**ECSE 443 - Assignment 1**

**\*\*\* USED ROUNDING UP FOR 5-9 NUMBERS AND ROUND DOWN FOR 0-4 NUMBERS.**

**Question 1 – a)** MATLAB values, Refer to **appendix A**, section a

|  |  |  |  |
| --- | --- | --- | --- |
| x = | 10 | 1000 | 1000000 |
| F(x) | 1.6227766016838 | 15.8153431255768 | 500.0001250000625 |

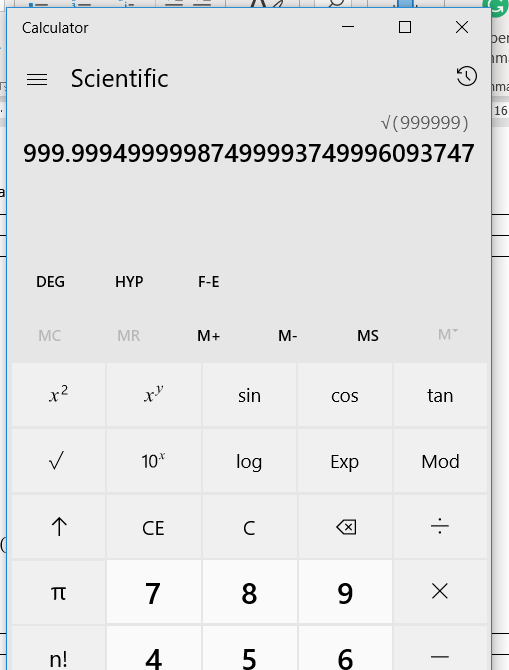
**b)** Calculator values: For these calculations I used my computer calculator which kept the most significant digits when compared to a standard calculator. This is due to the fact I wanted to keep as many digits as possible before rounding down.

For x = 10:

For x = 1000:

For x = 1000000:

I used my computer calculator and calculated 999.9994… which is a round down to 999.999 for 6 significant figures.



|  |  |  |  |
| --- | --- | --- | --- |
| x = | 10 | 1000 | 1000000 |
| F(x) | 1.6228 | 15.8000 | 1000 |

**c)** Matlab Error Results with Calculator results. Refer to **appendix A**, section c.

|  |  |  |  |
| --- | --- | --- | --- |
| x = | 10 | 1000 | 1000000 |
| Absolute Error | 0.0000233983162 | 0.0153431255768 | 499.9998749999375 |

|  |  |  |  |
| --- | --- | --- | --- |
| x = | 10 | 1000 | 1000000 |
| Percent Relative Err (%) | 0.001441869212467 | 0.097014180817616 | 99.999949999987507 |

The error associated with the results calculated with the calculator is from the limited number of significant figures that we can use in order to do the calculations. Therefore, we lose precision throughout the operations.

**d)** Refer to **appendix A**, section d for calculation.

Starting Function:

Simplified:

For X = 10:

For X = 1000:

For X = 1000000:

|  |  |  |  |
| --- | --- | --- | --- |
| x = | 10 | 1000 | 1000000 |
| F(x) | 1.62278 | 15.8153 | 500.000 |

**e)** Matlab Error Results compared with modified function results in d). Refer to **appendix A**, section e.

|  |  |  |  |
| --- | --- | --- | --- |
| x = | 10 | 1000 | 1000000 |
| Absolute Error | 3.398316206660e-6 | 4.3125576773662e-5 | 1.25000062496383e-4 |

|  |  |  |  |
| --- | --- | --- | --- |
| x = | 10 | 1000 | 1000000 |
| Per Relative Err (%) | 2.09413680424876e-4 | 2.72681891447036e-4 | 2.5000006249272e-5 |

The calculated error between the Matlab calculation and the modified function is lower for most inputs due to the elimination of subtraction in the function. When the subtraction operation occurs, it leads to a lost of significant figures when the two operands are close to one another. Therefore, the removal of this operation allows for a more precise result.

