EE254-2F Exam #4 Spring 2020 Name: Connor McGarty

Instructions:

Open book, open notes and open reference with the following exceptions:

1) No cell phones.
2) No email or messaging applications.
3) No direct copying from online resources. If online resources are used, they must be identified and documented. Your contribution must be easily identifiable. If in doubt, ask.

I am looking for certain keyword/concepts that demonstrate your understanding of Matlab, programming and numerical methods in general. You have 75 minutes to complete the exam and may only leave the room when you turn in the exam. Write clearly!

Question{01}: Briefly explain in your own words, the following terms and concepts (10):

a. What is your understanding of the purpose/use of numerical root finding techniques?
The purpose of numerical root finding is to approximate roots of equations that can not be solved directly (for instance, a quadratic equation has a direct solution to calculate the two roots). Numerical root finding techniques are also useful for when the function that governs some set of data is unknown (such as when reading and processing data from some sensor).

b. What is your understanding of the difference between bracketed and open root finding techniques?
Bracketed methods (such as bisection and false position) require knowledge/guesses of points which bracket or surround the location of a root. The approaches then systematically reduce the size of the bracket until an error threshold is met.
Open methods involve iterating with trial-and-error, yet do not require guesses that bracket a root. These methods are often more computationally efficient, but do not always produce a correct solution.

c. Describe the Incremental Search technique including an explanation of its advantages, disadvantages and how you could accidentally miss a root.
Incremental search takes advantage of the fact that on either side of a root, the value of the function changes sign. To do this, a function is broken into some number of segments, and each adjacent pair of function values' signs are compared – if they differ, there is at least one root between the pair of values. Incremental search is advantageous because it is trivial to implement and given an appropriate number of segments guarantees correctness. However, for functions that require a sufficiently small segment size (x_delta) it can be time consuming since every pair of values along the given domain must be compared. But if a sufficiently large enough x_delta is used, brackets that contain roots can either be 1) skipped over entirely, or 2) the procedure may return only one bracket that actually contains more than one root if the roots are closely spaced in comparison to the segment size.

d. How would you use the Incremental Search technique in conjunction with another technique to find a root?
Incremental search is useful for determining brackets that contain roots, which can then be followed up with bracketed root finding methods such as bisection or false position. After finding brackets with incremental search, the results can be passed on to one of these other methods to find a sufficient approximation for the exact location of the root.

e. In addition to the algorithmic techniques we built in our classes/project, we also saw that Matlab provides several built in tools. Describe the steps in using one of them to identify the x_root_actual that we use to calculate our e_t.
One of the functions provided by MATLAB that can be used to find roots of functions is `vpasolve`. `vpasolve` numerically solves a symbolic equation for any specified variable – it can be used to solve for a root. It can be provided with an initial guess or a search range if there are multiple solutions. Example:

```
syms x; y = 1 – x; % define function
% use vpasolve to find the value of x where y is 0 (i.e., the root)
% this example provides a search range between x = 0 and x = 10
x_root_actual = vpasolve(y == 0, x, [0 10]);

% x_root_actual = 1.0
```

Question{02}: Given the following factored polynomial, using incremental search, complete the table:

$$y = (x - 1)(x - \frac{3}{2})(x - \frac{7}{4})(x - \frac{15}{8})(x - \frac{31}{16})$$

x_{min}	x_{max}	x_{delta}	Roots found
0	10	1	x = 1
0	10	.5	x = 1, x = 1.5
0	10	.25	x = 1, x = 1.5, x = 1.75
0	10	.125	x = 1, x = 1.5, x = 1.75, x = 1.875
0	10	.0625	X = 1, x = 1.5, x = 1.75, x = 1.875, x = 1.9375

Explain any unexpected behavior.
When the xdelta is larger, sign changes can be “skipped over” resulting in missing roots. As the xdelta decreases and resolution of the graph/data increases, you are more likely to detect the changes in sign. If the behavior of the function is such that the there is relatively large (compared to xdelta) distances between roots, then this is an unlikely occurrence. But if the roots are close together, it is possible to skip over some (or all) given a large enough x_delta.

