Quantum Mechanics with MATLAB

Göran Lindblad[1]

Department of Physics
Royal Institute of Technology
SE-100 44 Stockholm, Sweden

April 18, 2001

[1]E-mail: `gli@theophys.kth.se`

Abstract

This manual introduces `schrodinger`, a set of simple MATLAB programs which give interactive solutions of some basic quantum mechanics problems. Most of these problems should be familiar to any student taking a first quantum mechanics course. The programs solves them numerically with great speed and reasonable accuracy, and gives a graphic display of the solutions. They are based on simple algorithms for numerical integration of the Schrödinger equation or numerical evaluation of special functions. Wherever possible they use the standard algorithms of MATLAB, but some non-standard ones are included which are specially adapted to the present set of problems. The files are available on the department web-server. The `schrodinger` toolbox has the home page:

`http://www.theophys.kth.se/mathphys/schrodinger.html`

You are welcome to use the files under the standard conditions of acknowledging the source.

Contents

Chapter 1

Introduction

1.1 Motivation

The examples set in a standard basic course in quantum mechanics are still essentially of the type found in the pre-computer era, like htos in the collection Flügge [10]. In order to make them solvable (using only Jahnke–Emde [16], pencil, slide rule and some midnight oil) they had to be carefully selected. Models solvable by such means are as rare and precious as gold dust, and the known ones figure in most of the classical collections.

The personal computer has made it possible to do problems selected on different principles. There are a number of commercial programs which calculate and display the solutions of various quantum problems with a reasonable turn of speed. One well-known example is the books and software by Brandt and Dahmen [4]. There now more ambitious combinations of physics texts and specially developed software; we cannot review theme here. The drawback of such proprietary software is that the teacher/student may find it difficult to modify them to solve problems other than those foreseen by the author. It seems more useful to base the problem solving on the kind of software which the student/professional will use for his own calculations, the modern replacements of the slide rule. There are now a great number of books using one or more of the standard commercial mathematics software packages.

Symbolic mathematics programs like Mathematica, Maple and MuPad have several standard or share library programs which can be used for quantum calculations. They contain routines for most of the special functions which can enter into quantum problems. They have some drawbacks, however. The symbolic calculations are helpful for the limited set of problems which are solvable by algebraic methods, and the speed of computation is limited. For the purpose of a graphical display of the solutions the symbolic calculations are not necessarily optimal, and certainly not the fastest way of doing things.

If we want solutions to be generated on the time scale of a few seconds on a personal computer, and want the ability to solve a larger class of problems, then numerical calculations are preferable. The symbolic programs mentioned above also have numerical capability. There are also the mainly numerical program MATLAB (and the related freeware Octave and SciLab). The matrix algorithms allows us do calculations with a reasonable speed for interactive solutions; the user can experiment with many different settings for the parameters of the system. MATLAB is easily programmable and crossplatform compatible. A most important aspect is the availability of the software; this institute has a site licence for student use of MATLAB on all platforms.

For the more professional aspects of scientific computing there are a great number of FORTRAN programs, for a book based on these, see e.g. Schmid et al [26].

1.2 The goals of `schrodinger`

- The aim is to illustrate some of the basics of quantum mechanics. This is neither a textbook in quantum mechanics. nor a professional guide in programming; the numerical methods used here are very basic.
- The programs should be fast and simple enough to be run on PCs and Macs in an interactive mode. Preferably they should avoid the use of large amounts of memory, but this is not always fulfilled in the present version.
- The files should be easy to modify for the user. For this purpose they contain only a few standard algorithms and some comments on their construction.
- It is possible to change the number of integration points and other settings in the M-files, to adapt to the computer power available. The accuracy should be god enough to allow the calculation of interesting examples.

1.3 Getting started

1. Get the MATLAB files from our Web server, starting from the home page `http://www.theophys.kth.se/mathphys/schrodinger.html`.
2. Decompress ome of the archives `schrod5.tar.gz, schrod5.zip`. Put the files where your MATLAB application can find them. You should check beforehand that there is no overlap between the names of the files in this set and those already in your MATLAB directory. The present (January 1999) version of the files have been made on a PC running Linux. They have also been checked on a Powermac 7600/120.
3. Start MATLAB and type `start`. The main menu comes up. You choose the problems by clicking buttons, and inserting some optional values in the editing boxes. There are default values which are meant to give some interesting output. In order to understand which range of values will give interesting results it is often enough to know the uncertainty relations. Running the programs without overtaxing your computer may require some experimenting.
4. In order to understand how the programs work one should read the MATLAB files. In order to enter values and functions via the text editing boxes it is necessary to understand something of the MATLAB syntax. Note the necessity to use the element-wise operations when the arguments are vectors or matrices, e.g. `.*` for the product, `.^` for the exponent.
5. You are welcome to report on errors, bugs and possible improvements.

1.4 Summary of notation

In the following we use the standard simplifications in the notation: we set $\hbar = 1$, particle mass $= 1$. We also use the notation D $(= \nabla)$ for differentiation with respect to the space coordinate. The time-independent Schrödinger equation then reads

$$(D^2u)(x) = 2[V(x) - E]u(x) \tag{1.1}$$

The time-dependent Schrödinger equation reads

$$i\frac{d}{dt}\psi(x,t) = -\frac{1}{2}D^2\psi(x,t) + V(x)\psi(x,t) \tag{1.2}$$

We will also use some convenient abbreviations:
BC = boundary condition. CON = complete orthonormal. DE = differential equation.

1.5 Summary of problem types

A list follows defining the types of problems which are included in main menu of the present version of `schrodinger`. The choice of problems is via clicking buttons. Some of the calculations involve the choice of an "arbitrary" potential, from a menu or by editing a MATLAB function file. In most of them you have to enter parameter values. In some cases you have to enter a function of a variable X, using the MATLAB syntax. You can construct as many new examples as you want by editing the relevant M-files where the potentials are defined.

1. **Integration of the Schrödinger equation**. A demo of the problems of numerical integration and of finding energy eigenvalues and eigenfunctions.

2. **Particle in a box**. Finding the eigenvalues and eigenfunctions of a particle on a finite interval with a choice of boundary conditions. Two methods are:
   (a) matrix approximation using a finite subset of a CON basis of functions satisfying the BCs.
   (b) numerical integration of the DE for a lattice of energy values and finding the energy eigenvalues as the values where the BCs at both ends of the interval are satisfied.

3. **Bound states in 1D and 3D**. Finding bound states for a particle on an infinite (or half-infinite) interval with a potential essentially zero outside a finite range, and the same problem for a 3D central potential. Here one can again use a discretization method or integration of the DE. For piecewise constant or linear potentials one can use transfer matrix methods.

4. **Evolution of wave packets in 1D** . Finding the evolution of a wave packet, again by two methods:
   (a) an expansion of the initial state in eigenstates (including plane waves in the scattering case).
   (b) an iterative solution of the time-dependent Schrödinger equation.

5. **Scattering and periodic systems**. Calculation of transmission and coefficients in 1D for a potential which is non-zero in a finite interval by integration of the DE. For piececonstant or linear potentials one can use transfer matrix methods. One can also find the Bloch solutions for a periodic potential, and hence the allowed and forbidden bands. For 2D and 3D system we can do a partial wave analysis for a central potential, finding the differential and total cross sections.

6. **Special functions and expansions**. We can calculate Legendre and Bessel functions, and standard types of orthogonal polynomials, hence find the eigenfunctions of the harmonic oscillator and hydrogen atom, etc. We can also easily find the numerical expansion of arbitrary functions in a CON basis of such eigenfunctions.

7. **Hydrogen atom eigenfunctions**. Graphical display of atomic orbitals, spherical harmonics, radial eigenfunctions.

1.6 Summary of techniques

1. Numerical integration of the time-independent Schrödinger equation in 1D using the Numerov algorithm. This is a constant-step algorithm particularly adapted to a second order DE without first-order term. MATLAB lets us solve for a vector-valued solution, say for a vector of different energy values. We can use this both for eigenvalue problems and scattering problems. The boundary values then impose conditions on the solution vector which can then be solved numerically for the energy eigenvalues or for the scattering amplitude. This method is used also for the radial part of a 3D central potential problem.