**Question 2 – a)** MATLAB values, Refer to **appendix B,** section a

|  |  |
| --- | --- |
| X= | 0.007 |
| F(x) | C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard5720896608659911961\image15487740869940.png |

**b)** Again, I used my computers calculator to perform the calculations due to its extra significant figures therefore when I perform my rounding later it less likely to be affected by initial rounding made by my standard calculator.

|  |  |
| --- | --- |
| X= | 0.007 |
| F(x) | 0.00342860 |

**c)** Refer to **appendix B**, section c.

|  |  |
| --- | --- |
| X= | 0.007 |
| Absolute Error | C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard5720896608659911961\image15487741274990.png |
| Percent Relative Error (%) | C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard5720896608659911961\image15487741348400.png |

The error associated with the results is from the limiting number of significant figure which limits the precision of the result compared to when we use all the significant figures.

**d)** Refer to **appendix B**, section d for calculation.

Original Function:

Simplified Function:

For X = 0.007:

|  |  |
| --- | --- |
| X= | 0.007 |
| F(x) | 0.00350001 |

**e)** Matlab Error Results compared with modified function results in d). Refer to **appendix B**, section e.

|  |  |
| --- | --- |
| X= | 0.007 |
| Absolute Error | C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard5720896608659911961\image15487763695540.png |
| Percent Relative Error (%) | C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard5720896608659911961\image15487763814660.png |

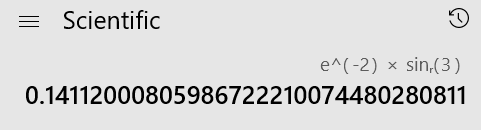
Similarly, to above in Q1, section e, the absolute and percent relative error are both much smaller due to the removal of the subtraction operation. The subtraction operation results in a loss of significant figures when the operations are similar.

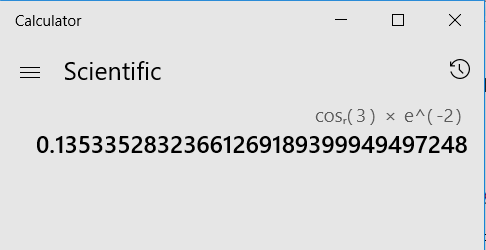
**Question 3 – a)**

Initial Function:

Derivative Function:

Solve for X = 0.5





0.11459109756681117859545744515281

Final Answer:

**b)** Refer to **appendix C**, section b.

Output from Matlab: output = C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard5720896608659911961\image15488133420010.png

**c)** The manual calculation is shown below, however it is also done in Matlab.

Taylor Series of :

Taylor Series of :

Multiply the first 3 terms:

Take the derivative:

The work is also done in Matlab to compare answers in **appendix C**, section c. Final result is:

out\_c = C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard8981640865152899635\image15489003857510.png

**d)** Refer to **appendix C**, section d for the program to determine the minimum step size of a function. The final output for the minimum step size is:

h = 1.000000000000000e-9

**Question 4 – a)** Refer to **appendix D**, section a for the script and work done. The final answer is illustrated bellow.

|  |  |
| --- | --- |
| X1 | -4.770833333333334 |
| X2 | -4.450757575757576 |
| X3 | 3.757575757575756 |
| X4 | 5.575757575757576 |

x\_a = 4×1

-4.770833333333334

-4.450757575757576

3.757575757575756

5.575757575757576

**b)** Refer to **appendix D**, section b and the function **gaussElim, in appendix E** for the implementation of the function.

|  |  |
| --- | --- |
| X1 | -4.770833333333338 |
| X2 | -4.450757575757579 |
| X3 | 3.757575757575760 |
| X4 | 5.575757575757579 |

x\_b = 4×1

-4.770833333333338

-4.450757575757579

3.757575757575760

5.575757575757579

**c)** Refer to **appendix** **D**, section c. The absolute error for each entry was calculated as the following.

abs\_error = 4×1

10-14 ×

0.444089209850063

0.266453525910038

0.444089209850063

0.266453525910038

The maximum absolute error is:

max\_abs =

4.440892098500626e-15

**Question 5 – a)** Refer to **appendix F**, section a to the steps in order to calculate the following Taylor Series.

f\_t(x) =   C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard5720896608659911961\image15487938496270.png

