# **Question 1**

**Part A**

To calculate the DC resistance of the object it can be split up into three distinct sections defined by the lengths , and respectively, the resistance for each section can then be calculated using the given formulas of;

Subbing into the equation the values given for and produces results for the first and third section of **0.412mΩ** and **24.9µΩ** respectively. Using the second formula above the resistance of the conical section is **0.312mΩ**. Therefore the total resistance of the object can be calculated by combining the resistance of each element to produce a total resistance of **0.748mΩ**.

**Part B**

At frequency the resistance of the object will change due to the skin effect. For the first and third section the skin effect will be constant across the length however, the radius of the conical section changes with length and therefore will be affected differently at higher frequencies. To calculate these changes the conical section is split into smaller chunks with increasing radius. The skin effect and the resistance can then be calculated for each chunk individually and summed together to produce an estimate for the resistance of the conical section at frequency.

The critical frequency is the point at which the skin effect is equal to the radius of the object, before this frequency is reached the resistance of the section will be at its normal DC value. The following MATLAB code calculates the critical frequency and DC resistance of each section of the object with the conical section being split into 1000 chunks.

alpha = atan((b-a)/Lc);

% Calculate critical frequencies and DC resistances

% First section

R1cf = critf(rho,a);

% Conical section

N = 1000;

Lc\_step = Lc/N;

for n = 1:1:N

h(n) = Lc\_step\*n.\*tan(alpha)+a;

RCcf(n) = critf(rho,h(n));

RCdc(n) = rwire(rho,Lc\_step,h(n));

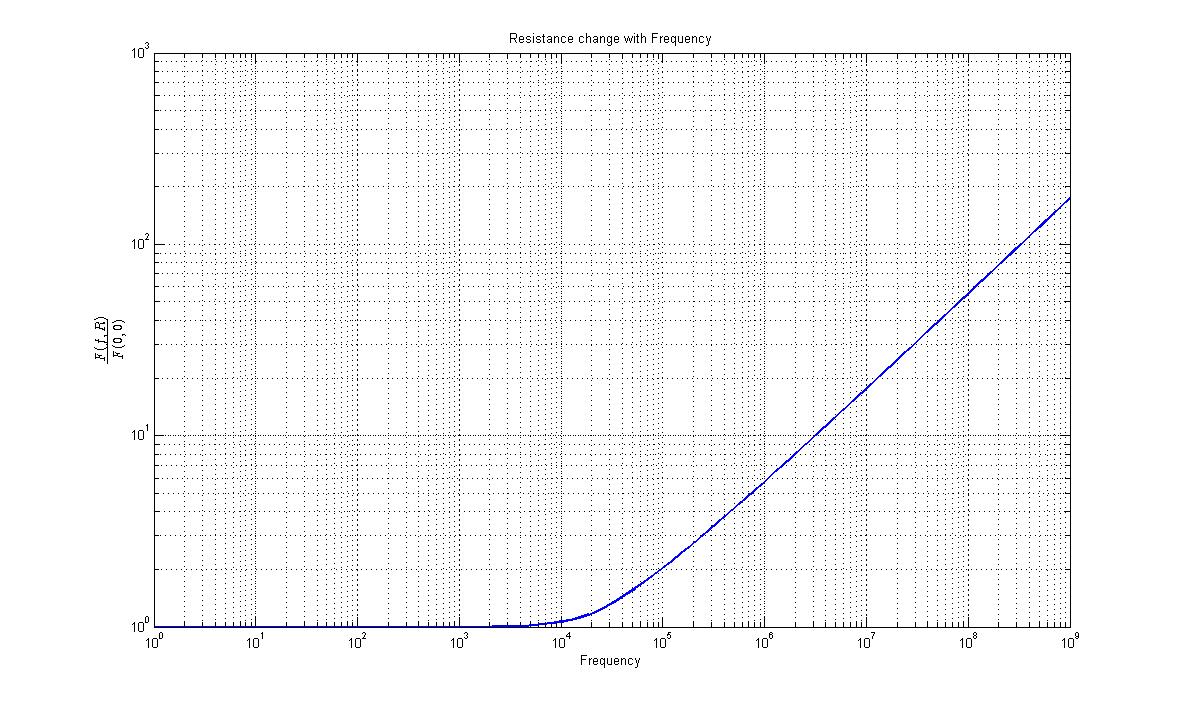
end

% End Section

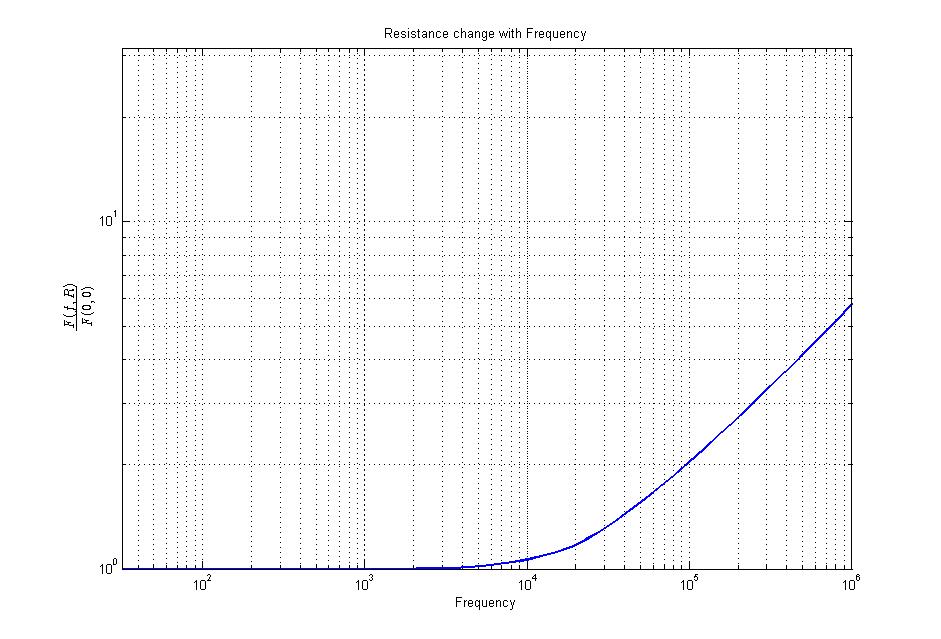
R2cf = critf(rho,b);

The code for the rwire, rconical and critf functions are found in Appendix B: Function Code.

These values are then used in a for loop to calculate the resistance of the object, as a whole, over a range of 100 frequencies on a logarithmic scale up to 1GHz. This for loop compares the current frequency value to the critical frequency of each section, if the frequency is equal to or greater than the critical frequency then the skin depth and resistance are calculated from that frequency, else the DC value is passed through. The code for this loop is found in Appendix A: Q1.m Part B. The result of this code is shown in Figure 1, a zoom of the area of change is shown in figure 2.



**Figure 1: Ratio of Resistance at frequency to DC Resistance.**

****

**Figure 2: Zoom of, Ratio of Resistance at frequency to DC Resistance.**

# **Question 2**

**Part A**

To calculate the resistance of the helical coil at DC the given formula is used. This produces a result of **17.55Ω**. The MATLAB code that produces this answer is found in Appendix A: MATLAB Code, this uses a user-defined function called rhelical, found in Appendix B: Function Code. This function calculates the resistance of a helical coil with constant or varying pitch at DC, the function returns both the DC resistance and the length of the wire in the coil.

**Part B**

At a frequency of 1MHz the resistance of the helical coil will be affected by both the skin and proximity effect. The following MATLAB code calculates the resistance at 1MHz by calculating the skin and proximity effects separately and multiplying them by the DC value.

