High Voltage Project

Voltage transients and lattice diagram

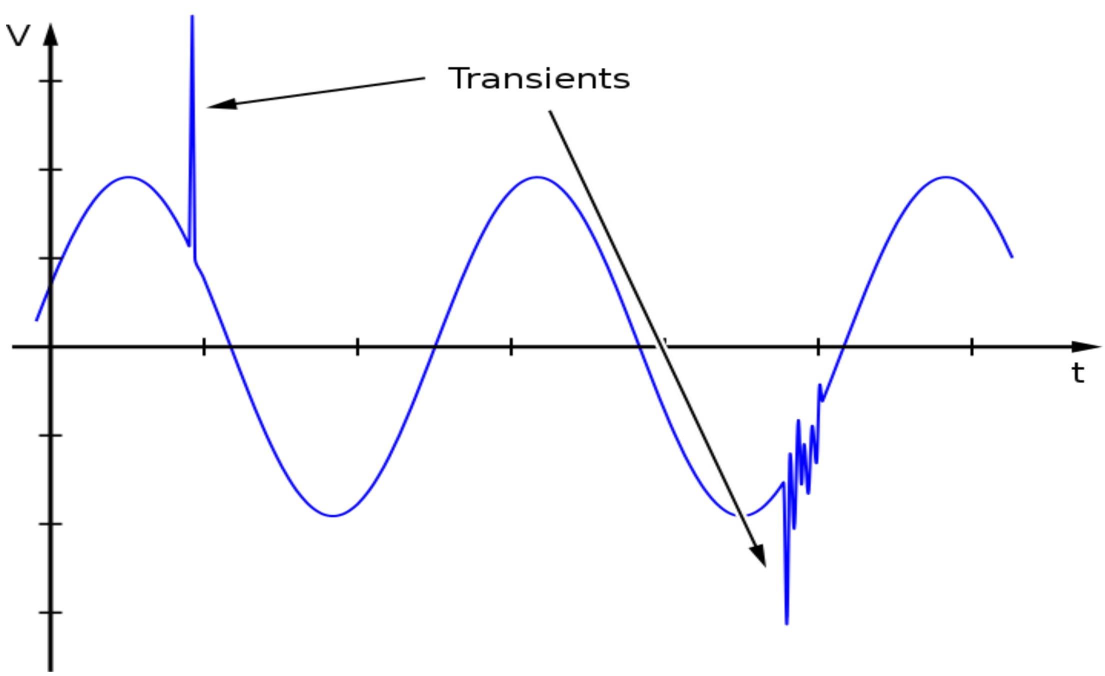
Index

1 Introduction

Voltage Transients are defined as short duration surges of electrical energy and are the result of the sudden release of energy previously stored or induced by other means, such as heavy inductive loads or lightning. In electrical or electronic circuits, this energy can be released in a predictable manner via controlled switching actions, or randomly induced into a circuit from external sources.

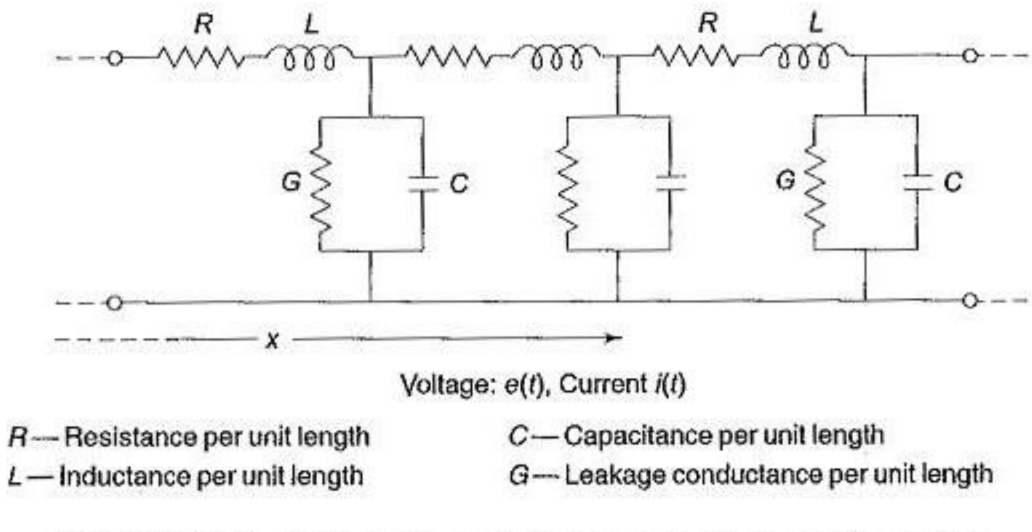
1.1 Travelling waves

Traveling Waves On Transmission Lines. Travelling waves are the current and voltage waves which travel from the sending end of a transmission line to the other end. When the switch is closed at the transmission line's starting end, voltage will not appear instantaneously at the other end. This is caused by the transient behavior of inductor and capacitors that are present in the transmission line. The transmission lines may not have physical inductor and capacitor elements but the effects of inductance and capacitance exists in a line. Therefore, when the switch is closed the voltage will build up gradually over the line conductors. This phenomenon is usually called as the voltage wave is travelling from transmission line's sending end to the other end. And similarly the gradual charging of the capacitances happens due to the associated current wave.



1.2 Transient propagation

- **Characteristics impedance:** Each Transmission line consists of the following components per unit length.
  - **Resistance (R)**
  - **Inductance (L)**
  - **Capacitance (C)**
  - **Conductance (G)**



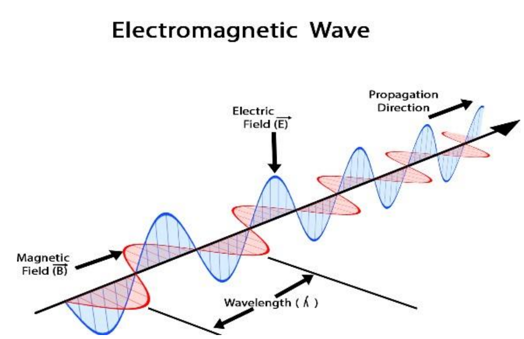
- **Propagation speed:** The propagation velocity of the wave can be obtained by the following.

$$v = \frac{c}{\sqrt{\mu r * \varepsilon r}}$$

  Ideally, propagating wave travels at speed of light in air insulated lossless cable.

  Actually, propagation speed decreases due to line losses like corona .

- **Energy transfer:** The energy is stored in the electromagnetic field of the propagating wave. Half in the electric field and half in the magnetic field.



1.3 Transient attenuation

Transient can be attenuated through:

line losses: as it decreases the wave's amplitude and front.

Corona effect: as it increases the effective diameter of the conductor hence, decreasing the wave impedance.

Skin effect: Due to the current density near the surface of the conductor is greater than at its core, the effective resistance of the conductor increases with the frequency of the current.

Earthing conditions: The penetration depth and the propagation speed depend on the frequency of the propagated wave.

1.4 Transient reflection

When the traveling wave reaches a discontinuity region part of it is transmitted and the other part is reflected

As **Transmitted wave = incident wave + reflected wave**

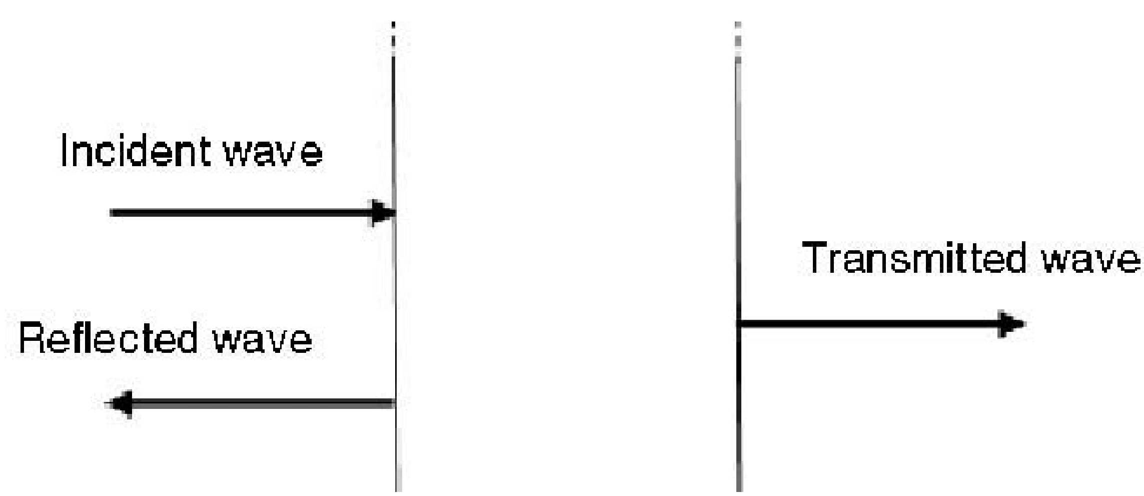
For the transmitted and reflected voltage it can be calculated as following:

Transmitted: $E2 = \boldsymbol{\tau}\, E1$

Reflected: $E2 = \rho\, E1$

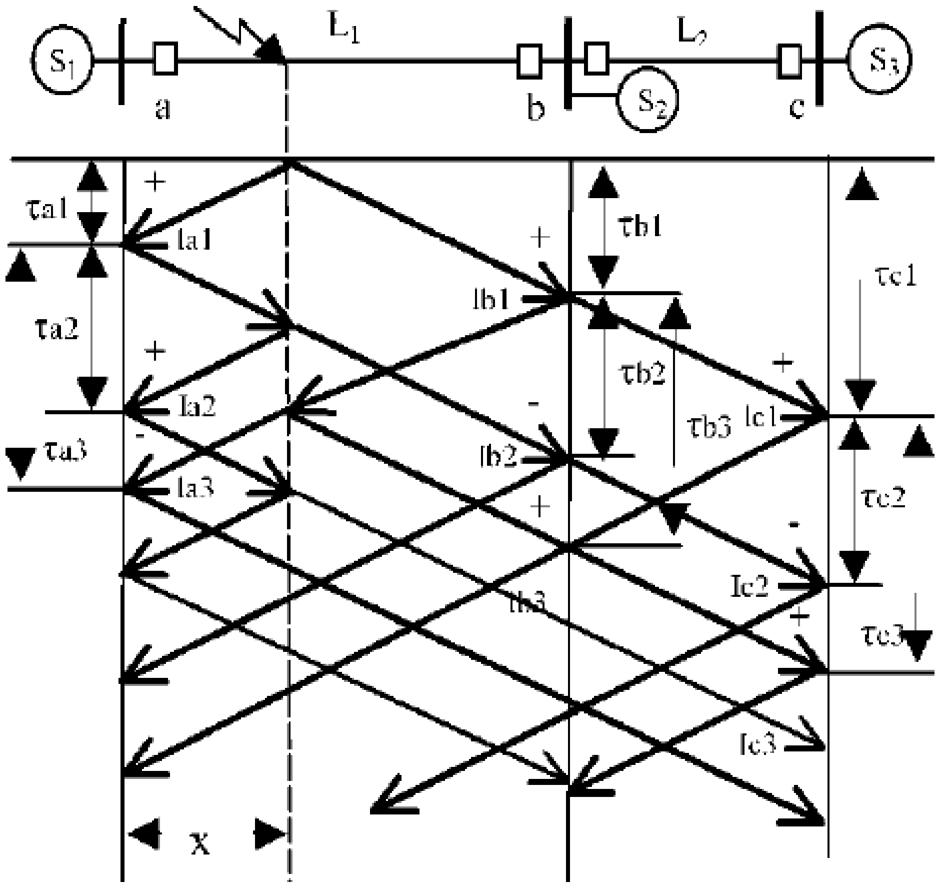
Transmission coefficient (τ) = $\frac{2Z2}{Z1+Z2}$

Reflection coefficient (ρ) = $\frac{Z2-Z1}{Z1+Z2}$ = τ -1



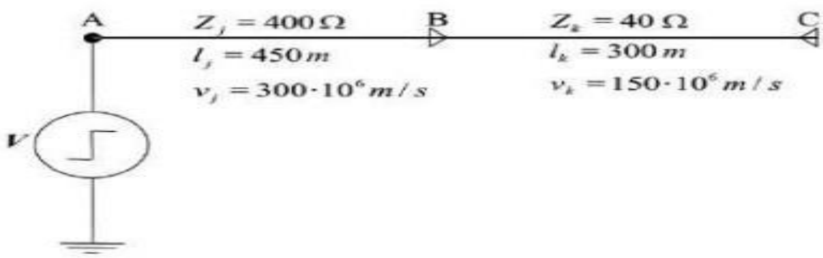
1.5-Bewely’s lattice diagram

Bewley's lattice diagram is a graphical method that has been widely used for determining the value of the travelling wave in transient analysis.



2 MATLAB code and Results

2.1 Part 1: For the given example



Code:

Code consists of 2 scripts the first is called tree which represents tree diagram structure in MATLAB and the second is main script.

```
classdef tree
    properties (SetAccess = private)
        Node = { [] };
        Parent = [ 0 ];
    end
    methods
        function [obj, root_ID] = tree(content, val)
            if nargin < 1
                root_ID = 1;
                return
            end
            if isa(content, 'tree')
                obj.Parent = content.Parent;
                if nargin > 1
                    if strcmpi(val, 'clear')
                        obj.Node = cell(numel(obj.Parent), 1);
                    else
                        cellval = cell(numel(obj.Parent), 1);
                        for i = 1 : numel(obj.Parent)
                            cellval{i} = val;
                        end
                        obj.Node = cellval;
                    end
                else
                    obj.Node = content.Node;
                end
            else
                obj.Node = { content };
                root_ID = 1;
            end
        end
```

```
        function [obj, ID] = addnode(obj, parent, data)
            if parent < 0 || parent > numel(obj.Parent)
                error('MATLAB:tree:addnode', ...
                    'Cannot add to unknown parent with index %d.\n', parent)
            end
            if parent == 0
                obj.Node = { data };
                obj.Parent = 0;
                ID = 1;
                return
            end
            obj.Node{ end + 1, 1 } = data;

            obj.Parent = [
                obj.Parent
                parent ];
            ID = numel(obj.Node);
        end
    end
end
```

Part 1:

by replacing the first 22 line in general code with the following piece of code or simply enter the desired parameters you get part 1

```
1 clear all ;
2 close all ;
3 %inputing the parametes of the system
4 General_Parameters= inputdlg ({'No#Stages','Incidence Voltage(kV)',...
5      'Impedense(ohm)on this form [z1,z2...,zn]',...
6      'Distance(m)on this form [d1,d2...,dn]',...
7      'Propagation Velocities(*10^6 m/s)on this form [v1,v2...,vn]'},...
8      'Parameters',[1 60]);
9 No_stages=str2num(General_Parameters{1});
10% n represents the number of junctions
11n=No_stages+1;
12% Input_String= Origin , Start Time , Value , End Time , Destination
13v_incidence=str2num(General_Parameters{2});
14Input_String=[0 0 v_incidence 0 1];
15% Forming both tau and rho matrices using line impedences
16impedance=str2num(General_Parameters{3});
17z=[0,impedance,-1];
18% -1 represents infinty because matlab can't deal with it
19% calculating the times between each two following junctions
20%using the propagation velocities and inner distances
21propegation_velocities=str2num(General_Parameters{5});
22inner_distances=str2num(General_Parameters{4});
```

