Bryan Jay
260738764

ECSE 443 - Assignment 2

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

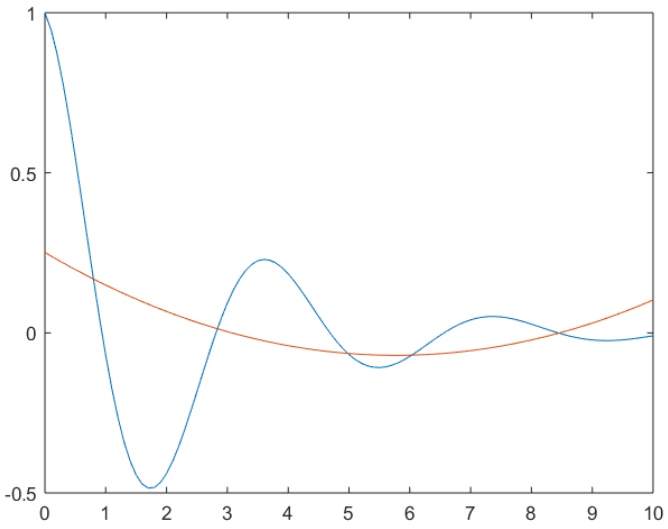
1- Function 1 breakdown.

$$f_1(t) = a_0 + a_1 t + a_2 t^2$$

Simply construct the A and B matrix to solve for a_0, a_1 and a_2. Using the **Normal Method,** the constants are found to be:

```
x1 = 0.250922262575244 -0.111345919449175   0.009651643345421
```

The fit of this polynomial with the data points is below.



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

$$f_2(t) = a_0 t^{a_1}$$

$$h_2(v) = \ln\big(f_2(t)\big) = \ln(a_0 t^{a_1})$$

$$h_2(v) = \ln(a_0) + a_1 \ln(t)$$

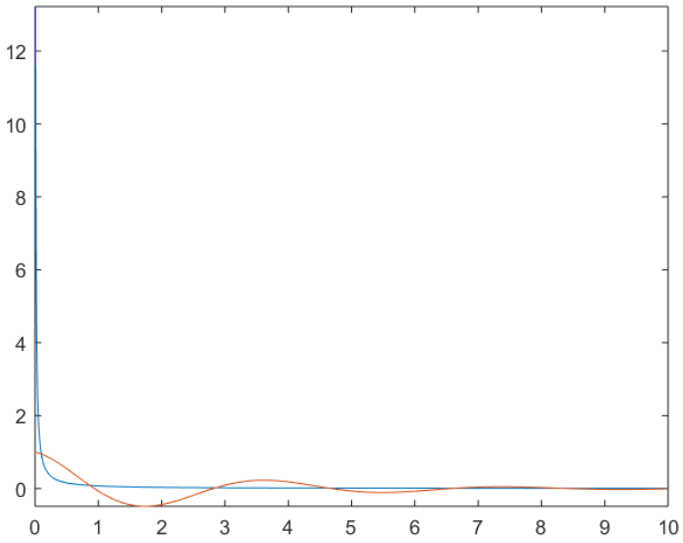
$$h_2(v) = c_0 + c_1 \mathrm{v}$$

where: $c_0 = \ln(a_0)$, $c_1 = a_1$, $h_2(v) = \ln\big(f_2(t)\big)$, $v = \ln(t)$

Calculate the adjusted constants.

```
x2 = 0.072688765244002 -1.109434267418068
```

Fitting this polynomial onto the data points is the following.



