Chapter 9
Two point boundary value problems

Abstract When differential equations are required to satisfy boundary conditions at more than one value of the independent variable, the resulting problem is called a *boundary value problem*. The most common case by far is when boundary conditions are supposed to be satisfied at two points - usually the starting and ending values of the integration. The Schrödinger equation is an important example of such a case. Here the eigenfunctions are typically restricted to be finite everywhere (in particular at $r = 0$) and for bound states the functions must go to zero at infinity.

9.1 Introduction

In the previous chapter we discussed the solution of differential equations determined by conditions imposed at one point only, the so-called initial condition. Here we move on to differential equations where the solution is required to satisfy conditions at more than one point. Typically these are the endpoints of the interval under consideration. When discussing differential equations with boundary conditions, there are three main groups of numerical methods, shooting methods, finite difference and finite element methods. In this chapter we focus on the so-called shooting method, whereas chapters 7 and 10 focus on finite difference methods. Chapter 7 solves the finite difference problem as an eigenvalue problem for a one variable differential equation while in chapter 10 we present the simplest finite difference methods for solving partial differential equations with more than one variable. The finite element method is not discussed in this text, see for example Ref. [48] for a computational presentation of the finite element method.

In the discussion here we will limit ourselves to the simplest possible case, that of a linear second-order differential equation whose solution is specified at two distinct points, for more complicated systems and equations see for example Refs. [49, 50]. The reader should also note that the techniques discussed in this chapter are restricted to ordinary differential equations only, while finite difference and finite element methods can also be applied to boundary value problems for partial differential equations. The discussion in this chapter and chapter 7 serves therefore as an intermediate step and model to the chapter on partial differential equations. Partial differential equations involve both boundary conditions and differential equations with functions depending on more than one variable.

In this chapter we will discuss in particular the solution of the one-particle Schödinger equation and apply the method to hydrogen-atom like problems. We start however with a familiar problem from mechanics, namely that of a tightly stretched and flexible string or rope, fixed at the endpoints. This problem has an analytic solution which allows us to define our numerical algorithms based on the shooting methods.

9.2 Shooting methods

In many physics applications we encounter differential equations like

$$\frac{d^2y}{dx^2} + k^2(x)y = F(x);\ a \leq x \leq b, \tag{9.1}$$

with boundary conditions

$$y(a) = \alpha,\ y(b) = \beta. \tag{9.2}$$

We can interpret $F(x)$ as an inhomogenous driving force while $k(x)$ is a real function. If it is positive the solutions $y(x)$ will be oscillatory functions, and if negative they are exponentionally growing or decaying functions.

To solve this equation we could start with for example the Runge-Kutta method or various improvements to Euler's method, as discussed in the previous chapter. Then we would need to transform this equation to a set of coupled first-order equations. We could however start with the discretized version for the second derivative. We discretise our equation and introduce a step length $h = (b-a)/N$, with N being the number of equally spaced mesh points. Our discretised second derivative reads at a step $x_i = a + ih$ with $i = 0, 1, \ldots$

$$y_i'' = \frac{y_{i+1} + y_{i-1} - 2y_i}{h^2} + O(h^2),$$

leading to a discretised differential equation

$$\frac{y_{i+1} + y_{i-1} - 2y_i}{h^2} + O(h^2) + k_i^2 y_i = F_i.$$

Recall that the fourth-order Runge-Kutta method has a local error of $O(h^4)$.

Since we want to integrate our equation from $x_0 = a$ to $x_N = b$, we rewrite it as

$$y_{i+1} \approx -y_{i-1} + y_i\left(2 - h^2k_i^2 + h^2F_i\right). \tag{9.3}$$

Starting at $i = 1$ we have after one step

$$y_2 \approx -y_0 + y_1\left(2 - h^2k_1^2 + h^2F_1\right).$$

Irrespective of method to approximate the second derivative, this equation uncovers our first problem. While $y_0 = y(a) = 0$, our function value y_1 is unknown, unless we have an analytic expression for $y(x)$ at $x = 0$. Knowing y_1 is equivalent to knowing y' at $x = 0$ since the first derivative is given by

$$y_i' \approx \frac{y_{i+1} - y_i}{h}.$$

This means that we have $y_1 \approx y_0 + hy_0'$.

9.2.1 Improved approximation to the second derivative, Numerov's method

Before we proceed, we mention how to improve the local truncation error from $O(h^2)$ to $O(h^6)$ without too many additional function evaluations.

Our equation is a second order differential equation without any first order derivatives. Let us also for the sake of simplicity assume that $F(x) = 0$. Numerov's method is designed to solve such an equation numerically, achieving a local truncation error $O(h^6)$.

We start with the Taylor expansion of the desired solution

$$y(x+h) = y(x) + hy^{(1)}(x) + \frac{h^2}{2!}y^{(2)}(x) + \frac{h^3}{3!}y^{(3)}(x) + \frac{h^4}{4!}y^{(4)}(x) + \cdots$$

Here $y^{(n)}(x)$ is a shorthand notation for the nth derivative $d^n y/dx^n$. Because the corresponding Taylor expansion of $y(x-h)$ has odd powers of h appearing with negative signs, all odd powers cancel when we add $y(x+h)$ and $y(x-h)$

$$y(x+h) + y(x-h) = 2y(x) + h^2 y^{(2)}(x) + \frac{h^4}{12}y^{(4)}(x) + O(h^6).$$

We obtain

$$y^{(2)}(x) = \frac{y(x+h) + y(x-h) - 2y(x)}{h^2} - \frac{h^2}{12}y^{(4)}(x) + O(h^6).$$

