5 Finding the Root of a Function

The need to determine the roots of a function – i.e. where that function crosses the x-axis – is a recurring problem in practical and computational physics. For example, the density of matter in the interior of a star may be approximately represented by a type of function called a polytropic function. The edge or surface of the star occurs where the appropriate polytropic function goes to zero. There are many other examples. All of you know how to find analytically the roots of a parabola. For more complex functions, it may not be possible or easy to find the roots analytically, and so it is necessary to use numerical methods. The numerical method that is used depends upon whether or not it is possible to determine the derivative of the function. The function, for instance, may be given to us as a "black box" – it may be in tabular form, or given to us as a "C" function, or it may be known only through a recursion equation. In cases, the appropriate method would be to more and more closely bracket the root until we know it to the precision that we desire. Alternately, we may be able to differentiate the function. In that case, we can use the Newton-Raphson method (or similar methods) to zero in on the root.

5.1 The Method of Bracketing and Bisection

The first thing that we need to know in root-finding is an approximate location of the root we are interested in. How can we determine this? The easiest way is to use a plotting program, such as `gnuplot`, `EXCEL`, etc. to plot the function. Another way is to calculate the function at a number of points. Then, unless the function is pathological in some way, we can say that a root exists in the interval (a, b) if $f(a)$ and $f(b)$ have opposite signs. An exception to this is a function with a singularity in this interval. Suppose c is in the interval (a, b), and the function is given by

$$f(x) = \frac{1}{x - c}$$

Then, $f(x)$ has a singularity at c and not a root. There are other pathological functions such as $\sin(1/x)$ which has infinitely many roots in the vicinity of $x = 0$. So, the user must be aware of such problems, as such pathologies will give problems to even the most clever programs.

Let us assume that our function is fairly free of pathologies and that we know that there is a root in the interval (a, b) and that we want to find it more exactly.

The simplest way to proceed is using the method of *bisection*. This method is straightforward. If we know that a root is in (a, b) because $f(a)$ has a different sign from $f(b)$, then evaluate the function f at the midpoint of the interval, $(a + b)/2$ and examine its sign. Replace whichever limit (e.g. a or b) that has the same sign. Repeat the process until the interval is so small that we know the location of the root to the desired accuracy.

The "desired" accuracy must be within limits. As we have discussed before in class and lab, since computers use a fixed number of binary digits to represent floating point numbers, there is a limit to the accuracy to which computers can represent numbers. To be precise, the smallest floating point number which, when added to the floating-point number 1.0 produces a floating point number different from 1.0, is termed the *machine accuracy*, ϵ_m. Using floating point, most machines have $\epsilon_m \approx 1 \times 10^{-8}$. Using double precision, $\epsilon_m \approx 1 \times 10^{-16}$. It is always a good idea to back off from these numbers, and thus setting ϵ_m to 10^{-6} and 10^{-14} respectively are practical limits. Sometimes even those limits are too optimistic. Let us suppose that our $n - 1^{th}$ and n^{th} evaluations of the midpoint are x_{n-1} and x_n and that the root was initially in the interval (a, b). We would then be justified in stopping our search if $|x_n - x_{n-1}| < \epsilon_m |b - a|$.

`Exercise 5.1:` The polynomial $f(x) = 5x^2 + 9x - 80$ has a root in the interval (3,4). Find the root to a precision of 10^{-6} using the *bisection* method. Verify your root analytically.

Plotting functions in Gnuplot:
When finding roots, it is a good idea to plot the function to get an idea of the position of the roots. This can be done in gnuplot. To plot the function of Exercise 5.1, get into gnuplot, and enter the following:

```
gnuplot> f(x) = 5*x**2 + 9*x - 80
gnuplot> plot f(x)
gnuplot> g(x) = 0
gnuplot> replot g(x)
```

Note that exponentiation in gnuplot is accomplished with "**". Notice in the plot that $f(x)$ has a root at about -5 and another between 3 and 4.

`Exercise 5.2:` Bessel functions $J_0(x)$, $J_1(x)$, etc. often turn up in mathematical physics, especially in the context of optics and diffraction. The file `comphys.c` has a canned version of the Bessel function $J_0(x)$. The function declaration (contained in the file `comphys.h`) for this function is

```
float bessj0(float x)
```

Modify your program from `Exercise 5.1` to find the first two positive roots of $J_0(x)$.

`Exercise 5.3:` The viscosity of air (in micropascal seconds) is given to good accuracy by the following polynomial (valid between $T = 100$K and 600K):

$$\eta = 3.61111 \times 10^{-8} T^3 - 6.95238 \times 10^{-5} T^2 + 0.0805437T - 0.3$$

Use this polynomial to find the temperature T at which $\eta = 20.1$ μPa s.

