Damian Franco

CS 375 – 001

Homework #3:

1. For problem 1, we were given a prompt and equation to solve for the *optimal value* of *h.* In homework 1 we approximated a derivative, now we must compare the results of the exact stencil and computer stencil. I based my understanding of this problem off the hint Professor Lau gave us in the Slack channel, thank you for that and all the other guidance. The MATLAB function below will approximate accordingly. It will first set up the variables for the equation we are approximating and then it will set up the exact stencil equation and the computer stencil equation based off the f(x) equation we were given. Then the rest will just approximate based off these equations and graph them versus *h*. The graphs show that when compared to problem 1 on homework 1, that this is a *highly more efficient* way to find the derivative of a function.

**Source Code**

*HW3\_1.m:*

1 %

2 % function: HW3\_1

3 %

4 % Set up and arithmetic for homework 3, number 1.

5 % This is based off of the hint Professor Lau

6 % gave us in the Slack channel. Therefore, this function

7 % is the estimate of the optimal value of h for the

8 % derivative of f.

9 %

10 % Variable set up

11 x = pi;

12 f = @(x) exp(sin(x));

13 h = logspace(-1,-9,9);

14

15 % Orginal eqs

16 ES = (f(x + h) - f(x)) ./ h;

17 CS = ES + (1e-16) ./ h;

18

19 % New variable set up

20 c = 0.9952696331; %f''(h)

21 M = 2;

22

23 % Inequality set up

24 b1 = abs(ES - (exp(sin(x)) \* cos(x)));

25 b2 = ((1e-16 ./ h) + ((1/2) \* M \* h));

26

27 % Graph set up

28 subplot(3,1,1)

29 title('Plot of abs(ES - (exp(sin(x)) \* cos(x)))')

30 xlabel('x')

31 ylabel('y')

32 plot(b1, '-o')

33 subplot(3,1,2)

34 xlabel('x')

35 ylabel('y')

36 title('Plot of ((1e-16 ./ h) + ((1/2) \* M \* h))')

37 plot(b2, '-o')

38 subplot(3,1,3)

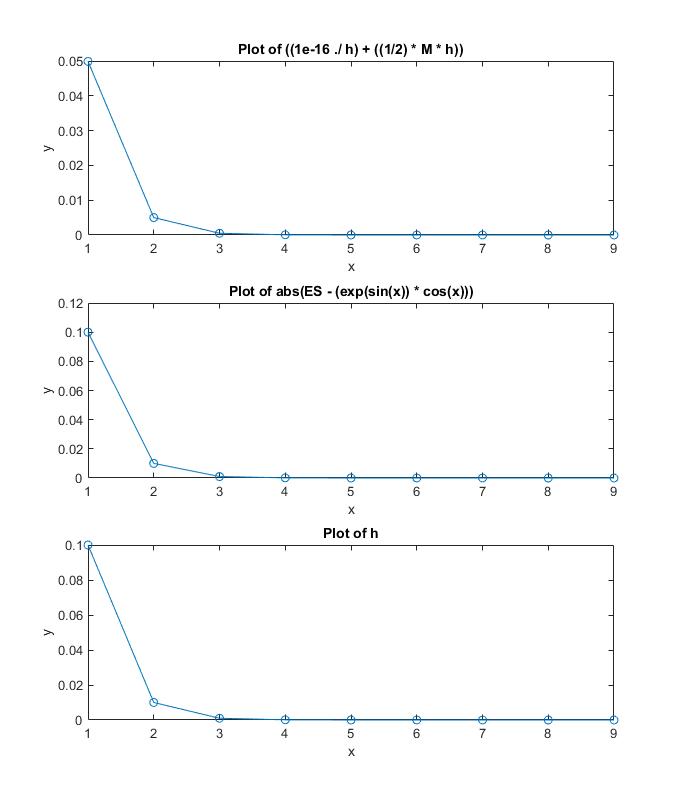
39 xlabel('x')

40 ylabel('y')

41 title('Plot of h')

42 plot(h, '-o')

**Graphs**

*HW3-1-Graphs.jpg:*

1. For question 2 on the homework, we had to compare the roots of function through two different methods. The first was the bisection method we had learned in the recordings by Professor Lau. He provided us with a clear understanding of how to do the bisection method and make a function to do so in MATLAB, and that is what the *bisection.m* code is based on. The second, was the fzero function call method and check where we get the root at between an interval. I set both functions up in MATLAB and compared the answers on a table. The roots seem to be calculating *close* to each other beside a *few precision differences* on the end of the roots, but besides that, both methods look pretty good. My work for this problem is shown below.

