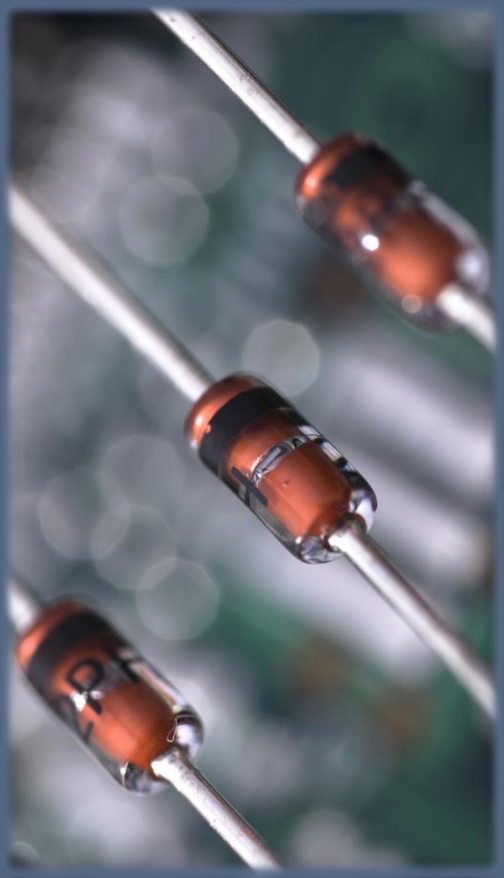
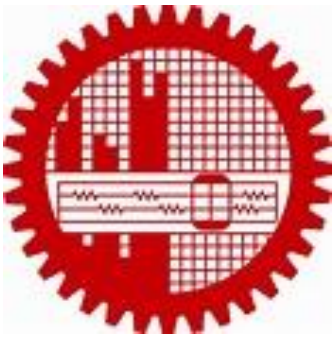


Simulating 4 Models Of Diodes

A MATLAB PROJECT

EEE 212 - Numerical Analysis Laboratory

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY



Department of Electrical and Electronics Engineering

Project title:

Simulating 4 Diode models(exponential,piecewise- linear, constant voltage drop,ideal diode model) Using MATLAB

Submitted to:-

Dr. Hafiz Imtiaz

Associate Professor

Electrical & Electronics Engineering (EEE),BUET

Mohammad Ali

Lecturer

Electrical and Electronics Engineering(EEE),BUET

Submitted by:-

Sayba Kamal Orni 2006009

Adib As-Ad Tonmoy 2006010

Group:-05 Level:-2 Term:-1

Documentation

Project Title:

Simulating 4 diode models (exponential, piece-wise linear, constant voltage drop, ideal model) using MATLAB.

Objective:

1. Netlist of the circuit will be taken as input through a GUI.
2. User can choose any of the 4 diode models.
3. The output will show which diodes are in forward bias and which are in reverse.
4. The output will also show the diode voltage and diode current according to the chosen model.
5. Input can be AC/DC.
6. Should be able to show plot of any node voltage or branch current.

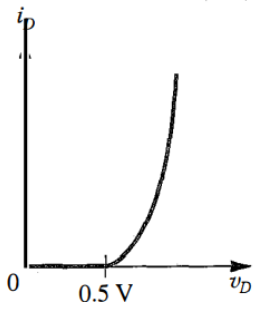
Relevant theory:

A diode is an electronic device that has 2 terminals positive terminals called an anode and a negative terminal called a cathode. Currently, almost every electronic devices use a diode.

Diode Models

- For circuit analysis and calculation of different parameters of diodes, mathematic expressions are formed called **diode models** in electronics.
- As we know V-I curve of a diode is not linear. For the understanding of practical behavior of diode simpler circuit models of diode are needed.

