Numerical Solutions to Schrodinger's Equation Notes by Jonathan Sievers for McGill PHYS 357

$$\mathrm{H}\psi = E\psi$$

Oh, what a deceptively simple-looking equation. Solving it analytically in even the simplest cases is fiendishly difficult since the vast majority of solutions are unphysical. In many textbooks, inspired guesses for operators appear seemingly by magic. As soon as you change the potential even a little, those inspired guesses become useless. Even if you, by some miracle, have an analytic solution, understanding its time evolution generally remains out of intuitive reach.

Fortunately, we can solve the Schrodinger equation numerically in a wide range of cases in just a few lines of code. We can add time evolution with just a few more lines of code. We can visualize the time-evolving solution, allowing us to get a gut feel for what happens in a way that is virtually impossible analytically. The purpose of this note is to lay out how to set up the Schrodinger equation numerically, and show how to solve it in a few cases.

1. Schrodinger as an Eigenvalue Problem

If the wave function were a vector and the Hamiltonian a matrix, then the time-independent Schrodinger equation is obviously an eigenvalue problem. Of course, the wave function is a function living in Hilbert space, but we can approximate it with a discretized version. In particular,

$$\Psi(x) \rightarrow \Psi(x_i)$$

for some finite set of x_i. We'll try to work out the wave function at those points, and hope it's "smooth enough" between points.

How do we convert the Schrodinger equation into discrete points? We'll stick with 1-D problems with contant potentials for now, in which case we have:

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x)\right)\Psi = \mathrm{E}\Psi$$

. The potential is straightforward - $\mathrm{V}(x_i)$ is just the potential evaluated at x_i. For the momentum term, we need to take the second derivative of Ψ using just the values of Ψ we have. If we Taylor expand Ψ to second order, we have

$$\Psi(x_0 + \delta x) = \Psi(x_0) + \frac{d\Psi}{dx}|_{x_0}\delta x + \frac{1}{2}\frac{d^2\Psi}{dx^2}|_{x_0}\delta x^2 + \mathcal{O}(\delta x^3)$$

If we add $\Psi(x_0 + \delta x)$ to $\Psi(x_0 - \delta x)$ then the first derivatives cancel[1] and we have

$$\Psi(x_0 + \delta x) + \Psi(x_0 - \delta x) = 2\Psi(x_0) + \Psi^{''}\delta x^2 + ...$$

[1]In fact, *all* odd derivatives cancel so in fact our expression here is accurate to fourth order. If we shrink δx by a factor of two, our estimate of the second derivative will improve by a factor of 16!

We can solve this for Ψ'' to get:

$$\Psi''(x_0) = \frac{\Psi(x_0 - \delta x) - 2\Psi(x_0) + \Psi(x_0 + \delta x)}{\delta x^2}$$

If we assume our x_i are equally-spaced (which they may as well be because you get to pick the x_i) with $x_{i+1} - x_i = \delta x$, let's introduce the shorthand that $\Psi x_i \equiv \Psi_i$. Our discretized second derivative is then:

$$\Psi_i'' = \frac{1}{\delta x^2}\left(\Psi_{i-1} - 2\Psi_i + \Psi_{i+1}\right)$$

We can plug this into the Schrodinger equation to get:

$$-\frac{\hbar^2}{2m\delta x^2}\left(\Psi_{i-1} - 2\Psi_i + \Psi_{i+1}\right) + \mathrm{V_i}\Psi_i = \mathrm{E}\Psi_i$$

We can clean this up by multiplying through by $\frac{2m}{\hbar^2}$ and defining $\tilde{V} \equiv \frac{2m}{\hbar^2}\mathrm{V}$ and $\tilde{E} \equiv \frac{2m}{\hbar^2}\mathrm{E}$, leaving us with:

$$-\Psi_{i-1}/\delta x^2 + (2/\delta x^2 + \tilde{V_i})\Psi_0 - \Psi_{i+1}/\delta x^2 = \tilde{E}\Psi_i \tag{1}$$

