

Matlab Code for Numerical Root Finding Methods

Michel Tillmann

Concordia University
ENGR 391: Numerical Methods
September 16, 2020

NOTES AND INSTRUCTIONS:

Since the eConcordia platform has issues uploading files with the “.m” extension, the code for the four root finding methods discussed in Tutorial Section F on September 14th is transcribed within this MS Word document. Keep in mind that these code snippets are to serve as preliminary guidance for you, as students, to create your own functions capable of extracting the information requested in your various course assignments and exams. Using the code as provided will be insufficient to tackle all the challenges in this course. That being said, please feel free to play around with it, try new things, and feel free to ask any questions you may be struggling with; I and the other TA’s are glad to be of assistance.

In order to execute the code as was done during the tutorial, you will need to copy the function definition into a new script (.m file). In order to execute the function, the file must be named exactly the same (case sensitive) as the function name in the first line of the code block. For example, the file containing the following function:

```
function [c, a, b, i] = Bisection(a, b, i)
```

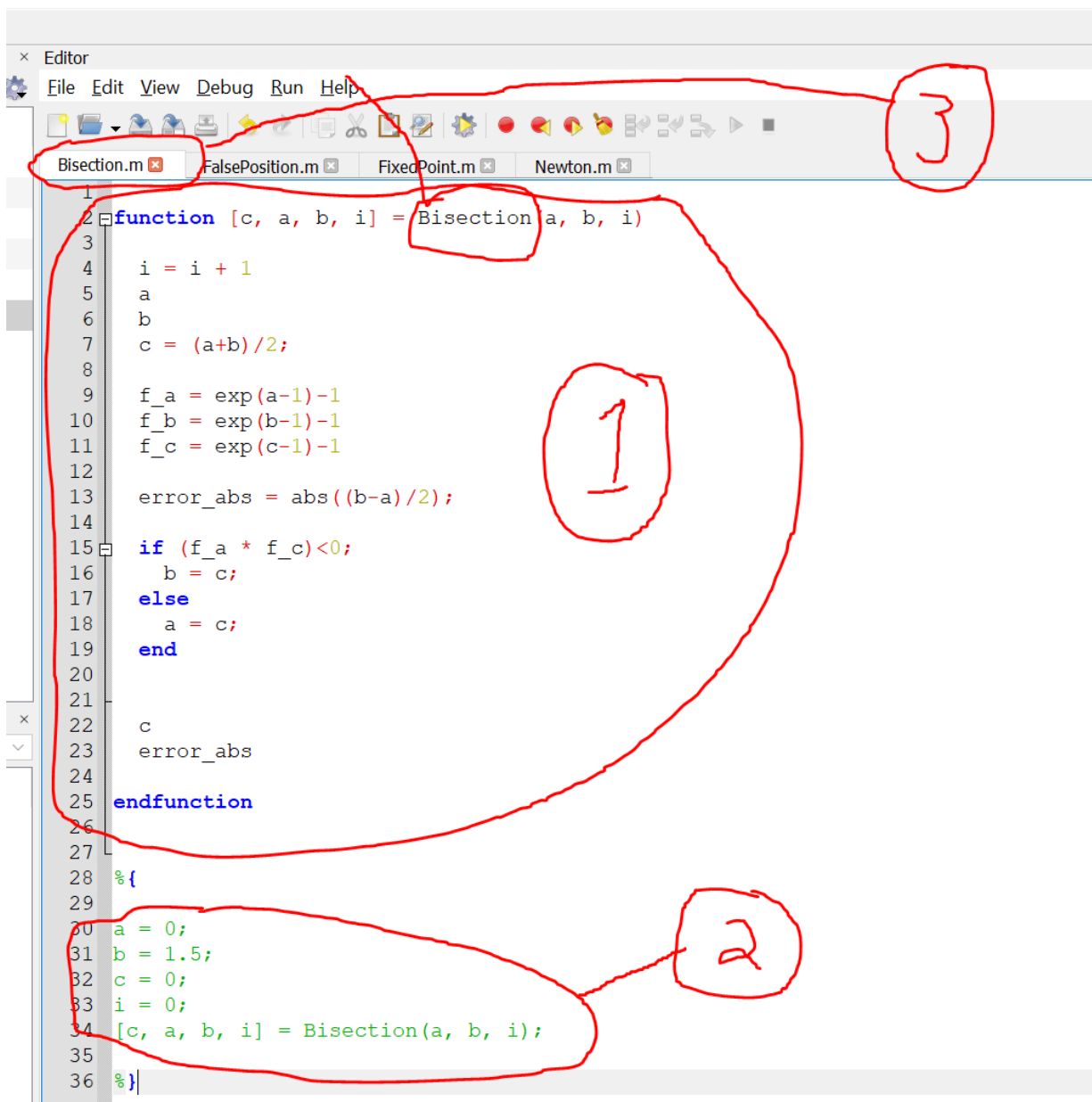
Must be named “Bisection.m”. For the sake of simplicity, it is advised that you start by only defining one function per .m file.