Exponential



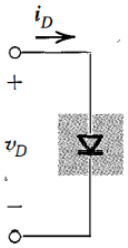
$$i_D = I_S e^{v_D/nV_T}$$

$$v_D = 2.3nV_T \log\left(\frac{i_D}{I_S}\right)$$

$$V_{D2} - V_{D1} = 2.3nV_T \log\left(\frac{I_{D2}}{I_{D1}}\right)$$

$2.3nV_T = 60$ mV for $n = 1$

$2.3nV_T = 120$ mV for $n = 2$



$I_S = 10^{-12}$ A to 10^{-15} A, depending on junction area

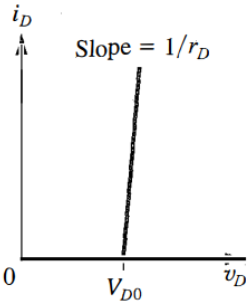
$V_T \cong 25$ mV

$n = 1$ to 2

Physically based and remarkably accurate model

Useful when accurate analysis is needed

Piecewise-linear (battery-plus-resistance)

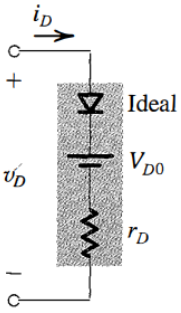


For $v_D \le V_{D0}$:

$i_D = 0$

For $v_D \ge V_{D0}$:

$$i_D = \frac{1}{r_D}(v_D - V_{D0})$$

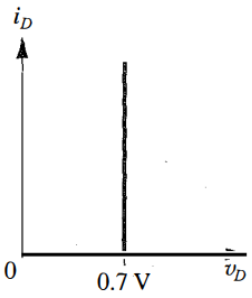


Choice of V_{D0} and r_D is determined by the current range over which the model is required.

For the amount of work involved, not as useful as the constant-voltage-drop model.

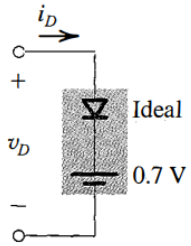
Used only infrequently.

Constant-voltage-drop (or the "0.7-V model")



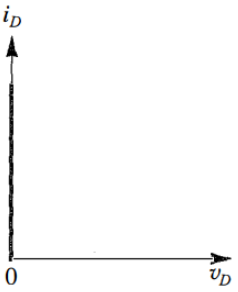
For $i_D > 0$:

$v_D = 0.7$ V



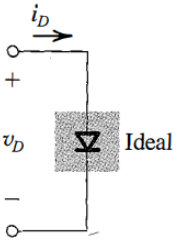
Easy to use and very popular for the quick, hand analysis that is essential in circuit design.

Ideal-diode



For $i_D > 0$:

$v_D = 0$



Good for determining which diodes are conducting and which are cutoff in a multiple-diode circuit.

Good for obtaining very approximate values for diode currents, especially when the circuit voltages are much greater than V_D.

Methodology:

Modified nodal analysis (MNA) method:

Solving a set of equations that represents a circuit is straightforward, if not always easy. However, developing that set of equations is not so easy. The two commonly taught methods for forming a set of equations are the node voltage (or nodal) method and the loop-current (or mesh) method.

However, a third method, Modified Nodal Analysis, that has some unique benefits. Among its benefits is the fact that it lends itself to algorithmic solution -- the ultimate goal of these pages is to describe how to use a MATLAB program for generating a set of equations representing the circuit that can be solved.

MNA often results in larger systems of equations than the other methods, but is easier to implement algorithmically on a computer which is a substantial advantage for automated solution. To use modified nodal analysis we have to write one equation for each node not attached to a voltage source (as in standard nodal analysis), and we augment these equations with an equation for each voltage source.

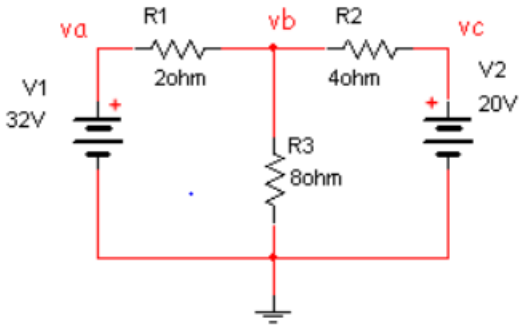
The steps followed in this method:

1. Selecting a reference node (usually ground) and naming the remaining n-1 nodes. Also labeling currents through each current source.
2. Assigning a name to the current through each voltage source. We will use the convention that the current flows from the positive node to the negative node of the source.
3. Applying Kirchoff's current law to each node. We will take currents out of the node to be positive.
4. Writing an equation for the voltage each voltage source.
5. Solving the system of *n-1* unknowns.