General Code:

```
clear all ;
close all ;
%inputing the parametes of the system
General_Parameters= inputdlg ({'No#Stages','Incidence Voltage(kV)',...
    'Impedense(ohm)on this form [z1,z2...,zn]',...
    'Distance(m)on this form [d1,d2...,dn]',...
    'Propagation Velocities(*10^6 m/s)on this form [v1,v2...,vn]'},...
    'Parameters',[1 60]);
No_stages=str2num(General_Parameters{1});
% n represents the number of junctions
n=No_stages+1;
% Input_String= Origin , Start Time , Value , End Time , Destination
v_incidence=str2num(General_Parameters{2});
Input_String=[0 0 v_incidence 0 1];
% Forming both tau and rho matrices using line impedences
impedance=str2num(General_Parameters{3});
z=[0,impedance,-1];
% -1 represents infinty because matlab can't deal with it
% calculating the times between each two following junctions
%using the propagation velocities and inner distances
propegation_velocities=str2num(General_Parameters{5});
inner_distances=str2num(General_Parameters{4});
time_skiped=inner_distances./propegation_velocities;
inner_intervals=[time_skiped(1),time_skiped,time_skiped(end)];
tauv=zeros(2,length(z)-1);
for i=1:length(z)-1
  % special case when z = infinity the value of tau is always=2
 if i==  length(z)-1
 if z(i+1)<0
     tauv(1,i)=2;
     tauv(2,i)=0;
  % if the last segment wasnot infinity it would be calcuted as usaul
   else
     tauv(1,i)=2*z(i+1)/(z(i)+z(i+1)) ;
     tauv(2,i)=2*z(i)/(z(i)+z(i+1)) ;
   end
  %an other special case because the first segement is always regarded
  %as short circuit so it would always be assigned to these values
 elseif i== 1
     tauv(1,i)=1;
     tauv(2,i)=0;
 % the inner segments fllow the rules normally
 else
  for j=1:2
   if j==1
      tauv(1,i)=2*z(i+1)/(z(i)+z(i+1)) ;
   else
```

```
        tauv(2,i)=2*z(i)/(z(i)+z(i+1));
     end
   end
 end
end
rhov=tauv-1;
rhoi=-rhov;
taui=rhoi+1;
% it is noticed that each arrow is devided into transimitted and reflected
%so each arrow has tw children and so on so each child has two children
% this can be represented only b tree diagram
t=tree(Input_String);
New_Signal =zeros(1,5) ;
transmited_indeces=[];
for i = 1:400
    % accessing the parent before each loop to generate its two childs
    Input_Signal=t.get(i);
    Origin=Input_Signal(1);
    Start_Time=Input_Signal(2);
    Value=Input_Signal(3);
    End_Time=Input_Signal(4);
    Destination=Input_Signal(5);
    %special case if the arrow goes to the end or to the start it has no
    %children
    if Destination==0 || Origin==n+1 || Destination==n+1
      continue;
    else
      % the destination of the parent is the origin of the child and
      % the end time of the parent is the strt time of the child
      New_Signal(1)=Destination;
      New_Signal(2)=End_Time;
      % ading the time between each two junctions based on the times
      % calcated before in general form
      for k=1:n+1
       if (New_Signal(1)==k+1 && New_Signal(5)==k) ||...
           (New_Signal(1)==k&&New_Signal(5)==k+1)
         New_Signal(4)=New_Signal(2)+inner_intervals(k);
       end
      end
      % Generating the transmitted and refelcted signals in case the arrow
      % was going from left to right
      if Destination > Origin
         % special case the whole tree parent
         if Origin==0
           % generating the transmitted child
           New_Signal(3)=Value*tauv(1,1);
           New_Signal(5)=New_Signal(1)+1;
           New_Signal(4)=New_Signal(2)+inner_intervals(1);
           t=t.addnode(i,New_Signal);
           transmited_indeces(1)=1;
           transmited_indeces(2)=2;
```

```
         %generating the reflected child
         New_Signal(3)=Value*rhov(1,1);
         New_Signal(5)=New_Signal(1)-1;
         New_Signal(4)=New_Signal(2)+inner_intervals(1);
         t=t.addnode(i,New_Signal);
      % General Case
      else
        % generating transmitted arrow
        New_Signal(5)=New_Signal(1)+1;
        New_Signal(3)=Value*tauv(1,Destination);
        for k=0:n+1
         if (New_Signal(1)==k+1 && New_Signal(5)==k) ||...
            (New_Signal(1)==k&&New_Signal(5)==k+1)
           New_Signal(4)=New_Signal(2)+inner_intervals(k+1);
         end
        end
        t=t.addnode(i,New_Signal);
        transmited_indeces=[transmited_indeces length(t.Node)];
        %generating reflected arrow
        New_Signal(5)=New_Signal(1)-1;
        New_Signal(3)=Value*rhov(1,Destination);
        for k=0:n+1
         if (New_Signal(1)==k+1 && New_Signal(5)==k) ||...
            (New_Signal(1)==k&&New_Signal(5)==k+1)
          New_Signal(4)=New_Signal(2)+inner_intervals(k+1);
         end
        end
        t=t.addnode(i,New_Signal);
       end
    % if the arrw was going from right to left
    else
        New_Signal(5)=New_Signal(1)-1;
        New_Signal(3)=Value*tauv(2,Destination);
        for k=0:n+1
         if (New_Signal(1)==k+1 && New_Signal(5)==k) ||...
            (New_Signal(1)==k&&New_Signal(5)==k+1)
           New_Signal(4)=New_Signal(2)+inner_intervals(k+1);
         end
        end
        t=t.addnode(i,New_Signal);
        transmited_indeces=[transmited_indeces length(t.Node)];
        New_Signal(5)=New_Signal(1)+1;
        New_Signal(3)=Value*rhov(2,Destination);
        for k=0:4
         if (New_Signal(1)==k+1 && New_Signal(5)==k) ||...
            (New_Signal(1)==k&&New_Signal(5)==k+1)
            New_Signal(4)=New_Signal(2)+inner_intervals(k+1);
         end
        end
        t=t.addnode(i,New_Signal);
    end
  end
end
```

```
plotting_points=zeros(1,5);
% it is needed to know both the start time and value of each transmitted arrow
for ij=1:length (transmited_indeces)
   plotting_points=t.get(transmited_indeces(ij));
   for ji=1:n
      % we added zero in the first element to trigger the formation of the
      % general cell
       if ij==1
           A{ji}(1)= 0;
           A{ji+n}(1)= 0;
       else
           if ji==plotting_points(1)
            A{ji}(end+1)= plotting_points(3);
            A{ji+n}(end+1)= plotting_points(2);
           else
             continue;
           end
       end
   end
 end
% it is also needed to sort the times ascendly (from small to higher)
%then find the simillar time elements sum them up and remove the
%duplicated elements which hve the same end time
for in=1:n
   % sort the times ascendly
   [A{in+n},I]=sort(A{in+n},'ascend');
   A{in} = A{in}(I);
   %find the simillar time elements
   [v, w] = unique( A{in+n}, 'stable' );
   duplicate_indices{in} = setdiff( 1:numel(A{in+n}), w );
   % sum the similar end time elements up
   for im=1:length(duplicate_indices{in})
        if im==1
          original_value{in}(im)=duplicate_indices{in}(im)-1;
        else
            if duplicate_indices{in}(im)-duplicate_indices{in}(im-1)==1
                original_value{in}(im)=original_value{in}(im-1);
            else
                original_value{in}(im)=duplicate_indices{in}(im)-1;
            end
        end
    end
    for iq=1:length(original_value)
A{in}(original_value{in}(iq))=A{in}(original_value{in}(iq))+A{in}(duplicate_indices{in}(iq));
    end
    %removing the duplicates
    A{in}(duplicate_indices{in})=[];
    A{in+n}(duplicate_indices{in})=[];
```

```
    %summing the transmitted voltage to get the voltage at each junction
    for ix=2:length(A{in})
     A{in}(ix)= A{in}(ix)+ A{in}(ix-1);
    end
end
%plotting voltage at each junction and the whole system lattice diagram
figure('Renderer', 'painters', 'Position', [250 60 900 600])
colors=['k','r','g','b','m','c','y'];
tab{1}=uitab('Title','Voltage Lattice Diagram');
% when the n>6 the lattice diagram becomes very un clear so it is recommended
% to use small system to see a clear looking lattice diagram
if n<=6
    ax{1} = axes(tab{1});
    lattice_plotting_points=zeros(1,5);
% sorting the arrows based on thier regoins so that they are given
% different colors in the graph for more clarity
    for iy=1:length(t.Node)
          lattice_plotting_points=t.get(iy);
              for io= 0:n
               if lattice_plotting_points(3)~=0
                   if (lattice_plotting_points(1)==io && lattice_plotting_points(5)==io+1)||...
                       (lattice_plotting_points(5)==io && lattice_plotting_points(1)==io+1)
                       B{io+1}(1,iy)=lattice_plotting_points(1);
                       B{io+1}(2,iy)=lattice_plotting_points(5);
                       B{io+2+n}(1,iy)=lattice_plotting_points(2);
                       B{io+2+n}(2,iy)=lattice_plotting_points(4);
                   end
               end
              end
     end
end
ylim([-1 20]);
xlim([0.5 n+0.5]);
% reversing the direction of y axis to let start from up to down
set(ax{1}, 'YDir','reverse');
for il=1:n+1
    if il==n+1
      line(ax{1},B{il},B{il+n+1},'LineWidth',0.5,'Color','k');
    else
line(ax{1},B{il},B{il+n+1},'LineWidth',0.5,'Color',colors(il));
    end
end
hold on
line(ax{1},B{1}(:,1),B{1+n+1}(:,1),'LineWidth',2,'Color',colors(2));
% plotting the vertical lines which represents each junction in the
% lattice diagram
for iu=1:n
 line([iu,iu], ylim, 'Color', 'k', 'LineWidth', 1.2);
end
```

```
grid(ax{1},'on') ;
grid(ax{1},'minor') ;
ylabel('t(us)');
if n>7
 colors=[colors colors];
end
%plotting the voltage at each junction
for iz=2:n+1
     tab_name=strcat('V' , num2str(iz-1));
     tab{iz}=uitab('Title',tab_name);
     ax{iz} = axes(tab{iz});
     stairs (ax{iz},A{(iz-1)+n},A{(iz-1)},'LineWidth',2,'Color',colors((iz)));
     grid(ax{iz}, 'on') ;
     grid(ax{iz}, 'minor') ;
     xlabel('t(us)');
     ylabel('V(kV)');
     xlim([0 20]);
 end
% the same flow for the voltage is used to get the current lattice diagram
% and the current transmitted at each junction
Input_String_i=[0 0 v_incidence/z(2) 0 1];
tt=tree(Input_String_i);
New_Signal_i=zeros(1,5) ;
transmited_indeces_i=[];
for i = 1:400
   % accessing the parent before each loop to generate its two childs
   Input_Signal=tt.get(i);
   Origin=Input_Signal(1);
   Start_Time=Input_Signal(2);
   Value=Input_Signal(3);
   End_Time=Input_Signal(4);
   Destination=Input_Signal(5);
   %special case if the arrow goes to the end or to the start it has no
   %children
   if Destination==0 || Origin==n+1 || Destination==n+1
     continue;
   else
     % the destination of the parent is the origin of the child and
     % the end time of the parent is the strt time of the child
     New_Signal_i(1)=Destination;
     New_Signal_i(2)=End_Time;
     % ading the time between each two junctions based on the times
     % calcated before in general form
     for k=1:n+1
      if (New_Signal_i(1)==k+1 && New_Signal_i(5)==k) ||...
         (New_Signal_i(1)==k&&New_Signal_i(5)==k+1)
        New_Signal_i(4)=New_Signal_i(2)+inner_intervals(k);
      end
     end
```

```
% Generating the transmitted and refelcted signals in case the arrow
    % was going from left to right
    if Destination > Origin
        % special case the whole tree parent
        if Origin==0
          % generating the transmitted child
          New_Signal_i(3)=Value*taui(1,1);
          New_Signal_i(5)=New_Signal_i(1)+1;
          New_Signal_i(4)=New_Signal_i(2)+inner_intervals(1);
          tt=tt.addnode(i,New_Signal_i);
          transmited_indeces_i(1)=1;
          transmited_indeces_i(2)=2;
          %generating the reflected child
          New_Signal_i(3)=Value*rhoi(1,1);
          New_Signal_i(5)=New_Signal_i(1)-1;
          New_Signal_i(4)=New_Signal_i(2)+inner_intervals(1);
          tt=tt.addnode(i,New_Signal_i);
       % General Case
       else
         % generating transmitted arrow
         New_Signal_i(5)=New_Signal_i(1)+1;
         New_Signal_i(3)=Value*taui(1,Destination);
         for k=0:n+1
          if (New_Signal_i(1)==k+1 && New_Signal_i(5)==k) ||...
              (New_Signal_i(1)==k&&New_Signal_i(5)==k+1)
            New_Signal_i(4)=New_Signal_i(2)+inner_intervals(k+1);
          end
         end
         tt=tt.addnode(i,New_Signal_i);
         transmited_indeces_i=[transmited_indeces_i
length(tt.Node)];
         %generating reflected arrow
         New_Signal_i(5)=New_Signal_i(1)-1;
         New_Signal_i(3)=Value*rhoi(1,Destination);
         for k=0:n+1
          if (New_Signal_i(1)==k+1 && New_Signal_i(5)==k) ||...
              (New_Signal_i(1)==k&&New_Signal_i(5)==k+1)
           New_Signal_i(4)=New_Signal_i(2)+inner_intervals(k+1);
          end
         end
         tt=tt.addnode(i,New_Signal_i);
        end
    % if the arrw was going from right to left
    else
         New_Signal_i(5)=New_Signal_i(1)-1;
         New_Signal_i(3)=Value*taui(2,Destination);
         for k=0:n+1
          if (New_Signal_i(1)==k+1 && New_Signal_i(5)==k) ||...
              (New_Signal_i(1)==k&&New_Signal_i(5)==k+1)
            New_Signal_i(4)=New_Signal_i(2)+inner_intervals(k+1);
          end
         end
```

```
tt=tt.addnode(i,New_Signal_i);
            transmited_indeces_i=[transmited_indeces_i length(tt.Node)];
            New_Signal_i(5)=New_Signal_i(1)+1;
            New_Signal_i(3)=Value*rhoi(2,Destination);
            for k=0:4
             if (New_Signal_i(1)==k+1 && New_Signal_i(5)==k) ||...
                (New_Signal_i(1)==k&&New_Signal_i(5)==k+1)
                New_Signal_i(4)=New_Signal_i(2)+inner_intervals(k+1);
             end
            end
            tt=tt.addnode(i,New_Signal_i);
        end
    end
end
plotting_points_i=zeros(1,5);
% it is needed to know both the start time and value of each arrow
for ij=1:length (transmited_indeces_i)
    plotting_points_i=tt.get(transmited_indeces_i(ij));
    for ji=1:n
         % we added zero in the first element to trigger the formation of the
         % general cell
        if ij==1
            C{ji}(1)= 0;
            C{ji+n}(1)= 0;
        else
            if ji==plotting_points_i(1)
             C{ji}(end+1)= plotting_points_i(3);
             C{ji+n}(end+1)= plotting_points_i(2);
            else
              continue;
            end
        end
    end
end
% it is also needed to sort the times ascendly (from small to higher)
for in=1:n
    [C{in+n},I]=sort(C{in+n},'ascend');
    C{in} = C{in}(I);
    [v1, wi] = unique( C{in+n}, 'stable' );
    duplicate_indices_i{in} = setdiff( 1:numel(C{in+n}), wi );
    for im=1:length(duplicate_indices_i{in})
          if im==1
            original_value_i{in}(im)=duplicate_indices_i{in}(im)-1;
          else
              if duplicate_indices_i{in}(im)-duplicate_indices_i{in}(im-1)==1
                    original_value_i{in}(im)=original_value_i{in}(im-1);
              else
original_value_i{in}(im)=duplicate_indices_i{in}(im)-1;
              end
          end
     end
```

```
for iq=1:length(original_value_i)
C{in}(original_value_i{in}(iq))=C{in}(original_value_i{in}(iq))+C{in}(duplicate_indices_i{in}(iq));
    end
    C{in}(duplicate_indices_i{in})=[];
    C{in+n}(duplicate_indices_i{in})=[];
    for ix=2:length(C{in})
     C{in}(ix)= C{in}(ix)+ C{in}(ix-1);
    end
 end
%figure('Renderer', 'painters', 'Position', [250 60 900 600])
colors=['k','r','g','b','m','c','y'];
tab_i{1}=uitab('Title','Current Lattice Diagram');
 if n<=6
    ax_i{1} = axes(tab_i{1});
    lattice_plotting_points_i=zeros(1,5);
    for iy=1:length(tt.Node)
         lattice_plotting_points_i=tt.get(iy);
             for io= 0:n-1
              if lattice_plotting_points_i(3)~=0
                   if (lattice_plotting_points_i(1)==io && lattice_plotting_points_i(5)==io+1)||...
                       (lattice_plotting_points_i(5)==io && lattice_plotting_points_i(1)==io+1)
                       D{io+1}(1,iy)=lattice_plotting_points_i(1);
                       D{io+1}(2,iy)=lattice_plotting_points_i(5);
                       D{io+2+n}(1,iy)=lattice_plotting_points_i(2);
                       D{io+2+n}(2,iy)=lattice_plotting_points_i(4);
                   end
              end
             end
     end
 end
ylim([-1 20]);
xlim([0.5 n+0.5]);
set(ax_i{1}, 'YDir','reverse');
for il=1:n
  line(ax_i{1},D{il},D{il+n+1},'LineWidth',0.5,'Color',colors(il));
 end
hold on
line(ax_i{1},D{1}(:,1),D{1+n+1}(:,1),'LineWidth',2,'Color',colors(2));
for iu=1:n
 line([iu,iu], ylim, 'Color', 'k', 'LineWidth', 1.2);
 end
grid(ax_i{1},'on') ;
grid(ax_i{1},'minor') ;
ylabel('t(us)');
if n>7
 colors=[colors colors];
 end
```

```
for iz=2:n+1
      tab_name=strcat('I' , num2str(iz-1));
      tab_i{iz}=uitab('Title',tab_name);
      ax_i{iz} = axes(tab_i{iz});
      stairs (ax_i{iz},C{(iz-1)+n},C{(iz-1)},'LineWidth',2,'Color',colors((iz)));
      grid(ax_i{iz}, 'on') ;
      grid(ax_i{iz}, 'minor') ;
      xlabel('t(us)');
      ylabel('I(kA)');
      xlim([0 20]);
 end
```

Results:

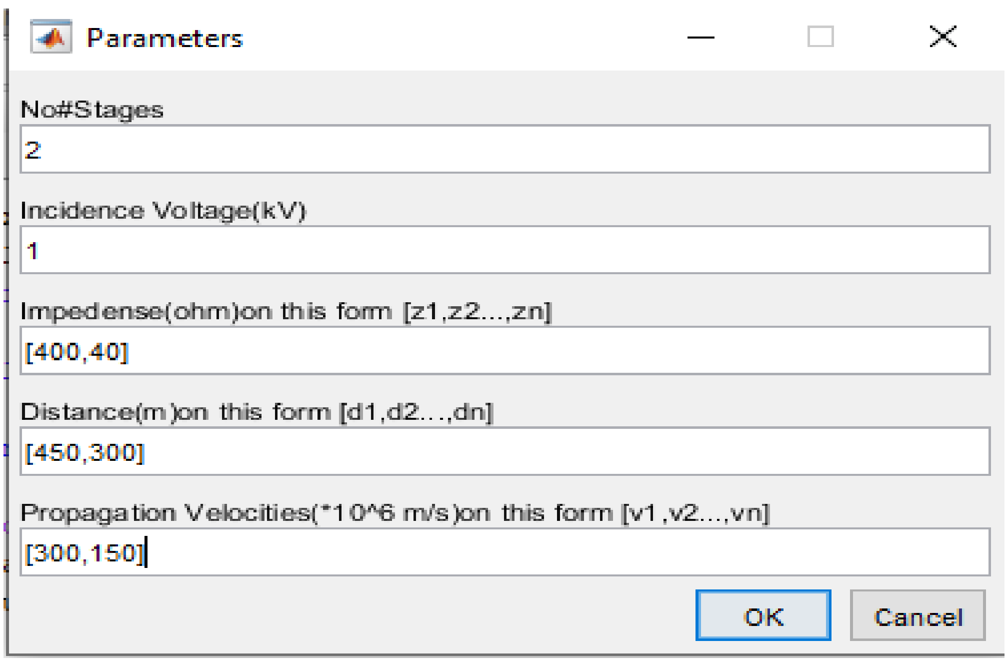
After running part 1 either by replacing the piece of code or using the general code you get

2.2 Part 2: generalized code

Results:

After running the code

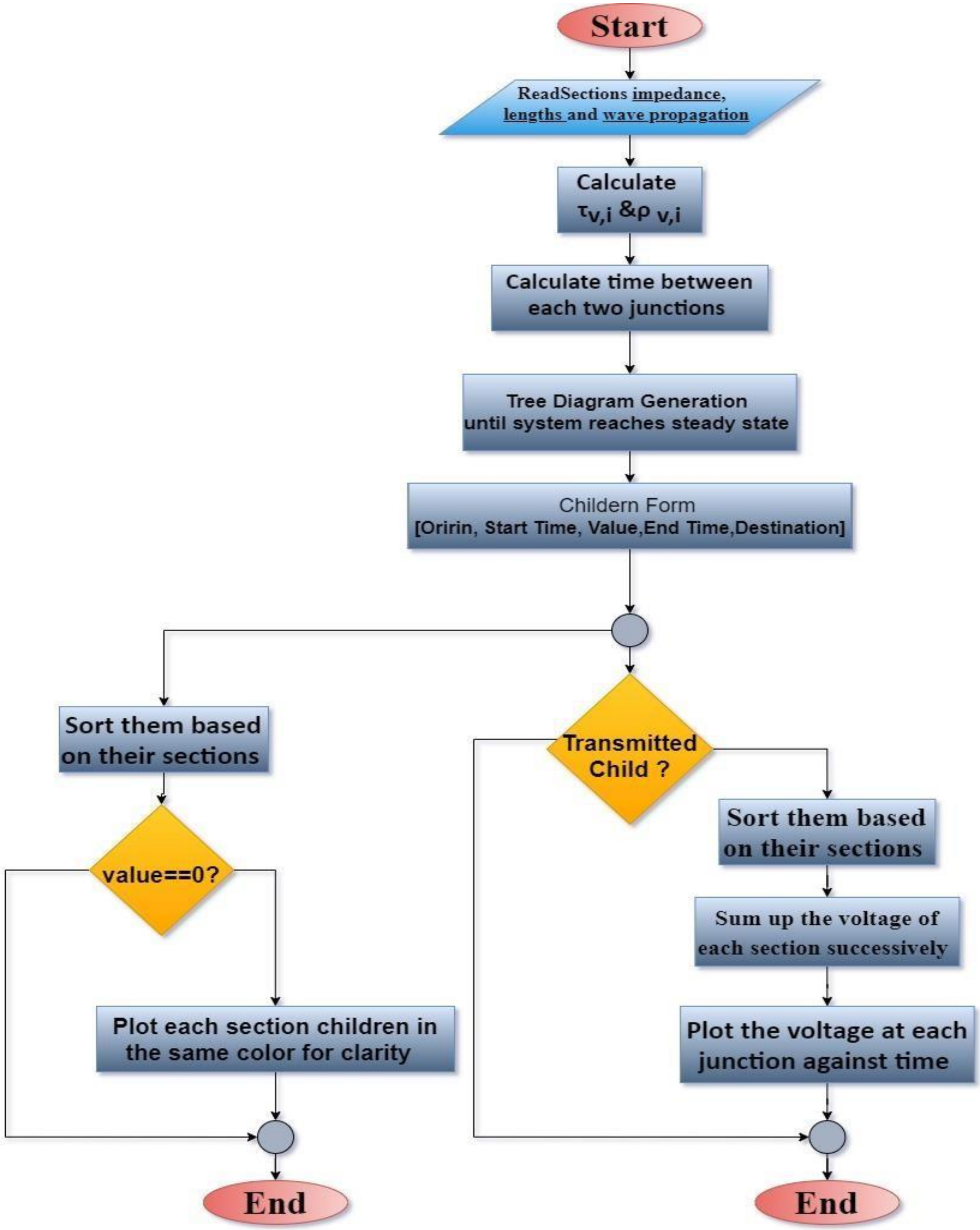
- Parameters inputting for general system GUI pops up as shown below



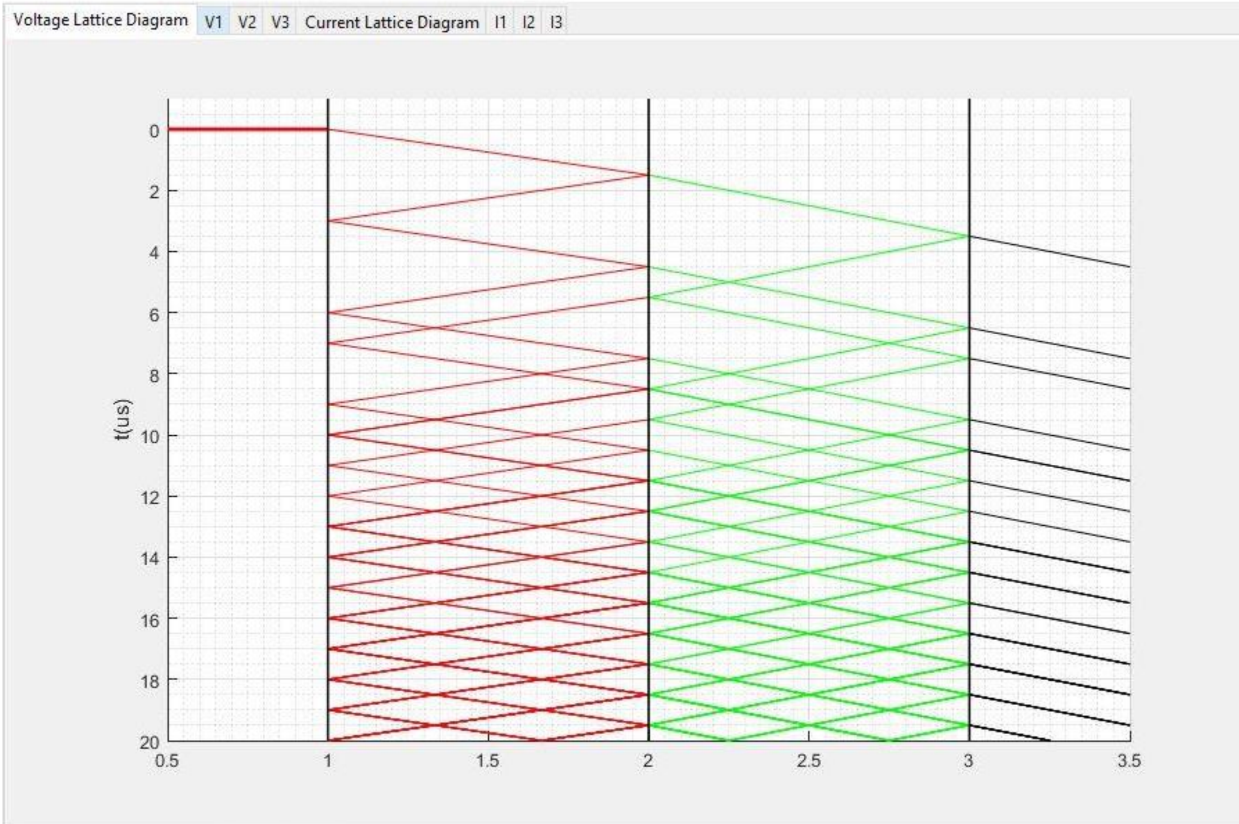
Graphs:

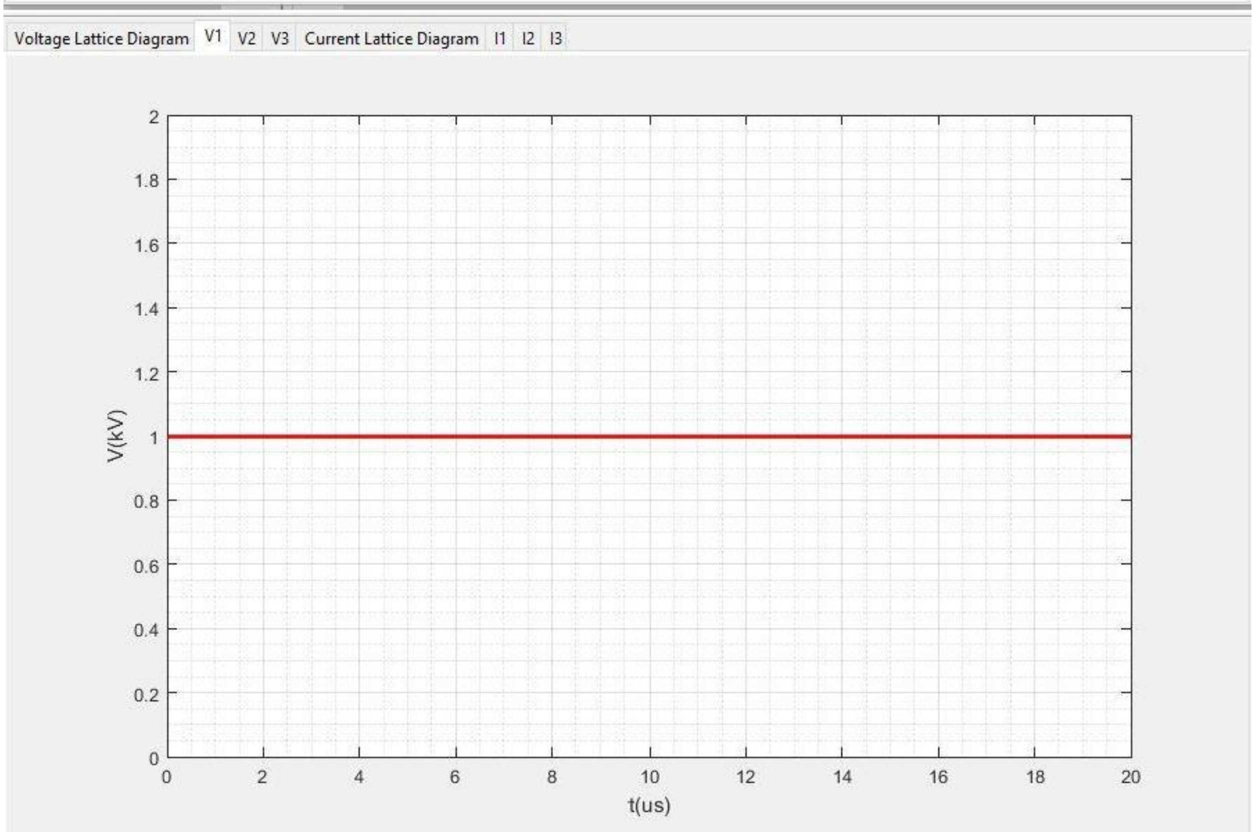
After inputting the parameters voltage and current lattice diagram and voltage and current at each junction Vs time are plotted.

