Math 4610 Lecture Notes

Root Finding Problems for Real Values Function of One Variable *

Joe Koebbe

September 10, 2019

*These notes are part of an Open Resource Educational project sponsored by Utah State University

Root Finding Problem: Definition of the Problem

Many problems can be recast in the form of finding places where a function is zero. In a standard first semester calculus course find extreme values of a function of one variable amounts to determining locations where the derivative of the function is zero. That is, a necessary condition for the existence of a local minimum or local maximum at a point x^8 is that the derivative is zero at x^* or

$$f'(x_0) = 0.$$

More often than not, we will need to deal with roots that are not exact in machine precision. For example, finding the roots of

$$sin(x) = 0$$

is easy from an analytic point of view, the zeros are $x_n = n\ \pi$ where n can be any integer. If n is not equal to zero, the root is an irrational number and cannot be represented exactly. So, we will need to settle for an approximation.

The general root finding problem can be written as follows: For a given real-valued function, f, of a single real variable find a real number, x^*, such that

$$f(x^*) = 0$$

There are all kinds of issues that arise in solving these types of problems. For example, the function may have multiple roots. In searching for a specific root, we may find other roots that are not of interest. To deal with all of the issues in this problem, we will develop a number of algorithms that can be used in a variety of root finding problems.

Root Finding Problems: Using Fixed Point Iteration

As a first attempt at determining the location of a root for a function, we might consider a modification of the root finding problem as follows. Given a function, f, we can rewrite the equation

$$f(x^*) = 0$$

as

$$x = x - f(x^*) = g(x^*)$$

The resulting equation is called a fixed point equation and the equation suggests an iteration of the form

$$x_1 = g(x_0), x_2 = f(x_1), \ldots$$

where x_0 must be supplied to start the iteration. This iteration formula will produce a sequence of real numbers. The hope is that the sequence will converge to a solution of the fixed point equation and also a solution of the root finding problem.

Root Finding Problems: Coding Fixed Point Iteration

One can easily write a routine or computer code that implements fixed point iteration. The following code provides a template of how a reusable routine might be written:

```
//
// Author: Joe Koebbe
//
// Routine Name:         fproot
// Programming Language: Java
// Last Modified:        09/10/19
//
```

```
// Description/Purpose: The routine will generate a sequence of numbers
// using fixed point iteration.
//
// Input:
//
// FunctionObject f - the function defined in the root finding problem
// double x0 - the initial guess at the location of a fixed point
// double tol - the error tolerance allowed in the approximation of the
//              root finding problem
// int maxit - the maximum number of iterations allowed in the fixed point
//             iteration.
//
// Output:
//
// double x1 - the last number in the finite sequence that is an
//             approximation in the root finding problem
//
public double fproot(FunctionObject f, double x0, double tol, int maxit) {
  //
  // initialize the error in the routine so that the iteration loop will be
  // executed at least one time
  // --------------------------
  //
  double error = 10.0 * tol;
  //
  // initialize a counter for the number of iterations
  // -------------------------------------------------
  //
  int iter = 0;
  //
  // loop over the fixed point iterations as long as the error is larger
  // than the tolerance and the number of iterations is less than the
  // maximum number allowed
  // ----------------------
  //
  while(error > tol && iter < maxit) {
    //
    // update the number of iterations performed
    // -----------------------------------------
    //
    iter++;
    //
    // compute the next approximation
    // ------------------------------
    //
    double x1 = x0 - f(x0);
    //
    // compute the error using the difference between the iterates in the
    // loop
    // ----
    //
    error = Math.abs(x1 - x0);
    //
    // reset the input value to be the new approximation
```

```
    // ------------------------------------------------
    //
    x0 = x1;
    //
  }
  //
  // return the last value computed
  // ------------------------------
  //
  return x1;
  //
}
```

There are a couple of features in the code that need to be explained.

- To make this work in the Java programming language, the method would need to be embedded in a class. That is, the code is not a standalone code.
- The first argument is an Java Object that needs to be created. The object is used to provide the function evaluation for any real input.
- The second argument is the initial guess at the solution of the problem.
- Since we know we are going to end up with at best an approximation of a root, the third argument in the function is an error tolerance that is acceptable to the calling routine.
- The final argument passed in limits the number of iterations allowed in the method. Note that if you are not careful, an infinite loop might be created due to the approximations used everywhere.

If we apply the code to any problem, we are assuming that the solution will pop out the end. There is no guarantee that this is the case. It is important to establish conditions that will guarantee the code will produce an approximate solution of the fixed point problem and thus provide a root for the original function, f.

Root Finding Problems: Analysis of Functional Iteration Using Taylor Series Expansion

---

The general iteration formula, given x_0, is the following.

$$x_{k+1} = g(x_k)$$

for $k = 0, 1, 2, \ldots$. We also know that for the fixed point problem, the solution satisfies the equation

$$x^* = g(x^*)$$

Subtracting the two equations gives

$$x_{k+1} - x^* = g(x_k) - g(x^*)$$

The Taylor expansion of $g(x_k)$ about the solution x^* is given by

$$g(x_k) = g(x*) + g'(x^*)(x_k - x^*) + \frac{1}{2}g''(x^*)(x_k - x^*)^2 + \ldots$$

Substituting the expansion into the equation above and truncating the series gives

$$x_{k+1} - x^* \approx g(x*) + g'(x^*)(x_k - x^*) - g(x^*) = g'(x^*)(x_k - x^*)$$

Taking absolute values the last equation can be written as

$$|x_{k+1} - x^*| \leq |g'(x^*)||x_k - x^*|$$

One can read the previous expression as the difference (or error) in x_{k+1} is less than the magnitude of the derivative of the fixed point iteration function, g, times the difference (or error) in the previous approximation, x_k. Using

$$e_k = |x_k - x^*|$$

allows use to relate the error at successive steps as

$$e_{k+1} \leq |g'(x^*)| \cdot |e_{k+1}|$$

To get convergence to the fixed point (or root) we would like the error to be reduced at each step. This requires the condition

$$|g'(x^*)| < 1$$

For the general fixed point problem, this condition is required for convergence to the fixed point, x^*, or solution of the root finding problem. Note that this is a significant drawback of fixed point iteration as a means of solving root finding problems.

Root Finding Problems: An Example Using Functional Iteration

Suppose that we are interested in computing the roots of

$$f(x) = e^x - \pi$$

Analytically we can compute the solution by solving for x in the equation

$$e^x - \pi = 0$$

The value is $x = ln(\pi) \approx 1.144729886$. This is a very simple problem. However, it is always a good idea to test general methods on simple problems while developing algorithms and coding these up for use on real problems. Let's apply functional iteration to this root finding problem. First, we will need to create an associated function that defines a fixed point problem. One possibility is to choose

$$g(x) = x - f(x) = x - (e^x - \pi) = x - e^x + \pi$$