Tristan Bouchard

260747124

February 3, 2019

ECSE 443, Introduction to Numerical Methods in Electrical Engineering

Assignment 1

Please note that the following assignment contains snippets of MATLAB code to aid the reader in understanding the procedure and logic in the problem. The full MATLAB script for all snapshots of code below can be found at the end of this assignment.

Question 1) (10 Marks)

$$f(x) = x * \left(\sqrt{x} - \sqrt{x-1}\right)$$

a)

```
format long
syms x;
f(x) = x*(sqrt(x) - sqrt(x-1));
X = [10;1000;1000000];

x1 = double(f(X(1)));
x2 = double(f(X(2)));
x3 = double(f(X(3)));

table = {"X:", 10, 1000, 1000000 ; "F(x)=", x1, x2, x3 }
```

```
table = 2×4 cell array
    {["X:"    ]}    {[               10]}    {[               1000]}    {[               1000000]}
    {["F(x)="]}    {[1.622776601683793]}    {[15.815343125576774]}    {[5.000001250000625e+02]}
```

Figure 1: MATLAB Script for question 1 a)

X=	10	1000	1000000
F(x)=	1.622776601683793	15.815343125576774	5.000001250000625e+02

b)

X = 10

$$f(10) = 10 * \left(\sqrt{10} - \sqrt{10-1}\right)$$

$$f(10) = 10 * (3.16228 - 3.00000)$$

$$f(10) = 10 * (0.16228)$$

$$f(10) = 1.6228$$

X = 1000

$$f(1000) = 1000 * \left(\sqrt{1000} - \sqrt{1000-1}\right)$$

$$f(1000) = 1000 * (31.6228 - 31.6070\ )$$

$$f(1000) = 1000 * (0.0158)$$

$$f(1000) = 15.8$$

X = 1000000

$$f(1000000) = 1000000 * \left(\sqrt{1000000} - \sqrt{1000000-1}\right)$$

$$f(1000000) = 1000000 * (1000.00 - 1000.00)$$

$$f(1000000) = 1000000 * (0.000)$$

$$f(1000000) = 0$$

X=	10	1000	1000000
F(x)=	1.6228	15.8	0

c)

$$Absolute\ Error = (Approximate\ Value) - (True\ Value)$$

$$Relative\ Error = \frac{Absolute\ Error}{True\ Value}$$

```
mat_val = [x1 x2 x3];
calc_val = [1.6228 15.8 0];
abs_err = calc_val - mat_val

abs_err = 1×3
10^2 ×
     0.000000233983162   -0.000153431255768   -5.000001250000625

rel_err = abs_err ./ mat_val

rel_err = 1×3
     0.000014418692125   -0.000970141808176   -1.000000000000000
```

Figure 2: Script for question 1 c)

Based on the above equations, here are the results for the absolute error and relative error between part a) and b):

X=	10	1000	1000000
Absolute Error	2.339831620679078e-05	0.015343125576774	-5.000001250000625e+02
Relative Error	1.441869212466626e-05	-9.701418081761649e-04	-1

The error above is mainly due to rounding errors, as there are many operations in the formula involving square root computations. This yields irrational numbers with decimal places that need to be rounded, according to the six-significant-figure rule. The rounding error also caused the value for x = 1000000 to be completely erroneous, which is why the absolute error is so large and the relative error is -1.

d) Simplifying the equation yields:

$$f(x) = x * \left(\sqrt{x} - \sqrt{x-1}\right) * \frac{\sqrt{x} + \sqrt{x-1}}{\sqrt{x} + \sqrt{x-1}}$$

$$f(x) = x * \left(\frac{x-(x-1)}{\sqrt{x} + \sqrt{x-1}}\right)$$

$$f(x) = \frac{x}{\sqrt{x} + \sqrt{x-1}}$$

Calculations for:

X = 10

$$f(10) = \frac{10}{\sqrt{10} + \sqrt{10-1}}$$

$$f(10) = \frac{10}{3.16228 + 3.00000}$$

$$f(10) = 1.62278$$

X = 1000

$$f(1000) = \frac{1000}{\sqrt{1000} + \sqrt{1000-1}}$$

$$f(1000) = \frac{1000}{31.6228 + 31.6070}$$

$$f(1000) = 15.8153$$

X = 1000000

$$f(1000000) = \frac{1000000}{\sqrt{1000000} + \sqrt{1000000-1}}$$

$$f(1000000) = \frac{1000000}{(2000.00)}$$

$$f(1000000) = 500.00$$

X=	10	1000	1000000
F(x)=	1.62278	15.8153	500.00

e)

```
calc_val2 = [1.62278 15.8153 500]
```

```
calc_val2 = 1×3
10^2 ×
    0.016227800000000   0.158153000000000   5.000000000000000
```

```
abs_err2 = calc_val2 - mat_val
```

```
abs_err2 = 1×3
10^-3 ×
    0.003398316206660  -0.043125576773662  -0.125000062496383
```

```
rel_err2 = abs_err2 ./ mat_val
```

```
rel_err2 = 1×3
10^-5 ×
    0.209413680424876  -0.272681891447036  -0.025000006249272
```

Figure 3: MATLAB Script used in 1 e)

X=	10	1000	1000000
Absolute Error	3.398316206659757e-06	-4.312557677366158e-05	-1.250000624963832e-04
Relative Error	2.094136804248757e-06	-2.726818914470363e-06	-2.500000624927195e-07

The error introduced in this section is generally much smaller than in part c). This is due to the calculations that did not involve such small numbers, as well as that the information to calculate the values was properly encoded in the entire range of the significant figures, not just the small latter portion. The principal source of error is rounding errors, but they are much more minimal than the error introduced in part c).

Question 2) (**10 Marks**)

a)

```
syms x;
func(x) = (1-cos(x))/sin(x)
```

func(x) =

$$-\frac{\cos(x)-1}{\sin(x)}$$

```
res = double(func(0.007))
```

res =

0.003500014291737

Figure 4: MATLAB Script used for question 2 a)

X=	0.007
F(x) =	0.003500014291737

b)

$$f(x) = \frac{1-\cos(x)}{\sin(x)}$$

$$f(0.007) = \frac{1-\cos(0.007)}{\sin(0.007)}$$

$$f(0.007) = \frac{1-1}{0.000122173}$$

$$f(0.007) = 0$$

X=	0.007
F(x) =	0

c)

As in question 1 c):

$$Absolute\ Error = (Approximate\ Value) - (True\ Value)$$

$$Relative\ Error = \frac{Absolute\ Error}{True\ Value}$$

```
syms x;
func(x) = (1-cos(x))/sin(x)
```

func(x) =

$$-\frac{\cos(x)-1}{\sin(x)}$$

```
res = double(func(0.007))
```

```
res =
   0.003500014291737
```

```
calc_valq2 = 0;
abs_errq2 = calc_valq2 - res
```

```
abs_errq2 =
  -0.003500014291737
```

```
rel_errq2 =  abs_errq2 / res
```

```
rel_errq2 =
    -1
```

Figure 5: MATLAB Script used for question 2 c)

X=	0.007
Absolute Error	-0.003500014291737
Relative Error	-1

Due to the six significant digit limitation, the result computed by the calculator rounded up from 0.999999992 to 1.00000. Such, this is a rounding error.

d)

$$f(x) = \frac{1-\cos(x)}{\sin(x)} * \frac{1+\cos(x)}{1+cosx}$$

$$f(x) = \frac{1-\cos(x)^2}{\sin(x)\,(1+\cos(x))}$$

Using the trigonometric identity $1 = \sin(x)^2 + \cos(x)^2$:

$$f(x) = \frac{\sin(x)^2}{\sin(x)\,(1+\cos(x))}$$

$$f(x) = \frac{\sin(x)}{1+\cos(x)}$$

Calculating the value of $f(0.007)$:

$$f(0.007) = \frac{\sin(0.07)}{1+\cos(0.07)}$$

$$f(0.007) = \frac{0.00122173}{1+1}$$

$$f(0.007) = \ 0.0006110865$$

$$f(0.007) = \ 0.000611087$$

X=	0.007
F(x) =	0.000611087

e)

As in part c), using the value found in part a) as true:

X=	0.007
Absolute Error	-0.002888927291737
Relative Error	-0.825404427221130

The sources of error here are yet again due to rounding, as was found in part c) of this question. The magnitude of the error is due to working with very small numbers which are greatly modified by rounding them up or down, which affects the final result.