**Source code:**

*HW\_2.m*:

1 %

2 % function HW3\_2

3 % This sets up and runs all the arithmetic

4 % behind number 2 on HW3

5 %

6 y = @(x) exp(x+2) + x^3 - x;

7 ftol = 1e-9;

8 MyOptions = optimset('TolX', ftol);

9 % fzero method

10 % r0 = -1 for root1

11 % r0 = 0 for root2

12 % r0 = 1 for root3

13 r0 = 100;

14 r = fzero(y, r0, MyOptions);

15 % bisection method

16 % [-2, -1] for root1

17 % [0, 0.5] for root2

18 % [0.5, 1] for root3

19 bisection(y, 0.5, 1)

*bisection.m*:

1 function [m k] = bisection(f, a, b, tol, kmax)

2 %

3 % function [m k] = bisection(f, a, b, tol, kmax)

4 % This function will approximate the root of the

5 % function being passed through.

6 %

7 switch nargin

8 case 4,

9 kmax = 1e5;

10 case 3,

11 tol = 1e-8;

12 kmax = 1e5;

13 case 5,

14 % fall through and assign nothing

15 otherwise,

16 error('Wrong # or args')

17 end

18

19 if (b <= a)

20 error('b > a not satisfied');

21 end

23 fa = f(a); if fa == 0, m = a; return; end;

24 fb = f(b); if fb == 0, m = b; return; end;

26 if (fa \* fb >= 0)

27 error('Intial interval error’);

29 end

30

31 k = 0;

32 m = 0.5 \* (a + b);

33 err = 0.5 \* (b - a);

34 tol2 = tol + eps \* max(abs(a), abs(b));

36 while (err >= tol2)

37 fm = f(m);

38 k = k + 1;

39 if (fm == 0)

40 return;

41 end

42 if (fa \* fm < 0)

43 b = m;

44 fb = fm;

45 else

46 a = m;

47 fa = fm;

48 end

50 m = 0.5 \* (a + b);

51 err = 0.5 \* (b - a);

52 tol2 = tol + eps \* max(abs(a), abs(b));

53

54 if (k >= kmax)

55 disp(['Reached max iterations ', num2str(kmax), '. Error is ',

56 num2str(err)]);

57 end

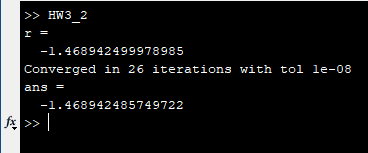
58 end

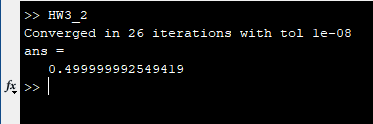
59 disp(['Converged in ', num2str(k), ' iterations with tol ',

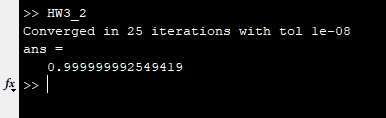
60 num2str(tol)]);

**Outputs :**

*HW3\_2-OUTS.jpg*:

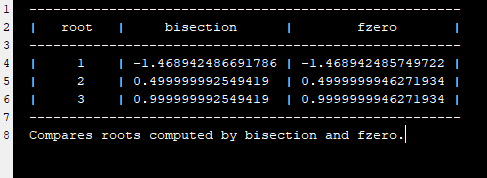






**Table Comparison:**

*HW3\_2-Table.txt:*



1. Question 3 on the homework asks us to find the fixed point of each function given in part (a) and part (b) and decide whether it is locally convergent to that fixed points. These given functions did not look too crazy, so my initial guess was that both were going to converge locally in very few iterations. I followed Professor Lau’s rules for fixed point in the narrations of root2 to solve this one and I came up with the following:

*Part (a):*

For this part, the MATLAB function *iteration.m* shows that function has a fixed point at the value *1.5*. The function converges with the tolerance of 1e-8 in *two iterations*.

*Part (b):*

For this part, we do the same by plugging in the equation to the MATLAB function *iteration.m*. This shows that function has a fixed point at the value *-0.5*. The function converges with the tolerance of 1e-8 in *two iterations*, just like in part (a).

**Source code:**

*HW3\_3.m*:

1 %

2 % function HW3\_3

3 % This sets up and runs all the arithmetic

4 % behind number 3 on HW3

5 %

6 % 3a:

7 a = @(x) x^2 - 3/2 \* x + 3/2;

8 % 3b:

9 b = @(x) x^2 + 1/2 \* x - 1/2;

10 % Test cases set up

11 x0 = 0;

12 tol = 1e-8;

13 kmax = 1e5;

14 % Test

15 iteration(b, x0, tol, kmax)

*iteration.m*:

1 function x = iteration(f, x0, tol, kmax)

2 %

3 % function x = iteration(f, x0, tol, kmax)

4 % This function will find the fixed point of the function

5 % and show whether it converges to it or if it computes

6 % an error.

7 %

8 switch nargin

9 case 2,

10 tol = 1e-8;

11 kmax = 1e5;

12 case 3,

13 kmax = 1e5;

14 case 4,

15 % fall through and assign nothing

16 otherwise,

17 error('Wrong # or args')

18 end

19 x = x0;

20 err = 100;

21 k = 0;

22 while (err >= tol)

23 format long g;

24 if (k < 20)

25 disp([k x0]);

26 end

27 x = f(x0);

28 err = abs(x - x0);

29 x0 = x;

30 k = k + 1;

31 if (k >= kmax)

32 disp(['No convergence after ', num2str(kmax) ' iterations'])

33 disp(['Error at this stage is ' num2str(err)]);

34 return;

35 end

36 end

37 if (isfinite(x))

38 disp(['Converged in ', num2str(k) ' iterations with tol ' ...

