Numerical Evaluation of Wavefunctions Using Finite Differences

---

The definition of a derivative of a function, $u(x)$:

$$u'(x) \;=\; \lim_{h\to 0} \frac{u(x+h)-u(x)}{h}$$

cannot be implemented directly using a digital computer which represents real numbers with finite precision. Eventually, as h becomes small enough, the difference calculation $[u(x+h)-u(x)]$ becomes meaningless. The simplest way of dealing with this is to make h small but keep it finite

$$u'(x+\frac{h}{2}) \;\approx \frac{u(x+h)-u(x)}{h}$$

with h being 'small enough' and yet 'not too small'. This estimate of the derivative can be used as an approximation anywhere in the interval between x and $x+h$, but it is most appropriate at the midpoint $(x+\frac{h}{2})$. When used at the midpoint, this approximation is called a central finite difference approximation. Any differential equation involving one or more derivative of a function $u(x)$ can be turned into a relationship among values of $u(x)$ (not differentiated) at two or more points: $u(x),\ u(x\pm h), u(x\pm 2h),$ *etc.* One example of this is the Schrödinger equation, which can be written as

$$u''(x) \;=\; f(x)u(x)$$

where $f(x) = (2m/\hbar^2)(V(x)-E)$. $V(x)$ is the potential energy function. Here it is assumed that the energy of the system, E, is known and the task is to find the eigenfunction corresponding to that energy. The straightforward way of proceeding is to use a finite difference approximation for $u''(x)$ on the left hand side. A central finite difference approximation for the second derivative can be obtained in an analogous way as the expression for u' above (apply the finite difference formula to u' to get an expression for u'')

$$u''(x) \;\approx \frac{u(x+h)+u(x-h)-2u(x)}{h^2}\ .$$

Inserting this into the Schrödinger equation gives

$$u(x+h) \;=\; (2+h^2 f(x))u(x) - u(x-h).$$

This is a recursion relation that can be used to obtain the function $u(x)$ at any point $u(x+nh)$ once it is known at two adjacent points. The fact that we need to know two values to start the calculation is directly related to the fact that Schrödinger equation is a second order differential equation and it takes two boundary (or initial) conditions to specify a unique solution.

It is relatively easy to use some kind of finite difference approximation to transform a differential equation into a recursion relation. But, not all ways of doing this are equally good, and some may not work at all. Two questions arise: (1) Is the algorithm stable? And, if so, (2) how accurate is the algorithm? The first question is hard to answer. The algorithm is stable if small errors (which necessarily are always there in a finite precision computer) made at a given step do not get magnified in subsequent steps. While there are mathematical ways to analyse stability, most practitioner in chemistry and physics start out by just trying the algorithm and too see if it is stable. It is always important to start working with simple problems for which the solution is known to test the algorithm, before proceeding to the harder problems. The issue of stability will be addressed more below. It turns out that one has to be careful to apply the recursion relation only in 'the right direction', in some cases.

The tool needed to address the second question, i.e. to analyse the accuracy is Taylor series expansions. This will also lead to an improved algorithm. A better way of deriving the algorithm above is to expand $u(x+h)$ and $u(x-h)$ about x and add the two expressions

$$u(x+h) + u(x-h) \;=\; 2u(x) \;+\; h^2 u''(x) \;+\; \frac{2h^4}{4!}u^{(4)}(x) \;+\; hot$$

where *hot* stands for 'higher order terms'. i.e. terms involving higher than the fourth power of h. We can eliminate the term involving $u''(x)$ by using the Schrödinger equation, $u''(x) \;=\; f(x)u(x)$. If we now ignore the term involving h^4 and the higher order terms, the result is the recursion relation derived before. From this analysis it is clear that the leading correction term (i.e. the term that is of lowest order in h of all the terms that have been neglected) is of fourth order. That is, the algorithm is correct to order h^4.

The Numerov Algorithm

It is possible to improve on the standard finite difference algorithm using a trick that is often applicable, not just on the time-independent Schrödinger equation. The improvement is substantial and this faster algorithm, called the Numerov method, is the method of choice in this case.

The higher the power of the step size, h, in the correction term, the larger the stepsize can be and fewer calculations need to be done. The algorithm we derived last time had a correction term of order h^4. It is possible to eliminate this correction term and get an algorithm that is of order h^6. The tool for doing this is again Taylor series expansions. We can do a Taylor series expansion for the second derivative of the wave function, u, at point x:

$$u^{(2)}(x+h) \;=\; u^{(2)}(x) \;+\; hu^{(3)}(x) \;+\; \frac{h^2}{2}u^{(4)}(x) \;+\; \frac{h^3}{3!}u^{(5)}(x) + \;\ldots$$

and

$$u^{(2)}(x+h) \;=\; u^{(2)}(x) \;-\; hu^{(3)}(x) \;+\; \frac{h^2}{2}u^{(4)}(x) \;-\; \frac{h^3}{3!}u^{(5)}(x) + \;\ldots$$

