Solving the Time-Independent Schrodinger Equation Computationally

Anne Stratman and Ben Riordan
Computational Lab in Quantum Mechanics, Spring 2018, University of Notre Dame

Introduction: Discrete Basis

The time-independent Schrodinger equation, given by

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi(x)+V(x)\psi(x)=E\psi(x) \tag{1}$$

can be solved as a matrix diagonalization problem in a discrete basis or a harmonic oscillator basis. The solution in the discrete basis is found using inner products of distinct points ($\langle x_i|H|x_j\rangle$). To solve the problem in the discrete basis, one rewrites the second derivative in the Schrodinger equation in terms of the value of ψ at three points. The resulting equation can then be used to define diagonal and nondiagonal matrix elements. Each of these matrix elements is evaluated at a discrete point, and the resulting matrix is diagonalized. This gives the Hamiltonian for the potential - a set of eigenvalues and eigenvectors that correspond to the allowed energies and their wavefunctions. To begin this process, the second derivative can be rewritten as

$$f''=\frac{f(x+h)-2f(x)+f(x-h)}{h^2}+O(h^2) \tag{2}$$

where h is the step size. Next, we can define minimum and maximum values for the variable x, R_{min} and R_{max}, respectively. For a given number of steps N, the step size h can be defined by

$$h=\frac{R_{max}-R_{min}}{N} \tag{3}$$

Defining an arbitrary value of x as x=R_{min}+kh, k=1,2,...,N_{step}-1, the Schrodinger equation can be rewritten for x_k as

$$-\frac{\hbar^2}{2m}\frac{\psi(x_k+h)-2\psi(x_k)+\psi(x_k-h)}{h^2}+V(x)\psi(x_k) = E\psi(x_k) \tag{4}$$

or, more compactly,

$$-\frac{\hbar^2}{2m}\frac{\psi_{k+1}-2\psi_k+\psi_{k-1}}{h^2}+V_k\psi_k=E\psi_k \tag{5}$$

where ψ_k=$\psi(x_k)$ and $\psi_{k\pm1}$=$\psi(x_k\pm h)$. Next, we can define diagonal matrix elements:

$$d_k=\frac{\hbar^2}{2m}\frac{2}{h^2}+V_k \tag{6}$$

and non-diagonal matrix elements:

