JupyterLab: The Evolution of the Jupyter Notebook

The JupyterLab Team

Chris Colbert, Jupyter
Steven Silvester, JPMorgan Chase
Afshin Darian, Two Sigma
Ian Rose, Berkeley
Jason Grout, Bloomberg
Brian Granger, Cal Poly
Jessica Forde, Jupyter
Grant Nestor, Cal Poly
Cameron Oelsen, Cal Poly
Fernando Perez, LBNL/Berkeley
Cal Poly Interns
The Larger Jupyter Team

@jupyterlab on GitHub
@ProjectJupyter on Twitter

The Jupyter Notebook

Jupyter Notebook

Interactive, Exploratory, Reproducible

- Interactive, browser-based computing environment
- Exploratory data science, ML, visualization, analysis, stats
- Reproducible document format:
  - Code
  - Narrative text (markdown)
  - Equations (LaTeX)
  - Images, visualizations
- Over 100 programming languages
- Everything open-source (BSD license)

Jupyter Notebook

A Jupyter Notebook document with a visualization of measles data.

Building Blocks for Interactive Computing

Building Blocks

File Browser

Notebooks

Terminal

Text Editor

Kernels

Output

Introducing JupyterLab

JupyterLab: Integrated Experience

File Edit View Run Kernel Tabs Settings Help

Files | Running | Commands | Cell Tools | Tabs

Lorenz.ipynb

Code | Python 3

We explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

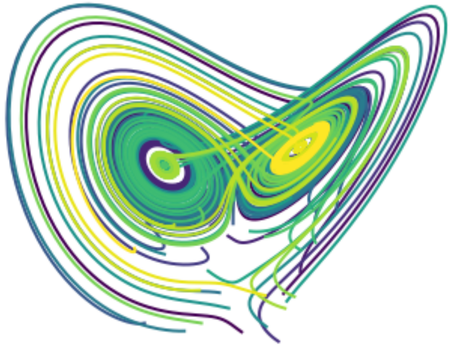
Let's change (σ, β, ρ) with ipywidgets and examine the trajectories.

In [2]:
```
from lorenz import solve_lorenz
w=interactive(solve_lorenz,sigma=(0.0,50.0),rho=(0.0,50.0))
w
```

sigma 10.00

beta 2.67

rho 28.00



lorenz.py

```
def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""

    max_time = 4.0
    N = 30

    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))

    # Solve for the trajectories
    t = np.linspace(0, max_time, int(250*max_time))
    x_t = np.asarray([integrate.odeint(lorenz_deriv, x0i, t)
                      for x0i in x0])

    # choose a different color for each trajectory
    colors = plt.cm.viridis(np.linspace(0, 1, N))

    for i in range(N):
        x, y, z = x_t[i,:,:].T
        lines = ax.plot(x, y, z, '-', c=colors[i])
        plt.setp(lines, linewidth=2)
    angle = 104
    ax.view_init(30, angle)
```

JupyterLab

Building Blocks

- Work with the building blocks in a flexible and integrated manner
- Modern JavaScript development: npm-based packaging, Typescript, phosphor.js
- Clean model/view separation
- Well-separated public/private APIs
- Fully extensible by third parties
- High performance
- Design!

July 2019

JupyterLab Today

- https://github.com/jupyterlab
- ~5 years worth of development
- ~250 contributors, ~70 components
- ~5,600 releases (npm+python)
- ~16,000 commits, ~classic notebook
- Currently 1.0, ready for use

Roadmap

JupyterLab 1.0: Use It Today

conda install -c conda-forge jupyterlab
or pip install jupyterlab

- Eventually:
  - Classic notebook will be retired

Live Demos!