2. Schrödinger-type eigenvalue problems in 1D with suitable boundary conditions can often be approximated by discretized versions, replacing the Hamiltonian by a finite Hermitian matrix. The algorithms of MATLAB makes the solution of the matrix problem for eigenvalues and eigenvectors a rather painless procedure for matrices of dimension 100×100 or more.

3. For 1D potentials which are piecewise constant or linear it it possible to use the transfer matrix method as a very rapid method to find eigenvalues and scattering amplitudes.

4. The evolution of wave packets may be calculated through an expansion of the initial wave packet in eigenfunctions of the Hamiltonian. For general potentials this method will easily lead to lengthy computations of a sufficient number of eigenstates. A useful alternative is the direct solution of the time-dependent Schrödinger equation by iteration.

5. Problems involving special functions are often solved rapidly by iterative solution for a finite number of polynomial coefficients or through the standard algorithms included in MATLAB.

Chapter 2

Numerical integration

The problem in this section is the numerical integration of the DEs (1.1) and (1.2). The techniques for dealing with the time-dependent and time-independent cases in numerical calculations are different, but both face some non-trivial stability problems.

2.1 Fundamental solutions and boundary conditions

Recall that in the absence of BCs there are two linearly independent solutions of the DE (1.1) for every value of E. If we solve the DE in a finite interval $[a, b]$ we can prescribe the values of $\{u(a), Du(a)\}$ as initial values for an integration from a to b. We can express any solution as a linear combination of the two fundamental solutions $\{u(x), v(x)\}$ defined by

$$\begin{aligned} u(a) &= 1, \qquad & Du(a) &= 0, \\ v(a) &= 0, \qquad & Dv(a) &= 1 \end{aligned}$$

We note that for any solution of (1.1) the complex conjugate function is also a solution of the same energy (due to the reality of the potential), hence the real and imaginary parts are also solutions. Clearly $\{u(x), v(x)\}$ are real.

From the two fundamental solutions associated with a given energy E and any given point a we can construct the family of transfer matrices

$$M(a, x) = \begin{pmatrix} u(x) & Du(x) \\ v(x) & Dv(x) \end{pmatrix}$$

If we do this for all pairs of points (x, y) we obtain a 2-parameter family $M(x, y)$ which satisfies $M(x, x) = \mathbb{1}$ and has the following group property under matrix multiplication

$$M(x, y) \cdot M(y, z) = M(x, z), \quad M(y, x) = M(x, y)^{-1}. \tag{2.1}$$

Of course, M also depends on the energy E.

The Wronski determinant of any two solutions $\{y_1, y_2\}$ of energies $\{E_1, E_2\}$

$$W(y_1, y_2)(x) = y_1(x)\, Dy_2(x) - Dy_1(x)\, y_2(x) \tag{2.2}$$

satisfies the differential equation

$$DW(y_1, y_2) = y_1\,(D^2\,y_2) - (D^2\,y_1)\,y_2 = (E_1 - E_2)y_1\,y_2. \tag{2.3}$$

From this we conclude that the Wronskian of any two solutions of the same energy is constant. For the two fundamental solutions the Wronskian is the determinant of the transfer matrix, and it is equal to 1; the transfer matrices $M(x, y)$ are all unimodular.

Integration of the DE (2.3) gives

$$(E_1 - E_2) \int_a^b dx\, y_1(x)\, y_2(x) = W(y_1, y_2)(a) - W(y_1, y_2)(b). \tag{2.4}$$

If we let $|a - b| \to 0$ we see that the Wronskian is continuous in any point. Solutions in two adjoining intervals are spliced by identifying the logarithmic derivatives in the common boundary point, in fact this continuation is implicit in the formula (2.1). When the logarithmic derivatives are the same, then the Wronskian (2.2) of the two solutions will be zero. However, one has to be alert to the possibility that the two solutions can have a common node in the boundary point, a fact which will create problems in some algorithms.

It is not difficult to check that the condition that (2.4) is zero is also the condition for the Schrödinger Hamiltonian on the interval $[a, b]$ bo be self-adjoint (Hermitian). The RHS of (2.4) will be zero for a suitable choice of BCs in a and b. The standard BCs are of the following general kind:

$$\alpha\, y(a) + \beta\, Dy(a) = 0, \quad \gamma\, y(b) + \delta\, Dy(b) = 0, \tag{2.5}$$

for real constants $\alpha, \beta, \gamma, \delta$. It is sometimes convenient to accept complex solutions and replace y_1 by y_1^* in (2.4), in particular when considering scattering solutions. The quantity (2.4) is then zero if we impose the periodic ($K = 0$) or Bloch type ($K \in \mathbb{R}$) BCs:

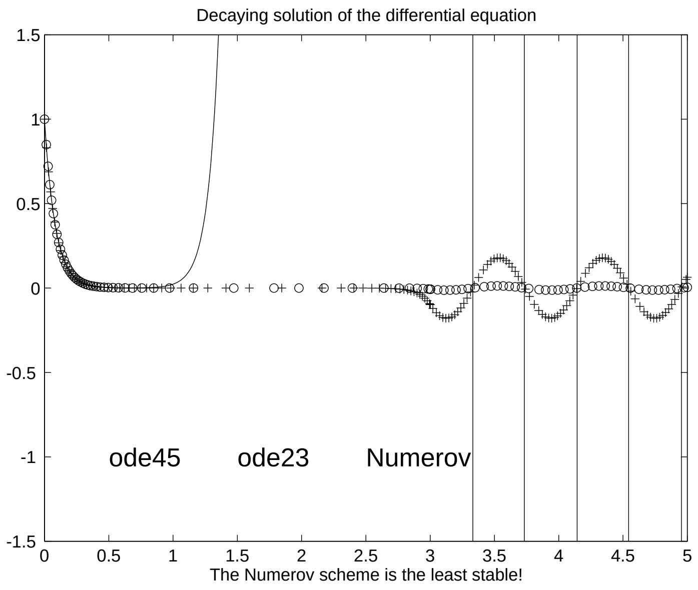
$$y(b) = \exp(iK)y(a), \quad Dy(b) = \exp(iK)Dy(a). \tag{2.6}$$

2.2 The stability problem

In the "classically allowed" regions where $V(x) \leq E$ the two linearly independent solutions are both oscillating and bounded (except in points where the potential is singular). In the "classically forbidden" regions where $V(x) \geq E$ the solutions have an exponentially diverging behavior. One solution is (asymptotically) exponentially increasing for increasing x, the other for decreasing x. This fact is crucial for the problem of numerical instability.

When integrating the DE in such a region the exponentially diverging solution must eventually take over and dominate the amplitude. How fast this happens depends both on the values of $V(x) - E$ and on the algorithm used.

This point is illustrated by the file `integrat.m`, where the left hand BC is chosen such to select the exponentially decaying solution. The exponentially decaying solution is eventually dominated by the divergent one introduced by the finite accuracy. Here the algorithms are: `ode45`$= +$, `ode23` $= \circ$ `numerov` $= —$. There are other integration algorithms in MATLAB which are better adapted to "stiff" DEs, they will need more time for the calculation.



When the forbidden region extends to $+\infty$ and/or $-\infty$, the interesting solutions will necessarily decay to zero at infinity. These solutions are relevant for bound states, where they must decay in both directions. The two boundary conditions will be satisfied only for a discrete set of energy values, the bound state energies.

The decaying solutions are also part of scattering states. Of course, there must be an allowed region extending to infinity in the opposite direction, and in which the solution stays finite.

