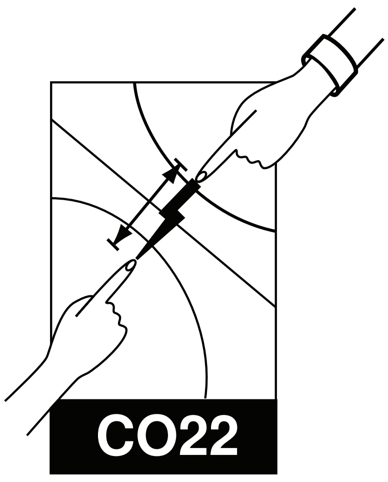
Solutions of Schrödinger's equation by numerical integration

revised EG, MK, NC October 2020



Preface

It is common practice in modern software engineering to write programs in a modular and standardised way. To help you get started, appendix A provides you with one or more specific function headers which you **MUST** use to write the functions around which your program should be built and in order to receive a satisfactory mark. Your comments in the header of each function must include:

- author and date,
- purpose (a brief description of what the function does),
- inputs and outputs,
- and, if appropriate, any constraints or limitations of use.

Do not forget that you will also need to keep good records of your progress during this practical in your logbook. If you make plots and/or write any notes electronically, ideally, you would print them and affix them into the pages of your logbook. You should have comments in function and script headers of your code, as well as comments within your code. These aspects may all be considered at marking time.

The template provided is for the default language of the Lab: MATLAB. If you have checked with a demonstrator about using another language and they have authorised you to do so, you must use the functional equivalent of the appendix A function headers in that language.

Assessment

Please see the Part A Guide in Canvas for Computing Lab deadlines as well as availability (schedule) of Computing Lab Demonstrators who mark your work as well as offer help and advice. Before you meet with a demonstrator for marking, **you must** upload your work (both your code and your report based on AD34 — *the art of scientific report writing*[1]) into WebLearn (also described in Canvas).

1 Introduction

The aim of this project is to find numerical solutions to the time-independent Schrödinger equation for the quantum harmonic oscillator. This problem illustrates the solution of a stiff differential equation.

2 Physical description

The time-independent Schrödinger equation in one dimension can be written

$$\frac{\mathrm{d}^2\Psi}{\mathrm{d}x^2} + \frac{2m}{\hbar^2}[E - V(x)]\Psi = 0 \qquad (1)$$

[1] www-teaching.physics.ox.ac.uk/practical_course/Admin/AD34.pdf

Solutions to this equation with suitable boundary conditions exist in closed form only for special choices of the $V(x)$, though numerical solutions can be found in principle, provided they exist, for any $V(x)$. In this example you are asked to obtain numerical solutions for the case $V(x) = {}^1\!/_2 kx^2$, the Simple Harmonic Oscillator, where k is the 'spring constant' of the oscillator. This potential is equivalent to a force $-kx$. In this case, analytic solutions exist and can be compared with the numerical solutions.

It is convenient to use non-dimensional variables. Let

$$\hat{x} = \left(\frac{mk}{\hbar^2}\right)^{1/4} x \quad \text{and} \quad \hat{E} = \frac{2}{\hbar}\sqrt{\frac{m}{k}}E$$

so that equation 1 reduces to

$$\frac{\mathrm{d}^2\Psi}{\mathrm{d}x^2} = (\hat{V}(\hat{x}) - \hat{E})\Psi \tag{2}$$

where $\hat{V}(\hat{x}) = \hat{x}^2$. For physically acceptable solutions, i.e. those for which $|\Psi|^2$ is integrable, we require Ψ to tend to zero as $\hat{x} \to \pm\infty$. This is only possible for certain values of $\hat{E}$, called eigenvalues.