%% Part B

% Estimate the resistance at 1MHz

delta = dskin(rho,1\*10^6);

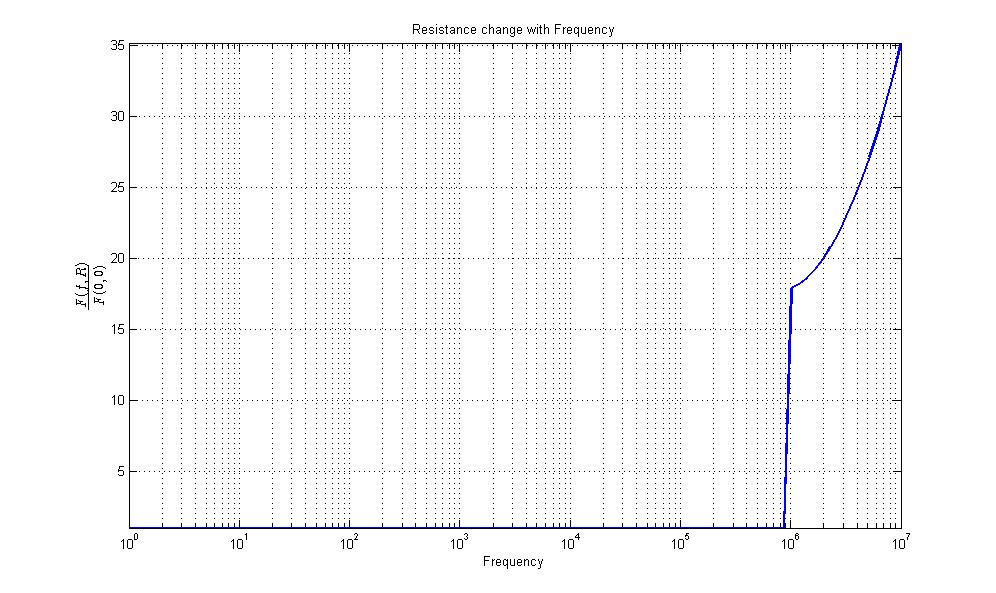
Sskin = 2.\*a.\*pi.\*delta.\*(1-(delta./(2.\*a)));

Rskin = (Coil\_length.\*rho)./Sskin;

RFreq = DCResis.\*Rskin.\*(1+((2.\*(a^2))./(p^2)))

This produces a result of approximately **314Ω** (313.79).

**Part C**

To calculate the resistance change over frequency the same method used in question 1 part b is used. Firstly the critical frequency of the helical coil is calculated using the created critf function, this value is then used in a for loop to calculate the resistance of the coil, taking into account the skin and proximity effect above the critical frequency. The result is shown in figure 3.

**Figure 3: Helical Coil Resistance Ratio at Frequency.**

**The following MATLAB code produces the results for the graph above, the full code can be found in Appendix A: MATLAB Code Q2.m.**

Rcf = critf(rho,a);

% Loop to calculate resistance with frequency

count = 100;

RT = zeros(count);

Freq = logspace(0,7,count);

for ln = 1:1:count;

if Freq(ln) >= Rcf

delta = dskin(rho,Freq(ln));

Sskin = 2.\*a.\*pi.\*delta.\*(1-(delta./(2.\*a)));

Rskin = (Coil\_length.\*rho)./Sskin;

RT(ln) = DCResis.\*Rskin.\*(1+((2.\*(a^2))./(p^2)));

else

RT(ln) = DCResis;

end

end

**Part D**

**To maintain a constant resistance at high frequencies the radius of the wire used needs to be reduced. This causes the critical frequency of the wire to increase as skin depth effects are reduced. However, by changing the radius of the wire the inductance will also change. To counteract this a stranded wire with individually insulated conductors can be used, each wire has a radius less than the skin depth at a high frequency and are individually insulated to avoid the wires shorting with each other. The wires are twisted together such that the changes in the electromagnetic field do not affect the resistance of the conductor. A wire of this type is called Litz Wire.**

# **Question 3**

**Design**

Blah blah blah design

**Figure 4: Ratio of Frequency to DC Resistance**

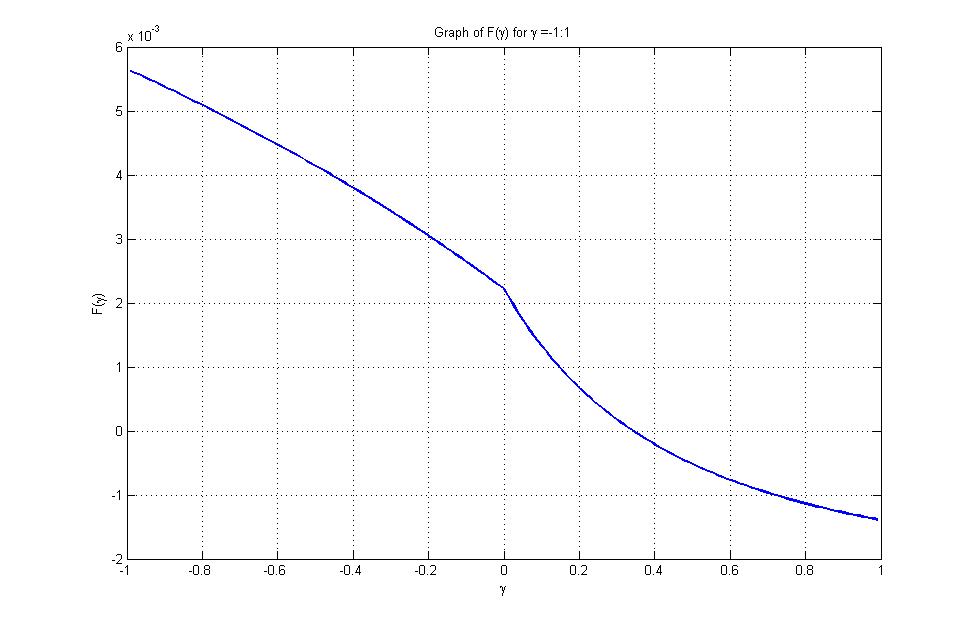
# **Question 4**

For Part A an Part B see notes 1 and 2 as attached.

**Part C**

To accurately calculate the values of R and L with the given variables the value of needs to be calculated. This is done by creating a function of by multiplying the given equations (1) and (2) and solving for zero ie. Finding the roots of the equation;

As the solution to the equation will lie between zero and one the graph for can be plotted for between minus one and one, shown in figure 5. Using the interp1 function in MATLAB the value of that solves the equation can be found as **0.3321**.



**Figure 5: F(γ) for γ = -1:1**

**From this value of γ the value of L can be calculated using equation (1) to produce a value of 0.38µH. Rearranging the equation for γ in terms of R and using the calculated value of L produces a result for R as 1.29Ω using the following MATLAB code.**

L = ((V0.\*sqrt(C).\*g(gamma))./Im1).^2

R = 2.\*gamma.\*sqrt(L./C)

**Part D**

**To estimate the values of L and R the current, as a function of time, is used to generate a set of data using the values calculated in part C. The following MATLAB code generates the graph for I(t), shown in figure 6 and calculates the estimates for L and R as 0.43µH and 1.46Ω respectively, generating an error of 13.8% and 13.1%.**

omega = (1/sqrt(L\*C)).\*sqrt(1-(gamma^2))

t = 0:1\*10^-10:0.6\*10^-5;

Im = (V0./(omega.\*L)).\*sin(omega.\*t).\*exp(-(R/(2\*L)).\*t);

tm = (sqrt(L.\*C)).\*((1./sqrt(1-(gamma.^2))).\*asin(sqrt(1-(gamma.^2))));

figure

plot(t.\*10^6,Im./(1\*10^3),'Linewidth',2)

grid on

xlabel('Time(\mus)')

ylabel('I(kA)')

title('Discharge Current for 30kV V\_{0}')

% Find Im2 and Tz

Im2 = min(Im);

Tzidx = find(Im == min(Im));

T2 = t(Tzidx);

Tz = T2-tm1;

