The Second-Order Boundary Eigenvalue Problem

Elijan Jakob Mastnak
Student ID: 28181157

December 2020

Contents

Assignment

1. Find the first few eigenfunctions and eigenvalues of an infinite potential well using the finite difference and shooting methods (and optionally other methods of your choice). Repeat the procedure for a finite potential well.

2. Investigate the contributions to error in the finite difference model. Error when discretizing the second derivative with a finite difference; error because of the discretization of the x interval (finite difference matrix dimension). Use the analytic solutions for reference.

---

1 Theory

To jump right to the solution, see Section 2.

A second-order ordinary boundary value problem involves solving a differential equation of the form

$$y'' = f(x, y, y')$$

on the interval $x \in [a, b]$ such that the solution satisfies boundary conditions at the endpoints a and b. This boundary conditions can come in a few forms, including:

- Dirichlet boundary conditions: $y(a) = \alpha$ and $y(b) = \beta$
- Von Neumann boundary conditions: $y'(a) = \alpha$ and $y'(b) = \beta$
- Robin boundary conditions: $A_1 y(a) + A_2 y'(a) = \alpha$ and $B_1 y(b) + B_2 y'(b) = \beta$

1.1 Finite Difference Method

To use the finite difference method to solve a second-order boundary problem on the interval $[a, b]$ with Dirichlet boundary conditions, we partition the interval into $n+1$ points $a = x_0 < x_1 < \ldots < x_n = b$, typically separated by a uniform step size h. We then rewrite the second derivative y'' with the finite difference approximation

$$y'' \approx \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2}$$

Applied to the stationary Schrödinger equation for a free particle, this reads

$$\psi_{i-1} - (2 - \lambda)\psi_i + \psi_{i+1} = 0$$

where $\lambda = Eh^2 = k^2h^2$. The resulting set of linear equations generally produces a tridiagonal system of the form

$$\mathbf{H}\boldsymbol{\psi} = \lambda\boldsymbol{\psi}$$

which we can solve with diagonalization methods to find the eigenvalues λ and corresponding eigenvectors $\boldsymbol{\psi} = (\psi_1, \psi_2, \ldots)^T$.

1.2 Shooting Method

With the shooting method, we treat the boundary value problem as an initial value problem. With a chosen ODE solver, we integrate the differential equation

$$y'' = f(x, y, y')$$

from the starting point $x = a$ to the endpoint $x = b$ with a chosen set of initial conditions on ψ and ψ'. We then check if the solution matches the desired boundary conditions and vary the initial conditions until the initial value problem solution satisfies the boundary value conditions to a desired level of accuracy.

For an eigenvalue problem involving the free parameter λ, we instead fix the initial conditions and, for a choice of the energy λ integrate the differential equation from $x = a$ to the boundary $x = b$ and check if the boundary conditions hold. In the likely even they do not, we vary the value of λ and repeat the procedure (e.g. with a root-finding algorithm) until the boundary condition holds to a desired accuracy.

For Schrödinger-type problems with a symmetric potential on a symmetric interval $x \in \left[-\frac{a}{2}, \frac{a}{2}\right]$, it is also possible to start the shooting method at the midpoint $x = 0$ and use the orthogonal boundary conditions $\psi(0) = 1, \psi'(0) = 0$ or $\psi(0) = 0, \psi'(0) = 1$, which yield symmetric and asymmetric solutions, respectively.

2 Preliminary Steps

This section presents some nonessential but useful steps that facilitate solving the remainder of the problem.

2.1 Transformation to Dimensionless Units

For convenience, I transformed to natural units better suited to numerical computation. I set $a = 1$ and $\frac{m}{\hbar^2} = 1$, so energy is measured in units of $\frac{\hbar^2}{ma^2}$. In this set of units, the Schrödinger equation reads

$$\psi'' = -2(E - V)\psi$$

2.2 Infinite Potential Well

The general solution for the wavefunctions ψ_n and energy eigenvalues E_n of a particle in an infinite potential well (IPW) of width a centered at $x = 0$ is

$$\psi_n(x) = \sqrt{\frac{2}{a}} \sin\left[\frac{n\pi}{a}\left(x + \frac{a}{2}\right)\right] \qquad \text{and} \qquad E_n = \frac{n^2\pi^2\hbar^2}{2ma^2}$$

where a is the well's width. Alternatively, we can write the solutions as

$$\psi_n(x) = \begin{cases} \sqrt{\frac{2}{L}} \sin\left[\frac{n\pi x}{a}\right] & n \text{ even} \\ \sqrt{\frac{2}{L}} \cos\left[\frac{n\pi x}{a}\right] & n \text{ odd} \end{cases}$$

In natural units, the Schrödinger equation and energy eigenvalues read

$$\psi_n(x) = \sqrt{2} \sin\left[n\pi\left(x + \tfrac{1}{2}\right)\right] \qquad \text{and} \qquad E_n = \frac{n^2\pi^2}{2} \tag{1}$$

These are implemented in the following Python functions

```python
import numpy as np
def ipw_energies(n, l=1.0):
""" Returns the energy of the infinite potential well's nth state """
    return (np.pi*n)**2/2

def ipw_wavefunction(x, n, l=1.0):
""" Returns infinite potential well's nth wavefunction """
    return np.sqrt(2)*np.sin(n*np.pi * (x + l/2))
```

2.3 Finite Potential Well

To solve the finite potential well (FPW), we split the x axis into three regions: $(-\infty, -\frac{a}{2})$, $(-\frac{a}{2}, \frac{a}{2})$ and $(\frac{a}{2}, \infty)$, which I'll call regions 1, 2 and 3, respectively. Regions 1 and 3 are solved with an exponential ansatz which must obey $\psi(x) \to 0$ as $x \to \pm\infty$. Region 2 is solved by an oscillatory ansatz, which can be written either in terms of complex exponents or sines and cosines.