It is not good idea to find the exponentially decaying solution by integration starting from an allowed region and finding a linear combination which goes to zero in the forbidden region. Instead, the stable procedure is to start from the forbidden region and integrate towards the allowed region. It is often possible to find the asymptotic decay properties of the exact solution, and use these for initial data for the numerical integration in the opposite direction. In this way one picks out the solution which is exponentially diverging in the direction of integration as the physically relevant one. This point is also illustrated by the file `integrat.m`. If there are several allowed regions separated by forbidden regions one can make the integration piecewise, from each maximum of the potential to the previous and the next minimum.

2.3 The Numerov algorithm

There are standard integration algorithms in MATLAB, like `ode23` and `ode45`. They are variable step algorithms which have good stability properties. In the recent versions of MATLAB there are other algoritms which handle instability ("stiffness") even better. However, all these algorithms which have a variable integration step have the drawback they do not allow us to use the matrix capabilities of MATLAB to the full.

A much used integration method for second order DEs without first-order term is the Numerov scheme. It is a 4-th order accurate constant step algorithm (hence allows matrix calculations) with good accuracy and speed [14, 17]. It is preferred over the standard algorithms for our purpose of doing the integration simultaneously for a lattice of energy parameters and graphical display of the results, all with a speed which allows an interactive mode of operation.

If the space lattice is denoted $x_n, n = 1, \ldots, N$, the lattice constant $dx = h$ and the values of the wave function are $u(x_n) = u_n$, then the two-term recurrence relation corresponding to the DE (1.1) reads

$$\begin{aligned}
&(1 - T_{n+1})u_{n+1} - (2 + 10T_n)u_n + (1 - T_{n-1})u_{n-1} = 0 \\
&T_n := -h^2(E - V(x_n))/6
\end{aligned}$$

In the version used in the present set of files we have used the notation

$$\begin{aligned}
F_n &= (1 - T_n)u_n \\
U_n &= (1 - T_n)^{-1}(2 + 10T_n)
\end{aligned}$$

to obtain the three-term recurrence relation

$$F_{n+1} - U_n F_n + F_{n-1} = 0$$

It is possible to reduce this to a two-term relation, but this is not used in these files at present. It is also necessary to introduce the boundary conditions, i.e. defining the starting values F_1, F_2 from given values of $[u(x_1), Du(x_1)]$, and $[u(x_N), Du(x_N)]$ from F_{N-1}, F_N. How this is done can be seen in the file `numerov1.m`.

Because the spacing of the set of points $\{x_n\}$ does not depend on the integrand the iterations can be done for a whole set (e.g. for a lattice) of values for E. This makes T into a matrix which can be calculated before starting the iteration procedure. The drawback with this method is the necessity of calculating and keeping a large matrix which will use up a lot of memory. A slower alternative with a smaller demand on memory is to calculate the vectors (T_n, U_n) at each step in the iteration.

If we want the solution only for finding the energy eigenvalues (or the transfer matrix or the transmission coefficient in a scattering problem) then it is not necessary to save the intermediate values of F_n. Needed are only the values of the function and derivative in the end point of the integration interval. This is done in the function file `numerov1.m`, while the file `numerov3.m` does the same thing with a modified algorithm and `transfer.m` finds the transfer matrix. When the energy eigenvalues have been found we can calculate the corresponding eigenfunctions using `numerov2.m`.

References Johnson [17], Schmid et al [26] (who call it the Fox-Goodwin method),

2.4 The time-dependent equation

The solution of the time-dependent Schrödinger equation (1.2) starting from a given wave function at $t = 0$ can be approximated by an iterative scheme of numerical integration which involves discretization of space and time coordinates. In this method the time derivative is replaced by a first order and the Laplace operator $\Delta = D^2$ by a second order differential approximation. For a finite step size dx:

$$D^2\psi(x,t) \approx dx^{-2}[\psi(x + dx) + \psi(x - dx) - 2\psi(x)]$$

With $H = -D^2/2 - V$ and a finite time step dt we can write the iteration as

$$\psi(t+dt) = \psi(t) + iH\psi(t)$$

However, the resulting algorithm is very badly behaved, being unstable for all choices of step sizes (dx, dt).

One standard method of constructing a stable algorithm is the Crank-Nicholson scheme. We will not go into the details here, as this is fully described in many places, see e.g. [11, 24]. The stability and accuracy is very good, but the speed can be a problem. In each step of the iteration a large system of linear equations has to be solved (this is a so called implicit method). MATLAB has built in algorithms solving such systems of linear equations, but the time per iteration step is still a bit too long when you want an interactive mode of operation.

Recently Visscher [27] described a simple explicit algorithm (i.e. one without implicit equation-solving steps). This algorithm is very fast, second-order accurate and conserves the total probability. It is also stable provided that the time step is chosen short enough.

The basic structure of this scheme is as follows. We use the real and imaginary parts of the wave function: $\psi = R + iI$ and define R at times $0, dt, 2dt, \ldots$ and I at times $0.5dt, 1.5dt, \ldots$, which transforms the iteration to the pair

$$\begin{aligned} R(t+0.5dt) &= R(t-0.5dt) + HI(t)dt \\ I(t+0.5dt) &= I(t-0.5dt) - HR(t)dt \end{aligned}$$

We need two initial functions $\{R(x,0), I(x,0.5dt)\}$ to start the iteration. The initial wave function $\psi(x,0)$ determines $R(x,0)$ directly, the value $I(x,0.5dt)$ can be found by other methods (in `evolut.m` by the Crank-Nicholson scheme). The displayed quantity is the probability density. In order to have the total probability conserved in time it is expressed in terms of R and I as follows

$$P(x,t) = R(x,t)^2 + I(x,t+0.5dt)I(x,t-0.5dt)$$

for values of t which are integral multiples of dt.

The Visscher scheme is implemented in the file `evolut.m`, where a few examples are constructed. In order to keep down the demands on memory the sparse matrix technique is used. Still the demand on memory capacity would be exorbitant if the calculated solution is saved for all time steps. In this file the solution is displayed only for a subset of time steps, and the solution is not saved, only some average values.

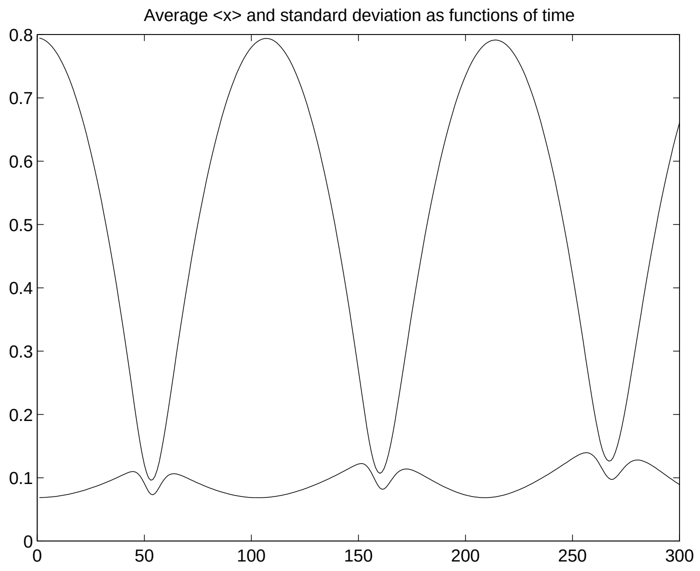
As the iteration technique demands a discrete set of points on a finite interval it is necessary to introduce the correct boundary conditions at the end point of the interval.

An alternative method for finding the time evolution is to calculate a number of the eigenvalues and eigenstates of the time-independent Schrödinger equation. We can then make a finite expansion of the initial state in these eigenfunctions. Evolving the amplitudes of the expansion using the calculated eigenvalues allows us to find an approximation to the wave function at any time.

No matter what method is used to evolve the wave packet, it is true that for most choices of the potential the wave packet will soon spread out and reduce to a rather random sea of waves. There are more interesting things to look at, to be sure. Only in the cases where the spectrum is nearly harmonic will there be some interesting effects to be seen in the long term evolution.

