**ECSE 443 - Assignment 2**

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

1. Function 1 breakdown.

Simply construct the A and B matrix to solve for , and . Using the **Normal Method,** the constants are found to be:

x1 = 1×3

0.2509 -0.1113 0.0097

1. Function 2 breakdown. Use logarithm in order to simplify the function into a linear function we can handle.

where: ,, ,

Calculate the adjusted constants.

1. Function 3 breakdown. Take the inverse of the function in order to calculate the coefficients.

Where: , , ,

|  |  |  |
| --- | --- | --- |
| **Desired Function** | **Intermediate Function** | **Calculated Function** |
|  | N/A |  |
|  |  |  |
|  |  |  |

syx =

25.412172419022365

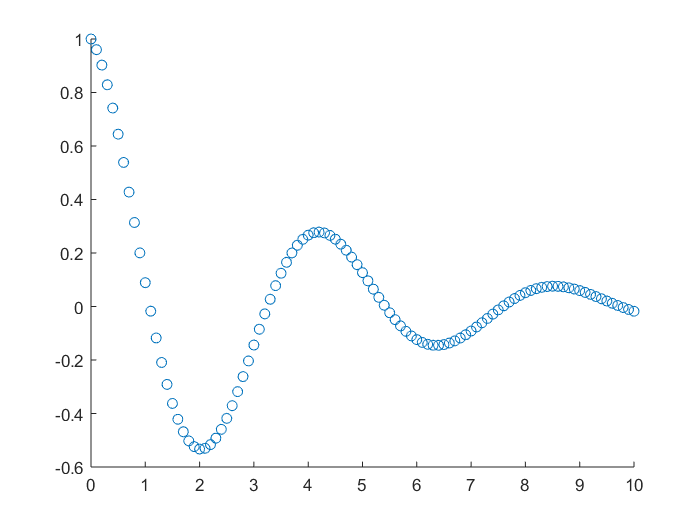
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Normal Equations Coefficients & Error** | | | | |
|  | **a0** | **a1** | **a2** | **Standard Error** |
| **Function 1** | 0.2509 | -0.1113 | 0.009651 | 0.25412172419022365 |
| **Function 2** |  |  |  | 76.753837532477405 |
| **Function 3** |  |  |  |  |
|  |  |  |  |  |

**Question 2 –** Refer to **Appendix B**, for the function to calculate the roots using the bisectional method. My bisectional method iterates through the function’s domain of [-100,100] and an interval of 1. The function will return all the roots within the domain accurate to the nearest 10-8.

The function above has the following root:

roots = 0.667968750000000

**Question 3 –** Refer to **Appendix C**, for the function to calculate the roots using the secant method. The points given represent a harmonic oscillating function, as seen below.



Similarly, to the bisectional method, I look for the places in between points where there is a possible root by checking if the sign changes. Then I go through one iteration using the following function to find the root. If we were given the function, we would be able to get a more precise root.

Using the above function, we calculate the roots from one iteration and we find the following roots:

roots = 1×5

1.1000 3.3000 5.5000 7.6000 9.8000

With the roots that we found, the operating frequency can be found by looking at the difference between the roots to find the half period, T. The avg half period, T, was found then to get the full period it was multiplied by 2 to give the following:

T = 4.350000000000001 s

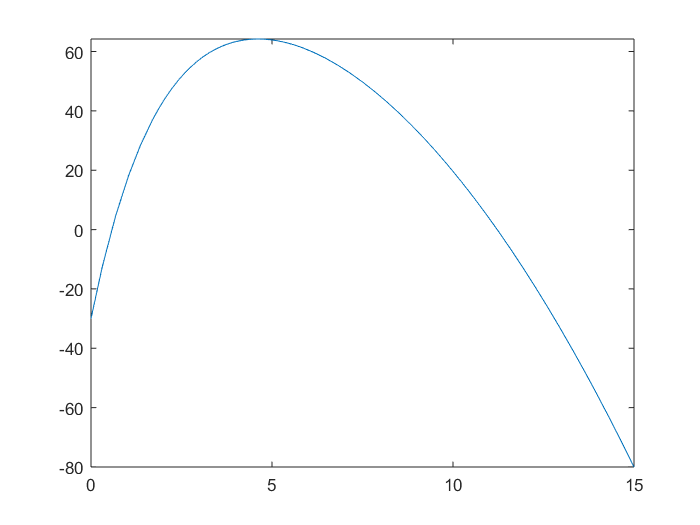
In order, determine the operating frequency, we will use , which returns the following as the operating frequency.

**f = 0.229885057471264 Hz**

**Question 4 –** Refer to **Appendix D**, for the function to compute the roots using the Newton method. The remaining energy function if found from taking the generated energy minus the dissipated energy.

f(w) = C:\Users\bjay2\AppData\Local\Temp\ConnectorClipboard4514267539207464331\image15507793580240.png

The plot of the function is below:



The roots are which means the operating frequency can be either one of the following frequencies:

roots = 1×2

0.573322845282703 kHz 11.221433847446772 kHz

|  |  |
| --- | --- |
| ω= | 573.322845282703 Hz |
|  | 11221.433847446772 Hz |

**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)

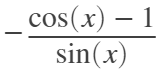
format long

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)

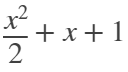
format long

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.