Tutorial

Numerical Solution of the Schrödinger Equation.

Nadezhda Gribova*

December 23, 2009
ICP, Uni Stuttgart

1 Algorithm

In physics, specifically quantum mechanics, the Schrödinger equation is an equation that describes how the quantum state of a physical system changes in time. If we reduced it to one dimension and make independent of time, the Schrödinger equation can be written as follows:

$$\left[-\frac{\hbar^2}{2\mu}\frac{\mathrm{d}^2}{\mathrm{d}x^2}+V(x)\right]\psi(x)=E\psi(x) \tag{1}$$

where $V(x)$ is the potential, x goes from minus infinity to plus infinity, and the mass is given in general by a reduced mass μ.

The equation in this form is simple to solve and to do this numerically there is a number of techniques available. Our task will be to find the bound states. These have the boundary conditions that the wave functions go to zero as x goes to $\pm\infty$. So numerically we pick a large, but finite value of $|x|$, say $|x| = a$ and assume that the wave functions are identically zero at those values of x. Then one has a so-called "two-point boundary value problem".

To solve the equation we will use a "finite difference" scheme. We obtain the values of the wave functions at a discrete set of points between $-a$ and a. In the approach we will use, these are equally spaced points, and the second derivative of the wave function is replaced by a finite difference approximation to it. This, together with the above boundary conditions, converts the differential Schrödinger equation into a matrix Eigenvalue/Eigenvector equation.

Let's go a bit more into details. At first we have to numerically approximate the second derivative in equation 1. We employ an equally spaced grid for points $x_1, x_2, \ldots, x_{n-1}, x_n$

*gribova@icp.uni-stuttgart.de

with uniform step h. Then we replace the second derivative by the simplest finite difference approximation:

$$\psi''(x) \approx \frac{1}{h^2}\left[\psi(x+h) - 2\psi(x) + \psi(x-h)\right]$$

We apply this expression at each grid point and keep the boundary conditions $\psi(x_1) = \psi(x_n) = 0$ in mind. So the differential equation is transformed to a homogeneous set of linear equations for the unknowns, i.e., the wave function at the n-2 interior points. In usual matrix form, this system of equations can be written as

$$(H - E)\psi = 0 \tag{2}$$

where

$$\begin{aligned} &(H-E)_{i,i} = \left(\frac{\hbar^2}{\mu h^2} + V(x_i) - E\right) \\ &(H-E)_{i,\pm i} = \frac{\hbar^2}{2\mu h^2} \\ &(H-E)_{i,j} = 0 \quad \text{otherwise} \\ &\psi_i = \psi(x_i),\ i = 2, \ldots, n-1 \end{aligned} \tag{3}$$

The homogeneous set of equations has a non-trivial solution provided $\det(H - E) = 0$. So, we have another example of a problem of finding Eigenvalues E_n and the corresponding Eigenvectors (the wave functions) ψ_n.

2 Practice

2.1 Eigenvalues and Eigenvectors of Matrices

2.1.1 Task (1 point)

Find analytically the Eigenvalues and Eigenvectors of the matrix

$$\begin{pmatrix} 3 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 3 \end{pmatrix} \tag{4}$$

2.1.2 Task (1 point)

Write a program that numerically finds Eigenvalues and Eigenvectors of the matrix 4 from task 2.1.1 in arbitrary dimensions.

Hint: You can use the open-source scientific library that you have already used in the previous tutorials, GSL, which provides functions the following functions to solve Eigenvalue problems:

- gsl_matrix * **gsl_matrix_alloc** (size_t n1, size_t n2)
  This function creates a matrix of size n1 rows by n2 columns, returning a pointer to a newly initialized matrix struct. A new block is allocated for the elements of the matrix, and stored in the block component of the matrix struct. The block is 'owned' by the matrix, and will be deallocated when the matrix is deallocated.

- gsl_vector * **gsl_vector_alloc** (size_t n)
  This function creates a vector of length n, returning a pointer to a newly initialized vector struct. A new block is allocated for the elements of the vector, and stored in the block component of the vector struct. The block is 'owned' by the vector, and will be deallocated when the vector is deallocated.

- gsl_eigen_symm_workspace * **gsl_eigen_symm_alloc** (const size_t n)
  This function allocates a workspace for computing Eigenvalues of n-by-n real symmetric matrices. The size of the workspace is O(2n).

- gsl_vector_view evec_i **gsl_matrix_column** (gsl_matrix * m, size_t j)
  This function returns a vector view of the j-th column of the matrix m. The data pointer of the new vector is set to null if j is out of range.