$$\psi(x) = \begin{cases} Ae^{\kappa x} & -\infty < x < -\frac{a}{2} \\ Be^{-ikx} + Ce^{ikx} & -\frac{a}{2} < \frac{a}{2} \\ De^{-\kappa x} & \frac{a}{2} < x < \infty \end{cases}$$

where the constants k and κ, in both SI units and natural units, are

$$k = \sqrt{\frac{2mE}{\hbar^2}} \to \sqrt{2E} \qquad \text{and} \qquad \kappa = \sqrt{\frac{2m(V_0 - E)}{\hbar^2}} \to \sqrt{2(V_0 - E)}$$

The wavefunction must be continuous and continuously differentiable at the well boundaries, leading to the requirements

$$\begin{aligned} \psi_1\left(-\tfrac{L}{2}\right) &= \psi_2\left(-\tfrac{L}{2}\right) & \psi_2\left(\tfrac{L}{2}\right) &= \psi_3\left(\tfrac{L}{2}\right) \\ \psi_1'\left(-\tfrac{L}{2}\right) &= \psi_2'\left(-\tfrac{L}{2}\right) & \psi_2'\left(\tfrac{L}{2}\right) &= \psi_3'\left(\tfrac{L}{2}\right) \end{aligned}$$

These boundary conditions generate a system of equations for the coefficients A, B, C and D. Because the potential $V(x)$ is symmetric about the origin, the finite potential well has two classes of solutions: symmetric solutions with $C = 0$ and $A = D$; and antisymmetric solutions with $B = 0$ and $A = -D$.
The energies of the symmetric and antisymmetric states, respectively, are found from the $\kappa(E)$ and $k(E)$ solving the transcendental equation

$$\kappa = k \tan\left(\frac{ka}{2}\right) \qquad \text{and} \qquad \kappa = -k \cot\left(\frac{ka}{2}\right)$$

A Python program for the FPW's energies with a root-finding algorithm might read

```python
from scipy.optimize import root_scalar  # root-finding algorithm
def get_fpw_analytic_energies(V0=50.0, l=1.0, n_points=100):
    """Finds bound state energies of a FPW with height V0 and width l"""
    E_grid = np.linspace(0.0, V0, n_points)  # energy grid spanning 0 to V0
```

```
    E_roots = []  # array to hold energy roots
    for n in [1, 2]:  # test both even/odd or symmetric/asymmetric states
        for i in range(n_points - 1):
            E_root = root_scalar(energy_root_fun_fpw, args=(V0, l, n),
            ↪  x0=E_grid[i], x1=E_grid[i+1])  # returns root_scalar object
            if E_root.converged:  # uses root_scalar's converged function
                if E_root not in E_roots:  # reject already-found energies
                    E_roots.append(E_root)
    return np.sort(energies)  # sort and return energy solutions

def fpw_energy_roots(E, V0=50, l=1, n=1):
    """ Auxiliary function for solving the FPW's transcendental equation with
    ↪  root-finding algorithms """
    k1, k2 = np.sqrt(2*E), np.sqrt(2*(V0 - E))
    if n % 2 == 1:  # for odd n (symmetric solutions)
        return k2 - k1*np.tan(k1*l/2)
    else:  # for even n (asymmetric solutions)
        return k2 + k1/np.tan(k1*l/2)
```

2.4 Normalization

A properly behaving wavefunction should be normalized, and normalization is practically essential when comparing a numerical solution to the analytic solution. Nominally, normalizing a wavefunction ψ involves finding the constant A such that

$$A^2 \int_{-\infty}^{\infty} |\psi|^2 \, \mathrm{d}x = 1$$

In discrete position space, assuming the x range is large enough to approximate the interval $(-\infty, \infty)$ a discrete normalization implementation might read

```
""" Normalizes the wavefunction psi defined on the position grid x """
def normalize(x, psi):
    probability = psi**2  % probability density
    norm = 0  # initialize normalization constant
    for i in range(len(x)-1):
        dx = x[i+1] - x[i]
        norm += 0.5*dx*(probability[i] + probability[i+1])  # integration
    return psi / np.sqrt(norm)  # normalize the wavefunction
```

