Homework 2

AMATH 481/581

Problem 1

Consider the eigenvalue problem

$$\psi'' + V(x)\psi + \lambda\psi = 0 \text{ with } \psi(-1) = 0 = \psi(1) \tag{1}$$

We found the eigenvalues λ and eigenfunctions ψ for this problem in class using a constant potential function $V(x) = C$. Now suppose that

$$V(x) = -100\left(\sin(2x) + 1\right)$$

For this problem, you will use the shooting method to calculate the first three eigenvalues and their corresponding eigenfunctions. You should use the same approach as we did in class. In particular, when you need to use a root finding method, use the bisection method with a tolerance of 10^{-8}. You should use the following code:

Python:

```
def bisection(f, a, b, tol):
    x = (a + b) / 2
    while np.abs(b - a) >= tol:
        if np.sign(f(x)) == np.sign(f(a)):
            a = x
        else:
            b = x
        x = (a + b) / 2
    return x
```

MATLAB:

```
function x = bisection(f, a, b, tol)
    x = (a + b) / 2;
    while abs(b - a) >= tol
        if sign(f(x)) == sign(f(a))
            a = x;
```

```
        else
            b = x;
        end
        x = (a + b) / 2;
    end
end
```

When implementing the shooting method, you will have to guess an initial velocity. You should guess $\psi'(-L) = 1$. You will also need to solve an initial value problem. You should use `scipy.integrate.solve_ivp` in python or `ode45` in MATLAB with all of the default options.

(a) The smallest eigenvalue is between 23 and 24. That is, $23 < \lambda_1 < 24$. Calculate λ_1 and store it in a variable named `A1`.

(b) Calculate the eigenfunction $\psi_1(x)$ corresponding to the smallest eigenvalue. That is, solve the BVP in equation (1) using $\lambda = \lambda_1$. You should use the builtin solver `scipy.integrate.solve_ivp` or `ode45` with the default options, but force it to evaluate the solution at $x = -1$, $x = 0$, and $x = 1$. That is, use

Python:

```
scipy.integrate.solve_ivp(f, tspan, init_condition,
                          t_eval=np.array([-1, 0, 1]))
```

MATLAB:

```
ode45(f, [-1, 0, 1], init_condition)
```

Store the value of $\psi(0)$ in a variable named `A2`. (**Note**: This is just to make it easy for python and MATLAB code to produce comparable output. You can and should find ψ at many x values on the interval $[-1, 1]$ and plot the eigenfunction, but you don't have to turn that work in to gradescope.)

(c) Find the second smallest eigenvalue λ_2. Store it in a variable named `A3`.

(d) Repeat part (b) using $\lambda = \lambda_2$. (That is, find $\psi(0)$ where ψ is the eigenfunction corresponding to λ_2.) Store your answer in a variable named `A4`.

(e) Find the third smallest eigenvalue λ_3. Store it in a variable named `A5`.

(f) Repeat part (b) using $\lambda = \lambda_3$. (That is, find $\psi(0)$ where ψ is the eigenfunction corresponding to λ_3.) Store your answer in a variable named `A6`.

Problem 2 - Relevant to the written report

The *trapezoidal method* is a method for solving initial value problems obtained by averaging together the forward and backward Euler methods at each time step. That is, if we are solving the IVP

$$\mathbf{x}' = f(t, \mathbf{x}) \text{ with } \mathbf{x}(0) = \mathbf{x}_0 \tag{2}$$

then our approximations follow the formula

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{\Delta t}{2} \left(f(t_k, \mathbf{x}_k) + f(t_{k+1}, \mathbf{x}_{k+1}) \right) \tag{3}$$

Notice that this is a one step, implicit method.

The *midpoint method* (also known as the *leapfrog method*) is another method for solving initial value problems. It can be obtained by using a second order central difference scheme to approximate $x'(t)$. The approximations follow the formula

$$\mathbf{x}_{k+1} = \mathbf{x}_{k-1} + 2\Delta t f(t_k, \mathbf{x}_k) \tag{4}$$

Notice that this is a two step, explicit method.