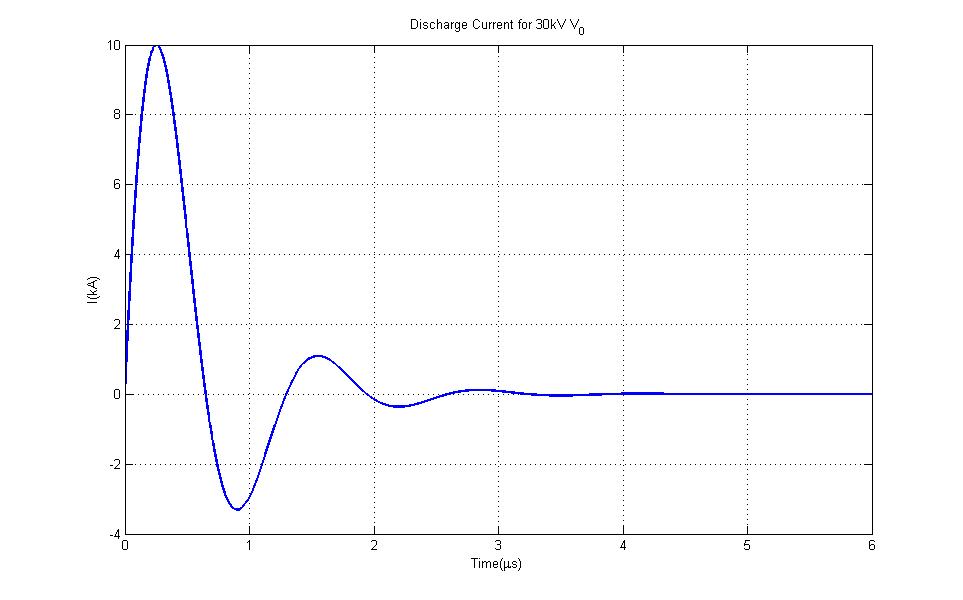
T = 2\*Tz;

Lest = (T^2)./(4.\*(pi^2).\*C)

Rest = (-2./pi).\*(sqrt(Lest./C)).\*(log(abs(Im2)./Im1))

Lerr1 = ((Lest-L)/L)\*100

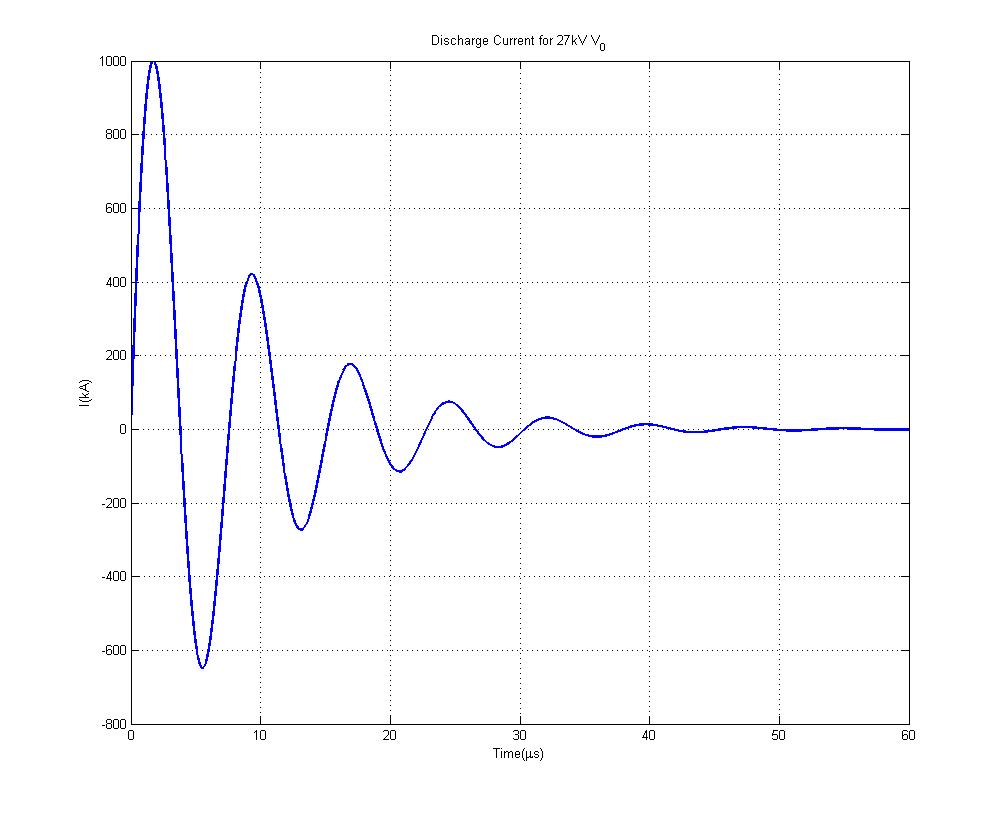
Rerr1 = ((Rest-R)/R)\*100



**Figure 6: Discharge Current at 30kV**

**Part E**

Repeating the steps in part C and D for the new conditions produces a value for **γ of 0.1464 and gives values for L and R of 26.83nH and 6.4mΩ respectively. Plotting the function of I(t) produces the graph shown in figure 7. From this dataset the values of L and R are estimated as 26.39nH and 6.5mΩ respectively producing errors of 2.2% and 2.2%.**



**Figure 7: Discharge Current for 27kV**

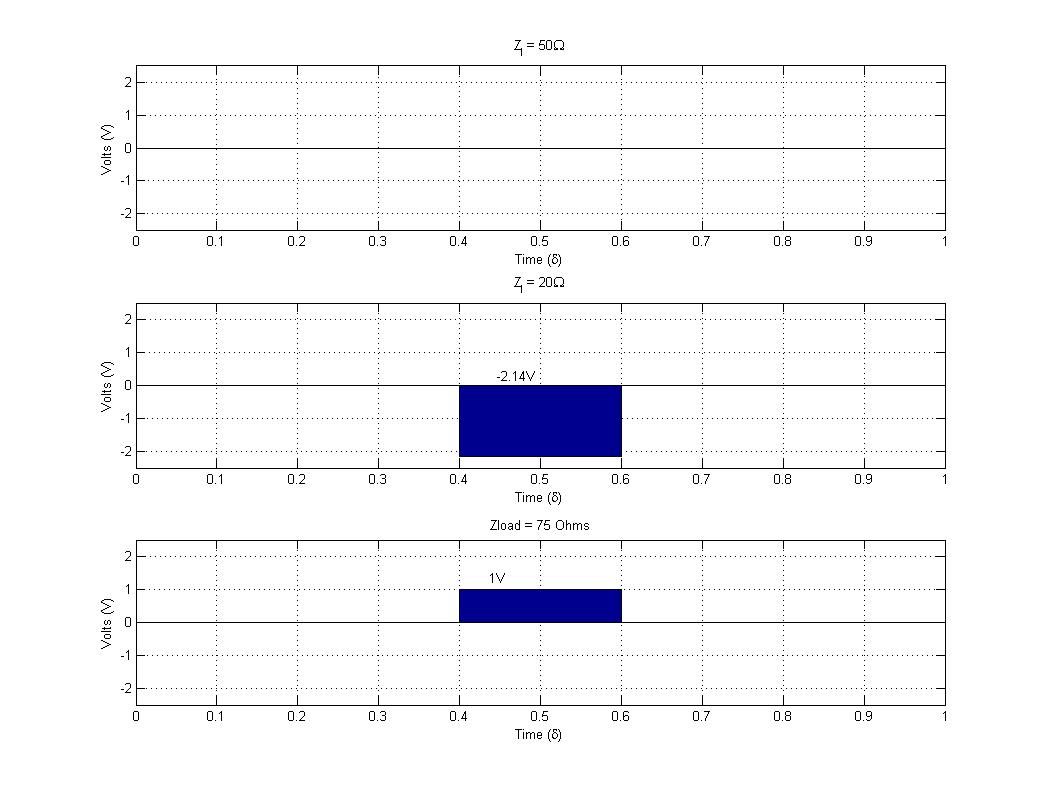
**Part F**

The equations to calculate inductance and resistance derived from equations (1) and (2) use the maximum current and charging voltage of the circuit. At this time the electromagnetic force created by this current can cause the inductors to move. This appears as an increase on the resistance of the circuit. The simplified calibration shot equations take into account the discharge period of the circuit which reduces the effect of the conductor movement under large current discharge.

