PHYS358: Homework 2 (due Sep 11, 9:30am)

Ordinary Differential Equations II

Note: Please read the complete homework instructions before you start.

We will develop a solver for boundary value problems (BVPs), more specifically, for BVPs with eigenvalues, or Sturm-Liouville (SL) problems. The general form of a SL problem is

$$\frac{d}{dx}\left(p(x)\frac{d}{dx}y(x)\right) + \lambda w(x)\, y(x) = 0, \tag{1}$$

where λ is the eigenvalue of the SL problem. The boundary conditions restrict the eigenvalues to an increasing sequence of positive numbers λ_m. The corresponding eigenfunctions $y_m(x)$ will be orthogonal with respect to $w(x)$. The prototypical example is the (standing) wave equation

$$y"(x) + \lambda\, y(x) = 0, \tag{2}$$

which we will discuss in this homework. A fancier (and much more painfull to solve) example is the Legendre equation

$$(1-x^2)f"(x) - 2xf'(x) + l(l+1)f(x) = 0 \tag{3}$$

with $x \equiv \cos\theta$, describing the polar angle dependence of the hydrogen wave function. The homework consists of several parts, discussing first the analytical solutions, and then the implementation of the BVP solver, including tests.

(2a) Analytical solution via Laplace transforms [5 pts]: We rewrite the standing wave equation as

$$y"(x) + (\lambda\pi)^2\, y(x) = 0, \tag{4}$$

with $0 <= x <= 1$, and the boundary conditions $y(0) = 0$ and $y(1) = 0$. Calculate (with pencil and paper) the Laplace transform of the above wave equation, and thus find the Laplace transform of $y(x)$. As a reminder, the Laplace transform of a function $f(x)$ is given by

$$\mathcal{L}_s(f) = \int_0^\infty f(x)\, e^{-sx} dx. \tag{5}$$

Hint: Set the boundary value for the slope $y'(0) \equiv a$.

Solution: The Laplace transform of the second derivative is

$$\mathcal{L}_s[y"] = -y'(0) - sy(0) + s^2\mathcal{L}_s[y],$$

with the boundary condition $y(0) = 0$, and the term proportional to $y(x)$ just gives $(\lambda\pi)^2\mathcal{L}$. Thus, the solution is

$$\mathcal{L}_s[y] = \frac{a}{(\lambda\pi)^2 + s^2}$$

(2b) Inverse Laplace Transform [5 pts]: To invert your Laplace transform, you can use any symbolic software package or look-up tables. Wolfram Alpha is entirely sufficient. Determine the value of $a = y'(0)$ such that the amplitude of the solution is 1.

Solution: The inverse Laplace transform is

$$\mathcal{L}_s^{-1}\left[\frac{a}{(\lambda\pi)^2 + s^2}\right] = \frac{a}{\lambda\pi}\sin(\lambda\pi t).$$

For the amplitude of the solution to be 1,

$$a = \lambda\pi.$$

The solution is not entirely surprising.

(2c) Implementation of the integrator [30 pts]: The goal is to develop a program `sturmliouville.py` that integrates the BVP

$$y"(x) + (\lambda\pi)^2\rho(x)y(x) = 0, \tag{6}$$

where $\rho(x)$ is a density function (compare to eq. 4). The boundary conditions are $y(0) = 0$ and $y(1) = 0$. Obviously, there are solutions only for certain values of λ, thus, we have a BVP eigenvalue problem. We will solve this BVP problem with the shooting method discussed in class.

What role does λ play in the shooting method? Technically, it is an eigenvalue of the BVP. But what does that mean in this specific case? Here's an example: the usual ballistics problem (throwing a ball, or "launching a projectile") is phrased in terms of: "Given a fixed launching velocity, what is the optimal angle under which to throw the ball so that it travels the largest distance?" In its simplest version, the answer is $\alpha = 45°$. If you want to have the ball end up at a shorter distance, you can either increase or decrease α. The key point is that α gives the initial slope of the trajectory. Now check eq. 4. If this were just an IVP, then we'd need the initial conditions $y(0)$ and $y'(0)$. The latter is, as in the ballistics problem, the slope of the "trajectory". Thus, to reach a desired x_{end} such that $y(x_{end}) = 0$, we can change $y'(0)$. This is what the shooting method is about: systematically finding the correct value for $y'(0)$.