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

$$f_3(t) = \frac{1}{a_0 t + a_1}$$

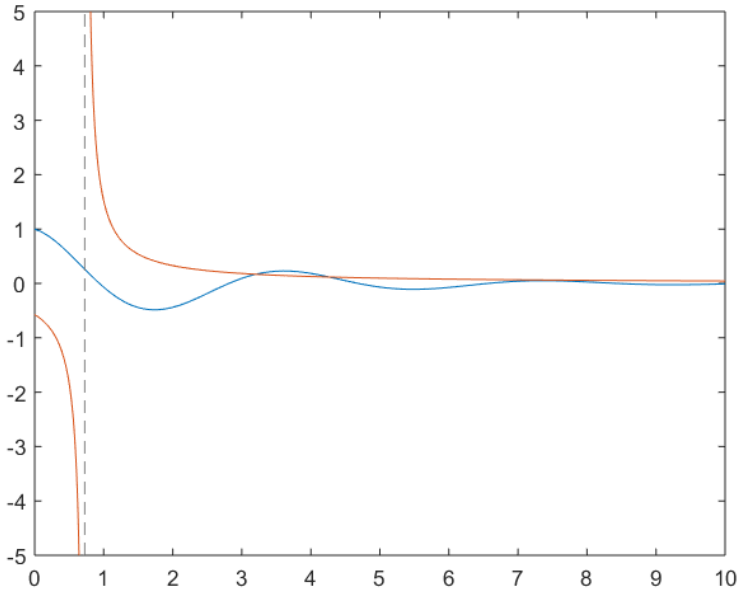
$$g_3(v) = \left(f_3(t)\right)^{-1} = a_0 t + a_1$$

$$g_3(v) = c_0 v + c_1$$

Where: $g_3(v) = \left(f_3(t)\right)^{-1}$, $v = t$, $c_0 = a_0$, $c_1 = a_1$

```
x3 = 2.403590506890200 -1.744846214839930
```

Fitting this polynomial onto the data points is the following.



Desired Function	Intermediate Function	Calculated Function
$f_1(t) = a_0 + a_1 t + a_2 t^2$	N/A	$f_1(t) = 0.2509 + -0.1113t + 0.0097t^2$
$f_2(t) = a_0 t^{a_1}$	$h_2(t) = c_0 + c_1 \ln(t)$	$f_2(t) = 0.07269t^{-1.109}$
$f_3(t) = \frac{1}{a_0 t + a_1}$	$g_3(v) = c_0 v + c_1$	$f_3(t) = \frac{1}{-1.7448t + 2.4036}$

Normal Equations Coefficients & Error					
	a_0	**a_1**	**a_2**	**r^2**	**Standard Error**
Function 1	0.2509	-0.1113	0.009651	0.103730806718611	0.25412172419022365
Function 2	0.07269	-1.1094	N/A	-	1.574523966614549
Function 3	-1.7448	2.4036	N/A	0.988676544514084	8.185803381515331

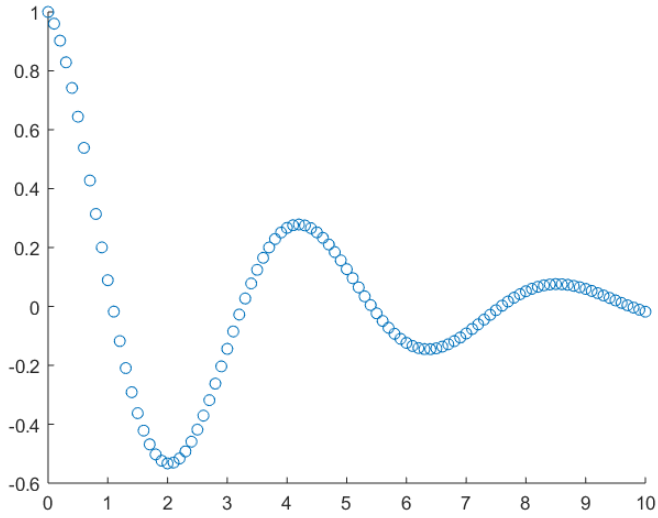
Question 2 – Refer to **Appendix B**, for the function to calculate the roots using the bisectional method. My bisectional method iterates through the function's domain of [-100,100] and an interval of 1. The function will return all the roots within the domain accurate to the nearest 10^{-8}.

$$f(x) = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$$

The function above has the following root:

```
roots = 0.666666664183140
```

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.

$$x_{i+1} = x_i - \frac{f(x_i) * (x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

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

```
roots = 5×1
   1.083632275955740
   3.250500419428178
   5.416695317223444
   7.583141867942982
   9.750001258406071
```