3 Flow chart

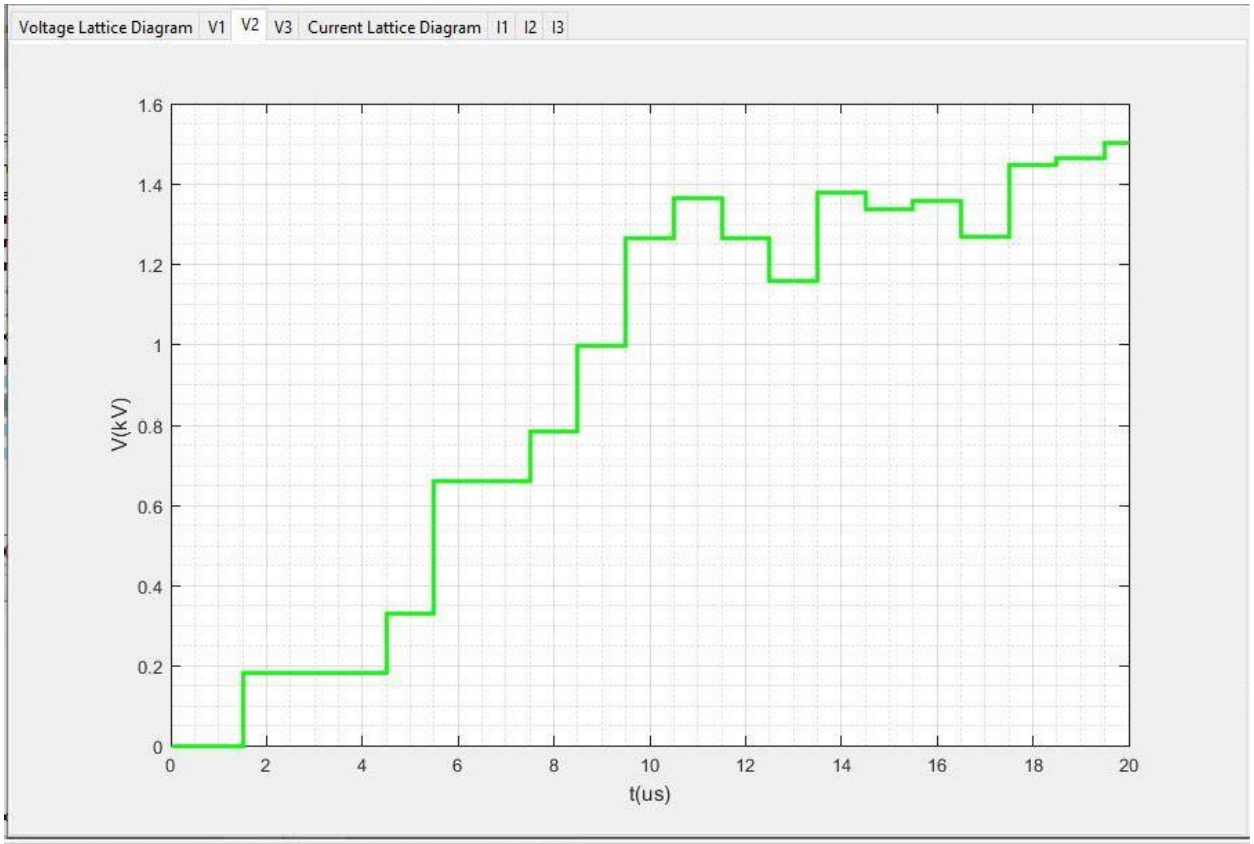

Start
ReadSections impedance, lengths and wave propagation
Calculate $\tau_{v,i}$ & $\rho_{v,i}$
Calculate time between each two junctions
Tree Diagram Generation until system reaches steady state
Childern Form [Oririn, Start Time, Value,End Time,Destination]
Sort them based on their sections
value==0?
Plot each section children in the same color for clarity
End
Transmitted Child ?
Sort them based on their sections
Sum up the voltage of each section successively
Plot the voltage at each junction against time
End

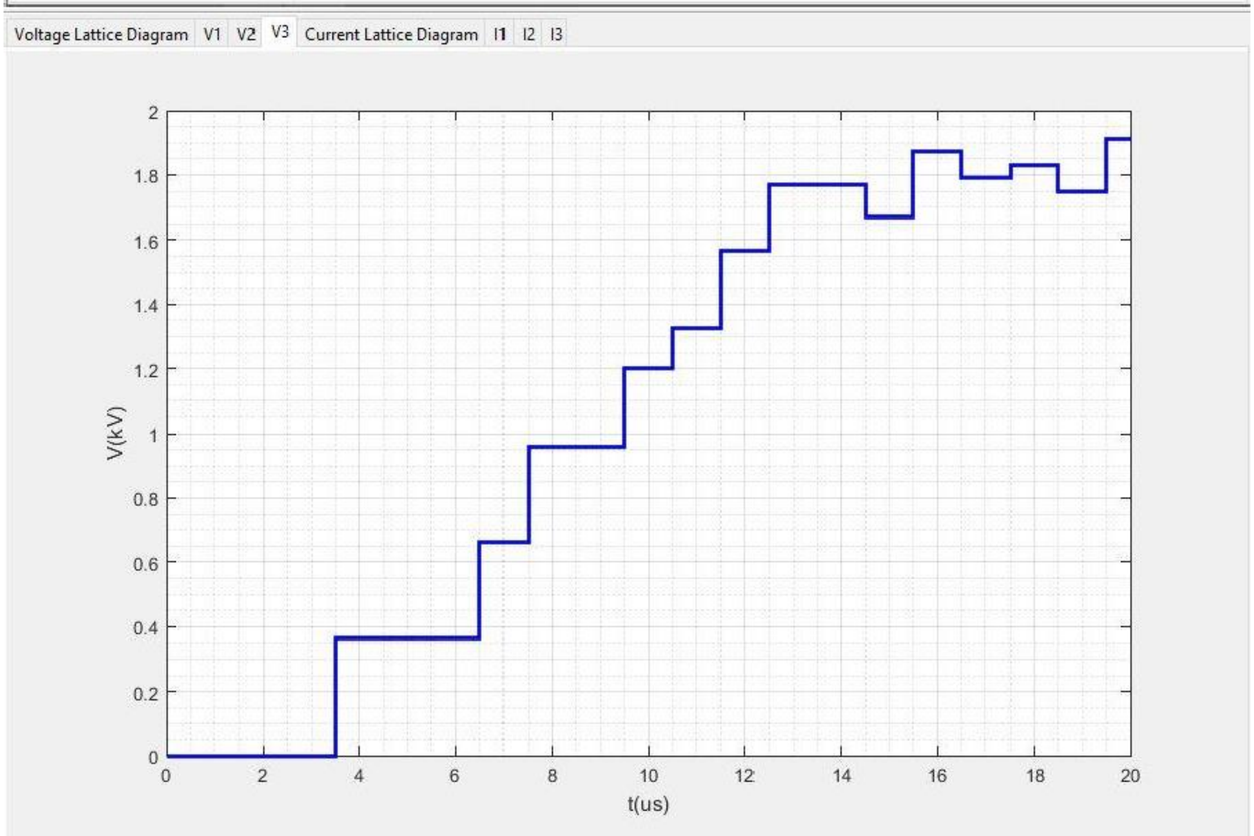
Part 1 Results:-

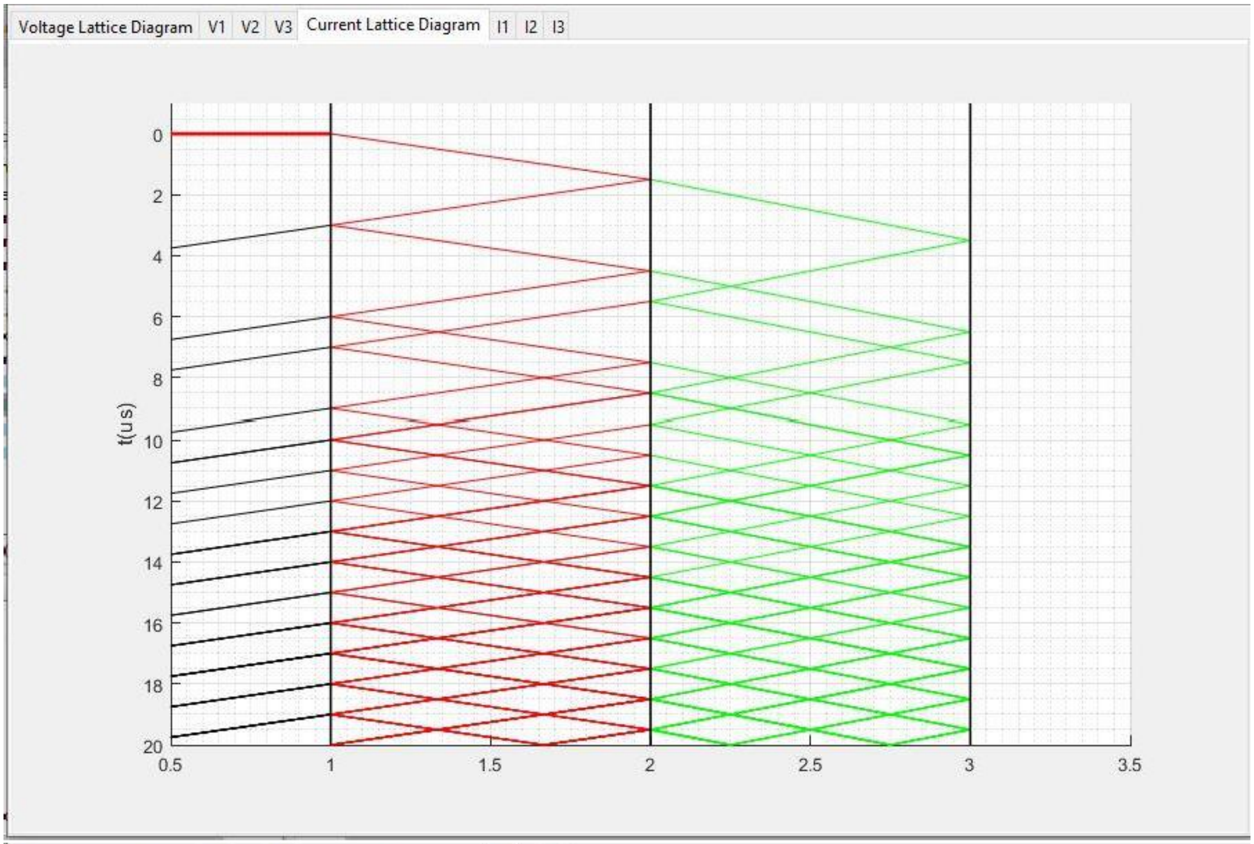




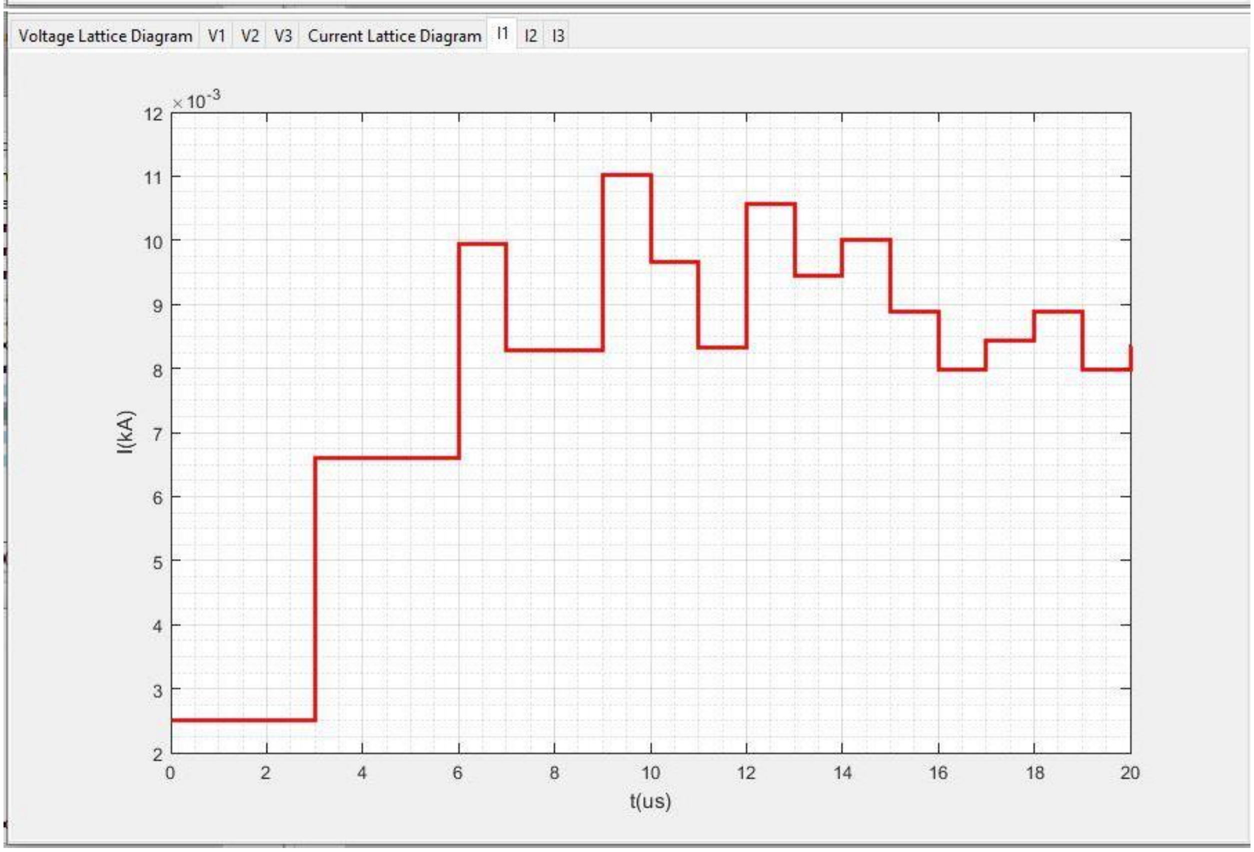
High Voltage Project



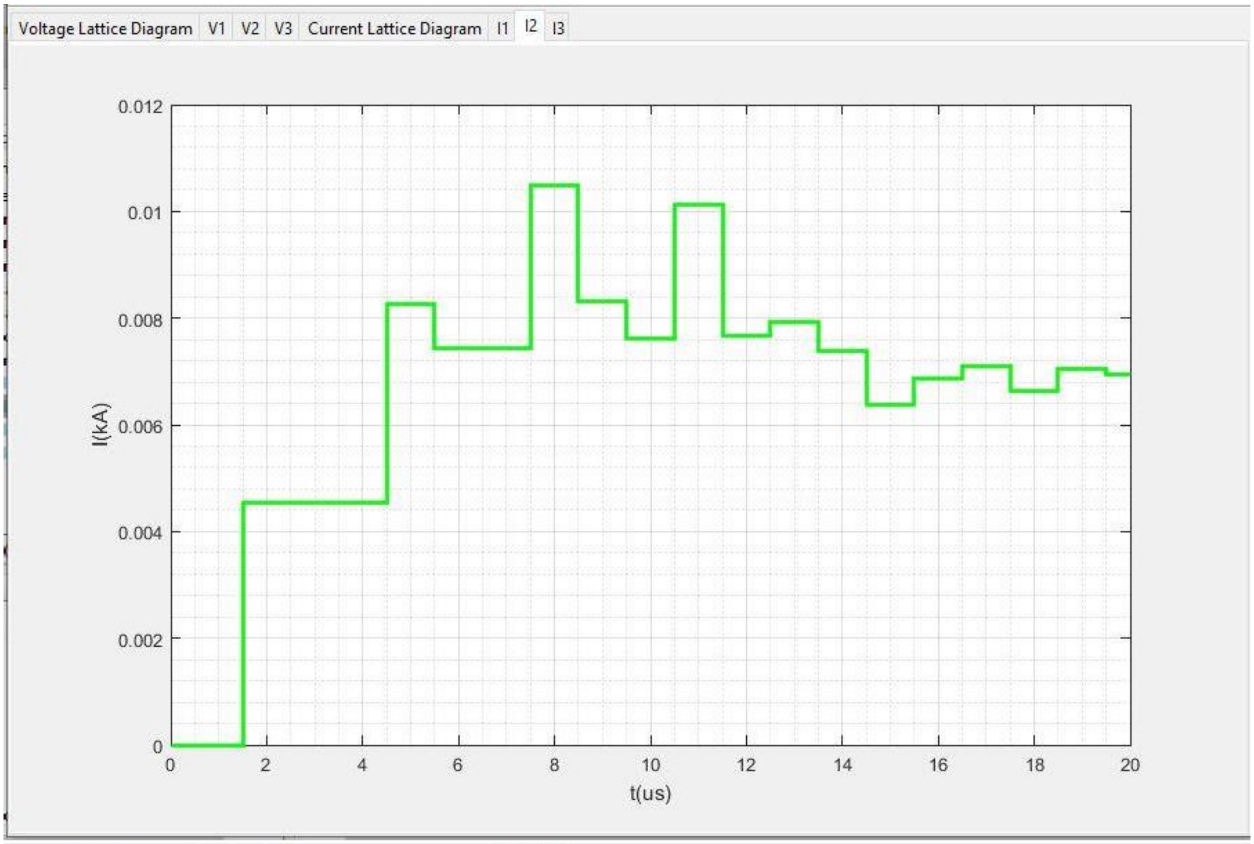




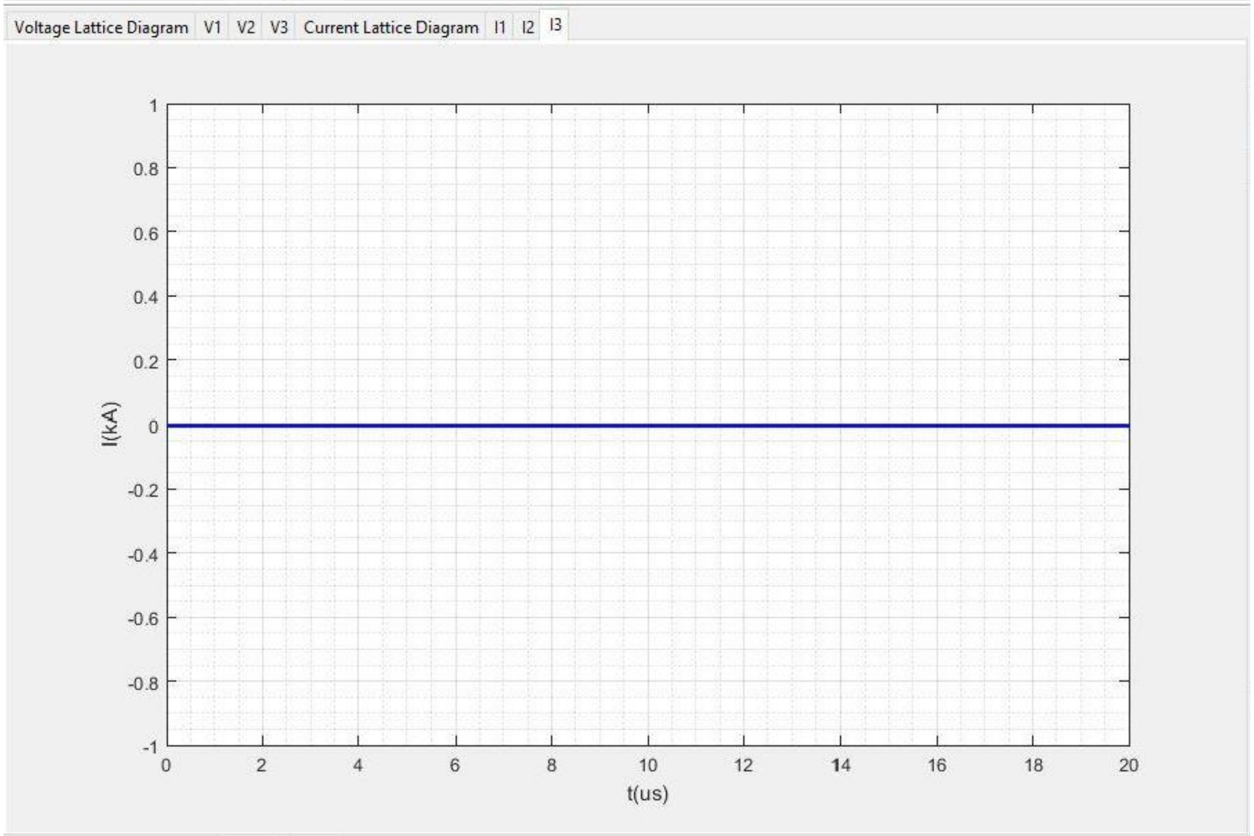
Voltage Lattice Diagram
V1
V2
V3
Current Lattice Diagram
I1
I2
I3
t(us)
0
2
4
6
8
10
12
14
16
18
20
0.5
1
1.5
2
2.5
3
3.5



Voltage Lattice Diagram
V1
V2
V3
Current Lattice Diagram
I1
I2
I3
× 10^-3
I(kA)
t(us)
12
11
10
9
8
7
6
5
4
3
2
0
2
4
6
8
10
12
14
16
18
20



