22 January 2025 – PHYS 6260
Prof. John Wise
Chapter 8: Ordinary Differential Equations

8.6: Boundary Value Problems

So far, we have limited ourselves to initial value problems, which are the most common in physics, but there are another class of problems with specific constraints on their boundaries, called *boundary value problems*.

One simple example is projectile motion with only gravity acting on an object:

$$\frac{d^2x}{dt^2} = -g, \tag{1}$$

We know how to solve this second-order equation by creating two first-order equations. For initial value problems, one would specify an initial position and velocity. However, what if we only knew information about its initial position and final position? Here the object would have height $x = 0$ at $t = 0$ and at some later unknown time $t = t_1$. Our goal would be to determine a solution that satisfies these initial and final conditions, i.e. *boundary values*. They are more difficult to solve than initial value problems, but we can tackle these problems by combining two previously studied techniques that we will cover now.

The shooting method

The shooting method is a trial-and-error technique that uses guesses at the initial values (such as position and velocity of a projectile), solves the ODE with standard techniques as discussed in previous lectures, and then compares them with the final boundary conditions. We then adjust the initial values accordingly, test again, and repeat until the final estimated values are within some tolerance of the given final values, as depicted in Figure 1.

For example, a projectile is launched at some unknown velocity v at $t = 0$ and lands at a later time $t = 10$. We make an initial guess and calculate when it lands. It probably didn't land at $t = 10$. So we decrease (increase) v if the previous solution landed too late (soon). We repeat as necessary until we converge on the final value. In principle, there exists a function $x = f(v)$ that tells us the height x of the object at time $t_1 = 10$. We don't know what this function is, but we can calculate it for any given value of v. Our goal in solving the boundary value problem is to find the value of v that makes the function zero, in other words, root finding.

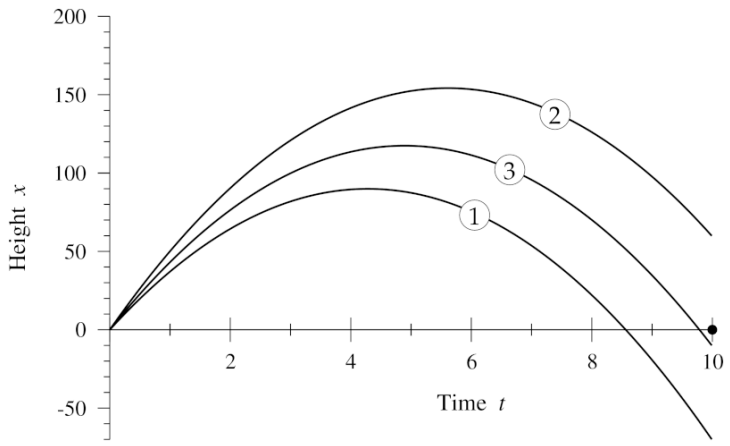

Figure 1: **The shooting method** allows us to match boundary conditions at the beginning and end of a solution. First, we undershoot our target of $t = 10$ on our first guess (trajectory 1). On the second guess, we overshoot. On the third, we are closer, but still not perfect.

> The shooting method combines a standard method of solving ODEs and a root finding method. For example, we could use a combination of fourth-order Runge-Kutta and binary search to solve boundary value problems.

Example 8.8: Vertical position of a thrown ball

Let's solve this exact problem that we were discussed just right now. For projectile vertical motion, we have two first-order ODEs:

$$\frac{dx}{dt} = y, \quad \frac{dy}{dt} = -g. \tag{2}$$

Let's solve this problem with fourth-order Runge-Kutta and binary search in the program below.

```
import numpy as np

g = 9.81          # Acceleration due to gravity
a = 0.0           # Initial time
b = 10.0          # Final time
N = 1000          # Number of Runge-Kutta steps
h = (b-a)/N       # Size of Runge-Kutta steps
target = 1e-10    # Target accuracy for binary search

# Function for Runge-Kutta calculation
```

```
def f(r):
    x = r[0]
    y = r[1]
    fx = y
    fy = -g
    return np.array([fx,fy],float)

# Function to solve the equation and calculate the final height
def height(v):
    r = np.array([0.0,v],float)
    for t in np.arange(a,b,h):
        k1 = h*f(r)
        k2 = h*f(r+0.5*k1)
        k3 = h*f(r+0.5*k2)
        k4 = h*f(r+k3)
        r += (k1+2*k2+2*k3+k4)/6
    return r[0]

# Main program performs a binary search
v1 = 0.01
v2 = 1000.0
h1 = height(v1)
h2 = height(v2)

while abs(h2-h1)>target:
    vp = (v1+v2)/2
    hp = height(vp)
#
# To be completed in class: binary search, given the two solutions, v1
# and v2.
#

V = (v1+v2)/2
print("The required initial velocity is %f m/s" % (v))
```

which returns an initial velocity of `49.05 m/s`, and we could, in principle, return the trajectory in addition to this initial speed.

The relaxation method

Another method to solve boundary value problems is the relaxation method that *defines a shape for the initial guess but most importantly matches the boundary conditions*. The initial guess is not the correct solution, but the shape can be successively modified to bring it closer

and closer to a solution of the given differential equation.

In a way, the relaxation method is the opposite of the shooting method. The shooting method starts with a correct solution to the differential equation that may not match the boundary conditions and modifies it until it does. The relaxation method starts with a solution that matches the boundary conditions, but may not be a correct solution to the equation.

The relaxation method is most commonly used to solve *partial differential equations*, which is the subject of the next chapter, so we will delay our discussion until then.