To eliminate the fourth-derivative term we apply the operator $(1 + \frac{h^2}{12}\frac{d^2}{dx^2})$ on the differential equation

$$y^{(2)}(x) + \frac{h^2}{12}y^{(4)}(x) + k^2(x)y(x) + \frac{h^2}{12}\frac{d^2}{dx^2}\left(k^2(x)y(x)\right) \approx 0.$$

In this expression the $y^{(4)}$ terms cancel. To treat the general x dependence of $k^2(x)$ we approximate the second derivative of $(k^2(x)y(x)$ by

$$\frac{d^2(k^2y(x))}{dx^2} \approx \frac{\left(k^2(x+h)y(x+h) + k^2(x)y(x)\right) + \left(k^2(x-h)y(x-h) + k^2(x)y(x)\right)}{h^2}.$$

We replace then $y(x+h)$ with the shorthand y_{i+1} (and similarly for the other variables) and obtain a final discretised algorithm for obtaining y_{i+1}

$$y_{i+1} = \frac{2\left(1 - \frac{5}{12}h^2k_i^2\right)y_i - \left(1 + \frac{1}{12}h^2k_{i-1}^2\right)y_{i-1}}{1 + \frac{h^2}{12}k_{i+1}^2} + O(h^6),$$

where $x_i = ih$, $k_i = k(x_i = ih)$ and $y_i = y(x_i = ih)$ etc.

It is easy to add the term F_i since we need only to take the second derivative. The final algorithm reads then

$$y_{i+1} = \frac{2\left(1 - \frac{5}{12}h^2k_i^2\right)y_i - \left(1 + \frac{1}{12}h^2k_{i-1}^2\right)y_{i-1}}{1 + \frac{h^2}{12}k_{i+1}^2} + \frac{h^2}{12}\left(F_{i+1} + F_{i-1} - 2F_i\right) + O(h^6).$$

Starting at $i = 1$ results in, using the boundary condition $y_0 = 0$,

$$y_2 = \frac{2\left(1 - \frac{5}{12}h^2k_1y_1\right) - \left(1 + \frac{1}{12}h^2k_0^2y_0\right)}{1 + \frac{h^2}{12}k_2^2} + \frac{h^2}{12}\left(F_2 + F_0 - 2F_1\right) + O(h^6).$$

This equation carries a local truncation error proportional to h^6. This is an order better than the fourth-order Runge-Kutta method which has a local error proportional to h^5. The global for the fourth-order Runge-Kutta is proportional to h^4 while Numerov's method has an error proportional to h^5. With few additional function evulations, we have achieved an increased accuracy.

But even with an improved accuracy we end up with one unknown on the right hand side, namely y_1. The value of y_1 can again be determined from the derivative at y_0, or by a good guess on its value. We need therefore an additional constraint on our set of equations before we start. We could then add to the boundary conditions

$$y(a) = \alpha,\ y(b) = \beta,$$

the requirement $y'(a) = \delta$, where δ could be an arbitrary constant. In quantum mechanical applications with homogenous differential equations the normalization of the solution is normally not known. The choice of the constant δ can therefore reflect specific symmetry requirements of the solution.

9.2.2 Wave equation with constant acceleration

We start with a well-known problem from mechanics, that of a whirling string or rope fixed at both ends. We could think of this as an idealization of a jumping rope and ask questions about its shape as it spins. Obviously, in deriving the equations we will make several assumptions in order to obtain an analytic solution. However, the general differential equation it leads to, with added complications not allowing an analytic solution, can be solved numerically. We discuss the shooting methods as one possible numerical approach in the next section.

Our aim is to arrive at a differential equation which takes the following form

$$y'' + \lambda y = 0;\ y(0) = 0,\ y(L) = 0,$$

where L is the length of the string and λ a constant or function of the variable x to be defined below.

We derive an equation for $y(x)$ using Newton's second law $F = ma$ acting on a piece of the string with mass $\rho\Delta x$, where ρ is the mass density per unit length and Δx is small displacement in the interval $x, x+\Delta x$. The change Δx is our step length.

We assume that the only force acting on this string element is a constant tension T acting on both ends. The net vertical force in the positive y-direction is

$$F = Tsin(\theta + \Delta\theta) - Tsin(\theta) = Tsin(\theta_{i+1}) - Tsin(\theta_i).$$

For the angles we employ a finite difference approximation

$$sin(\theta_{i+1}) = \frac{y_{i+1} - y_i}{\Delta x} + O(\Delta x^2).$$

Using Newton's second law $F = ma$, with $m = \rho\Delta x = \rho h$ and a constant angular velocity ω which relates to the acceleration as $a = -\omega^2 y$ we arrive at

$$T\frac{y_{i+1} + y_{i-1} - 2y_i}{\Delta x^2} \approx -\rho\omega^2 y,$$

and taking the limit $\Delta x \to 0$ we can rewrite the last equation as

$$Ty'' + \rho\omega^2 y = 0,$$

and defining $\lambda = \rho\omega^2/T$ and imposing the condition that the ends of the string are fixed we arrive at our final second-order differential equation with boundary conditions

$$y'' + \lambda y = 0;\ y(0) = 0,\ y(L) = 0.$$

The reader should note that we have assumed a constant acceleration. Replacing the constant acceleration with the second derivative of y as function of both position and time, we arrive at the well-known wave equation for $y(x,t)$ in $1+1$ dimension, namely

$$\frac{\partial^2 y}{\partial t^2} = \lambda \frac{\partial^2 y}{\partial x^2}.$$

We discuss the solution of this equation in chapter 10.