$$e_k=-\frac{\hbar^2}{2m}\frac{1}{h^2} \tag{7}$$

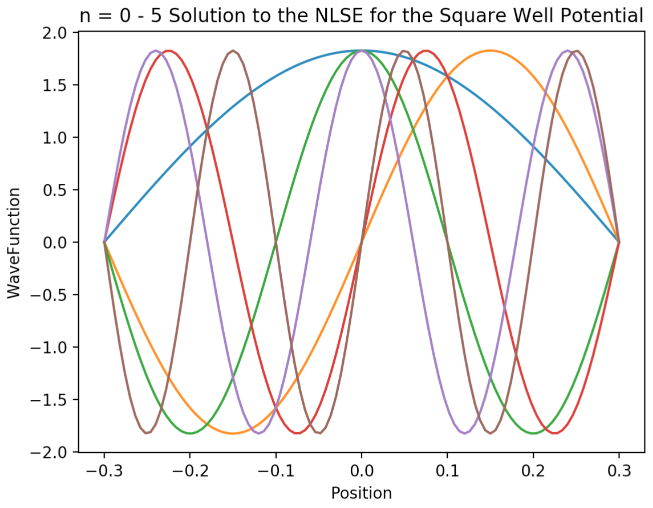

FIG. 1. Square well solved in the discrete basis, using 100 points

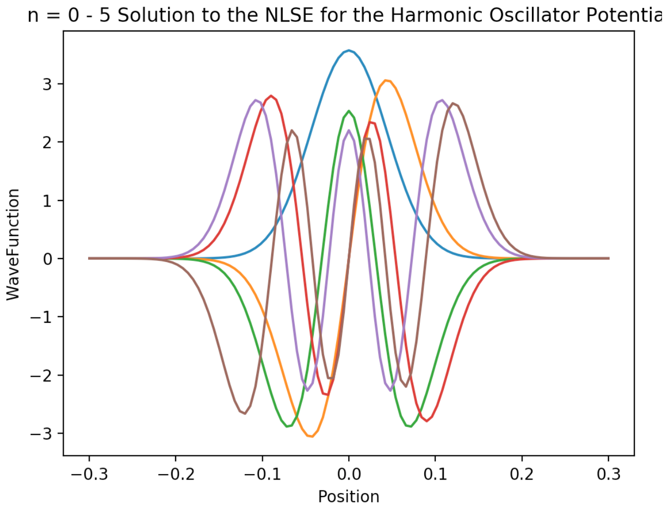

FIG. 2. Harmonic oscillator solved in the discrete basis, using 100 points

This gives the following form for the Schrodinger equation:

$$d_k\psi_k+e_{k-1}\psi_{k-1}+e_{k+1}\psi_{k+1}=E\psi_k \tag{8}$$

and allows us to solve the Schrodinger equation as a matrix eigenvalue problem.

Introduction: Harmonic Oscillator Basis

The harmonic oscillator basis solution to the Schrodinger equation uses inner products of wavefunctions ($\langle\psi_m|H|\psi_n\rangle$) to find the matrix elements, and transforms the resulting Hamiltonian into the harmonic oscillator basis. The solutions to the harmonic oscillator are given by the equation

$$\psi_n(x) = \left(\frac{m\omega}{\pi\hbar}\right)^{1/4} \frac{1}{\sqrt{2^n n!}} H_n(\xi) e^{-\xi^2/2} \tag{9}$$

where

$$\xi = \sqrt{\frac{m\omega}{\hbar}} x \tag{10}$$

The matrix elements are found by taking the inner product $\langle m|H|n\rangle$, which yields

$$\begin{aligned} &\frac{1}{2m}\langle n|p^2|m\rangle + \langle n|V|m\rangle \\ &= -\frac{\hbar\omega}{4}[\sqrt{m+1}\sqrt{m+2}\delta_{n,m+2} - \sqrt{m+1}\sqrt{m+1}\delta_{m,n} \\ &\quad - \sqrt{m}\sqrt{m}\delta m, n + \sqrt{m}\sqrt{m+1}\delta_{n,m-2}] \\ &\quad + \int_{-\infty}^{\infty} \psi_n^*(x)V(x)\psi_m(x)dx \end{aligned} \tag{11}$$

After the matrix elements are found, the resulting matrix is diagonalized to give the Hamiltonian, which has dimensions (number of wavefunctions, number of wavefunctions). This matrix is then multiplied by a transformation matrix with dimensions (number of steps, number of wavefunctions) to give the solution in the harmonic oscillator basis.

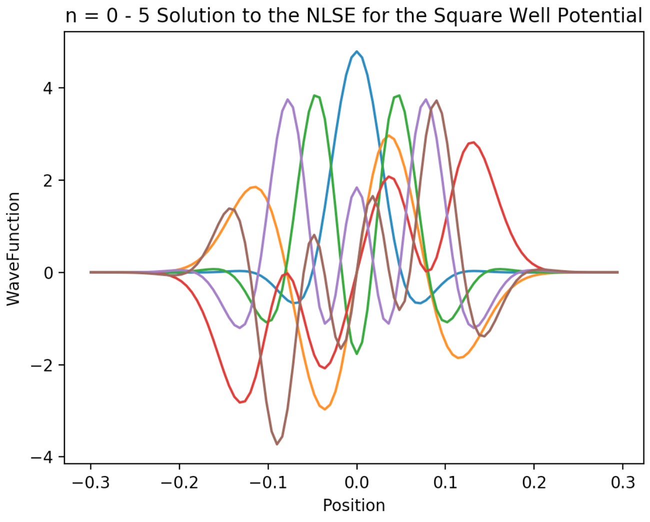

FIG. 3. Square well solved in the harmonic oscillator basis, using 5 basis functions

