7. Numerical Solutions of the TISE

Besides the infinite square well, there aren't many potentials $V(x)$ for which you can find the energies and eigenfunctions exactly, using pencil and paper. Nowadays, however, that's hardly a handicap because we have powerful computer systems that can solve the TISE numerically for virtually any $V(x)$. In this lesson I'll describe two different approaches to computer-assisted numerical solutions.

The shooting method

For any given $V(x)$ and energy E, the TISE is an explicit second-order differential equation that tells you the curvature of the function $\psi(x)$ at any point if you already know the value of $\psi(x)$ at that point:

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

If you also know the derivative of $\psi(x)$ at that same point, then you can use it and the calculated curvature to find the approximate value of ψ and its derivative a little to one side, say at $x + dx$. The smaller the step dx, the more accurate the approximation. Repeating the process, you can construct the entire function $\psi(x)$. Plotting an accurate graph of the entire function $\psi(x)$ might require a thousand calculation steps or more, but today's computers can carry out a billion calculations per second. The point is that a unique solution $\psi(x)$ exists, and isn't hard to find, for any given $V(x)$, E, and boundary conditions consisting of ψ and $d\psi/dx$ at an arbitrary point.

To illustrate the method, let me pick a specific $V(x)$: the *finite square well*, defined as

$$V(x) = \begin{cases} 0 & \text{for } -a/2 < x < a/2, \\ V_0 & \text{elsewhere.} \end{cases} \tag{2}$$

This is the same potential as for the infinite square well, with ∞ replaced by V_0; I've shifted the well to center it at $x = 0$ because the resulting symmetry will slightly simplify the computer code and the description of the solutions. (This is actually an example that *can* be solved exactly, aside from the need to numerically solve a transcendental equation to match the wavefunction at the well boundary. But here I'll use it to illustrate the much more general method of numerically solving the TISE.)

Before typing a physics equation into a computer, you should almost always rewrite it in a system of units that is "natural" to the problem being solved. Doing so will free you from working with numbers that are awkwardly large or small, and from having to supply numerical values for parameters that turn out to be irrelevant

to the mathematics. For example, in this problem the natural unit of distance is a, the width of the well, so I'll set $a = 1$ in my computer code. I will also set $m/\hbar^2 = 1$; this combination has dimensions of $(\text{energy})^{-1}(\text{distance})^{-2}$, so setting it equal to 1 determines our unit of energy: all energies will now be measured in units of $\hbar^2/ma^2$.

Note that after making these choices, we do *not* have the freedom to also set $V_0 = 1$. In other words, different V_0 values (in these units) represent different problems to solve, and we'll have to choose a specific V_0 before solving the problem on a computer. And what would be some interesting V_0 values to choose? Well, recall that for an *infinite* square well, the energy eigenvalues are $h^2n^2/8ma^2$. Plugging in $h = 2\pi\hbar$ and setting $\hbar^2/ma^2 = 1$, this becomes $(\pi^2/2)n^2 \approx 5n^2$, so the lowest energies in our units would be roughly 5, 20, 45, 80, and so on. The most interesting V_0 values should be in this range; values much less than 5 would tend not to trap the particle at all, while values much more than 100 start looking similar to infinity (at least for the low-energy states). I'll use $V_0 = 50$.

Without further ado, here is some Mathematica code for solving the TISE for a finite square well, using the units just described, with $V_0 = 50$:

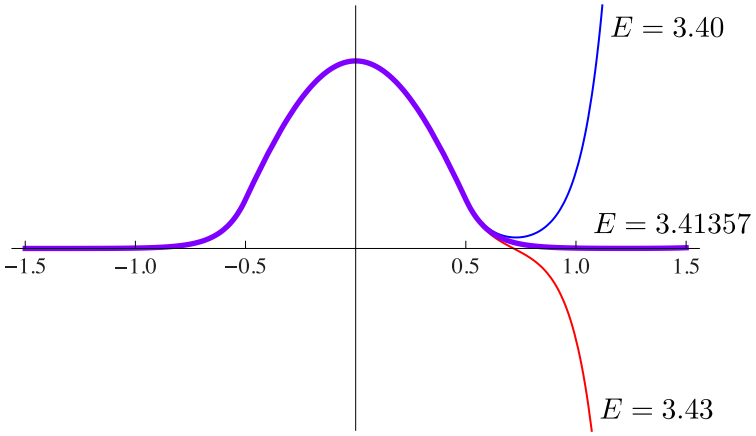
```
v[x_] := If[Abs[x] < 0.5, 0, 50];
energy = 5;
xMax = 1.5;
solution = NDSolve[{psi''[x]==-2(energy-v[x])psi[x],
    psi[-xMax]==0, psi'[-xMax]==0.001}, psi, {x, -xMax, xMax}];
Plot[psi[x] /. solution, {x, -xMax, xMax}]
```

