Interactive Python Computing Integration

BY CHU-CHJNG HUANG

Abstract

Traditionally, the whole scientific workflow usually requires versatile user interface to work with, professional softwares to do computation, analysis, and make data visualization, and office suite for publication. To build up such working environment, it almost requires plenty of resources involved.

After about ten years' development, IPython brought a breathtaking surge in this problem from its first release at 2012. IPython Notebook provides an interactive notebook interface that runs Python codes and generates the output in one's browser; moreover, output with pretty publication formats is also supported.

In this study, the project, "On-line Laboratory for Dynamical Systems On IPython" (DyLabIPy), will be introduced how to develop ubiquitous on-line Lab for Dynamic systems, with minimal resources but unlimited gains.

Key Words: IPython Notebook, DyLabIPy

Extended Abstract

Dynamical systems are one of important scientific researches including Mathematics, Physics, Biology, Management and almost other sciences. DyLabIPy is a project created on the purpose to integrate the mathematical theory and computer techniques to provide powerful tools applied to dynamical systems. The design of DyLabIPy is based on the Python trinity:

a) User Interface: ipython notebook interface;

b) Computational Capacities: numpy, scipy and sympy;

c) Data Visualization: matplotlib, pandas and seaborn.

First, IPython notebook interface brings the computations from backend (desktop environment) to frontend (internet environment); working, computing, displaying data are within the same IPython worksheet format, ipynb, which could retrieved by recent web client (whatever work online or offline). Secondly, the algorithms implemented in DyLabIPy are also optimized to make the computation more efficiently; the consideration of implements includes:

a) the symbolic computing capacities in DyLabIPy is based on SymPy with which we implement the functions capable of exploring the existence of theoretical solutions, analysing the stability of solution if exists, and also used to generated the form of numerical schemes, which makes the computation more reliably. Furthermore, we also develop the unique library, PySDE, in Python which can solves stochastic differential equations both symbolically and numerically.

b) As well-known fact, the computing velocity by scripting Python is less efficiient than that by compiled language; for the purpose of improving computing efficiency, all the numeric schemes implemented in DyLabIPy were developed by vectorizated method which also keep the codes clearer and more readably.

```
(Bad: Computation by loop)
         for i in range(n):
             do u[i+1]=f(u[i]) + ...

(Good: computation by vectorization method)
         u[1:n] += func(u[0:n-1]) + ...
```

c) The functionalities of data visualization in DyLabIPy mainly come from Python plotting libraries and IPython's versatile display function, especially on animation and interactivity. JSAnmation implements an HTML/Javascript writer for Matplotlib animations. The result can be embedded in a web page, or an IPython notebook. The drawbacks are that would make the worksheet too big to run and is for Python-2.x only but not for Python-3.x. The purely python function, movie.py, in DyLabIPy is developed to make animation by playing a sequence of static pictures, created by Python, which is very suit for cross-platform use since it is independent of other application and independent of Python version. With IPython's experimental widget module, we can make sophisticated interactive plot; widgets with control buttons are drawn in a special widget area, Changing a widget's attribute will automatically update that widget everywhere it is displayed in the notebook.

d) It's a long road from modelling and computation to publication and we almost requires manipulate different softwares in the processing work. With IPython notebook, we can work with DyLabIPy like as "what you compute what you get", i.e the worksheet we used to make computation is as the same as we want to make publication. Incorporated with IPython's Nbconvert module, we could convert the computational result directly into publication formats, TEX, html, or slides etc. In DyLabIPy project, we also simplify the documentation conversion work and create template to make the output capable of being displayed offline if bandwidth is limited.