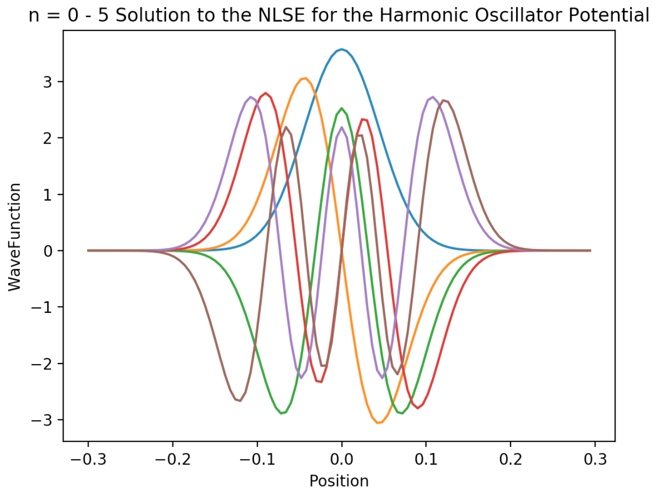

FIG. 4. Harmonic oscillator solved in the harmonic oscillator basis, using 5 basis functions

Programming

We structured our code by defining two classes, one for the discrete basis and one for the harmonic oscillator basis. Both classes have attributes *potential* (the potential function), x_{min}, x_{max}, *n steps* (number of steps - used for determining matrix dimensions), *particle mass*, *h* (to determine the step size), and *xPoints* (to determine the points to evaluate the wavefunction at). The harmonic oscillator class also has attributes *omega* and *hbar*.

The discrete class includes separate functions for finding each element of the matrix, creating the matrix, and diagonalizing the matrix. The "matrix element finder" function uses equations (6) and (7) to find the diagonal and off-diagonal matrix elements, respectively. The "matrix maker" function creates a zero matrix of dimensions (*n steps* + 1, *n steps* + 1). This function then calls "matrix element finder" to fill in the diagonal and off-diagonal elements of the matrix, giving the Hamiltonian. The "matrix solver" function uses Numpy's "linalg.eigh" function to diagonalize the Hamiltonian, and then takes the transpose of this matrix, giving the eigenvectors and eigenvalues.

The harmonic oscillator class first includes a function for defining the harmonic oscillator wavefunction. This function calls a separate module to evaluate Hermite polynomials. The momentum and potential inner products corresponding to the Hamiltonian are defined in separate functions. In addition, a function for defining (NOT integrating) the integrand in the potential term is defined separately. A "matrix element finder" function adds the momentum operator and potential operator terms, and "matrix maker" and "matrix solver" functions are defined similarly to those in the discrete class. The zero matrix initially created by "matrix maker" has dimensions (number of functions, number of functions).

The plotting function, defined outside the two classes, includes a component for transforming the Hamiltonian given by the harmonic oscillator solver into the harmonic oscillator basis. The transformation component takes the Numpy dot product of a transformation matrix and the eigenvector matrix given by the harmonic oscillator solver. The transformation matrix has dimensions (number of steps, number of functions), with elements representing each of the harmonic oscillator functions evaluated at a different point; thus multiplying the transformation matrix and the eigenvector matrix gives the solution in the harmonic oscillator basis.

Analysis of Correctness

To test whether this program gives the correct results, we used our code to solve two problems with known solutions - the infinite square well and the harmonic oscillator. Various plots of our solutions are shown throughout the report.

As shown in *Figure 3*, the solution to the square well potential solved in the harmonic oscillator basis is not correct, which is expected. When solving the TISE in the harmonic oscillator basis, one should integrate the inner product out to positive and negative infinity (the SciPy "integrate.quad" function does this). When solving the square well, however, one should just integrate the inner products between the endpoints of the well. Therefore, the limits of integration in the harmonic oscillator basis are much too large for solving square well potentials, and should not yield the correct results.

Figure 2 and *Figure 4* show that certain solutions for the harmonic oscillator potential in the harmonic oscillator basis are negative solutions of the results from the discrete basis. Since any "negative" solution to an eigenvalue problem is still a solution, though, this is no cause for concern. *Figure 5* shows the eigenvalues of the square well as a function of n (corresponding to the n^{th} wavefunction). *Figure 6* shows the eigenvalues of the harmonic oscillator as a function of n. As expected, these are a line, since for the harmonic oscillator $E_n = (n + \frac{1}{2})\hbar\omega$.