If $\lambda > 0$ the above wave equation has a solution of the form

$$y(x) = Acos(\alpha x) + Bsin(\alpha x),$$

and imposing the boundary conditions results in an infinite sequence of solutions of the form

$$y_n(x) = sin(\frac{n\pi x}{L}),\ n = 1,2,3,\ldots$$

with eigenvalues

$$\lambda_n = \frac{n^2\pi^2}{L^2},\ n = 1,2,3,\ldots$$

For $\lambda = 0$ we have

$$y(x) = Ax + B,$$

and due to the boundary conditions we have $y(x) = 0$, the trivial solution, which is not an eigenvalue of the problem. The classical problem has no negative eigenvalues, viz we cannot find a solution for $\lambda < 0$. The trivial solution means that the string remains in its equilibrium position with no deflection.

If we relate the constant angular speed ω to the eigenvalues λ_n we have

$$\omega_n = \sqrt{\frac{\lambda_n T}{\rho}} = \frac{n\pi}{L}\sqrt{\frac{T}{\rho}},\ n = 1,2,3,\ldots,$$

resulting in a series of discretised critical speeds of angular rotation. Only at these critical speeds can the string change from its equilibrium position.

There is one important observation to made here, since later we will discuss Schrödinger's equation. We observe that the eigenvalues and solutions exist only for certain discretised values $\lambda_n, y_n(x)$. This is a consequence of the fact that we have imposed boundary conditions. Thus, the boundary conditions, which are a consequence of the physical case we wish to explore, yield only a set of possible solutions. In quantum physics, we would say that the eigenvalues λ_n are quantized, which is just another word for discretised eigenvalues.

We have then an analytic solution

$$y_n(x) = sin(\frac{n\pi x}{L}),$$

from

$$y'' + \frac{n^2\pi^2}{L^2}y = 0;\ y(0) = 0,\ y(1) = 0.$$

Choosing $n = 4$ and $L = 1$, we have $y(x) = sin(4\pi x)$ as our solution. The derivative is obviously $4\pi cos(\pi x)$. We can start to integrate our equation using the exact expression for the derivative at y_1. This yields

$$y_2 \approx -y_0 + y_1\left(2 - h^2k_1^2 + h\right) = 4h\pi cos(4\pi x_0)\left(2 - 16h^2\pi^2\right) = 4\pi\left(2 - 16h^2\pi^2\right).$$

If we split our interval $x \in [0,1]$ into 10 equally spaced points we arrive at the results displayed in Table 9.1. We note that the error at the endpoint is much larger than the chosen mathematical approximation $O(h^2)$, resulting in an error of approximately 0.01. We would have expected a smaller error. We can obviously get better precision by increasing the number of integration points, but it would not cure the increasing discrepancy we see towards the endpoints.

Table 9.1 Integrated and exact solution of the differential equation $y'' + \lambda y = 0$ with boundary conditions $y(0) = 0$ and $y(1) = 0$.

$x_i = ih$	$sin(\pi x_i)$	$y(x_i)$
0.000000E+00	0.000000E+00	0.000000E+00
0.100000E+00	0.951057E+00	0.125664E+01
0.200000E+00	0.587785E+00	0.528872E+00
0.300000E+00	-.587785E+00	-.103405E+01
0.400000E+00	-.951056E+00	-.964068E+00
0.500000E+00	0.268472E-06	0.628314E+00
0.600000E+00	0.951057E+00	0.122850E+01
0.700000E+00	0.587785E+00	-.111283E+00
0.800000E+00	-.587786E+00	-.127534E+01
0.900000E+00	-.951056E+00	-.425460E+00
0.100000E+01	0.000000E+00	0.109628E+01

With $N = 100$, we have $0.829944E-02$ at $x = 1.0$, while the error is $\sim 10^{-4}$ with 100 integration points.

It is also important to notice that in general we do not know the eigenvalue and the eigenfunctions, except some of their limiting behaviors close to the boundaries. One method for searching for these eigenvalues is to set up an iterative process. We guess a trial eigenvalue and generate a solution by integrating the differential equation as an initial value problem, as we did above except that we have here the exact solution. If the resulting solution does not satisfy the boundary conditions, we change the trial eigenvalue and integrate again. We repeat this process until a trial eigenvalue satisfies the boundary conditions to within a chosen numerical error. This approach is what constitutes the so-called shooting method.

Upon integrating to our other boundary, $x = 1$ in the above example, we obtain normally a non-vanishing value for $y(1)$, since the trial eigenvalue is normally not the correct one. We can then readjust the guess for the eigenvalue and integrate and repeat this process till we obtain a value for $y(1)$ which agrees to within the precision we have chosen. As we will show in the next section, this results in a root-finding problem, which can be solved with for example the bisection or Newton methods discussed in chapter 4.

The example we studied here hides however an important problem. Our two solutions are rather similar, they are either represented by a $sin(x)$ form or a $cos(x)$ solution. This means that the solutions do not differ dramatically in behavior at the boundaries. Furthermore, the wave function is zero beyond the boundaries. For a quantum mechanical system, we would get the same solutions if a particle is trapped in an infinitely high potential well. Then the wave function cannot exist outside the potential. However, for a finite potential well, there is always a quantum mechanical probability that the particle can be found outside the classical region. The classical region defines the so-called turning points, viz points from where a classical solution cannot exist. These turning points are useful when we want to solve quantum mechanical problems.

Let us however perform our brute force integration for another differential equation as well, namely that of the quantum mechanical harmonic oscillator.