Equation 1 gives us a set of N coupled linear equations. Whenever you find yourself faced with coupled linear equations, I strongly encourage you to write them as a matrix. In our case we can indeed do this by defining $\tilde{\mathrm{H}}$ to be:

$$\begin{bmatrix}
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\cdots & -\frac{1}{\delta x^2} & \frac{2}{\delta x^2} + \tilde{V_i} & -\frac{1}{\delta x^2} & 0 & 0 & \cdots \\
\cdots & 0 & -\frac{1}{\delta x^2} & \frac{2}{\delta x^2} + \tilde{V}_{i+1} & -\frac{1}{\delta x^2} & 0 & \cdots \\
\cdots & 0 & 0 & -\frac{1}{\delta x^2} & \frac{2}{\delta x^2} + \tilde{V}_{i+2} & -\frac{1}{\delta x^2} & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots
\end{bmatrix}$$

In slightly more detail, $\tilde{H}$ is a tri-diagonal matrix where

$$\tilde{H}_{i,i} = \frac{2}{\delta x^2} + \tilde{V_i} \tag{2}$$

$$\tilde{H}_{i,i+1} = \tilde{H}_{i,i-1} = -\frac{1}{\delta x^2} \tag{3}$$

We can now write the coupled system of Equation 1 as:

$$\tilde{H}\Psi = \tilde{E}\Psi \tag{4}$$

While it might seem like we've just looped around back to where we started from, we now know how to write out the elements of $\tilde{H}$, and so we can solve this with a single call to an eigenvalue routine (*e.g. numpy.linalg.eigh*) to give us all the eigenvalues (the energies) of the system and their corresponding eigenvectors (the stationary wave functions).

2. Example: The Simple Harmonic Oscillator

Most quantum texts devote a full chapter to solving the quantum simple harmonic oscillator (SHO). After a *lot* of math, one can show that the energy levels are (1/2,3/2,5/2...) $\hbar\omega$, where ω is the classical frequency of the SHO. If we want to solve this numerically, then we'll need to set up $\tilde{H}$. Python code to do this is:

```
import numpy as np
from scipy.linalg import eigh_tridiagonal
x=np.linspace(-15,15,2001)
dx=x[1]-x[0]
V=0.25*x**2
H0=2/dx**2*np.ones(len(x))+V
H1=-1/dx**2*np.ones(len(x)-1)
e,v=eigh_tridiagonal(H0,H1)
```

Python's scipy library has a specialized routine to find the eigenvalues/eigenvectors of tridiagonal matrices. We don't have to use it, but it runs much, much faster than the usual numpy routines[2]. We can now check the smallest eigenvalues:

```
>>> print(e[:10])
[0.49999648 1.49998242 2.4999543  3.49991211 4.49985585 5.49978554
 6.49970116 7.49960271 8.4994902  9.49936363]
```

This agrees very well with our expected values of (1/2, 3/2, 5/2...). They won't be exact because our computation is only an approximation of the ideal case (finite spacing between points, finite x-range), but they're very close. The eigenvectors should contain the wave functions corresponding to the energies. Figure 1 shows the numerical vs. analytic ground-state wave functions. Once again, the agreement is excellent, with agreement in the part-per-million range.

In just 8 lines of code (two of which were imports), we were able to solve for the energy levels and corresponding wave functions of generic potentials. We happened to use the simple harmonic oscillator, but the same code would work had we put in a different potential in line 5. Want to see what happens when you add a barrier to the center of a harmonic oscillator? Just add a barrier to the potential! Got some other wacky potential you'd like to investigate? Go for it!

3. Validity of Results

Of course, we do need to make sure that our numerical answers are at least approximately correct[3]. The astute reader may have noticed we've been incredibly sloppy at the edges of our region. The second derivative requires looking at neighboring points on both sides, but there's only one neighbor at the edges. Did we mess up? Not really, because to live in Hilbert space, a wave function needs to go to zero at infinity.

[2]In fact, most eigenvalue routines start by reducing a matrix to tri-diagonal form, then using specialized tri-diagonal routines to get the final eigenvalues/eigenvectors. The initial reduction is expensive, so we can happily skip it.