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 = 4.333184491225166 s
```

In order, determine the operating frequency, we will use $f = 1/T$, which returns the following as the operating frequency.

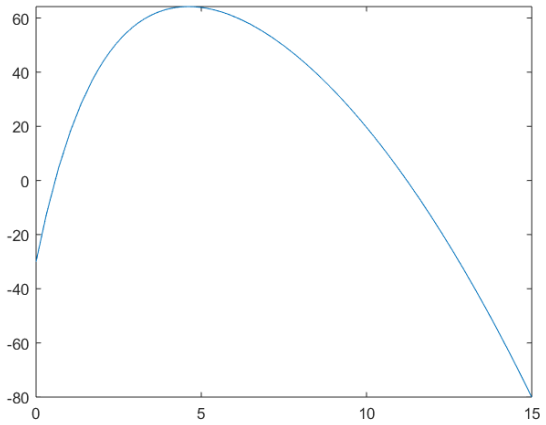
f = 0.230777157544303 Hz

<u>Question 4</u> – 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.

$$f(w) = P(w) = E(w)$$

$$f(w) = 5\,w - 100\,e^{-\frac{14\,w}{25}} - w^2 + 70$$

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 - Function 1 using the normal equations method

```
format long
% initialize variables for all the functions
syms f1(t) f2(t) f3(t)
% import the data
data = importdata('Ass_2_Q1_data.txt');
lenData = length(data);
A1 = ones(lenData, 3);
B1 = zeros(lenData, 1);
% create the desired matrices
for i=1:lenData
    A1(i,2) = data(i,1);
    A1(i,3) = data(i,1)^2;
    B1(i) = data(i,2);
end
% find the normal matrices on each side of the eqn
A_T1 = transpose(A1);
sqr1 = A_T1*A1;
ATB1 = A_T1*B1;
% calculate the upper and lower triangle using cholosky
L1 = chol(sqr1, 'lower');
L_T1 = chol(sqr1, 'upper');
z1 = inv(L1)*ATB1;
x1 = inv(L_T1)*z1;
x1 = x1
```

```
x1 = 3×1
   0.250922262575244
  -0.111345919449174
   0.009651643345421
```

```
% compare with the using the equation
x1 = inv(sqr1)*ATB1
```

```
x1 = 3×1
   0.250922262575244
  -0.111345919449175
   0.009651643345421
```

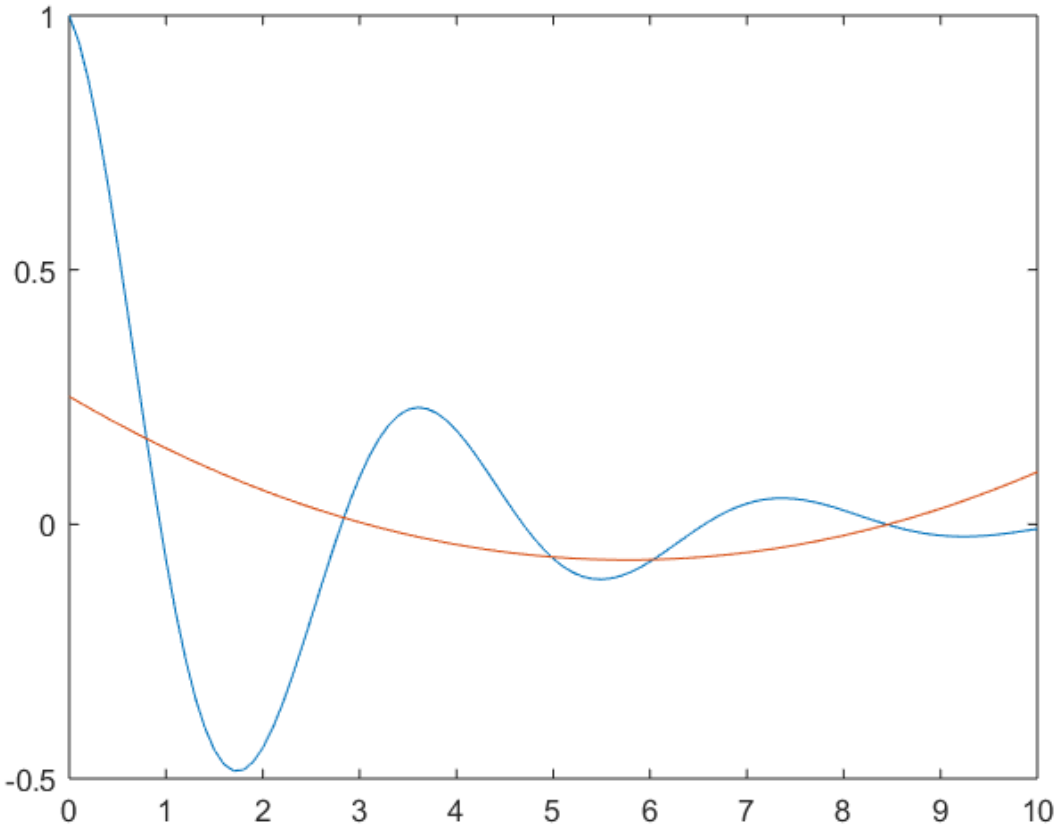
```
% show the final function
f1(t) = x1(1) + x1(2)*t + x1(3)*t^2;
% calculate the Standard Error
Sr1 = sum((B1 - A1*x1).^2);
r1 = 1 - Sr1/sum((B1 - mean(B1)).^2)
```