First I define the potential energy function `v[x_]` (using lower case to avoid conflicts with built-in Mathematica functions, which always start with capital letters). Then I set my energy constant to an arbitrary initial guess (roughly equal to the ground state energy of an infinite well), and define a constant for the range of `x` values, `-xMax` to `xMax`, that I'll ask the computer to look at. The actual work is done by Mathematica's `NDSolve` function, to which I must first provide a list (in curly braces) of the differential equation(s) and the boundary condition(s). Note that all these equations are defined using double `==` signs, and that derivatives are denoted by primes (`'`). I'm supplying boundary conditions at the extreme left end of my interval (`-xMax`), where I'm hoping the wavefunction will have died out to practically zero—but to get a nontrivial result I must provide a nonzero initial slope, which I'm arbitrarily taking to be 0.001. After this list of equations, I supply the name of the function to solve for and a list consisting of the independent variable and its beginning and ending points. The result from `NDSolve` is stored in the variable `solution` as what Mathematica calls an interpolating function; the last line of code plots a graph of this function.

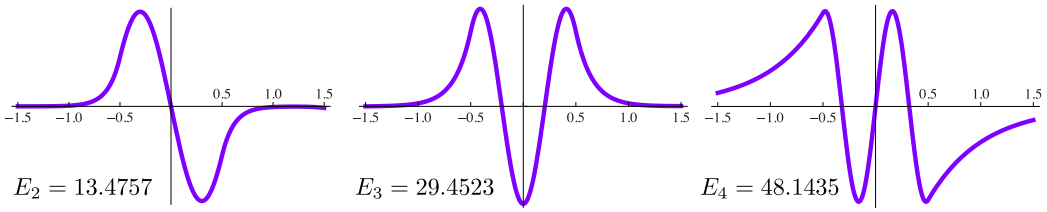
If you execute the code above (or its equivalent in some other software environment), you'll get a plot of a function that rises gradually from left to right, peaks a little to the left of $x = 0$, then falls gradually and crosses the x axis a little to the

left of the edge of the well at $x = 0.5$. The function then becomes negative, but soon reaches $x = 0.5$ where it begins curving *away* from the horizontal axis, blowing up exponentially in the negative direction. This is *not* a normalizable wavefunction, and it teaches us a lesson: You can solve the TISE for *any* energy E, but not all energy values allow solutions that are normalizable.

The procedure, then, is to rerun the code with different E values until you get a solution that "lies down flat" to the right of the potential energy well. It's essentially a trial-and-error process, but with a little practice you can zero-in on an energy value that works, to several significant figures, in about 20 trials. Here is a plot showing three trials, one with the energy too low, one with the energy too high, and one with the energy just right to give the one-bump (ground-state) wavefunction:



In the same way, I found the next three excited-state energies and wavefunctions:



Notice that all of these solutions are sinusoidal inside the well and exponential outside it. (Because the exponential fall-off is so gradual with the last of these wavefunctions, I increased `xMax` to 4.0 to get a consistent result—though I still cut off the plot at ± 1.5.) Notice also that the energies are all significantly *less* than the corresponding infinite square well energies, $(\pi^2/2)n^2 \approx 4.93, 19.74, 44.41, 78.96$; that's because the finite well lets part of the wavefunction "spill out" beyond the edges, allowing the wavelength inside to be longer for the same number of bumps. There are no further normalizable solutions with $E < 50$. For $E > 50$, the solutions are sinusoidal even outside the well, like the solutions for a free particle. Thus,

this particular potential well admits exactly four bound-state solutions to the time-independent Schrödinger equation.

I haven't labeled the vertical axes on any of these graphs, because the vertical scales are determined by my arbitrary choice of $d\psi/dx = 0.001$ at the extreme left edge. To obtain normalized wavefunctions, we would have to compute $\int |\psi|^2 dx$ in each case, and divide ψ by the square root of the result.