To achieve reasonable numerical precision, at least 100 points should be used when solving a potential in the discrete basis. This yields at least 100 points at which the code can evaluate the wavefunction solutions and plot the results. *Figure 7* and *Figure 8* show the convergence of the eigenvalues for different numbers of points for the square well and harmonic oscillator potentials, respectively, solved in the discrete basis. As shown in *Figure 8*, the eigenvalues of the harmonic oscillator potential appear linear at first, then parabolic, and are "choppy" near *xmax*. This is due to the fact that the actual harmonic oscillator

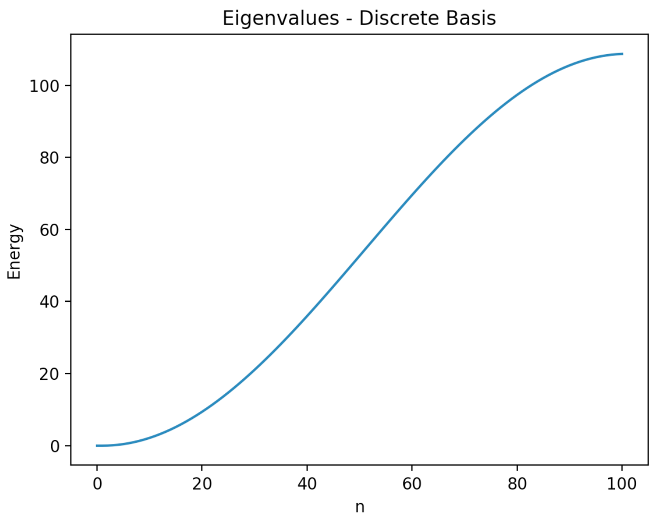

FIG. 5. Eigenvalues of the square well potential, solved in the discrete basis using 100 points

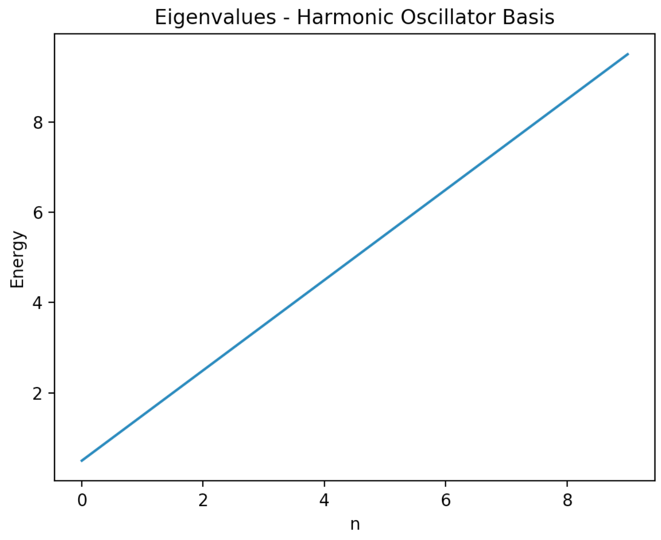

FIG. 6. Eigenvalues of the harmonic oscillator potential, solved in the harmonic oscillator basis using 5 basis functions

wavefunction goes to zero at infinity, but the discrete basis only evaluates the wavefunction out to a finite *xmin* and *xmax*. Therefore, the harmonic oscillator potential is restricted as if it were a square well potential instead, so the behavior of its eigenvalues approaches the behavior of the square well eigenvalues as x increases.