# **Question 5**

**Part A**

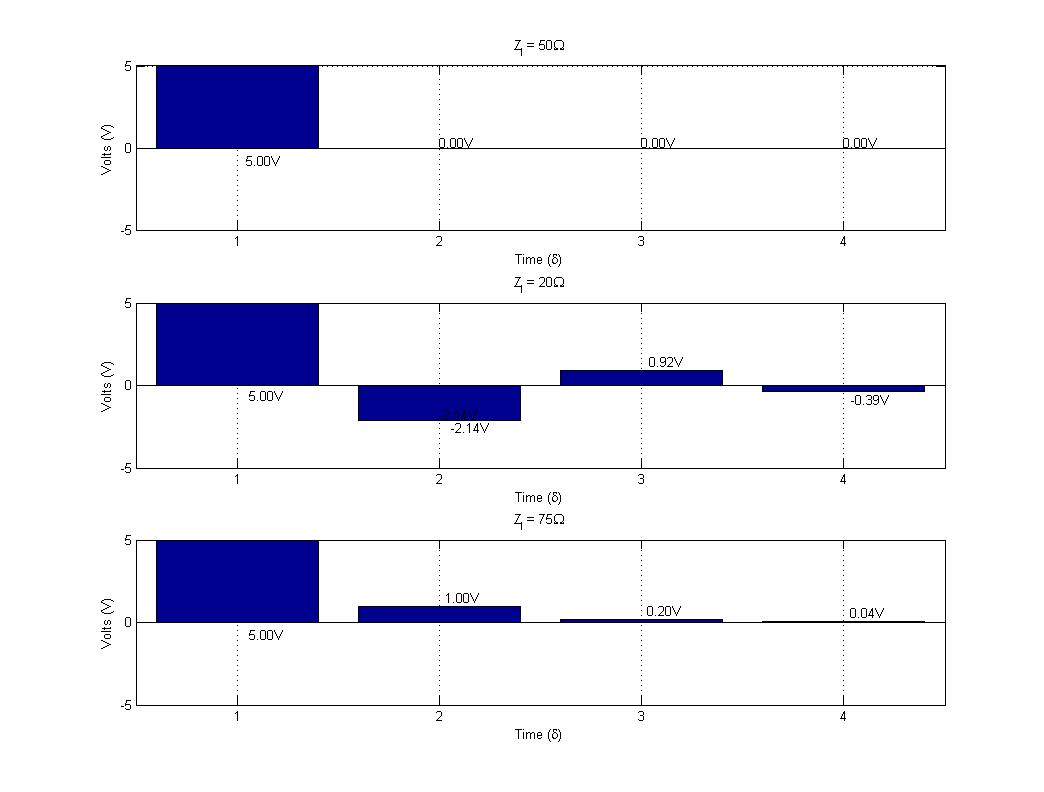
Figure 8 shows the voltage pulse for the three cases.



**Figure 8: Voltage Pulse at time**

**Part B**

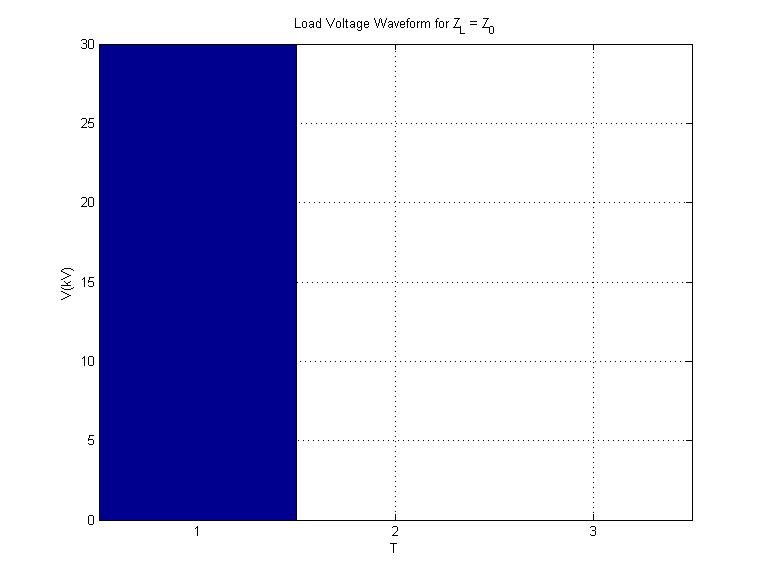
Figure 9 shows the load voltage pulse from a t=0 up to t=4δ.



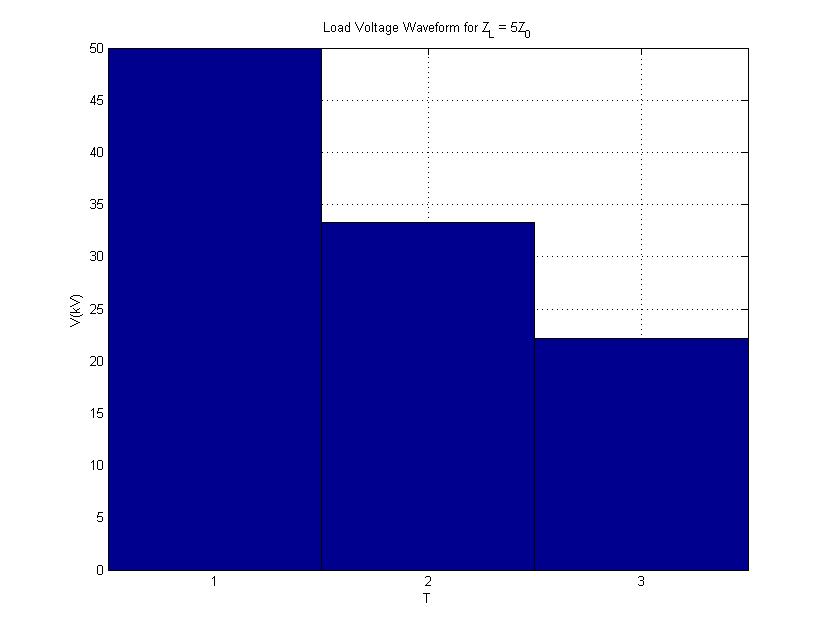
**Figure 9: Voltage Load History**

# **Question 6**

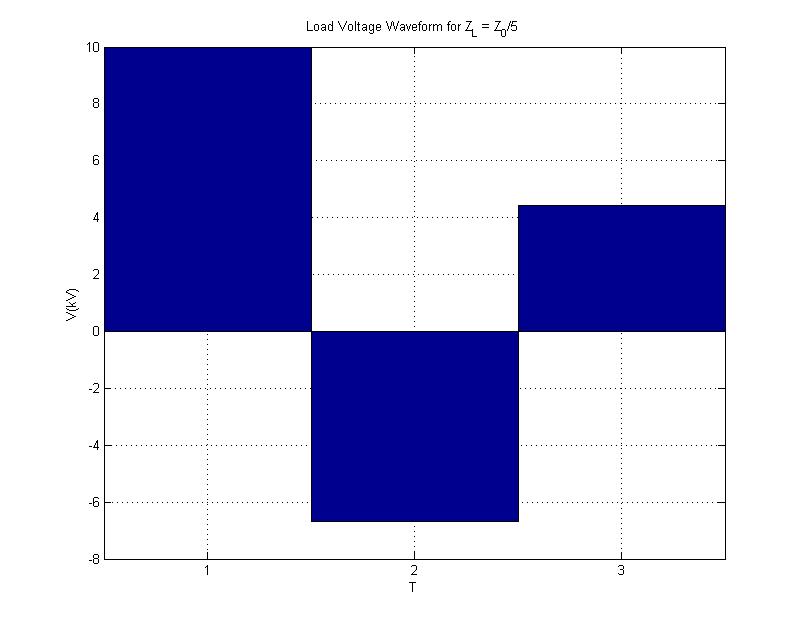
To calculate the number of pulses the amount of time for the pulse to travel twice the length of the line is calculated. The number of pulses is then the total time given (37.6ns) divided by the pulse width. This gives the number of pulses as approximately 3, hence the figures 10, 11 and 12 show the load voltage for the three cases of load impedance for a time of 3T.