The situation worsens dramatically now. We have then a one-dimensional differential equation of the type, see Eq. (14.6), (all physical costants are set equal to one, that is $m = c = \hbar = k = 1$)

$$-\frac{1}{2}\frac{d^2y}{dx^2} + \frac{1}{2}x^2y = \varepsilon y; \ -\infty < x < \infty,$$

with boundary conditions $y(-\infty) = y(\infty) = 0$. For the lowest lying state, the eigenvalue is $\varepsilon = 1/2$ and the eigenfunction is

Table 9.2 Integrated and exact solution of the differential equation $-y''+x^2y=2\varepsilon y$ with boundary conditions $y(-\infty)=0$ and $y(\infty)=0$.

$x_i = ih$	$\exp(-x^2/2)$	$y(x_i)$
-.100000E+02	0.192875E-21	0.192875E-21
-.800000E+01	0.126642E-13	0.137620E-13
-.600000E+01	0.152300E-07	0.157352E-07
-.400000E+01	0.335462E-03	0.331824E-03
-.200000E+01	0.135335E+00	0.128549E+00
0.000000E-00	0.100000E+01	0.912665E+00
0.200000E+01	0.135335E+00	0.118573E+00
0.400000E+01	0.335463E-03	-.165045E-01
0.600000E+01	0.152300E-07	-.250865E+03
0.800000E+01	0.126642E-13	-.231385E+09
0.900000E+01	0.257677E-17	-.101904E+13

$$y(x)=\left(\frac{1}{\pi}\right)^{1/4}\exp(-x^2/2).$$

The reader should observe that this solution is imposed by the boundary conditions, which again follow from the quantum mechanical properties we require for the solution. We repeat the integration exercise which we did for the previous example, starting from a large negative number ($x_0=-10$, which gives a value for the eigenfunction close to zero) and choose the lowest energy and its corresponding eigenfunction. We obtain for y_2

$$y_2\approx -y_0+y_1\left(2+h^2x^2-h^2\right),$$

and using the exact eigenfunction we can replace y_1 with the derivative at x_0. We use now $N=1000$ and integrate our equation from $x_0=-10$ to $x_N=10$. The results are shown in Table 9.2 for selected values of x_i. In the beginning of our integrational interval, we obtain an integrated quantity which resembles the analytic solution, but then our integrated solution simply explodes and diverges. What is happening? We started with the exact solution for both the eigenvalue and the eigenfunction!

The problem is due to the fact that our differential equation has two possible solution for eigenvalues which are very close ($-1/2$ and $+1/2$), either

$$y(x)\sim\exp(-x^2/2),$$

or

$$y(x)\sim\exp(x^2/2).$$

The boundary conditions, imposed by our physics requirements, rule out the last possibility. However, our algorithm, which is nothing but an approximation to the differential equation we have chosen, picks up democratically both solutions. Thus, although we start with the correct solution, when integrating we pick up the undesired solution. In the next subsections we discuss how to cure this problem.

9.2.3 Schrödinger equation for spherical potentials

We discuss the numerical solution of the Schrödinger equation for the case of a particle with mass m moving in a spherical symmetric potential.

The initial eigenvalue equation reads

$$\widehat{\mathbf{H}}\psi(r) = (\widehat{\mathbf{T}} + \widehat{\mathbf{V}})\psi(r) = E\psi(r). \tag{9.4}$$

In detail this gives

$$\left(-\frac{\hbar^2}{2m}\nabla^2 + V(r)\right)\psi(r) = E\psi(r). \tag{9.5}$$

The eigenfunction in spherical coordinates takes the form

$$\psi(r) = R(r)Y_l^m(\theta,\phi), \tag{9.6}$$

and the radial part $R(r)$ is a solution to

$$-\frac{\hbar^2}{2m}\left(\frac{1}{r^2}\frac{d}{dr}r^2\frac{d}{dr} - \frac{l(l+1)}{r^2}\right)R(r) + V(r)R(r) = ER(r). \tag{9.7}$$

Then we substitute $R(r) = (1/r)u(r)$ and obtain

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}u(r) + \left(V(r) + \frac{l(l+1)}{r^2}\frac{\hbar^2}{2m}\right)u(r) = Eu(r). \tag{9.8}$$

We introduce a dimensionless variable $\rho = (1/\alpha)r$ where α is a constant with dimension length and get

$$-\frac{\hbar^2}{2m\alpha^2}\frac{d^2}{d\rho^2}u(\rho) + \left(V(\rho) + \frac{l(l+1)}{\rho^2}\frac{\hbar^2}{2m\alpha^2}\right)u(\rho) = Eu(\rho). \tag{9.9}$$

In our case we are interested in attractive potentials

$$V(r) = -V_0 f(r), \tag{9.10}$$

where $V_0 > 0$ and analyze bound states where $E < 0$. The final equation can be written as

$$\frac{d^2}{d\rho^2}u(\rho) + k(\rho)u(\rho) = 0, \tag{9.11}$$

where

$$\begin{aligned} k(\rho) &= \gamma\left(f(\rho) - \frac{1}{\gamma}\frac{l(l+1)}{\rho^2} - \varepsilon\right) \\ \gamma &= \frac{2m\alpha^2 V_0}{\hbar^2} \\ \varepsilon &= \frac{|E|}{V_0} \end{aligned} \tag{9.12}$$

9.2.3.1 Schrödinger equation for a spherical box potential

Let us now specify the spherical symmetric potential to

$$f(r) = \begin{cases} 1 & \text{for} \quad r \le a \\ -0 & \quad\quad r > a \end{cases} \tag{9.13}$$

and choose $\alpha = a$. Then

$$k(\rho) = \gamma\begin{cases} 1 - \varepsilon - \frac{1}{\gamma}\frac{l(l+1)}{\rho^2} & \text{for} \quad r \le a \\ -\varepsilon - -\frac{1}{\gamma}\frac{l(l+1)}{\rho^2} & \quad\quad r > a \end{cases} \tag{9.14}$$

The eigenfunctions in Eq. (9.5) are subject to conditions which limit the possible solutions. Of importance for the present example is that $u(r)$ must be finite everywhere and $\int |u(r)|^2 d\tau$ must be finite. The last condition means that $rR(r) \longrightarrow 0$ for $r \longrightarrow \infty$. These conditions imply that $u(r)$ must be finite at $r = 0$ and $u(r) \longrightarrow 0$ for $r \longrightarrow \infty$.

