PAPER WORK

1) $x_{n+2} = \frac{15}{4.1} x_{n+1} - \frac{14}{16.81} x_n \Rightarrow \frac{1}{4^{n+2}} = \left(\frac{15}{4.1}\right)\left(\frac{1}{4.1^{n+1}}\right) - \frac{14}{16.81} x_n$

$\frac{1}{4^{n+2}} = \frac{15}{4.1^{n+2}} - \frac{14}{16.81} x_n$

$\frac{\cancel{14}}{16{,}81} x_n = \frac{15}{4{,}1^{n+2}} - \frac{1}{4{,}1^{n+2}} = \frac{\cancel{14}}{4{,}1^{n+2}}$

$x_n = \cancel{16{,}81} \frac{1}{4{,}1^{\cancel{n+2}}}$, $16{,}81 = 4{,}1^2$

$\therefore x_n = \frac{1}{4{,}1^n}$

$\lim_{n \to \infty} x_n = \frac{1}{\infty} = \emptyset$

2) $q(t) = q_0 e^{-\frac{Rt}{2L}} \cos\left(\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2}\, t\right)$

$\frac{R}{2L} = a, \sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} = b$

$q(t) = q_0 e^{-at} \cos(bt)$

$\dot{q}(t) = -[q_0 a e^{-at} \cos(bt) + q_0 b e^{-at} \sin(bt)] = -q_0 e^{-at}(a \cos(bt) + b \sin(bt))$

$\ddot{q}(t) = q_0 a e^{-at}(a \cos(bt) + b \sin(bt)) - q_0 e^{-at}(-ab \sin(bt) + b^2 \cos(bt))$

$= q_0 e^{-at} \{[a^2 \cos(bt) + ba \sin(bt)] - [ab \sin(bt) + b^2 \cos(bt)]\}$

$\ddot{q}(t) = q_0 e^{-at} \{(a^2 - b^2) \cos(bt) + 2ab \sin(bt)\}$

$L\ddot{q} + R\dot{q} + \frac{q}{C} = \varnothing$

ICs: $q(\varnothing) = q_0 = V_0 C$
$\dot{q}(\varnothing) = -\frac{R}{2L} q_0$

$q(\varnothing) = q_0 \cancelto{1}{e^{\varnothing}} \cancelto{1}{\cos(\varnothing)} = q_0$ ✓

$\dot{q}(\varnothing) = -q_0 \cancelto{1}{e^{\varnothing}} (a \cancelto{1}{\cos(\varnothing)} + b \cancelto{\varnothing}{\sin(\varnothing)}) = -q_0 a = -\frac{qR}{2L}$ ✓

$\ddot{q}(\varnothing) = q_0 \cancelto{1}{e^{\varnothing}} \{(a^2 - b^2) \cancelto{1}{\cos(\varnothing)} + 2ab \cancelto{\varnothing}{\sin(\varnothing)}\} = q_0(a^2 - b^2)$
$= q_0\left(\frac{R^2}{4L^2} - \frac{1}{LC} - \frac{R}{2L}\right)$

$$\cancel{L} q_0 \left(\frac{R^2}{4L^2} - \frac{1}{\cancel{L}C} - \frac{R}{2\cancel{L}}\right) + R\left(-\frac{q_0 R}{2L}\right) + \frac{q_0}{C} = \phi$$

$$\emptyset = q_0 \left\{\left(\frac{R^2}{4L} - \frac{1}{\cancel{C}} - \frac{R}{2} - \frac{R^2}{2L} + \frac{1}{\cancel{C}}\right) = \left(\frac{R^2}{4L} - \frac{R}{2} - \frac{R^2}{2L}\right.\right.$$

$$\phi = q_0 R\left\{\frac{R}{4L} - \frac{R}{2L} - \frac{1}{2}\right\} = \frac{q_0 R}{2}\left\{\frac{1}{2}\frac{R}{L} - \frac{R}{L} - 1\right\}$$

$$\frac{R}{2L} - \frac{R}{L} = 1$$

$$\frac{R}{2} - R = L$$

$$-\frac{R}{2} = L$$

$R = -2L = -10\,H$ ← this doesn't make sense unit-wise

$$f(R) = e^{-\frac{Rt}{2L}} \cos\left(\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2}\, t\right) - \frac{q}{q_0}$$