**Figure 10: Load Voltage 15Ω**



**Figure 11: Load Voltage 75Ω**



**Figure 12: Load Voltage 3Ω**

# **Question 7**

**Stray Capacitance**

To calculate the stray capacitance between the generator circuit and the outer conductor the distance between the two needs to be determined. This is calculated from the breakdown voltage of nitrogen and the desired peak voltage of the Marx generator. From this the capacitance can be calculated using the equation for a transmission line and gives a value of **5.28pF**.

**Pressure and Spark Gap Capacitance**

Rearranging the given formula for the breakdown voltage of a spark gap with a Vbreak of 20kV gives the pressure required for the spark gap to break at 20kV as approximately **535kPa (534.69)**. Using the formula given in the notes the spark gap capacitance is **30pF (29.95)**.

**Voltage Loss per stage and number of Stages**

With the calculated spark gap and stray capacitance the voltage loss can be calculated as **5.37% per stage**. The following MATLAB code then calculates the required amount of stages needed to reach a peak voltage of 170kV. This produces the number of stages required as **12**.

**Feed Forward, Charging Current and Power**

Given that the Marx generator needs to operate at 10Hz the time for the capacitor charging is 0.1s. The following MATLAB code calculates the feed forward resistors as **1MΩ**, required charging current as **48mA** and a max power of **2.3kW**.

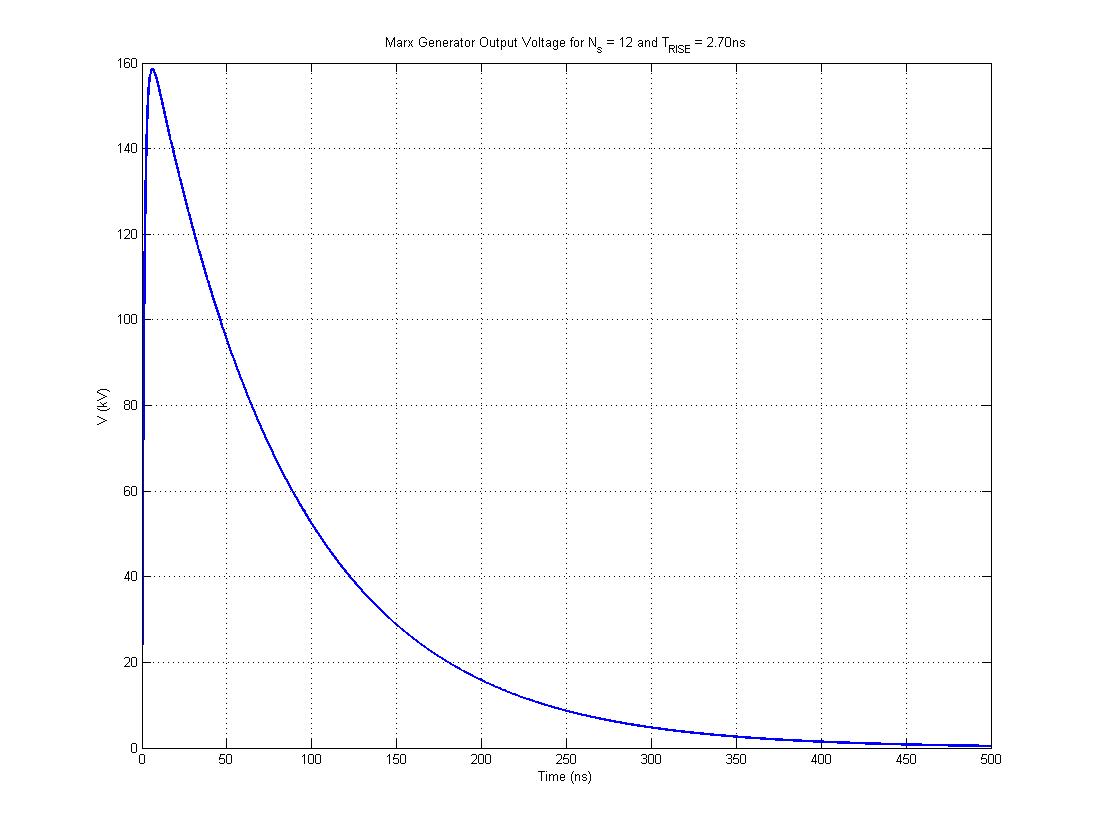
RFF = 0.1/(5\*CMarx)

CCharge = (CMarx\*CMarxCharge)\*Ns;

I = CCharge/0.1

MaxPow = (I^2)\*RFF

Given these conditions and using the equation provided in the notes, the load voltage for a time period of 500ns is shown in figure 13 with a rise time of **2.7ns** and a peak voltage of **158.57kV**.



**Figure 13: Marx Generator Pulse for Ns = 12**

**Additional Stages**

**As the peak voltage is below the required 170kV additional stages need to be added to the generator. The following MATLAB code re-calculates the values above when adding an additional stage. The code will continue to add stages until the peak voltage on the load is greater than the peak voltage required.**

while VMaxPeak<VPeak

% Add stage

Ns = Ns+1;

% Re-calculate VMax

VCStage = VPrevStage\*VLoss;

VMax = VCStage+VMax;

VPrevStage = VCStage;

% Re-calculate Power

CCharge = (CMarx\*CMarxCharge)\*Ns;

I = CCharge/0.1;

MaxPow = (I^2)\*RFF;

% VLoad

RTotal = RSparkGap.\*Ns;

CTotal = CMarx./Ns;

Beta1 = 1./(RTotal.\*CLoad);

Beta2 = 1./(RLoad.\*CTotal);

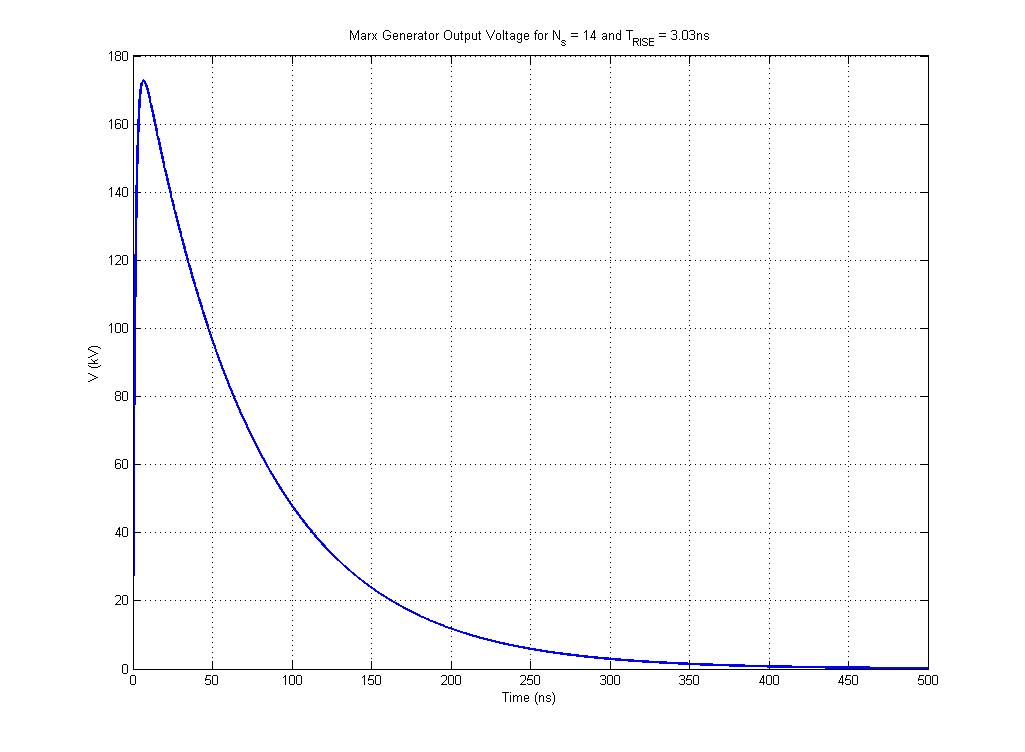
VLoad=(VMax/((Beta1-Beta2)\*RTotal\*CLoad))\*(exp(-Beta2\*(t))-exp(-Beta1\*(t)));

