Ayush Gaggar ASSIGNMENT 3 10/22/20

1. $f(x) = 27x^3 - 4x^2 + 6x - 12$

$f'(x) = 81x^2 - 8x + 6 \rightarrow f'(3) = 81(9) - 8(3) + 6 = 711$

FDA: $f'(x) = \frac{27(x+h)^3 - 4(x+h)^2 + 6(x+h) - 12 - 27x^3 + 4x^2 - 6x + 12}{(x+h) - x} + O(h)$

$= \frac{1}{h}\left(27(x+h)^3 - 4(x+h)^2 + 6(x+h) - 27x^3 + 4x^2 - 6x\right) + O(h)$

$f'(3) = \frac{1}{0.1}\left(27(3.1)^3 - 4(3.1)^2 + 6(3.1) - 27(3)^3 + 4(9) - 18\right)$

$= 735.17$

$\varepsilon_T = \left|\frac{(711 - 735.17)}{711}\right| \times 100\% = 3.40\%$

The Taylor Series approximation for Forward Difference Approx. will have a remainder term of magnitude $O(h)$. Given that the function is INCREASING at x, it makes sense that the FDA approximation is an overestimation.

BDA: $f'(x) = \frac{27x^3 - 4x^2 + 6x - 12 - 27(x-h)^3 + 4(x-h)^2 - 6(x-h) + 12}{x - (x-h)} + O(h)$

$= \frac{1}{h}\left(27x^3 - 4x^2 + 6x - 27(x-h)^3 + 4(x-h)^2 - 6(x-h)\right) + O(h)$

$f'(3) = \frac{1}{0.1}\left(27(3)^3 - 4(9) + 18 - 27(2.9)^3 + 4(2.9)^2 - 6(2.9)\right)$

$= 687.37$

$\varepsilon_T = |(711 - 687.37)| / 711 \times 100\% = 3.32\%$

The Taylor Series approximation for Backward Difference Approx. will have a remainder term of magnitude $O(h)$. Given that the function is INCREASING at x, it makes sense that the approximation is an underestimation.

CDA: $f'(x) = \dfrac{27(x+h)^3 - 4(x+h)^2 + 6(x+h) - 27(x-h)^3 + 4(x-h)^2 - 6(x-h)}{2h} - O(h^2)$

$$f'(3) = \frac{27(3.1)^3 - 4(3.1)^2 + 6(3.1) - 27(2.9)^3 + 4(2.9)^2 - 6(2.9)}{0.2}$$

$$= 711.27$$

$$\varepsilon_T = \frac{|711.27 - 711|}{711} \times 100\% = 0.038\%$$

The Taylor Series approximation for Backward Difference Approx. will have a remainder term of magnitude $O(h^2)$, rather than $O(h)$ like the FDA & BDA. Since the error magnitude is $O(h^2)$, the error will be significantly smaller than when $O(h)$. As we can see, the CDA was by far the best approx. of the three.

2. $f'' = \dfrac{f'_{FDA} - f'_{BDA}}{h} = \dfrac{1}{h}\left[\dfrac{1}{h}\left(27(x+h) - 4(x+h)^2 + 6(x+h) - 27x^3 + 4x^2 - 6x\right) - \dfrac{1}{h}\left(27x^3 - x^2 + 6x - 27(x-h)^3 + 4(x-h)^2 - 6(x-h)\right)\right]$

$$f''(3) \approx \left[\frac{1}{0.1}(735.17 - 687.37)\right] = 478$$

$$f''(3)_{actual} = 162(3) - 8 = 478$$

$$\varepsilon_T = \frac{|478 - 478|}{478} \times 100\% = 0\% \text{ Error}$$

3. $CN = \frac{1}{f(\tilde{x})}(f'(\tilde{x})\,\tilde{x})$; CN is used to show how well you can approximate f at $\tilde{x}$.

a) $f(x) = 5$, $\tilde{x} = 2$

$f'(x) = 0$

$CN = \frac{1}{f(2)}(f'(2)(2)) = \frac{1}{5}(2 \cdot 0) = 0 \Rightarrow CN = 0$

Any condition number > 1 means that the function's error will amplify. A condition number of 0 tells us that the output value will <u>not</u> change for a small change in the input

b) $f(x) = e^{-x} + 10$, $\tilde{x} = 5$

$f'(x) = -e^{-x} \Rightarrow f'(5) = -e^{-5}$

$CN = \frac{1}{e^{-5}+10}(-5e^{-5}) = -0.003367$

A small change in the input would not change the output by much. In other words, the function is not very sensitive.

c) $f(x) = \tan(x)$, $\tilde{x} = 0.9999\frac{\pi}{2}$

$f'(x) = \sec^2 x$; $f'(\tilde{x})$ approaches infinity

$CN = \frac{1}{\tan(\tilde{x})}(\tilde{x}\, f'(\tilde{x}))$ approaches infinity

With such a large CN, a small change in input would result in a massively different output value. At this x, the function is very sensitive; if an approximation were used, the error would continue to amplify.