For a harmonic potential we know that the evolution is strictly periodic, and in particular the coherent states evolve in a very classical fashion, retaining the form of the probability distribution. Another case which comes rather close is that of a bouncing particle. In 1D there is a linear potential $V(x) = kx$ with BC $u = 0$ at both end points. We can see this as a model of a quantum particle in a uniform gravitational field which bounces on the floor. The spectrum (coming from the Airy functions) is nearly harmonic and we can see how the initial wave packet probability distribution revives periodically, with a superposed slow decay.

References Visscher [27]. Other algorithms, for instance the Crank-Nicholson are found in [11, 13, 24].



The bouncing ball as calculated by `evolut.m`, with a Gaussian wave packet and velocity $= 0$ at $t = 0$.

Chapter 3

Eigenvalues and eigenstates

We consider how the integration of the DE described above leads to eigenvalue equations and to scattering states.

3.1 Finding the eigenvalues

We can solve numerically for the lowest energy eigenvalues of the Schrödinger equation (1.1) using the numerical integration methods described above and the BCs (2.5) or (2.6). For instance, when integrating the DE over a finite interval $[a, b]$, we can start at $x = a$, using the BC there given by (2.5). At $x = b$ we obtain a relation

$$f(E) := \gamma\, y(b, E) + \delta\, Dy(b, E) = 0$$

where the energy dependence of the solution is indicated. This relation then holds precisely when the BCs are fulfilled at both ends of the integration interval; it defines a discrete series of energy eigenvalues. Thus we have to find approximate values for these zeros.

This is done in a rough and ready way by doing the integration, using the Numerov algorithm, for a lattice (a few hundred, say) of values of E. Thus we obtain a numerical function $f(E)$ as f function of E, and the intersections with the $y = 0$ axis gives the energy eigenvalues. The program `findzero.m` finds the zeros using spline interpolation rather than a slower but more accurate iterative calculation by e.g. Newton's method. The accuracy of the zeros found by `findzero.m` depends on the density of the lattice in the energy parameter, and on the distribution of the zeros. This means high demands on memory if we want to find large number of zeros in one single operation with high accuracy. For such applications one may have to divide the energy interval into pieces and scan each separately. For maximal accuracy an iterative method of finding the zeros will always be preferable.

Instead of a lattice in the energy variable it is sometimes better to use a lattice for a wave number variable like $k = \sqrt{2(E - V)}$ or $\kappa = \sqrt{2(V - E)}$ for some relevant value V of the potential. This gives a greater density of points at the end of the interval where $|E - V|$ is small.

The program `findzero.m` fails if the zeros are too close and sometimes when they are close to one end point of the interval. For finding and separating very closely spaced eigenvalues an iterative interactive method is available in `schr1.m` or `schr2.m`.

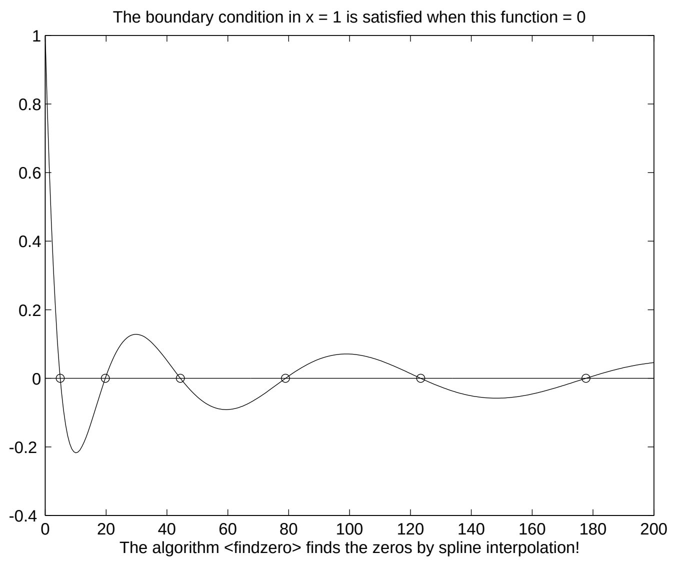
We can use the same method to find the bound state energies for Schrödinger particles on an infinite or semi-infinite interval if we can consider the potential to be constant (0) outside a finite interval $[a, b]$. The solutions outside this interval are then exponentially decaying at ∞, and we join the solutions continuously at $x = a$ and $x = b$. For instance, at $x = a$ the solution $y(x, E)$ in $[a, b]$. Setting the the BC for the integration of y ($E < 0$!)

$$\kappa \exp(\kappa a)\, y(a, E) - \exp(\kappa a)\, Dy(a, E) = 0, \quad \kappa := \sqrt{-2E}$$

(recall (2.2)!) we get at $x = b$ a function

$$f(E) = \kappa \exp(-\kappa b)\, y(b, E) - \exp(-\kappa b)\, Dy(b, E),$$

the zeros of which are the bound state energies.



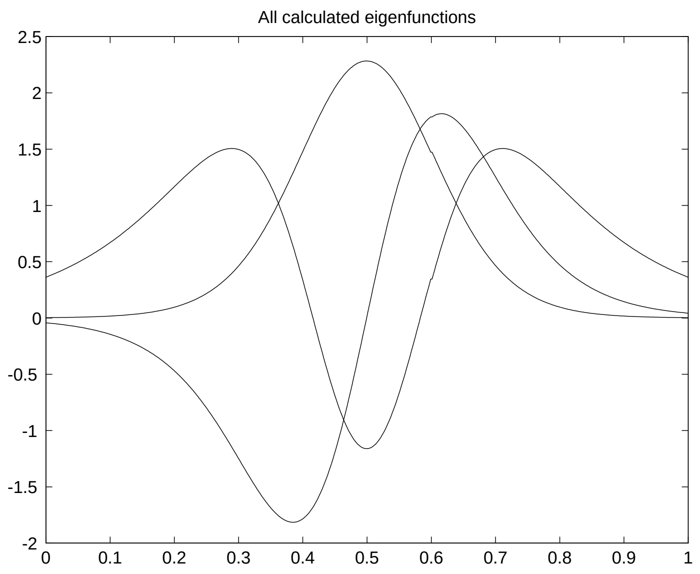
An example from `integrat.m`, the zeros are the energy eigenvalues.

The instability of the numerical integration means that large errors can creep in when the integration range contains intervals which are "classically forbidden". The errors are minimized if the integration is always performed from the forbidden into allowed regions, and the solutions joined in points in the allowed regions.

If the potential is piecewise constant there is no need to do the integration. For each subinterval $[a, b]$ with V constant, the two solutions are harmonic or exponential, and we can write down the transfer matrix $M(a, b)$ as a function of E, and then use (2.1) to obtain the transfer matrix for the whole integration range.

Once approximation for the eigenvalues are found we can do the integration again, for these energies only, to obtain the eigenfunctions. The instability of the integration procedure can show up in distorted functions, as shown in `integrat.m`.

References For "shooting" methods see e.g. [13] Chapter 10 or [24] Chapter 16.



Bound states in a 1D potential found by `schr2.m` using an interactive search for solutions of the BCs. These involve joining the solutions in the interval continuously to the exponentially decaying solutions outside.

3.2 Scattering and Bloch states

Consider a 1D scattering problem where the potential is assumed to be constant in both directions (not necessarily the same value) outside a finite interval. We then know the analytic forms of the incoming, reflected and transmitted waves, and need to find the energy dependence of the coefficients for the last two if the incoming is normalized to 1. The Schrödinger equation is integrated using the known form of the the transmitted wave at the relevant end point of the interval. Integration to the other end point allows us to match the solution to the incoming and reflected waves, identifying the amplitudes of these two components, finally renormalizing to unit incoming amplitude.

The Numerov algorithm permits us to do this integration for a finite lattice of energy (or wave number) values in a single process, displaying the transmission coefficient e.g. as a function of energy.