Question{03}: Given the following table of functions, x_{min} = .0, x_{max} = 2, determine the number of iterations required for the given method to find the root for $\varepsilon = .001$. Use xmin for the procedures that only need one starting point

Function	Method	N
Poly from Question{02}	Bisection	1
Poly from Question{02}	False Position	1000 (diverges) (there's two roots in the interval)
Poly from Question{02}	Simple Fixed-Point Iteration	1000 (diverges)
Poly from Question{02}	Newton-Raphson	2
Poly from Question{02}	Secant	6
Function	Method	N
$x^2 - 1$	Bisection	1
$x^2 - 1$	False Position	7
$x^2 - 1$	Simple Fixed-Point Iteration	1000 (diverges)
$x^2 - 1$	Newton-Raphson	3
$x^2 - 1$	Secant	2
Function	Method	N
$x^6 - 1$	Bisection	1
$x^6 - 1$	False Position	91
$x^6 - 1$	Simple Fixed-Point Iteration	1000 (diverges)
$x^6 - 1$	Newton-Raphson	13
$x^6 - 1$	Secant	10
Function	Method	N
$20\cos\left(\frac{x\pi}{2}\right)$	Bisection	1
$20\cos\left(\frac{x\pi}{2}\right)$	False Position	1
$20\cos\left(\frac{x\pi}{2}\right)$	Simple Fixed-Point Iteration	1000 (diverges)
$20\cos\left(\frac{x\pi}{2}\right)$	Newton-Raphson	3
$20\cos\left(\frac{x\pi}{2}\right)$	Secant	3
Function	Method	n
$-e^{x+1}$	Bisection	
$-e^{x+1}$	False Position	
$-e^{x+1}$	Simple Fixed-Point Iteration	
$-e^{x+1}$	Newton-Raphson	
$-e^{x+1}$	Secant	

Observations about the behavior?

Note to Professor Myers: I sent an email regarding this question but have not heard back from you yet – I know that you corrected the last function in the table from what I thought was $-e^{x-1}$ to $-e^{x+1}$, however I also know that all of these functions are supposed to have a root at x = 1, and neither of those two do. So I went ahead and skipped it since I did not hear back.

OBSERVATIONS: I will address each method one by one. Bisection appears as if it appears the best, but this just dumb luck, since the root of all these functions is located at x = 1. Since the search interval is x = 0 to 2 and bisection approximates its root by using the mid point of the bracket (which would be 1), we get lucky and "find" the root first try every time.

False Position: False position is a very good method for some functions and not for others. As we saw in class and is described in the text, when a function has significant curvature one of the bracket points will stay fixed which leads to slow convergence (as in seen for $x^6 - 1$). It does not converge at all for the polynomial because there are two roots in the interval.