Question 3) (**14 Marks**)

a)

$$f(x) = e^{(-4x)}cos(6x)$$

By the product rule, the derivative

$$\frac{df(x)}{dx} = \ -4 * e^{-4x} * \cos(6x) - \ 6 * e^{-4x} * \sin(6x)$$

Using MATLAB to compute the above function at x = 0.5:

```
syms x;
f(x) = -(4*exp(-4*x)*cos(6*x) + 6*exp(-4*x)*sin(6*x));
result = (f(0.5))
```

result = $-4\cos(3)\,e^{-2} - 6\,e^{-2}\sin(3)$

```
double(result)
```

```
ans =
   0.421332562151359
```

$$\frac{df(0.5)}{dx} = 0.421332562151359$$

b)

```
f_original(x) = exp(-4*x)*cos(6*x)
```

f_original(x) = $\cos(6\,x)\,e^{-4\,x}$

```
h = 0.01;
derivative = double((f_original(0.5+h) - f_original(0.5))/(h))
```

```
derivative =
   0.438478802436531
```

c)

Given that the Taylor series expansions for the following functions are:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

Then the expression for the function:

$$f(x) = (1 - 4x + 8x^2)(1 - 18x^2 + 54x^4)$$

$$f(x) = (1 - 18x^2 + 54x^4 - 4x + 72x^3 - 216x^5 + 8x^2 + 144x^4 + 432x^6)$$

$$f(x) = \ 1 - 4x + x^2(8 - 18) + x^3(72) + x^4(54 + 144) + \ x^5(-216) + x^6(432)$$

Keeping only the first three lower order polynomials:

$$f(x) = 1 - 4x - 10x^2$$

Then, using this function to compute the approximation with a step size of h = 0.01:

```
syms x;
h = 0.01;
f(x) = 1 - 4*x - 10*x^(2);
der = ((f(x + h) - f(x))/h)
```

der =

$$1000\,x^2 - 1000\,\left(x + \frac{1}{100}\right)^2 - 4$$

Figure 6: MATLAB script used in question 3 c)

So, the first order approximation of the derivative of the function above is:

$$\frac{df(x)}{dx} = 1000x^2 - 1000\left(x + \frac{1}{100}\right)^2 - 4$$

d)

Using the result obtained in part a) for the derivative computed using calculus, the “true” reference value is equal to 0.421332562151359.

```
% This algorithm will be used to compute the the step size h which results
% in the least error when computing the first order approximation of the
% derivative of f(x)
syms x;
f(x) = exp(-4*x)*cos(6*x);
% Define the "true" value obtained in part a)
true_val = 0.421332562151359;
% Define the value for which we are trying to compute the derivative
x = 0.5;
last_err = 100; % Set error very large for first loop
step = 0.001; % Set starting step size
cont = 1; % Set loop variable

while (cont == 1)
    % For each value of h, compute the first order approximation
    temp = ((f(x + step) - f(x))/step);
    derivative = double(temp);
    err = derivative - true_val;
    if( err >= last_err)
        % Break out of loop, error has started to increase
        cont = 0;
        return_val = [last_err, last_step, last_der];
    end
    last_der = derivative;
    last_step = step;
    last_err = err;

    step = step / 2;
end
titles = {"Computed Error", "Computed Step", "Computed Derivative"};
disp(titles), disp(return_val);
```

```
    ["Computed Error"]    ["Computed Step"]    ["Computed Derivative"]

  -0.000000001082816   0.000000001907349   0.421332561068543
```

Figure 7: MATLAB Script used to compute optimal step size

Minimum Error	Optimal Step	Computed Derivative
-1.082816059039260e-09	1.907348632812500e-09	0.421332561068543