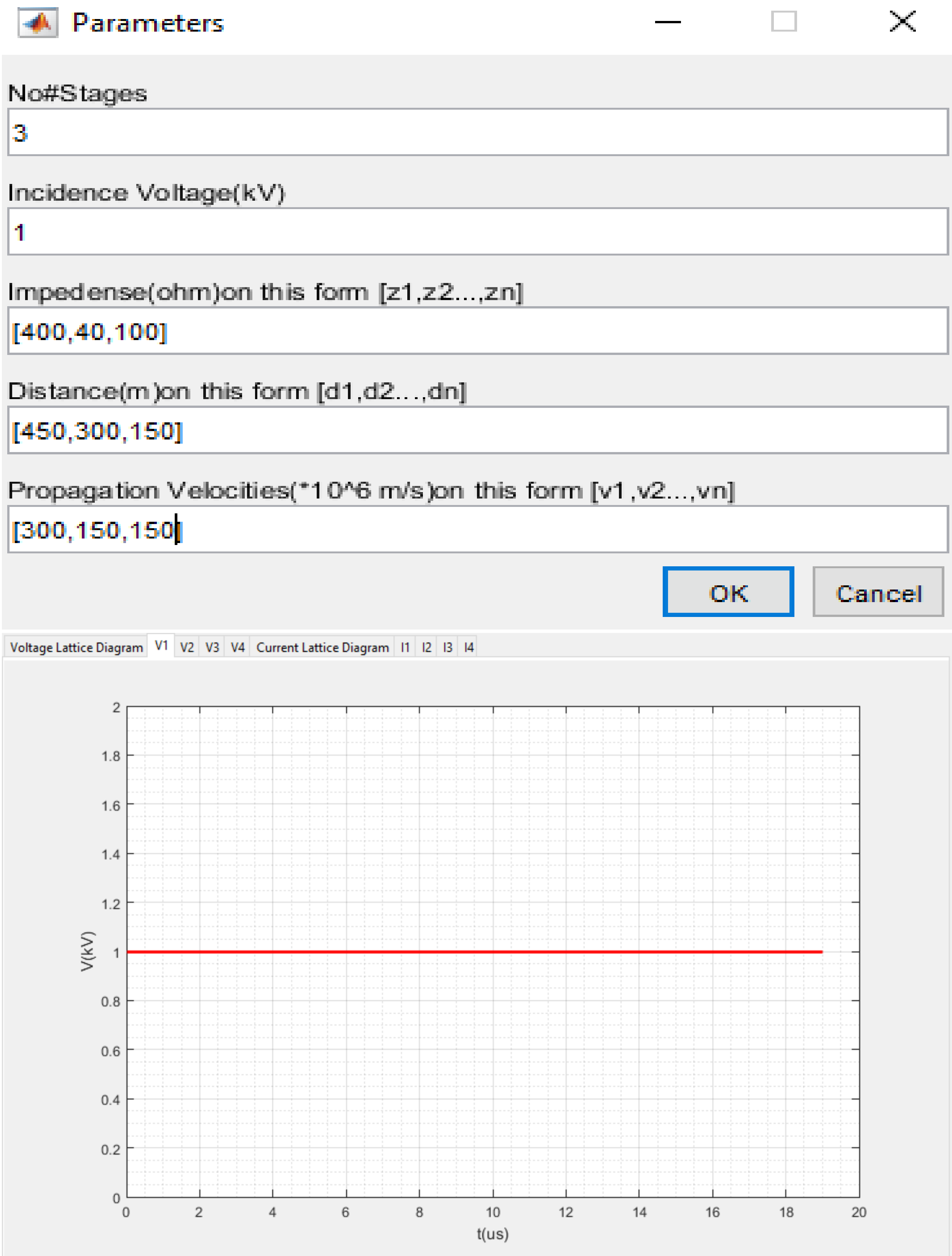
Voltage Lattice Diagram
V1
V2
V3
Current Lattice Diagram
I1
I2
I3
I(kA)
t(us)

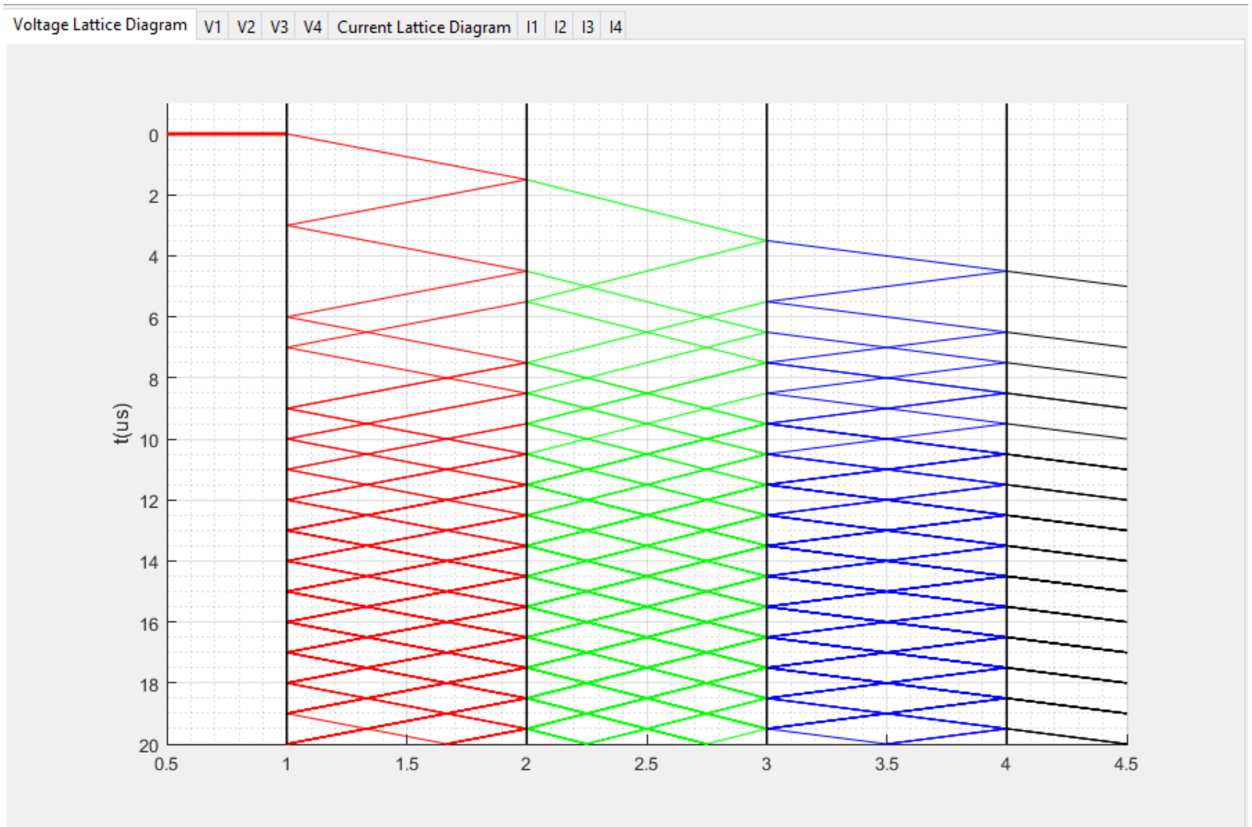


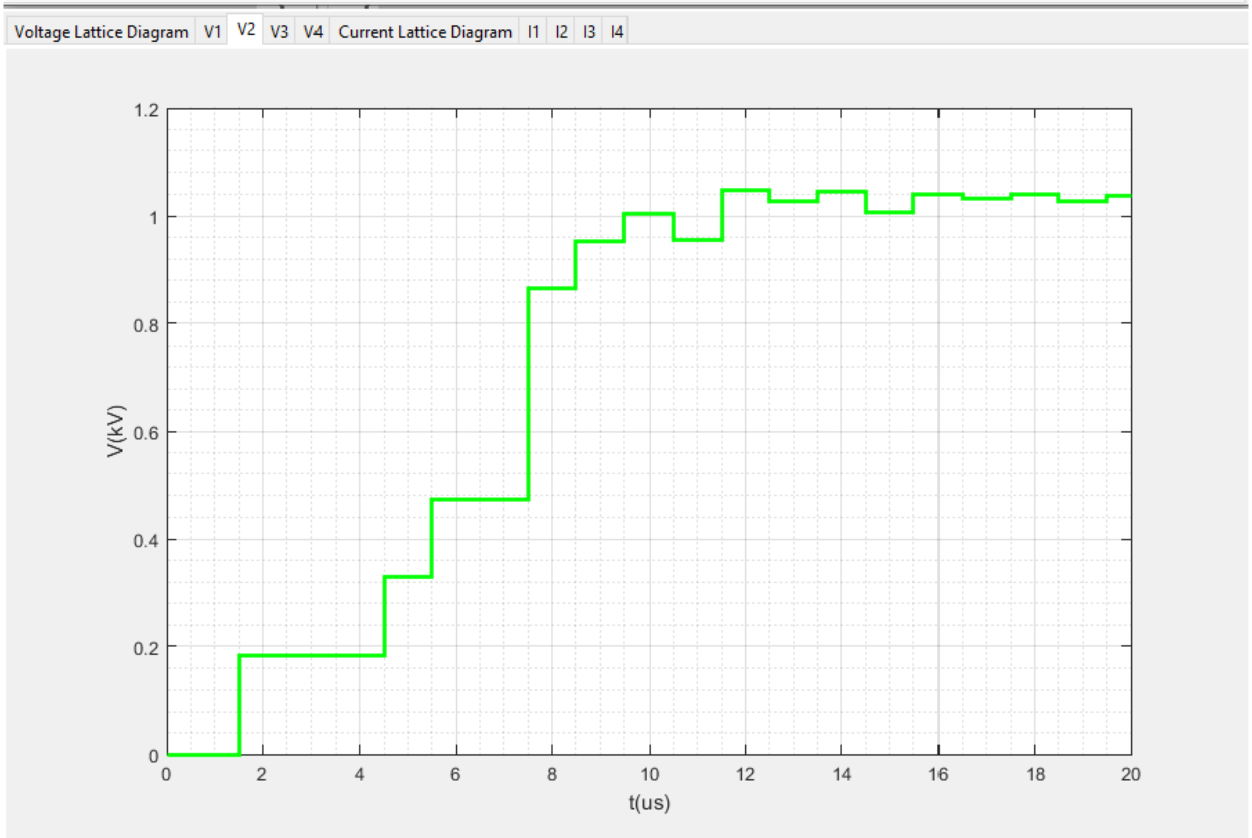
Voltage Lattice Diagram
V1
V2
V3
Current Lattice Diagram
I1
I2
I3
I(kA)
t(us)

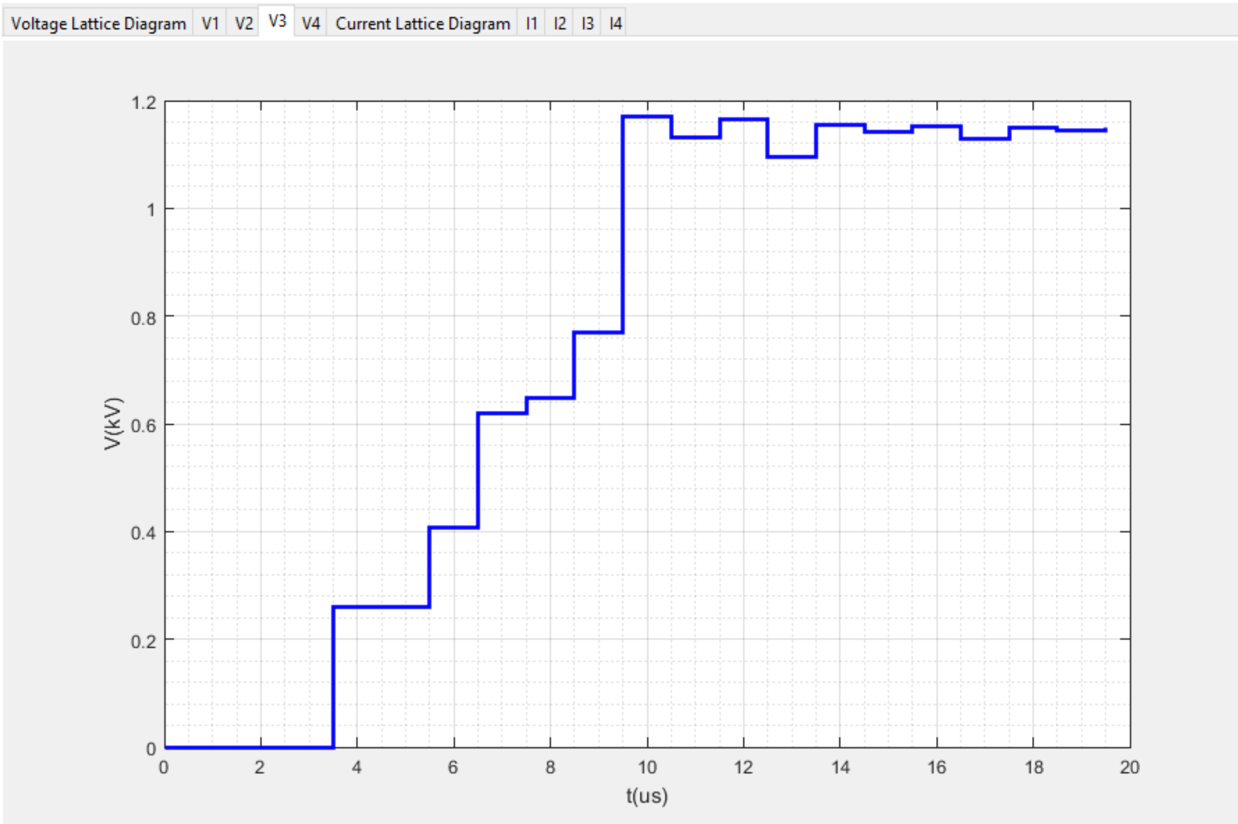
4 Different scenarios & Test cases:-

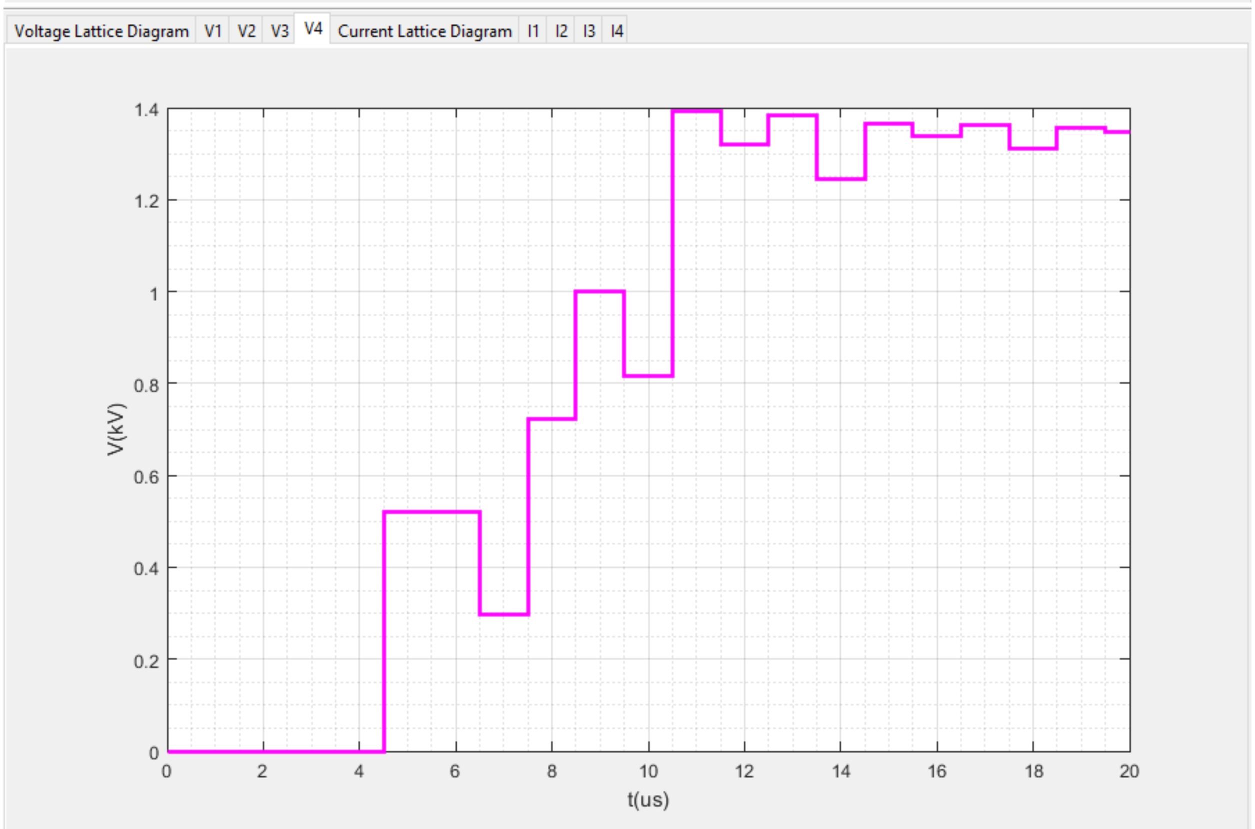
Part 2 Test Case1:-

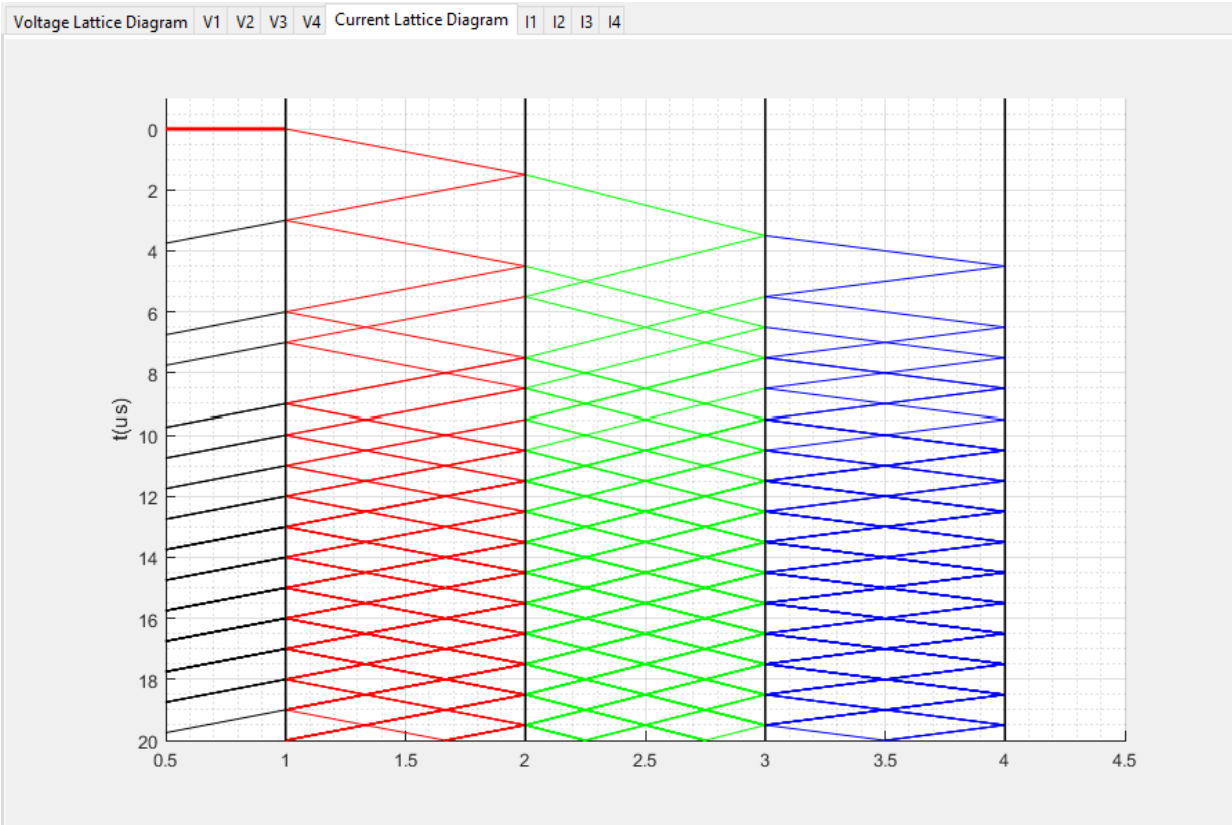


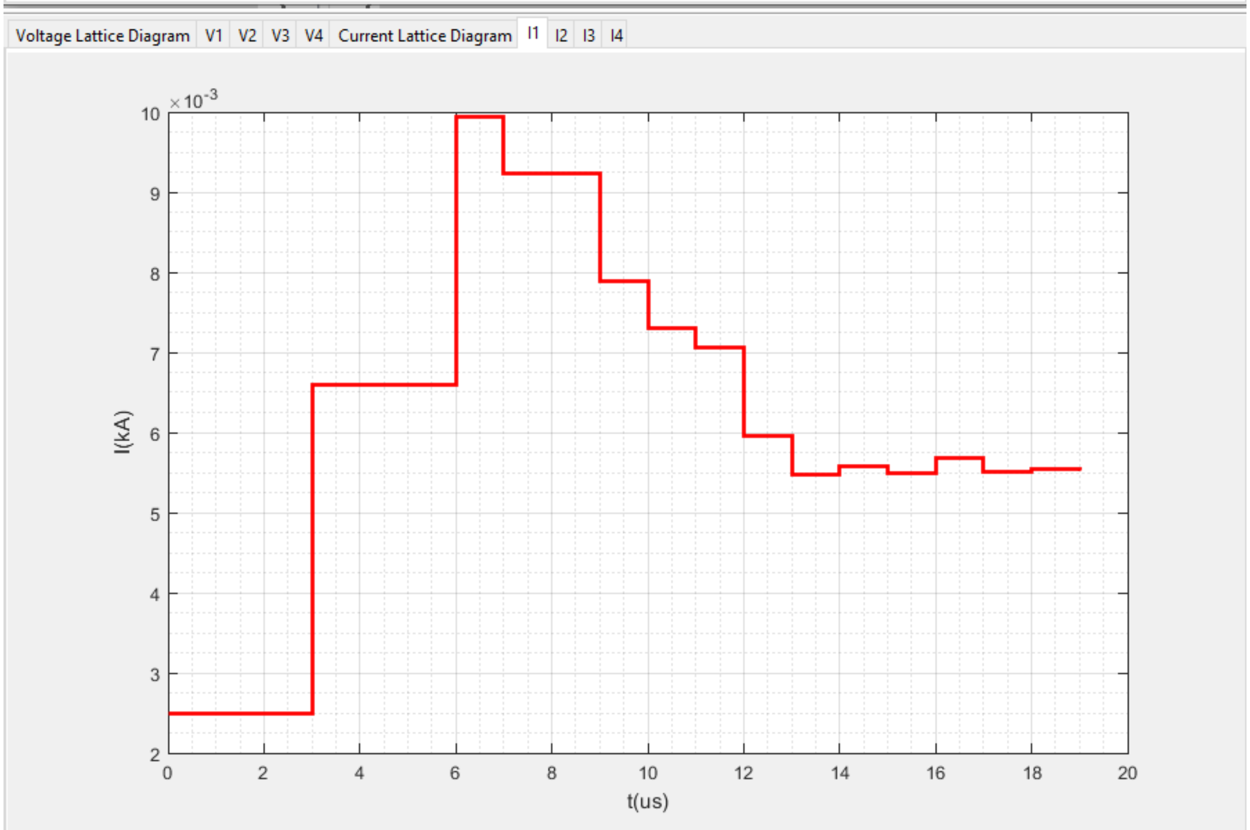

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
t(us)
0
2
4
6
8
10
12
14
16
18
20
0.5
1
1.5
2
2.5
3
3.5
4
4.5