VMaxPeak = max(VLoad);

rt = risetime(VLoad)./10; %rise time in ns

end

**This generates the following values and produces the graph shown in figure 14; Stages = 14, Charging Current = 56mA, Maximum Power = 3.1kW, Peak Voltage = 172.8kV and Rise Time = 3.03ns.**

**Figure 14: Marx Generator Pulse for Ns = 12**

# **Appendix A: MATLAB Code**

**Q1.m**

%% Fast Tansient Sensors - Q1 - Coursework 2

% B126949 - Tom Young

%% Pre - Cursor

clear all

clc

%% Constants

% Using 1&9 give constants as O-O

a = 3.5\*10^-3; %mm

b = 10.5\*10^-3; %mm

La = 22\*10^-3; %mm

Lb = 12\*10^-3; %mm

Lc = 50\*10^-3; %mm

rho = 7.2\*10^-7; %Ohm-Meter

%% Part A

% Calculate the DC Resistance of the conductor

% Resistence of small wire

R1 = rwire(rho,La,a);

% Resistence of large wire

R2 = rwire(rho,Lb,b);

% Resistance of conical section

RC\_DC = rconical(a,b,Lc,rho);

% Total resistance

DCResis = R1 + R2 + RC\_DC

%% Part B

% Draw a graphical representation of the conductor resistance variation

% with frequency up to 1GHz

alpha = atan((b-a)/Lc);

% Calculate critical frequencies and DC resistances

% First section

R1cf = critf(rho,a);

% Conical section

N = 1000;

Lc\_step = Lc/N;

for n = 1:1:N

h(n) = Lc\_step\*n.\*tan(alpha)+a;

RCcf(n) = critf(rho,h(n));

RCdc(n) = rwire(rho,Lc\_step,h(n));

end

% End Section

R2cf = critf(rho,b);

% Loop to calculate resistance with frequency

% Number of loops

count = 100;

RFreq=zeros(count);

Freq = logspace(0,9,count);

for ln = 1:1:count;

RC = 0;

% First Section

if Freq(ln) >= R1cf

delta = dskin(rho,Freq(ln));

Sskin = 2.\*a.\*pi.\*delta.\*(1-(delta./(2.\*a)));

RF = (La.\*rho)./Sskin;

%RF = R1.\*(a./(2.\*delta));

else

RF = R1;

end

% Conical Section

for n = 1:1:N

if Freq(ln) >= RCcf(n)

delta = dskin(rho,Freq(ln));

Sskin = 2.\*h(n).\*pi.\*delta.\*(1-(delta./(2.\*h(n))));

RC(n) = (Lc\_step.\*rho)./Sskin;

%RC(n) = RCdc(n).\*(h(n)./(2.\*delta));

else

RC(n) = RCdc(n);

end

end

% End Section

if Freq(ln) >= R2cf

delta = dskin(rho,Freq(ln));

Sskin = 2.\*b.\*pi.\*delta.\*(1-(delta./(2.\*b)));

RE = (Lb.\*rho)./Sskin;

%RE = R2.\*(b./(2.\*delta));

else

RE = R2;

end

% Total resistance

RFreq(ln) = (RE+RF+sum(RC));

end

% Plot data

loglog(Freq,RFreq./DCResis,'b','linewidth',2)

grid on

xlabel('Frequency')

ylabel('$${F(f,R)\over F(0,0) }$$','Interpreter','Latex')

xlim([0 1\*10^9])

title('Resistance change with Frequency')

**Q2.m**

%% Fast Tansient Sensors - Q2 - Coursework 2

% B126949 - Tom Young

%% Pre - Cursor

clear all

clc

%% Constants

% Using 1&9 give constants as O-O

d = 0.14\*10^-3; %mm

D = 4\*10^-2; %cm

L = 10\*10^-2; %cm

p = 0.8\*10^-3; %mm

rho = 1.72\*10^-8; %Ohm-Meter

a = d/2; %radius mm

%% Part A

% Caculate the DC resistance

[DCResis, Coil\_length] = rhelical(L,d,p,rho,D);

DCResis

%% Part B

% Estimate the resistance at 1MHz

delta = dskin(rho,1\*10^6);

Sskin = 2.\*a.\*pi.\*delta.\*(1-(delta./(2.\*a)));

Rskin = (Coil\_length.\*rho)./Sskin;

RFreq = DCResis.\*Rskin.\*(1+((2.\*(a^2))./(p^2)))

%% Part C

% Draw a graphical representation of the resistance variation with

% frequency up to a frequency of 1GHz.

Rcf = critf(rho,a);

% Loop to calculate resistance with frequency

count = 100;

RT = zeros(count);

Freq = logspace(0,7,count);

for ln = 1:1:count;

if Freq(ln) >= Rcf

delta = dskin(rho,Freq(ln));

Sskin = 2.\*a.\*pi.\*delta.\*(1-(delta./(2.\*a)));

Rskin = (Coil\_length.\*rho)./Sskin;

RT(ln) = DCResis.\*Rskin.\*(1+((2.\*(a^2))./(p^2)));

else

RT(ln) = DCResis;

end

end

% Plot data

semilogx(Freq,RT./DCResis,'b','linewidth',2)

%semilogx(Freq,RT,'b','linewidth',2)

grid on

xlabel('Frequency')

ylabel('$${F(f,R)\over F(0,0) }$$','Interpreter','Latex')

xlim([0 10\*10^6])

ylim([1 inf])

title('Resistance change with Frequency')

**Q3.m**

**Q4.m**

%% Fast Tansient Sensors - Q4 - Coursework 2

% B126949 - Tom Young

%% Pre - Cursor

clear all

clc

%% Constants

C = 0.1\*10^-6;

V0 = 30\*10^3;

Im1 = 10\*10^3;

tm1 = 250\*10^-9;

%% Part C

% From the equations given make an accurate calculation of R and L.

gamma = -1:0.01:1;

F = C.\*V0.\*f(gamma).\*g(gamma)-(tm1\*Im1);

% Plot

figure

plot(gamma,F,'linewidth',2)

grid on

xlabel('\gamma')

ylabel('F(\gamma)')

title('Graph of F(\gamma) for \gamma =-1:1')

F(1) = [];

F(200) = [];

% From datapoint method at F(gamma) = 0, gamma = 0.3421

gamma = interp1(F,gamma,0);

% Calculate L using Im1 equation and R using gamma equation.

L = ((V0.\*sqrt(C).\*g(gamma))./Im1).^2

R = 2.\*gamma.\*sqrt(L./C)

%% Part D

% Calculate second peak and time using c

%gamma = 0.5\*R\*sqrt(C/L)

omega = (1/sqrt(L\*C)).\*sqrt(1-(gamma^2))

t = 0:1\*10^-10:0.6\*10^-5;

Im = (V0./(omega.\*L)).\*sin(omega.\*t).\*exp(-(R/(2\*L)).\*t);

tm = (sqrt(L.\*C)).\*((1./sqrt(1-(gamma.^2))).\*asin(sqrt(1-(gamma.^2))));

figure

plot(t.\*10^6,Im./(1\*10^3),'Linewidth',2)

grid on

xlabel('Time(\mus)')

ylabel('I(kA)')

title('Discharge Current for 30kV V\_{0}')

% Find Im2 and Tz

Im2 = min(Im);

Tzidx = find(Im == min(Im));

Tz = t(Tzidx);

T = 2\*Tz;

Lest = (T^2)./(4.\*(pi^2).\*C)

Rest = (-2./pi).\*(sqrt(Lest./C)).\*(log(abs(Im2)./Im1))

Lerr1 = ((Lest-L)/L)\*100

Rerr1 = ((Rest-R)/R)\*100

%% Part E

% Repeat C and D from the following conditions.

C = 54\*10^-6;

V0 = 27\*10^3;

Im1 = 1\*10^6;

tm1 = 1.7\*10^-6;

% Repeat from part C

gamma = -1:0.01:1;

F = C.\*V0.\*f(gamma).\*g(gamma)-(tm1\*Im1);

% Plot

figure

plot(gamma,F,'linewidth',2)

grid on

xlabel('\gamma')

ylabel('F(\gamma)')

title('Graph of F(\gamma) for \gamma =-1:1')

% From datapoint method at F(gamma) = 0, gamma = 0.1464

F(1) = [];

F(200) = [];

% Find gamma

gamma = interp1(F,gamma,0)+0.01;

% Calculate L using Im1 equation and R using gamma equation.

