10.001: Solution of Non-Linear Algebraic Equations

R. Sureshkumar
G. C. Rutledge

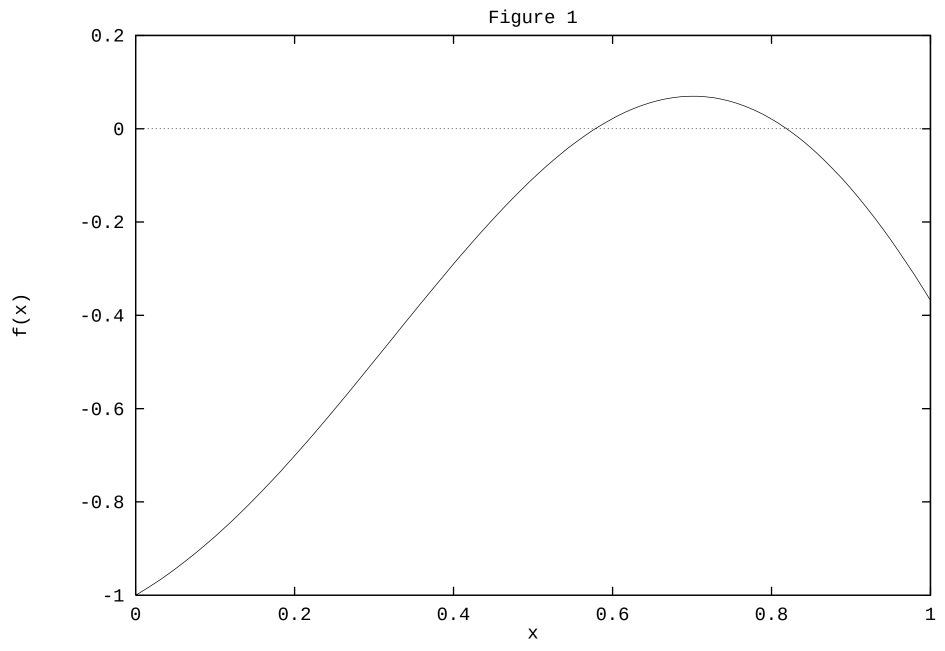
September 28, 1997

1 Introduction

In this class, we will discuss algorithms for finding the roots of non-linear algebraic equations. The problem we are dealing with here can be stated mathematically as follows: Find values of x such that the non-linear equation, $f(x) = 0$ is satisfied. When we say $f(x)$ is a *non-linear function* of x, it means that $f(x)$ can not be written as $\alpha x + \beta$, where α and β are constants. When we say that $f(x)$ is an *algebraic* equation, it means that $f(x)$ does not involve any differentials of the form $d^n y/dx^n$. A simple example is that of the familiar quadratic equation where $f(x) = ax^2 + bx + c = 0$. Similarly, $x - \sin(x) = 0$, $x^4 + 35x^3 + 20x^2 + x - 3 = 0$ etc. are all examples of non-linear equations. Evidently, the roots can not be obtained through analytical means except for a few simple cases. So our aim is to learn numerical methods which will evaluate the roots approximately.

There are many ways to locate an interval $a \le x \le b$ where the curve defined by $f(x) = 0$ intersects the x axis (the x coordinate of the point of intersection is a root of the equation $f(x) = 0$). One way is to simply plot the function in a given interval. Another way is to find two points on the x axis, say $x = a$ and $x = b$, such that the condition $f(a) * f(b) < 0$ is satisfied. i.e., the function changes sign as we move along the x axis from $x = a$ to $x = b$ (Let's assume that $a \le b$. Moreover, we restrict our attention to real roots.). This implies, for continous functions, that the graph (curve) of the function $y = f(x)$ intersects the x axis at least once between $x = a$ and $x = b$.

Example 1.: For instance, let $f(x) = x * sin(\pi x) - exp(-x)$. Since $f(0) < 0$ and $f(2/3) > 0$ and since $f(x)$ is continuous in this interval, there has to be a root between



0 and 2/3. Instead, we can simply plot the function using some mathematical software shown in Figure 1.