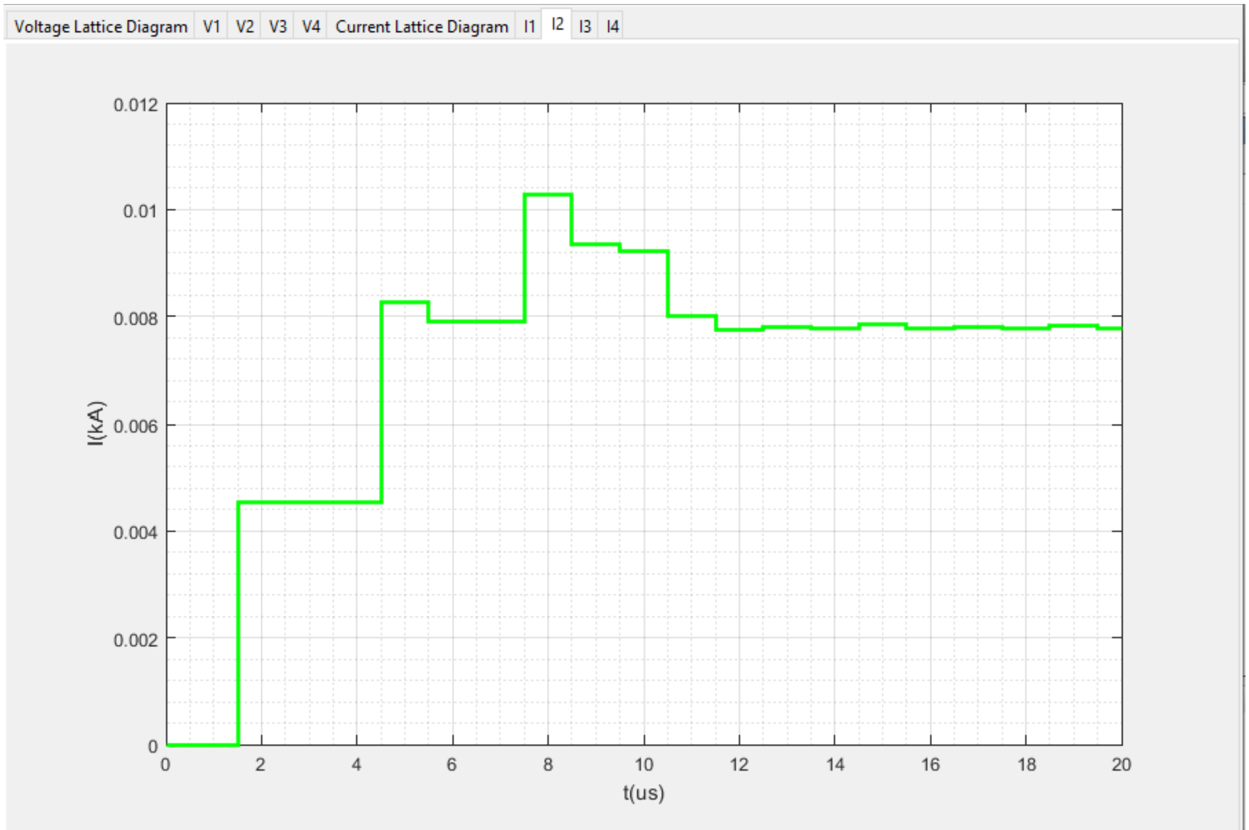

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
V(kV)
1.2
1
0.8
0.6
0.4
0.2
0
t(us)
0
2
4
6
8
10
12
14
16
18
20

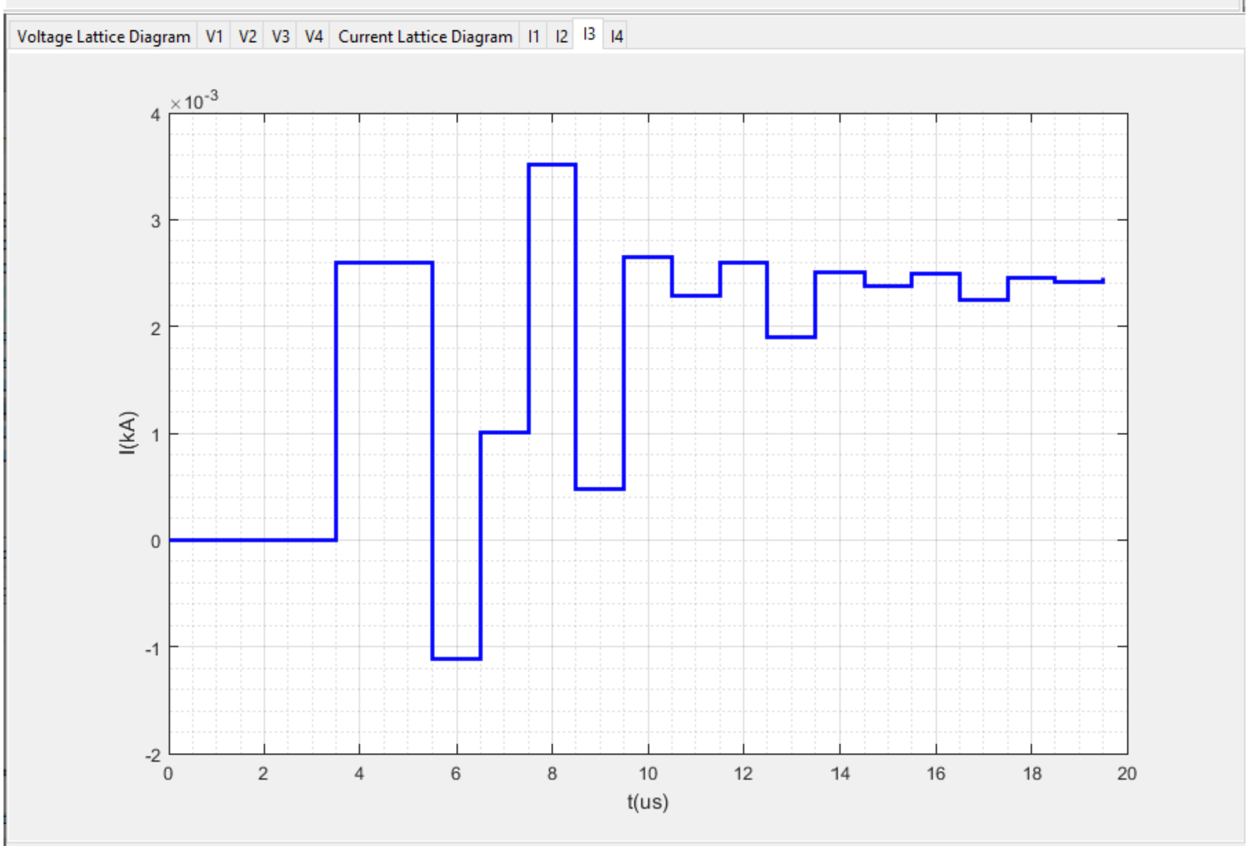

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
V(kV)
t(us)
0
0.2
0.4
0.6
0.8
1
1.2
0
2
4
6
8
10
12
14
16
18
20

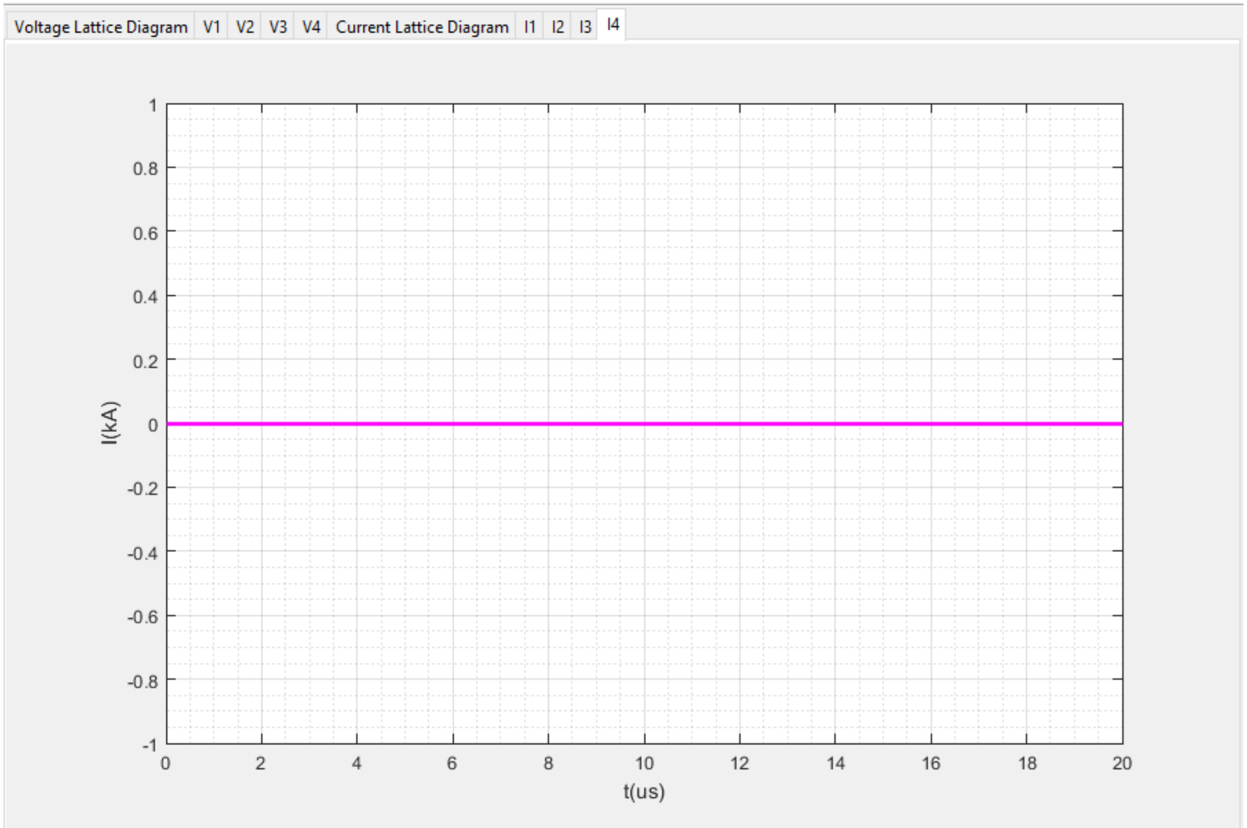

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
V(kV)
t(us)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
2
4
6
8
10
12
14
16
18
20


Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
t(us)

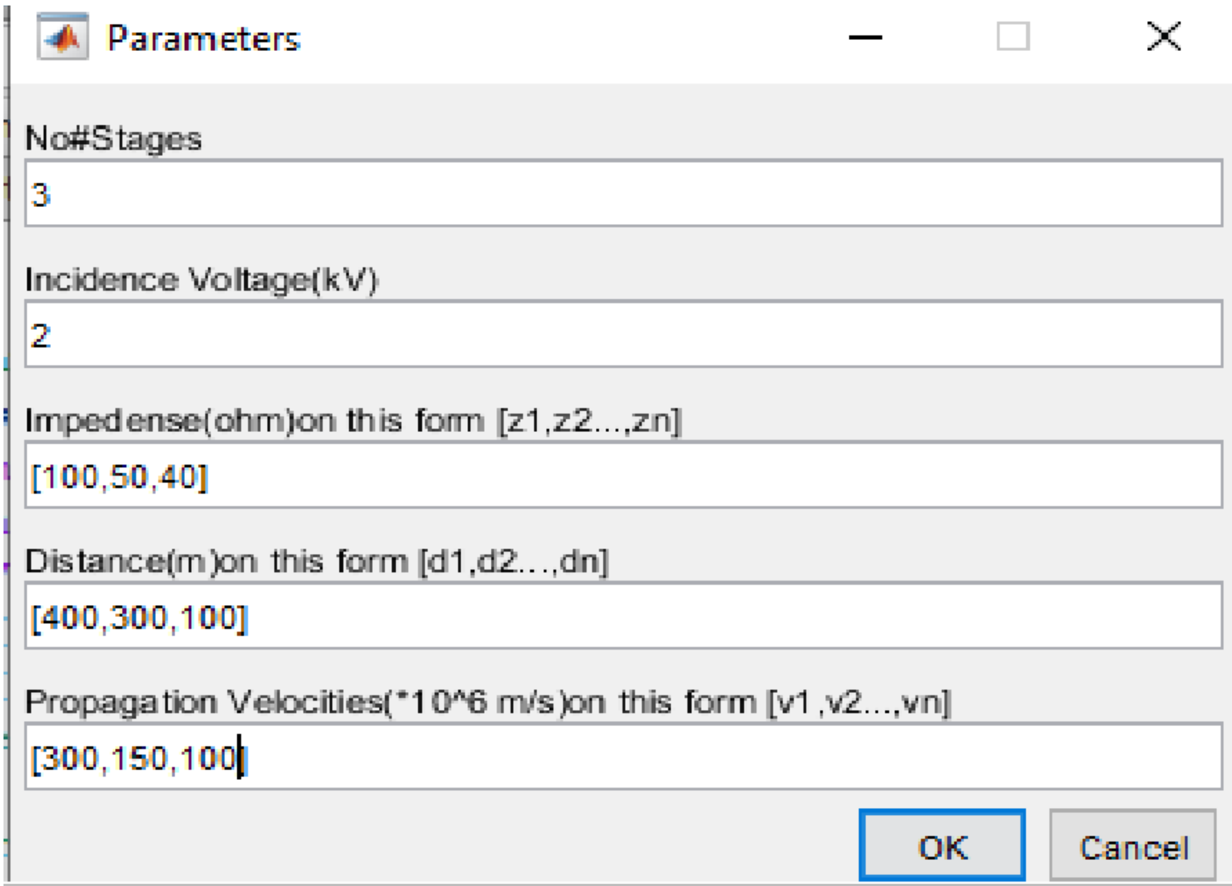

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
×10^-3
I(kA)
t(us)

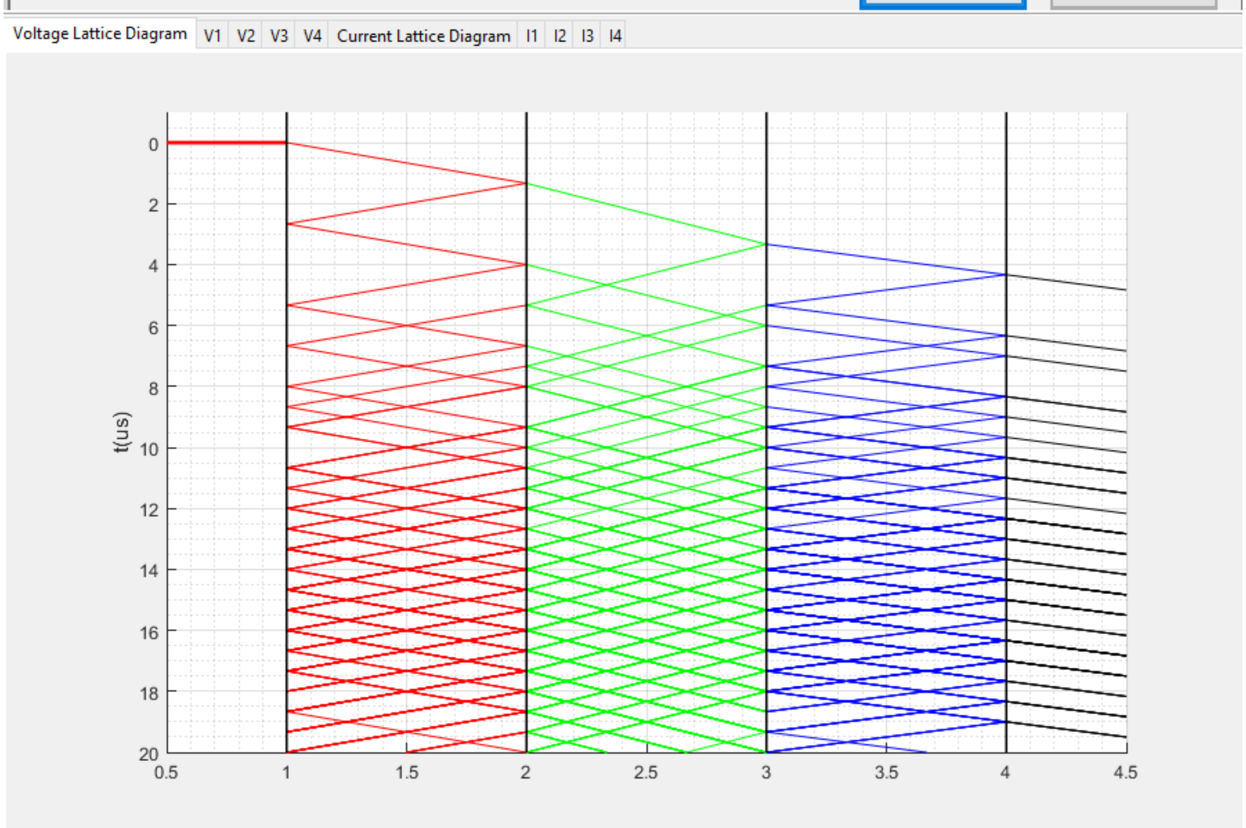

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
0.012
0.01
0.008
0.006
0.004
0.002
0
I(kA)
0
2
4
6
8
10
12
14
16
18
20
t(us)

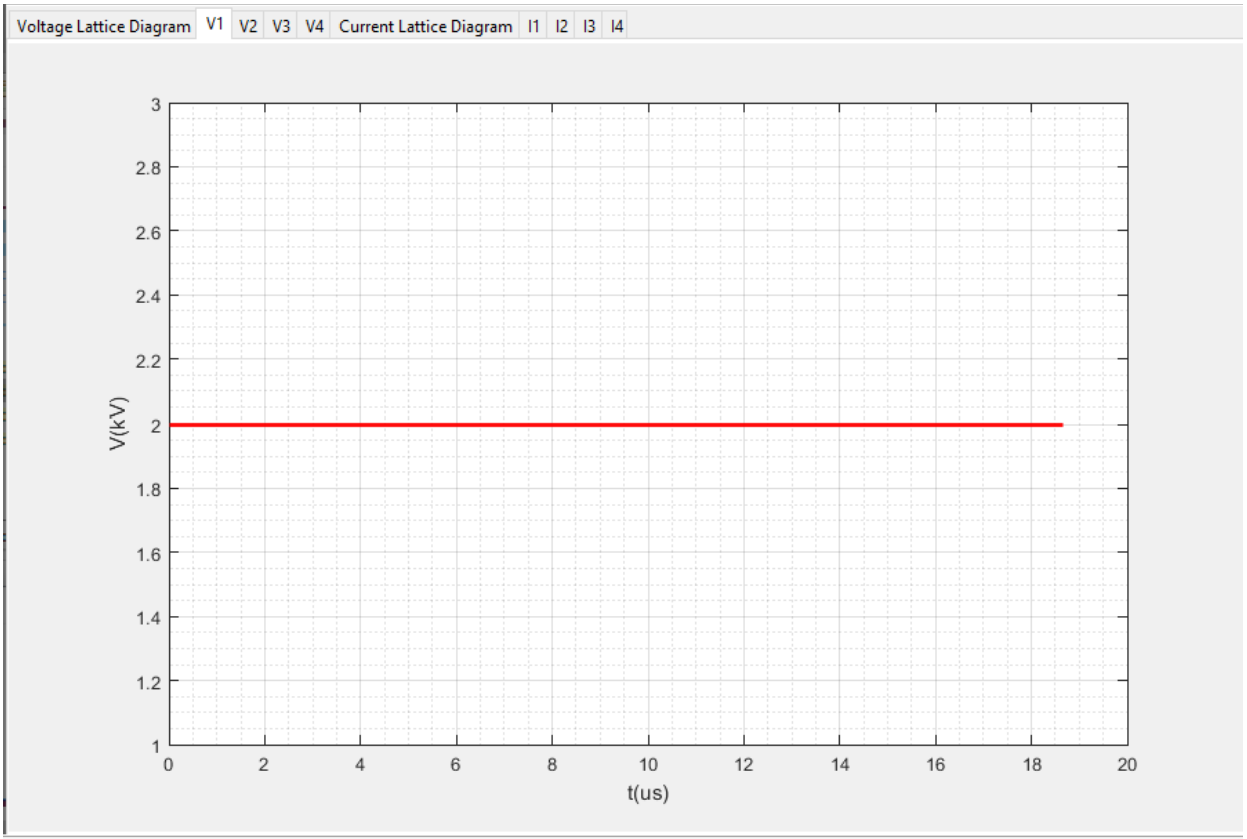

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
× 10^-3
4
3
2
1
0
-1
-2
I(kA)
0
2
4
6
8
10
12
14
16
18
20
t(us)


Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
I(kA)
t(us)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12
14
16
18
20