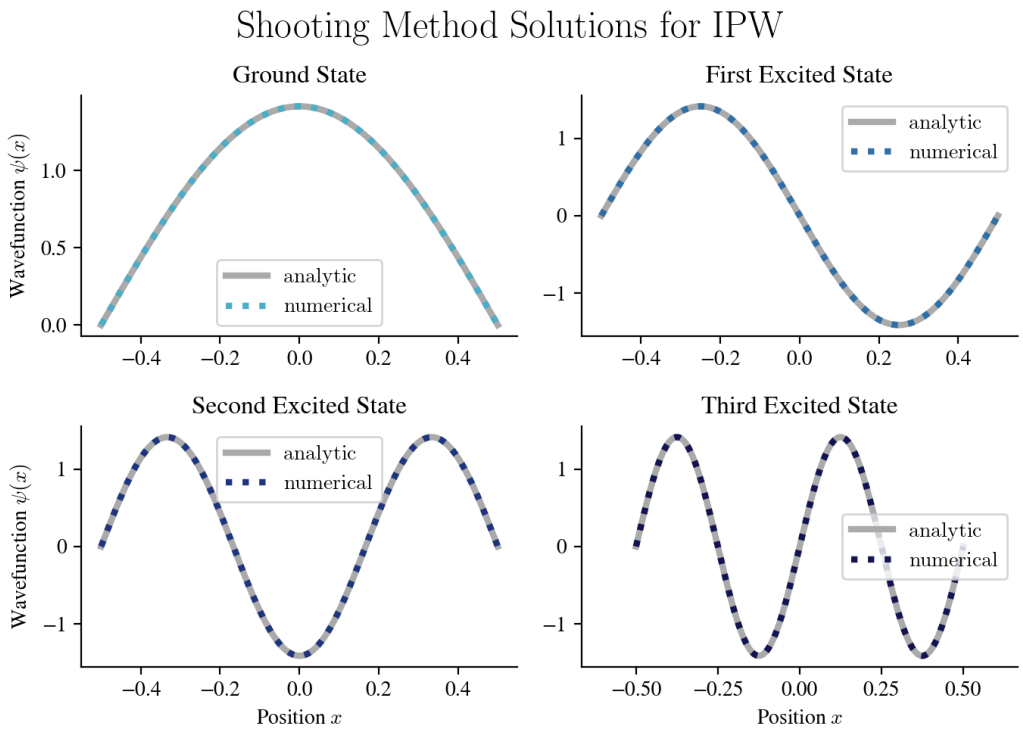

Figure 1: Numerical solutions for the IPW's first few eigenfunctions, found with the shooting method, compared to the analytic solutions. Both sets of functions are normalized as described in Subsection 2.4.

3 Infinite Potential Well

The infinite potential well involves solving the Schrödinger equation $\psi'' = -2E\psi$ on the interval $x \in \left(-\frac{a}{2}, \frac{a}{2}\right)$ with the boundary conditions $\psi\left(-\frac{a}{2}\right) = \psi\left(\frac{a}{2}\right) = 0$. Figure 1 shows the IPW's first few eigenfunctions, found with the shooting method, compared to the analytic solution.

3.1 Shooting Method

I implemented the shooting method for the IPW as follows:

1. Create a finely-spaced grid of energies from 0 to some arbitrarily chosen energy $E_{\max}$, corresponding to the highest desired energy eigenvalue, and a grid of position values $\{x_i\}_0^n$ spanning the well from $\frac{-a}{2}$ to $\frac{a}{2}$ on which to integrate the shooting method's initial value problem.[1]

2. For each energy, using the initial condition $\psi_0 = 0, \psi_0' = 1$, integrate from the left endpoint x_0 to the right endpoint x_n. Store each endpoint value ψ_n in an

[1] Alternatively, integrate from $x = 0$ to $x = \frac{a}{2}$ using the orthogonal boundary conditions $\psi_0 = 0, \psi_0' = 1$ and $\psi_0 = 1, \psi_0' = 0$ to find the symmetric and asymmetric solutions separately. When testing, I found this method gave slightly more accurate energy eigenvalues, but the difference was small enough that I stayed with the more general scheme $\frac{-a}{2}$ to $\frac{a}{2}$ for the rest of the report.

array, to get ψ_n as a function of the energy parameter. Use the $\psi_n(E)$ data to estimate the roots E_n of the function

$$f(E) = \psi_n(E) - \psi_{\text{BC}} = \psi_n$$

where $\psi_{\text{BC}} \equiv 0$ is the boundary condition at the endpoint x_n. Figure 5 (a few pages ahead) demonstrates this process for the finite potential well.

3. Refine the guesses for E_n using a root-finding algorithm—I used Newton's method—applied to the function

$$f(E) = \psi_n(E) - \psi_{\text{BC}} = \psi_n(E)$$

The thus-found roots E_n of the function $f(E)$ are the energy eigenvalues.

4. Find the eigenfunction ψ_n corresponding to the each eigenvalue E_n by solving the Schrödinger equation as an initial value problem using $E = E_n$.

If desired, see the attached `bvp.py` for the exact Python implementation, which I felt was too long to reasonably include in the report itself.

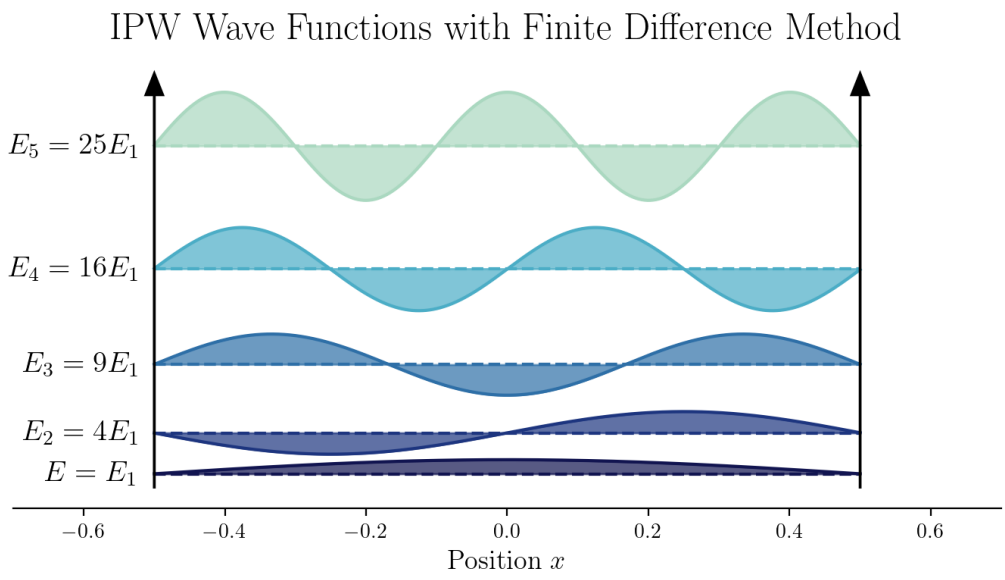

Figure 2: The IPW's first few eigenvalues and eigenfunctions found with the finite difference method. Note the theoretically expected energy growth $E_n = n^2 E_1$.

