**CSCI 2150L**

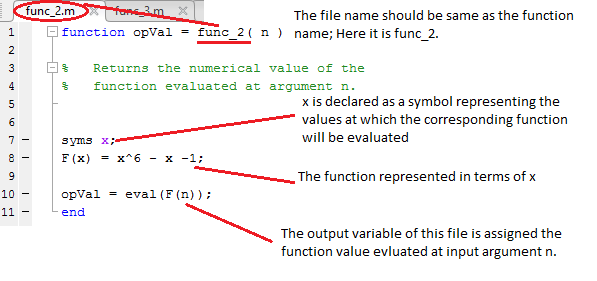
**Session 11 Topics:**

**10/15/2014**

**Fall 2014**

1. In this session we discussed bisection method, a method to find roots of a give function. Before discussing the method in detail we created the following three functions in three different files.

We took help of command **syms x** to create a symbol x in each of our function files. Subsequently, we evaluated the function for the argument parameter passed to the function using **eval(F(n))** command and returned the value from the function. Here the argument of command eval is the function *F* expressed in terms of function argument n. The following figure shows one such function file written for function a) above.

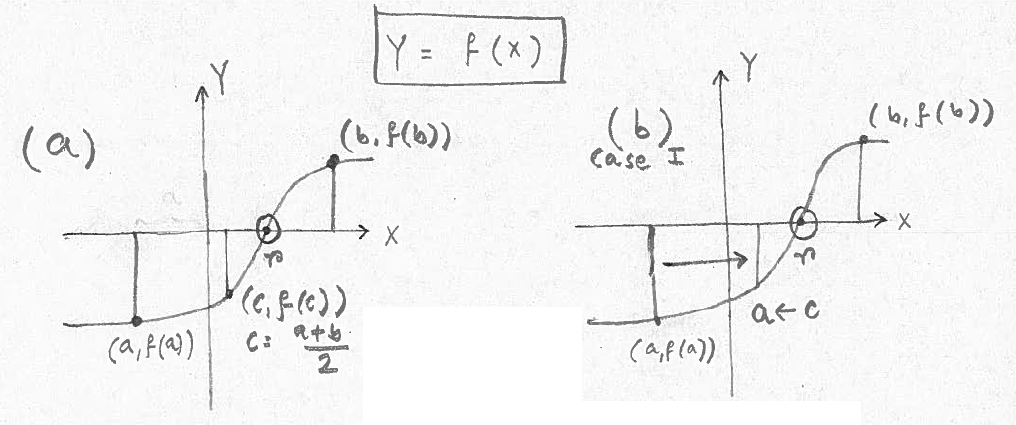


Similarly we created two other files corresponding to other two functions.

1. Next we discussed the bisection method algorithm described as follows.

Bisection Algorithm ():

1. Find two numbers *a* and *b* at which function *f* has different signs such that.
2. Compute root based on these values. New root ; Compute tolerance term
3. If then accept *c* as new root and exit algorithm. Otherwise, go to step d).
4. If then set *c* as new *a* (i.e.) or set *c* as new *b* (i.e. ).
5. The algorithm above is described with the help of the following Fig. 2. In the figure a function *f* has been considered such that. The first step of the algorithm above is depicted in Fig.2 (a) where the initial points and are given such that the points and lies on either side of the actual root (encircled in the figure) of the function. [Note that at root, ]. Now based on these points given a new root is computed following step 2 of the algorithm above. Depending on the values of and the assumed root may satisfy three situations. In first case it may be found fairly close to the actual root. If this condition is satisfied then the tolerance condition in step 3 would be satisfied and the program will stop after deciding that is the actual root (or fairly close to the actual root) of the given function. However, can also be found to be not close to the actual root. It might go more towards or from the actual root . These two cases have been demonstrated in Fig. 2(b) and Fig.2(c).



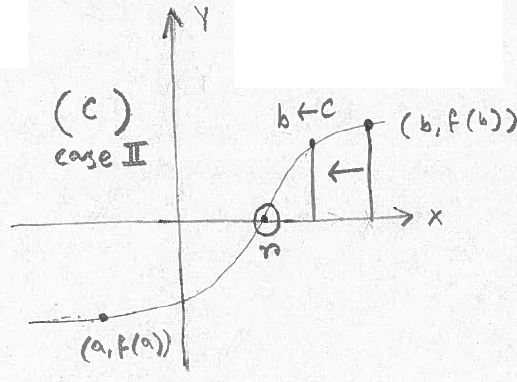


Fig. 2(a) Step 1 of algorithm; (b) case I: when assumed root is towards *a* from actual root *r*; (c) case II: when assumed root is towards *b* from actual root *r*.

In Fig. 2(b) is found to be located towards from actual root . For this situation the condition is true as is positive and is negative (, ). Subsequently, the variable will be updated and will hold the value of . This is because in the next iteration the algorithm will be supplied another set of fresh numbers and such that the necessary condition is satisfied to initiate bisection method. Fig. 2(c) depicts the complementary situation where is found to be towards more from root *r* (Think yourself about this situation to understand which variable would be updated in this case and why; see algorithm). These two situations has been taken care of in step 4 of the algorithm. After step 4 the algorithm re-iterates on the new values of and to find a more accurate root.

Remember the code for bisection had two extra parameters. One specified the name of the function file on which the algorithm would be evaluated. The other one was a number that restricts the maximum number of iterations bisection method can go through. This is a precautionary measure as in some situations numerical algorithms would run for long time before converging. We’ll talk about this more in subsequent sessions.