There's one more thing to notice about the four solutions pictured above: Each of them is either an even or odd function of x. This will be true whenever the potential function $V(x)$ is an even function, so for all such potentials there's actually a better choice of boundary conditions: Instead of starting far out along the $-x$ axis, start at $x = 0$ and either set $\psi = 1$ and $d\psi/dx = 0$ to obtain the even functions, or set $\psi = 0$ and $d\psi/dx = 1$ to obtain the odd functions. (In both cases, the "1" is arbitrary; any other nonzero value will do.) Then both "tails" of the wavefunction will "wag" as you vary the energy, lying down flat when the energy is tuned to an eigenvalue. These boundary conditions avoid the awkwardness that arises when the starting point isn't far enough to the left. In this example I used the more awkward boundary conditions because this method works even when $V(x)$ isn't symmetric.

The algorithm that I've just described, in which we start at a point with a known boundary condition and adjust the energy until the other boundary condition is met, is called the *shooting method*, because it is reminiscent of shooting a projectile and tuning its launch speed (or angle) to hit a fixed target. The shooting method is extremely accurate and computationally efficient, though it can be a bit tedious, finicky, and difficult to automate.

Matrix diagonalization

Now let me describe a totally different numerical method for solving the TISE. This method tends to be more computationally intensive and less accurate than the shooting method, but it can often be useful nevertheless, for a variety of reasons. It is also extremely elegant and instructive.

Again, our goal is to solve the time-independent Schrödinger equation,

$$H\psi = E\psi, \tag{3}$$

for the unknown eigenfunctions ψ and their corresponding eigenvalues E. The basic idea is to write ψ as a linear combination of some collection of orthonormal basis functions, ψ_n:

$$\psi = \sum_{n=1}^{\infty} c_n \psi_n, \tag{4}$$

so that the goal is now to find the unknown coefficients c_n. The basis functions ψ_n are *not* the solutions to the Schrödinger equation that we seek (if they were, the problem would already be solved!). They should instead be some collection of relatively simple functions that form an orthonormal basis. Here I will take the

ψ_n to be the sine functions that are energy eigenfunctions for an *infinite* square well that's wide enough to contain (to a good approximation—we hope!) all of the eigenfunctions ψ that we care about:

$$\psi_n(x) = \sqrt{\frac{2}{b}} \sin\Big(\frac{n\pi x}{b}\Big) \qquad \text{for } 0 < x < b, \tag{5}$$

where b is the width of the (hypothetical) infinite well. (When I apply this method to the finite square well example in a moment, I will take b to be several times larger than a, and I'll shift the finite well so it's centered at $b/2$.)

Still working in general, we now insert the expansion for ψ (equation 4) into the TISE (equation 3), and move H and E inside the sums to obtain

$$\sum_{n=1}^{\infty} H c_n \psi_n = \sum_{n=1}^{\infty} E c_n \psi_n. \tag{6}$$

We can get rid of the sum on the right by using *Fourier's trick*: multiply on the left by ψ_m^* (for some arbitrary index m) and integrate over x:

$$\int_{-\infty}^{\infty} \psi_m^* \sum_{n=1}^{\infty} H c_n \psi_n \, dx = \int_{-\infty}^{\infty} \psi_m^* \sum_{n=1}^{\infty} E c_n \psi_n \, dx. \tag{7}$$

Now, on the right-hand side, we can move the integral inside the sum, and factor Ec_n out of the integral; the integral then gives simply δ_{mn}, because the ψ_n functions are orthonormal, and therefore the only term that contributes to the sum is the one with $n = m$. The left-hand side isn't so simple, but we can at least move the integral inside the sum and factor the c_n out of the integral. Thus we obtain

$$\sum_{n=1}^{\infty} \Big(\int_{-\infty}^{\infty} \psi_m^* H \psi_n \, dx \Big) c_n = E c_m. \tag{8}$$

The quantity inside the big parentheses is called the *mn matrix element* of the Hamiltonian operator,

$$H_{mn} = \text{“matrix element”} = \int_{-\infty}^{\infty} \psi_m^*(x) H \psi_n(x) \, dx, \tag{9}$$

and it's something that a computer can calculate straightforwardly. With this abbreviation, the TISE becomes

$$\sum_{n=1}^{\infty} H_{mn} c_n = E c_m, \tag{10}$$

which has the precise form of the eigenvalue equation for the *matrix* whose elements are H_{mn}:

$$\begin{pmatrix} H_{11} & H_{12} & \cdots \\ H_{21} & H_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \end{pmatrix} = E \begin{pmatrix} c_1 \\ c_2 \\ \vdots \end{pmatrix}. \tag{11}$$