```
r1 =
   0.103730806718611
```

```
syx1 = sqrt(Sr1/(lenData - length(x1)))
```

```
syx1 =
   0.254121724190224
```

```
% function plotted against the points
plot(data(:,1), data(:,2))
hold on
fplot(f1(t), [0,10])
hold off
```



Function 2 -

```
A2 = ones(lenData-1, 2);
B2 = zeros(lenData-1, 1);
% create the desired matrices by taking the natural logarithm of the points
for i=2:lenData
    A2(i-1, 2) = log(data(i,1));
    B2(i-1) = log(data(i,2));
end
% calculate using normal equations method
A_T2 = transpose(A2);
sqr2 = A_T2*A2;
ATB2 = A_T2*B2;
% calculate the upper and lower triangle using cholosky
%L2 = chol(sqr2, 'lower');
%L_T2 = chol(sqr2, 'upper');
%z2 = inv(L2)*ATB2;
```

```
%x2 = inv(L_T2)*z2;
%x_comp = x2
%x2 = real(transpose(x2))
% compare with the using the equation
x2 = inv(sqr2)*ATB2;
x2 = [exp(x2(1)); x2(2)];
x_comp = x2
```

```
x_comp = 2×1 complex
  0.072688765244002 + 0.294592186754100i
 -1.109434267418068 + 0.275376751272873i
```

```
x2 = real(transpose(x2))
```

```
x2 = 1×2
   0.072688765244002  -1.109434267418068
```

```
% final function
f2(t) = x2(1)*t^(x2(2))
```

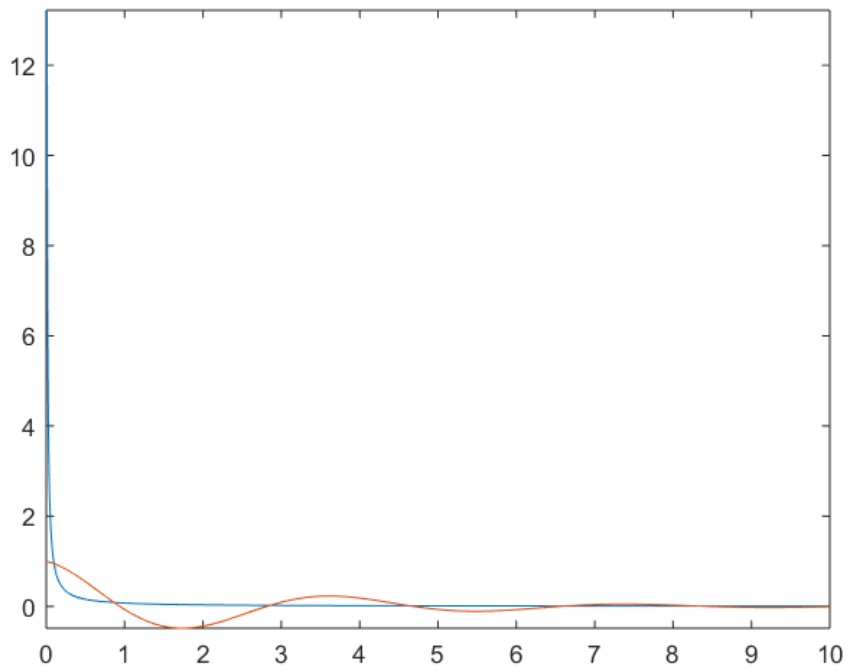
```
f2(t) =
```

$$\frac{5237777537070565}{72057594037927936\, t^{624555969167009/562949953421312}}$$