Analytic solutions exist to equation 2 in the form

$$\Psi = H_n(\hat{x})e^{-\hat{x}^2/2}$$

where n takes any non-negative integer value and H_n are the Hermite polynomials, defined by $H_0 = 1$, $H_1 = 2\hat{x}$, and the recurrence relation:

$$H_{n+1}(\hat{x}) = 2\hat{x}H_n(\hat{x}) - 2nH_{n-1}(\hat{x})$$

The corresponding eigenvalues are particularly simple:

$$\hat{E}_n = 2n + 1 \qquad (n \text{ non-negative integer})$$

3 Numerical approach

For a differential equation in the general form

$$\frac{\mathrm{d}^2\Psi}{\mathrm{d}\hat{x}^2} = f(\hat{x})\Psi \tag{3}$$

i.e. with no first derivative terms, a very good integration algorithm is Numerov's Method [1]. For a uniformly spaced set of grid points with spacing δ the value of Ψ at grid point $j+1$ is approximately related to the values at grid points j and $j-1$ by

$$\left(1 - \frac{\delta^2}{12}f_{j+1}\right)\Psi_{j+1} = \left(2 + \frac{5}{6}\delta^2 f_j\right)\Psi_j - \left(1 - \frac{\delta^2}{12}f_{j-1}\right)\Psi_{j-1} \tag{4}$$

where f_j and Ψ_j are the values of f and Ψ at $\hat{x} = j\delta$.

We must think about where to start the integration, and about the boundary conditions. The obvious starting points are at $\hat{x} = 0$ or at $\hat{x} = \infty$ (or in practice some large value).

Since $\hat{V}(\hat{x})$ is symmetric, solutions fall into even and odd categories. If we start at $\hat{x} = 0$, convenient boundary conditions will be

$$\begin{aligned} \text{even:} \quad & \Psi(0) = 1 \quad \tfrac{\mathrm{d}\Psi}{\mathrm{d}\hat{x}} = 0 \\ \text{odd:} \quad & \Psi(0) = 0 \quad \tfrac{\mathrm{d}\Psi}{\mathrm{d}\hat{x}} = 1 \end{aligned}$$

Note that you will need values of Ψ at two grid points to start this calculation, but the boundary condition only provides the value at one point for a given solution. The value at a second point can be calculated using a Taylor series expansion:

$$\Psi(\delta) = \Psi(0) + \Psi'(0)\delta + {}^1\!/\!_2\Psi''(0)\delta^2 + {}^1\!/\!_6\Psi'''(0)\delta^3 + {}^1\!/\!_{24}\Psi''''(0)\delta^4 + \ldots$$

where the prime denotes differentiation with respect to $\hat{x}$.

▶ **Show that this leads to:**

$$\textbf{even:}\quad \Psi(\delta) = \Psi(0) + \frac{\delta^2}{2} f(0)\Psi(0) + \frac{\delta^4}{24}\left\{f''(0)\Psi(0) + 2f'(0)\Psi'(0) + f(0)^2\Psi(0)\right\} + \ldots$$

$$\textbf{odd:}\quad \Psi(\delta) = \delta\Psi'(0) + \frac{\delta^3}{6}\left\{f(0)\Psi'(0) + f'(0)\Psi(0)\right\} + \ldots$$

Note that, for the harmonic oscillator of section 2, $f'(0) = 0$ and even n corresponds to even solutions and odd n to odd solutions. This is simply the standard method of solving differential equations in terms of a polynomial.

4 Computation

Write the function to solve equation 3 with algorithm 4 using the template given in appendix A.1. Then write a script to call the function to integrate equation 2 from $\hat{x} = 0$ to some suitable upper bound, $\hat{x}_1$, and plot both the numerical result and the analytic solution. Your program should take δ, $\hat{x}_1$, n and $\hat{E}$ as input, and should be able to deal with both even and odd n. You may find it convenient to write $f(\hat{x})$ and the Numerov equation as functions. Note that equation 2 is homogeneous, so that a solution may be scaled by an arbitrary factor and still be a solution. The analytical and numerical solutions may have different scalings which must be corrected for when they are compared.

Choose $\delta = 0.05$ and $\hat{x}_1 = 5$ initially. Explore the dependence of the solution on the value of $\hat{E}$ by running your program with $\hat{E} = 0.95$, 1.0 and 1.05.