3.2 Finite Differences

The infinite potential well's simple boundary conditions make a finite difference solution straightforward. I divided the interval $\left[\frac{a}{2}, \frac{a}{2}\right]$ into $n+1$ evenly spaced points $x_0, x_1, \ldots, x_n$, on which the finite difference approximation of the Schrödinger equation reads

$$\frac{\psi_{i-1} - 2\psi_i + \psi_{i+1}}{h^2} = -2E\psi_i$$

The boundary conditions $\psi_0 \equiv 0$ and $\psi_n \equiv 0$ produce

$$0 - 2\psi_1 + \psi_2 = -2h^2 E\psi_1 \qquad \text{and} \qquad \psi_{n-2} - 2\psi_{n-1} + 0 = -2h^2 E\psi_{n-1}$$

Combining these boundary conditions with the general expression finite difference approximation $\psi_{i-1} - 2\psi_i + \psi_{i+1} = -2h^2 E\psi_i$, reduces the IPW to the eigenvalue problem

$$\mathbf{H}\boldsymbol{\psi} = E\boldsymbol{\psi}$$

where $\mathbf{H}$ is the $(n-1) \times (n-1)$ tridiagonal matrix

$$\mathbf{H} = -\frac{1}{2h^2}\begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & & \\ 0 & \dots & 0 & 1 & -2 & 1 \\ 0 & 0 & \dots & 0 & 1 & -2 \end{bmatrix}$$

I solved the eigenvalue problem $\mathbf{H}\boldsymbol{\psi} = E\boldsymbol{\psi}$ with the following Python code

```python
from scipy.linalg import eigh_tridiagonal
def ipw_fd(n_points=1000, l=1.0):
""" Finds infinite potential well energy eigenvalues and eigenvectors """
    x = np.linspace(-l/2, l/2, n_points)  # divide well into n_points
    h = x[1]-x[0]  # step size
    d = 1.0*np.ones(n_points-0)/h**2  # main diagonal of form 1/h^2
    e = -0.5*np.ones(n_points-1)/h**2  # off-diagonals of form -0.5/h^2
    return eigh_tridiagonal(d, e)  # returns eigenvalues and eigenvectors
```

Figure 2 shows the IPW's first few eigenvalues and eigenfunctions, found with the finite difference method.

3.3 Accuracy

I investigated the accuracy of the shooting and finite difference methods in the following ways:

1. Analyzing accuracy of the finite difference approximation as a function of the number of points N used to partition the interval $\left[-\frac{a}{2}, \frac{a}{2}\right]$, which corresponds to the dimension of the Hamiltonian matrix $\mathbf{H}$. As expected, accuracy improved with N, as shown in Figure 3. Note the decrease in error at the interval's boundaries (characteristic for a boundary value problem) and the eigenfunction nodes, which I would not expect for an initial value problem.

2. Determining accuracy of the shooting method, shown in Figure 4. The absolute error in ψ is roughly of the order 10^{-10}—much better than with the finite difference method. Note also the smaller error at the boundaries and wavefunction nodes and the growth in error in the higher excited states.

3. Analyzing error in the energy eigenvalues found by both methods, shown in Table 2. The shooting method outperformed the finite difference method. As seems reasonable, the error increase for higher excited states for both methods.

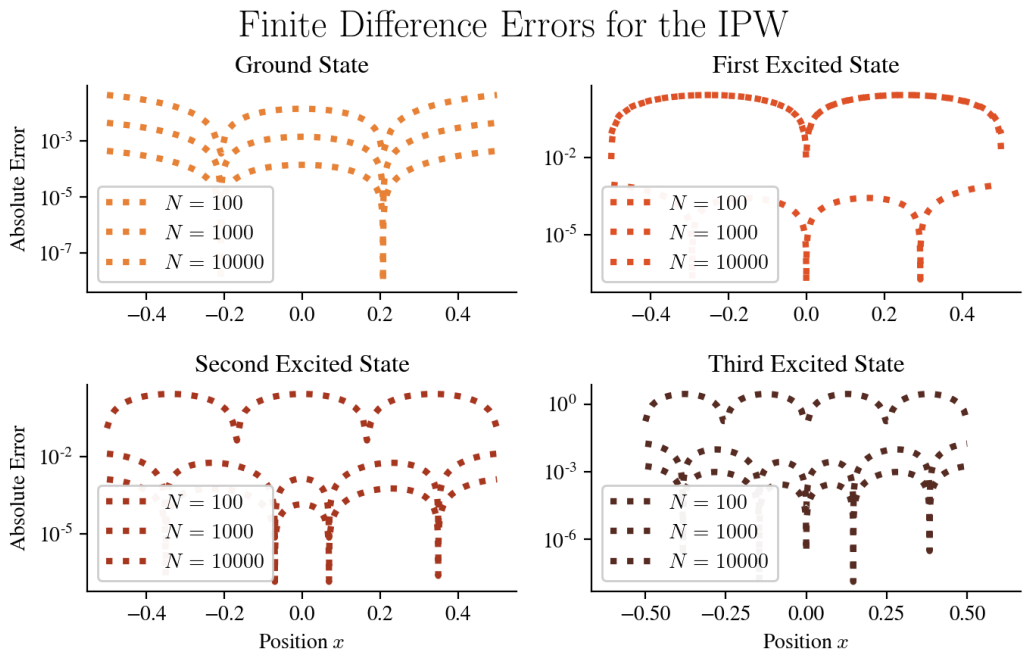

Figure 3: The absolute error of the finite difference solution to the IPW's wavefunctions improves with increasing Hamiltonian matrix dimension N. Tested with respect to the analytic solution in Equation 1. Note the logarithmic scale.