```
% calculate the Standard Error
Sr2 = abs(real(sum((data(:,2)- A2*x_comp).^2)));
%r2 = 1 - Sr2/sum((B2 - mean(B2)).^2)
syx = sqrt(Sr2/(lenData-1 - length(x_comp)))
```

```
syx =
   1.023766013230174
```

```
fplot(f2(t), [0,10])
hold on
plot(data(:,1), data(:,2))
hold off
```



Function 3

```
A3 = ones(lenData, 2);
B3 = zeros(lenData, 1);
% calculate the desired matrices by taking the inverse of the y-coords
for i=1:lenData
    A3(i, 1) = 1;
    A3(i, 2) = data(i,1);

    B3(i) = 1/data(i,2);
end
% calculate using normal equations method
A_T3 = transpose(A3);
sqr3 = A_T3*A3;
ATB3 = A_T3*B3;
% calculate the upper and lower triangle using cholosky
%L3 = chol(sqr3, 'lower');
%L_T3 = chol(sqr3, 'upper');
%z3 = inv(L3)*ATB3;
%x3 = inv(L_T3)*z3;
%x3 = transpose(x3)
% compare with the using the equation
x3 = inv(sqr3)*ATB3
```

```
x3 = 2×1
   2.403590506890200
  -1.744846214839930
```

```
f3(t) = ((x3(2) + x3(1).*t)).^(-1)
```

```
f3(t) =
```

$$\frac{1}{\frac{676550581948873\,t}{281474976710656} - \frac{7858088762971929}{4503599627370496}}$$

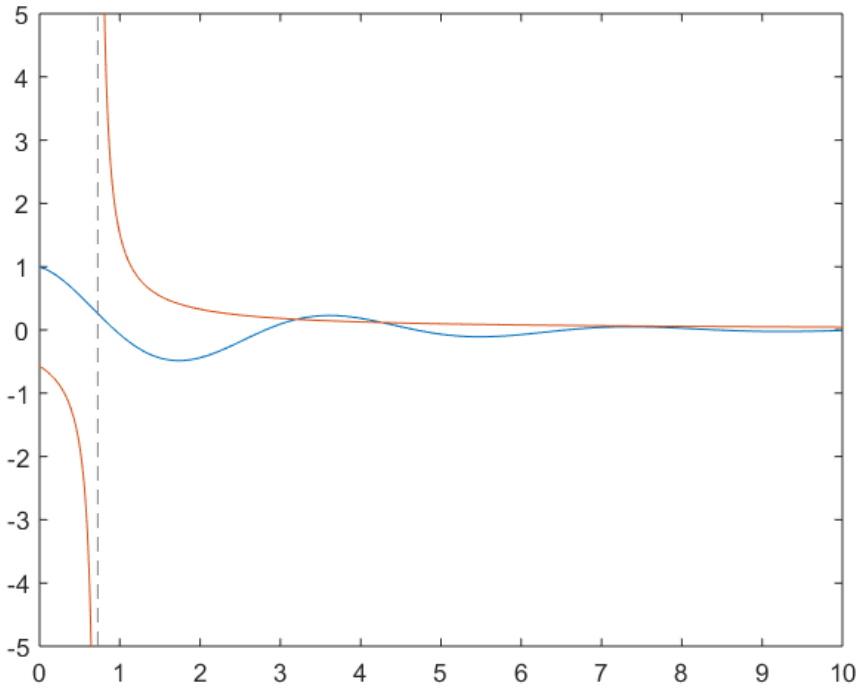
```
% calculate the Standard Error and regression
Sr3 = sum((data(:,2)- A3*x3).^2);
r3 = 1 - Sr3/sum((data(:,2) - mean(data(:,2))).^2)
```