L = ((V0.\*sqrt(C).\*g(gamma))./Im1).^2

R = 2.\*gamma.\*sqrt(L./C)

% Repeat from Part D

omega = (1/sqrt(L\*C)).\*sqrt(1-(gamma^2))

t = 0:1\*10^-10:0.6\*10^-4;

Im = (V0./(omega.\*L)).\*sin(omega.\*t).\*exp(-(R/(2\*L)).\*t);

tm = (sqrt(L.\*C)).\*((1./sqrt(1-(gamma.^2))).\*asin(sqrt(1-(gamma.^2))));

figure

plot(t.\*10^3,Im./(1\*10^3),'Linewidth',2)

grid on

xlabel('Time(ms)')

ylabel('I(kA)')

title('Discharge Current for 27kV V\_{0}')

% Find Im2 and Tz

% Second peak

Im2 = min(Im);

Tzidx = find(Im == Im2);

Tz = t(Tzidx);

T = 2\*Tz;

Lest = (T^2)./(4.\*(pi^2).\*C)

Rest = (-2./pi).\*(sqrt(Lest./C)).\*(log(abs(Im2)./Im1))

Lerr2 = ((Lest-L)/L)\*100

Rerr2 = ((Rest-R)/R)\*100

**Q5.m**

%% Fast Tansient Sensors - Q5 - Coursework 2

% B126949 - Tom Young

%% Pre - Cursor

clear all

clc

%% Constants

Z0 = 50; % 50 Ohms

V1 = 5; % 5V

% Setup delta

delta = [0/4,1/4,2/4,3/4,4/4];

%% Part A

% i)

Zl1 = 50;

Pl1 = (Zl1-Z0)/(Zl1+Z0);

V1l = V1\*Pl1;

V1l = [0,0,V1l,0,0];

% ii)

Zl2 = 20;

Pl2 = (Zl2-Z0)/(Zl2+Z0);

V2l = V1\*Pl2;

V2l = [0,0,V2l,0,0];

% iii)

Zl3 = 75;

P3 = (Zl3-Z0)/(Zl3+Z0); % 0.2

V3l = V1\*P3; % 1V

V3l = [0,0,V3l,0,0];

% Plot

figure

% i)

subplot(3,1,1)

bar(delta,V1l)

title('Z\_{l} = 50\Omega')

ylabel('Volts (V)')

xlabel('Time (\delta)')

ylim([-2.5, 2.5])

xlim([0,1])

grid on

% ii)

subplot(3,1,2)

bar(delta,V2l)

title('Z\_{l} = 20\Omega')

ylabel('Volts (V)')

xlabel('Time (\delta)')

ylim([-2.5, 2.5])

xlim([0,1])

grid on

% iii)

subplot(3,1,3)

bar(delta,V3l)

title('Zload = 75 Ohms')

ylabel('Volts (V)')

xlabel('Time (\delta)')

ylim([-2.5, 2.5])

xlim([0,1])

grid on

%% Part B

V0 = 5;

delta = [1,2,3,4];

% i)

Pl1 = (Zl1-Z0)/(Zl1+Z0);

V1a = V0; %

V1b = V1a\*Pl1;

V1c = V1b\*Pl1;

V1d = V1c\*Pl1;

V1 = [V1a, V1b, V1c, V1d];

V1vals = strtrim(cellstr(num2str(V1(:),'%.2fV')));

% ii)

Pl2 = (Zl2-Z0)/(Zl2+Z0);

V2a = V0;

V2b = V2a\*Pl2;

V2c = V2b\*Pl2;

V2d = V2c\*Pl2;

V2 = [V2a, V2b, V2c, V2d];

V2vals = strtrim(cellstr(num2str(V2(:),'%.2fV')));

% iii)

P3 = (Zl3-Z0)/(Zl3+Z0);

V3a = V0;

V3b = V3a\*P3;

V3c = V3b\*P3;

V3d = V3c\*P3;

V3 = [V3a, V3b, V3c, V3d];

V3vals = strtrim(cellstr(num2str(V3(:),'%.2fV')));

% Plot graph

figure

% i)

subplot(3,1,1)

bar(delta,V1)

text(delta,V1,V1vals,'VerticalAlignment','baseline');

title('Z\_{l} = 50\Omega')

ylabel('Volts (V)')

xlabel('Time (\delta)')

ylim([-5, 5])

grid on

% ii)

subplot(3,1,2)

bar(delta,V2)

text(delta,V2,V2vals,'VerticalAlignment','baseline');

title('Z\_{l} = 20\Omega')

ylabel('Volts (V)')

xlabel('Time (\delta)')

ylim([-5, 5])

grid on

% iii)

subplot(3,1,3)

bar(delta,V3)

text(delta,V3,V3vals,'VerticalAlignment','baseline');

title('Z\_{l} = 75\Omega')

ylabel('Volts (V)')

xlabel('Time (\delta)')

ylim([-5, 5])

grid on

**Q6.m**

%% Fast Tansient Sensors - Q6 - Coursework 2

% B126949 - Tom Young

%% Pre - Cursor

clc

clear all

%% Variables

u0 = 4\*pi\*10^-7; % Free space permeability

e0 = 8.85\*10^-12; % Free space permittivity

Z0 = 15; % 15Ohms

L = 1.25; % 1.25m - axial length

er = 2.25; % relative permittivity

V0 = 60\*10^3; % 60kV

t = 37.6\*10^-9; % 37.6ns

T = [1,2,3];

%% Part A

Z1 = Z0;

VPFL1 = (V0\*Z1)/(Z0+Z1);

rhoL = (Z1-Z0)/(Z1+Z0);

VL1 = VPFL1;

VL2 = VL1\*rhoL;

VL3 = VL2\*rhoL;

V1 = [VL1,VL2,VL3];

% Plot

figure

bar(T,V1./10^3,1);

title('Load Voltage Waveform for Z\_{L} = Z\_{0}');

ylabel('V(kV)');

xlabel('T');

grid on

%% Part B

Z2 = 5\*Z0;

VPFL2 = (V0\*Z2)/(Z0+Z2);

rhoL = (Z2-Z0)/(Z2+Z0);

VL1 = VPFL2;

VL2 = VL1\*rhoL;

VL3 = VL2\*rhoL;

V2 = [VL1,VL2,VL3];

% Plot

figure

bar(T,V2./10^3,1);

title('Load Voltage Waveform for Z\_{L} = 5Z\_{0}');

ylabel('V(kV)');

xlabel('T');

grid on

%% Part C

Z3 = Z0/5;

VPFL3 = (V0\*Z3)/(Z0+Z3);

rhoL = (Z3-Z0)/(Z3+Z0);

VL1 = VPFL3;

VL2 = VL1\*rhoL;

VL3 = VL2\*rhoL;

V3 = [VL1,VL2,VL3];

% Plot

figure

bar(T,V3./10^3,1);

title('Load Voltage Waveform for Z\_{L} = Z\_{0}/5 ');

ylabel('V(kV)');

xlabel('T');

grid on

**Q7.m**

%% Fast Tansient Sensors - Q7 - Coursework 2

% B126949 - Tom Young

%% Pre - Cursor

clear all

clc

%% Constants

% Marx Constants

Freq = 10;

VPeak = 170\*10^3;

TRise = 3\*10^-9;

RLoad = 50;

CLoad = 2.5\*10^-9;

CMarx = 20\*10^-9;

CMarxCharge = 20\*10^3;

EFBreak = 70\*10^5; %kV/m

MarxSL = 50\*10^-3;

MarxOD = 70\*10^-3;

SgBreak = 20\*10^3;

RSparkGap = 50\*10^-3;

SgSpRadius = 2.5\*10^-3;

SgDist = 0.5\*10^-3;

% Other constants

P0 = 101.325\*10^3;

E0 = 8.85\*10^-12;

u\_0 = 4\*pi\*10^-7;