(a) Consider the IVP

$$x'' - x = 0 \text{ with } x(0) = 1 \text{ and } x'(0) = 0 \tag{5}$$

The true solution to this IVP is

$$x(t) = \frac{1}{2} \left(e^t + e^{-t} \right) \tag{6}$$

Solve this IVP from $t = 0$ up to $t = 1$ using the trapezoidal method with $\Delta t = 0.1$. (**Note**: Since the trapezoidal method is implicit, you will have to solve a system of equations in order to find an explicit formula for $\mathbf{x}_{k+1}$. This system should be relatively easy to solve by hand.) Store your final approximation for x (i.e., the approximation of x at time $t = 1$) in a variable named `A7` and store the global error (i.e., $|x_N - x(t_N)|$) in a variable named `A8`.

(b) Repeat part (a) using $\Delta t = 0.01$. Store the final approximation for x (i.e., the approximation of x at time $t = 1$) in a variable named `A9` and store the global error in a variable named `A10`.

(c) Now consider the IVP

$$x'' + x = 0 \text{ with } x(0) = 1 \text{ and } x'(0) = 0 \tag{7}$$

The true solution to this IVP is

$$x(t) = \cos(t) \tag{8}$$

Solve this IVP from $t = 0$ up to $t = 1$ using the midpoint method with $\Delta t = 0.1$. (**Note**: Since the midpoint method is a two step method, it cannot be used to find $\mathbf{x}_1$. You should use forward Euler to find this first point.) Store your final approximation for x (i.e., the approximation of x at time $t = 1$) in a variable named `A11` and store the global error in a variable named `A12`.

(d) Repeat part (c) using $\Delta t = 0.01$. Store the final approximation for x (i.e., the approximation of x at time $t = 1$) in a variable named `A13` and store the global error in a variable named `A14`.

For the written report: Use a numerical method (such as the one from the previous written report) to determine the order of accuracy for both the trapezoidal and midpoint methods. Include any relevant data and/or plots to support your conclusion.

Derive the stability region for the trapezoidal method and plot it. (That is, solve the test problem $x' = \lambda x$ with the trapezoidal method, where $\lambda \in \mathbb{C}$ is a constant, and find all values of $\lambda \Delta t$ such that the approximation goes to zero.)

Was the trapezoidal method stable for the problem in part (a) and/or (b)? Explain how you could have predicted that result using the stability region.

Problem 3

The Chebyshev equation is a second order linear equation of the form

$$(1 - x^2)y'' - xy' + \alpha^2 y = 0 \tag{9}$$

This ODE has polynomial solutions when α is a positive integer (as long as the initial/boundary conditions are chosen carefully), called Chebyshev polynomials. In this problem, you will find some of these polynomials by using the direct method for BVPs discussed in class. (That is, you will discretize the ODE by replacing y'' and y' with the corresponding second order central difference schemes and then solve the resulting system of equations.)

(a) Solve the Chebyshev equation with $\alpha = 1$, $y(-0.5) = -0.5$ and $y(0.5) = 0.5$. Use $\Delta x = 0.1$. (That is, use 11 total points, including the two boundary points.) The true solution is $y = x$. Store the approximation of y at $x = 0$ in a variable named `A15` and store the maximum error

$$\max_{k=0,\ldots,10} |y_k - y(x_k)| \tag{10}$$

in a variable named `A16`.

(b) Solve the Chebyshev equation with $\alpha = 2$, $y(-0.5) = 0.5$ and $y(0.5) = 0.5$. Use $\Delta x = 0.1$. (That is, use 11 total points, including the two boundary points.) The true solution is $y = 1 - 2x^2$. Store the approximation of y at $x = 0$ in a variable named `A17` and store the maximum error

$$\max_{k=0,\ldots,10} |y_k - y(x_k)| \tag{11}$$

in a variable named `A18`.

(c) Solve the Chebyshev equation with $\alpha = 3$, $y(-0.5) = -1/3$ and $y(0.5) = 1/3$. Use $\Delta x = 0.1$. (That is, use 11 total points, including the two boundary points.) The true solution is $y = x - 4x^3/3$. Store the approximation of y at $x = 0$ in a variable named `A19` and store the maximum error

$$\max_{k=0,\ldots,10} |y_k - y(x_k)| \tag{12}$$

in a variable named `A20`.