9.2.3.2 Analysis of $u(\rho)$ at $\rho = 0$

For small ρ Eq. (9.11) reduces to

$$\frac{d^2}{d\rho^2}u(\rho) - \frac{l(l+1)}{\rho^2}u(\rho) = 0, \tag{9.15}$$

with solutions $u(\rho) = \rho^{l+1}$ or $u(\rho) = \rho^{-l}$. Since the final solution must be finite everywhere we get the condition for our numerical solution

$$u(\rho) = \rho^{l+1} \quad \text{for small } \rho \tag{9.16}$$

9.2.3.3 Analysis of $u(\rho)$ for $\rho \longrightarrow \infty$

For large ρ Eq. (9.11) reduces to

$$\frac{d^2}{d\rho^2}u(\rho) - \gamma\varepsilon u(\rho) = 0 \quad \gamma > 0, \tag{9.17}$$

with solutions $u(\rho) = \exp(\pm\gamma\varepsilon\rho)$ and the condition for large ρ means that our numerical solution must satisfy

$$u(\rho) = e^{-\gamma\varepsilon\rho} \quad \text{for large } \rho \tag{9.18}$$

As for the harmonic oscillator, we have two solutions at the boundaries which are very different and can easily lead to totally worng and even diverging solutions if we just integrate from one endpoint to the other. In the next section we discuss how to solve such problems.

9.3 Numerical procedure, shooting and matching

The eigenvalue problem in Eq. (9.11) can be solved by the so-called shooting methods. In order to find a bound state we start integrating, with a trial negative value for the energy, from small values of the variable ρ, usually zero, and up to some large value of ρ. As long as the potential is significantly different from zero the function oscillates. Outside the range of the potential the function will approach an exponential form. If we have chosen a correct eigenvalue the function decreases exponentially as $u(\rho) = e^{-\gamma\varepsilon\rho}$. However, due to numerical inaccuracy the solution will contain small admixtures of the undesireable exponential growing function $u(\rho) = e^{+\gamma\varepsilon\rho}$. The final solution will then become unstable. Therefore, it is better to generate two solutions, with one starting from small values of ρ and integrate outwards to some matching point $\rho = \rho_m$. We call that function $u^<(\rho)$. The next solution $u^>(\rho)$ is then obtained by integrating from some large value ρ where the potential is of no importance, and inwards to the same matching point ρ_m. Due to the quantum mechanical requirements the logarithmic derivative at the matching point ρ_m should be well defined. We obtain the following condition

$$\frac{\frac{d}{d\rho}u^<(\rho)}{u^<(\rho)} = \frac{\frac{d}{d\rho}u^>(\rho)}{u^>(\rho)} \quad \text{at} \quad \rho = \rho_m. \tag{9.19}$$

We can modify this expression by normalizing the function $u^<u^<(\rho_m) = Cu^>u^>(\rho_m)$. Then Eq. (9.19) becomes

$$\frac{d}{d\rho}u^<(\rho) = \frac{d}{d\rho}u^>(\rho) \quad \text{at} \quad \rho = \rho_m \tag{9.20}$$

For an arbitary value of the eigenvalue Eq. (9.19) will not be satisfied. Thus the numerical procedure will be to iterate for different eigenvalues until Eq. (9.20) is satisfied.

We can calculate the first order derivatives by

$$\begin{aligned}\frac{d}{d\rho}u^<(\rho_m) &\approx \frac{u^<(\rho_m) - u^<(\rho_m - h)}{h} \\ \frac{d}{d\rho}u^>(\rho_m) &\approx \frac{u^>(\rho_m) - u^>(\rho_m + h)}{h}\end{aligned} \tag{9.21}$$

Thus the criterium for a proper eigenfunction will be

$$f = u^>(\rho_m + h) - u^<(\rho_m - h) = 0. \tag{9.22}$$

9.3.1 Algorithm for solving Schrödinger's equation

Here we outline the solution of Schrödinger's equation as a common differential equation but with boundary conditions. The method combines shooting and matching. The shooting part involves a guess on the exact eigenvalue. This trial value is then combined with a standard method for root searching, e.g., the secant or bisection methods discussed in chapter 4.

The algorithm could then take the following form

- Initialise the problem by choosing minimum and maximum values for the energy, $E_{\min}$ and $E_{\max}$, the maximum number of iterations max_iter and the desired numerical precision.
- Search then for the roots of the function f, where the root(s) is(are) in the interval $E \in [E_{\min}, E_{\max}]$ using for example the bisection method. Newton's method, also discussed in chapter 4 requires an analytic expression for f. A possible approach is to use the standard bisection method for localizing the eigenvalue and then use the secant method to obtain a better estimate.