[3]One can skip this section on first reading. Before doing extensive simulations, you probably want to have a look at the numerical pitfalls you're likely to encounter, at which point I would encourage you to look at this section more carefully.

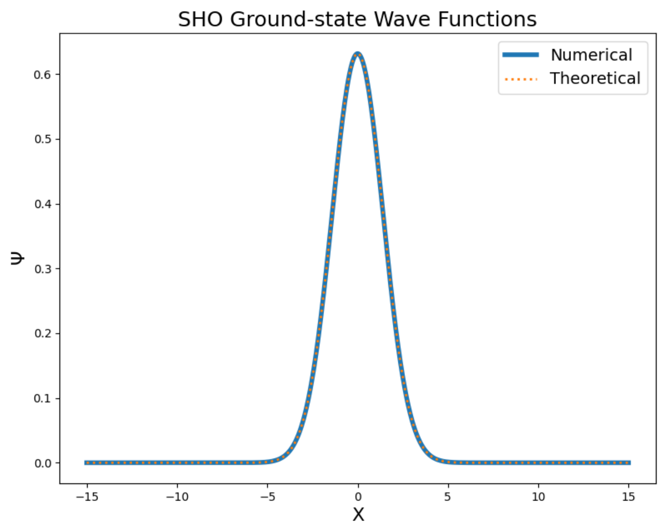

Fig. 1.— Numerical vs. Analytic wave functions for the simple harmonic oscillator ground state. The fractional accuracy is in the parts-per-million range after normalization.

Generically, any wave function that hasn't gone to zero at the domain edge probably won't do so outside of the region we've calculated. If a solution has gone to zero, though, we can look at our system and see that if we extended the range, any eigenvector that had gone to zero by the edges will remain an eigenvector of our extended range. To see this, notice that multiplying a vector by a tridiagonal matrix just mixes a point in the vector with its neighbors to generate the output. If my eigenvector has at least two zeros at an edge, and I extend the matrix by one row and zero-pad the vector by one element (from size n to $n+1$), then v_n goes from $A_{n,n-1}v_{n-1}+A_{n,n}v_n$ to $v_n \rightarrow A_{n,n-1}v_{n-1}+A_{n,n}v_n+A_{n,n+1}v_{n+1}$. Since we required $v_{n-1}=v_n=0$ and we put in $v_{n+1}=0$, then this remains true since our three relevant v_n are all zero. So, we didn't screw up the n^{th} entry in the eigenvector - it's still zero. The $n+1^{th}$ entry is:

$$v_{n+1} = A_{n+1,n}v_n + A_{n+1,n+1}v_{n+1} = 0 \tag{5}$$

if both v_n and v_{n+1} are zero. We can repeat this procedure to keep extending our matrix, and keep zero-padding our eigenvectors. So, because our Hamiltonian is tri-diagonal, once we have an eigenvector of a piece of the Hamiltonian that has gone to zero at the edges of that piece, that eigenvector remains an eigenvector if we take a larger piece of the Hamiltonian. Conversely, if the eigenvector has not gone to zero, then Equation 5 breaks down.

With this in mind, we'll look at the first entry of each eigenvector for our solution. Before we do that, though, let's try to figure out what we might expect to see. In the case of the SHO, we know what the energy levels are, so we can estimate where our solution might fail. Our $x-$domain (randomly picked) was [-15,15], and since the potential is $0.25x^2$, then at the edge, the potential is $0.25 * 15^2 = 56.25$. Any state with energy larger than that will generically have a non-zero wavefunction at the edge, while any state with energy much smaller than that should have exponentially decayed away. Since the energy levels are $0.5+n$, then we expect ~ 50 well-behaved states from our numerical solution. If we look at the eigenvectors at the edge in Figure 2, this is exactly what we see. The first ∼50 eigenvectors are zero at the domain edge, while the eigenvectors rapidly become non-zero above that. You can also see the eigenvalues in the right panel - these also change their behavior around mode 50. All these lines of evidence tell use we should trust the first ∼50 modes, and not trust the rest. In the case of the SHO, we were able to predict where our numerical solutions would break down, but in general you will have to be careful and decide for yourself what you can trust. A pretty hard-and-fast rule is you should never trust any state with an energy level higher than the potential at the edge of the domain.