Add the two expressions to get

$$\frac{1}{2}[u^{(2)}(x+h) + u^{(2)}(x-h)] \;=\; u^{(2)}(x) + \frac{1}{2}h^2u^{(4)}(x) \;+\; O(h^6)$$

This gives an expression for the first and largest error term in the previous algorithm, $\frac{h^4}{4!}u^{(4)}(x)$, without introducing unknown quantities. The second derivative can easily be evaluated directly from the Schrödinger equation. The expression is

$$\frac{h^4}{4!}u^{(4)}(x) \;=\; \frac{h^2}{24}[u^{(2)}(x+h) + u^{(2)}(x-h) \;-\; 2u^{(2)}(x)]$$

After inserting this expression to eliminate the error term in the previous algorithm, and then using the Schrödinger equation to eliminate the second derivative, i.e. replacing $u^{(2)}$ with $f(x)u(x)$, the algorithm becomes

$$u(x+h) \;=\; \left(\frac{2+(10/12)h^2f(x)}{1-(1/12)h^2f(x+h)}\right)\; u(x) - u(x-h) \;+\; O(h^6)$$

This rather complicated recursion relation is the Numerov algorithm.

This is a very fast and stable method for calculating a wavefunction as long as the iterative procedure is not carried out (deep) into a classically forbidden region. The reason

for the instability can be understood in the following way. The general solution is a linear combination of an exponentially decreasing and an exponentially increasing function. The true wavefunction is exponentially decreasing into the classically forbidden region. At each step in the iterative calculation, a numerical error is necessarily made, first of all due to finite precision of the computer and secondly due to truncation errors (i.e. the higher order terms in the Taylor series expansions that were neglected). These numerical errors mix in a bit of the other component of the general solution, the exponentially increasing one. Since these errors grow at the same time that the true solution decreases, the function that is generated by the recursive algorithm is eventually dominated by the errors and becomes meaningless. Therefore, it is important to always start the Noumerov iterations deep inside a classically forbidden region and intergrate out towards lower potential energy regions.

Determining bound state energies

When determining bound states, we are dealing with a two point boundary condition for the Schrödinger equation

$$u(x) \to 0 \quad \text{as} \quad x \to -\infty$$

and

$$u(x) \to 0 \quad \text{as} \quad x \to \infty.$$

This means that we can't start up the finite difference algorithm using the boundary conditions directly. Also, the algorithm requires that we know the energy, but the bound state energies are unknown. The way to cope with this is as follows.

An iterative procedure is used where a guess is made for the energy, a solution of the differential equation is generated and then a correction to the energy is estimated. This process is repeated several times. Eventually, the algorithm converges to the true bound state energy (usually the one closest to the guess). A solution to the differential equation is generated by starting deep into the classically forbidden region at low x (on the left side of the well). Since the true solution decays exponentially, the wavefunction can safely be set to zero at a point that is deep enough into the classically forbidden region. This function will be called $v_L(x)$, and the starting point is $v_L(1) = 0$. In order to start the iterative algorithm, a value of the function is needed at the next point, $v_L(2)$. Since the

normalization of the wavefunction is arbitrary at this point (but will be fixed later), any value of $v_L(2)$ can be chosen. A large value of $v_L(2)$ will simply make the norm of the wavefunction large. Its good to work with a function whose maximum is on the order of unity, and it is typically sufficient to have about 5 significant figures of accuracy, so a value of $v_L(2) = 10^{-6}$ is a reasonable choice. Now the iterative algorithm can be started and a sequence of values for the wavefunction, $v_L(n)$, generated. As the iterations are continued beyond the potential minimum and into the classically forbidden region on the other side, the algorithm will become unstable. To cope with this, a second, function is generated starting from the other side, i.e. starting deep into the classically forbidden region on the large x side, iterating towards decreasing values of x (the iteration formula is the same, but the time step becomes negative, $h < 0$). This function will be called $v_R(x)$.

A true solution to the boundary value problem should be continuously differentiable (unless the potential is singular, the kinetic energy term blows up if the first derivative is not continuous). This means the function and its first derivative are continuous. We can construct a function for the whole interval from the two pieces $v_L(x)$ and $v_R(x)$ by picking a joining point, x_j and defining $u(x) = v_L(x)$ if $x < x_j$ and $u(x) = v_R(x)$ if $x > x_j$. This function satisfies the two boundary conditions in the sense that the function is zero at points deep into the classically forbidden regions, and it satisfies the Schrödinger equation (to a certain accuracy) at all points except x_j. There, the function is, in general, neither continuous nor is the first derivative continuous. The discontinuity of the function can easily be fixed by fixing the normalization. For example, dividing each function with the value at x_j $u(x) = v_L(x)/v_L(x_j)$ if $x < x_j$ and $u(x) = v_R(x)/v_R(x_j)$ if $x > x_j$ guarantees continuity of the function at x_j. The second problem, that the first derivative will in general not be continuous results from the fact that the energy value used in the finite difference algorithm is not a bound state energy. In fact, the difference between the derivative of the left hand solution, $v'_L(x_j)$, and the right hand solution, $v'_R(x_j)$, can be used to improve the guess that was made for the bound state energy. We can derive the iteration formula by evaluating the expectation value of the energy, $< H >$, using the generated function, $v(x)$. Calling the initial guess for the energy Q, the new guess for the energy is