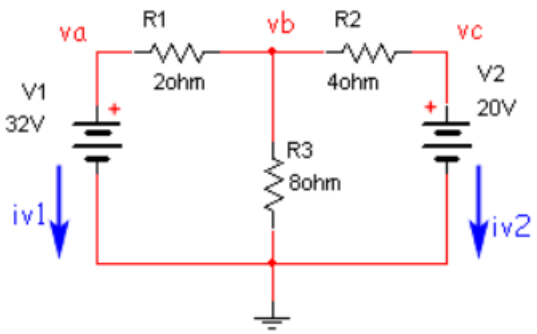
The benefit of using this method:

- In usual nodal analysis, we consider node currents but the problem arises when there is presence of supernodes in the circuit, as we cannot determine the current through voltage source using the equation ΔV/R=ΔI. That's why we introduce a new variable that represents the current through each voltage source. This is the basis of modified nodal analysis.
- By using this method, we can write a complete equation of a single node.

Let's consider the circuit shown below:



Applying step 2 (currents through the voltage sources with current from positive node to negative node):



Applying step 3 (with positive currents out of the node):

$$Node\ a: \quad i_{v1} + \frac{v_a - v_b}{R_1} = 0$$
$$Node\ b: \quad \frac{v_b - v_a}{R_1} + \frac{v_b}{R_3} + \frac{v_b - v_c}{R_2} = 0$$
$$Node\ c: \quad i_{v2} + \frac{v_c - v_b}{R_2} = 0$$

Applying step 4:

$$v_a = V_1$$
$$v_c = V_2$$

Applying step 5:

$$i_{v1} + \frac{v_a - v_b}{R_1} = 0$$

$$\frac{v_b - v_a}{R_1} + \frac{v_b}{R_3} + \frac{v_b - v_c}{R_2} = 0$$

$$i_{v2} + \frac{v_c - v_b}{R_2} = 0$$

$$v_a = V_1$$

$$v_c = V_2$$

or

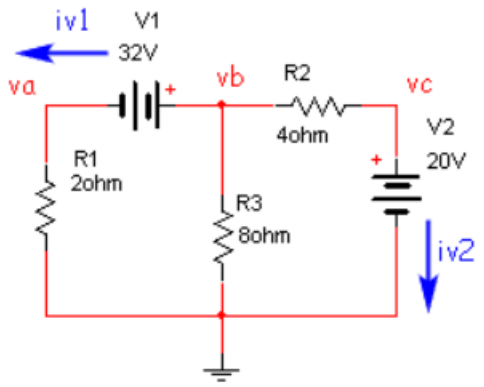
$$\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 1 & 0 \\ -\frac{1}{R_1} & \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} & -\frac{1}{R_2} & 0 & 0 \\ 0 & -\frac{1}{R_2} & \frac{1}{R_2} & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \\ i_{v1} \\ i_{v2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ V1 \\ V2 \end{bmatrix}$$

Now all that is left is to solve the 5x5 set of equations . Solving the 5x5 equation is difficult by hand, but not so with a computer.

Observations about MNA:

By examining the matrix equations that resulted from the application of the MNA method, several patterns become apparent that we can use to develop an algorithm. All of the circuits resulted in an equation of the form: Ax=z

Let's take an example:



This circuit had 3 nodes and 2 voltage sources (n=3, m=2). The resulting matrix is shown below:

$$\begin{bmatrix} \frac{1}{R_1} & 0 & 0 & -1 & 0 \\ 0 & \frac{1}{R_2}+\frac{1}{R_3} & -\frac{1}{R_2} & 1 & 0 \\ 0 & -\frac{1}{R_2} & \frac{1}{R_2} & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \\ i_{v1} \\ i_{v2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ V1 \\ V2 \end{bmatrix}$$

MNA applied to a circuit with only passive elements (resistors) and independent current and voltage sources results in a matrix equation of the form:

Ax=z

For a circuit with n nodes and m independent voltage sources:

- The **A** matrix:
  - is $(n+m)\times(n+m)$ in size, and consists only of known quantities.
  - the $n\times n$ part of the matrix in the upper left:
    - has only passive elements
    - elements connected to ground appear only on the diagonal
    - elements not connected to ground are both on the diagonal and off-diagonal terms.
  - the rest of the **A** matrix (not included in the $n\times n$ upper left part) contains only 1, -1 and 0 (other values are possible if there are dependent current and voltage sources; We have not considered these cases.)
- The **x** matrix:
  - is an $(n+m)\times 1$ vector that holds the unknown quantities (node voltages and the currents through the independent voltage sources).
  - the top n elements are the n node voltages.