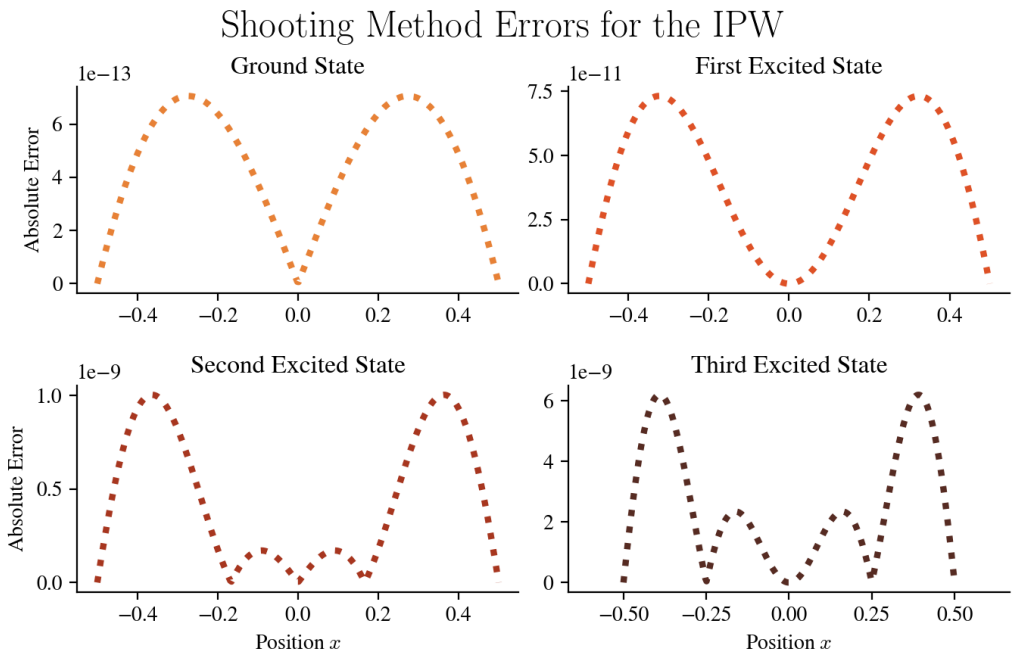

Figure 4: Shooting method errors in the first few wavefunctions ψ_n with respect to the analytic solution. Note the logarithmic scale and considerably smaller error than with the finite difference method.

F.D. $N = 100$	F.D. $N = 1000$	F.D. $N = 10000$	Shooting
1.938852e-01	1.970420e-02	1.973201e-03	2.005447e-07
7.801268e-01	7.886380e-02	7.894802e-03	1.978213e-07
1.772479e+00	1.776248e-01	1.776480e-02	1.195098e-06
3.193854e+00	3.162292e-01	3.158721e-02	3.791285e-06

Table 1: Error in the IPW's first few energy eigenvalues for various solution methods.

Analytic	F.D. $N = 100$	F.D. $N = 1000$	F.D. $N = 10000$	Shooting
4.934802	4.740917	4.915098	4.932829	4.934802
19.739209	18.959082	19.660345	19.731314	19.739209
44.413220	42.640741	44.235595	44.395455	44.413221
78.956835	75.762981	78.640606	78.925248	78.956839

Table 2: The IPW's first few energy eigenvalues found with various methods.

4 Finite Potential Well

4.1 Shooting Method

The general logic of my shooting method implementation for the FPW is similar to the implementation for the IPW in Subsection 3.1, with the following differences:

- I test energies only up to the well's depth V_0, which correspond to bound states.
- Inspired by [1], I create a position grid extending far beyond the well boundaries (instead of from $-\frac{a}{2}$ to $\frac{a}{2}$ as for the IPW), to where the exponentially decaying wavefunction can be reasonably approximated as zero.
- Of course, I use a modified Schrödinger equation, including the potential energy term, when integrating the initial value problem.

Figure 5 shows the process of finding the energy roots E_n satisfying the boundary condition $\psi_n \to 0$, while Figure 6 shows the bound eigenfunctions and eigenvalues of a finite potential well of depth V_0. As expected, the wavefunction sinusoidally oscillates inside the well and decays exponentially outside the well, while the energy eigenvalues of a given state are lower than for the IPW.

4.2 Finite Differences

My finite difference attempt at solving the FPW strictly on the interval $x \in \left[-\frac{a}{2}, \frac{a}{2}\right]$ was unsuccessful and is relegated to Appendix A. As a work-around, like for the shooting method, I solved the FPW on the interval $x \in [-1.5, 1.5]$ where the wavefunction is nearly zero at the endpoints, and proceeded analogously to the finite difference approach for the IPW. The problem reduces to the eigenvalue equation

$$\mathbf{H}\boldsymbol{\psi} = E\boldsymbol{\psi}$$

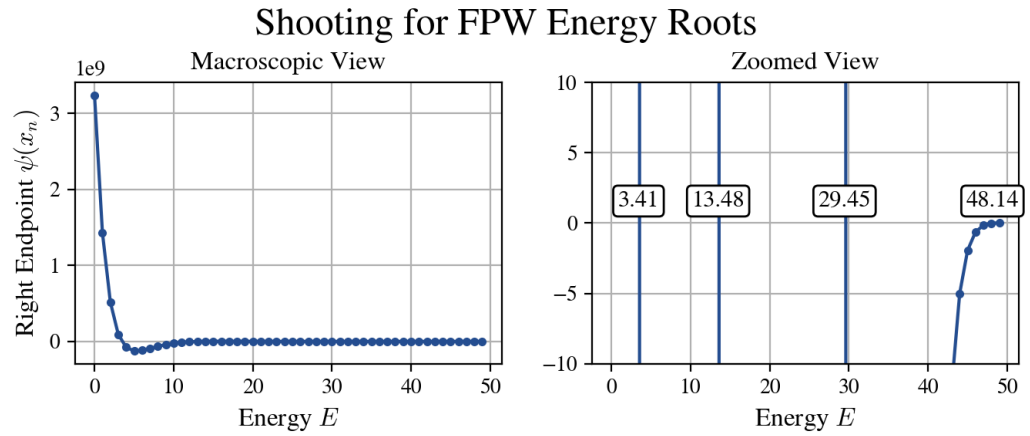

Figure 5: Estimating finite potential well energy eigenvalues with the shooting method.

