

The Bisection and False Position Methods for Finding Roots

Save the following two files to your N: drive in your m439 folder: (Right click and choose save link as)

[Bisect.m](#) (Bisection Method) [Regula.m](#) (False Position Method – Also called Regula Falsa Method).

Open MATLAB and make your N:\m439 folder you current directory

Choosing Appropriate Initial Intervals [a,b] for Root Finding Algorithm

To make sure our algorithm will converge to a root, we need to make sure it contains one (and only one) root. If there are more than one roots for $f(x) = 0$, then we need to run the algorithm several times, once for each root with an appropriate initial interval choice. A good way to determine what the appropriate initial interval choice should be is to graph the function. The MATLAB command for the exercise 1a) of Problem Set 2 would be:

```
>> fplot('x-cos(x)',[-2,2])
```

Decide on an appropriate subinterval. Then check the values of the function at the endpoints subinterval to make sure they have the opposite signs, using

```
>> ??-cos(??)
```

The Bisection Algorithm for Finding Roots

1. a) Create a file called **f.m** (save to your N:\m439 folder) that contains the following:

```
function y = f(x)
```

```
% F: Defines the function y =
```

```
% x is a real number
```

```
y = x.*sin(x) - 1;
```

b) This function will have an infinite number of zeros. Plot this function for domain -3 to 3:

```
>> fplot(@f,[-3 3])
```

2. Open the file **Bisect.m** with the MATLAB editor (File-Open) which has the code for the Bisection Method for finding zeros of a function. Note the header for the function and the documentation indicating how it should be called. Execute the function for the interval $[0,2]$ on the function $y = x \cdot \sin(x) - 1$ stored in **f.m** with an error tolerance of $1e-6$ (that is, 10^{-6}).

```
>> [?, ?, ?] = Bisect (@f, ?, ?, ?)
```

What is the value of the root (zero) returned by this call? _____ What is the value of the value of the function at this zero? _____

3. In the algorithm the following line determines the maximum number of iterations required by the Bisection Method to find the zero within the required tolerance parameter delta. Why does the line of code below give appropriate value for max1?

```
max1=1+round((log(b-a)-log(delta))/log(2));
```

Answer _____

4. To determine the value of max1 for our interval [0,2] enter the following command at the MATLAB prompt:

```
>> 1+round((log(2)-log(1e-6))/log(2))
```

What value is returned ? _____

5. Modify the code for Bisect.m to make it more robust. It is a bad idea to check for equality of floating point numbers in a computer program (because of round off error). Yet this program does this in the line:

```
if yc == 0  
    a=c;
```

Replace this (why?) with

```
if abs(yc) < eps  
    a=c;
```

6. a) A second improvement to the code can be made. Note that the "delta" tolerance is how close we want the width of the last subinterval bracketing our proposed root c when we quit. This does not mean that the function f(c) (stored in yc) will actually be within "delta" of 0. Why -
- what kind of function would potential have a value f(c) not very close to zero even though c is close to the location of the true root?

Answer

b) To fix this we want to update the code. Add another parameter **epsilon** to the function heading to provide for the tolerance for the value of the function $yc = f(c)$ at the proposed root c. Then modify the last line in the body of the for loop:

```
if b-a < delta , break,end
```

to be instead:

```
if (b-a < delta) && (abs(yc) < epsilon), break,end
```

Save your code as **Bisect2.m** and re-execute it, adding a value now for the extra parameter epsilon:

```
[c,err,yc]=Bisect2('f',0,2,1e-6,1e-6)
```

The False Position Algorithm for Finding Roots

7. Run the false position method algorithm on the same interval for the above function.

```
[c,err,yc] = Regula('f',0,2,1e-6,1e-6,100)
```

8. Note they return similar values for the root. But which is faster? -- How many iterations are actually performed by each? One way of checking this is to add a line to each program to print out the iteration number and current values for c (root) and value yc = f(c). In **both** of the m-files, add the following line:

```
c,yc,k,pause;
```

at the bottom of the **for** loop (just *before* the code **if (b-a < delta ...:)**)

Save each m-file and then re-execute each. The pause commands waits for you to hit any key before continuing.

How many iterations the bisection algorithm take? _____ How many iterations did false position take? _____ Which is faster? _____