NE 336: Lab 2 Group 1

Pendar Mahmoudi

IVPs

Introduction – Solving single ODE using builtin methods

In class, we have seen the `solve_ivp` method from scipy but we havent looked at all the ways we can customize it. When we look at the documentation

```
scipy.integrate.solve_ivp(fun, t_span, y0, method='RK45', t_eval=None,
    dense_output=False, events=None, vectorized=False, args=None, **options)
```

Where python uses the notation t for the independent and y for the dependent variables.

The ODE fun

You need to write a function, for example `myode(t,x)`, describing the equation. This function must always take arguments at least two arguments such as t and x (where t is the independent variable and x is the dependent variable or a vector of dependent variables) and return the corresponding value of dx/dt.

If the independent variable does not show up in the ODE, you can either just not use it or write the function as `myode(_,x)` (the `_` tells python there will be an input argument but it can be discarded). Please note that `myode(x)` will not be correct in this case since now x is assume to be the independent variable. So either write `myode(t,x)` or `myode(_,x)`.

t_span

This depends on the problem you are trying to solve. The time span is from t_0 (which is not necessarily zero but may be) up to t_f. If you are not provided with t_f solving until steady state is reached is a good idea.

y0

This is the value of your dependent variable at t_0. This would be $y(t_0)$ based on python notation.

The rest of the arguments are optional and we are going to test out a few shortly.

The solution

Please note that for the solution returned from `solve_ivp`, `sol.t` are the values for the independent variable, which are chosen suitably to achieve convergence. `sol.y` are the integrated values for the dependent variable at the points in `sol.t`.

One thing to keep in mind is that no matter what the name of your independent or dependent variable are, these methods of calling the solution from `solve_ivp` remain the same.

Let's start testing these different parts out with a simple example where the ODE has a parameter we would like to modify in our solution.

Task 1 : solving single odes

This question has a template file attached to the dropbox which you can use to help you.

Use the in-built solver `solve_ivp`, we would like to solve

$$dx/dt = \beta x \sin(\beta t)$$

with $\beta = 2$ subject to initial condition $x(t = 10) = 5$. Solve this from t=10 to t=20.

Note that you may define the ODE using `lambda` functions (even as an argument) or a full function definition. Both methods will give same results.

Submit the work for this task as `single_ode.py`. Make sure it includes the solution to the following tasks.

1. Define the ODE by entering in the number for β in the definition (not as an argument) in the function. This function will be called `ode_fun_1` and its solution `sol1`. Solve and plot. Make sure to apply appropriate labels etc to all your graphs.

2. Use your results to determine $x(t = 15)$. Do not simply read the graph! Try changing the timespan to be a vector of times that you would like the solution at (change the `t_eval` parameter). Include this as a print statement.

3. Read the documentation to find out how to pass the parameter β to `solve_ivp`. So now your ode should take in three inputs (t, x, β). This function will be called `ode_fun_2` and its solution `sol2`. Solve and plot for $\beta = 2$ in a new figure. (Hint : look at the args parameter).

4. Now that you have figured out how to pass β as a parameter to the solver, solve this ode for a range of values for β. Specifically, solve the ode for

   $$\beta = 2, 4, 6, 8$$

   and plot them all on one figure. You can use `ode_fun_2` to solve for the various values for β.

5. **Bonus**: can you get the labels for the plots in the previous task to show the value of β for each solution?

Systems of Equations

As you have seen in the documentation, python's in-built functions can also solve systems of equations. Call the function in the same way, however the dependent variable x is now a vector (can be given using a collection of your choice) `[x1,x2,x3]`,the initial conditions x[0] should be a corresponding vector `[x1[0],x2[0],x3[0] ...]` and the function `solve_ivp` should return a vector `[dx1/dt,dx2/dt,dx3/dt ...]`.

Task 2 : solving system of odes

Petrarch

The following model is proposed to represent the relationship between the inspiration (Z) of the poet Petrarch, the magnitude (P) of Petrarch's love for Laura (his muse), and the magnitude (L) of Laura's love for Petrarch:

$$\begin{aligned}
\frac{\mathrm{d}L}{\mathrm{d}t} &= -3.6L + 1.2\left(P\left(1-P^2\right)-1\right) \\
\frac{\mathrm{d}P}{\mathrm{d}t} &= -1.2P + 6\left(L + \frac{2}{1+Z}\right) \\
\frac{\mathrm{d}Z}{\mathrm{d}t} &= -0.12Z + 12P
\end{aligned}$$

Where the time, t, is measured in years.

- Using `solve_ivp` and the initial conditions L(0)=P(0)=Z(0)=0 , simulate Petrarch's emotions (i.e., solve the equations) over a 21-year period.
- Make plots to illustrate your results :
  - Plot L, P and Z over those 21 years.
  - Plot Petrarch's love for Laura (P) vs the inspiration (Z).
  - Plot Laura's love (L) vs Petrarch's love for Laura (P).
- **Bonus** Reading the original research paper for these equations (which is attached to the dropbox) or by the graphs you have plotted, answer this important question as a comment in your code : Did Laura ever come to love Petrarch as well? Explain your reasoning!

Submit the work for this task as `Petrarch.py`.

BVPs

In class we talked about changing higher order ODEs into a system of first order ODEs to be able to solve them.

We would like to solve

$$\frac{d^2y}{dt^2} + 0.1\frac{dy}{dt} + 9.81\sin(y) = \cos(t)$$

Which describes a pendulum with damped motion and an external force. The boundary conditions are $y(t = 0) = \pi/6$ and $y(t = 2\pi) = \pi/6$. Use $u = \dfrac{dy}{dt}$ and substitute in main ODE to get your system of ODEs.

Solve this system using any one of the approaches we talked about in class for the shooting method that would be applicable to this problem here and plot y vs t.

Bonus Show the same results when the external force and damping are removed.

Submit as `pendulum.py`.

How to submit your answers to the lab

Please keep these points in mind as you go on to complete the lab questions.

- When working on these questions, please ask for help from the TAs and myself as needed.
- Discussion with classmates is encouraged but takings somebody else's code and submitting their work as your own is not acceptable. Please ask for clarification on this item if unsure.
- In terms of formatting your submission:
  - If the question is asking for a script or function, please submit as a python file.
  - If asked for analysis, you may include this as comments in your python file, handwritten work or however else you wish.
- Please include your name and student ID in each file that you submit.

Checklist of submission items

This is included for your convenience. Parts marked for bonus are a maximum total 10% overall.

1. `single_ode.py`.
2. `Petrarch.py`.
3. `pendulum.py`.