2 Bisection Methods:

We can pursuse the above idea a little further by narrowing the interval until the interval within which the root lies is small enough. For the function in Example 1, we can *bisect* the interval $[0, 2/3]$ to two subintervals, $[0, 1/3]$ and $[1/3, 2/3]$. Now, it is easily verified that $[0, 1/3]$ and $[1/3, 2/3]$. Now, it is easily verified that $f(x))$ does not change in the subinterval $[0, 1/3]$ and that it changes sign in the subinterval $[1/3, 2/3]$. Hence we choose the subinterval $[1/3, 2/3]$ and bisect it further. This will result in the choice of the new subinterval $[1/2, 2/3]$. We approximate the root at this stage as the arithmetic average of the end-point coordinates of this interval, this gives for the root $x_r = 0.58333...$ Now $f(0.5833) \approx 5.4 \times 10^{-3}$; if the root is desired only to this accuracy, we can stop here or if further accuracy is desired, we can proceed further with the bisection method. The above method can be generalized as a *bisection* algorithm as follows:

1. Given $f(x)$, choose the initial interval $[x_1, x_2]$ such that $x_1 < x_2$ and $f(x_1) * f(x_2) < 0$. Choose $\epsilon > 0$, the tolerance level. Choose N, maximum number of bisections. Define a counter, say ib, to keep track of the number of bisections performed.

2. *for* $ib \leq N$
```
Compute $x_3 = (x_1 + x_2)/2$.
*if* {$|f(x_3)| \leq \epsilon$ then print results and exit the program}
*if* {$f(x_1) * f(x_3) < 0$ then set $x_2 = x_3$}
*else* {$f(x_3) * f(x_2) < 0$ then set $x_1 = x_3$}.
```

3. If convergence was not obtained in N bisections, print current values for x_1, x_2 and $f(x_3)$ and inform the user that the tolerance criterion was not satisfied in N bisections and exit the program.

Notice that this algorithm locates only one root of the equation at a time. This is generally true of numerical methods for solving nonlinear equations. When an equation has multiple roots, it is the choice of the initial interval provided by the user which determines which root is located. The choice of an interval $[a, b]$ such that $f(a) * f(b) < 0$ only ensures that there is *at least* one real root between a and b, and therefore that the method can converge to a root. The contrary condition, $f(a) * f(b) > 0$ does *not* imply that there are *no* real roots in the interval $[a, b]$, however. Consider the example given above, with a starting interval of $[0, 1]$. What one can say, is that there is no guarantee of there being a root in the interval $[a, b]$ when $f(a) * f(b) > 0$, and the bisection algorithm will fail in this case. Thus the choice of starting interval is important to the success of the bisection method.

2.1 The Error Criterion

Instead of the criterion $|f(x)| \leq \epsilon$ used above, we can implement criteria which are insensitive to the value of the function. One such criterion is obtained by requiring that $|x_1 - x_2| \leq \delta$. This is an *absolute* tolerance criterion. There are instances when this is not a good criterion to use. For example: suppose $x_1 = 1.0000$ and $x_2 = 1.0001$; then $|x_1 - x_2| = 1.0E - 4$. Now, say the same equation was rescaled so that $x_1 = 1.0000E + 10$ and $x_2 = 1.0001E+10$; then $|x_1 - x_2| = 1.0E+6$, a number much larger than the previous estimate before rescaling. This is not a desirable property for an error criterion. Hence in order to make the criterion independent of the absolute magnitude of x, we use a *relative* *tolerance criterion*, given by

$$\frac{|x_1 - x_2|}{|(x_1 + x_2)/2|} \leq \delta_{rel}. \tag{1}$$

Note that for both the intervals $[1.0000, 1.0001]$ and $[1.0000E + 10, 1.0001E + 10]$, we get $\frac{|x_1 - x_2|}{|(x_1+x_2)/2|} \approx 10^{-4}$.

2.2 Convergence

Suppose our initial interval size, i.e., $x_2 - x_1$, is given by I_0. Then after the 1st bisection the interval size $I_1 = I_0/2$, after the second bisection, $I_2 = I_1/2 = I_0/4$ etc. Generalizing, after the nth bisection,

$$I_n = I_{n-1}/2 = I_0/(2^n). \tag{2}$$

So if the tolerance level is δ, then the number of bisections required to reduce the interval width to δ is given by

$$n_\delta \geq \log_2(I_0/\delta). \tag{3}$$

Or in other words, if we think of the current interval size as the error, the error in the nth step is proportional to the error in the $(n - 1)$th step. This is characteristic of a linearly convergent algorithm. We say that a method is *linearly convergent* if the error in the nth step is proportional to the error in the $(n - 1)$th step raised to the first power, *quadratically convergent* if the error in the nth step is proportional to the error in the $(n - 1)$th step raised to the second power, *quartically convergent* if raised to the fourth power, and so forth.