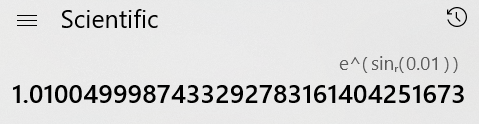
**b)** Refer to **appendix F**, section b.

f\_b =   1.0100499999999998923527755323448218405246734619140625000000000000

**c)**

Starting Function:

Solve x = 0.01:



**d)** The results are relatively similar however, the result for c using the regular function will be more precise due to the fact that it is doing less approximation. In the case of the Taylor Series, we are approximating the value using the first 3 terms which leaves x4 error in the answer.

**Appendix**

**Appendix A – Question 1 Matlab Code**

Q1 - a)

format long

syms f(x)

input = [10, 1000, 1000000];

out\_mat = zeros(size(input));

f(x) = x\*(sqrt(x) - sqrt(x-1))

f(x) = C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard2264924943845852324\image15492119356430.png

for i = 1:length(input)

x = input(i);

out\_mat(i) = f(x);

end

out\_mat

out\_mat = 1×3

102 ×

0.016227766016838 0.158153431255768 5.000001250000625

c)

% values from calculator

out\_calc = [1.6228, 15.8, 1000];

% absolute error

abs\_error = abs(out\_calc - out\_mat);

abs\_error

abs\_error = 1×3

102 ×

0.000000233983162 0.000153431255768 4.999998749999375

rel\_error = zeros(size(input));

% relative error

for i=1:length(input)

rel\_error(i) = abs\_error(i)/out\_mat(i);

end

% Percent of Relative error

per\_relative\_err = rel\_error\*100

per\_relative\_err = 1×3

0.001441869212467 0.097014180817616 99.999949999987507

d) Calculation done in word with a calculator

e)

out\_mat\_new = [1.62278, 15.8153, 500];

% absolute error

abs\_error = out\_mat\_new - out\_mat;

abs\_error

abs\_error = 1×3

10-3 ×

0.003398316206660 -0.043125576773662 -0.125000062496383

rel\_error = zeros(size(input));

% relative error

for i=1:length(input)

rel\_error(i) = abs\_error(i)/out\_mat(i);

end

% Percent of Relative error

per\_relative\_err = rel\_error\*100

per\_relative\_err = 1×3

10-3 ×

0.209413680424876 -0.272681891447036 -0.025000006249272

**Appendix B - Question 2 Matlab Code**

Question 2

a)

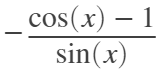
format long

syms f(x)

input = 0.007;

f(x) = (1-cos(x))/sin(x)

f(x) =



x = input;

out\_mat = vpa(f(input))

out\_mat = 0.003500014291736696180563008388486

b) Please see refer to written paper

F(x) = 0.00342860

c)

out\_calc = 0.00342860;

% absolute error

abs\_error = out\_mat - out\_calc

abs\_error = 0.000071414291736696216917261329836799

% relative error

rel\_error = abs\_error/out\_mat;

per\_relative\_err = rel\_error\*100

per\_relative\_err = 2.040400003660004142025675973315

d) Calculated using a calculator and demonstrated in word

e)

out\_mat\_new = 0.00350001;

% absolute error

abs\_error = abs(out\_mat\_new - out\_mat);

abs\_error

abs\_error = 0.0000000042917366961670926724307090239394

% relative error

rel\_error = abs\_error/out\_mat;

% Percent of Relative error

per\_relative\_err = rel\_error\*100

per\_relative\_err = 0.00012262054775889347305826475631838

**Appendix C - Question 3 Matlab Code**

Question 3

a) refer to paper

b)

format long

syms p(x)

h = 0.01;

x\_pt = 0.5;

p(x) = exp(-4\*x)\*cos(6\*x)

p(x) = C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard6065402713333656964\image15490742660720.png

