Computational Lab Ω: Solving ODEs and Understanding Chaos

Due February 30th, 2020 (@25:00)

- Read this document and do its suggested readings to help with the pre-labs and labs.
- Ask questions if you don't understand something in this background material: maybe we can explain things better, or maybe there are typos.
- Whether or not you are asked to hand in pseudocode, you need to strategize and pseudocode before you start coding. Writing code should be your last step, not your first step.
- Test your code as you go, not when it is finished. The easiest way to test code is with print('') statements. Print out values that you set or calculate to make sure they are what you think they are.
- Practice modularity. It is the concept of breaking up your code into pieces that as independent as possible form each other. That way, if anything goes wrong, you can test each piece independently.
- One way to practice modularity is to define external functions for repetitive tasks. An external function is a piece of code that looks like this:

```
def MyFunc(argument):
    """A header that explains the function
    INPUT:
    argument [float] is the angle in rad
    OUTPUT:
    res [float] is twice the argument"""
    res = 2.*argument
    return res
```

In this lab, you will implement your own RK4 integrator and likely a wrapper to integrate your ODEs. You will then use this RK4 integrator many times. You should be able write the integrator to take any set of ODEs and solve them without changing the integrator code. You will lose 'code' marks if there is excessive copy pasting of code!

Physics background

Chaotic system For any chaotic system, the model should satisfy the following properties:

- For any two different individual initial conditions, the motion in this vector field will be completely different no matter how small the difference is. In another word, the future development is very sensitive to the initial conditions.
- For any large set of data, the over all solutions have some special trends. In another word, the chaotic system is statistically stable.

Lorenz attractor Lorenz attractor is an attractor from a system of three non-linear ordinary differential equation studied by E. N. Lorenz in 1963 to explain some of the unpredictable behavior of the weather.

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = rx - y - xz$$

$$\frac{dz}{dt} = xy - bz$$

The result led to a three-dimensional system of differential equations which involve three parameters: the Prandtl number σ, the Rayleigh number r, and another parameter b that is related to the physical size of the system. Notice that all three parameters are assumed to be positive and $\sigma > b + 1$.

Rossler attractor Rossler attractor is an attractor from a similar system as the Lorenz attractor that studied by Otto Rossler. The system of three non-linear ode are shown below:

$$\frac{dx}{dt} = -y - z$$

$$\frac{dy}{dt} = x + ay$$

$$\frac{dz}{dt} = b + (x - c)z$$

Here, a, b and c are three parameters.

Computational Background

Runge-Kutta The Runge–Kutta fourth order method (RK4) is a widely used algorithm for solving systems of ODEs. In general it is much more accurate than the Euler or Euler–Cromer methods for ODEs used in initial value problems.

The RK4 method involves using the model function to predict the next time step in the ODE (as in the Euler method) but using those predictions not as final value, but as a way to improve the prediction using intermediate values to better approximate the ODE evolution.

As discussed in the text, this method involves calculating the RHS vector of $\frac{d\vec{r}}{dt} = \vec{f}(\vec{r}, t)$ at various intermediate points between steps. Full implementation requires coding the following 5 lines which are iterated over t values:

$$\overrightarrow{k_1} = h\vec{f}(\vec{r}, t)$$

$$\overrightarrow{k_2} = h\vec{f}\left(\vec{r} + \frac{1}{2}\overrightarrow{k_1}, t + \frac{1}{2}h\right)$$

$$\overrightarrow{k_1} = h\vec{f}\left(\vec{r} + \frac{1}{2}\overrightarrow{k_2}, t + \frac{1}{2}h\right)$$

$$\overrightarrow{k_1} = h\vec{f}\left(\vec{r} + \frac{1}{2}\overrightarrow{k_3}, t + h\right)$$

$$\vec{r}(t+h) = \vec{r}(t) + \frac{1}{6}(\vec{k}_1 + 2\vec{k}_1 + 2\vec{k}_1 + \vec{k}_1)$$

3D Plot In order to plot a 3 dimensional graph, we have to import some other functions to help us.