Assignment 3

Table of Contents

Problem 4

```
clc; close all; clear all;
i = 0;
upper_bound_error = 0.01;
approx_error_bi = [10000]; %placeholder since first approx. error = N/
A
evals_b = 0;

% Bisection method
f_lower = p(0.000001);
evals_b = evals_b + 1;
f_upper = p(1);
evals_b = evals_b + 1;

if f_lower == 0
    root = 0;
end
if f_upper == 0
    root = 1;
end
if f_upper*f_lower > 0
    fprintf('Cannot use bisection method if both brackets are the same
 sign.');
end
upper = 1;
lower = 0.000001;
root = .5*(upper + lower);
f_root = p(root);
evals_b = evals_b + 1;

while approx_error_bi(end) > 0.01
    i = i + 1;
    if f_lower*f_root == 0
        root = lower;
        break
    end
    if f_lower*f_root < 0
        upper = root;
    else
        lower = root;
```

```
    end
    f_lower = p(lower);
    evals_b = evals_b + 1;

    f_upper = p(upper);
    evals_b = evals_b + 1;

    oldroot = root;
    root = .5*(upper + lower);
    approx_error_bi = [approx_error_bi, abs((root-oldroot)/root*100)];
    f_root = p(root);
    evals_b = evals_b + 1;
end

fprintf('Using the bisection method: \n');
fprintf('a) The APR is %f%%\n', root*100*12);
fprintf('b) The number of iterations is %d\n', i);
fprintf('c) The total number of evaluations made: %d\n\n', evals_b);

% False Position Method
approx_error_fp = [10000];
ifp = 0;
evals_p = 0;

f_lower = p(0.000001);
evals_p = evals_p + 1;
f_upper = p(1);
evals_p = evals_p + 1;

if f_lower == 0
    root = 0;
end
if f_upper == 0
    root = 1;
end
if f_upper*f_lower > 0
    fprintf('Cannot use false position method if both brackets are the
 same sign.');
end
upper = 1;
lower = 0.000001;
root = .5*(upper + lower);
f_root = p(root);
evals_p = evals_p + 1;

while approx_error_fp(end) > 0.01
    ifp = ifp + 1;
    if f_lower*f_root == 0
        root = lower;
        break
    end
    if f_lower*f_root < 0
        upper = root;
    else
```

```
        lower = root;
    end

    f_lower = p(lower);
    evals_p = evals_p + 1;
    f_upper = p(upper);
    evals_b = evals_b + 1;

    oldroot = root;
    root = (upper + f_upper*(upper - lower)/(f_lower - f_upper));
    approx_error_fp = [approx_error_fp, abs((root-oldroot)/root*100)];
    f_root = p(root);
    evals_p = evals_p + 1;
end

fprintf('Using the false position method: \n');
fprintf('a) The APR is %f%%\n', root*100*12);
fprintf('b) The number of iterations is %d\n', ifp);
fprintf('c) The total number of evaluations made: %d\n\n', evals_p);
```

Problem 5

```
clc; close all; clear all;
wo = 2.3; %kN/cm
L = 750; %cm
E = 55000; %kN/cm^2
I = 25000; %cm^4
max_delta = -1;

f_lower = displacement(0);
f_upper = displacement(L - 1);
if f_lower == 0
    max_delta = 0;
end
if f_upper == 0
    max_delta = L - 0.01;
end
if f_upper*f_lower > 0
    fprintf('Cannot use bisection method if both brackets are the same
 sign.');
end
upper = 750;
lower = 0;

while 1
    root = .5*(upper + lower);
    f_root = displacement(root);
    if abs(f_lower*f_root) < 1*10^-15 % check if "equals 0"
        max_delta = lower;
        break
    end
    if f_lower*f_root < 0
        upper = root;
```