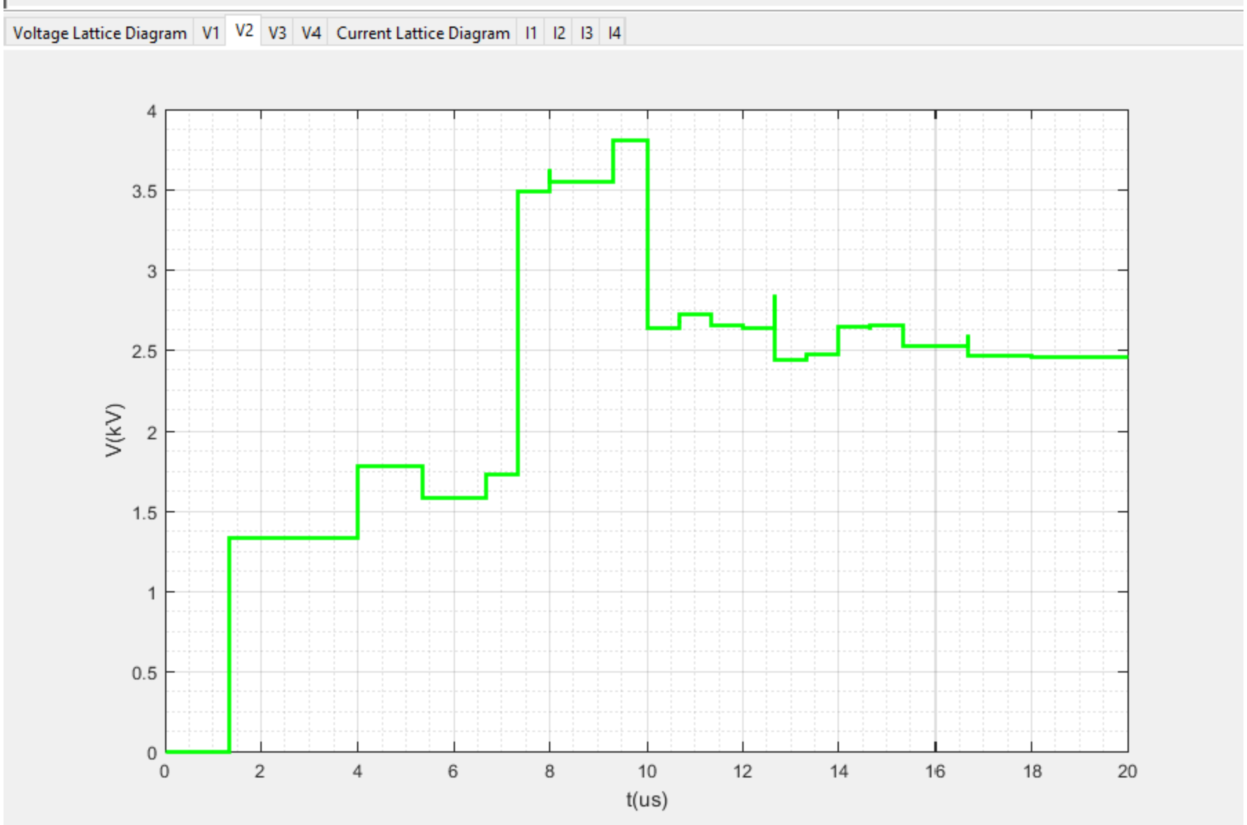
Part 2 Test Case2:-



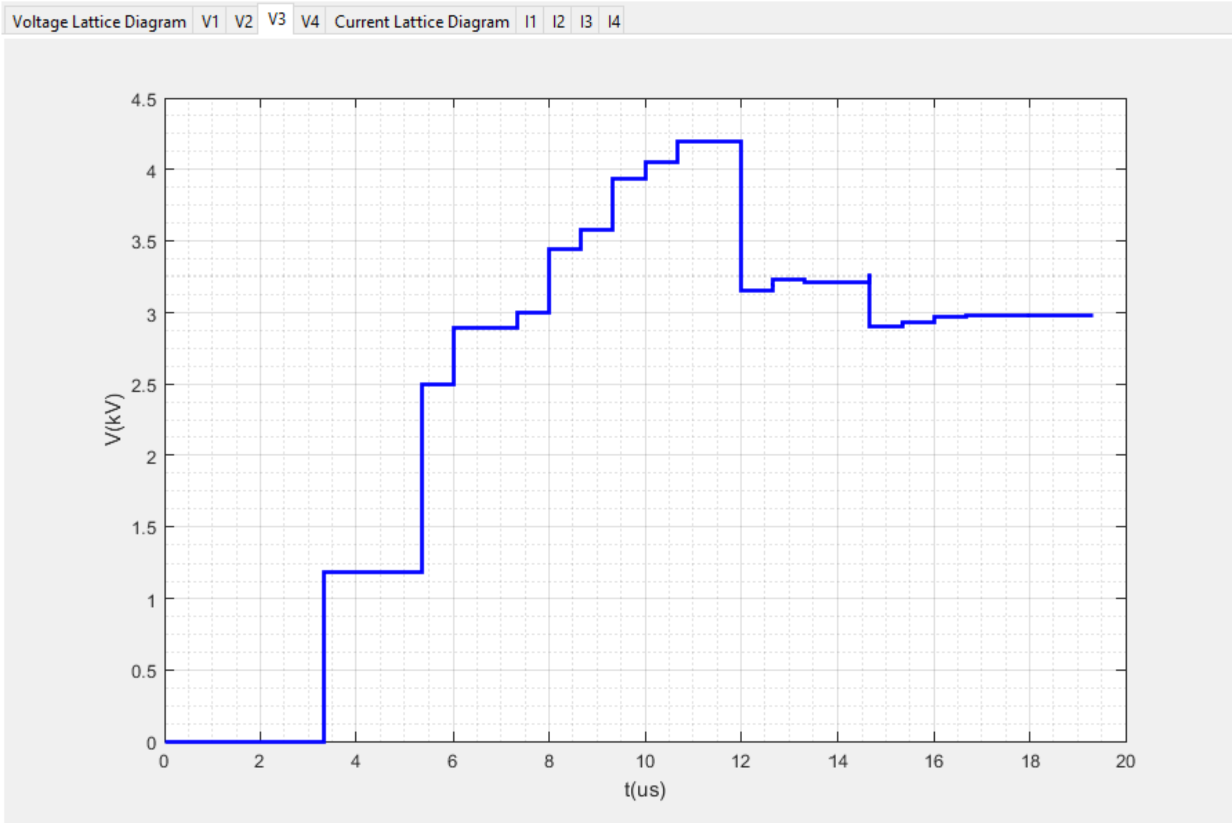


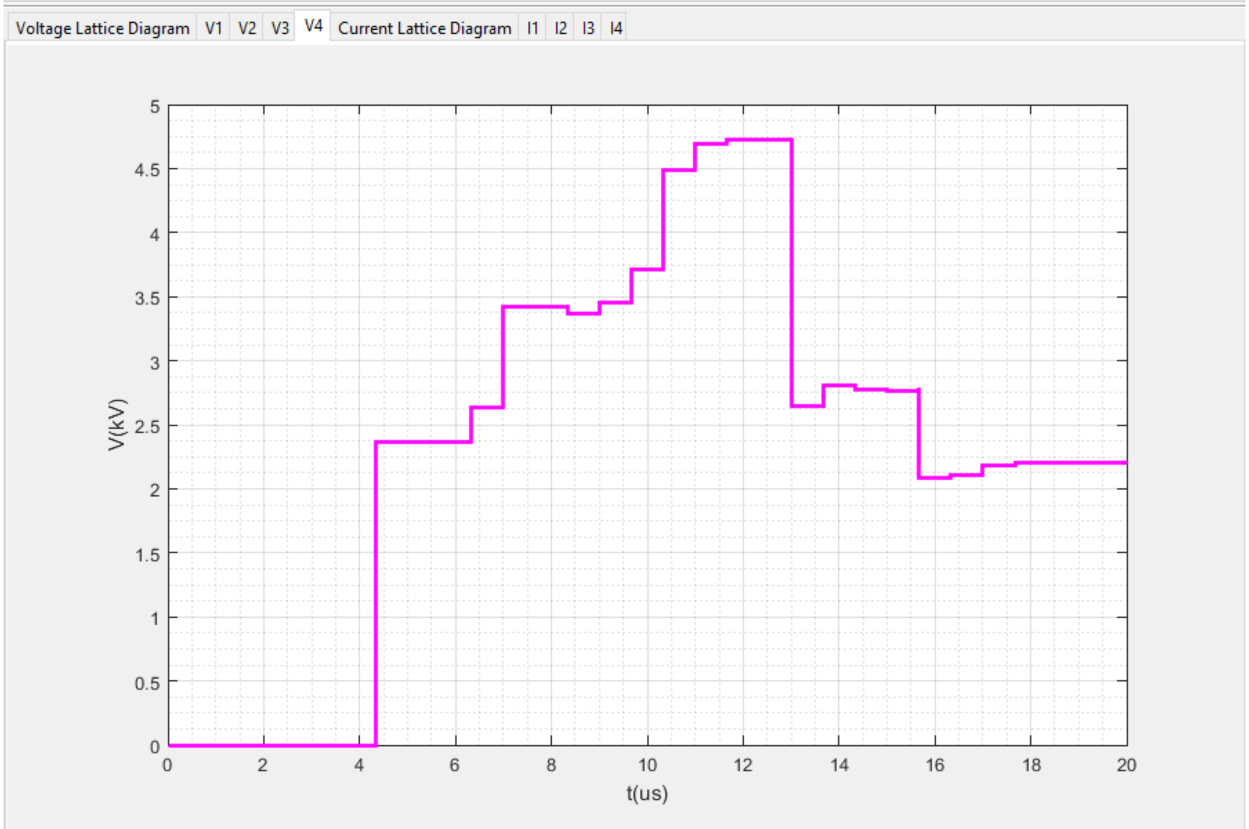


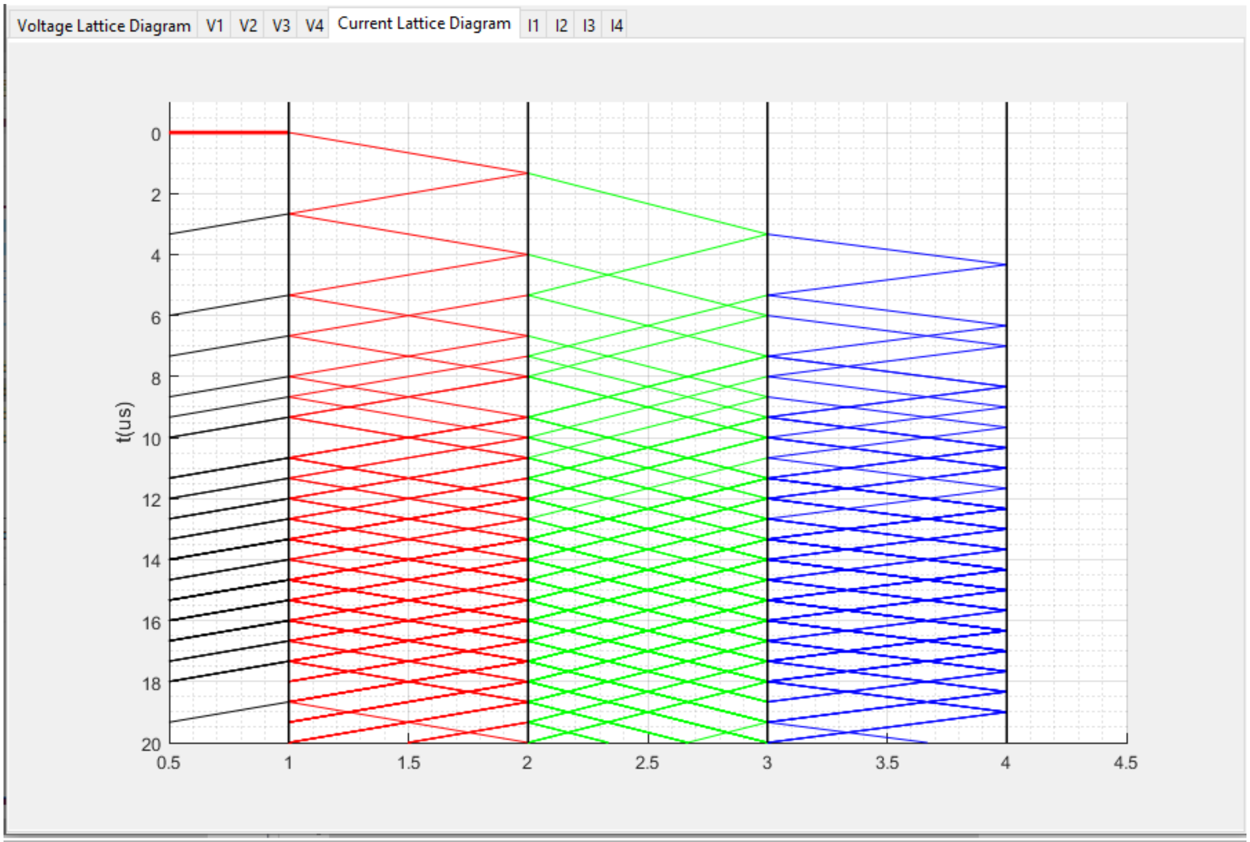
Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
V(kV)
t(us)

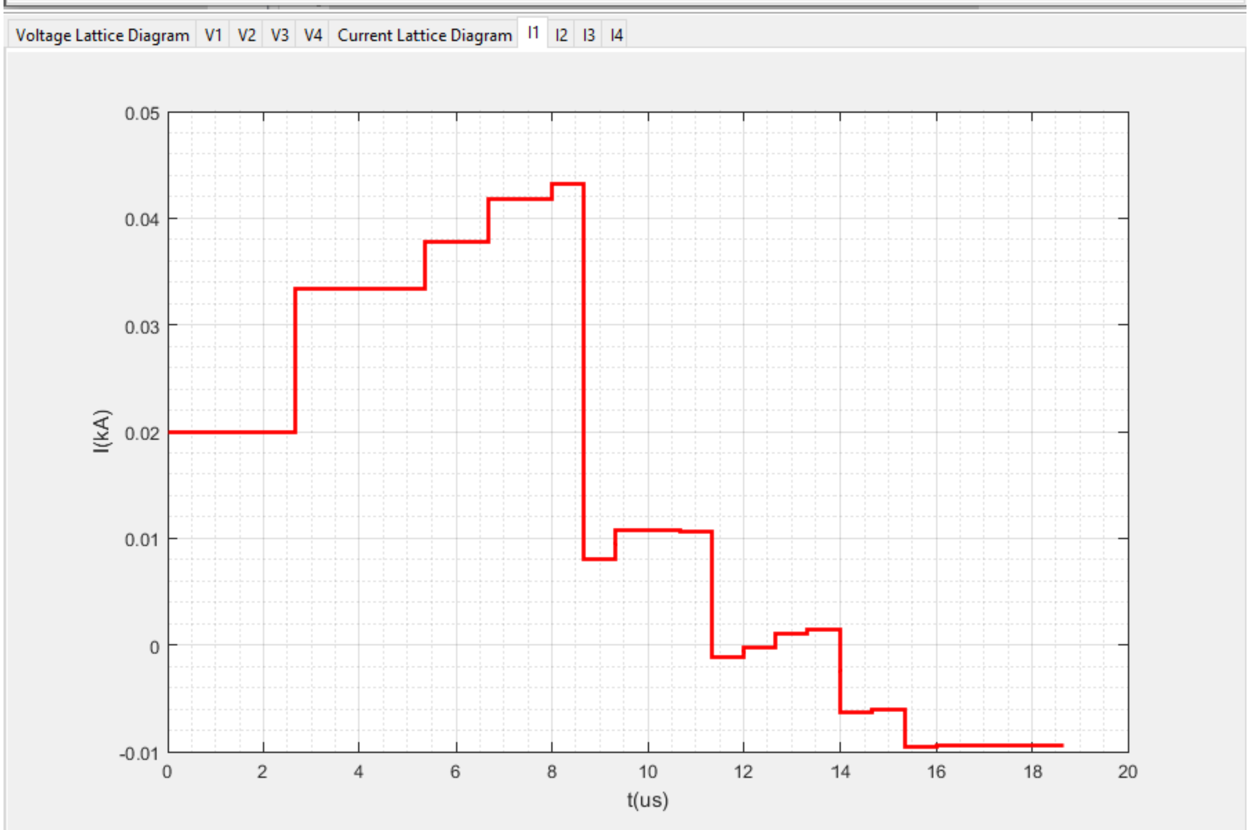


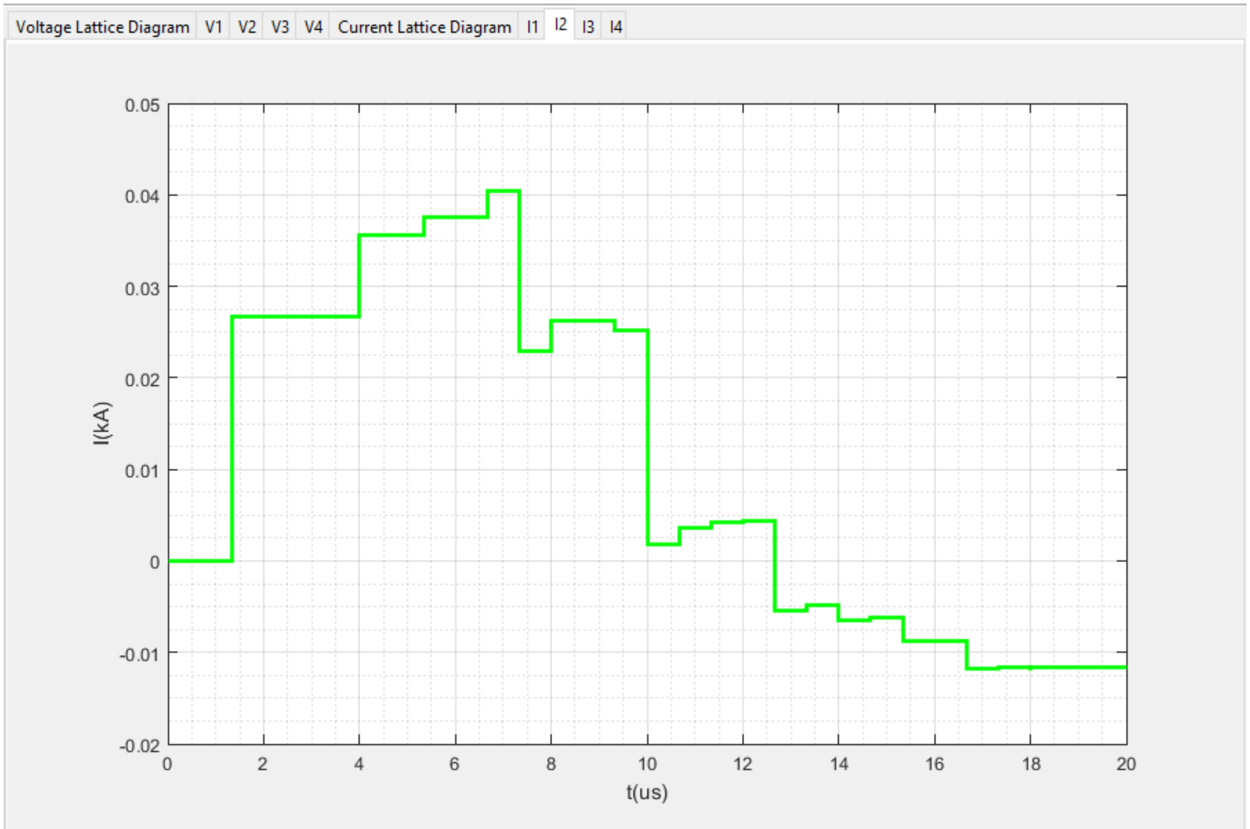
Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
V(kV)
t(us)

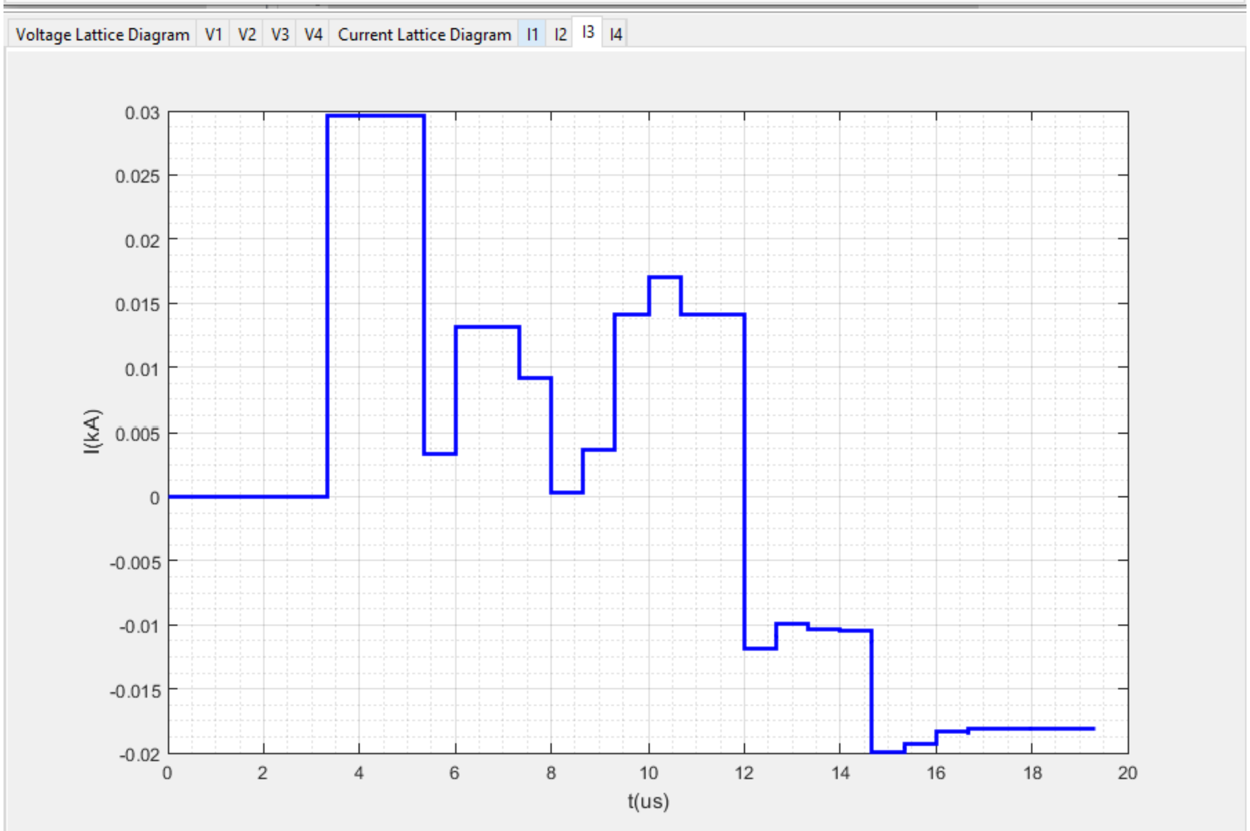

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
V(kV)
t(us)