  The pseudocode for such an approach can be written as

```
do {
   i++;
   e = (e_min+e_max)/2.; /* bisection */
   if ( f(e)*f(e_max) > 0 ) {
     e_max = e;      /* change search interval */
   }
   else {
     e_min = e;
   }
} while ( (fabs(f(e) > convergence_test) !! (i <= max_iterations))
```

  The use of a root-searching method forms the shooting part of the algorithm. We have however not yet specified the matching part.
- The matching part is given by the function $f(e)$ which receives as argument the present value of E. This function forms the core of the method and is based on an integration of Schrödinger's equation from $\rho = 0$ and $\rho = \infty$. If our choice of E satisfies Eq. (9.22)

we have a solution. The matching code is given below. To choose the matching point it is convenient to start integrating inwards, that is from the large r-values. When the wave function turns, we use that point to define the matching point. The reason for this is that we start integrating from a region which corresponds normally to classically forbidden ones, and integrating into such regions leads normally to inaccurate solutions and the pick up of the undesired solutions. The consequence is that the solution diverges. We can therefore define as a matching point the classical turning point and start to integrate from large r-values. In the absence of such a point, we can use the point where the wave function turns.

The function $f(E)$ above receives as input a guess for the energy. In the version implemented below, we use the standard three-point formula for the second derivative, namely

$$f_0'' \approx \frac{f_h - 2f_0 + f_{-h}}{h^2}.$$

We leave it as an exercise to the reader to implement Numerov's algorithm.

```
//
// The function
//     f()
// calculates the wave function at fixed energy eigenvalue.
//

double f(double step, int max_step, double energy, double *w, double *wf)
{
  int    loop, loop_1,match;
  double const sqrt_pi = 1.77245385091;
  double fac, wwf, norm;
// adding the energy guess to the array containing the potential
  for(loop = 0; loop <= max_step; loop ++) {
    w[loop] = (w[loop] - energy) * step * step + 2;
  }
// integrating from large r-values
  wf[max_step] = 0.0;
  wf[max_step - 1] = 0.5 * step * step;
// search for matching point
  for(loop = max_step - 2; loop > 0; loop--) {
    wf[loop] = wf[loop + 1] * w[loop + 1] - wf[loop + 2];
    if(wf[loop] <= wf[loop + 1]) break;
  }
  match = loop + 1;
  wwf  = wf[match];
// start integrating up to matching point from r =0
  wf[0] = 0.0;
  wf[1] = 0.5 * step * step;
  for(loop = 2; loop <= match; loop++) {
    wf[loop] = wf[loop -1] * w[loop - 1] - wf[loop - 2];
    if(fabs(wf[loop]) > INFINITY) {
      for(loop_1 = 0; loop_1 <= loop; loop_1++) {
        wf[loop_1] /= INFINITY;
      }
    }
  }
// now implement the test of Eq. (10.25)
  return (wf[match-1]-wf[match+1]);
} // End: funtion plot()
```

The approach we have described here suffers from the fact that the matching point is not properly defined. Using a Green's function approach we can easily determine the matching

point as the midpoint of the integration interval and compute safely the solution. This is the topic of the next section.

9.4 Green's function approach

A slightly different approach, which however still keeps the matching procedure discussed above, is based on the computation of the Green's function and its relation to the solution of a differential equation with boundary values.

Consider the differential equation

$$-u(x)'' = f(x), \qquad x \in (0,1), \qquad u(0) = u(1) = 0, \tag{9.23}$$

and using the fundamental theorem of calculus, there is a constant c_1 such that

$$u(x) = c_1 + \int_0^x u'(y)dy,$$

and a constant c_2

$$u'(y) = c_2 + \int_0^y u''(z)dz.$$

This is true for any twice continuously differentiable function u

If u satisfies the above differential equation we have then

$$u'(y) = c_2 - \int_0^y f(z)dz.$$

which inserted into the equation for u gives

$$u(x) = c_1 + c_2 x - \int_0^x \left(\int_0^y f(z)dz \right) dy,$$

and defining

$$F(y) = \int_0^y f(z)dz,$$

and performing an integration by parts we obtain

$$\int_0^x F(y)dy = \int_0^x \left(\int_0^y f(z)dz \right) dy = \int_0^x (x-y) f(y)dy.$$

This gives us

$$u(x) = c_1 + c_2 x - \int_0^x (x-y) f(y)dy.$$

The boundary condition $u(0) = 0$ yields $c_1 = 0$ and $u(1) = 0$, resulting in

$$c_2 = \int_0^1 (1-y) f(y)dy,$$

meaning that we can write the solution as

$$u(x) = x \int_0^1 (1-y) f(y)dy - \int_0^x (x-y) f(y)dy$$

The solution to our differential equation can be represented in a compact way using the so-called Green's functions, which are also solutions to our differential equation with $f(x) = 0$.

If we then define the Green's function as

$$G(x,y) = \begin{cases} y(1-x) \text{ if } 0 \leq y \leq x \\ x(1-y) \text{ if } x \leq y \leq 1 \end{cases}$$

we can write the solution as

$$u(x) = \int_0^1 G(x,y) f(y) dy,$$

The Green's function, see for example Refs. [50, 51] is

1. continuous
2. it is symmetric in the sense that $G(x,y) = G(y,x)$
3. it has the properties $G(0,y) = G(1,y) = G(x,0) = G(x,1) = 0$
4. it is a piecewise linear function of x for fixed y and vice versa. G' is discontinuos at $y = x$.
5. $G(x,y) \geq 0$ for all $x,y \in [0,1]$
6. it is the solution of the differential equation

$$\frac{d^2}{dx^2} G(x,y) = -\delta(x-y).$$

The Green's function can now be used to define the solution before and after a specific matching point in the domain.