Back to λ. As you already saw in task (2b), λ appears linearly in the argument of $y(x)$, therefore, $y'(x) \propto \lambda$. If you modify λ, you change the slope of $y(x)$. Thus, instead of finding the correct value $y'(0)$, we will search for the correct value of λ, such that the boundary condition $y(1) = 0$ is fulfilled. In other words, we are looking for the root λ_i of the function $y(x = 1, \lambda_i) = 0$. Thus, we will need a rootfinder.

How do we approach this? Let's start from top down. We are looking for a function $y(x)$ that solves eq. 6. Obviously, $y(x)$ is the solution to an ODE. So, we'll need a prescription to calculate the RHS (steps [i] and [ii]), and an integrator (step [iii]). We will also need a rootfinder (step [v]). I suggest bisection, but if you're adventurous, feel free to implement a full-fledged Newton method.

To facilitate grading, please write all the functions into `sturmliouville.py`, which already contains the function definitions.

(i) The ODEs: Write down the three ODEs governing the system. "Three?" you may ask. Check out Numerical Recipes, chapter 17.0, discussing the formulation of an eigenvalue problem. Implement your three ODEs in the function `dydx_string(x,y)`. The function $\rho(x)$ will be implemented in step (ii).

Solution:

$$\begin{aligned}\dot{y}_0 &= y_1 \\ \dot{y}_1 &= -(\lambda\pi)^2\rho(x)y_0 \\ \dot{y}_2 &= 0\end{aligned}$$

(ii) The density function: Implement the density function $\rho(x)$ such that it can return the following choices:

$$\begin{aligned}\rho(x) &= 1 \\ \rho(x) &= 1 + \exp(10x).\end{aligned} \tag{7}$$

(iii) The integrator: Implement the integrator (driver `ode_ivp` and stepper `rk4`). You can take the code from the homework solutions.

(iv) The "trial shot" function: This is an optional step, but it simplifies the coding. The shot function takes a guess at the eigenvalue λ as argument, sets the boundary conditions $y, y'(\lambda), \lambda$ at $x = 0$, does one trial integration by calling the integrator, and compares the result (in our case, the value of $y(1)$) to the desired value. It should return the difference (which here is just $y(1)$). The "trial shot" function is called `bvp_shoot` in `sturmliouville.py`.

(v) The root finder: Implement the rootfinder in `bvp_root`. This can be a simple bisection. The "function evaluation" consists of a call to `bvp_shoot`. I recommend to "bracket" the root before trying to find it. The easiest way to do this is to take the initial guess λ_0 provided to `bvp_root` as an argument, calculate $f_0 =$`bvp_shoot`(λ_0), and then calculate a series of $f_i =$`bvp_shoot`$(\lambda_0(1.1)^i)$. For some i, $f_0 f_i < 0$, and the root will be bracketed. `bvp_root` should return the solution of the eigenvalue problem in terms of x, $\mathbf{y}(x)$.

(vi) The main program: The main program should call `bvp_root` returning x and $\mathbf{y}(x)$, and then plot $y(x)$ and $\lambda(x)$. The latter should be constant.

Solution: See code solutions. For each step (i)–(vi), [5 pts].

(2d) Tests [10 pts]: Now comes the fun part: Does everything work as intended? First, think of an easy test case. I'd suggest setting $\rho(x) = 1$, and the initial guess $\lambda_0 = 0.5$. Your bracketing step in the rootfinder should push $\lambda_i > 1$. If you're convinced your code found the correct result, try $\lambda_0 = 1.5$. Stepping through λ will generate the spectrum of eigenvalues. Finally, try $\rho(x) = 1 + \exp(10x)$.

Optional [+5 pts] You can try finding the "closed form" solution via Laplace transforms with the help of e.g. Wolfram Alpha, Mathematica, Maple or anything similar. It should contain Bessel functions and the Gamma function – clearly, a seemingly simple modification of the wave equation leads to quite – let's say – non-trivial results.