Root finding methods in MATLAB

Wednesday, January 28, 2015
11:13 AM

Finding the roots of an equation is often of physical significance, and so finding an efficient and fast way to solve for roots numerically is very important in the field of numerical methods.

A root of a single input variable function is defined as an x-value such that f(x) = 0. This is represented graphically as a plot of the function crossing the x-axis.

We have covered 4 methods of root solving in class and they will be summarized here. They are subdivided into 2 groups.

Bracketed methods - those were two initial guesses are required, and the algorithm searches for a root between the given values

Open methods - those were one initial guess is given and the algorithm searches for a solution near the given value

Bracketed Methods

1) **Bisection** - This method searches for a root between the brackets by repeatedly dividing the space in half and checking which of the two spaces the root lies in.

   The program starts with a lower bracket *a* and an upper bracket *b*, such that the root lies in the domain
   $a < x < b$. To ensure this is the case, $f(a)$ and $f(b)$ must evaluate as opposite signs. That is, one must be positive and the other must be negative. The program then takes the following steps.

   i) Calculate the midpoint $m = \frac{a+b}{2}$
   ii) Evaluate $f(m)$
   iii) Replace either the upper or lower bracket with m, ensuring the root stays between the two brackets
   That is,

   if
   $sign(f(a)) == sign(f(m))$
   $a = m$
   else
   $b = m$
   end

   The above conditions maintains the requirement that at one bracket the function be positive, and the other negative, so that a root must lie somewhere in between.

   iv) This process repeats itself until a certain tolerance is met.
   Either

   $f(m) < tol$

   Or

$$m - a < tol$$

I greatly prefer the first method.

2) **False Position** - False position is VERY similar to bisection, except that the domain is not divided in to 2 equal parts each iteration. It is instead divided in to two pieces in a slightly more clever way. The function is evaluated at both endpoints and a straight line is drawn between them. The dividing point, *c*, is the root of this line.

   The steps are then:

   i) Calculate the dividing point $c = b - (b - a)\frac{f(b)}{f(b)-f(a)}$
   ii) Evaluate $f(c)$
   iii) Replace either the upper or lower bracket with c, ensuring the root stays between the two brackets
   That is,

   if
   $sign(f(a)) == sign(f(c))$
   $a = c$
   else
   $b = c$
   end

   The above conditions maintains the requirement that at one bracket the function be positive, and the other negative, so that a root must lie somewhere in between.

   iv) This process repeats itself until a certain tolerance is met.
   Either

   $$f(c) < tol$$

   Or

   $$c - a < tol$$

   Or

   $$c_k - c_{k-1} < tol$$

   I greatly prefer the first method.

Open Methods

1) **Newton-Raphson** - This method requires knowledge of both the function and its derivative. Given an initial guess for the root, x_0 , the tangent line at x_0 is calculated using the derivative. The next guess for the root of the function f is taken to be the root of the tangent line. The algorith is then simply

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

This process repeats itself until a certain tolerance is met, either

$f(x_k) < tol$

Or

$|x_k - x_{k-1}| < tol$

I prefer the first method.

2) **Secant Method** - In this algorithm, the exact slope is not known. Instead, we use two initial guesses and build a secant line which passes through the two points. The next guess in the algorithm is based on the secant line passing through the two most recent points generated.

   The method starts with guesses, x_0 and x_1, and subsequent points are given by

   $$x_{k+1} = x_k - (x_k - x_{k-1})\frac{f(x_k)}{f(x_k) - f(x_{k-1})}$$

3) **Modified secant method** - In this algorithm, a secant line is created from just one point, and another just slightly less than that. The method starts with an initial guess x_0 and subsequent guesses are generated by

   $$x_k = x_k - \frac{\delta x_k f(x_k)}{f(x_k) - f(x_k - \delta x_k)}$$