Question 4) (12 Marks)

a)

```
A  = [4 -2 -3 6; 6 -7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1]
```

```
A = 4×4
     4.000000000000000  -2.000000000000000  -3.000000000000000   6.000000000000000
     6.000000000000000  -7.000000000000000   6.500000000000000  -6.000000000000000
     1.000000000000000   7.500000000000000   6.250000000000000   5.500000000000000
   -12.000000000000000  22.000000000000000  15.500000000000000  -1.000000000000000
```

```
B = inv(A)*[12; -6.5; 16; 12]
```

```
B = 4×1
    -4.770833333333334
    -4.450757575757576
     3.757575757575756
     5.575757575757576
```

Figure 8: MATLAB Script used in question 4 a)

b)

```
% Gaussian elimination for matrix.
% Begin by augmenting the matrix from part a) with its solutions
B = [12; -6.5; 16; 12];
concat = [A B]
```

```
concat = 4×5
     4.000000000000000  -2.000000000000000  -3.000000000000000   6.000000000000000  12.000000000000000
     6.000000000000000  -7.000000000000000   6.500000000000000  -6.000000000000000  -6.500000000000000
     1.000000000000000   7.500000000000000   6.250000000000000   5.500000000000000  16.000000000000000
   -12.000000000000000  22.000000000000000  15.500000000000000  -1.000000000000000  12.000000000000000
```

```
% This script assumes that the matrix is a 4x4 (augmented)
% Begin iterating at the second row
for row=2:4
    % For all rows
    for j=row:4
        % Compute ratio by which to subtract first row from above row
        subs = (concat(j,row-1)/concat(row-1,row-1))*concat(row-1,:);
        concat(j, :) = concat(j,:) - subs;
    end
end
disp(concat);
```

```
   4.000000000000000  -2.000000000000000  -3.000000000000000   6.000000000000000  12.000000000000000
                   0  -4.000000000000000  11.000000000000000 -15.000000000000000 -24.500000000000000
                   0                   0  29.000000000000000 -26.000000000000000 -36.000000000000000
                   0                   0                   0   2.275862068965516  12.689655172413794
```

```
% Then, clear the rest of the rows by solving for each value of x by
% working our way up the matrix from the bottom row
% Create empty array in which to store answers
answ = zeros(1,4);
for i=1:4
    j = 5-i;
    % Solve for xi in each row
    answ(j) = (concat(j,5)-concat(j,4)*answ(4) - concat(j,3)*answ(3) - concat(j,2)*answ(2))/concat(j,j);
end
for i=1:length(answ)
    disp("x"+i + ":" + char(vpa(answ(i),15))) %, disp(answ(i));
end
```

```
x1:-4.77083333333333
x2:-4.45075757575758
x3:3.75757575757576
x4:5.57575757575758
```

Figure 9: MATLAB Script used for question 4b)

c)

```
% Calculate absolute error between results in a) and b)
% Have to take transpose of answ vector to get correct dimensions.
absolute_err = transpose(answ) - B_res
```

```
absolute_err = 4×1
10^-14 ×
   -0.444089209850063
   -0.355271367880050
    0.444089209850063
    0.266453525910038
```

Figure 10: MATLAB Script used for question 4c)

Question 5) (5 Marks)

a)

```
syms x;
f = exp(sin(x));
t = taylor(f,x)
```

t =

$$-\frac{x^5}{15}-\frac{x^4}{8}+\frac{x^2}{2}+x+1$$

```
f2(x)  = 1 + x + x^2/2
```

f2(x) =

$$\frac{x^2}{2}+x+1$$

b)

```
% Using the approximation in f2, compute f(0.01)
answer1 = f2(0.01);
disp(double(answer1));
```

```
1.010050000000000
```

Figure 11: MATLAB script used for question 5 a) and b)

c)

$$f(0.07) = e^{\sin(0.07)}$$

$$f(0.07) = e^{0.00122173}$$

$$f(0.07) = 1.00122248$$

d)

Comparing the results in part b) and c), many things are apparent. Obviously, approximating the function $f(x) = e^{\sin(x)}$ with only three terms of the infinite Taylor series is not going to be the most precise operation, which is why the result obtained in b) is quite different from the result obtained in c). Even with the rounding error introduced in part c), the result is probably more realistic with the exact value, as much more of the series of both functions are used in the calculator to compute the value.