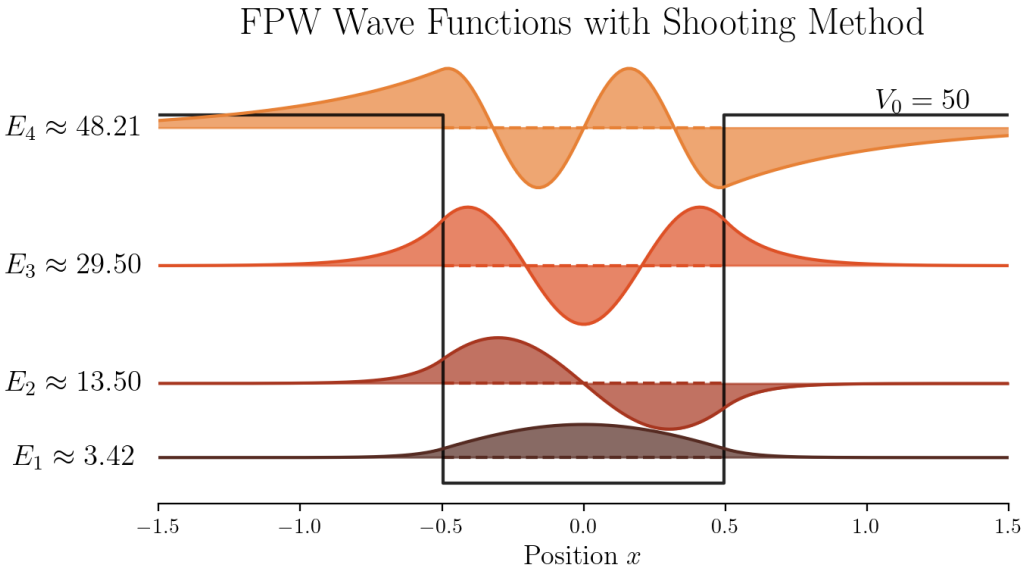

Figure 6: Bound states in a FPW of depth $V_0 = 50$ found with the shooting method.

where the tridiagonal matrix $\mathbf{H}$ now includes a potential energy term

$$\mathbf{H} = -\frac{1}{2h^2}\begin{bmatrix} -2-2h^2V_1 & 1 & 0 & 0 & \dots & 0 \\ 1 & -2-2h^2V_2 & 1 & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & & \\ 0 & \dots & 0 & 1 & -2-2h^2V_{n-2} & 1 \\ 0 & 0 & \dots & 0 & 1 & -2-2h^2V_{n-1} \end{bmatrix}$$

where $V_i = V(x_i)$. The Python implementation in shown in the code block below.

```python
def fpw_fd(n_points=1000, l=1.0, V0=50.0):
""" Finds finite potential well energy eigenvalues and eigenvectors """
    x = np.linspace(-1.5 * l, 1.5 * l, n_points)  # extends beyond well
    ↪ boundaries
    h = x[1]-x[0]  # time step
    d = 1.0 * np.ones(n_points) / h**2 + V_fpw(x, V0=V0)  # main diagonal
    e = -0.5 * np.ones(n_points - 1) / h**2  # off diagonal
    return eigh_tridiagonal(d, e)  # returns eigenvalues and eigenvectors
```

Figure 7 shows the bound states of a finite potential well of depth $V_0 = 100$ found with the above finite difference method.

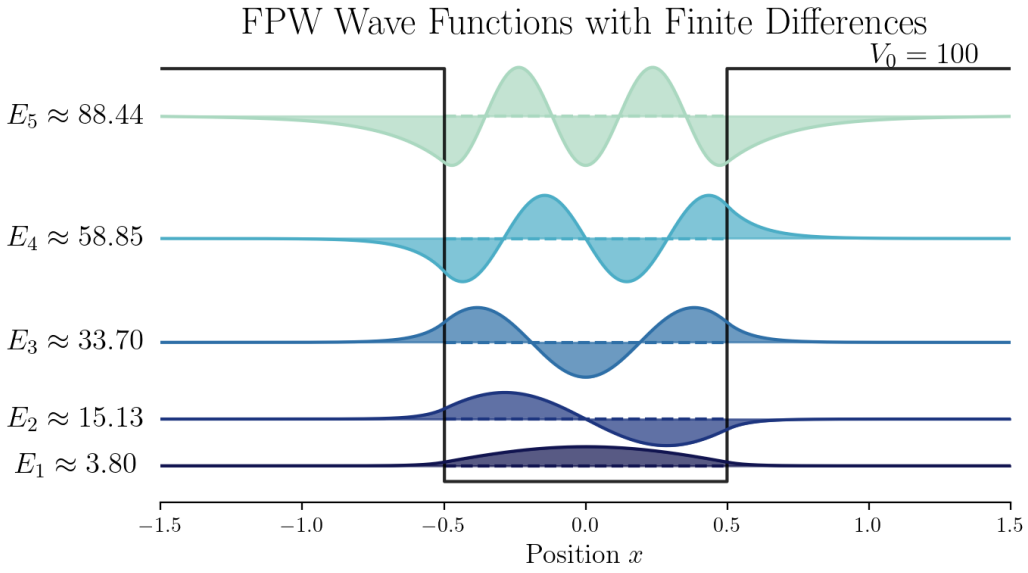

Figure 7: Bound states of a FPW of depth $V_0 = 100$ found with finite differences.

