Solving the Time-Independent Schrödinger Equation: Instructor Notes[*]

David G. Robertson[†]

Department of Physics and Astronomy

Otterbein College, Westerville, OH 43081

(Dated: December 14, 2009)

Abstract

Some notes on the simulations are presented, along with solutions to the exercises. Please contact the author if you have any comments or suggestions.

[*] Work supported by the National Science Foundation under grant CCLI DUE 0618252.

[†]Electronic address: `drobertson@otterbein.edu`

Contents

I. GOALS AND TIME NEEDED

This module is an introduction to the "shooting" technique for solving boundary value problems in one dimension, with application to the time-independent Schrödinger equation. Its principal goals are:

- to provide an opportunity to develop insight into simple quantum systems; and
- to familiarize students with basic concepts involved in the numerical solution of boundary value problems.

In my experience, working through these kinds of exercises is a great way to develop intuition about how quantum mechanics works. The shooting method itself provides a nice way to see why bound states generically have quantized energy levels, for example, and the ability to generate solutions quickly for arbitrary potentials allows one to develop a real feel for how wavefunctions behave. A number of interesting and common numerical issues also arise, including integration techniques for ordinary differential equations, root finding algorithms, and numerical quadrature.

The minimal physics background required includes a basic exposure to wavefunctions and the time-independent Schrödinger equation, at the level of a typical sophomore-level course on modern physics. However, more advanced students will be able to do more with the exercises, and the experience can be correspondingly richer for them.

My estimate is that one to two weeks of class time, assuming 3-4 hours per week in class and some work outside of it, should be sufficient to explore the basic numerical issues and develop codes to solve the time-independent Schrödinger equation, at least for symmetric potentials (where the wavefunctions are parity eigenstates). This depends somewhat on the level of detail pursued on the numerical side. With a minimal approach to the numerics, e.g., using the simple Euler method for integration, the module can likely be done in a week. If the instructor wishes to explore the material in the appendix of the Student Manual, then another week will probably be required.

II. NOTES ON THE SIMULATION PROJECTS

1. **Infinite Square Well**

   Solving this problem will get the students through most of the technical difficulties, making the other projects relatively straightforward. It is also a simple problem analytically and the exact results are all known.

   My preference at the outset is to choose m so that $\hbar^2/2m = 4/\pi^2$; this makes the exact eigenvalues very simple: $E_n = n^2$. After the code is working, students can recast the problem in more physical units, e.g., with an electron and using eV and Å (or nm). Then do all the remaining exercises in these units.

   In this regard it is helpful to keep in mind that $\hbar c \approx 1973$ eV Å and $m_e c^2 = 0.511 \times 10^6$ eV. Thus, for example,

$$\frac{\hbar^2}{2m_e} = \frac{(\hbar c)^2}{2m_e c^2} = \frac{(1987)^2}{2 \times 0.511 \times 10^6} \approx 3.86 \tag{2.1}$$

   in these units.

   My experience is that the the part of the code that brackets E will be the most confusing for the students. The boundary condition in this case is very simple: just that ψ vanish at the edge of the well. They will need to keep track of the last value of ψ at the edge (or at least its sign), then compare the new value to the last. If the sign has changed, go back to the previous energy and halve the step size. If not, the new value becomes the old value and the energy is incremented.

   The sample codes all use the 4th-order Runge-Kutta integration, which is the most accurate one presented in the module. With this algorithm rather modest step sizes may be used, say 0.01. If Euler's method is used the step size will need to be significantly smaller to get accurate results.

2. **Finite Square Well**

   This problem is really the heart of the module.

   The new wrinkle is the need to integrate past the edge of the well. With a wrong eigenvalue the solution will eventually diverge towards either plus or minus infinity. The same bracketing logic applies, but we now test the value of ψ at some x_{max} beyond the edge of the well. In fact, the code should integrate until x_{max} is reached or the absolute value of ψ grows beyond some reasonable value, say 5.0. When this happens it is clear that the solution is diverging, and additional integration is a waste.

   Students will need to integrate far enough into the classically forbidden region, however, and they should spend some time thinking about just how far "far enough" is. You can start them off by noting that in ths region the true wavefunction decays as $\exp(-\kappa x)$ where κ depends on E. So they can estimate the penetration depth and then set the integration to go comfortably past this (a few more e-foldings, say). They should beware that these considerations change as they approach the top of the well; for weakly-bound states, the wavefunctions can extend significantly into the classically forbidden region.