Therefore the energy eigenvalues E are the eigenvalues of the H matrix, and the eigenfunctions $\psi(x)$ can be built from the basis functions $\psi_n(x)$ using the elements of the corresponding eigenvectors. This form of the TISE is useful because many mathematical computing environments include routines that can quickly find the eigenvalues and eigenvectors of reasonably large matrices. Of course, you first have to calculate all the matrix elements using equation 9, and that's often the most time-consuming part of the process.

Calculating the matrix elements is somewhat simplified if we break the Hamiltonian operator into two pieces:

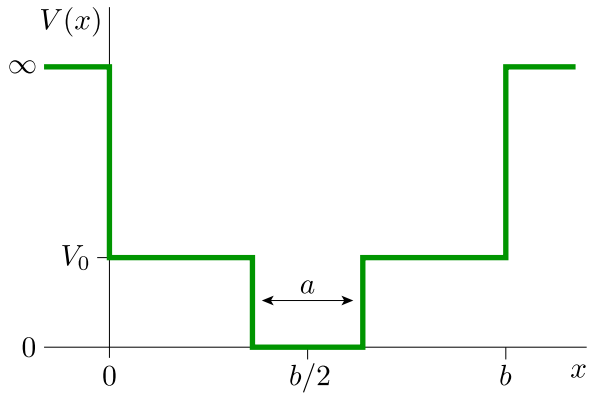
$$H = H_0 + \Delta V(x). \tag{12}$$

Here H_0 is the Hamiltonian whose eigenfunctions are $\psi_n(x)$, and $\Delta V(x)$ is whatever is left of our actual Hamiltonian. For our infinite-square-well basis, H_0 is just the kinetic energy operator, $\Delta V(x)$ is simply the $V(x)$ for our actual potential energy function, and the integrals will run only from 0 to b because the basis functions are zero elsewhere. Thus, the matrix elements are

$$\begin{aligned} H_{mn} &= \int_0^b \psi_m^*(x) H_0 \psi_n(x)\, dx + \int_0^b \psi_m^*(x) V(x) \psi_n(x)\, dx \\ &= E_n \delta_{mn} + \int_0^b \psi_m^*(x) V(x) \psi_n(x)\, dx, \end{aligned} \tag{13}$$

where in the first term I've used the fact that ψ_n is an eigenfunction of H_0 with eigenvalue E_n, and the fact that the ψ_n functions are orthonormal. Written out as a matrix, the first term would be simply a diagonal matrix whose entries are the eigenvalues of the infinite square well, $\pi^2 n^2 / 2b^2$ in natural units.

Now let me apply this method to the same example used above: a finite square well of width $a = 1$ and depth $V_0 = 50$ (in natural units). I'll center the finite well inside the infinite well:



Here is some Mathematica code to find the energy eigenvalues E and the corresponding eigenvectors $(c_1, c_2, \ldots)$ for this system:

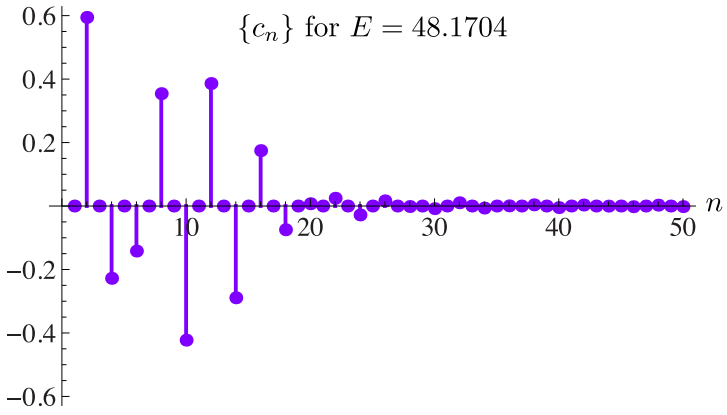
```
b = 4;
v[x_] := If[Abs[x - b/2] < 0.5, 0, 50];
nMax = 50;
basis[n_, x_] := Sqrt[2/b]*Sin[n Pi x/b];
vMatrix = Table[NIntegrate[basis[n, x]*v[x]*basis[m, x],
                {x, 0, b}], {n, 1, nMax}, {m, 1, nMax}];
h0Matrix = DiagonalMatrix[Table[n^2 Pi^2/(2*b^2), {n, 1, nMax}]];
{eValues, eVectors} = Eigensystem[h0Matrix + vMatrix];
eValues
```