In fact it is convenient to calculate the whole transfer matrix, using `transfer.m`, from this one can easily obtain transmission and reflection coefficients. We note that (if $a = 0$) the plane wave states are related to our choice of fundamental solutions as follows

$$\exp(\pm ik_{r,l}x) = u(x) \pm ik_{r,l}\, v(x)$$

where $k_r(E), k_l(E)$ are the wave numbers of a given energy E to the right and to the left of the potential. If the incident plane wave is $\exp(-ik_r x)$ (coming in from the right) the transmitted amplitude is expressed in terms of a scalar product involving the transfer matrix M (where M is again from the left to the right end point of the interval containing the potential)

$$S = 2\left[(1, i/k_r) \cdot M \cdot \begin{pmatrix} 1 \\ -ik_l \end{pmatrix}\right]^{-1}$$

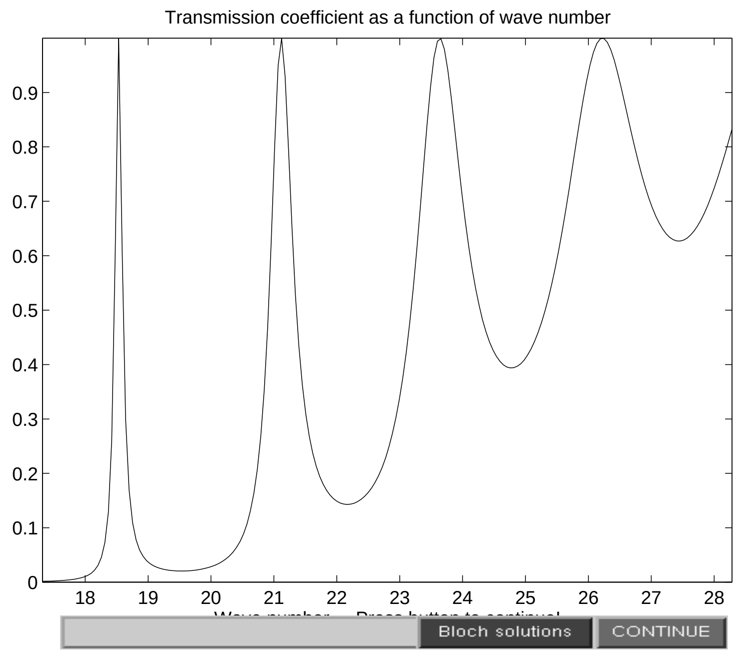
The transmission coefficient is then

$$T = \frac{k_l}{k_r}|S|^2$$

The Bloch solutions in a 1D periodic potential also come out from the transfer matrix $M(E)$ for the unit cell. We know that for a given energy E there are two eigenvalues of $M(E)$. If there is a Bloch solution with Bloch wave vector K they are $\exp(\pm iKa$ if a is the length of the unit cell. In this case $\operatorname{tr} M = 2\cos Ka$, otherwise we have $|\operatorname{tr} M| > 2$ and the energy is in a forbidden band. Conversely, if $|\operatorname{tr} M| \leq 2$ the value of K is defined up to sign (both signs give solutions as there is time reversal invariance) and up to a multiple of $2\pi/a$. In this way we can identify the allowed and forbidden bands, and the Bloch solutions.

Multiple barriers are easily handled with the transfer matrix technique, expecially when the potential is piecewise constant and the transfer matrix for a single such interval is calculated from the harmonic and exponential functions. In intervals with a linear potential we can use the Airy functions to find the transfer matrix.

References Cohen-Tannoudji [7] Complement N_{III}, Kroemer [18] Ch. 5, Messiah [22] Ch. 3.



Transmission resonances calculated using `transm.m`

3.3 The 3D case

The Schrödinger equation for a central potential $V(r)$ can be treated using the same algorithms as the 1D case. The equation separates in in spherical coordinates $(r, \theta, \phi) \equiv (r, \Omega)$: each energy eigenfunction of eigenvalue E is of the form

$$\psi(r, \Omega) = R(r) Y_{\ell,m}(\Omega)$$

where the radial function $u(r) = r\, R(r)$ satisfies

$$D^2 u(r) = \{2[V(r) - E] + \ell(\ell+1) r^{-2}\}\, u(r) \tag{3.1}$$

This radial DE is of the 1D standard form (1.1), just replace $V(x)$ there by

$$V(r)+\frac{\ell(\ell+1)}{2r^2}.$$

The singularity of this DE near $r=0$ makes it more difficult to get a good accuracy in the numerical integration; there are obviously established methods of dealing with this type of problem. In the present state of `schrodinger` such refinements have not been included.

If the potential $V(r)$ is very well behaved, being bounded below and going to zero fast enough when $r\to\infty$, we can use the known free solutions (defined by spherical Bessel functions) to get reasonable boundary conditions on the wave functions at two values of the radius. The inner point must be such that the r^{-2} contribution dominates over the non-constant part of $V(r)$, the outer point is one where V is small enough to be neglected.

The solution of (3.1) which is regular near $r=0$ is proportional to $r\,j_\ell(kr)$, where $k=\sqrt{2(E-V(0))}$ provided that $V(r)$ is sufficiently smooth for small values of r.

If $E>0$ there are two regular solutions for large r, and we obtain the scattering phase shift by integrating from some $r=\epsilon$ to some sufficiently large $r=A$, such that $V(r)\approx 0$ for $r>A$. Identifying the the solution near $r=A$ with the linear combination

$$kr\,j_\ell(kr)-\tan(\delta_\ell)kr\,y_\ell(kr)$$

(up to a constant factor) allows us to calculate the phase shifts $\delta_\ell(k)$ as functions of the wave number $k=\sqrt{2E}$. It is enough to identify the logarithmic derivative Du/u for the two solutions at $r=A$. In applications we do this calculation simultaneously for a few hundred values of the energy E, then displaying the phase shift as a function of E or of k.

Once we have the phase shifts $\delta_\ell(k)$, we can find the scattering amplitude as a sum over partial wave amplitudes

$$f(k,\Omega)=k^{-1}\sum_{\ell=0}^{\infty}(2\ell+1)\exp(i\delta_\ell)\sin\delta_\ell\,P_\ell(\cos\theta),$$

the differential cross section $\sigma(k,\Omega)=|f(k,\theta)|^2$, and, integrating over the angles, the total cross section as a sum over partial cross sections

$$\sigma_{\text{tot}}=\sum_{\ell=0}^{\infty}\sigma_\ell(k),\qquad \sigma_\ell(k)=\frac{4\pi}{k^2}(2\ell+1)\sin^2\delta_\ell.$$

The Bessel functions calculated by the MATLAB programs `besselj`, `bessely` are the standard Bessel functions, so we must use

$$kr\,j_\ell(kr)=\sqrt{\frac{\pi\,kr}{2}}\,J_{\ell+1/2}(kr),$$

and the same relation where $\{j,J\}$ are replaced by $\{y,Y\}$.

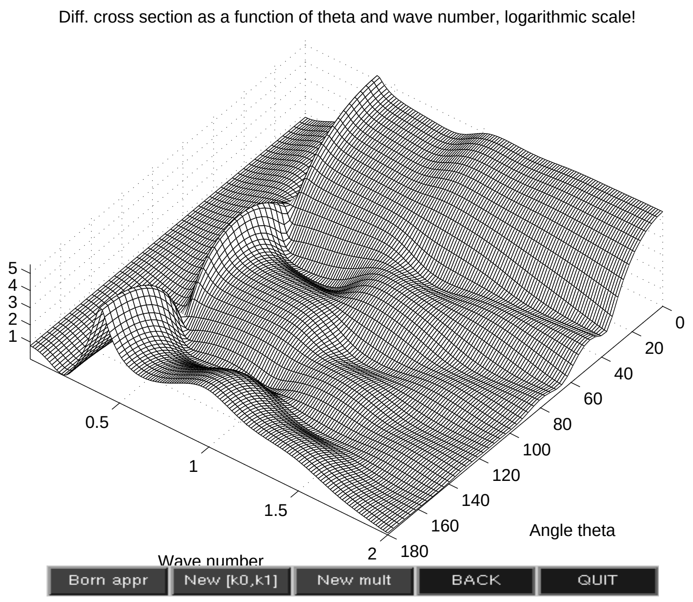
When $E < 0$ there is just one regular solution for large values of r, and this is the function

$$\sqrt{\frac{\pi\,\kappa r}{2}}K_{\ell+1/2}(\kappa\,r), \qquad \kappa = \sqrt{-2E},$$