   Results for the eigenvalues can be compared to those obtained in the standard analytical calculation, although this involves solving numerically the transcendental equations that arise. Doing this for an electron in a well 2 nm wide and 1 eV deep, I find four bound state eigenvalues:

   $$E_1 = 0.0663 \text{ eV}, \quad E_2 = 0.262 \text{ eV}, \quad E_3 = 0.574 \text{ eV}, \quad E_4 = 0.946 \text{ eV}. \tag{2.2}$$

3. **Other Potentials**

   (a) **Harmonic Oscillator**

   This problem holds few surprises once the finite well has been mastered. Again, students will want to make sure they are integrating far enough into the forbidden region. In this case they might have their code determine the classical turning points for the assumed E, then integrate some reasonable distance past this.

   (b) **Anharmonic Oscillator**

A nice exercise is to ask the students to predict how the energies will change if the coefficient of the anharmonic term is increased (or decreased).

(c) **Linear Potential**

Students can examine in this case how the (de Broglie) wavelength of the particle changes across the well.

4. **Well with Perturbation**

   This is a good exercise if the students have seen stationary-state perturbation theory – they can compute the eigenvalues numerically and compare to the perturbation results. Otherwise it is just another potential to solve. Again, you can ask the students how the energies will change with the addition of the bump. A subtle point is that the odd solutions are affected much less than the even ones, because the unperturbed wavefunctions are anyway passing through zero in the neighborhood of the bump.

5. **Orthogonality**

   No surprises here.

6. **Model for Covalent Bonding**

   This is a very crude model of molecular binding, but it does show some of the relevant features of the real problem. In my experience students find this quite illuminating, perhaps more so than it deserves. It is based on material in ref. [1].

   For a well width of 1.03 Å, I find a depth of 18.72 eV gives the desired ground state energy. The first excited state (odd parity) is then at -5.22 eV. The proton-proton contribution to the energy is

$$V_{pp} \approx \left(\frac{e^2}{4\pi\epsilon_0\hbar c}\right)\frac{\hbar c}{B},$$

   where the (dimensionless) factor in parentheses is known as the "fine structure constant" and has the value 1/137. Hence

$$V_{pp} \approx \frac{1973}{137 \times 1.2}\ \text{eV} = 12\ \text{eV}.$$

   Adding this to the electron energies calculated above gives the total energy of the system.

The result is that the ground state configuration of the electron is bound (has negative total energy), while the first excited state has positive total energy. If our model included electromagnetic interactions, it would break apart and become a separate H atom and proton; this is the configuration with minimum energy.

The wavefunctions tell the basic tale. In the ground state, the electron has significant probability to be found between the protons, hence it can hold them together via the Coulomb force. The electron is in some sense "shared" between the protons. In the unbound state, on the other hand, the wavefunction vanishes at the center of the bump and the electron has much lower probability to be found between the protons compared to the ground state. It thus fails to bind the protons together.

III. NOTES ON THE ODE EXERCISES

1. *Consider the equation*

$$\frac{dy}{dx} = -xy$$

with initial condition $y(0) = 1$*, which has the exact solution* $y = \exp(-x^2/2)$*. Study the numerical integration of this using the methods described above. In particular, verify that the errors (difference between numerical and exact solutions) decrease according to the expected power of* ϵ*.*

A good approach here is to evaluate the solution at a fixed endpoint, say, $x = 1$, and vary the stepsize ϵ. The error should obey a power law in ϵ,

$$\text{error} \propto \epsilon^n,$$

so a plot of log(error) versus $\log(\epsilon)$ should give a line of slope n. Results for Euler's method and the second order RK algorithm are shown in fig. 1. The calculated slopes are very close to the expected values $n = 1$ and $n = 2$, respectively.