4.3 SciPy's Built-In `solve_bvp`

As an extra, I tested the built-in `solve_bvp` function from SciPy's `integrate` package. Figure 8 shows the results for a FPW of depth $V_0 = 50$—the wavefunctions are macroscopically analogous the those found with the shooting and finite difference methods, although the error in energy eigenvalues, at least for the low-energy states, is multiple orders of magnitude smaller (discussed more in the next subsection). However, `solve_bvp`'s improved accuracy came at a price—the function consistently took at least an order of magnitude longer to find a solution than the much-faster shooting and finite difference methods.

4.4 Accuracy

Table 3 shows the error, with respect to the analytic solution in Subsection 2.3, of the bound-state energy eigenvalues of a finite potential well of depth $V_0 = 50$ and width $a = 1$ found with the shooting method, `solve_bvp`, and the finite difference method for three values of N. Table 4 shows the corresponding energy values, with the analytic solution for reference. Error increases with energy for all methods, although this increase is fastest with `solve_bvp`.

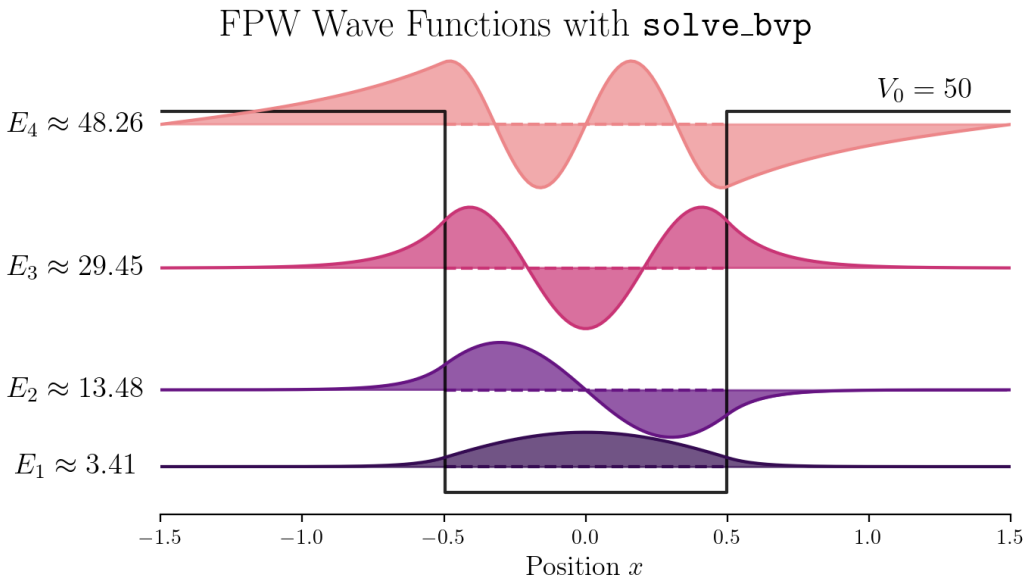

Figure 8: Bound states of a FPW of depth $V_0 = 50$ found with SciPy's `solve_bvp`.

F.D. $N = 100$	F.D. $N = 1000$	F.D. $N = 10000$	Shooting	`solve_bvp`
1.630150e-01	1.690801e-02	1.696010e-03	5.781010e-03	6.601000e-05
6.164423e-01	6.517526e-02	6.549260e-03	2.255926e-02	3.492600e-04
1.226205e+00	1.336164e-01	1.348941e-02	4.732041e-02	2.479410e-03
1.323991e+00	2.302478e-01	9.866979e-02	6.386779e-02	1.174988e-01

Table 3: Error with respect to the analytic solution in Subsection 2.3 of the bound-state energy eigenvalues of a finite potential well of depth $V_0 = 50$ and width $a = 1$ found with various methods. Note the increase in error for less-bound states.

Analytic	F.D. $N = 100$	F.D. $N = 1000$	F.D. $N = 10000$	Shooting	`solve_bvp`
3.413571	3.576586	3.430479	3.415267	3.419352	3.413637
13.475723	14.092165	13.540898	13.482272	13.498282	13.476072
29.452308	30.678513	29.585924	29.465797	29.499628	29.454787
48.143464	49.467455	48.373712	48.242134	48.207332	48.260963

Table 4: Bound-state energy eigenvalues of a finite potential well of depth $V_0 = 50$ and width $a = 1$ found with various methods.

A Attempt at a FPW Finite Difference FPW $[-\frac{a}{2}, \frac{a}{2}]$

Below is my thought process in my unsuccessful attempt at solving the FPW strictly on the interval $x \in \left[-\frac{a}{2}, \frac{a}{2}\right]$ (and not using an interval beyond the well boundaries as I did above).

Inside the well, with sinusoidal solutions, we have

$$\psi_{\text{in}}'' = -k^2 \psi_{\text{in}}, \qquad k = \sqrt{2E}$$

Outside the well, we wavefunction is exponential and obeys

$$\psi_{\text{in}}'' = -\kappa^2 \psi_{\text{in}}, \qquad \kappa = \sqrt{2(V_0 - E)}$$

Continuity and continuous differentiability require

$$\psi_{\text{out}}(x_0) = \psi_{\text{in}}(x_0) \qquad \text{and} \qquad \psi_{\text{out}}'(x_0) = \psi_{\text{in}}'(x_0)$$

