

Physics 305, Fall 2009
Assignment 4
Due Monday, September 21

In this assignment, you will develop a program to find roots of equations by the bisection method. You will apply that program to three physics examples. One of the goals of the week is that you should be able to solve these three examples by only changing the function whose roots you are seeking. By putting this function in a separate file (well, three files), you should be able to solve all three examples with only changes of the compile command.

You will write two separate programs to use that function. First, you will write a program to generate a table of values of the function, suitable as input for your graphing program (simpleplot, graph, xmgrace, or whatever you want to use). You will need to plot the function in order to see its behavior and plan your attack on the roots.

Second, you will write a program to start at some point x_0 and then search along in steps of Δx looking for a bracketed root. Once you have found a Δx region that contains a root, take that bracket and refine it with the bisection method to at least 5 significant figures. You should print the value of the root and the value of the function at the root (which may be non-zero because of numerical precision or your choice of stopping criterion). Your code should then continue stepping along looking for additional roots, up to a number N_{root} supplied by the user.

Your code should prompt the user for the values of x_0 , Δx , and N_{root} . You should develop your choices for these values based on the graph of the function. Your Δx has to be large enough that the linear search for brackets is efficient, but small enough that you don't have more than one root in a single step, as that would often be missed. Your choice of N_{root} should be based on the number of roots: you don't want to tell your code to keep looking for a second root when only one exists!

Your code should guard against infinite loops. There are two places where infinite loops could arise. First, in refining a root with bisection, you could reach a point where the function is non-zero and yet the computer cannot bisect any further because no floating point numbers exist between the two endpoints. You can avoid iterating forever in this situation by limiting the number of bisections to something like 30 or 50. Second, you may have supplied a x_0 or Δx that have sent the code looking for a root that doesn't exist. You can avoid this infinite loop by limiting yourself to some number of Δx steps, like 100 or 1000.

You should use double precision. Remember that scanf requires a format of "%lf" when reading a variable of type `double`.

Remember that a root is bracketed when the values of the function at the two end points

are of opposite sign. This can be easily tested as $f(x_1) * f(x_2) < 0$. Note that you have to be watchful for the case where one of the endpoints has $f(x) = 0$; if that happens, you've found the root and can stop!

Make sure that your programs are commented so that we can see what you have done and why.

Because this problem is moderately complicated, we strongly recommend that you build your programs in several steps, debugging each along the way. The first step is to cast the physical application as a root finding problem. In other words, you need to manipulate the equations into the form $f(x) = 0$. The next step is to write a C function (in a separate file) for the function $f(x)$. Then write the program to call that function and produce a table of outputs. Graph the function and double check that it looks like what you expected.

Then you're ready to proceed to the root finding. Your code to refine the root by bisection should be in a separate function (with arguments for the lower and upper bracket value and perhaps some tolerance choice), so that it can be called from the function that is doing the stepping and search for brackets. You should probably start with a `main()` program that is hardwired to search for a single root with a known bracket, so that you can debug the bisection function first. Then you can implement the bracket search into `main()`.

If this is confusing, you might want to start by writing an outline describing the different functions and how they'll fit together. Then solve the problem one step at a time!

The best implementations of bisection try to minimize the number of times the function f is called, on the assumption that f may be expensive to compute. This is easily done if you hold the values of f at the two endpoints in two variables, and update those as you update the endpoints when bisecting. That way you can use the value of f at a particular point multiple times without recomputing it. If this is confusing, don't worry about it at first; it may be more clear how to avoid recomputing f at a given x after you have the program working.

As a minor note for your program to generate the table of function values, you may want to prompt the user for the interval and spacing of the table. This prompting has the problem that when you redirect the output to a file, your printed prompt goes to the file instead of the screen, so the user doesn't know when or what to type. There are many possible work-arounds, but a simple one is to change the printing of the prompt to go to the `stderr` stream so that it prints on the screen instead of going to the file. This is easy to do: just replace your `printf()` calls for the prompt, e.g.,

```
printf(Please enter the lower bound for the table");
```

with

```
fprintf(stderr, "Please enter the lower bound for the table");
```

When you want to print the table, use `printf()` again so that those lines will go to `stdout` and be redirected to the file.