2. *Generalize one or more of the schemes presented here to solve a system of two coupled ODEs, and apply it to solve Newton's second law for the simple harmonic oscillator*

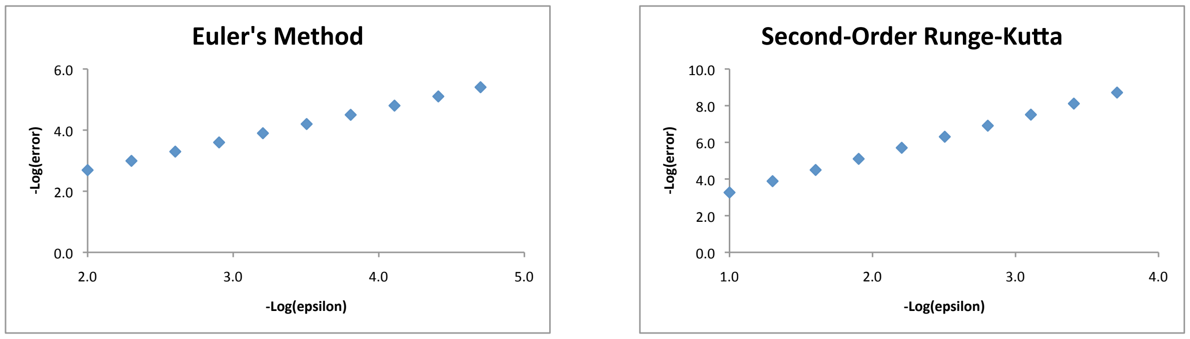

FIG. 1: Error scaling for Euler's method and the second order Runge-Kutta algorithm.

(with $m = 1$):

$$\begin{aligned} \frac{dx}{dt} &= v \\ \frac{dv}{dt} &= -\omega^2 x. \end{aligned}$$

A simple criterion for accuracy is the degree to which a known integral of the motion, for example, the energy, is conserved in the evolution. Study the constancy of $2E = v^2 + \omega^2 x^2$ for the various algorithms and choices for ϵ.

As an example, here is the generalization of the second-order Runge-Kutta algorithm for a pair of ODEs involving two functions $y_1(x)$ and $y_2(x)$. We define

$$\begin{aligned} k_{11} &= \epsilon f_1(x, y_1, y_2) \\ k_{12} &= \epsilon f_2(x, y_1, y_2) \end{aligned}$$

and

$$\begin{aligned} k_{21} &= \epsilon f_1(x + \epsilon/2, y_1 + k_{11}/2, y_2 + k_{12}/2) \\ k_{22} &= \epsilon f_2(x + \epsilon/2, y_1 + k_{11}/2, y_2 + k_{12}/2), \end{aligned}$$

that is, separate ks for each of the functions. Then

$$\begin{aligned} y_1(x + \epsilon) &= y_1(x) + k_{21} \\ y_2(x + \epsilon) &= y_2(x) + k_{22}, \end{aligned}$$

correct to $\mathcal{O}(\epsilon^3)$. The fourth-order algorithm generalizes in an analogous way.

Another measure of the error that can be used in this case is the difference between the starting value of x and the value of x at times $t_n = nT$ with $T = 2\pi/\omega$ and $n = 1, 2, 3, \ldots$

3. *Another common test of accuracy is to integrate backwards to the original starting point, using the ending x, y as the initial condition. The difference between the resulting value for y and the original initial condition then gives a measure of the overall accuracy of the result. Apply this test to the example from problem 1, for the various algorithms discussed above.*

   This criterion can be used as a measure of the error when the exact solution is not already known.

4. *Study the approximation given in eq. (3.19), by using it to solve the equation*

$$\frac{dy}{dx} = -y,$$

   with the initial condition $y(0) = 1$. This equation has the exact solution

$$y = e^{-x},$$

   of course. To start the calculation off we need both $y(0)$ and $y(\epsilon)$; you can use eq. (3.15) to obtain $y(\epsilon)$. Eq. (3.19) can then be used to generate the solution for other values of x. Does this give a reasonable approximation to the exact answer? If not, can you determine why?

   The result of this calculation should look something like fig. 2. An oscillation eventually develops, with a growing amplitude, that dominates the exponentially decreasing solution we want.