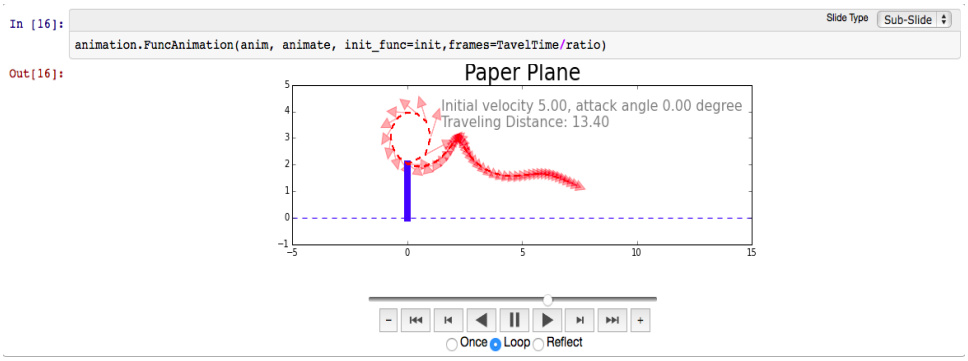

Figure 1. The animation of computational result of phugoid model was generated by JSAnimation module;

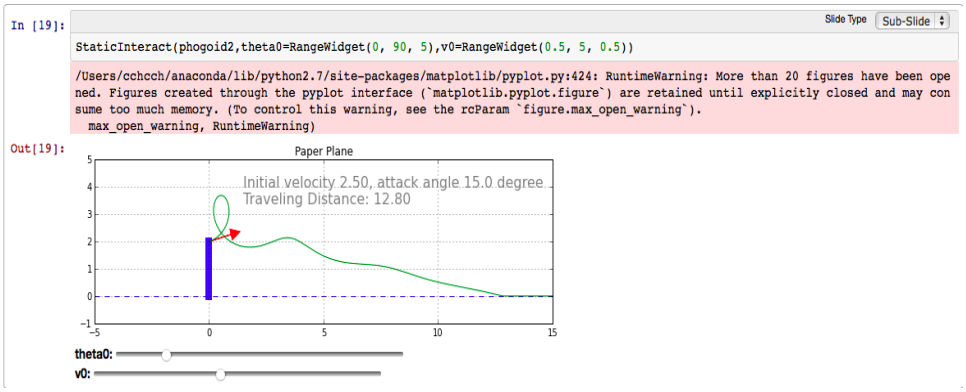

Figure 2. Changing the values of initial velocity and attack angle in widget's sliders would update the simulated flight trjaectory of of phugoid.

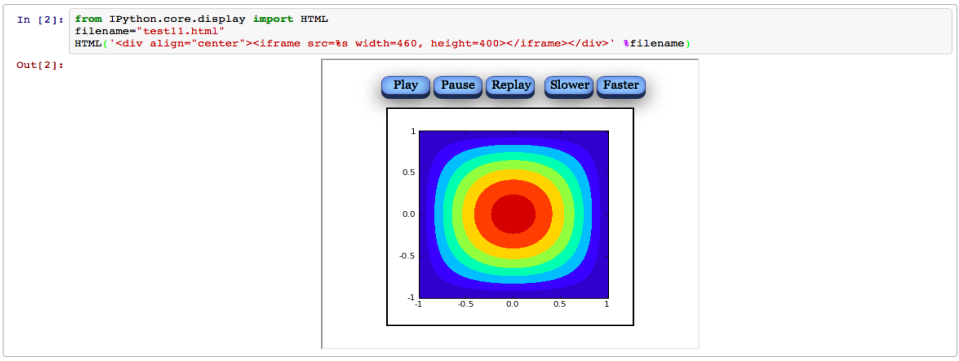

Figure 3. Movie.py was used to play a sequence of pictures which are simulated heart beat of cardiac excitation in cubic region.

Apart from developing the core functionalities of DyLabIPy, we also make some studies on how to promote Python computing in educational use and get some efforts, especially at the topics:

a) Packing fully-feature Python platform into USB stick; this self-contained devices actually help us to promote the lectures with scientific computations in Chang-Gung University;

b) how to equip single-user IPython Notebook to multi-user mode; it is useful to set up the public Python platform in group-cooperative project and for educational purpose;

c) setup the code repository on Github; a minimal procedure but to reduce the loading of the codes management.

IPython provides a rich architecture for interactive computing; this is the reson why we decide to develop DyLabIPy based on this architecture and it actually shorten the period of development time. Thought IPython is known as a rich architecture for “Interactive Pthton computing” with, it is also the best one for "Integrating" computing with, not only for Python-clone softwares but also for others.

Key Words: IPython Notebook, DyLabIPy, Vectorization Calculation

[**DyLabIPy**] https://github.com/cchuang2009/DyLabIPy