$L = 5H$, $C = 10^{-4} F$, $\frac{q}{q_0} = 0{,}01$, $t = 0{,}05$

$a = \frac{t}{2L}$ $\quad b = \frac{1}{4L^2}$ $\quad d = \frac{1}{LC}$

$$f(R) = e^{-aR} \cos\left(\sqrt{d - bR^2}\, t\right) - \frac{q}{q_0}$$

$$f'(R) = -a e^{-aR} \cos\left(\sqrt{d - bR^2}\, t\right) + \frac{bRt}{\sqrt{d - bR^2}} e^{-aR} \sin\left(\sqrt{d - bR^2}\, t\right)$$

$$\left\{ \frac{d}{dR} \cos\left(\sqrt{d - bR^2}\, t\right) = \cancel{-}\sin\left(\sqrt{d - bR^2}\, t\right)\left(\frac{\cancel{1}}{\cancel{2}}t\,(d - bR^2)^{-\frac{1}{2}}\right)\left(\cancel{-}\cancel{2}bR\right)\right.$$

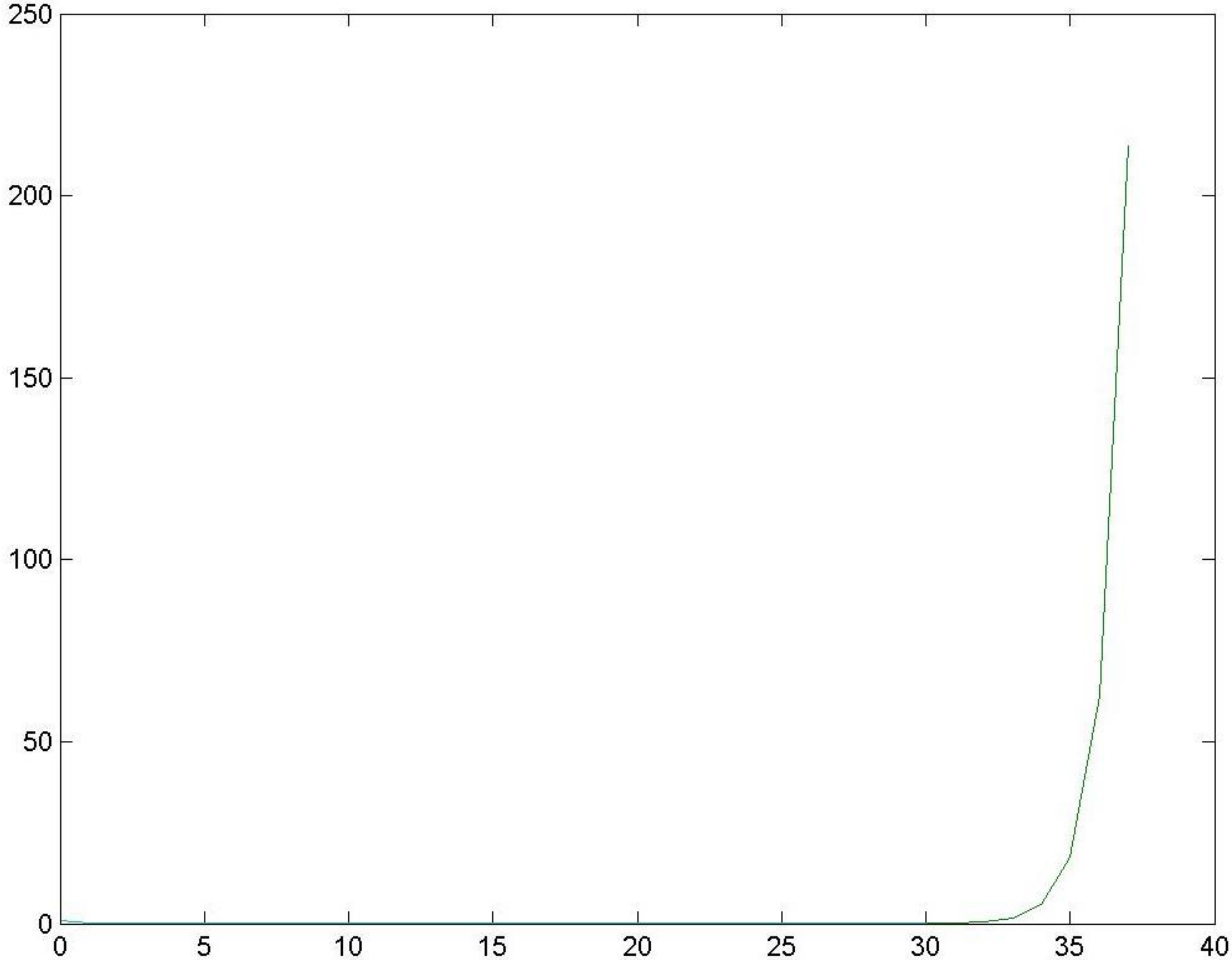
$$\left. = \frac{bRt}{\sqrt{d - bR^2}} \sin\left(\sqrt{d - bR^2}\, t\right)\right\}$$