`Exercise 5.4:` The structure of the stellar interior may be approximated by a polytropic gas sphere, where the surface is located at the position where a certain quantity y, related to the density, goes through zero. In the table below, the radial coordinate is given by x, and the density variable is given by y. Write a function that will interpolate between these points using 4-pt Lagrange interpolation, and then use the bracketing and bisection method to find the x value at the surface of the star.

```
 x       y
6.4   0.022718
6.5   0.017866
6.6   0.013162
6.7   0.008598
6.8   0.004168
6.9  -0.000134
7.0  -0.004312
7.1  -0.008373
7.2  -0.012321
7.3  -0.016161
```

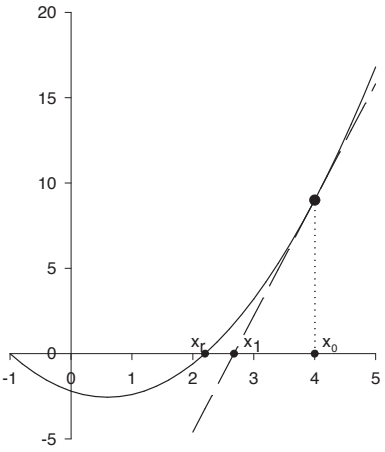
`Exercise 5.5:` An atomic state decays with two time constants τ_1 and τ_2 and can be described by the equation:

$$N = \frac{N_0}{2}(e^{-t/\tau_1} + e^{-t/\tau_2})$$

find by the method of bisection the time at which $N = N_0/2$, i.e. the half-life of the state. Let $\tau_1 = 5.697$ and $\tau_2 = 9.446$. The program should first display (with gnuplot) the function (use $N_0 = 100.0$) and then prompt the user for an initial bracket. Hints: Gnuplot does not recognize the variable `t`, so use `x`. In addition, you may wish to add the command `set xrange[0:15]` just before the `plot f(x)` command to give the plot a nice scaling. Use `exp(x)` for e^x in gnuplot.

5.2 The Newton-Raphson Method

If we can calculate the first derivative of our function, then we can use a speedier (although somewhat more dangerous) method called the Newton-Raphson method. Let us suppose that we have a function as illustrated below, which has a root at $x = x_r$, and that we have a rough estimate for that root of $x = x_0$. Let us evaluate the derivative of this function at x_0, $f'(x_0)$. This derivative will be the slope of the tangent line to the function at the point $(x_0, f(x_0))$, and you can see that where it crosses the x-axis, x_1, is a much better estimate to the root of the function than x_0. The process can be repeated to give an even better estimate, and so on. The equation of the first tangent line is (the student should verify):



$$y = f'(x_0)(x - x_0) + f(x_0)$$

if we set $y = 0$ to find the root of this line, we get

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

from which we can derive the Newton-Raphson formula for the n^{th} estimate to the root:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}$$

`Exercise 5.6:` Write a program to find the root for the polynomial of `Exercise 5.1` using the Newton-Raphson method. Begin with an estimate $x_0 = 4$. See what happens if you use a different starting point. Explore both positive and negative values for the starting point.

`Exercise 5.7:` Write a program to solve the problem of Exercise 5.3 using the Newton-Raphson method. Begin with an estimate of 200K.

`Exercise 5.8:` Write a program to solve the problem of Exercise 5.5 using the Newton-Raphson method.

An obvious danger to the Newton-Raphson method is that if there is a minimum or a maximum between the root and your starting point, the method will either not converge to the desired root at all, but to another one, or wander out to $\pm\infty$ at an ever accelerating pace. In addition, if you happen by bad luck to choose your beginning point at a maximum or minimum point, your program will never converge, but will blow up! Hence, as always, you should have a pretty good idea of the nature of your function before you attempt to find and "polish" roots.

`Exercise 5.9:` Use the fact that

$$\frac{dJ_0(x)}{dx} = -J_1(x)$$

to calculate the first two roots of $J_0(x)$. The code for $J_1(x)$ is in the file `comphys.c`, and the function definition is

`float  bessj1(float x)`

which is contained in `comphys.h`.