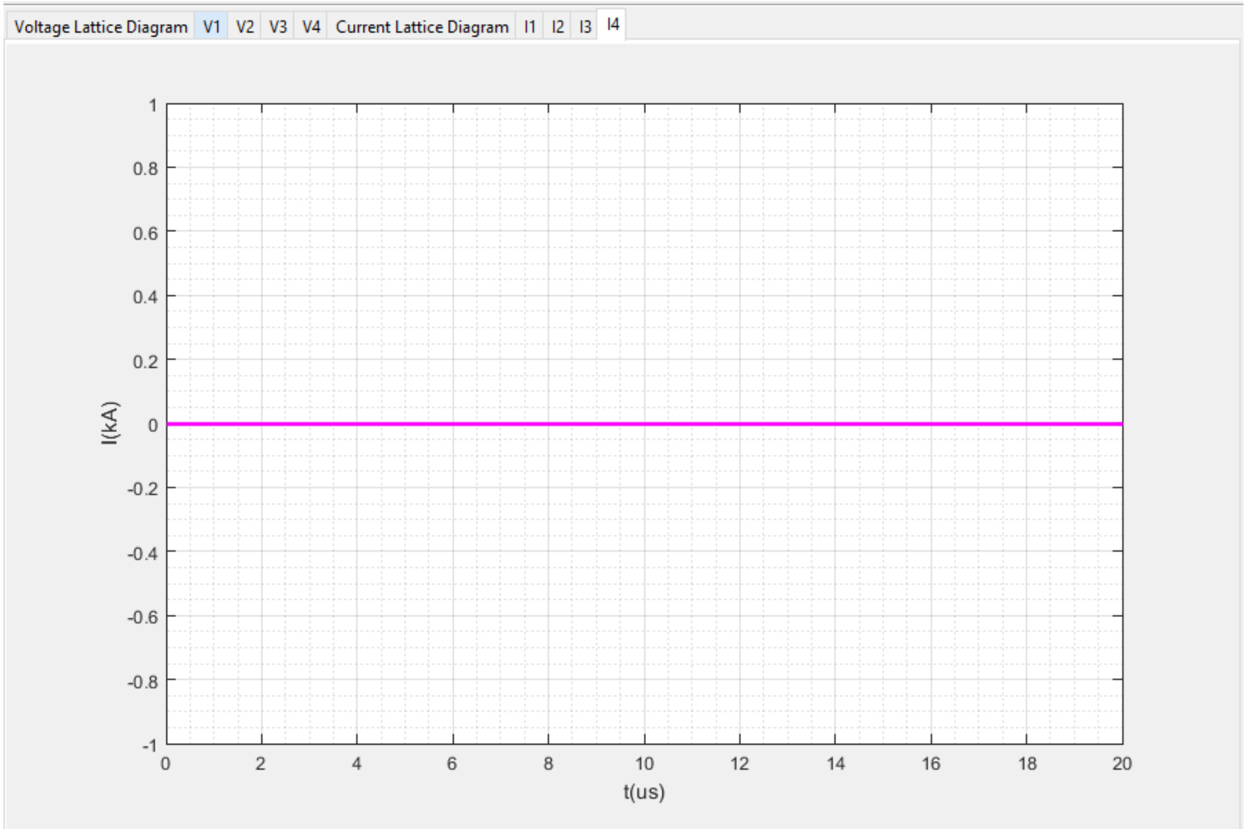

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
V(kV)
t(us)


Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
t(us)
0
2
4
6
8
10
12
14
16
18
20
0.5
1
1.5
2
2.5
3
3.5
4
4.5

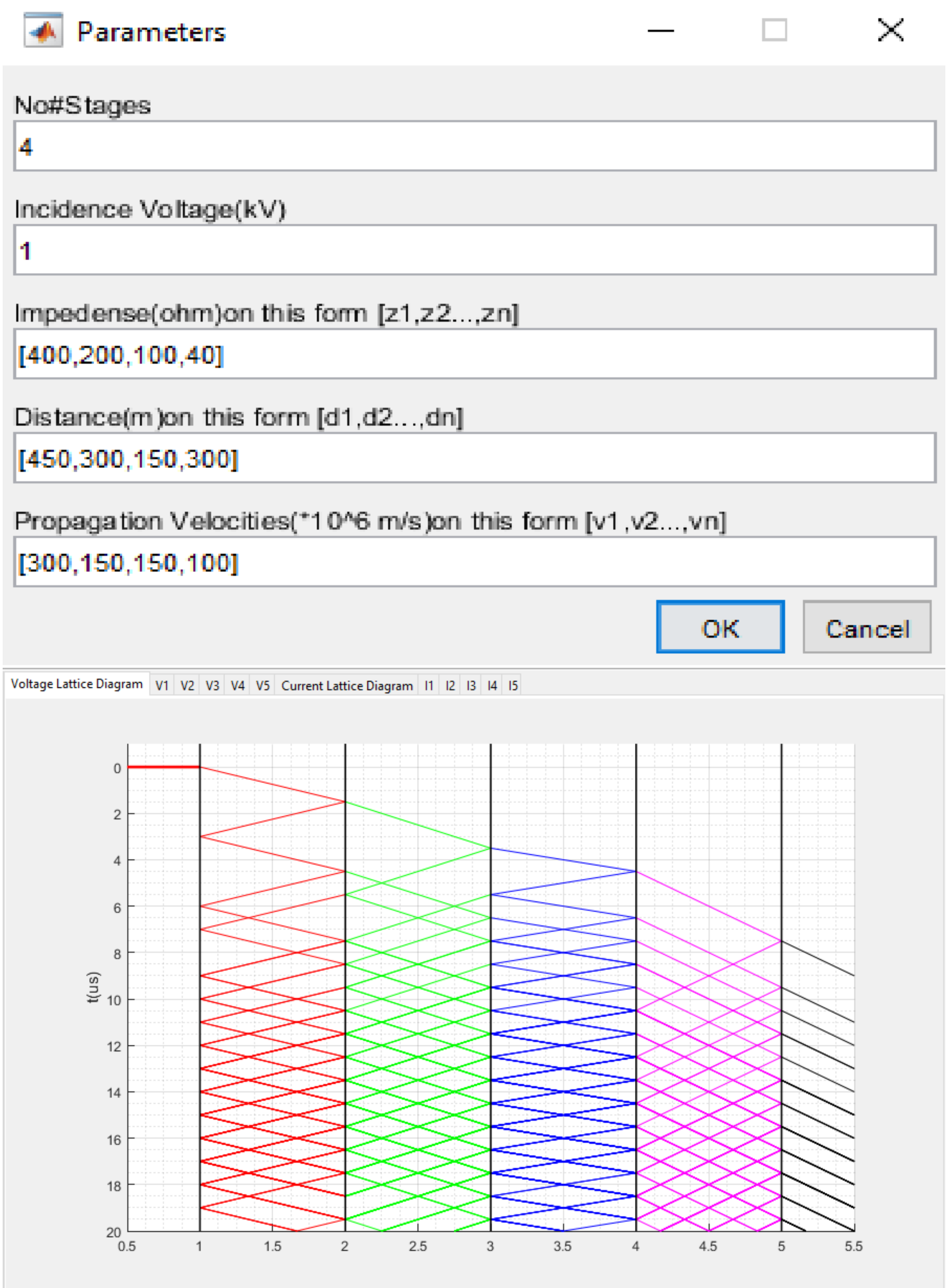

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
I(kA)
0.05
0.04
0.03
0.02
0.01
0
-0.01
0
2
4
6
8
10
12
14
16
18
20
t(us)

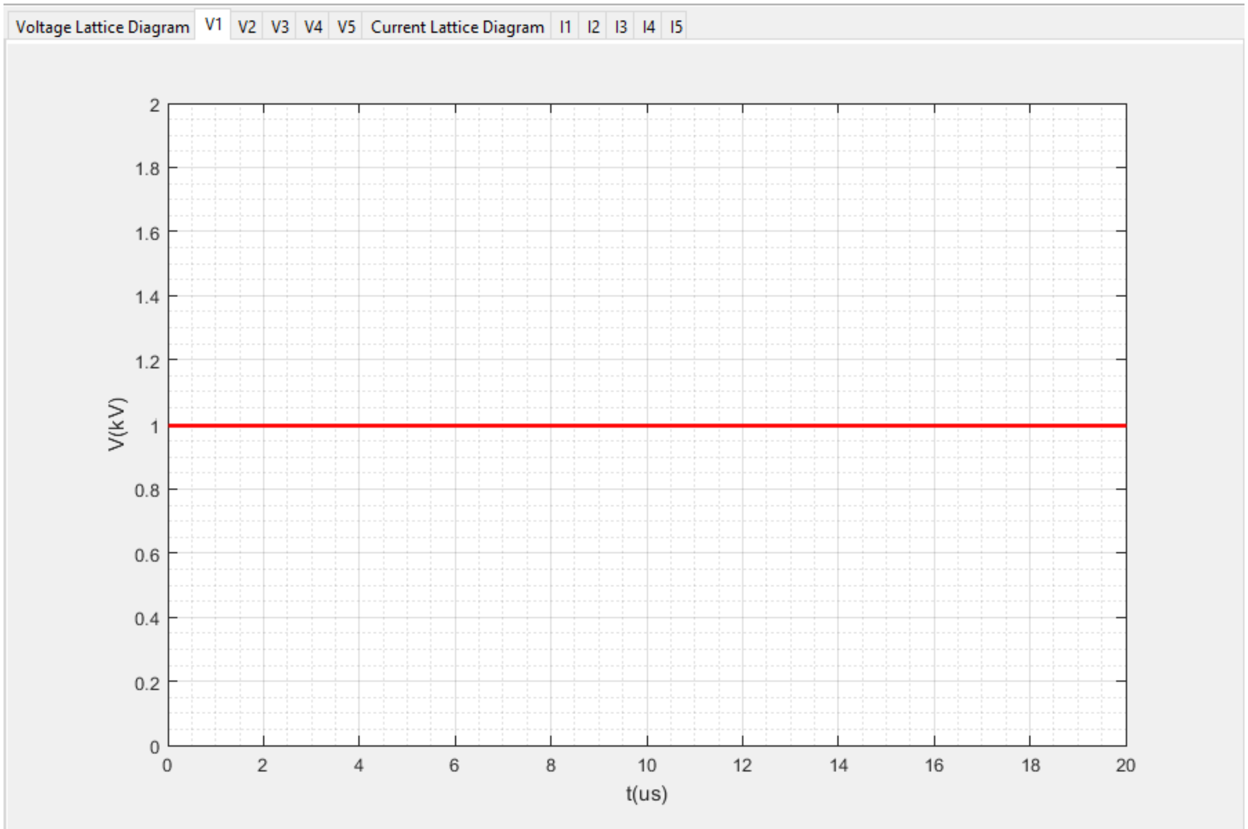

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
I(kA)
t(us)

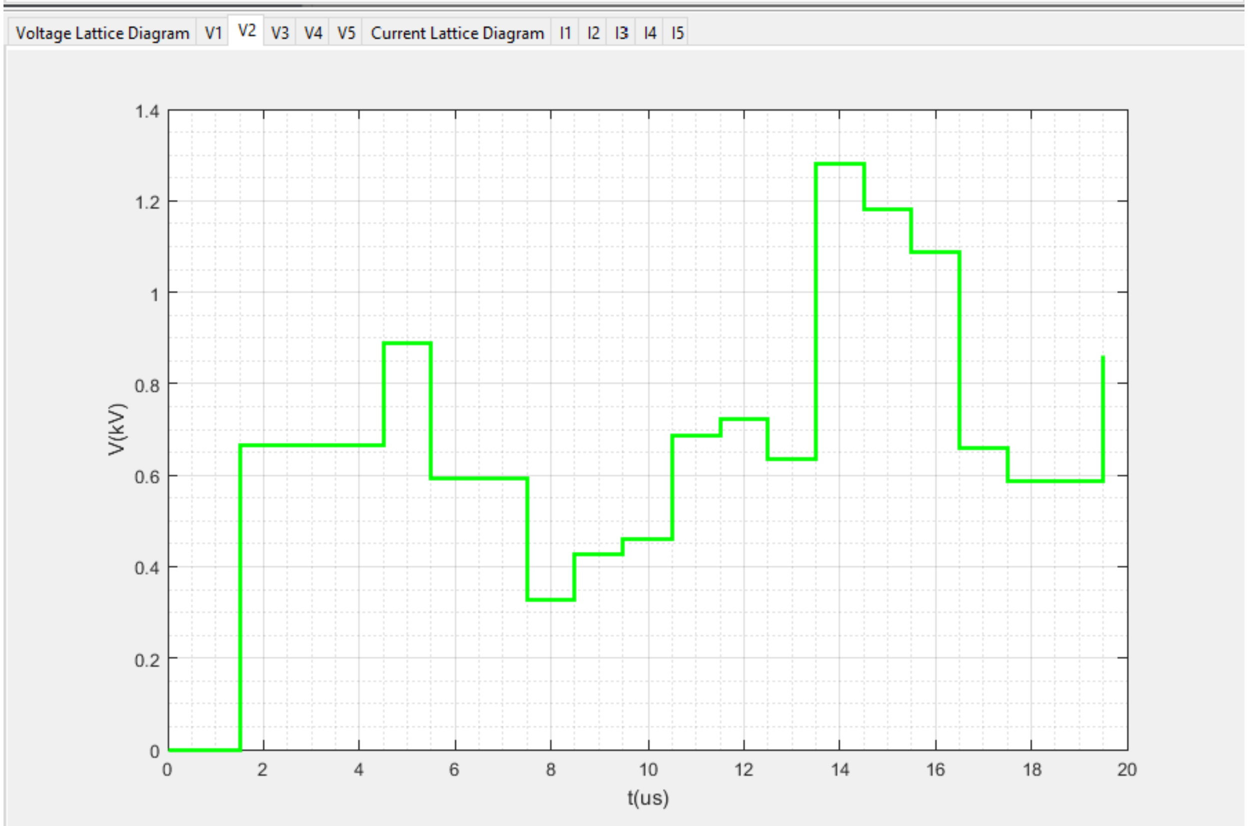

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
I(kA)
t(us)

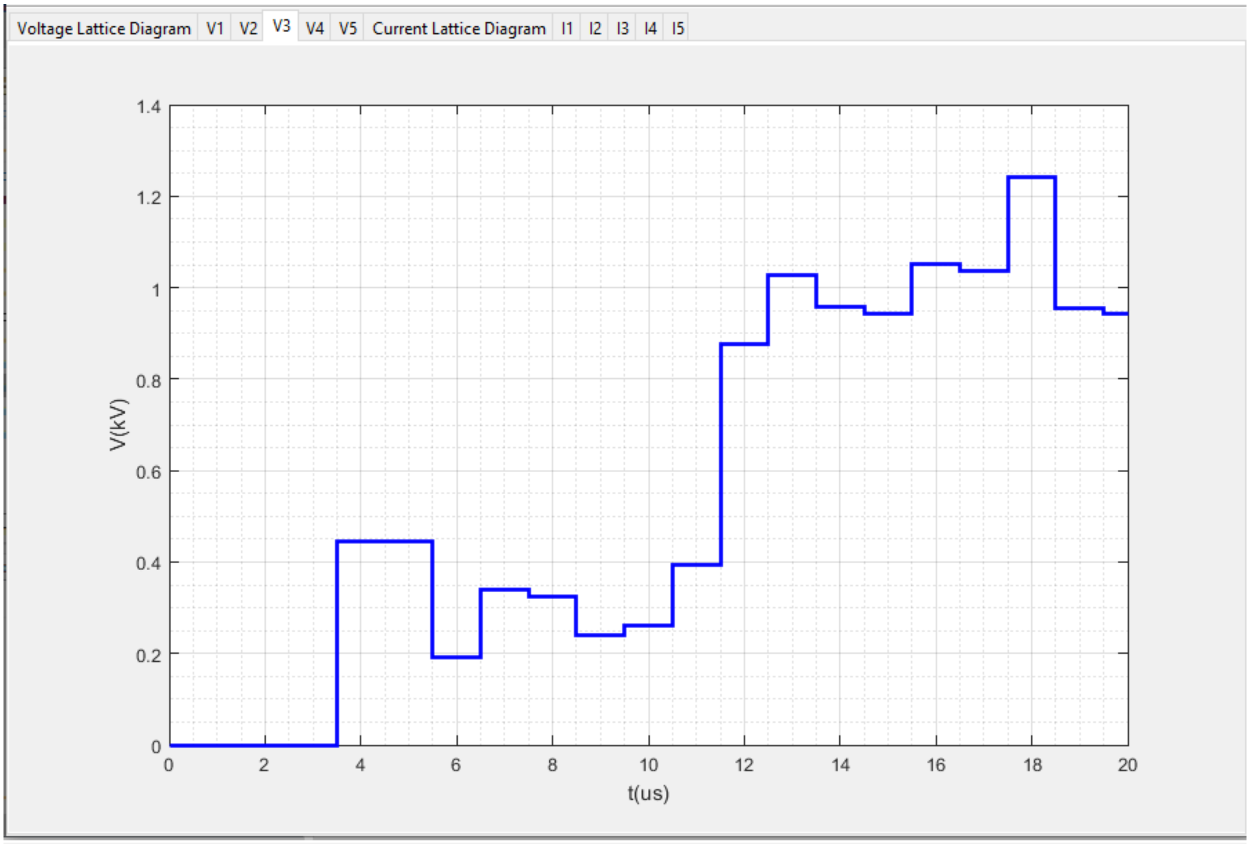

Voltage Lattice Diagram
V1
V2
V3
V4
Current Lattice Diagram
I1
I2
I3
I4
I(kA)
t(us)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
2
4
6
8
10
12
14
16
18
20

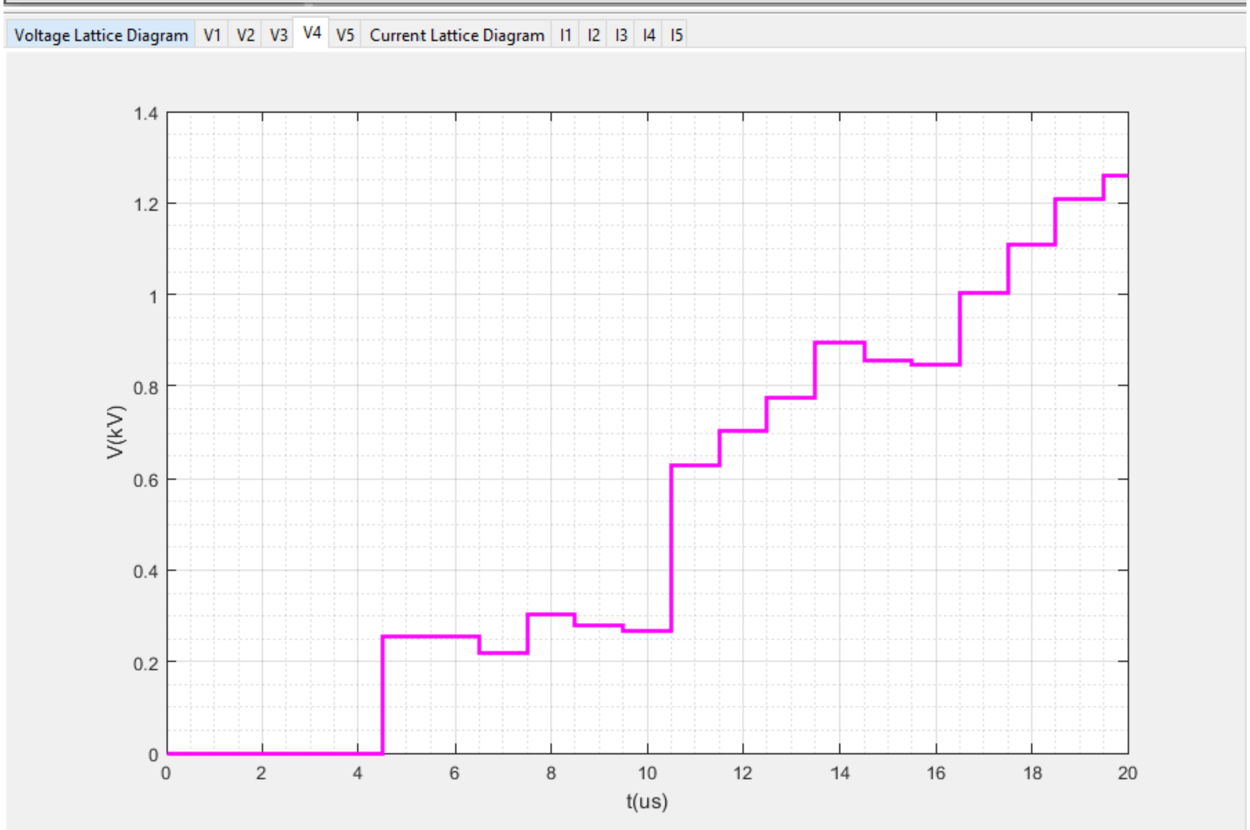
Part 2 Test Case3:-

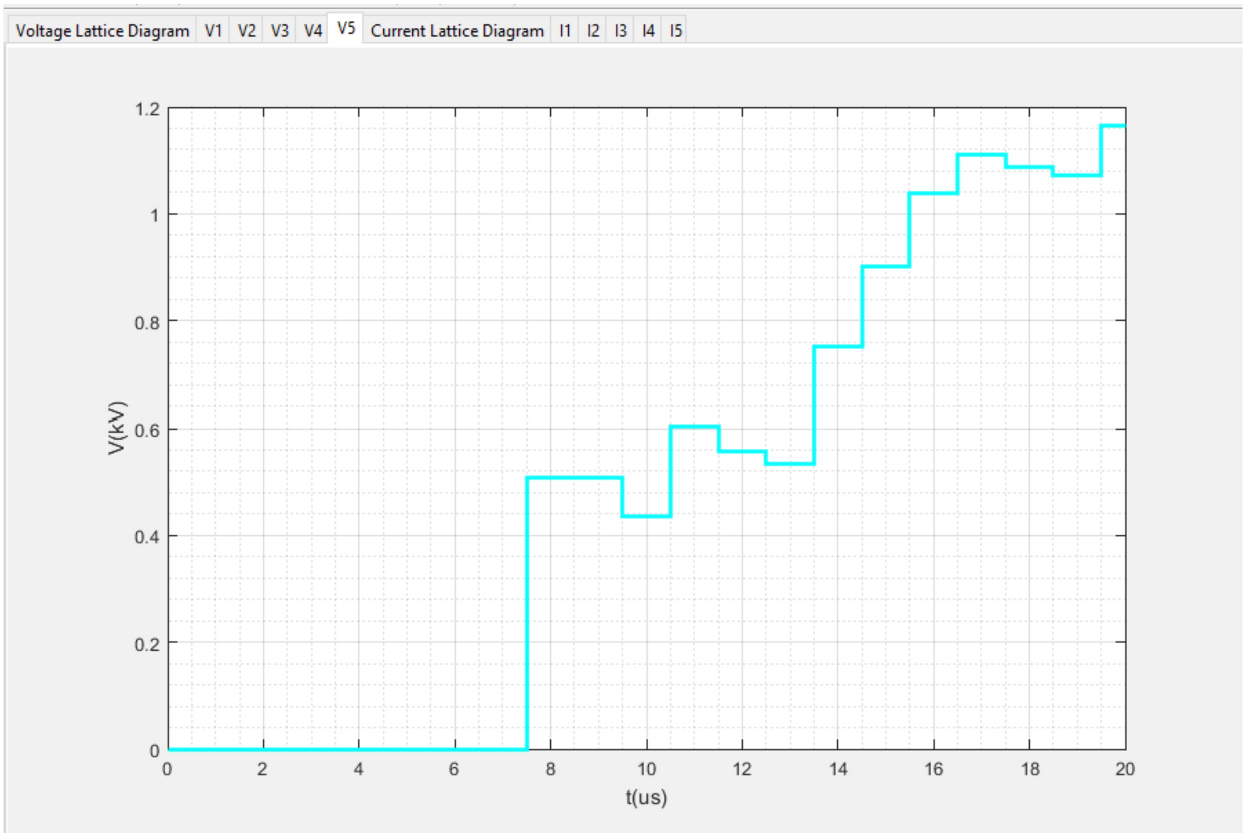



Voltage Lattice Diagram
V1
V2
V3
V4
V5
Current Lattice Diagram
I1
I2
I3
I4
I5
V(kV)
t(us)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
2
4
6
8
10
12
14
16
18
20