Eigenvalue problems

A special type of boundary value problem arises when the equation(s) are linear and homogeneous, meaning that every term in the equation is linear in the dependent variable. The Schrödinger equation is a good example,

$$-\frac{\hbar^2}{2m}\frac{d^2\phi}{dx^2} + V(x)\psi(x) = E\psi(x), \tag{3}$$

where $\psi(x)$ is the wave function, $V(x)$ is the potential energy at position x, and E is the total energy of the particle. Note how each term is linear in ψ. Let's consider the problem of a infinitely deep, square potential well: $V(x) = 0$ for $x = (0, L)$ and infinite elsewhere. This problem can be easily solved analytically, but we can compute it numerically.

Equation (3) can be written as two first-order differential equations,

$$\frac{d\psi}{dx} = \phi, \quad \frac{d\phi}{dx} = \frac{2m}{\hbar^2}[V(x) - E]\psi \tag{4}$$

We set know that $\psi = 0$ at $x = 0$ but the value of $\phi = d\psi/dx$ is unknown, so we make an initial guess. Figure 2 shows an example solution using fourth-order Runge-Kutta. From the potential, we know that $\phi(x)$ should be zero again at $x = L$ but it's nowhere close. However, we can adjust $\phi = d\psi/dx$ to search for the correct initial values. However, in linear and homogeneous equations, this method does not work. Because it is linear, if we double the initial value of ϕ, then the whole solution doubles, that is, $\psi(x) \to 2\psi(x)$. In other words, it retains its shape, meaning that no matter how much we adjust the initial value of ϕ we will never match the right boundary value.

The fundamental problem is that there is no general solution to this problem. The only solutions that exist have special values of E, the eigenvalues. So we vary the energy E to find each solution where $\psi = 0$ at $x = L$. We can think of the solution to the Schrödinger equation as giving us a function $f(E)$ equal to the value of the wave function at $x = L$, and we want to find the value of E that makes this function zero. As with the shooting method, we can find the root using previously discussed root finding methods.

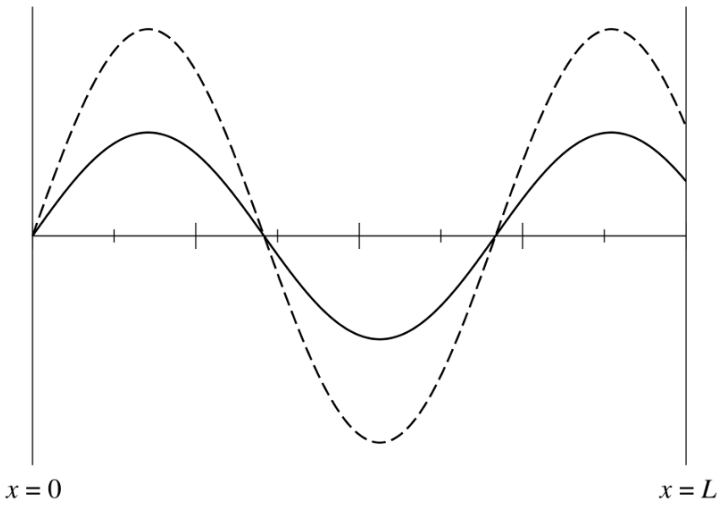

Figure 2: Solution of the Schrödinger equation in a square well for a general solution without considering the eigenvalues.

We haven't address the initial value of $\phi = d\psi/dx$ yet, but in the end, it doesn't make a difference. For the case in quantum mechanics, the solution of $\phi(x)$ is just normalized in the end ($\int |\psi(x)|^2 dx = 1$), so adjusting ϕ doesn't have an effect on the final normalized solution.

Example 8.9: Ground state energy in a square well

Let's calculate the ground state energy E of an electron in the square well in the previous example but with $L = 5.292 \times 10^{-11}$ m, the Bohr radius.

```
import numpy as np

# Constants
m = 9.1094e-31      # Mass of electron
hbar = 1.0546e-34   # Planck's constant over 2*pi
e = 1.6022e-19      # Electron charge
L = 5.2918e-11      # Bohr radius
N = 1000
h = L/N

# Potential function
def V(x):
    return 0.0

def f(r,x,E):
    psi = r[0]
```

```
    phi = r[1]
    fpsi = phi
    fphi = (2*m/hbar**2)*(V(x)-E)*psi
    return np.array([fpsi,fphi],float)

# Calculate the wavefunction for a particular energy
def solve(E):
    psi = 0.0
    phi = 1.0
    r = np.array([psi,phi],float)

    for x in np.arange(0,L,h):
        k1 = h*f(r,x,E)
        k2 = h*f(r+0.5*k1,x+0.5*h,E)
        k3 = h*f(r+0.5*k2,x+0.5*h,E)
        k4 = h*f(r+k3,x+h,E)
        r += (k1+2*k2+2*k3+k4)/6

    return r[0]

# Main program to find the energy using the secant method
E1 = 0.0
E2 = e
psi2 = solve(E1)

target = e/1000
while abs(E1-E2)>target:
#
# To be completed in class
#

print("E = %.6f eV" % (E2/e))
```

which gives the answer `134.286 eV`. The square well may be easy to solve analytically, but these numerical methods are useful when the potential has a non-trivial shape. Then energy levels can be easily computed once we have a general program that takes potential in some functional form.

These problems were all one-dimensional. However reality is three-dimensional, in which the Schrödinger equation becomes a partial differential equation, which we will study in the next chapter and lecture.