```
r3 =
   0.988676544514084
```

```
syx = sqrt(Sr3/(lenData - length(x3)))
```

```
syx =
   8.185803381515331
```

```
%y = ((x3(1) + x3(2).*data(:,1))).^(-1);
%plot(data(:,1), data(:,2), 'r-*', data(:,1), y, 'k');
plot(data(:,1), data(:,2))
hold on
fplot(f3(t), [0,10])
hold off
```



Appendix B - Question 2 Matlab Code

Q2 - my bisectional method will scan the function at intervals of 1 of a domain [-50, 50] for all the possible roots

```
format long
syms f(x)
f(x) = x^3 - 2*x^2 + 4*x/3 -8/27
```

```
f(x) =
```

$$x^3 - 2\,x^2 + \frac{4\,x}{3} - \frac{8}{27}$$

```
threshold = 10^-8;
roots = [];
for i=-100:100
    lower = i;
    higher = i+1;

    if(f(lower)*f(higher)) < 0
        middle = (lower+higher)/2;
        while abs(f(middle)) > threshold
            if f(higher)*f(middle) < 0
                lower = middle;
            else
                higher = middle;
            end
            middle = (lower+higher)/2;
        end
```

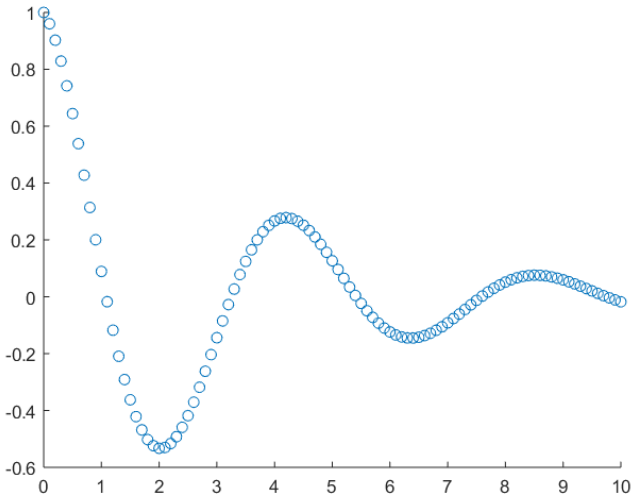
```
        roots = [roots, middle];
    end
end
roots
```

```
roots =
   0.667968750000000
```

Appendix C - Question 3 Matlab Code

Q3

```
data = importdata('Ass_2_Q3_data.txt');
scatter(data(:,1), data(:,2))
```




```
length(data)
```

```
ans =
   101
```

```
roots = [];
x = data(:,1);
f = data(:,2);
for i=2:length(data)
    % we know there is a root if there is a sign change in between points
    if f(i)*f(i-1) < 0
        % run a single iteration of the secant method to find a more
        % precise point
        x_i1 = x(i) - (f(i)*(x(i)-x(i-1))/(f(i)-f(i-1)));
        roots = [roots, x_i1];
    end
end
roots'
```

```
ans = 5×1
   1.083632275955740
   3.250500419428178
   5.416695317223444
```

```
   7.583141867942982
   9.750001258406071
```

```
T_half = 0;
% calculate the avg half period by using all roots
for i=2:length(roots)
    T_half = T_half + roots(i)-roots(i-1);
end
T_half = T_half/4;
% get the full period
T = T_half*2
```