   The reason for this can be traced to the recursion relation (A19). If we assume a solution of the form

$$y(n\epsilon) = Az^n, \tag{3.1}$$

   with A and z constants, then plugging this into eq. (A19) gives an equation for z:

$$z^2 + 2\epsilon z - 1 = 0. \tag{3.2}$$

   This has solutions

$$\begin{aligned} z_+ &= \sqrt{1+\epsilon^2} - \epsilon \approx 1 - \epsilon & (3.3) \\ z_- &= -\sqrt{1+\epsilon^2} - \epsilon \approx -(1+\epsilon), & (3.4) \end{aligned}$$

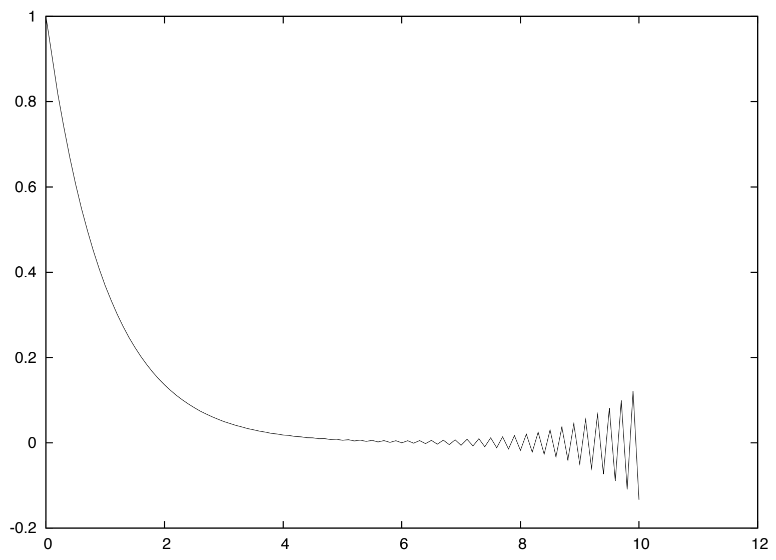

FIG. 2: Solution for $y(x)$ versus x using the "bad" second-order algorithm described in the text (with $\epsilon = 0.1$). The onset of the unwanted oscillating solution can be seen around $x = 5$.

where the approximate forms are obtained for small ϵ. The solution z_+ is positive and less than one, and so decreases from step to step; this is the exponentially decreasing solution we want. The solution z_-, on the other hand, is negative with magnitude greater than one; it corresponds to a spurious solution

$$y \propto (z_-)^n = (-1)^n (1 + \epsilon)^n. \tag{3.5}$$

It therefore grows in size with increasing n (because $1 + \epsilon > 1$) and alternates in sign from step to step.

Now the point is that the general solution to the difference equation (A19) is a linear combination of the two solutions involving $z_\pm$. Even if we can somehow arrange that the initial values $y(0)$ and $y(\epsilon)$ match only the desired (z_+) solution, roundoff errors will inevitably introduce a small admixture of the unwanted solution as we proceed. Since this solution naturally grows with n, it is guaranteed to eventually dominate the results.

The moral here is that the apparent accuracy of an algorithm is not the only important consideration – stability is vital as well. The stability and accuracy of integration algorithms is a large and technical subject, one to which I cannot hope to do justice here. Readers should be aware that these issues exist, and if further details are needed a reference on numerical analysis should be consulted.

IV. SAMPLE PROGRAMS

A set of sample programs, written in C, is distributed with the module. Typically they take some command line arguments, with other parameters hard-wired via `#define` statements. Comments in the programs explain the arguments.

Programs related to the ODE exercises are:

1. `euler1.c`

   Solves the equation from exercise A.4.1 using Euler's method.

2. `rk1.c`

   Solves the equation from exercise A.4.1 using the second order Runge-Kutta algorithm.

3. `rk2.c`

   Solves the equation from exercise A.4.2 (harmonic oscillator) using the second order Runge-Kutta algorithm generalized to two coupled equations.

4. `bad4.c`

   Solves the equation from exercise A.4.4 using the "bad" algorithm from the text (eq. (A19)).

Programs related to the physics projects are:

5. `ISW-euler.c`

   Solves the TISE for the infinite square well using Euler's method.

6. `ISW-rk4.c`

   Solves the TISE for the infinite square well using the fourth-order Runge-Kutta algorithm.

7. `FSW-rk4.c`

   Solves the TISE for the finite square well using the fourth-order Runge-Kutta algorithm. Integration from the center of the well towards the edge.

8. `FSW-rk4.c`

   Solves the TISE for the finite square well using the fourth-order Runge-Kutta algorithm. Integration from outside the well to the center, with an estimate for the starting derivative ψ'.

9. `H2mol.c`

   Solves the TISE for the double square well used in the model of H_2^+.

---

[1] R.D. Knight, *Physics for Scientists and Engineers* (Addison Wesley, 2004).