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
```

For the plotting, we can use

```
Axes3D.plot(xs, ys, *args, **kwargs)
```

Where xs, ys are coordinates of vertices in array and zs can be places at the position of *args. Also the axis labeling is different from the original one. Instead of just using plt.xlabel(), we are running:

```
Axes3D.set_xlabel('x')
```

Discrete Fourier Transforms This lab focusses on applications of the Fourier Transform. See §§ 7.1 and 7.2 of the textbook for general statements about the Discrete Fourier Transform (DFT). In particular, the

DFT finds the coefficients c_k for a set of discrete function values y_n through:

$$c_k = \sum_{n=1}^{N-1} y_n \exp\left(-i\frac{2\pi kn}{N}\right)$$

If the function if real, then the coefficients in the range $N/2 \to N$ are just the complex conjugates of the coefficients in the range$0 \to N/2$. i.e.:

$$c_{N-i} = c_i^*$$

This means we only have to calculate $\sim N/2$ coefficients for a real function with N values.

Since the Fourier coefficients are complex, we usually plot their absolute value vs. k to make a Fourier Transform plot. The c_k's with large magnitudes represent periodicities with k values which dominate the signal. The k value is related to the period n_{cyc} of the signal through:

$$\left(\frac{2\pi k n_{cyc}}{N}\right) = 2\pi \Rightarrow n_{cyc} = \frac{N}{k}$$

The DFT requires $O(N^2)$ evaluations of $\exp\left(-i\frac{2\pi kn}{N}\right)$. This is a lot and would not make the DFT useful for many operations. Luckily, the Fast Fourier Transform (FFT) is an algorithm that rearranges the DFT to make it much faster. When this faster algorithm is implemented, you require only $Nlog_2N$ evaluations of $\exp\left(-i\frac{2\pi kn}{N}\right)$. This is significantly fewer and hence significantly faster.

Although doable, you should never really code an FFT yourself, because the amount of things to keep track of is dizzying. Instead, there are standard library functions in python (as well as other programming languages) for the FFT.

Question

(a) Solve the Lorenz attractor equations with the following input parameters: $\sigma = 10, r = 28, b = \frac{8}{3}$. The particle starts at the position (-10, 10, 25). By choosing appropriate time interval, integrating over 100 second. Make a 3D plot.

SUBMIT YOUR CODE AND PLOT