`Exercise 5.10` An Iterative Problem
In ballistics, it is common to solve the equations of motion of a particle shot at an angle, θ, above a horizontal surface with an initial velocity v_θ. The effects of air resistance play an important role that is often neglected in introductory or intermediate coursework. If the force of air resistance can be modeled as:

$$F_x = -km\dot{x} \tag{1}$$

$$F_y = -km\dot{y} \tag{2}$$

then the $x(t)$ and $y(t)$ solutions are:

$$x(t) = \frac{v_o \cos\theta}{k}\left(1 - e^{-kt}\right) \tag{3}$$

$$y(t) = -\frac{gt}{k} + \frac{kv_o \sin\theta + g}{k^2}\left(1 - e^{-kt}\right) \tag{4}$$

The time of flight of the projectile (the time elapsed before the projectile lands) is given as:

$$T = \frac{kv_0 \sin\theta + g}{gk}\left(1 - e^{-kT}\right) \tag{5}$$

Notice that T appears on both sides of equation (5). This is known as a "transcendental equation" which can be solved numerically using iteration. Note that this equation collapses to the simple expression for T when resistance is neglected ($k = 0$):

$$T(k = 0) = \frac{2v_0 \sin\theta}{g} \tag{6}$$

The range of motion (the distance traveled by the projectile before landing) is:

$$R = x(t = T) \tag{7}$$

In the exercises below, use $\theta = 60°$, $v_o = 600\text{m/s}$, and $|g| = 9.8\text{m/s}^2$.
a) Solve for T (equation 5) numerically using iteration. In the first iteration, use the non-resistive expression for T (equation 6). Iterate until the solution converges to within a user-defined tolerance (error) – e.g. ΔT over successive

iterations is less than the tolerance. Use a value of $k = 0.005$ for this part only.
b) Find the value of k that allows the projectile to travel 1500 meters ($R = 1500$m).
c) [**GRAD STUDENTS ONLY**] Plot y vs. x using gnuplot for k values of 0.0, 0.005, 0.01, 0.02, 0.04 and 0.08. Also, show analytically how to get equation (6) from equation (5) (*Hint: Use perturbation methods – i.e. expand* $1 - e^{-kT}$ *in a power series expansion, and keep only the first few terms*).

`Exercise 5.11` An Iterative Problem: A Simple Climate Model incorporating a one-layer atmosphere.

The climate model we will describe below is called a zero-dimensional energy balance model. It assumes a one-layer atmosphere, and a single uniform surface temperature and atmospheric temperature. Both layers – the surface and the atmosphere – must be in equilibrium with their energy inputs and outputs. The input is radiant energy from the sun, the output is infrared radiation. Without going into too much of the physics (if you want to know more, take my Environmental Physics class), the two equilibrium equations are, first, for the surface

$$-t_A(1 - a_s)\frac{S}{4} + c(T_S - T_A) + \sigma T_S^4(1 - a'_A) - \sigma T_A^4 = 0$$

and, for the single atmospheric layer

$$-(1 - a_A - t_A + a_S t_A)\frac{S}{4} - c(T_S - T_A) - \sigma T_S^4(1 - t'_A - a'_A) + 2\sigma T_A^4 = 0$$

where T_S and T_A are the temperatures (in kelvins) of the surface and atmosphere respectively, a_S is the albedo of the surface, and a_A the albedo of the atmosphere, both in the optical region of the spectrum. The transmission of the atmosphere in the optical is t_A. The albedo and transmission of the atmosphere in the infrared are given by a'_A and t'_A. The constant c describes convection in the atmosphere.

These are transcendental equations which means that they cannot be solved algebraically (try it – those who succeed will be given a free all-expenses-paid trip to Spitzbergen in the winter). They must be solved via an iterative process:
1) make an initial guess for T_S. A good guess is 273K.
2) Add the two equations to get a third in which the $c(T_S - T_A)$ term has

Typical Values of "constants" for the Energy Balance Model

short λ	long λ
$a_s = 0.11$	${t_A}' = 0.06$
$t_A = 0.53$	${a_A}' = 0.31$
$a_A = 0.30$	
$c = 2.5$ Wm^{-1}K^{-1}	

been canceled. Solve that equation for T_A^4, and determine a value for T_A using the current value for T_S.
3) Solve the first equation for T_S^4 (a term in T_S will be on the RHS). Using the value for T_A from step 2, and, using the current value for T_S for the T_S term on the RHS, find a new value for T_S^4, and thus a new value for T_S. Compare with the old value for T_S. If they are not in agreement better than 0.0001K, go back to step 2 and continue until convergence. (Hint – to assure stability, it may be necessary to use the mean of the old value of T_S and the value determined in step 3 as the new value for T_S.)

The table gives values for the constants in the two equilibrium equations. S is the solar constant, and may be taken to be 1365W/m^2.