%% Stray Capacitance

MarxCR = (VPeak/EFBreak)+(MarxOD/2);

CStray = ctrans(MarxCR,MarxOD/2)\*MarxSL

%% Pressure Calculation

PressCalc\_1 = 12260\*exp((-2.1\*10^-3)/SgSpRadius);

PressCalc\_2 = (SgDist\*10^3)^(0.49-(4\*SgSpRadius)+288\*(SgSpRadius^2));

P = (SgBreak\*P0)/(PressCalc\_1\*PressCalc\_2)

%% Spark Gap Self Capacitance

CGap = csparkgap(SgSpRadius,SgDist,15)

%% Voltage Loss

VLoss = (CStray/(CGap+CStray));

VLossPerCent = 100-(VLoss\*100)

%% Stages

VPrevStage = CMarxCharge;

VCStage = 0;

VMax = 0;

Ns = 0;

while VMax < VPeak

VCStage = VPrevStage\*VLoss;

VMax = VCStage+VMax;

VPrevStage = VCStage;

Ns = Ns + 1;

end

Ns

%% Charge Current, feed forward, power

%ffr

RFF = 0.1/(5\*CMarx)

CCharge = (CMarx\*CMarxCharge)\*Ns;

I = CCharge/0.1

MaxPow = (I^2)\*RFF

%% Time dependance of load

% Load voltage

RTotal = RSparkGap.\*Ns;

CTotal = CMarx./Ns;

Beta1 = 1./(RTotal.\*CLoad);

Beta2 = 1./(RLoad.\*CTotal);

t=0:1\*10^-10:5\*10^-7;

VLoad=(VMax/((Beta1-Beta2)\*RTotal\*CLoad))\*(exp(-Beta2\*(t))-exp(-Beta1\*(t)));

rt = risetime(VLoad)./10

% Plot

plot(t.\*10^9,VLoad./10^3,'Linewidth',2)

grid on

str = sprintf('Marx Generator Output Voltage for N\_{s} = %d and T\_{RISE} = %.2fns',Ns,rt);

title(str);

ylabel('V (kV)');

xlabel('Time (ns)');

VMaxPeak = max(VLoad)

% Increase stages if necessary

while VMaxPeak<VPeak

% Add stage

Ns = Ns+1;

% Re-calculate VMax

VCStage = VPrevStage\*VLoss;

VMax = VCStage+VMax;

VPrevStage = VCStage;

% Re-calculate Power

CCharge = (CMarx\*CMarxCharge)\*Ns;

I = CCharge/0.1;

MaxPow = (I^2)\*RFF;

% VLoad

RTotal = RSparkGap.\*Ns;

CTotal = CMarx./Ns;

Beta1 = 1./(RTotal.\*CLoad);

Beta2 = 1./(RLoad.\*CTotal);

VLoad=(VMax/((Beta1-Beta2)\*RTotal\*CLoad))\*(exp(-Beta2\*(t))-exp(-Beta1\*(t)));

VMaxPeak = max(VLoad);

rt = risetime(VLoad)./10; %rise time in ns

end

% Plot final result

figure

plot(t.\*10^9,VLoad./10^3,'Linewidth',2)

grid on

str = sprintf('Marx Generator Output Voltage for N\_{s} = %d and T\_{RISE} = %.2fns',Ns,rt);

title(str);

ylabel('V (kV)');

xlabel('Time (ns)');

# **Appendix B: Function Code**

**rwire.m**

function [ rwire ] = rwire( resistivity, length, radius )

%RWIRE rwiretace of a wire of length l, with cross sectional area A, made

%of a material with risitivity ro.

% rwiretace of a wire derived from Ohm's law.

cross\_sectional\_area = pi.\*(radius.^2);

rwire = resistivity.\*(length./cross\_sectional\_area);

end

**rconical.m**

function [ rconical ] = rconical( starting\_radius, ending\_radius, length, resistivity )

%RCONICAL Resistace of a conical conductor of length l, with variable

% cross sectional area A, made of a material with risitivity ro.

% Resistance of a conical conductor of length l with varable cross

% section over length c.

rconical = resistivity.\*(length./(pi.\*starting\_radius.\*ending\_radius));

end

**critf.m**

function [ critf ] = critf( resitivity, radius )

%CRITF Calculates critical frequency

% Calculates ciritcal frequency when the skin effect is equal to the

% radius of the object.

%% Constants

u\_0 = 4\*pi\*10^-7;

%% Equations

critf = resitivity./((radius.^2).\*pi.\*u\_0);

end

**dskin.m**

function [ dskin ] = dskin( resitivity, frequency )

%DSKIN Calculates skin effect

% Calculates the skin effect depth at frequency f of a material with

% resistivity rho.

%% Constants

u\_0 = 4\*pi\*10^-7;

%% Equations

dskin = sqrt(resitivity./(frequency.\*pi.\*u\_0));

end

**rhelical.m**

function [ wire\_resis, wire\_length ] = rhelical( coil\_height, wire\_thickness, pitch, resistivity, varargin )

%rhelical Resistace of a helical coil made from a wire with constant pitch.

% Resistance of a helical coil of length l, made from a wire turned with

% a constant pitch on a mandral with constant radius r or varying radius

% r1, r2.

if length(varargin) == 1

length\_pre = coil\_height./pitch;

length\_coil = length\_pre.\*sqrt((2.\*pi.\*varargin{1}/2).^2 + pitch.^2);

else

r1 = varargin{0};

r2 = varargin{1};

alpha = (r2-r1)./coil\_height;

c\_pre = (pitch./(2\*pi)).^2;

c = c\_pre.\*(1+tan(alpha)^2);

a = sqrt((r2.^2)+(c.^2));

b = sqrt((r1.^2)+(c.^2));

length\_pre = pi./(pitch.\*tan(alpha));

length\_coil = length\_pre.\*((a.\*r2)- (b.\*r1)+c.\*log((a.\*r2)./(b.\*r1)));

end

wire\_resis = rwire(resistivity,length\_coil,wire\_thickness/2);

wire\_length = length\_coil;

end

**f.m**

function [ f ] = f( gamma )

%ASINGAMMA Function expressing the t expression of gamma.

% Detailed explanation goes here

f\_pre = 1./sqrt(1-(gamma.^2));

f = f\_pre.\*asin(sqrt(1-(gamma.^2)));

end

**g.m**

function [ g ] = g( gamma )

%EXPGAMMA Function expressing the I expression of gamma.

% Detailed explanation goes here

g\_pre = (-1.\*gamma)./sqrt(1-(gamma.^2));

g = exp(g\_pre.\*asin(sqrt(1-(gamma.^2))));

end

**ctrans.m**

function [ ctrans ] = ctrans( outer\_radius, inner\_radius, varargin )

%UNTITLED Summary of this function goes here

% Detailed explanation goes here

%% Constants

e\_0 = 8.854\*10^-12;

%% Equations

if length(varargin) > 0

e\_r = [varargin{1}];

area = [varargin{2}];

d1 = [varargin{3}];

d2 = [varargin{4}];

ctrans = (e\_r\*e\_0\*area)/(e\_0\*d1+e\_r\*d2);

else

ctrans = (2\*pi\*e\_0)/log(outer\_radius/inner\_radius);

end

end

**csparkgap.m**

function [ csparkgap ] = csparkgap( sphere\_radius, spark\_gap, N )

%CSPARKGAP Calculates spark gap capacitance

% Calculates self capacitance of a spark gap with sphere radius 'R'

% and spark gap length 'gap'.

%

% N is an empirical value.

%% Constants

E0 = 8.85\*10^-12;

%% Equation

csparkgap\_pre = pi.\*E0.\*sqrt((((2\*sphere\_radius)+spark\_gap)^2)-(4\*(sphere\_radius^2)));

csparkgap\_end = zeros(N,1);

for j=0:1:N

csparkgap\_end(j+1) = coth((j+0.5).\*acosh(((2.\*sphere\_radius)+spark\_gap)./(2.\*sphere\_radius)))-1;

end

csparkgap=csparkgap\_pre.\*sum(csparkgap\_end);

end