(b) Make a plot of x-axis verse time. Comment on the shape of the graph. Calculate the coefficients of the discrete/fast Fourier transform of the data using the rfft function from numpy.fft and make sure that inverting with irfft returns you to the original data. Make 3 plots that set all but the first 2% and 4% of the Fourier coefficients to zero, inverse the FFT and plot the new dataset on the same graph as the original data. Comments on the changes as we increase the amount of the Fourier coefficients.

SUBMIT YOUR CODE, PLOT AND WRITTEN ANSWERS TO QUESTION

(c) Make another plot with the particle starts at the position (-10, 10, 25.001). Compare this with part (a) and comment on the differences and similarities.

SUBMIT YOUR PLOT AND WRITTEN ANSWERS TO QUESTION

(d) Create 100 particles at the initial position from (-10, 10, 25) to (-10, 10, 26) by changing only the z coordinate. Record the ratio of points passing through a suitable region and return them as an array. (Suitable means there particle must pass through that region in order to get any valid result) Comment on what you observe. (This may takes several minutes to run. If you like you can try using a longer time interval instead of 100 seconds and you can see a more convergent answer)

SUBMIT YOUR CODE AND WRITTEN ANSWERS TO QUESTION

(e) Combine your answer from part (c) and (d). Explain whether the Lorenz attractor is a chaotic system.

SUBMIT YOUR WRITTEN ANSWERS TO QUESTION

(f) Repeat part (a) to (e), except (b), on Rossler attractor with the following setup.

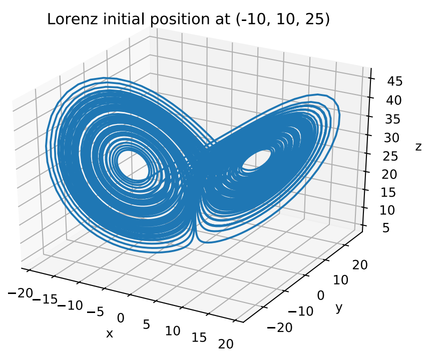
- On part (a), use $a = b = 0.1, c = 14$ and initial position (-10, 10, 10), integrating over 200 second
- On part (c), use initial position (-10, 10.01, 10)
- On part (d), use range from (-10, 10, 10) to (-10, 11, 10)

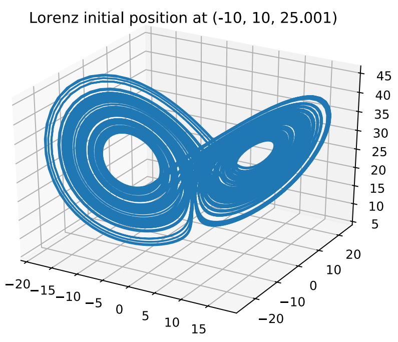
SUBMIT YOUR CODE, PLOT AND WRITTEN ANSWERS TO QUESTIONS

(g) Discuss any real life application of the chaos theory or any physical meaning on the research of chaos theory. You may include your own opinions and any suggestions from articles, movies and papers.

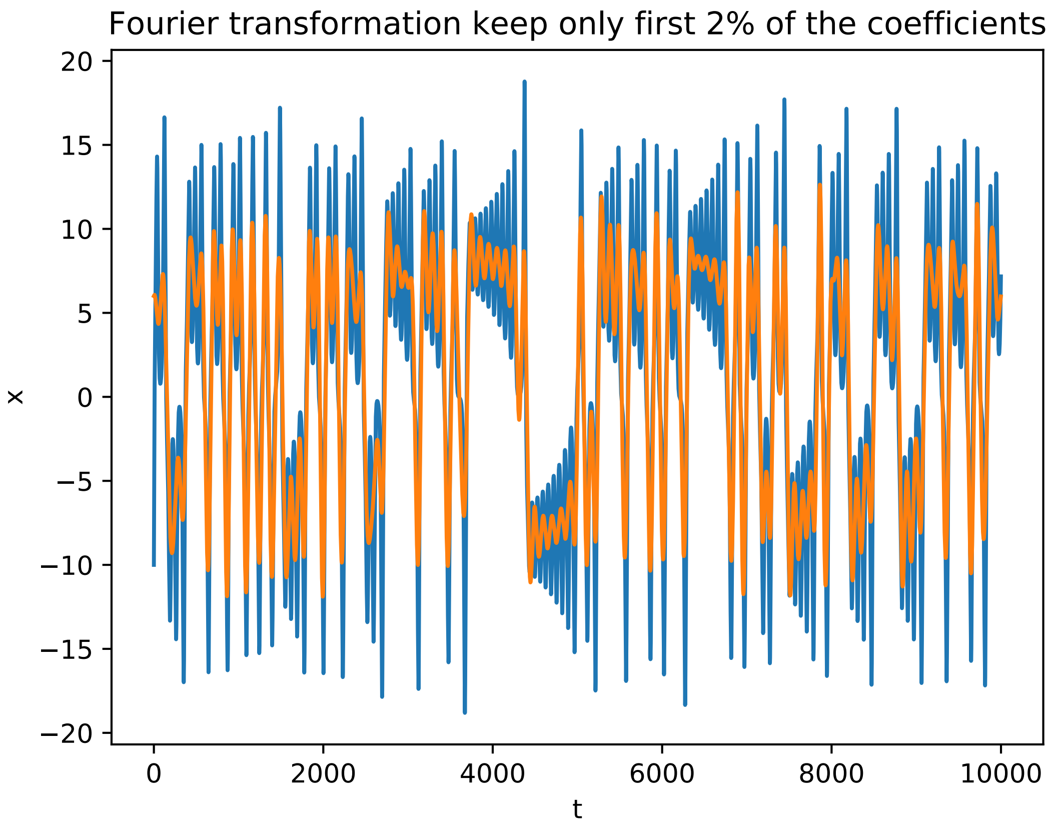
Answer

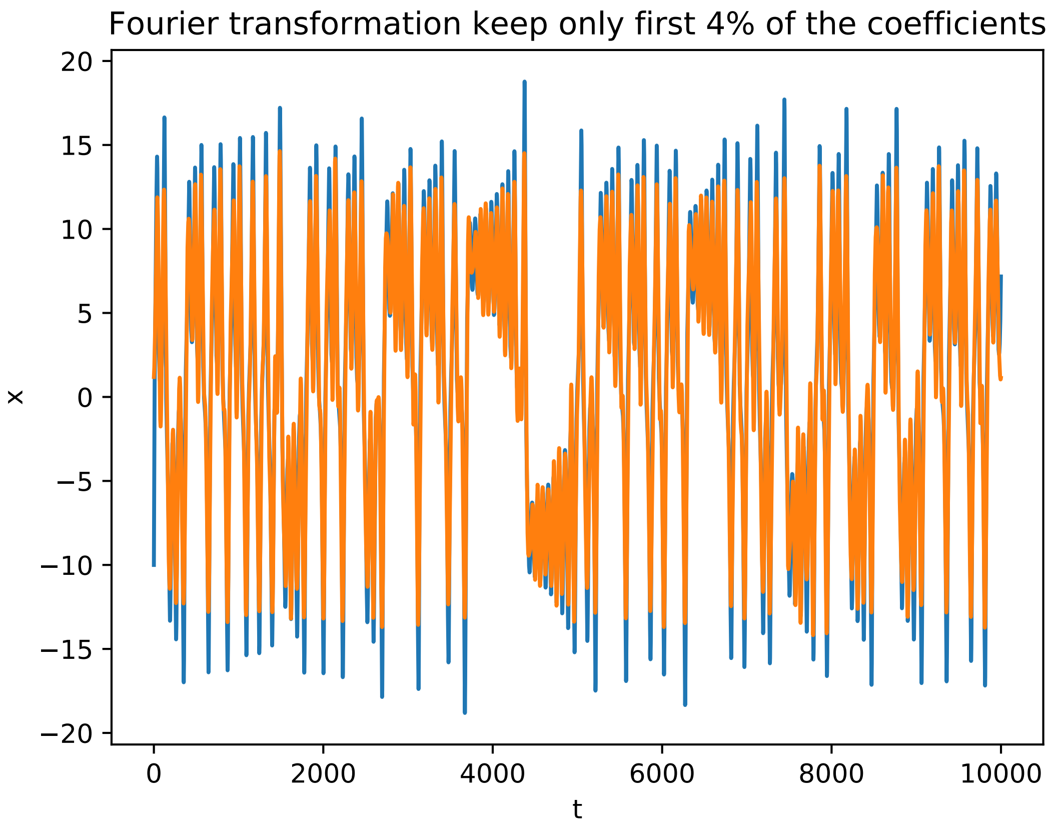
(a) (Students' answers should clearly show the shape of Lorenz attractor.)





(b) (Students' answer must match the graph and explains the development of Fourier transformation.)