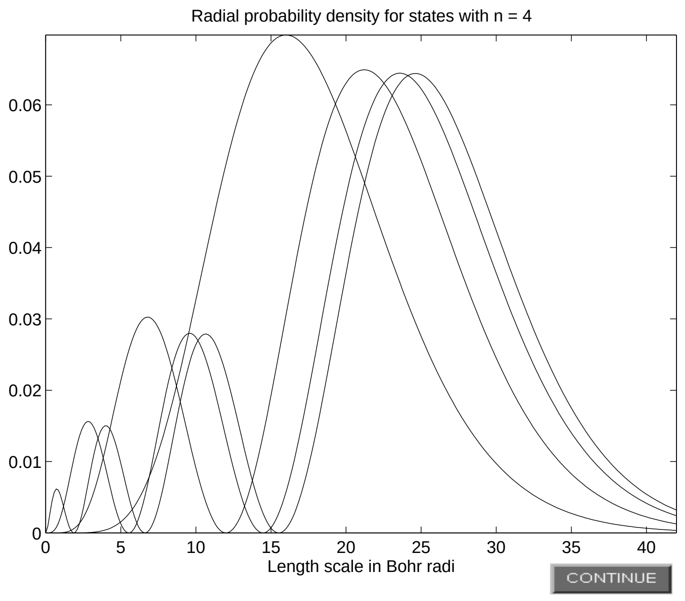
which is calculated by the MATLAB function `besselk`. If we again identify the logarithmic derivative of the solution obtained by numerical integration with that of this asymptotic solution at $r = A$ we obtain a discrete set of eigenvalue solutions for E.

References Cohen-Tannoudji [7] Ch. 8, Flügge [10], Messiah [22] Ch. 9, 10,

The file `pw3d.m` calculates the phase shifts of a finite number of partial waves for a central potential $V(r)$ as function of the wave number k, then the differential cross section as a function of (k,θ). It can also find the Born approximation.



The file `radial.m` calculates the H-atom radial eigenfunctions ($\ell = 0, 1, \dots n-1$) for a principal quantum number (n) of your choice, using a numerical evaluation of the Laguerre polynomials. You can then find the radial probability density



3.4 The WKB method

There are methods of calculating approximate energy spectra other than the numerical integration of the DE. These can sometimes be used to get a first approximation for the eigenvalues. It is then possible to use these values as starting values for an iterative numerical method of finding better approximations.

The semiclassical approximation called the WKB method allows a very rapid numerical estimate of the energy levels in simple 1D situations. It is easiest to use on potentials which have only one minimum (extending the method to potentials with several minima is possible but this refinement has not been included).

The WKB approximation for bound states is implemented by the function file `wkb.m`. It is called by a command `wkb(X,Y,Emax,N)` where X is row vector of space coordinates, $Y = V(X)$ the corresponding vector of values of the potential, E_{max} an upper bound on the output spectrum, and N the number of points in the energy lattice.

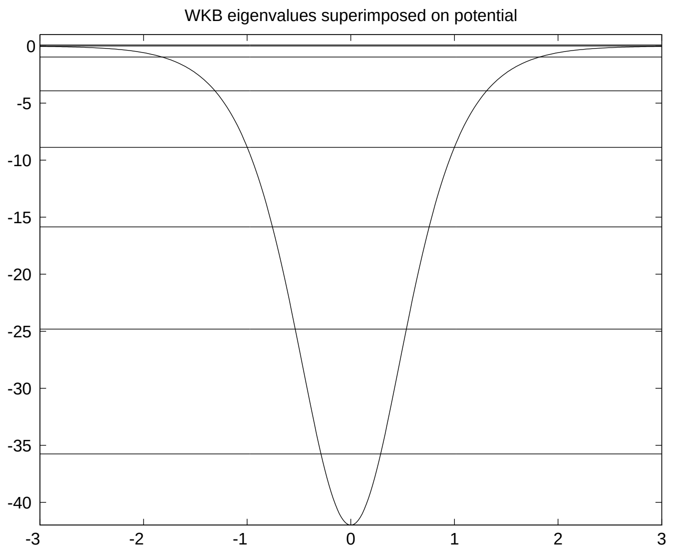
The energy levels in a single-minimum potential $V(x)$, numbered $n = 0, 1, \ldots$ are the values of E which

$$(n + \frac{1}{2})\pi = \int_a^b dx\, \sqrt{2\,[E - V(x)]}$$

where $(a(E), b(E))$ are the two *turning points* where $E - V(x) = 0$. By the assumption of a single minimum there are precisely two such values for all relevant values of E. Here we can deal with potentials going to a finite value at $x \to \pm\infty$ and those which increase without bound, like the harmonic oscillator.

The file `wkb.m` calculates the RHS for a lattice of values of E, then finds the energy spectrum E_N by interpolation.

References Messiah [22] Ch. 6, Flügge [10] Sect. II.E, Kroemer [18] Ch. 6.



An example of WKB approximations for bound state energies. The potential has the exact eigenvalues $= -n^2, n = 1, 2, \ldots, 6$.

3.5 Discrete approximations

There are several methods of finding approximate eigenvalues and eigenvectors for a differential operator with boundary values by reducing the problem to the diagonalization of a finite matrix. There is the very general approach called the finite element method (FEM), which we will not use here.

In `schrodinger` there are implemented two simple discretization procedures. One of these is the Fourier Grid Hamiltonian (FGH) method [21]. We can use this method for finding the eigenstates in an 1D potential on a finite interval, or the bound states in an attractive potential (including the radial case) provided that the wave functions have fast decay to zero outside a finite interval. We then have to choose the interval large enough, enabling us to neglect the wave functions outside, and search for solutions which are zero at both end points.

In the chosen interval we define a configuration space lattice with spacing Δx, the (odd) number of lattice points is $N = 2M + 1$. There is a corresponding limit on the resolution of the wave number $\Delta k = 2\pi/\Delta x$. The Hamiltonian is represented by a $N \times N$ matrix. The potential is represented by a diagonal matrix, while the kinetic term is obtained as a Fourier transform of the momentum space expression. The matrix Hamiltonian and the normalization of the eigenfunctions are given by

$$\begin{aligned}
H_{jk} &= K_{jk} + V(x_j)\delta_{jk} \\
K_{jk} &= \sum_{m=1}^{M} \left(\frac{2\pi m}{N\Delta x}\right)^2 \frac{1}{N} \cos\left(\frac{2\pi(j-k)m}{N}\right) \\
\Delta x &\sum_{n=1}^{N} |\psi_n|^2 = 1
\end{aligned}$$

Note that the kinetic part is a Toeplitz matrix, that is, a function of $j - k$ only; such matrices are easily generated in MATLAB using the `toeplitz` algorithm. The matrix eigenvalue equation is solved by the MATLAB algorithm `eig`, which gives both eigenvalues and eigenvectors.

The FGH method is surprisingly accurate in many problems. The restriction is that we may have to choose the number N very large if the bound state wave functions combine a slow decay with high wave numbers (weakly bound states in deep potentials). Depending on your computer N may be chosen in the range 50 to 150 to obtain a rapid solution.