$$Q_{new} = < H >$$

(This is very much in the spirit of iterative solutions to second order equations, for example, and converges only if the guess is 'close enough').

The expression for $< H >$ can be derived as follows:

$$< H > = \frac{\int\ v(x) \left[-\frac{\hbar^2}{2M} \frac{d^2 v}{dx^2} \ + \ V(x) v(x) \right]\ dx}{\int\ v^2(x)\ dx} \ .$$

(Note that the wavefunction that is generated by the iterative algorithm is necessarily real since we started with real values at the two starting points. Hence v^2 rather than $|v|^2$ in the denominator). The function v has been constructed in such a way that

$$-\frac{\hbar^2}{2M} \frac{d^2 v}{dx^2} \ = \ (Q \ - \ V(x))$$

at all points where the finite difference algorithm was applied. If this was true at all points, then $< H >= Q$. But, at one point, x_j, is there a correction, because two different solutions have simply been joined together there. The first derivative is (in general) discontinuous there. The first derivative of the joined function can be written as

$$u'(x) \ = \ \Theta(x - x_j)\ (u'_R(x) - u'_L(x))\ u'_L(x)$$

where $\Theta(x)$ is the Heavyside function,

$$\begin{aligned} \Theta(x) \ &= \ 0 \quad \text{when } x < 0 \\ &= \ 1 \quad \text{when } x > 0 \ . \end{aligned}$$

When the second derivative is taken, the discontinuity at x_j results in a delta function contribution, analogous to

$$\frac{d}{dx}\Theta(x) \ = \ \delta(x) \ .$$

Therefore, the second derivative of the joined function is

$$-\frac{\hbar^2}{2M} \frac{d^2 v}{dx^2} \ = \ (Q \ - \ V(x)) \ + \ (-\frac{\hbar^2}{2M})\ (v'_R(x_j) - v'_L(x_j)\ \delta(x - x_j) \ .$$

Plugging this into the integral expression for $< H >$ above, gives

$$< H > = \ Q \ - \ \frac{\hbar^2}{2M} \frac{(v'_R - v'_L) v(x_j)}{\int\ v^2(x)\ dx}$$

so the original guess Q should be corrected with the second term before next iteration. Sometimes this estimate tends to overshoot, so it is often good to be more conservative and

use only some fraction of the calculated change in the energy (i.e. multiply the correction term with a factor between zero and one).

In order to use this expresssion, an efficient estimate for the first derivatives, v'_R and v'_L, at the point x_j is needed. The simplest central difference formula is

$$u'(x) \;=\; \frac{[u(x+h) - u(x-h)]}{2h} \;.$$

Again, the accuracy of this formula can be analysed by applying Taylor series expansions. Expanding $u(x+h)$ and $u(x-h)$ about x and subtracting, gives

$$\frac{1}{2}\,[u(x+h) - u(x-h)] \;=\; hu' \;+\; \frac{h^3}{3!}u^{(3)} \;+\; hot$$

so the leading error term is of third order in h, much worse than the Numerov algorithm itself which is of sixth order. A better approximation can be derived in a way analogous to the derivation of the Numerov algorithm. An expression is developed for the leading error term, $\frac{h^3}{3!}u^{(3)}$, by Taylor expanding the second derivatives and subtracting

$$\frac{1}{2}\,[u^{(2)}(x+h) - u^{(2)}(x-h)] \;=\; hu^{(3)} \;+\; \frac{h^3}{3!}u^{(5)} \;.$$

Using the Schrödinger equation to eliminate the second derivatives and multiplying with $h^2/3!$ gives a formula for the error term

$$\frac{h^3}{3!}\;u^{(3)} \;=\; \frac{h^2}{12}\;\left(f(x+h)u(x+h) \;-\; f(x-h)u(x-h)\right)$$

which is good to fifth order, nearly as good as the Numerov algorithm itself. The final expression for the first derivative is

$$u'(x) \;=\; \frac{h}{24}\;\left(\left[\frac{12}{h^2} - f(x+h)\right]u(x+h) - \left[\frac{12}{h^2} - f(x-h)\right]u(x-h)\right) \;.$$