The Green's function satisfies the homogeneous equation for $y \neq x$ and its derivative is discontinuous at $x = y$. We can see this if we integrate the differential equation

$$\frac{d^2}{dx^2} G(x,y) = -\delta(x-y)$$

from $x = y - \varepsilon$ to $x = y + \varepsilon$, with ε as an infinitesmally small number. We obtain then

$$\frac{dG}{dx}|_{x=y+\varepsilon} - \frac{dG}{dx}|_{x=y-\varepsilon} = 1.$$

The problem is obvioulsy to find G.

We can obtain this by considering two solutions of the homogenous equation. We choose a general domain $x \in [a,b]$ with a boundary condition on the general solution $u(a) = u(b) = 0$.

One solution is obtained by integrating from a to b (called $u_<$) and one by integrating inward from b to a, labelled $u_>$.

Using the continuity requirement on the function and its derivative we can compute the Wronskian [50, 51]

$$W = \frac{du_>}{dx} u_< - \frac{du_<}{dx} u_>,$$

and using

$$\frac{dG}{dx}|_{x=y+\varepsilon} - \frac{dG}{dx}|_{x=y-\varepsilon} = 1,$$

and one can then show that the Green's function reads

$$G(x,y) = u_<(x_<) u_>(x_>), \tag{9.24}$$

where $x_<$ is defined for $x = y - \varepsilon$ and $x_> = y + \varepsilon$. Using the definition of the Green's function in Eq. (9.24) we can now solve Eq. (9.23) for $x \in [a,b]$ using

$$u(x) = u^{>}(x)\int_a^x u^{<}(x')f(x')dx' + u^{<}(x)\int_x^b u^{>}(x')f(x')dr' \tag{9.25}$$

The algorithm for solving Eq. (9.23) proceed now as follows: Your task is to choose a matching point, say the midpoint, and then compute the Greens' function after you have used Numerov's algo to find u (inward and outward integration for all points). Find u integrating with the Green's function.

A possible algorithm could be phrased as follows:

- Compute the solution of the homogeneous part of Eq. (9.23) using Numerov's method. You should then have both the inward and the outward solutions.
- Compute the Wronskian at the matching point using

$$\frac{du}{dx} \approx \frac{u(x+h)-u(x+h)}{2h},$$

  for the first derivative and choose the matching point as the midpoint. You should try the stability of the solution by choosing other matching points as well.
- Compute then the outward integral of the Green's function approach, including the inhomogeneous term. For the integration one can employ for example Simpson's rule discussed in chapter 5.
- Compute thereafter the inward integral of the Green's function approach. Adding these two integrals gives the resulting wave function of Eq. (9.25).

An example of a code which performs all these steps is listed here

```
void wfn(Array<double,2> &k, Array<double,2> &ubasis, Array<double,1> &r, Array<double,2>
    &F,Array<double,1> &uin, Array<double,1> &uout)
{
 int    loop, loop_1, midpoint, j;
 double norm, wronskian, sum, term;

 ubasis=0;uin=0;uout=0;

 // Compute inwards homogenous solution
 for(j=0;j<mat_size;j++){

 uin(max_step) = 0.0;
 uin(max_step-1) = 1.0E-10;

 for(loop = max_step-2; loop >= 0; loop--) {
  uin(loop) = (2.0*(1.0-5.0*k(loop+1,j)/12.0)* uin(loop+1)- (1.0+k(loop+2,j)/12.0)*
       uin(loop+2))/(1.0+k(loop,j)/12.0);
 }

 // Compute outwards homogenous solution

 uout(0) = 0.0;
 uout(1) = 1.0E-10;

 for(loop = 2; loop <= max_step; loop++) {
  uout(loop) = (2.0*(1.0-5.0*k(loop-1,j)/12.0)* uout(loop-1)-
       (1.0+k(loop-2,j)/12.0)*uout(loop-2))/(1.0+k(loop,j)/12.0);
 }
```

```
 // Compute Wronskian at matching mid-point

 midpoint = (max_step)/2;
 // first part of Wronskian
 wronskian = (uin(midpoint+1)-uin(midpoint-1))* uout(midpoint)/(2*step);
 // second part
 wronskian -= (uout(midpoint+1)-uout(midpoint-1))* uin(midpoint)/(2*step);

 // Outward integral of Greens function

 sum = 0.0;
 for(loop = 0; loop <= max_step; loop++) {
  term = uout(loop)*F(loop,j);
  sum += term;
  ubasis(loop,j) = uin(loop)*sum*step;
 }

 // Inward integral of Greens function

 sum = 0.0;
 for(loop = max_step; loop >= 0; loop--) {
  term = uin(loop)*F(loop,j);
  sum += term;
  ubasis(loop,j) = (ubasis(loop,j)+uout(loop)*sum*step)/wronskian;
 }

 // Compute the norm

 for(loop = 0, norm = 0.0; loop <= max_step; loop++) {
  norm += ubasis(loop,j)*ubasis(loop,j) * step;//wf[loop] * step;//fabs(wf[loop] *
      step);//wf[loop]* wf[loop] * step;
 }

 if(fabs(norm) < 1.0e-15) {
  printf("\n\nError in norm in function wfn(): ");
  //exit(1);
 }

 norm = 1./sqrt(norm); //

 for(loop = 0; loop <= max_step; loop++) {
  ubasis(loop,j) *= norm;
 }
 }
} // End: funtion wfn()
```

9.5 Exercises

9.1. In this project we will solve the one-dimensional Poissson equation

$$-u''(x) = f(x), \qquad x \in (0,1), \qquad u(0) = u(1) = 0.$$

with the inhomogeneous given by $f(x) = 100e^{-10x}$. This equation has $u(x) = 1-(1-e^{-10})x - e^{-10x}$ as analytic solution.