In the first line I set $b = 4$, making the infinite well and its sine-wave basis functions four times as wide as the finite well of width $a = 1$ that we want to study. Then I define the potential energy function for the finite well, and next I set a cutoff `nMax` of 50, which will be the largest n value used throughout the calculations (and the dimension of the eigenvectors and matrices). The fourth line defines the sine-wave basis functions, properly normalized. The fifth line carries out 50×50 numerical integrals to obtain all the matrix elements of $V(x)$; this line is where nearly all of the computational time is spent, and takes about a minute to execute on my laptop computer. With those calculations out of the way, the next line sets up the diagonal matrix H_0, and finally we add these two matrices together, call Mathematica's `Eigensystem` function to find its eigenvalues and eigenvectors, and write out the list of eigenvalues. Here is the output:

```
{811.179, 780.225, 749.343, 719.877, 690.352, 662.866, 634.565, 608.144,
581.992, 555.874, 531.607, 506.755, 483.011, 460.415, 437.107, 415.862,
394.466, 373.352, 354.171, 333.967, 315.485, 297.698, 279.237, 263.283,
246.413, 230.559, 216.365, 200.724, 187.821, 174.449, 161.206, 150.544,
137.827, 128.003, 118.079, 107.628, 100.49, 90.6015, 84.3394, 77.3956,
70.5479, 66.78, 59.9372, 57.9902, 52.9032, 52.0802, 48.1704, 29.4697,
13.4824, 3.41566}
```

Mathematica inconveniently sorts the eigenvalues in descending order (and sorts the eigenvectors to correspond); although we could fix this with some list manipulation functions, I won't bother. The eigenvalues that we want are the last four, which are less than $V_0 = 50$ and therefore correspond to bound states of the finite well. Moreover, all four of these values agree to three significant figures with the more accurate values found by the shooting method. The other 46 eigenvalues are artifacts of the fictitious infinite well; in fact the finite well allows *any* energy greater than V_0.

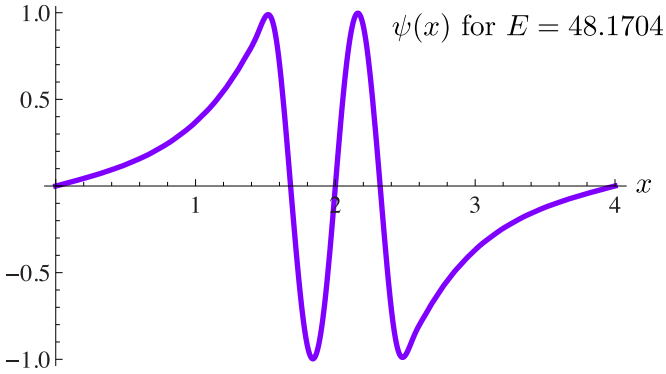
We could also print out the components of the eigenvectors, but a picture is worth a thousand numerals. You can plot the components of any eigenvector using an instruction like `ListPlot[eVectors[[47]], PlotRange -> All]`, remembering again that the bound states are numbered 47 through 50. With some further tweaking for aesthetics, this instruction produces the following plot:



To see the actual function of x that corresponds to this eigenvector, we simply build it out of the basis functions according to equation 4:

```
Plot[Sum[eVectors[[47,n]]*basis[n,x], {n,1,nMax}], {x,0,b}]
```

This instruction (again with some tweaking for aesthetics) produces the following plot:



To the eye, this graph is identical to the one obtained using the shooting method, except near the end points where it is forced to zero more quickly by the walls of the fictitious infinite well. (We could avoid this problem by making the infinite well wider, but then we would also need to increase the maximum n value to incorporate sufficiently short wavelengths.) Building the other three bound-state wavefunctions is no more difficult.

Finding the eigenvalues and eigenvectors of a matrix is often called *diagonalization*, because the eigenvectors could then be used as a new basis in which the matrix would be diagonal with entries equal to the eigenvalues. Solving the TISE is always equivalent to diagonalizing a matrix, and this method is practical whenever the infinite sums are well approximated by a reasonably small number of terms.