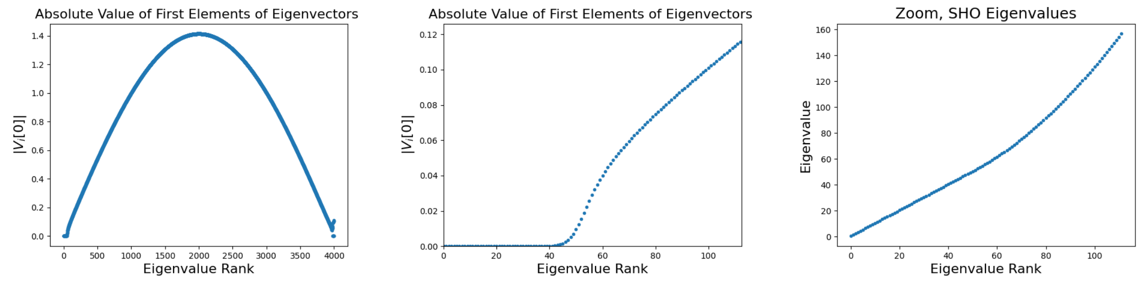

Fig. 2.— First entry of the eigenvectors for our simple harmonic oscillator. The left panel shows the full eigenvalue range. Most of these solutions are not physical, but if you zoom in on the low-energy eigenvectors(center panel), you can see that there are ∼50 modes that are indeed zero at the domain edge. It is only these modes that can correspond to physically realistic solution to the Schrodinger equation. The right panel shows the corresponding eigenvalues, and you can see there's a kink in the spacing of the energy levels around 50 as well. These results show that we should really only trust the first ∼50 states from our numerical simulation. If you need more states, then you would need to extend the $x-$range of the simulation.

In addition to making sure the wave function has gone to zero at the edges for valid solutions, we also need to make sure our domain has been sampled finely enough. The high-level picture here is that we need to make sure our second derivative estimate doesn't contain any surprises. You might think that we need to sample finely enough that the second derivative doesn't change appreciably between simulation points. That is in fact too strict a requirement. A more realistic (and much looser!) requirement is that we can predict the second derivative at a point by taking the average of the second derivatives at the neighboring points. Mathematically, that's equivalent to saying the *fourth* derivative has to be small, since the slope of the second derivative is the third derivative, and errors in our prediction based on the third derivative come from fourth-order (and higher) effects. In general, as you look at higher energy states, they oscillate faster and so have larger higher-order derivatives.

Figures 3 and 4 show these numerical effects in action. Solutions where the numerical curvature at a point is well predicted by averaging the numerical curvature of neighboring points are generally valid. In the case of the SHO, we can check this by comparing the eigenvalues of those solutions to the theoretical ones. When the curvature model is robust, the eigenvalues come out very close to what we expect, with the errors in the energy much smaller than the differences between energy levels. As the neighbor-predicted curvature diverges from the numerical value, then the energies diverge from their true values and the solutions can no longer be trusted.