- double **gsl_vector_get** (const gsl_vector * v, size_t i)
  This function returns the i-th element of a vector v. If i lies outside the allowed range of 0 to n-1 then the error handler is invoked and 0 is returned.

- void **gsl_matrix_set** (gsl_matrix * m, size_t i, size_t j, double x)
  This function sets the value of the (i,j)-th element of a matrix m to x. If i or j lies outside the allowed range of 0 to n1-1 and 0 to n2-1 then the error handler is invoked.

- void **gsl_matrix_set_zero** (gsl_matrix * m)
  This function sets all the elements of the matrix m to zero.

- int **gsl_eigen_symmv** (gsl_matrix * A, gsl_vector * eval,
  gsl_eigen_symm_workspace * w)
  This function computes the Eigenvalues of the real symmetric matrix A. Additional workspace of the appropriate size must be provided in w. The diagonal and lower triangular part of A are destroyed during the computation, but the strict upper triangular part is not referenced. The Eigenvalues are stored in the vector eval and are unordered.

- void **gsl_eigen_symmv_free** (gsl_eigen_symm_workspace * w)
  This function frees the memory associated with the workspace w.

- int **gsl_eigen_symmv_sort** (gsl_vector * eval, gsl_matrix * evec,
  gsl_eigen_sort_t sort_type)
  This function simultaneously sorts the Eigenvalues stored in the vector eval and the corresponding real Eigenvectors stored in the columns of the matrix evec into ascending or descending order according to the value of the parameter sort_type.
  GSL_EIGEN_SORT_ABS_ASC - ascending order in magnitude

- int **gsl_vector_fprintf** (FILE * stream, const gsl_vector * v, const char * format)
  This function writes the elements of the vector v line-by-line to the stream stream using the format specifier format, which should be one of the %g, %e or %f formats for floating point numbers and %d for integers. The function returns 0 for success and GSL_EFAILED if there was a problem writing to the file.

- void **gsl_vector_free** (gsl_vector * v)
  This function frees a previously allocated vector v. If the vector was created using gsl_vector_alloc then the block underlying the vector will also be deallocated. If the vector has been created from another object then the memory is still owned by that object and will not be deallocated. The vector v must be a valid vector object (a null pointer is not allowed).

- void **gsl_matrix_free** (gsl_matrix * m)
  This function frees a previously allocated matrix m. If the matrix was created using gsl_matrix_alloc then the block underlying the matrix will also be deallocated. If the matrix has been created from another object then the memory is still owned by that object and will not be deallocated. The matrix m must be a valid matrix object (a null pointer is not allowed).

To use these GSL-functions the following libraries should be included:

```
<gsl/gsl_math.h>
<gsl/gsl_matrix.h>
<gsl/gsl_eigen.h>
```

For further details consult the online documentation and examples of GSL:
http://www.gnu.org/software/gsl/manual/html_node/Matrices.html
http://www.gnu.org/software/gsl/manual/html_node/Eigensystems.html

2.2 Stationary Schrödinger equation

2.2.1 Task (3 points)

Find analytically all energies E_n and wave functions $\psi_n(x)$ of the stationary Schrödinger equation 1 for an infinite well:

$$V(x) = \begin{cases} \infty & \text{if} \quad |x| > 1 \\ 0 & \text{if} \quad |x| \leq 1 \end{cases} \tag{5}$$

Provide how the solution was obtained.

2.2.2 Task (3 points)

Write a program that solves numerically task 2.2.1 using the pattern explained in section 1. For convenience set $\frac{\hbar}{\mu} = 1$. Instead of $V(x) = \infty$ outside of the box use a two step potential at both boundaries: for $1 \leq |x| \leq 5$, choose $V(x) = 20; 50; 100; 1000$, and only at $|x| \geq 5$ apply the boundary condition $\psi(x) = 0$. Use different numbers of grid points when solving 3. Provide a plot of the first 5 Eigenfunctions for $V(x) = 20$. Compare the first 5 Eigenvalues and Eigenvalues you get with the analytic solution for $V(x) = \infty$. How fast does the solution at finite barrier converge against the solution for infinite potential?

2.2.3 Task (2 points)

We consider the Morse potential

$$V(x) = D\left(1 - e^{a(x-x_0)}\right) \tag{6}$$

which is a simple model for the potential energy of a diatomic molecule with average bond distance x_0. We choose $D = 20$ and $a = 1$. What are the allowed energies? How do the corresponding wave functions look like? Describe and provide the plots.