ECSE 443 - Assigment 1

Tristan Bouchard, 260747124

Question 1) (10 Marks)

a)

```
format long
syms x;
f(x) = x*(sqrt(x) - sqrt(x-1));
X = [10;1000;1000000];

x1 = double(f(X(1)));
x2 = double(f(X(2)));
x3 = double(f(X(3)));

table = {"X:", 10, 1000, 1000000 ; "F(x)=", x1, x2, x3 }
```

```
table = 2×4 cell array
    {["X:"   ]}    {[                10]}    {[              1000]}    {[               1000000]}
    {["F(x)="]}    {[1.622776601683793]}    {[15.815343125576774]}    {[5.000001250000625e+02]}
```

c)

```
mat_val = [x1 x2 x3];
calc_val = [1.6228 15.8 0];
abs_err = calc_val - mat_val
```

```
abs_err = 1×3

10^2 ×
   0.000000233983162  -0.000153431255768  -5.000001250000625
```

```
rel_err = abs_err ./ mat_val
```

```
rel_err = 1×3

   0.000014418692125  -0.000970141808176  -1.000000000000000
```

e)

```
calc_val2 = [1.62278 15.8153 500]
```

```
calc_val2 = 1×3

10^2 ×
   0.016227800000000   0.158153000000000   5.000000000000000
```

```
abs_err2 = calc_val2 - mat_val
```

```
abs_err2 = 1×3

10^-3 ×
   0.003398316206660  -0.043125576773662  -0.125000062496383
```

```
rel_err2 = abs_err2 ./ mat_val
```

```
rel_err2 = 1×3

10^-5 ×
   0.209413680424876  -0.272681891447036  -0.025000006249272
```

Question 2) (10 Marks)

```
syms x;
f(x) = (1-cos(x))/sin(x);
res = double(f(0.007))
```

```
res =
   0.003500014291737
```

```
calc_valq2 = 0;
abs_errq2 = calc_valq2 - res
```

```
abs_errq2 =
  -0.003500014291737
```

```
rel_errq2 =  abs_errq2 / res
```

```
rel_errq2 =
    -1
```

```
abs_err3 = 0.000611087 - res
```

```
abs_err3 =
  -0.002888927291737
```

```
rel_err3 = abs_err3/res
```

```
rel_err3 =
  -0.825404427221130
```

Question 3) (**14 Marks**)

a)

```
syms x;
f(x) = -(4*exp(-4*x)*cos(6*x) + 6*exp(-4*x)*sin(6*x));
result = (f(0.5))
```

result = $-4\cos(3)\,e^{-2} - 6\,e^{-2}\sin(3)$

```
true_value = double(result)
```

```
true_value =
   0.421332562151359
```

```
f_original(x) = exp(-4*x)*cos(6*x)
```

f_original(x) = $\cos(6\,x)\,e^{-4\,x}$

```
h = 0.01;
derivative = double((f_original(0.5+h) - f_original(0.5))/(h))
```

```
derivative =
   0.438478802436531
```

c)

```
syms x;
h = 0.01;
f(x) = 1 - 4*x - 10*x^(2);
der = ((f(x + h) - f(x))/h)
```

der =

$$1000\,x^2 - 1000\,\left(x + \frac{1}{100}\right)^2 - 4$$

d)

```
% This algorithm will be used to compute the the step size h which results
% in the least error when computing the first order approximation of the
% derivative of f(x)
syms x;
f(x) = exp(-4*x)*cos(6*x);
% Define the "true" value obtained in part a)
true_val = 0.421332562151359;
% Define the value for which we are trying to compute the derivative
x = 0.5;
last_err = 100; % Set error very large for first loop
step = 0.001; % Set starting step size
cont = 1; % Set loop variable

while (cont == 1)
    % For each value of h, compute the first order approximation
    temp = ((f(x + step) - f(x))/step);
    derivative = double(temp);
    err = derivative - true_val;
    if( err >= last_err)
        % Break out of loop, error has started to increase
        cont = 0;
        return_val = [last_err, last_step, last_der];
    end
    last_der = derivative;
    last_step = step;
    last_err = err;

    step = step / 2;
end
titles = {"Computed Error", "Computed Step", "Computed Derivative"};
disp(titles), disp(return_val);
```

```
    ["Computed Error"]    ["Computed Step"]    ["Computed Derivative"]

  -0.000000001082816   0.000000001907349   0.421332561068543
```