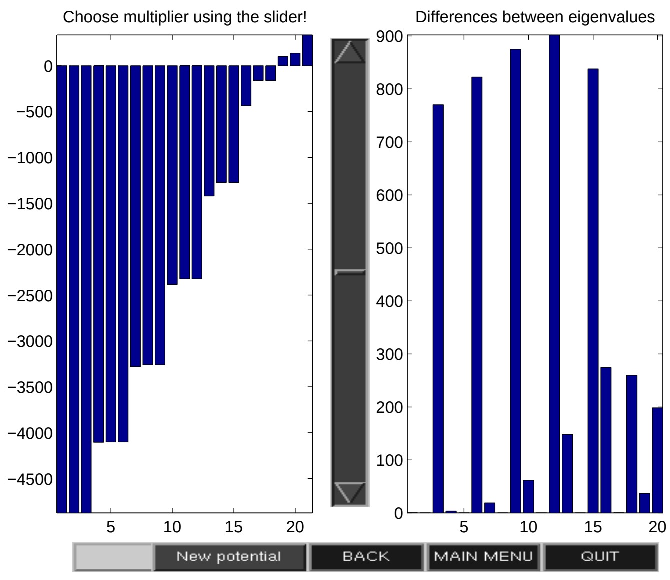
Another method of doing the discretization is to expand the Hamiltonian in a suitable basis set of functions which satisfy the boundary conditions. Consider a 1D system restricted to a finite interval, say $[0, 1]$. The Schrödinger Hamiltonian $H = -D^2/2 + V$ is defined when we have specified the BCs in the end points of the interval. For any CON set (basis) of functions $\{e_n(x)\}$ on the interval satisfying these BCs we can then transform the eigenvalue problem to a matrix eigenvalue problem for the matrix

$$H_{mn} = \langle e_m|H|e_n\rangle = \int dx\, e_m(x)^*(He_n)(x).$$

For instance, the basis functions can be chosen as the harmonic functions, i.e. eigenfunctions of $H = -D^2/2$, satisfying the BCs. The matrix elements of D^2 then form a diagonal matrix. The matrix elements of the potential V are calculated numerically, and they form a Toeplitz matrix. For the harmonic basis functions we can use the FFT for the calculation. Restricting the indices to a finite range we again get a finite matrix eigenvalue problem.

It is not so easy to estimate the error due to the finite size of the basis; in the present context this is perhaps not an important drawback. One can get an idea of their magnitude through a study of a number of exactly solvable problems (see next section). The errors are sometimes quite evident. We know that reflection symmetric potentials have eigenfunctions which are alternating even and odd: the ground state is even, the first excited state odd, and so on. In 1D the bound states are always non-degenerate, the ground state has no node, the N-th excited state has N nodes. When the symmetric potential has two minima separated by a high barrier, there will be pairs of nearly degenerate levels where the lower one must be even, the higher one odd. The lack of accuracy means that the file `matrix1.m` will sometimes fail to give the right eigenfunctions (or the right order) for such nearly degenerate levels.

The file `matrix2.m` calculates the eigenvalues of a potential defined on $[0, 1]$ with BC $y(0) = y(1) = 0$. Here the potential has 3 equally deep minima, the lowest eigenvalues come in closely spaced triplets. The strength of the potential relative to the kinetic term can be changed with the slider.



3.6 Exactly solvable cases

In testing methods of numerical integration and other approximations it is very useful to have a number of examples which are solvable by analytic methods, allowing us to compare the numerical and exact solutions. The **harmonic oscillator** is an obvious case

$$V(x) = \frac{k}{2}x^2, \quad E(n) = (n + \frac{1}{2})\sqrt{k},\ n = 0, 1, \ldots$$

The **Morse potential** goes to $+\infty$ as $x \to -\infty$ and to 0 as $x \to +\infty$; it is defined by two positive parameters $\{k_1, k_2\}$:

$$V(x) = k_1\{\exp(-2k_2 x) - 2\exp(-k_2 x)\}$$

There are $N :=$ integer part of $(1/2 + \sqrt{2k_1}/k_2)$ bound states

$$E(n) = \frac{k_2}{2}\,(n + \frac{1}{2})\{2\sqrt{2k_1} - k_2(n + \frac{1}{2})\} - k_1, \quad n = 0, 1, \ldots, N-1,$$

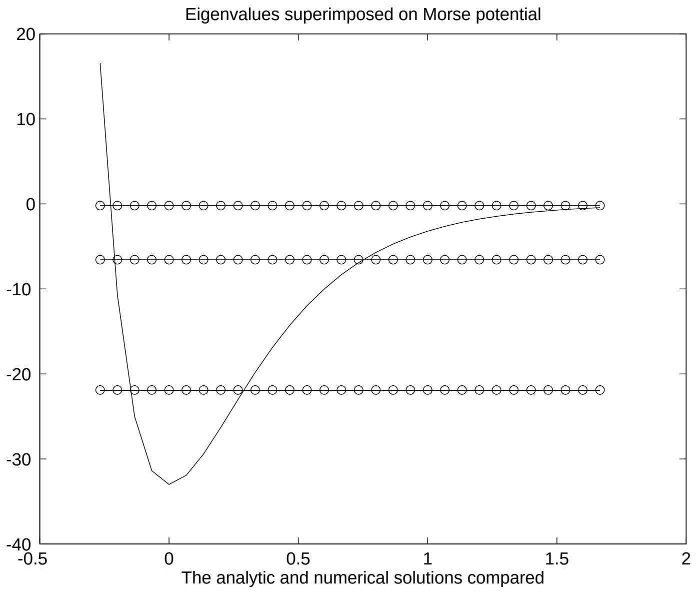
and `morse.m` compares them with the numerically calculated ones.

A family of solvable potentials which go to zero as $x \to \pm\infty$ is the following

$$V(x) = -N(N+1)\,\mathrm{sech}^2\sqrt{2}\,x, \quad E(n) = -n^2,\ n = 1, 2, \ldots, N,\ N = 1, 2, \ldots$$

These potentials are also reflectionless, the transmission coefficient is 1 for all energies! You can check this numerically using `transm.m`!

References Morse potential: [9] Chapter 13, [10], Problem 70; sech^2 potential: [19].



The file `morse.m` displays the bound states in a Morse potential where you can choose the parameters $\{k_1, k_2\}$. The exact $(- - -)$ and numerical $(\circ)$ values are compared. The latter are obtained by integration of the differential equation and an automatic search for solutions of the boundary conditions.

Chapter 4

Special functions

Some of the most used special functions are in the standard MATLAB toolbox. In version 5.2 these include the Airy, Bessel, Beta, Gamma, Jacobi elliptic, Legendre, and error functions, as well as the exponential integral.

In `schrodinger` there are algorithms for calculating Hermite and Laguerre polynomials, spherical harmonics, the radial eigenfunctions of hydrogen, and the Fresnel integral. In addition there are programs for finding the zeros of Bessel functions J_m and of the Airy functions. There are also programs illustrating the expansion of arbitrary functions into series of some of these special functions. These are not necessarily the best from a numerical point of view, but they use the matrix methods of MATLAB to speed up the calculations.

In general we want the functions for a finite lattice of M values of the space variable, and often a finite number N of functions which form the N first elements in a CON basis. Such a set thus make up a $M \times N$ matrix, which can be of a considerable size with a large demand on memory. In the interest of speed it is still useful to use the matrix form as far as possible.

There are numerous FORTRAN programs available for the calculation of the special functions, and they can be compiled and used in MATLAB when accurate results are needed. The convergence problems may be serious for some of these functions. It is also possible to use the much larger libraries of special functions in Maple and Mathematica from MATLAB. We have avoided all such complications here, only a few functions are used.

For special functions defined by series or integrals with slow convergence there are often available approximations involving expansions in e.g. Chebychev polynomials. This is the case of the Fresnel integrals (for calculating interference patterns), see [15], and the file `fresn.m`. Starting from a relatively small number of numerical coefficients, the computation of a the approxmination is very fast and the accuracy good over the whole range of definition.