3 The Method of False Position

The poor convergence of the bisection method as well as its poor adaptability to higher dimensions (i.e., systems of two or more non-linear equations) motivate the use of better techniques. One such method is the *Method of False Position.* Here, we start with an initial interval $[x_1, x_2]$, and we assume that the function changes sign only *once* in this interval. Now we find an x_3 in this interval, which is given by the intersection of the x axis and the straight line passing through $(x_1, f(x_1))$ and $(x_2, f(x_2))$. It is easy to verify that x_3 is given by

$$x_3 = x_1 - \frac{(x_2 - x_1) * f(x_1)}{f(x_2) - f(x_1)}. \tag{4}$$

Now, we choose the new interval from the two choices $[x_1, x_3]$ or $[x_3, x_2]$ depending on in which interval the function changes sign.

The false position method differs from the bisection method only in the choice it makes for subdividing the interval at each iteration. It converges faster to the root because it is an algorithm which uses appropriate weighting of the intial end points x_1 and x_2 using the information about the function, or the data of the problem. In other words, finding x_3 is a *static* procedure in the case of the bisection method since for a given x_1 and x_2, it gives *identical* x_3, no matter what the function we wish to solve. On the other hand, the false position method uses the information about the function to arrive at x_3.

4 Newton-Raphson Technique

The Newton-Raphson method is one of the most widely used methods for root finding. It can be easily generalized to the problem of finding solutions of a system of non-linear equations, which is referred to as Newton's technique. Moreover, it can be shown that the technique is quadratically convergent as we approach the root.

Unlike the bisection and false position methods, the Newton-Raphson (N-R) technique requires only one inital value x_0, which we will refer to as the *initial guess* for the root. To see how the N-R method works, we can rewrite the function $f(x)$ using a Taylor series expansion in $(x - x_0)$:

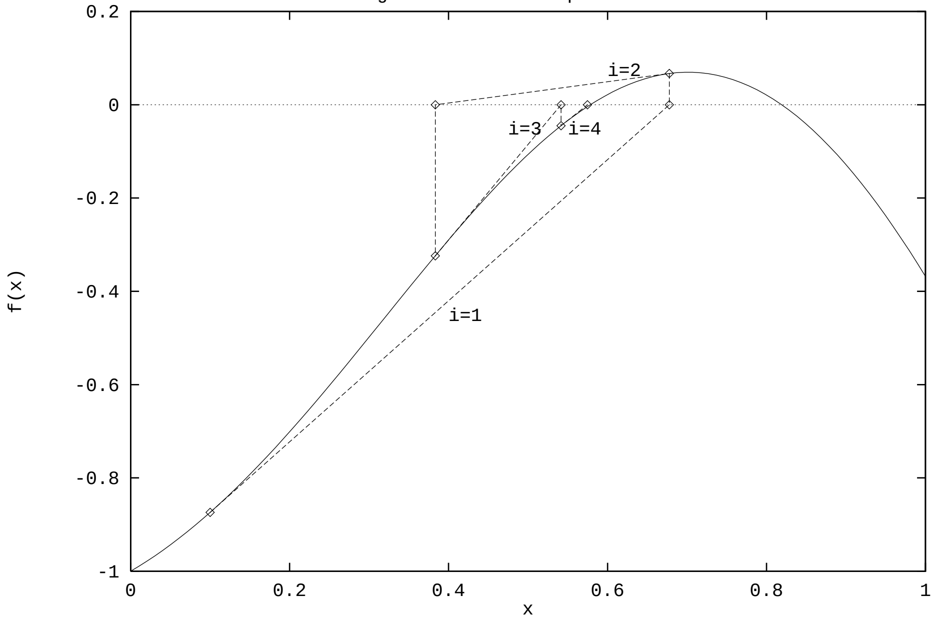
$$f(x) = f(x_0) + f'(x_0)(x - x_0) + 1/2 f''(x_0)(x - x_0)^2 + ... = 0 \tag{5}$$

where $f'(x)$ denotes the first derivative of $f(x)$ with respect to x, $f''(x)$ is the second derivative, and so forth. Now, suppose the initial guess is pretty close to the real root. Then $(x - x_0)$ is small, and only the first few terms in the series are important to get an accurate estimate of the true root, given x_0. By truncating the series at the second term (linear in x), we obtain the N-R iteration formula for getting a better estimate of the true root:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{6}$$