In the harmonic oscillator basis, SciPy's "integrate.quad" function integrates the inner products of wavefunctions and the potential out to positive and negative infinity, which effectively takes the place of evaluating the inner products at many discrete points, as is done in the discrete basis solver. Eigenvalues of the two test potentials in the harmonic oscillator basis are not plotted, however,

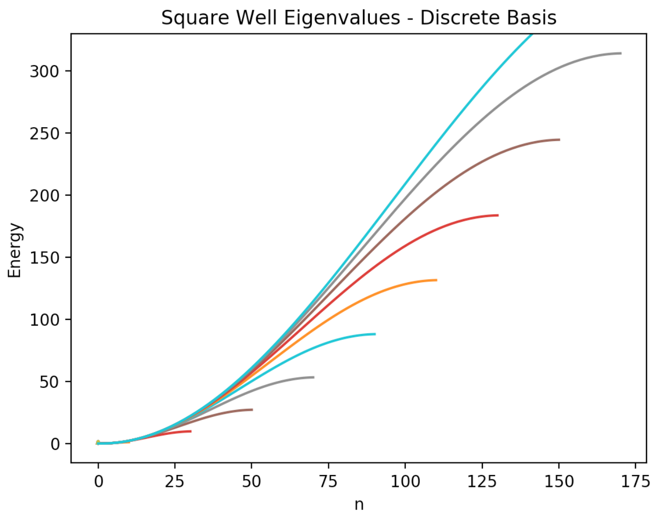

FIG. 7. Convergence of eigenvalues for the square well potential solved in the discrete basis. Minimum number of steps = 10, maximum number of steps = 190, plotted in increments of 20.

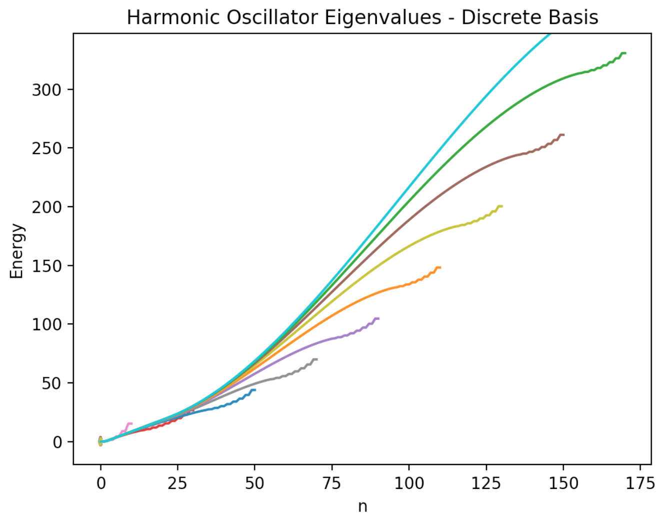

FIG. 8. Convergence of eigenvalues for the harmonic oscillator potential solved in the discrete basis. Minimum number of steps = 10, maximum number of steps = 190, plotted in increments of 20.

since the solution to the square well potential in the harmonic oscillator basis is known to be incorrect, while the eigenvalues of the harmonic oscillator potential are simply a straight line. Solutions to potentials in the harmonic oscillator basis vary depending on the number of basis functions used, and because the inner products are already integrated out to extremely large values, a much smaller number of basis functions yields an accurate solution. We found accurate solutions using as few as five basis functions.

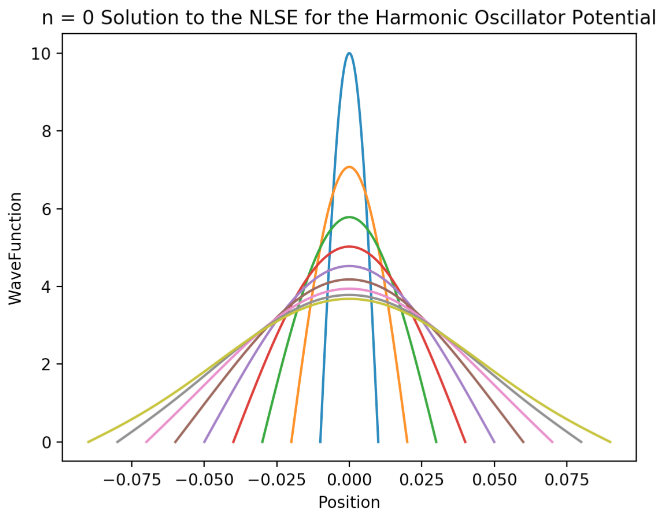

FIG. 9. Ground state of the harmonic oscillator, solved in the discrete basis with varying point density.