For the Hermite, Legendre and Laguerre polynomials the recursion relations are often enough for our purposes. As an example look at the Hermite polynomials defined by the recursion relation

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x), \quad H_0(x) = 1,\ H_1(x) = 2x.$$

which gives H_n as a polynomial of n-th order, containing either even or odd powers. In

MATLAB a polynomial is most conveniently represented by a vector of the coefficients in descending powers. The M-file `hp.m` iterates these vectors as follows (removing some comments)

```
function f=hp(n);

x0=[1];  x1=[2 0]; % starting values.
if n==0
w=x0;
else
w=[0 x0;x1];
end

for m=2:n
y=2*[x1 0]-2*(m-1)*[0 0 x0];
x0=x1;  x1=y;
w=[zeros(m,1) w; x1];
end

f=w;
```

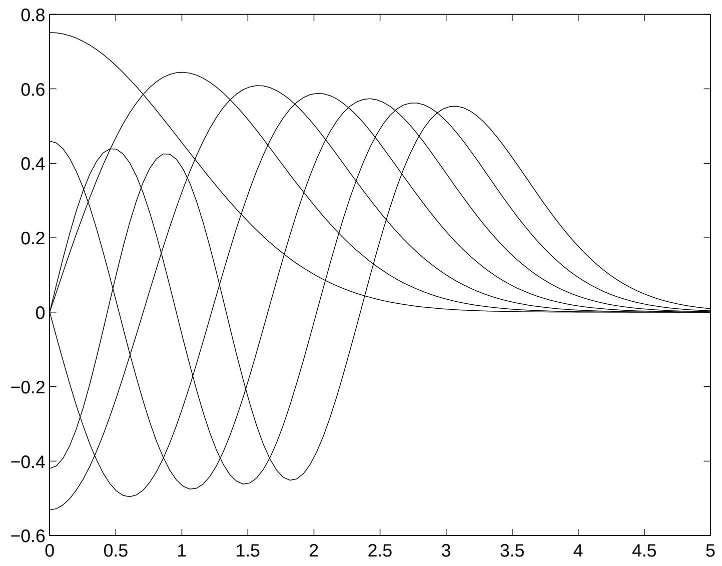
The result is a matrix of order $(n+1) \times (n+1)$, for example

```
>> hp(6)

ans =

     0     0     0     0     0     0     1
     0     0     0     0     0     2     0
     0     0     0     0     4     0    -2
     0     0     0     8     0   -12     0
     0     0    16     0   -48     0    12
     0    32     0  -160     0   120     0
    64     0  -480     0   720     0  -120
```

From this matrix we can calculate the harmonic oscillator eigenfunctions, this is done by the file `ho.m`: the call `ho(n,x)` calculates the values of the $n+1$ first of these functions in a the points of the row vector x. The M-file uses some other algorithms of `schrodinger`, which make the algorithm simple, deleting comments:

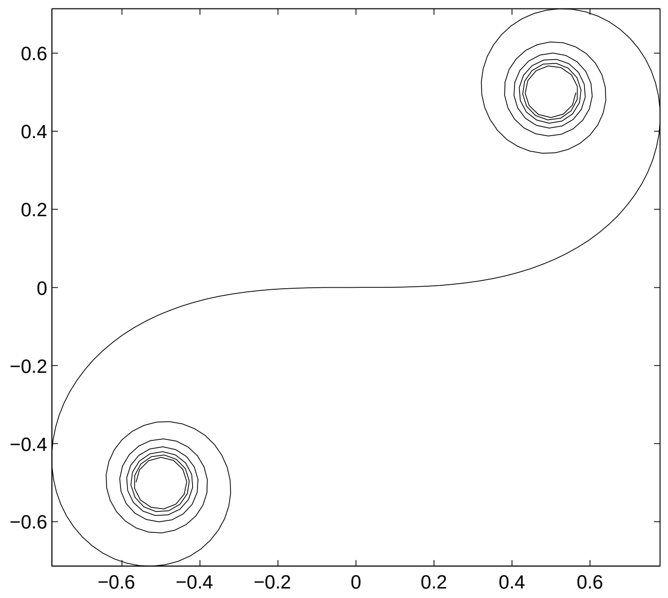
```
function H = ho(n,x);

nn=[0:n]';
w= 1./sqrt(sqrt(pi)*fact(nn).*2.^nn)*exp(-x.^2/2);

H = w.*evalpol(hp(n),x);
```

Below we see the graph given by `x=linspace(0,5); plot(x,ho(6,x));`, it pictures the 6 first eigenfunctions of the harmonic oscillator for $x > 0$:



Using `fresn.m` we generate the Cornu spiral below (the imaginary versus the real parts of the Fresnel integral) as follows

```
x=linspace(-5,5,300); y=fresn(x);
plot(real(y),imag(y)); axis equal,  axis tight;
```



References [1,3].

Bibliography

[1] A. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, Dover 1965

[2] S. K. Adhikari, "Quantum scattering in two dimensions", Am. J. Phys. **54**(4), 362–367 (1986)

[3] G. Arfken and H. Weber, *Mathematical Methods for Physicists*, 4. ed, Academic Press 1995

[4] S. Brandt and H. D. Dahmen,
– *Quantum Mechanics on the Macintosh*, Springer-Verlag 2. ed 1995
– *Quantum Mechanics on the Personal Computer*, Springer-Verlag 3. ed 1994
– *The Picture Book of Quantum Mechanics*, Springer-Verlag, 2. ed 1995

[5] B. H. Bransden and C. J. Joachain, *Quantum Mechanics* , Longman Scientific 1989.

[6] F. Brau and C. Semay, "The three-dimensional Fourier grid Hamiltonian method", J. Comput. Phys. **139**, 127–136 (1998)

[7] C. Cohen-Tannoudji et al, *Quantum Mechanics*, 2 vols, Wiley/Hermann 1977

[8] P. L. DeVries, "Resource Letter CP-1: Computational Physics", Am. J. Phys. **64**(4), 364–8 (1996)

[9] J. M. Feagin, *Quantum Methods with Mathematica*, Springer-Verlag/Telos 1994

[10] S. Flügge, *Practical Quantum Mechanics*, Springer-Verlag 1994

[11] A. Garcia, *Numerical Methods for Physics*, Prentice Hall 1994

[12] S. Gasiorowicz, *Quantum Physics*, 2. ed, Wiley 1995

[13] N. J. Giordano, *Computational Physics*, Prentice-Hall 1997

[14] H. Gould and J. Tobochnik, *An Introduction to Computer Simulation Methods*, 2. ed, Addison-Wesley, 1996

[15] M. A. Heald, " Rational approximations for the Fresnel integrals", Math. Comp. **44**, 459–461 (1985)

[16] E. Jahnke and F. Emde, *Tables of Functions with Formulae and Curves*, 4. ed, Dover 1945

[17] B. R. Johnson, "New numerical methods applied to solving the one-dimensional eigenvalue problem", J. Chem. Phys. **67**(9), 4086–93 (1977)

[18] H. Kroemer, *Quantum Mechanics* , 2. ed, Prentice-Hall 1994

[19] G. L. Lamb, *Elements of Soliton Theory* , Wiley 1980 (Section 2.5)

[20] Mathematica: Wolfram Research, `http://www.wri.com/`,
Maple: Waterlo Maple Inc, `http://www.maplesoft.com/`,
MuPAD: University of Paderborn and SciFace, `http://www.mupad.de/`,
MATLAB: MathWorks, `http://www.mathworks.com`,
Octave: GNU, `http://www.che.wisc.edu/octave/`,
SciLab: INRIA, `http://arikara.inria.fr/www-rocq.inria.fr/scilab/`.

[21] C. C. Marston and G. G. Balint-Kurti, "The Fourier grid Hamiltonian method for bound state eigenvalues and eigenfunctions", J. Chem. Phys. **91**(6), 3571–76 (1989)

[22] A. Messiah, *Quantum Mechanics*, 2 vols., North Holland 1964

[23] E. Merzbacher, *Quantum Mechanics*, 3. ed, Wiley 1998

[24] W. H. Press et al, *Numerical Recipes*, Cambridge UP 1986.

[25] R. W. Robinett, *Quantum Mechanics: Classical Results, Modern Systems and Visualized Examples*, Oxford UP 1997

[26] E. W. Schmid, G. Spitz and W. Lösch, *Theoretical Physics on the Personal Computer*, 2. ed, Springer-Verlag, 1990

[27] P. B. Visscher, "A fast explicit algorithm for the time-dependent Schrödinger equation", Computers in Physics Nov/Dec 1991, 596–8