Once the script is created and saved, copy the commands listed below their respective functions in this document into the command window of Octave or Matlab. Make sure you are in the directory where the function file is saved. Re-run the function call as-is to iterate.

Example:

In the picture below:

1. Function definition
2. Command window commands
3. File name and function name must be the same



```
1
2 function [c, a, b, i] = Bisection(a, b, i)
3
4     i = i + 1;
5     a
6     b
7     c = (a+b)/2;
8
9     f_a = exp(a-1)-1;
10    f_b = exp(b-1)-1;
11    f_c = exp(c-1)-1;
12
13    error_abs = abs((b-a)/2);
14
15    if (f_a * f_c) < 0;
16        b = c;
17    else
18        a = c;
19    end
20
21
22    c
23    error_abs
24
25 endfunction
26
27
28 %{
29
30 a = 0;
31 b = 1.5;
32 c = 0;
33 i = 0;
34 [c, a, b, i] = Bisection(a, b, i);
35
36 %|
```

BISECTION METHOD

Function Code:

```
function [c, a, b, i] = Bisection(a, b, i)

    i = i + 1
    a
    b
    c = (a+b)/2;

    f_a = exp(a-1)-1
    f_b = exp(b-1)-1
    f_c = exp(c-1)-1

    error_abs = abs((b-a)/2);

    if (f_a * f_c)<0;
        b = c;
    else
        a = c;
    end

    c

    error_abs

endfunction
```

Command Window Commands:

```
a = 0;
b = 1.5;
c = 0;
i = 0;
[c, a, b, i] = Bisection(a, b, i);
```

FALSE POSITION METHOD

Function Code:

```
function [c, a, b, i] = FalsePosition(a, b, i)

    i = i + 1
    a
    b

    f_a = exp(a-1)-1;
    f_b = exp(b-1)-1;

    c = ((a*f_b)-(b*f_a))/(f_b- f_a)
    f_c = exp(c-1)-1;

    error_rel = ((b-a)/2)/c

    if (f_a * f_c)<0;
        b = c;
    else
        a = c;
    end

endfunction
```

Command Window Commands:

```
a = 0;
b = 1.5;
c = 0;
i = 0;
[c, a, b, i] = FalsePosition(a, b, i);
```

FIXED POINT METHOD

Function Code:

```
function [x_i, i] = FixedPoint(x_i, i)

    i = i + 1

    x = x_i;

    x_i = 1 + (1/x_i)

    f = x_i^2 - x_i - 1

    error_rel = abs((x_i - x) / x_i)

endfunction
```

Command Window Commands:

```
i = 0
x_i = 2
[x_i, i] = FixedPoint(x_i,i);
```

NEWTON'S METHOD

Function Code:

```
function [x_i, i] = Newton(x_i, i)

    i = i + 1

    x = x_i

    f_x = exp(x-1)-1
    df_x = exp(x-1)

    x_i = x - (f_x/df_x)

    error_rel = abs((x_i - x)/x_i)

endfunction
```

Command Window Commands:

```
i = 0
x_i = 0
[x_i, i] = Newton(x_i,i)
```