Question 4) (12 Marks)

a)

```
A  = [4 -2 -3 6; 6 -7 6.5 -6; 1 7.5 6.25 5.5; -12 22 15.5 -1]
```

```
A = 4×4
   4.000000000000000  -2.000000000000000  -3.000000000000000   6.000000000000000
   6.000000000000000  -7.000000000000000   6.500000000000000  -6.000000000000000
   1.000000000000000   7.500000000000000   6.250000000000000   5.500000000000000
 -12.000000000000000  22.000000000000000  15.500000000000000  -1.000000000000000
```

```
B_res = inv(A)*[12; -6.5; 16; 12]
```

```
B_res = 4×1
  -4.770833333333334
  -4.450757575757576
   3.757575757575756
   5.575757575757576
```

b)

```
% Gaussian elimination for matrix.
% Begin by augmenting the matrix from part a) with its solutions
B = [12; -6.5; 16; 12];
```

```
concat = [A B]
```

```
concat = 4×5
   4.000000000000000  -2.000000000000000  -3.000000000000000   6.000000000000000  12.000000000000000
   6.000000000000000  -7.000000000000000   6.500000000000000  -6.000000000000000  -6.500000000000000
   1.000000000000000   7.500000000000000   6.250000000000000   5.500000000000000  16.000000000000000
 -12.000000000000000  22.000000000000000  15.500000000000000  -1.000000000000000  12.000000000000000
```

```
% This script assumes that the matrix is a 4x4 (augmented)
% Begin iterating at the second row
for row=2:4
    % For all rows
    for j=row:4
        % Compute ratio by which to subtract first row from above row
        subs = (concat(j,row-1)/concat(row-1,row-1))*concat(row-1,:);
        concat(j, :) = concat(j,:) - subs;
    end
end
disp(concat);
```

```
   4.000000000000000  -2.000000000000000  -3.000000000000000   6.000000000000000  12.00000000000000
                   0  -4.000000000000000  11.000000000000000 -15.000000000000000 -24.50000000000000
                   0                   0  29.000000000000000 -26.000000000000000 -36.00000000000000
                   0                   0                   0   2.275862068965516  12.68965517241379
```

```
% Then, clear the rest of the rows by solving for each value of x by
% working our way up the matrix from the bottom row
% Create empty array in which to store answers
answ = zeros(1,4);
for i=1:4
    j = 5-i;
    % Solve for xi in each row
    answ(j) = (concat(j,5)-concat(j,4)*answ(4) - concat(j,3)*answ(3) - concat(j,2)*answ(2))/concat(j,j);
end
for i=1:length(answ)
    disp("x"+i + ":" + char(vpa(answ(i),15))) %, disp(answ(i));
end
```

```
x1:-4.77083333333333
x2:-4.45075757575758
x3:3.75757575757576
x4:5.57575757575758
```

c)

```
% Calculate absolute error between results in a) and b)
% Have to take transpose of answ vector to get correct dimensions.
absolute_err = transpose(answ) - B_res
```

absolute_err = 4×1

10^{-14} ×

```
  -0.444089209850063
  -0.355271367880050
   0.444089209850063
   0.266453525910038
```

Question 5) (5 Marks)

a)

```
syms x;
f = exp(sin(x));
t = taylor(f,x)
```

```
t =
```

$$-\frac{x^5}{15}-\frac{x^4}{8}+\frac{x^2}{2}+x+1$$

```
f2(x)  = 1 + x + x^2/2
```

```
f2(x) =
```

$$\frac{x^2}{2}+x+1$$

b)

```
% Using the approximation in f2, compute f(0.01)
answer1 = f2(0.01);
disp(double(answer1));
```

```
    1.010050000000000
```