By comparing the 2% graph and the 4% graph, we can see that first 2%, which is around 10 Fourier coefficients, has already set up the shape of this function. Then when we move on to 4%, which are around 20 Fourier coefficients, the most significant change is the amplitude of details is filling up.

(c) (Students should answer about the difference between two graphs.)

By comparing to part (a), we can see that the thickness and pattern are quite different. But overall the size and shape are the same. This clearly shows that the slightly change in the initial position, even 0.001 can make a complete different development in the future.

(d) (Students must choose a valid region, which there is appearance of points inside and get similar ratio.)

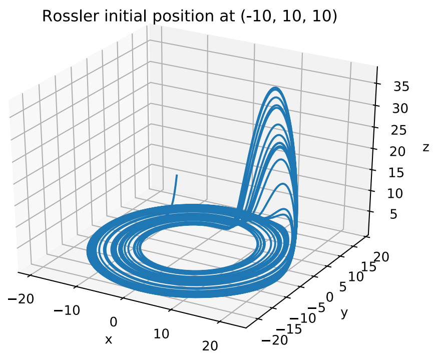
I record the ratio of points passing through the region of $|x| \leq 5, |y| \leq 5$. Comparing the ratio of that certain region, we can see that the percentage of points contained in that region is between 31%~37%. Although, by part (c), we know that all of these particles are moving in completely different routes, but there is come kind of structural stability on this model.

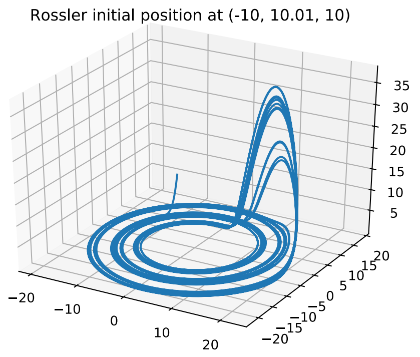
Data: [0.3349, 0.3349, 0.3444, 0.3282, 0.3852, 0.3578, 0.3404, 0.3243, 0.3415, 0.3322, 0.3442, 0.3382, 0.3443, 0.3638, 0.3368, 0.3415, 0.3655, 0.3407, 0.3641, 0.326, 0.3697, 0.3644, 0.3624, 0.3436, 0.3339, 0.3583, 0.351, 0.3556, 0.3588, 0.3154, 0.3751, 0.3419, 0.3543, 0.3552, 0.3479, 0.3338, 0.3722, 0.3562, 0.3626, 0.3357, 0.3436, 0.3491, 0.3513, 0.3294, 0.3413, 0.3548, 0.3404, 0.3731, 0.3528, 0.3559, 0.3674, 0.3791, 0.3485, 0.3514, 0.3539, 0.3739, 0.3639, 0.343, 0.3528, 0.3528, 0.3479, 0.3441, 0.3605, 0.3576, 0.3762, 0.3703, 0.3149, 0.3397, 0.3499, 0.3648, 0.3483, 0.3453, 0.3345, 0.3471, 0.3336, 0.3649, 0.3063, 0.3383, 0.3261, 0.3479, 0.3436, 0.3587, 0.3408, 0.3139, 0.3568, 0.3451, 0.3563, 0.3431, 0.3561, 0.3489, 0.3506, 0.3391, 0.3475, 0.3539, 0.3616, 0.3315, 0.3221, 0.3376, 0.3544, 0.3354]

(e) (Students should explain the answer with the properties of chaotic system)

Combine our observation from part (c) and part (d), we notice that the Lorenz attractor satisfies the basic properties of chaotic system. Firstly, the future motion is very sensitive to the initial condition. A small change can cause a complete different future. On the other hand, it's hard to believe that this chaotic system performs a structural stability. No matter how different the motion is, the distribution of points are nearly the same.