Keep your root finding program around. We will need it for at least one later assignment in this class, and it may come in handy in some of your other classes.

Use your program to solve the following problems.

1) When the wave equation is solved in spherical coordinates, the radial dependence of the solution involves a special function known as the spherical Bessel function $j_n(x)$. Here x is a rescaled radius. To solve the wave equation in a spherical cavity, we need to know the zeroes of these functions. The zeroth spherical Bessel function j_0 is simply $\sin(x)/x$, so its zeroes are trivial to find. The roots of the higher functions are harder to find.

For this problem, find the first three positive zeroes of the first spherical Bessel function

$$j_1(x) = \frac{\sin(x)}{x^2} - \frac{\cos(x)}{x}$$

to five significant figures.

Note: Bessel functions are actually in the standard C math library, just like `exp` and `sin`. But for this assignment please implement the expression above, so that you can see what is going on.

2) Ferromagnetism is the property in which the atomic spins of a material can spontaneously align, giving a large magnetic dipole moment even without an external magnetic field. Ferromagnets lose this ability at high temperature, as the heat causes the spins to misalign. Interesting, the transition from ferromagnetism to paramagnetism occurs as a phase transition at a temperature known as the Curie temperature.

The Ising model is a simple model for ferromagnetism. When the “mean field approximation” is used to analyze this model, for temperatures below the Curie temperature the magnetization is found by solving the equation

$$\tanh\left(\frac{6J}{k_B T} M\right) = M$$

where M is the magnetization (as a fraction of the maximum possible magnetization). J is the strength of the interatomic coupling, T is the temperature, and k_B is Boltzmann's constant. For temperatures below the Curie temperature, $6J/k_B T > 1$. Find M to five significant figures when $6J/k_B T = 1.2$. Find a nonzero solution — the trivial solution at $M = 0$ doesn't count. Note that we are just treating this as an algebra problem — you don't have to know what ferromagnetism or the Ising model is in order to do this problem.

3) Here is another real physics example where for the moment you don't have to understand how we got the equation. In elementary quantum mechanics, you may be asked to find the ground state energy for a particle moving in a one dimensional "square well" potential, $V(x) = -V_0$ when $|x| < a$, and $V(x) = 0$ when $|x| > a$. The ground state energy is found by solving the equation

$$\frac{\sqrt{\lambda - \xi^2}}{\xi} = \tan(\xi) \quad (1)$$

This equation is to be solved for ξ , using

$$\lambda = \frac{2(mc^2)V_0a^2}{(\hbar c)^2} \quad (2)$$

ξ is related to the ground state energy by

$$\xi = \sqrt{\lambda \left(\frac{V_0 + E}{V_0} \right)} \quad (3)$$

Note that for a bound state energy, $E < 0$. In fact, we know that $-V_0 < E < 0$.

Consider a square well potential with $V_0 = 1$ electron volt (eV) and $a = 1$ Å. Use your root finding program with Eq 1 to find the smallest positive root for ξ . Then use this to compute the ground state energy in electron volts. (Once you find ξ , you use Eq. 3 find E analytically!)

Use $\hbar c = 1970$ eV-Å and $mc^2 = 5.11 \times 10^5$ eV. (1 Å = 10^{-8} cm., but you don't need to convert to cm. – just use the units given in the problem.) Just to get you started, you should find $\lambda = 0.263$.

Reference: *Quantum Mechanics*, by Amit Goswami, section 4.2 (W.C. Brown, 1997), or almost any other elementary quantum mechanics text.

This assignment is due before 10:00 PM on Monday, September 21.