Write a code which solves the above differential equation using Numerov's algorithm and the Green's function method. Can you find an analytic expression for the Green's function?

Compare these results with those obtained by solving the above differential equation as a set of linear equations, as done in chapter 6. Which method would you prefer?

9.2. We are going to study the solution of the Schrödinger equation for a system with a neutron and a proton (the deuteron) for a simple box potential. This potential will later be replaced with a realistic one fitted to experimental phase shifts.

We begin our discussion of the Schrödinger equation with the neutron-proton (deuteron) system with a box potential $V(r)$. We define the radial part of the wave function $R(r)$ and introduce the definition $u(r) = rR(R)$ The radial part of the SE for two particles in their center-of-mass system and with orbital momentum $l = 0$ is then

$$-\frac{\hbar^2}{2m}\frac{d^2u(r)}{dr^2} + V(r)u(r) = Eu(r), \tag{9.26}$$

with

$$m = 2\frac{m_p m_n}{m_p + m_n}, \tag{9.27}$$

where m_p and m_n are the masses of the proton and neutron, respectively. We use here $m = 938$ MeV. Our potential is defined as

$$V(r) = \begin{cases} 0 & r > a \\ -V_0 & 0 < r \leq a \\ \infty & r \leq 0 \end{cases}, \tag{9.28}$$

displayed in Fig 9.1.

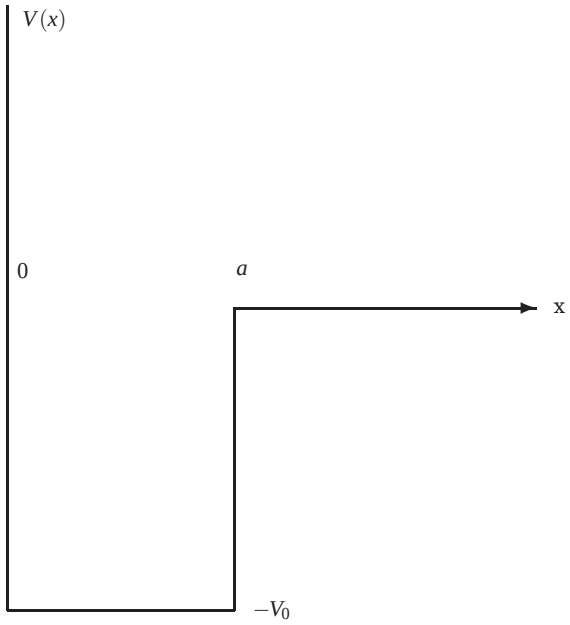

Fig. 9.1 Example of a finite box potential with value $-V_0$ in $0 < x \leq a$, infinitely large for $x \leq 0$ and zero else.

Bound states correspond to negative energy E and scattering states are given by positive energies. The SE takes the form (without specifying the sign of E)

$$\frac{d^2u(r)}{dr^2}+\frac{m}{\hbar^2}(V_0+E)u(r)=0 \qquad r<a, \tag{9.29}$$

and

$$\frac{d^2u(r)}{dr^2}+\frac{m}{\hbar^2}Eu(r)=0 \qquad r>a. \tag{9.30}$$

1. We are now going to search for eventual bound states, i.e., $E<0$. The deuteron has only one bound state at energy $E=-2.223$ MeV. Discuss the boundary conditions on the wave function and use these to show that the solution to the SE is

$$u(r)=Asin(kr) \qquad r<a, \tag{9.31}$$

and

$$u(r)=B\exp(-\beta r) \qquad r>a, \tag{9.32}$$

where A and B are constants. We have also defined

$$k=\sqrt{m(V_0-|E|)}/\hbar, \tag{9.33}$$

and

$$\beta=\sqrt{m|E|}/\hbar. \tag{9.34}$$

Show then, using the continuity requirement on the wave function that at $r=a$ you obtain the transcendental equation

$$kcot(ka)=-\beta. \tag{9.35}$$

2. Insert values of $V_0=60$ MeV and $a=1.45$ fm (1 fm = 10^{-15} m) and make a plot of Eq. (9.35) as function of energy E in order to find eventual eigenvalues. See if these values result in a bound state for E.
   When you have localized on your plot the point(s) where Eq. (9.35) is satisfied, obtain a numerical value for E using for example Newton-Raphson's method or similar methods, see chapter 4. To use these functions you need to provide the function $kcot(ka)+\beta$ and its derivative as function of E.
   What is smallest possible value of V_0 which gives one bound state only?
3. Write a program which implements the Green's function method using Numerov's method for this potential and find the lowest eigenvalue for the case that V_0 supports only one bound state. Use the results from b) to guide your choice of trial eigenvalues. Plot the wave function and discuss your results.
4. We turn now to a fitted interaction which reproduces the low-lying phase shifts for scattering between a proton and neutron. The parametrized version of this potential fits the experimental phase-shifts. It is given by

$$V(r)=V_a\frac{e^{-ax}}{x}+V_b\frac{e^{-bx}}{x}+V_c\frac{e^{-cx}}{x} \tag{9.36}$$

   with $x=\mu r$, $\mu=0.7$ fm^{-1} (the inverse of the pion mass), $V_a=-10.463$ MeV and $a=1$, $V_b=-1650.6$ MeV and $b=4$ and $V_c=6484.3$ MeV and $c=7$. Replace the box potential from point c) and find the wave function and possible eigenvalues for this potential as well. Discuss your results.