```
T =
   4.333184491225166
```

```
% find the frequency
f = 1/T
```

```
f =
   0.230777157544303
```

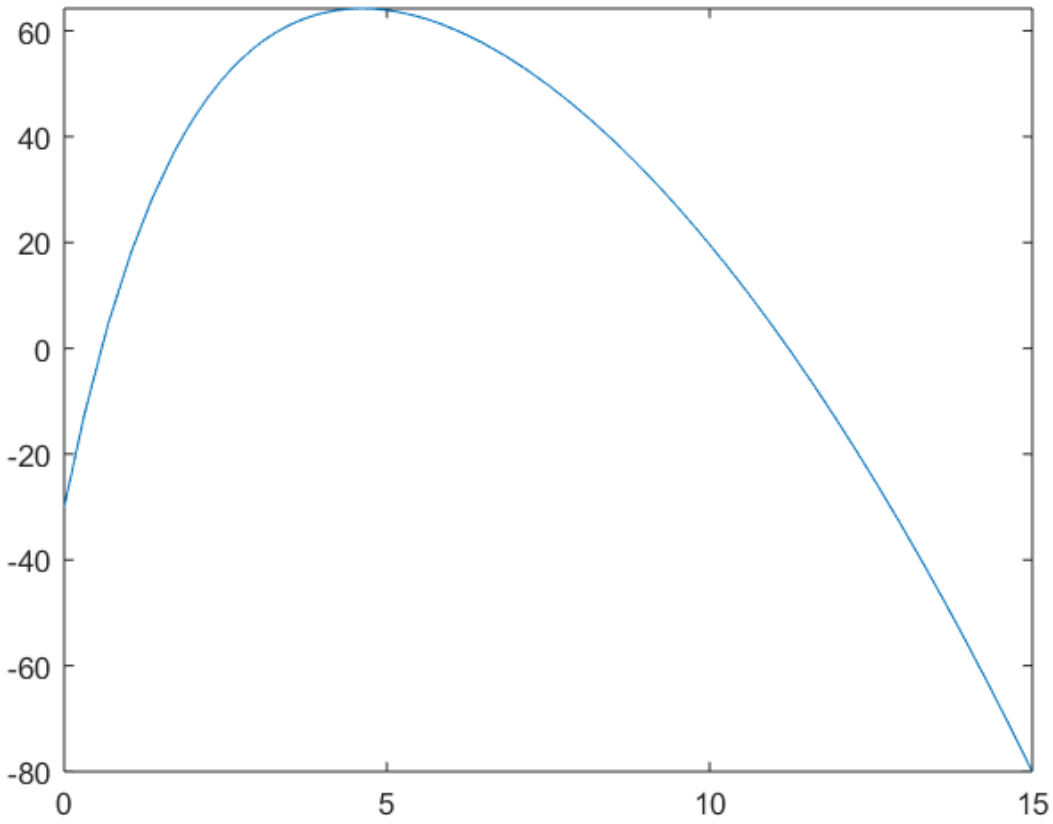
Appendix D - Question 4 Matlab Code

```
format long
syms P(w) E(w) f(w) f_d(w)
% accuracy value for the function
threshold = 10^-8;
% generator function and its derrivative
P(w) = 100*(1-exp(-0.56*w));
% energy dissipated function
E(w) = w^2 - 5*w + 30;
% starting points and roots
roots = [];
x_k = [0, 8];
% the total energy function
f(w) = P-E
```

```
f(w) =
```

$$5\,w - 100\,\mathrm{e}^{-\frac{14\,w}{25}} - w^2 + 70$$

```
f_d(w) = diff(f);
fplot(f(w), [0,15])
```




```
% find both possible roots for the functions
for i=1:length(x_k)
    % loop through the function using newtons method to get roots, stop
    % when its less than the threshold
    while abs(f(x_k(i))) > threshold
        x_k1 = x_k(i) - f(x_k(i))/f_d(x_k(i));
        x_k(i) = x_k1;
    end
    roots = [roots, x_k1];
end
roots = double(roots)
```

```
roots = 1×2
   0.573322845282703  11.221433847446772
```

```
% in Hz
vpa(roots*1000)
```

```
ans = (573.32284528270270129723940044641  11221.433847446771324030123651028)
```