Thus the N-R method finds the tangent to the function $f(x)$ at $x = x_0$ and extrapolates it to intersect the x axis to get x_1. This point of intersection is taken as the new approximation to the root and the procedure is repeated until convergence is obtained whenever possible. Mathematically, given the value of $x = x_i$ at the end of the ith

Figure 2: Newton-Raphson Method



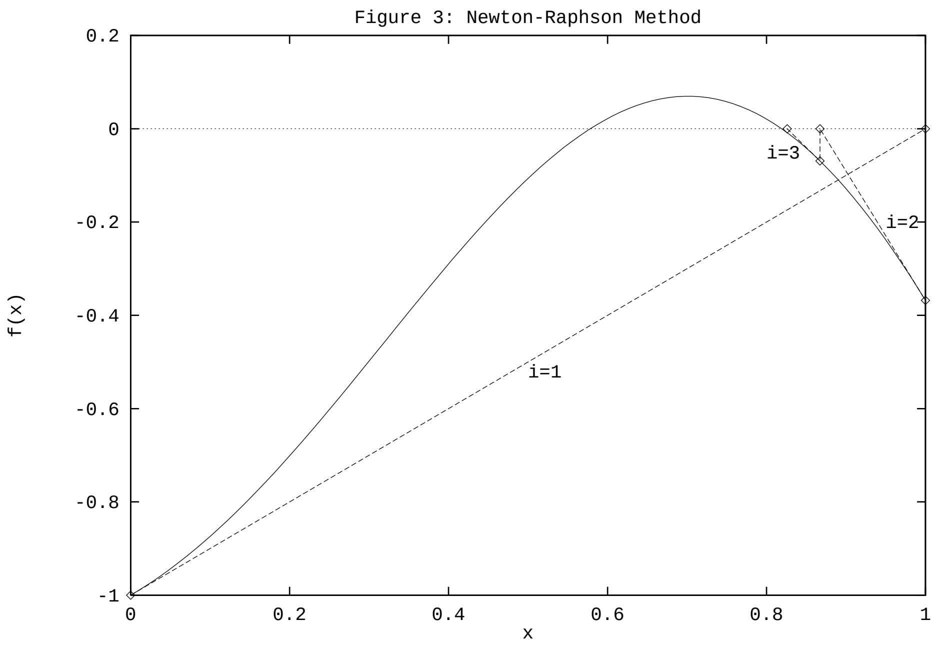
iteration, we obtain x_{i+1} as

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \tag{7}$$

We assume that the derivative does not vanish for any of the x_k, $k = 0, 1, \cdots, i+1$. The result obtained from this method with $x_0 = 0.1$ for the equation of Example 1, $x * sin(\pi x) - \exp(-x) = 0$, is graphically shown in Figure 2. Here also, when multiple roots are present, the root evenutally identified by the algorithm depends on the starting conditions supplied by the user. For example, if we had started with $x_0 = 0.0$, the N-R method will converge to the larger root, as shown in Figure 3. So, before using any of the methods, we should try to get as good a feel as we can afford to get for the function. An approximate idea of the roots can be developed from the physics the equation represents, preliminary mathematical analysis and using plotting routines.

4.1 The Initial Guess

One good way to estimate an initial guess for starting the N-R method is to *linearize* the equation being solved. For example, the equation $f(x) = x * sin(\pi x) - exp(-x)$ may be



rewritten as a sum of two series:

$$f(x) = x * \sum (-1)^n \frac{(\pi x)^{2n+1}}{(2n+1)!} - \sum \frac{(-x)^n}{n!} \tag{8}$$

Linearization implies that we approximate this equation with one in which we neglect all terms containing x raised to a power $m > 1$ (x^2, x^3, etc.) (*c.f.* Introduction). In this example, linearization reduces the equation to:

$$f(x) = 1 - x \tag{9}$$

The root of the linearized equation $f(x) = 1 - x = 0$ is 1, which provides a reasonable initial guess for the actual roots on the nonlinear equation, in the absence of a plot such as Figure 1.