Outside the well, with the exponential ansatz $\psi_{\text{out}} \propto e^{\kappa x}$, we have

$$\psi_{\text{out}}'(x_0) = \kappa \psi_{\text{out}}(x_0)$$

The continuous differentiability requirement $\psi_{\text{out}}'(x_0) = \psi_{\text{in}}'(x_0)$ leads to

$$\psi_{\text{in}}'(x_0) = \kappa \psi_{\text{out}}(x_0)$$

Combining this equality with the continuity requirement $\kappa\psi_{\text{in}}(x_0) = \kappa\psi_{\text{out}}(x_0)$ leads to the mixed (Robin) boundary condition

$$\psi_{\text{in}}'(x_0) - \kappa\psi_{\text{in}}(x_0) = 0 \implies \kappa\psi_{\text{in}}(x_0) - \psi_{\text{in}}'(x_0) = 0$$

An analogous derivation at the right endpoint $x = x_n$ with the ansatz $\psi_{\text{out}} \propto e^{-\kappa x}$ gives the boundary conditions

$$\kappa\psi_{\text{in}}(x_n) + \psi_{\text{in}}'(x_n) = 0$$

Using the second-order Taylor expansion

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \mathcal{O}(h^2)$$

we can re-write the first derivatives ψ_0' and ψ_n' as

$$\psi_0' \approx \frac{\psi_1 - \psi_0}{h} - \frac{h}{2}\psi_0'' \qquad \text{and} \qquad \psi_n' \approx \frac{\psi_n - \psi_{n-1}}{h} + \frac{h}{2}\psi_n''$$

Using the Schrödinger equation $\psi''(x) = -2E\psi(x)$ with $V(x) = 0$ inside the well, these conditions become

$$\psi_0' \approx \frac{\psi_1 - \psi_0}{h} + hE\psi_0 \qquad \text{and} \qquad \psi_n' \approx \frac{\psi_n - \psi_{n-1}}{h} - hE\psi_n$$

Substituting these expressions for ψ_0' and ψ_n' into the mixed boundary conditions $\kappa\psi_0 - \psi_0' = 0$ and $\kappa\psi_n + \psi_n' = 0$ gives

$$\kappa\psi_0 - \frac{\psi_1 - \psi_0}{h} - hE\psi_0 = 0 \qquad \text{and} \qquad \kappa\psi_n + \frac{\psi_n - \psi_{n-1}}{h} - hE\psi_n = 0$$

Or, collecting the coefficients in front of the ψ terms,

$$\left(\kappa + \frac{1}{h} - hE\right)\psi_0 - \frac{\psi_1}{h} = 0 \qquad \text{and} \qquad -\frac{\psi_{n-1}}{h} + \left(\kappa + \frac{1}{h} - hE\right)\psi_n = 0$$

The remaining terms for $i = 1, 2, \ldots, n-1$ read

$$\psi_{i-1} - 2\psi_i + \psi_{i+1} = -2h^2 E\psi_i \implies \psi_{i-1} - 2(1 - h^2 E)\psi_i + \psi_{i+1} = 0$$

Together, these form a system of $n+1$ equations for the $n+1$ unknowns ψ_0, ψ_1, …, ψ_n, ψ_{n+1} for a given value of E.

$$\begin{bmatrix} \left(\kappa + \frac{1}{h} - hE\right) & -\frac{1}{h} & 0 & 0 & \ldots & 0 \\ 1 & -2(1-h^2E) & 1 & 0 & \ldots & 0 \\ \vdots & & \ddots & & & \vdots \\ 0 & 0 & \ldots & 1 & -2(1-h^2E) & 1 \\ 0 & 0 & 0 & \ldots & -\frac{1}{h} & \left(\kappa + \frac{1}{h} - hE\right) \end{bmatrix} \boldsymbol{\psi} = \mathbf{0}$$

This matrix equation naturally produces a trivial solution for the wavefunction vector $\boldsymbol{\psi}$. Evidently, I need at least one non-zero element in the matrix's right-hand vector, which I assume would come from properly implemented boundary conditions. I tried a few modified configurations, which where more tinkering than anything of physical substance, but without great success. Among these was the below equation, roughly motivated by the scattering problem in [3]

$$\begin{bmatrix} \left(\kappa + \frac{1}{h} - hE\right) & -\frac{1}{h} & 0 & 0 & \ldots & 0 \\ 1 & -2(1-h^2E) & 1 & 0 & \ldots & 0 \\ \vdots & & \ddots & & & \vdots \\ 0 & 0 & \ldots & 1 & -2(1-h^2E) & 1 \\ 0 & 0 & 0 & \ldots & -\frac{1}{h} & \left(\kappa + \frac{1}{h} - hE\right) \end{bmatrix} \boldsymbol{\psi} = \begin{bmatrix} 2\kappa e^{-\frac{\kappa a}{2}} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This at least produced non-zero, oscillatory solutions for $\vec{\psi}$ where the number of nodes increased with increasing energy E_n, which is a step in the right direction. However, the results did not closely resemble the wavefunctions for a finite potential well, and I did not include them in the report.

References

[1] Schroeder, D. "Numerical Solutions of the TISE". *Notes on Quantum Mechanics*. January 2020. https://physics.weber.edu/schroeder/quantum/Numerical.pdf.

[2] Ertl, M. C. "Solving The Stationary One Dimensional Schrödinger Equation With The Shooting Method". Bachelor Thesis, TU Wien, Vienna. September 2016. https://www.iue.tuwien.ac.at/uploads/tx_sbdownloader/Bachelor-Arbeit_Marie_ERTL_09-2016.pdf.

[3] King, J; Dhakal, P. "Finite Difference Schemes and the Schrödinger Equation." 2 June 2014. http://www.dartmouth.edu/~pawan/final%20project.pdf.

[4] Becerril, R; et al. "Solving the time-dependent Schrödinger equation using finite difference methods". *Revista Mexicana de Fisica* **54**(2), 120-132. (2008). https://aip.scitation.org/doi/pdf/10.1063/1.4823060.