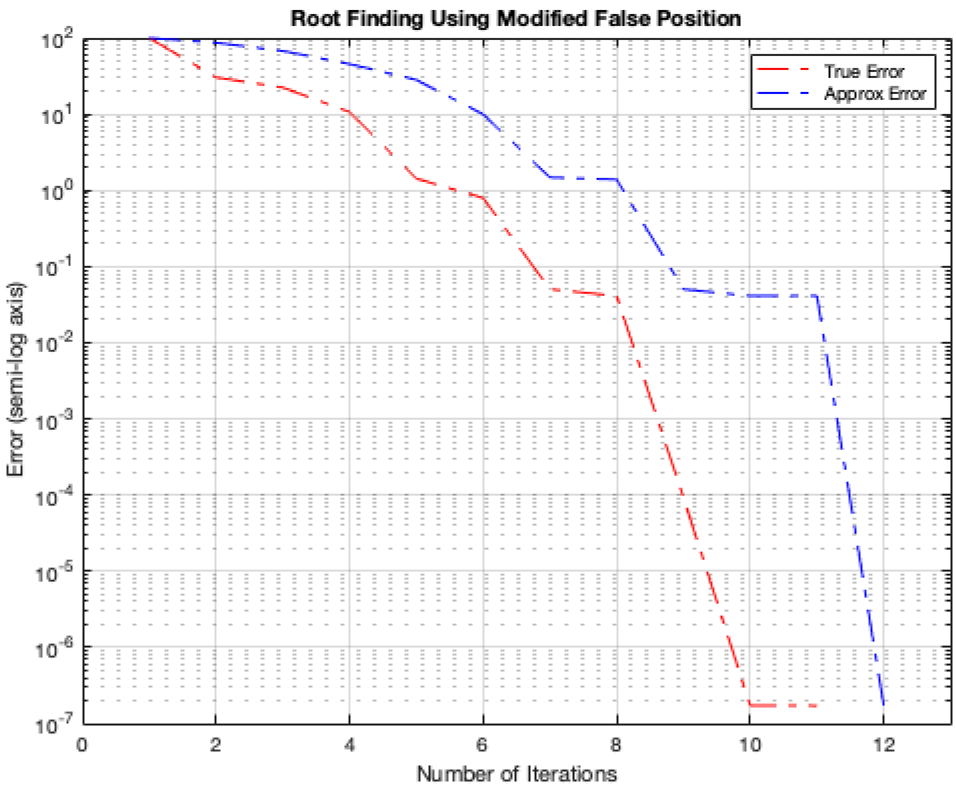
```
    else
        lower = root;
    end
end

fprintf('The maximum deflection occurs at %f cm\n', max_delta);
deflection = wo/(120*E*I*L)*(-(max_delta^5) + 2*L^2*(max_delta)^3 -
L^4*max_delta);
fprintf('The maximum deflection is %f cm\n', deflection);

The maximum deflection occurs at 335.410196 cm
The maximum deflection is -1.262362 cm
```

Problem 6

```
clc; close all; clear all;
upper_bound_error = 0.001;
true_error = [100];
approx_error = [100];
root = [];
i = 1;
iu = 0;
il = 0;

upper = 1.3;
lower = 0;
f_lower = f(lower);
f_upper = f(upper);

if f_lower == 0
    root = 0;
end
if f_upper == 0
    root = 1.3;
end
if f_upper*f_lower > 0
    fprintf('Cannot use false position method if both brackets are the
 same sign.');
end
root = [root, .5*(upper+lower)];
f_root = f(root(end));

while approx_error(end) > upper_bound_error
    i = i+1;
    if f_lower*f_root == 0
        root(end) = lower;
        break
    end
    if f_lower*f_root < 0
        upper = root(end);
        f_upper = f(upper);
        iu = 0;
        il = il + 1;
```

```
        if il >= 2
            f_lower = f_lower/2;
        end
    else
        lower = root(end);
        f_lower = f(lower);
        il = 0;
        iu = iu + 1;
        if iu >= 2
            f_upper = f_upper/2;
        end
    end

    oldroot = root(end);
    root = [root, upper + f_upper*(upper - lower)/(f_lower -
 f_upper)];
    ea1 = abs((root(end) - oldroot)/root(end)*100);
    eal = abs((root(end) - lower)/root(end)*100);
    eaf = abs((root(end) - upper)/root(end)*100);
    % The above 3 lines of code are necessary to make sure
    % the algorithm does not underestimate the error.
    approx_error = [approx_error, max([ea1, eal, eaf])];
    true_error = [true_error, abs((1-root(end))*100)];
    f_root = f(root(end));
end

fprintf('The root of the given function is at %f\n', root(end))

semilogy(linspace(1, 12, 12), true_error, '-.r');
grid on
hold on
semilogy(linspace(1, 12, 12), approx_error, '-.b');
legend('True Error', 'Approx Error');
title('Root Finding Using Modified False Position');
xlabel('Number of Iterations');
ylabel('Error (semi-log axis)');
xlim([0, 13]);

% Note: True Error is also a 1x12 vector, except the value of the true
% error for the final root is equal to 0, which is not plotted on a
% semilog plot.
```

The root of the given function is at 1.000000



Function Calls from Each Section

Problem 4

```
function payment = p(i)
payment = 39190*i*(i+1)^72/((1+i)^72 - 1) - 587;
end

% Problem 5
function deltax = displacement(x)
wo = 2.3; %kN/cm
L = 750; %cm
E = 55000; %kN/cm^2
I = 25000; %cm^4
deltax = wo/(120*E*I*L)*(-5*x^4 + 6*L^2*x^2 - L^4);
end

% Problem 6
function y = f(x)
y = x^10 - 1;
end
```

Using the bisection method:
a) The APR is 2.516466%
b) The number of iterations is 22
c) The total number of evaluations made: 69

```
Using the false position method:
a) The APR is 2.516238%
b) The number of iterations is 12
c) The total number of evaluations made: 27
```

Published with MATLAB® R2020a