Fixed-Point: Unfortunately fixed point iteration does not converge for any of these functions. The book states that if the absolute value of the 1st derivative of g(x) evaluated at the root is greater than 1, than the error will grow with each iteration. Unfortunately, this is the case for all of these functions, when g(x) is determined by just adding x to both sides. I read that if you manipulate the functions in another way it is possible to get a convergent solution however, even if both are algebraically identical.
Newton-Raphson: NR requires a good guess to converge quickly, but can take a while with a bad one. I could not get NR to converge at all with the initial guess at x = xmin (0), so I substituted the guess with a random number in the interval.
Secant: I used the modified secant method with a random guess in the interval and a perturbation fraction of .01. Good performance (or convergence at all) is reliant on a proper value for the perturbation fraction and also other parameters, such as the size of the interval, and whether or not f' = 0 on the interval (in this case it may not converge).

```
format compact;clear;clc;
syms x;
y1 = (x-1)*(x-(3/2))*(x-(7/4))*(x-(15/8))*(x-(31/16));
y2 = x^2 - 1;
y3 = x^6 - 1;
y4 = 20*cos((x*pi)/2);
y5 = -exp(x+1);

threshold = 0.001;
xmin = 0;
xmax = 2;
n_max = 1000;

fprintf("Method\tFxn\t\tIdeal\tApprox\tf(x)\t|e_a|\tIterations\n");
fprintf("================================================================================
==============\n");

%for i = [1:5]
for i = [1:4]
    switch i
        case 1
            f = y1;
            name = 'Poly';
        case 2
            f = y2;
            name = 'x^2 - 1';
        case 3
            f = y3;
            name = 'x^6 - 1';
        case 4
            f = y4;
            name = 'cosine';
        case 5 % skipped this one since error in test???
            f = y5;
            name = '-e^(x+1)';
    end

    % bisection
    method = "Bisect";
    [approx, e_a, actual, y, iter] = ...
        Bisection_by_symbolic(f, x, xmin, xmax, threshold, n_max);
    fprintf("%s\t%s\t%0.4f\t%0.4f\t%0.4f\t%0.4f\t%.0f\n", method, name,
actual(1), approx, y, e_a(1), iter);
    fprintf("------------------------------------------------------------------------
------------------\n");
    % false position
    method = "FaPos";
    [approx, e_a, actual, y, iter] = False_Position_by_symbolic(f, x, xmin,
xmax, threshold, n_max);
    fprintf("%s\t%s\t%0.4f\t%0.4f\t%0.4f\t%0.4f\t%.0f\n", method, name,
actual(1), approx, y, e_a(1), iter);
    fprintf("------------------------------------------------------------------------
------------------\n");
    % fixed point
    method = "FixPnt";
    [approx, e_a, actual, y, iter] = ...
```

```
        Fixed_point_by_symbolic(f,x,threshold,n_max);
    fprintf("%s\t%s\t%0.4f\t%0.4f\t%0.4f\t%0.4f\t%.0f\n", method, name,
actual(1), approx, y, e_a(1), iter);
    fprintf("-------------------------------------------------------------------
------------------\n");
    % newton-raphson
    method = "NR";
    [approx, e_a, actual, y, iter] = ...
        Newton_Raphson_by_symbolic(f,2*rand(1),threshold,n_max);
    fprintf("%s\t\t%s\t%0.4f\t%0.4f\t%0.4f\t%0.4f\t%.0f\n", method, name,
actual(1), approx, y, e_a(1), iter);
    fprintf("-------------------------------------------------------------------
------------------\n");
    % secant
    method = "Sec";
    [approx, e_a, actual, y, iter] = ...
        Secant_by_symbolic(f,2*rand(1),.01,threshold,n_max);
    fprintf("%s\t\t%s\t%0.4f\t%0.4f\t%0.4f\t%0.4f\t%.0f\n", method, name,
actual(1), approx, y, e_a(1), iter);
    fprintf("-------------------------------------------------------------------
------------------\n");
end
```

Results:

Method	Fxn	Ideal	Approx	f(x)	|e_a|	Iterations
Bisect	Poly	1.0000	1.0000	0.0000	0.0000	1
FaPos	Poly	1.0000	1.9482	0.0001	0.9482	1000
FixPnt	Poly	1.0000	-Inf	-Inf	Inf	1000
NR	Poly	1.0000	0.9998	-0.0001	0.0002	2
Sec	Poly	1.0000	0.9995	-0.0001	0.0005	6
Bisect	x^2 - 1	1.0000	1.0000	0.0000	0.0000	1
FaPos	x^2 - 1	1.0000	0.9991	-0.0018	0.0009	7
FixPnt	x^2 - 1	1.0000	Inf	Inf	Inf	1000
NR	x^2 - 1	1.0000	1.0008	0.0016	0.0008	3
Sec	x^2 - 1	1.0000	1.0007	0.0014	0.0007	2
Bisect	x^6 - 1	1.0000	1.0000	0.0000	0.0000	1
FaPos	x^6 - 1	1.0000	0.9990	-0.0058	0.0010	91
FixPnt	x^6 - 1	1.0000	Inf	Inf	Inf	1000
NR	x^6 - 1	1.0000	1.0000	0.0002	0.0000	13
Sec	x^6 - 1	1.0000	1.0000	0.0003	0.0000	10

```
Bisect    cosine    1.0000    1.0000    0.0000    0.0000    1
--------------------------------------------------------------------------------
FaPos     cosine    1.0000    1.0000    0.0000    0.0000    1
--------------------------------------------------------------------------------
FixPnt    cosine    1.0000    -19830    -20.      19831     1000
--------------------------------------------------------------------------------
NR        cosine    1.0000    1.0000    -0.0000   0.0000    3
--------------------------------------------------------------------------------
Sec       cosine    1.0000    0.9997    0.0091    0.0003    3
--------------------------------------------------------------------------------
```