(f) (Students must perform similar actions as part (a) to (e), except part (b))





Although these two graphs look quite similar, we can still see some differences that on the peak and the circular loop. On the ratio side, I record the ratio from $10 \leq x \leq 15, 5 \leq y \leq 10$, we can see that the answer is around 2%~3%. Therefore the sensitivity of initial condition and stability of structure are also presented in Rossler attractor.

Data: [0.0295, 0.0295, 0.0302, 0.0296, 0.028, 0.0259, 0.0273, 0.0268, 0.026, 0.029, 0.0271, 0.0271, 0.027, 0.0271, 0.0281, 0.0293, 0.029, 0.028, 0.0277, 0.0289, 0.0306, 0.0267, 0.0303, 0.0299, 0.0291, 0.0344, 0.0265, 0.0278, 0.0272, 0.028, 0.0281, 0.027, 0.0252, 0.0259, 0.0276, 0.0253, 0.0243, 0.0265, 0.0247, 0.0277, 0.0266, 0.0296, 0.0277, 0.0244, 0.0309, 0.0259, 0.0286, 0.0284, 0.0276, 0.0253, 0.0292, 0.0265, 0.0271, 0.0273, 0.0286, 0.0291, 0.0288, 0.0281, 0.027, 0.0274, 0.0271, 0.0296, 0.0302, 0.0281, 0.0262, 0.0264, 0.0275, 0.0287, 0.0297, 0.03, 0.0297, 0.03, 0.0302, 0.0302, 0.0303, 0.0302, 0.0303, 0.0303, 0.0303, 0.0303, 0.0303, 0.0303, 0.0302, 0.0302, 0.0302, 0.03, 0.0297, 0.0298, 0.0297, 0.0293, 0.0287, 0.0272, 0.0264, 0.0265, 0.0289, 0.0304, 0.0279, 0.0279, 0.0275, 0.0276]

(g) (Open Question)

The significance of the Chaos theory is that, previously we are more focusing on prediction and accuracy. We are trying to construct a lot of equations or formulas that given the information at a certain time, we can predict every motion in the future. But what Chaos theory shows us that it is impossible to predict accurately since any uncertainty can cause a completely different result. Therefore nowadays our researches are focusing more on statistical results, probability and so on.

The butterfly effect (may be mentioned by a lot of students) can be explained in a new perspective. The original statement says, “A tornado may be influenced by the flapping of the wings of a butterfly”. But since we know the statistical stability of the chaos system, the influence from a butterfly only changes the order of weather, which make the tornado arrives later or earlier because the probability of a certain weather will not change no matter how the initial condition change.