=None*, *y=None*, *z=None*, *sigma=10*, *beta=2.67*, *rho=6*)

Plot 3*1 subplots of 2D plot for lorenz

Parameters

- **x** – x-axis data
- **y** – y-axis data
- **z** – z-axis data
- **sigma** – param sigma for lorenz
- **beta** – param beta for lorenz
- **rho** – param rho for lorenz

Returns bool

2.1.4 lorenz.run module

This file may contain a convenient interface/function for

1: computing a trajectory using an ODE solver from solver.py 2: save data to file 3: plot data

and possible another function that

2: load data from file 3: plot data

Functions

run_save_plot(sigma, beta, rho, initial_arrays)	Run Euler_solve, save and plot trajectory
run_load_plot(target_file, sigma, beta, rho)	Read and plot trajectory

lorenz.run.**run_save_plot**(*sigma*, *beta*, *rho*, *initial_arrays*)
Run Euler_solve, save and plot trajectory

Parameters

- **sigma** – sigma
- **beta** – beta
- **rho** – rho
- **initial_arrays** – [x[0], y[0], z[0]]

Returns the name of saved file.

Return type String

lorenz.run.**run_load_plot**(*target_file*, *sigma*, *beta*, *rho*)
Read and plot trajectory

Parameters

- **target_file** – The pickle file contain the traj of x, y, z
- **sigma** – the sigma used for this trajectory
- **beta** – the beta used for this trajectory
- **rho** – the rho used for this trajectory

Returns None

Return type None

2.1.5 lorenz.solver module

This file may contain the ODE solver

Classes

ODE_solvers(sigma, beta, rho)	An integrated ODE solvers

class lorenz.solver.**ODE_solvers**(*sigma*, *beta*, *rho*)
Bases: object

An integrated ODE solvers By giving the initial parameters, we can construct the ODE solver

Example: solver = ODE_solver(sigma, beta, rho)

Euler_solve(*initial_arrays*, *N=50000*, *t_sigma=0.01*)
Solve the equation with Euler approach

Parameters

- **initial_arrays** – list of initial values for x[0], y[0], z[0]
- **N** – the total steps to calculate the equation
- **t_sigma** – the minimal step

Returns arrays of [x, y, z], size(3*n)

Return type numpy.arrays

2.1.6 lorenz.util module

This file may contain utility functionalities to the extend you will need it

Functions

mkdir_p(mypath)	Creates a directory.

lorenz.util.**mkdir_p**(*mypath*)
Creates a directory. equivalent to using mkdir -p on the command line

2.1.7 Module contents

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

l