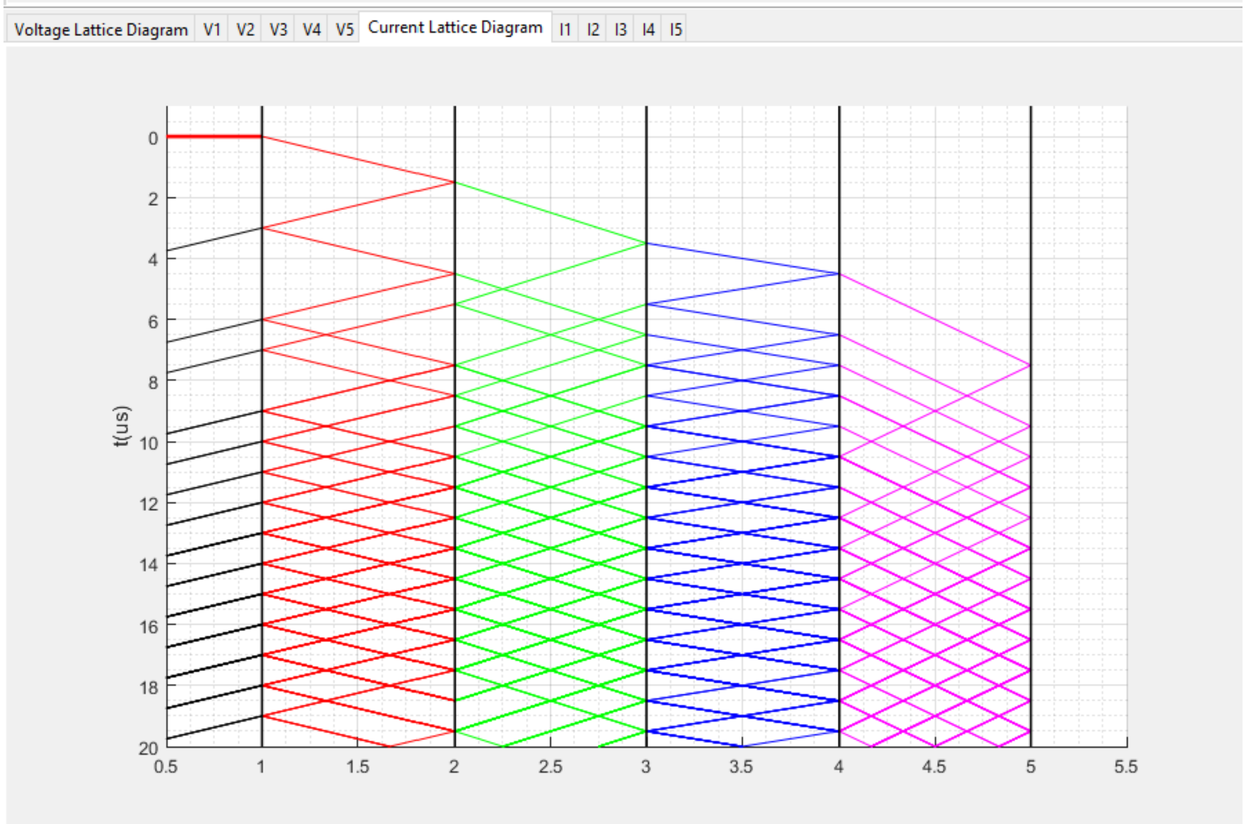

Voltage Lattice Diagram
V1
V2
V3
V4
V5
Current Lattice Diagram
I1
I2
I3
I4
I5
V(kV)
t(us)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
2
4
6
8
10
12
14
16
18
20


Voltage Lattice Diagram
V1
V2
V3
V4
V5
Current Lattice Diagram
I1
I2
I3
I4
I5
V(kV)
t(us)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
2
4
6
8
10
12
14
16
18
20

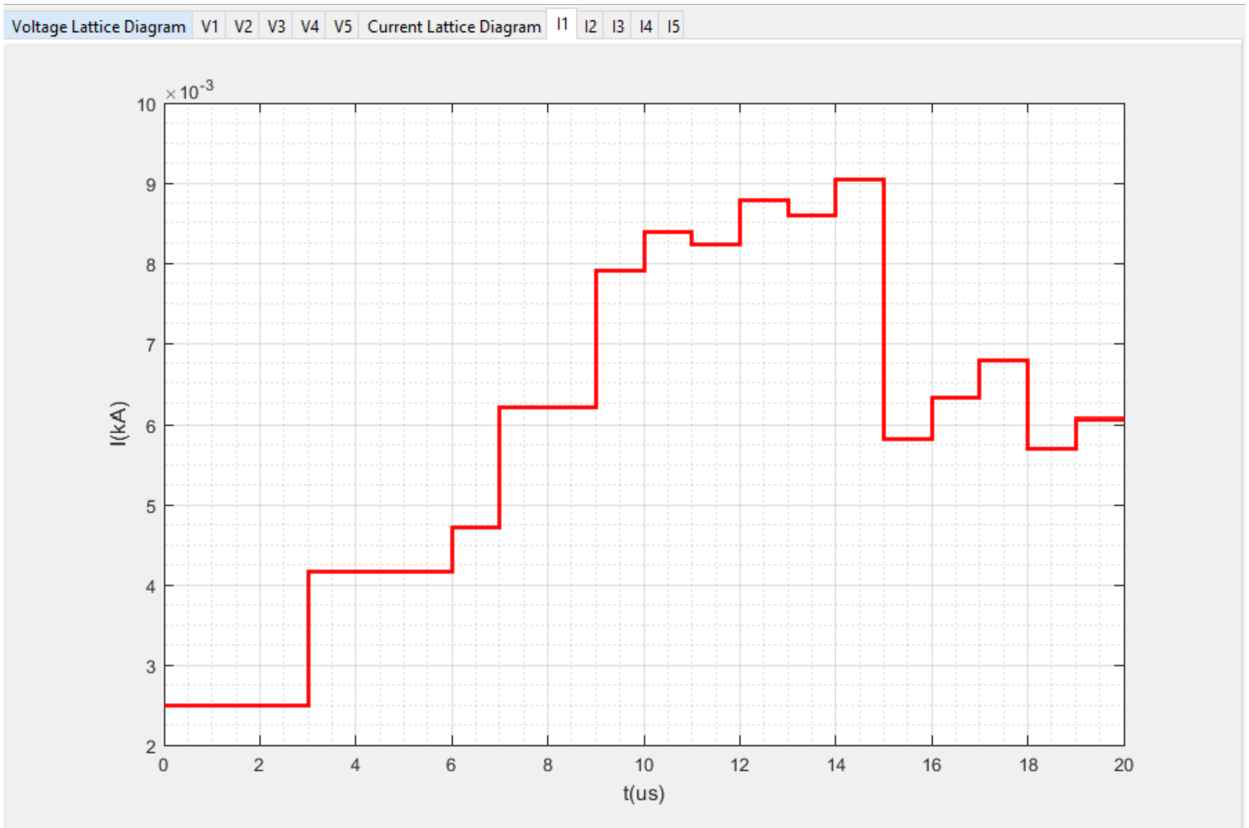

Voltage Lattice Diagram
V1
V2
V3
V4
V5
Current Lattice Diagram
I1
I2
I3
I4
I5
V(kV)
t(us)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
2
4
6
8
10
12
14
16
18
20



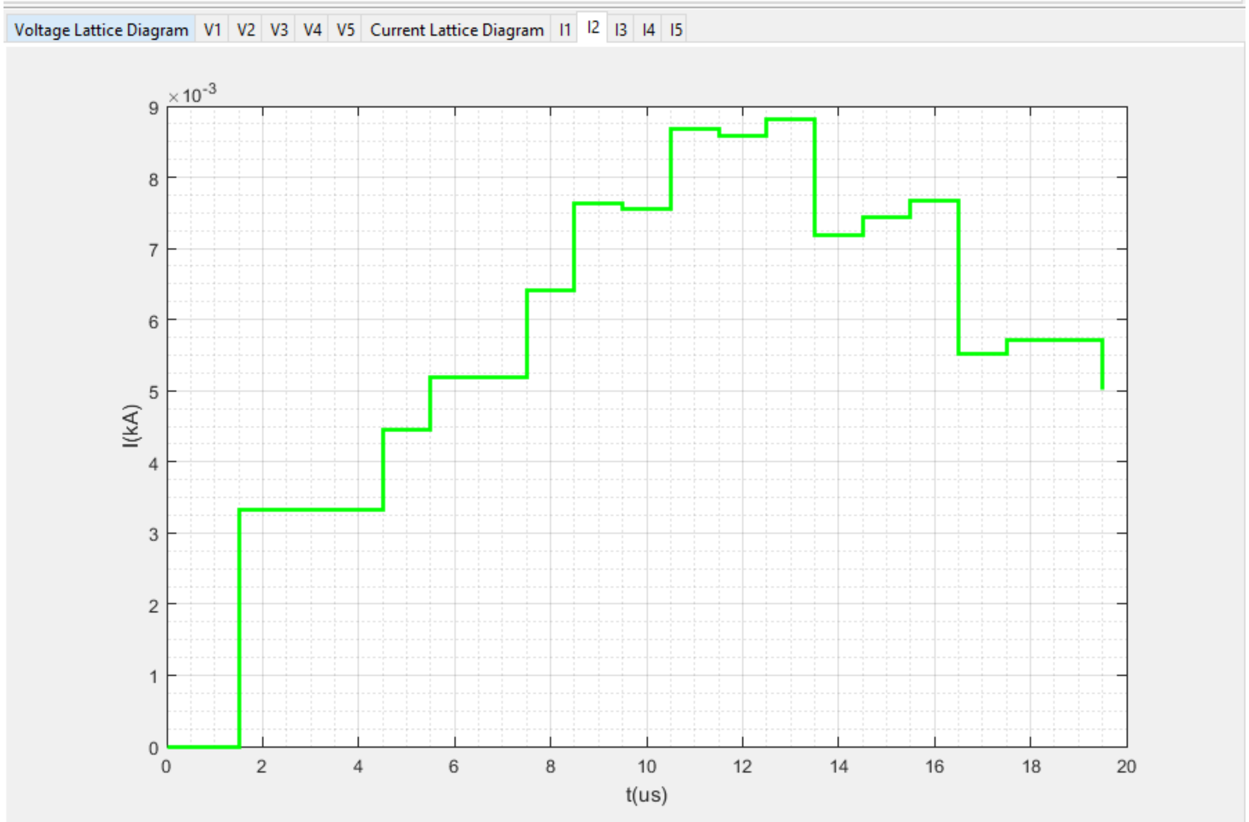
Voltage Lattice Diagram
V1
V2
V3
V4
V5
Current Lattice Diagram
I1
I2
I3
I4
I5
V(kV)
t(us)



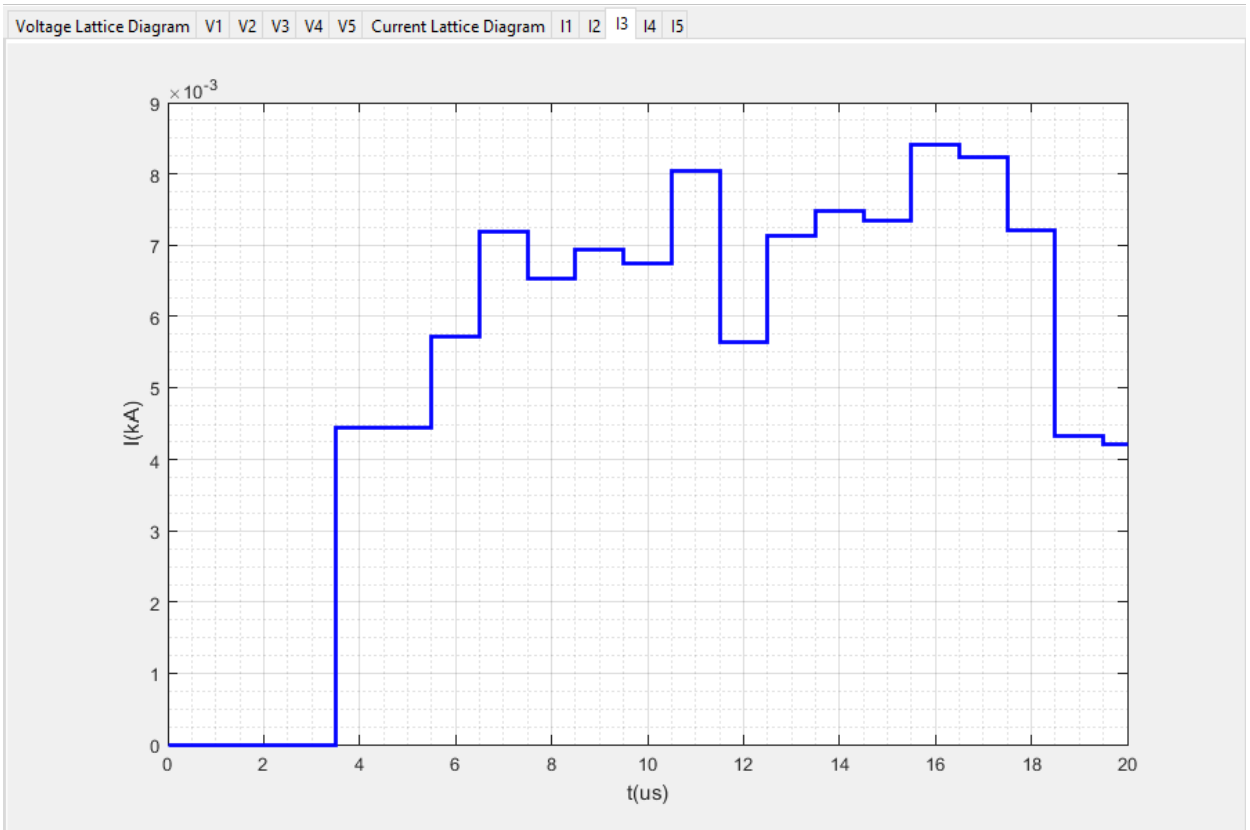
Voltage Lattice Diagram
V1
V2
V3
V4
V5
Current Lattice Diagram
I1
I2
I3
I4
I5
t(us)

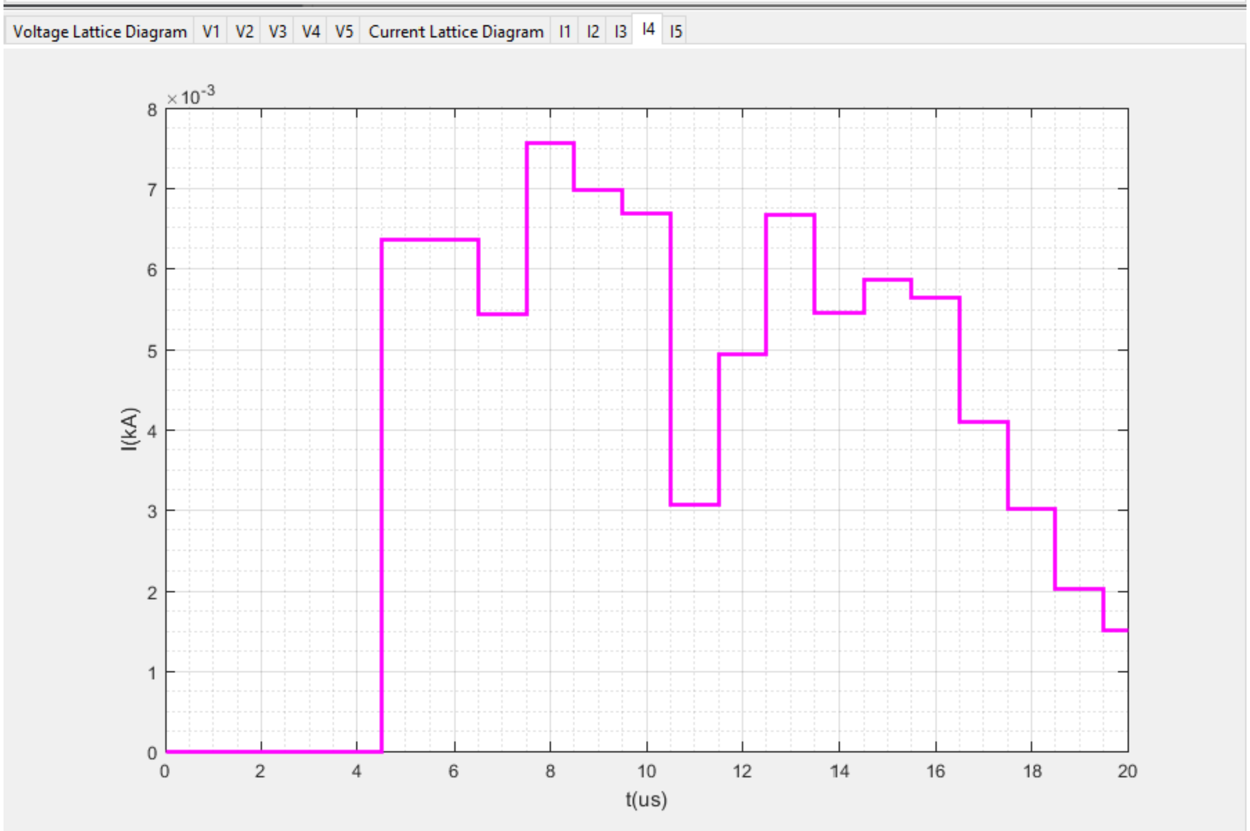


Voltage Lattice Diagram
V1
V2
V3
V4
V5
Current Lattice Diagram
I1
I2
I3
I4
I5
× 10^-3
I(kA)
t(us)

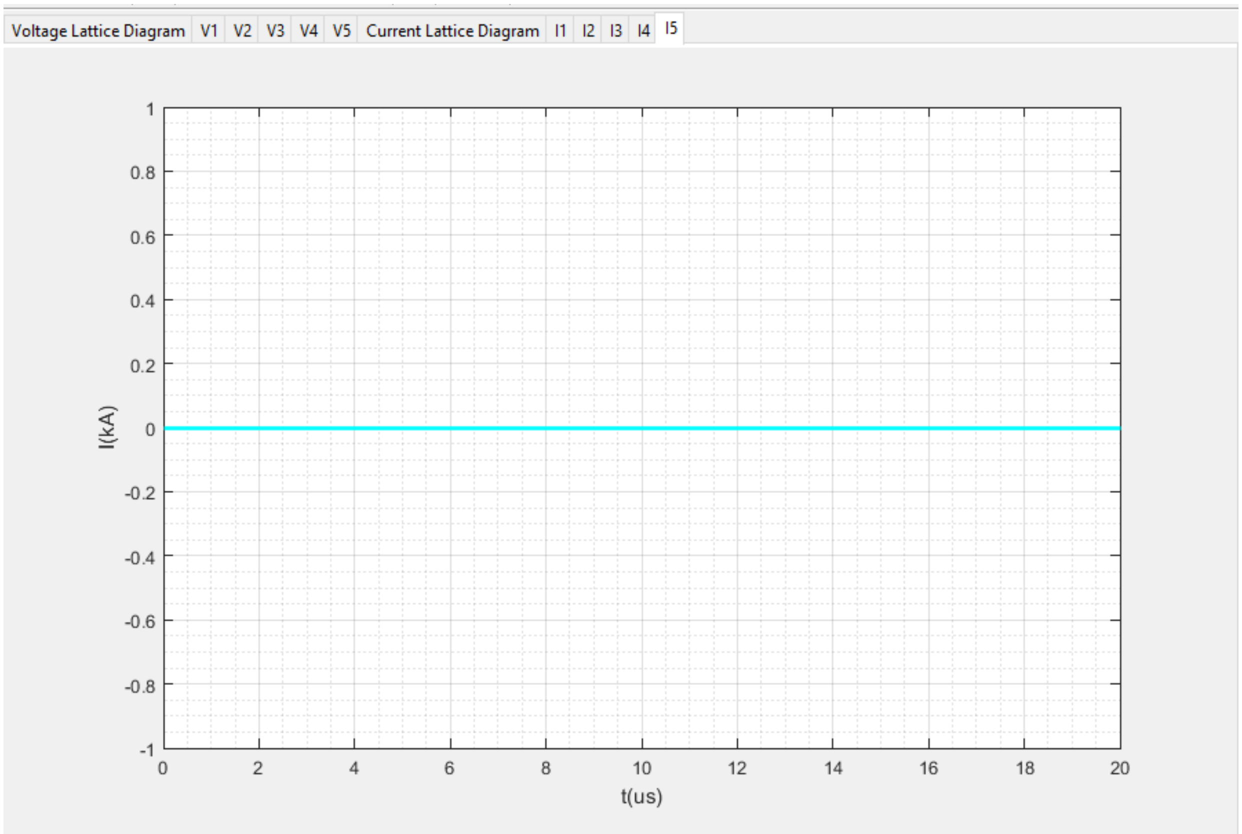


Voltage Lattice Diagram
V1
V2
V3
V4
V5
Current Lattice Diagram
I1
I2
I3
I4
I5
× 10^-3
I(kA)
t(us)


Voltage Lattice Diagram
V1
V2
V3
V4
V5
Current Lattice Diagram
I1
I2
I3
I4
I5
× 10^-3
I(kA)
t(us)
0
2
4
6
8
10
12
14
16
18
20
1
3
5
7
9


Voltage Lattice Diagram
V1
V2
V3
V4
V5
Current Lattice Diagram
I1
I2
I3
I4
I5
× 10^-3
I(kA)
t(us)
0
2
4
6
8
10
12
14
16
18
20
1
3
5
7

High Voltage Project




```
        This repository contains various MATLAB codes from different
disciplines in electrical engineering such as:
1- Robotics: 3DOF delta robot workspace analysis,
              Controlling robotic car with MATLAB and Arduino
              Using SimMechanics to simulate a humanoid.
2- High Voltage: General code for plotting Lattice Diagram of
transmitting waves in transmission lines.
3- Power Flow Analysis: using two techniques (Gauss-Seidel technique,
Newton-Raphson Technique).
4- Electric machines: Simulation of different types of electric motors
and generators in various conditions and study cases.
5- Power Electronics: Simulation of some power electronics circuits using
Simulink.
```