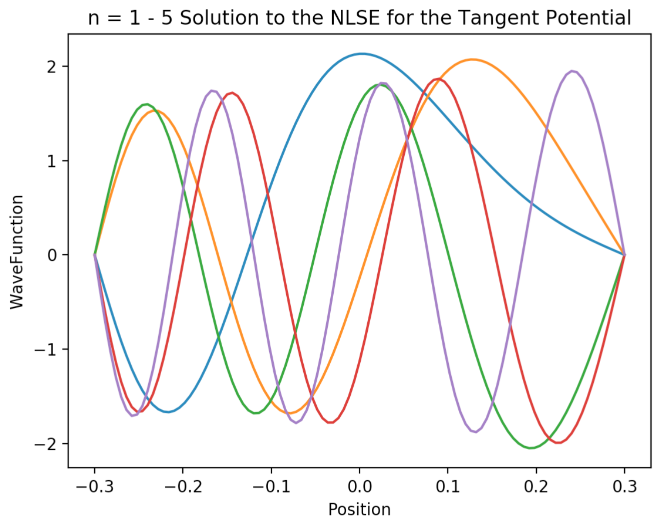

FIG. 10. Tangent potential solved in the discrete basis, using 100 points

We also varied the point density in our plots. *Figure 9* shows the ground state solution to the harmonic oscillator, solved in the discrete basis, using 100 points but with different values of $xmin$ and $xmax$, thus different point densities.

We also used this code to solve the tangent potential $V(x) = tan(x)$ in both the discrete and harmonic oscillator bases (*Figure 10* and *Figure 11*). The $n = 0$ case is not plotted for the discrete basis, since this yields a delta function.

Units

The units of mass in this code are keV. In the harmonic

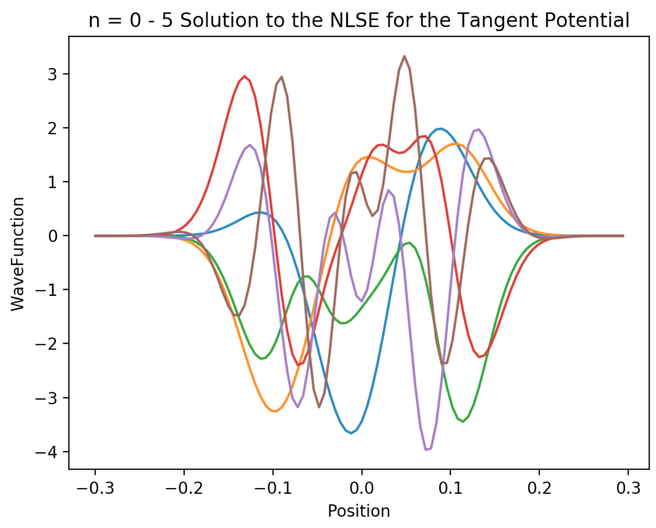

FIG. 11. Tangent potential solved in the harmonic oscillator basis, using 5 basis functions

oscillator basis, we worked in units such that $\hbar$ and ω were equal to 1.

Conclusion

The Schrodinger equation can be solved computationally through matrix diagonalization in multiple bases. We have developed a program to solve the time-independent Schrodinger equation in two bases - the discrete basis and the harmonic oscillator basis. As shown in our plots, the solutions given by our code are consistent with known solutions. Therefore, this code can be used to solve the Schrodinger equation for potentials that can only be solved numerically.

Acknowledgements

I would like to thank my partner, Ben, for working with me for eight entire weeks on this project, and for spending hours trying to figure out why our solutions were "wrong" (i.e. negative) when they were actually correct because we both forgot how linear algebra worked. I would also like to thank Sam and Judah, for helping us fix our harmonic oscillator basis when their code worked and ours didn't. Finally, I would like to thank Patrick for his guidance in this project (#eigenboisneedhelp), and for bringing us doughnuts. (Please bring more doughnuts.) Eigenbois out.