39 num2str(tol)])

40 else

41 disp(['No convergence after ', num2str(k) ' iterations. Iterations'

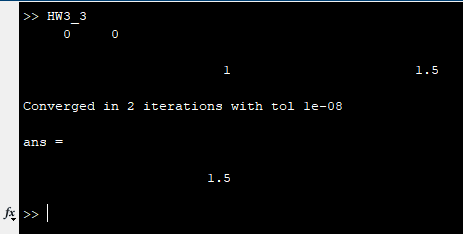
42 ...

43 'overflowed to infinity'])

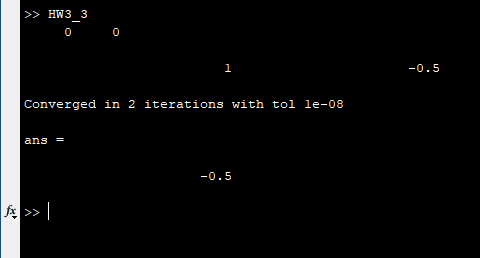
44 end

**Outputs:**

*HW3\_3-PART-A.jpg*



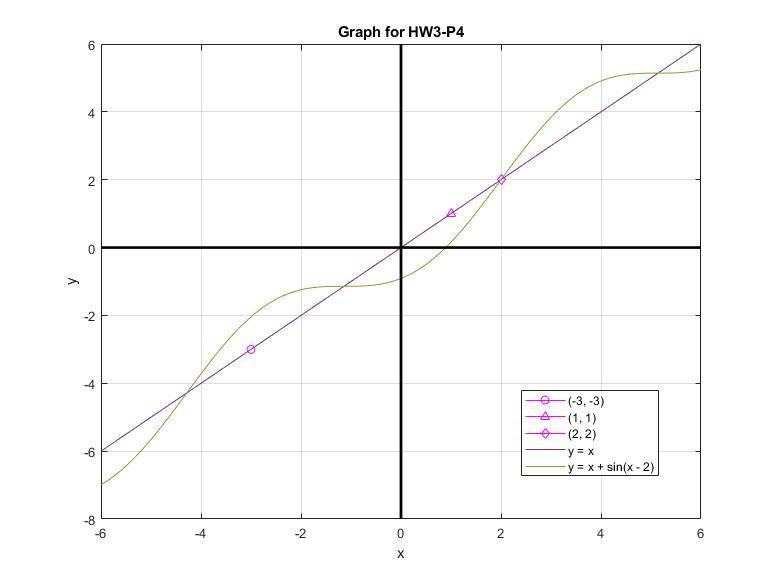
*HW3\_3-PART-B.jpg*



1. For problem 4, we were given a set of fixed points on the function phi(x). We were then given some rules to follow to help us solve this problem. First, the fixed points are set at (-3, -3), (1, 1) and (2, 2). Second, the slope at (-3, -3) is equal to 2.4. Lastly the fixed point 2 converges to 2, therefore -1 < fphi’(2) < 0. This problem was, once again, based off Professor Lau’s help in the Slack channel. First, I drew the line y = x and plotted the fixed points on them. Then I noticed a pattern and figured out the equation must be y = x + sin(x – FP) for FP is equal to fixed point. Then I plotted that in MATLAB. After, I took that equation to Desmos calculator online and calculated the fphi’(1) must be equal to approximately *1.052335956*. The graphical representation is shown below, as well as the MATLAB code and the Desmos calculator.

**Desmos calculation:**

**Graphical Representation:**

*HW\_3-4-Graph.jpg:*

**Source Code:**

*HW3\_4.m:*

1 %

2 % function HW3\_3

3 % This sets up and runs all the arithmetic

4 % behind number 4 on HW3

5 %

6 % Set up equation

7 x = linspace(-6,6,100);

8 f = x;

9 fphi = x + sin(x - 2); % Guess of f(phi)

10 xaxis = 0;

11 yaxis = 0;

12 % Set up graph

13 plot(-3, -3, '-mo')

14 hold on

15 plot(1, 1, '-m^')

16 plot(2, 2, '-md')

17 plot(x, f);

18 plot(x, fphi);

19 line([0,0], ylim, 'Color', 'k', 'LineWidth', 2); % y axis

20 line(xlim, [0,0], 'Color', 'k', 'LineWidth', 2); % x axis

21 title('Graph for HW3-P4')

22 xlabel('x')

23 ylabel('y')

24 legend('(-3, -3)', '(1, 1)', '(2, 2)', 'y = x', 'y = x + sin(x - 2)')

25 hold off

26 grid on