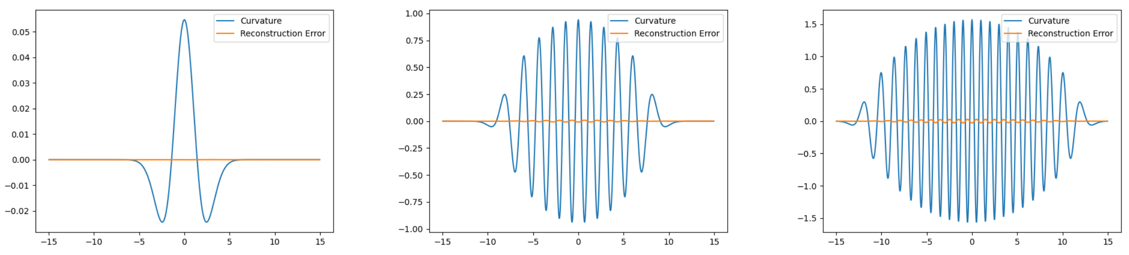

Fig. 3.— Second-derivatives and error in linear interpolation for the $(0, 20, 40)^{th}$ modes of the SHO with $x-$spacing of 0.03. For all of these modes, the typical interpolation error is much smaller than the typical wave function, so we would expect these solutions to be relatively accurate. The corresponding eigenvalues are [0.49998, 20.488, and 40.454], in good agreement with the theoretical expectation of [0.5,20.5, and 40.5]. As we go to higher energy solutions, the accuracy does drop, as expected from the increasing curvature.

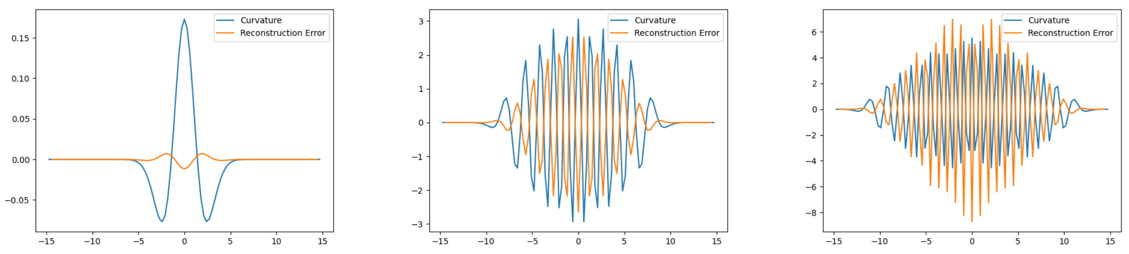

Fig. 4.— Same as Figure 3, except now with $x-$spacing of 0.3. The ground state looks reasonable, but we can see our second derivative has become a relatively poor estimate for the higher-energy solutions. Our recovered eigenvalues are now [0.4986, 19.24, and 35.10]. The ground state is close to the theoretical expectation of 0.5, there's a noticeable error in the $n = 20$ state, and the $n = 40$ state is now off by quite a bit (35.1 vs. 40.5 expected).

We have now seen how to determine just by looking at the outputs of an eigen decomposition which of the eigenvectors/values correspond to true physical solutions of our desired system. In the case of the SHO, we could check these numbers because we knew what to expect, but we didn't actually need to do that. If you're running up against computational limits, it's absolutely worth going through this exercise. However, if you have computing room to spare, the lazy thing is to re-run your problem with 1) double the resolution and 2) double the $x-$range. If the solutions you care about only change by amounts much smaller than you care about, you're probably fine.

4. Time Evolution

The eigen-decomposition we've seen gives us the solutions to the time-independent modes to the Schrodinger equation. We can evolve these solutions the same way we'd evolve our analytic ones.

Namely, let's start with a wave function at time $t = 0$:

$$\Psi(x, t = 0) = \sum a_i \Psi_i(x) \tag{6}$$

where $\Psi_i(x)$ is the i^{th} eigenvector with corresponding energy/eigenvalue E_i, and $\sum a_i^* a_i = 1$. The solution at time t is then:

$$\Psi(x, t) = \sum a_i e^{iE_i t/\hbar} \Psi_i(x) \tag{7}$$

That is, each mode picks up a phase set by its energy and the length of time the system has been evolving. Once we have our starting amplitudes a_i, then we just evaluate our system at time t using Equation 7.

In order to find our initial amplitudes, all we have to do is take our starting wave function $\Psi(x, t = 0)$, which is generally an input to our solver, and multiply by the transpose of our eigenvectors. Since the eigenvectors are orthogonal by construction, then

$$\Psi_i^\dagger \Psi = \Psi_i^\dagger \sum a_k \Psi_k = a_i \tag{8}$$

So, we just take our matrix of eigenvectors, and multiply it by our starting wave function to get all the a_i. We then take those coefficients along with their associated eigenvalues/eigenvectors, and plug them into Equation 7. We'll see an example of this next.