% p(x+h)

pin\_xh = p(x\_pt+h);

%p(x)

pin\_x = p(x\_pt);

p\_out = vpa((pin\_xh - pin\_x)/h)

p\_out = 0.43847880243653103711684144164038

c)

syms f(x) f\_t(x) k(x) k\_t(x) g(x)

f(x) = exp(-4\*x);

% use order 3 to get first 3 terms

f\_t(x) = taylor(f, x, 'Order', 3)

f\_t(x) = C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard6065402713333656964\image15490742660891.png

% use order 5 to get the first 3 terms

k(x) = cos(6\*x);

k\_t(x) = taylor(k, x, 'Order', 5)

k\_t(x) = C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard6065402713333656964\image15490742660992.png

g(x) = k\_t(x)\*f\_t(x);

expand(g(x))

ans = C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard6065402713333656964\image15490742661153.png

g(x) = 1-4\*x-10\*x^2;

% g(x+h)

g\_xh = g(x\_pt+h);

% g(x)

g\_x = g(x\_pt);

out\_c = vpa((g\_xh-g\_x)/h)

out\_c = C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard6065402713333656964\image15490742661154.png

d) True value = 0.42133256215135927525016357669181 from a

Program to calculate the minimum step size DQA Dac DAQ GEDAAZx11

% true calue from part a

true\_val = 0.42133256215135927525016357669181;

% starting step count

h = 0.01;

x = 0.5;

% find the initial value

min\_abs\_err = 1;

while(min\_abs\_err >= 0)

p\_out = (p(x+h)-p(x))/h;

abs\_error = vpa(abs(p\_out-true\_val));

if abs\_error >= min\_abs\_err

% want to keep the minimum step size

h\_min = h\*10;

break;

end

% set new minimum

min\_abs\_err = abs\_error;

h = h/10;

end

h\_min

h\_min =

1.000000000000000e-09

**Appendix D - Question 4 Matlab Code**

Question 4

a)

format long

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

B = [12; -6.5; 16; 12];

x\_a = inv(A)\*B

x\_a = 4×1

-4.770833333333334  
 -4.450757575757576  
 3.757575757575756  
 5.575757575757576

b)

X = [A B];

[n,m] = size(X);

X = gaussElim(X);

% get the last column

x\_b = X(:,m)

x\_b = 4×1

-4.770833333333338  
 -4.450757575757579  
 3.757575757575760  
 5.575757575757579

c)

% absolute error

abs\_error = abs(x\_b - x\_a)

abs\_error = 4×1

10-14 ×

0.444089209850063  
 0.266453525910038  
 0.444089209850063  
 0.266453525910038

max\_abs = max(abs\_error(:))

max\_abs =

4.440892098500626e-15

**Appendix E - Question 4 Gaussian Elimination function**

% function to calculate Gaussian Elimination

function X = gaussElim(X)

% get the size of the array

[n, m] = size(X);

for i=1:n

p = i;

for k=i:n

if(abs(X(p,i)) >= abs(X(k,i)))

p = i;

end

end

if(p ~= i)

temp = X(p);

X(p) = X(i);

X(i) = temp;

end

% check if the diagonal is 0

if X(i,i) == 0

return

end

j = i;

temp = X(i,j);

X(i,:) = X(i,:)/temp;

% loop through the matrix to get the triangular form

for k=1:n

if k ~= i

X(k,:) = X(k,:)-X(i,:)\*X(k,j);

end

end

end

**Appendix F - Question 5 Matlab Code**

Question 5

a)

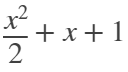
format long

syms f(x) f\_t(x)

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

f\_t(x) = taylor(f, x, 'Order', 4)

f\_t(x) =



f\_a = vpa(f\_t(0))

f\_a = 1.0

b)

f\_b = vpa(f\_t(0.01))

f\_b = 1.01005

fprintf('%.64f', f\_b);

1.0100499999999998923527755323448218405246734619140625000000000000

c) Calculation done in word and f\_c =

d) Comparison completed in the word document.