Matlab Driver Trace

```
-------------------------PROBLEM #1-------------------------

-------------------------RUN N = 30-------------------------

x_a =

  Columns 1 through 13

    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000

  Columns 14 through 26

    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000    0.0000    0.0000    0.0000    0.0001    0.0002
0.0009

  Columns 27 through 31

    0.0035    0.0145    0.0595    0.2439    1.0000


x_b =

  Columns 1 through 13

    0.0395    0.0116    0.0034    0.0010    0.0003    0.0001
0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000

  Columns 14 through 26

    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000    0.0000    0.0000    0.0000    0.0001    0.0002
0.0009

  Columns 27 through 31

    0.0035    0.0145    0.0595    0.2439    1.0000
```

```
----------------------RUN N = 37------------------------

x_a =

  Columns 1 through 13

    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000

  Columns 14 through 26

    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000

  Columns 27 through 38

    0.0000    0.0000    0.0000    0.0000    0.0001    0.0002
0.0009    0.0035    0.0145    0.0595    0.2439    1.0000


x_b =

  Columns 1 through 13

  213.8447   62.6260   18.3405    5.3711    1.5730    0.4607
0.1349    0.0395    0.0116    0.0034    0.0010    0.0003
0.0001

  Columns 14 through 26

    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
0.0000

  Columns 27 through 38

    0.0000    0.0000    0.0000    0.0000    0.0001    0.0002
0.0009    0.0035    0.0145    0.0595    0.2439    1.0000
```

PROBLEM 1 GRAPH



The cause of the deviation of the iterative method in problem 1 is due to the loss of accuracy from roundoff error.

****

Problem 1 is printed on the screen above. And graphs have been made as jpgs in the folder this was run from for 30 and 37 iterations. The output for problem 2 has been put into a file output.txt in the folder this was run as well.

****

OUTPUT.TXT

```
-------------------------PROBLEM #2-------------------------
----------------------BISECTION METHOD----------------------

Iteration: 53
Error: 0
Root: 328.1514
Function Value at Root: -8.6736e-18
Bisection Method Time: 0.019184


-----------------------NEWTONS METHOD-----------------------

Iteration: 5
Error: 0
Root: 328.1514
Function Value at Root: -8.6736e-18
Newton Method Time: 0.0068939


-------------------------SEC METHOD-------------------------

Iteration: 6
Error: 8.6736e-18
Root: 328.1514
Function Value at Root: -8.6736e-18
Secant Method Time: 0.0068939


------------------------ F-ZERO ------------------------

F-Zero Method Time: 0.0068939
```

Matlab Functions

PROBLEM 1 FUNCTION

```
      function [x_n, x_n_1, x_n_2, x_n_a, x_n_b] = xn(iterations, filename)
if(nargin < 2)
    filename = 'xn';
end
x_n = 1;
x_n_1 = 1/4.1;
a = 15/4.1;
b = 14/16.81;
x_a = [];
x_b = [];
iter = [];
if(iterations >= 0)
    while(iterations >= 0)
        if(iterations >= 2)

            while(iterations >=2)
                x_n_2 = a*x_n_1 - b*x_n;
                x_n = x_n_1;
                x_n_1 = x_n_2;
                x_n_b = x_n_2;
                x_n_a = 1/power(4.1, iterations);
                iter(end+1) = iterations;
                x_a(end+1) = x_n_a;
                x_b = [x_n_b, x_b];
                iterations = iterations-1;


            end

        elseif(iterations == 1)
            x_n_b = 1/4.1;
            x_n_a = 1/power(4.1, iterations);
            iter(end+1) = iterations;
            x_a(end+1) = x_n_a;
            x_b(end+1) = x_n_b;
            iterations = iterations - 1;

        elseif(iterations == 0)
            x_n_b = 1;
            x_n_a = 1/power(4.1, iterations);
            iter(end+1) = iterations;
            x_a(end+1) = x_n_a;
            x_b(end+1) = x_n_b;
            iterations = iterations - 1;
        end
    end

    x_a
    x_b
    xn_plot = plot(iter, x_a);
    hold all;
```

```
    xn_plot = plot(iter, x_b);
    file = strcat(filename, '.jpg');
    saveas(xn_plot, file);
elseif (iterations < 0)
        error('N must not be negative');
end
```