▶ **Why does your numerical solution differ from the analytic solution for large $\hat{x}$?**

Devise a method by which the eigenvalue $\hat{E} = 1$ can be obtained when starting from a trial value of $\hat{E}$ reasonably near $\hat{E} = 1$. One possibility, by no means the best, is to look for the value of $\hat{E}$ which gives the smallest absolute value of the wave function at $\hat{x}_1 = 5$. Starting from the first trial value, determine n of the closest analytical eigenvalue $\hat{E}_n$, provide the corresponding boundary condition to your Numerov method and proceed in small steps in the direction of $\hat{E}$ which reduces the absolute value of the wave function. The implementation needs to follow the template given in Appendix A.2.

Compute also the solutions for $\hat{E}$ near $\hat{E} = 3$, 5 and 7.

▶ **You may opt at any point to discuss results or plots with a demonstrator before proceeding.**

5 Final considerations

Consider the relative advantages and disadvantages of Numerov's method along with its mathematical origin and its associated errors. What other numerical methods could be used to solve equation 2? You ~~may wish to consult [1] and our local resources~~[2] on numerical methods.

[2]http://www-teaching.physics.ox.ac.uk/computing/NumericalMethods/nummethods.html

6 Preparing for assessment

Part A Computing Practicals are an exercise in computer programming as well as in scientific report writing. Your report must be written in the style described in AD34 — *the art of scientific report writing*[a]. To write your report, you have a few options; Microsoft Word or LaTeX typesetting are most common. Whatever your choice, the system you use must be able to produce text-readable PDF format (not PDF created from a scanned image on a printer).

When you have finished assembling your code (and have tested it completely) and your report is complete, you must upload your work (both the code and the report) electronically to WebLearn (described in the Part A Guide in Canvas). Be sure to complete these uploads well in advance of meeting with a demonstrator for marking. Deadlines and Computing Demonstrator schedules are also found in Canvas.

At marking time, be prepared with a copy of your report and your logbook (where you wrote extra notes during your program development). Be prepared to describe your code and demonstrate its execution.

[a]www-teaching.physics.ox.ac.uk/practical_course/Admin/AD34.pdf

A Functions to be implemented

A.1 Numerov

```
function [psi] = solve_numerov(f, x, psi0, dpsi0)
% Author: ??? , Date: ??/??/????
% Solving d^2(psi)/dx^2 = f(x) psi from x0 to x1 with boundary condition
% psi(x0) = psi0 and d(psi(x0))/dx = dpsi0
% Input:
% * f: The function to be called on the right hand side of the equation (see hint
%      section), receives x and returns the value of f(x).
% * x: array of the integration, the first element is the lower bound, x0, and the last
%      element is the upper bound, x1.
% * psi0: the value of \psi at x0, i.e. \psi(x0)
% * dpsi0: the value of \psi derivative at x0, i.e. d\psi(x0)/dx
%
% Output:
% * psi: array of values of \psi with each element in the array corresponding to the same
%        element in x
%
% Example use:
% >> f = @(x) x^2 - 1;
% >> x = linspace(0, 1, 100);
% >> psi0 = 1;
% >> dpsi0 = 0;
% >> psi = solve_numerov(f, x, psi0, dpsi0);
% >> plot(x, psi);
end
```

A.2 Finding eigenvalue for oscillator

```
function [E] = find_oscillator_eigenvalue(E0)
% Author: ??? , Date: ??/??/????
% Finding the eigenvalue for harmonic oscillator potential by iteration with starting
% value E0.
% Input:
% * E0: the initial guess of the eigenvalue
%
% Output:
% * E: the eigenvalue near the initial guess of the system with harmonic oscillator
%        potential
%
% Constraints:
% * E0 is a positive real number
%
% Example use:
% >> E = find_oscillator_eigenvalue(1.2);
% >> disp(E);
end
```

B Hint: Lambda function

If you want to pass a function as an argument of another function, you need to define it as a 'lambda' or 'anonymous' function. A lambda function is defined by the syntax:

```
func_name = @(arg1, arg2, ...) ...
```

with `arg*` the arguments of the function. The example below shows how to define a lambda function that returns the square value of the argument, and how to call it.

```
>> fsquare = @(x) x.*x; % defining the function
>> a = fsquare(4); % calling the function
>> a
16
```

Bibliography

[1] C. E. Fröberg, *Introduction to Numerical Analysis*, 2nd edition, Addison Wesley, 1969 (p. 25).