Solving Dynamical Systems in Python

Cody Riley

8/10/2022

Chapter 1

The Development Phase

1.1 Stages of Development

The development process of my dynamical system solver was relatively straightforward in the early stages of development. The first stage of development was to build a solid plan for what I planned to make. As discussed in my statement of intent, I wanted to develop an easy-to-use piece of software that would allow any user to input a set of first-order differential equations and have the python script approximate a solution to the corresponding dynamical system. In the early stages of planning, I thought it would be good to allow the user to input how many equations they would like to include in the solver. However, although the final build allows this to occur, it is more likely for bugs to occur, and the app becomes very tedious to use.

At first, I thought building such a program would be relatively straightforward, and I assumed I would have no issues. However, it proved more difficult than I anticipated, as I had to ensure that the correct number of equations were being generated alongside the number of variables. The name of the variables was also a challenge, and for simplicity, during the early stages of development, I decided to use the variable naming convention of x_n, where $n \in \mathbb{Z}^+$.

1.2 Positive Outcomes

There were many positive outcomes in the development of my Dynamical Systems app. Firstly, I successfully allowed users to enter a system of n equations in the terminal version of the application.

1.3 Challenging Parts of Development

Handling discontinuities of functions a user inputs was one of the biggest challenges in the development of the Python application. For example, say you had a simple one-dimensional problem, equivalent to

$$\frac{dy}{dt} = \frac{1}{t}$$

over a timespan of $[-5, 5]$ (this is just the domain we will plot over). In this case, we would encounter an error because our solver cannot solve the differential equation for $t > 0$ and hence we are unable to plot an accurate solution to that part of the solution.

1.4 Future Developments for the Project

I plan to work on this project further past the deadline. I just like to have projects to do in my spare time, and there are a few features I will be adding to my project. Firstly, I would like to add some functionality that I ran out of time to add. The plan was to give the user the option to add as many equations as they would like to solve the dynamical system. This would result in a much more complex system being solved. This functionality works in the terminal version of the program, however, I just ran out of time to give the GUI the ability to change how many equations the algorithm had to solve. I would also like to add the ability for the user to clear the current plot showing on the GUI, as this would result in better accessibility to the overall use of the tool. For example, at the moment, the current build of the Dynamical Systems Solver App only allows for you to plot a single graph on the canvas, and after plotting this graph you would have to restart the app to plot the next one. I have tried a few approaches to this problem already,

Chapter 2

How to Use the Application

As mentioned in the development process of the program, there are two different versions of the program that has been developed. One version uses the terminal and has more capability, and the other version takes advantage of a GUI but currently has less functionality.

The first version we will discuss is the version involving only the terminal. The first instruction that a user is prompted with is how many equations they would like to enter. There is no limit to the number of equations a user can enter, however, it is highly recommended that at maximum three equations are entered. Next, the user will be prompted to enter a lower bound t_{min} which is required to be a float. After successfully entering a lower bound, the user will be prompted to enter an upper bound t_{max} which requires the same conditions as t_{min}.

The user will then be prompted with a statement saying that they are solving a dynamical system with n equations, where n is the number entered in the first prompt. In the same statement, the variables generated will be defined. This has the form of a vector $x = (t, x_1, \cdots, x_n)$. For example, if we were to enter $n = 2$, then the variables are defined to be $x = (t, x_1, x_2)$ where t is the time component. Next, the user will be asked to define their equations in-terms of the variables that have been defined. In mathematical terms, this is denoted as $f_n(x)$. After the user has entered all the equations, then the user will be prompted to enter initial conditions for all the solutions. These initial conditions are defined to occur at $t_0 = t_{min}$. For example, if $t_{min} = -5$, then $x_n(t_{min}) = \alpha_n$.

The final step that we need to take is to enter the number of steps we wish to use to numerically approximate a solution to the dynamical system. This will then be used to split our time span into k sub intervals.

Example Calculation

We will consider a basic example for our calculation. We will consider the case where the number of equations we enter is $n = 2$. The timespan that we will be considering is $\mathbb{T} = [-5, 5]$ which means the lower bound is -5 and the upper bound is 5.

GUI Version

The graphical user interface version of the dynamical system solver application is much more accessible and easier to use compared to the terminal version.