BISECTION METHOD

```
      function[iter, root, root_value, err] = bisection(a, b, fun, max_iter,
max_error, outputfile)

%Initalize values for tests
iter = 0;
err = 999;
root = 999;
output = fopen(outputfile, 'a');
%Test to see if a nd b have the same values or if a > b
% If a > b, swap a and b
if( a == b )
    fprintf('Please specify valid endpoints.');
    error('Endpoints are equal. Cannot evaluate.');
elseif( a > b )
    c = b;
    a = b;
    b = c;
    clear c;
end

%Test to see if interval is valid.
fa = fun(a);
fb = fun(b);
test = fa * fb;
if ( test >= 0 )
    fprintf('Given interval is not suitable for calculating the root.');
    error('Value of fa * fb is greater than or equal to zero.');
end


%Run first iteration of the bisection method manually for while loop
%to work
c_old = (a+b)/2;
fc = fun(c_old);
if(fc < fb)
    b = c_old;
elseif (fc > fa)
    a = c_old;
end

%Iteration of bisection method
while(iter < max_iter && err > max_error && fc ~= 0)
    iter = iter + 1;
    c_curr = (a+b)/2;
    err = abs((c_curr - c_old)/(c_curr));
    fa = fun(a);
    fb = fun(b);
    fc = fun(c_curr);
    %Test to determine which interval value should move towards the root
    if(fc > 0)
        b = c_curr;
    elseif (fc < 0)
```

```
        a = c_curr;
    end
    c_old = c_curr;
end

%Set values for output
root = c_old;
root_value = fc;

%Print all relevant information
fprintf(output, 'Iteration: %s\n', num2str(iter));
fprintf(output, 'Error: %s\n', num2str(err));
fprintf(output, 'Root: %s\n', num2str(root));
fprintf(output, 'Function Value at Root: %s\n', num2str(root_value));
fclose('all');
```

NEWTON-RAPHSON METHOD

```
      function[iter, root, root_value, err] = nr_method(x_n, fun, fprime,
max_iter, max_error, outputfile)

%Initalize values for tests
iter = 0;
err = 999;
fxn = fun(x_n);
output = fopen(outputfile, 'a');
%Check to see if f' is too close to zero
if(fprime(x_n) < 10^-12)
    error('F-Prime is too close to zero!');
end

%Check to see if given starting point is the root.
if(fxn == 0)
    error('Given x_0 is the root!');
end

%Newton-Raphson Method Iteration
while(iter < max_iter && err > max_error && fxn ~= 0)
    x_n_1 = x_n - (fun(x_n)/fprime(x_n));
    err = abs((x_n_1 - x_n)/(x_n));
    x_n = x_n_1;
    iter = iter+1;
    fxn = fun(x_n);
end

%Set root and root function value.
root = x_n;
root_value = fxn;


%Print all relevant information
fprintf(output, 'Iteration: %s\n', num2str(iter));
fprintf(output, 'Error: %s\n', num2str(err));
fprintf(output, 'Root: %s\n', num2str(root));
fprintf(output, 'Function Value at Root: %s\n', num2str(root_value));
fclose('all');
```

SECANT METHOD

```
    function[iter, root, root_value, err] = secant_method(x_nm_1, fun,
max_iter, max_error, outputfile)

%Initalize values for tests
iter = 0;
err = 999;
fxn = fun(x_nm_1);
x_n = x_nm_1 + 10^-8;
output = fopen(outputfile, 'a');

%Check to see if given starting point is the root.
if(fxn == 0)
    error('Given x_0 is the root!');
end

%Secant Method Iteration
while(iter < max_iter && err > max_error && fxn ~= 0)
    %Using x_n+1 = x_n - f(x_n)/Q(x_n-1, x_n),
    %where Q = [f(x_n-1) - f(x_n)]/[x_n-1 - x_n]:
    %diff 1 and 2 are the numerator and denominator of Q, respectively,
    %and denominator is the value of Q.
    diff1 = fun(x_nm_1) - fun(x_n);
    diff2 = (x_nm_1 - x_n);
    denominator = diff1/diff2;

    x_np_1 = x_n - (fun(x_n) / denominator);

    fxn = fun(x_np_1);
    err = abs(fxn);

    x_nm_1 = x_n;
    x_n = x_np_1;

    iter = iter + 1;
end

%Set root and root function value.
root = x_n;
root_value = fun(x_n);

%Print all relevant information
fprintf(output, 'Iteration: %s\n', num2str(iter));
fprintf(output, 'Error: %s\n', num2str(err));
fprintf(output, 'Root: %s\n', num2str(root));
fprintf(output, 'Function Value at Root: %s\n', num2str(root_value));
fclose('all');
```

FUNCTION

```
      function [y] = fun(R)

L = 5;
C = 10^-4;
t = 0.05;
qq0 = 0.01;


sq = sqrt([1/(L*C)] - [R/(2*L)]^2);
y = exp(-[R*t]/[2*L]) * cos(sq * t) - qq0;
```

F-PRIME

```
      function [y] = fprime(R)

L = 5;
C = 10^-4;
t = 0.05;

sq = sqrt([1/(L*C)] - [R/(2*L)]^2);
e = exp(-[(t*R)/(2*L)]);

y = -[t/(2*L)] * e * cos(sq * t) + [(R*t)/(4*(L^2)*sq)] * e * sin(sq*t);
```