Solving Equations in one Real Variable

We will study several methods for solving a fundamental problem in applied math

$$f(x) = 0.$$

Here we assume that f is a continuous real-valued function of a real variable, so we would like to find some real number, say x^*, such that $f(x^*) = 0$. We will call x^* a *zero* or *root* of f. If f is simple enough (and you are lucky enough), then it may be possible to find x^* analytically, that is, write a formula. But in the vast majority of cases the solution cannot be written as a formula, and iterative numerical techniques are needed.

Note that any equation in one variable, say $g(x) = h(x)$ can be changed into $f(x) = 0$. There are many ways to do this, and no best way, in general. For example, $e^x - 1 = \cos(x)$ could be turned into $f(x) = 1 + \cos(x) - e^x = 0$, or could also be turned into $f(x) = x - \cos^{-1}(e^x - 1) = 0$; in either case the solution to $f(x) = 0$ is also a solution to $e^x - 1 = \cos(x)$. Another common form (that we won't talk about in this class) is $F(x) = x$; solutions to this problem are called *fixed points* of F, and we can always turn root-finding problems into fixed-point problems, and vice-versa.

We will survey several root finding methods. There is no single best method; some are faster than others, some require a good guess, some require a smooth function, etc. You will have some choice in matching method to problem. Of the many properties we might use to describe a method, we will compare by generality and cost. A method which only works for polynomials is not as general as one which works for all continuous functions. Method A is more general than method B if A can solve everything B can solve, and more. Generality is a good quality, but it usually comes at a cost, so for example, the method that only works for polynomials will probably solve a polynomial problems faster than the more general method. Generality and cost usually work against each other in this way.

We usually think of cost in terms of *memory requirements* and *time requirements*. For the single variable problem at hand, memory requirements are usually trivial, so we will measure cost by computer time. But we don't know f, and we don't know what kind of computer our program will run on, so how do we measure computer time? We assume that evaluating f requires significantly more computer time than adding or multiplying several numbers, so the time required to solve $f(x) = 0$ is roughly the time required to evaluate f multiplied by the number of evaluations. Therefore our unit of cost will be 1 function evaluation. All of our methods will assume that we can evaluate f at any float near some initial guess, and we assume the user will provide a subroutine that does this. Our cost, then, is the number of calls to that subroutine.