    - the bottom *m* elements represent the currents through the *m* independent voltage sources in the circuit.
- The **z** matrix:
    - is an $(n+m)\times 1$ vector that holds only known quantities
    - the top *n* elements are either zero or the sum and difference of independent current sources in the circuit.
    - the bottom *m* elements represent the *m* independent voltage sources in the circuit.

The circuit is solved by a simple matrix manipulation:

$x=A^{-1}z$

Though this may be difficult by hand, it is straightforward and so is easily done by computer.

Generating the MNA matrices

There are three matrices we need to generate, the **A** matrix, the **x** matrix and the **z** matrix. Each of these will be created by combining several individual sub-matrices.

The A matrix

The **A** matrix will be developed as the combination of 4 smaller matrices, **G**, **B**, **C**, and **D**.

A=[G B;C D]

The **A** matrix is $(m+n)\times(m+n)$ (*n* is the number of nodes, and *m* is the number of independent voltage sources) and:

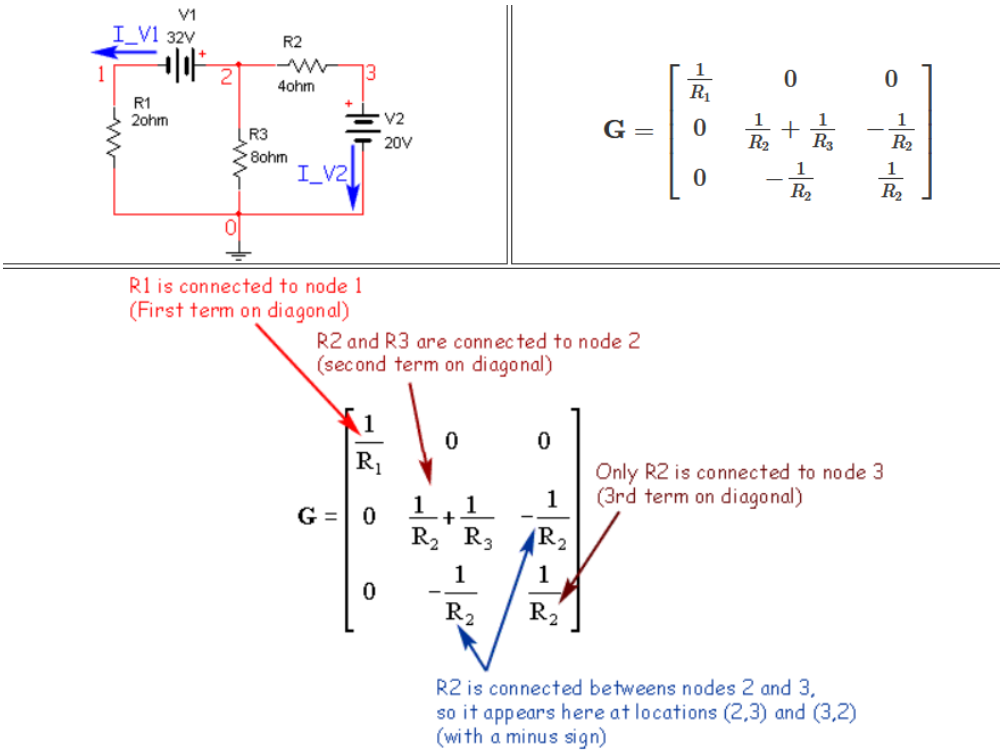
- the **G** matrix is $n\times n$ and is determined by the interconnections between the passive circuit elements (resistors)

- the **B** matrix is *n*×*m* and is determined by the connection of the voltage sources.
- the **C** matrix is *m*×*n* and is determined by the connection of the voltage sources. (**B** and **C** are closely related, particularly when only independent sources are considered).
- the **D** matrix is *m*×*m* and is zero if only independent sources are considered.

Rules for making the G matrix

The **G** matrix is an *n*×*n* matrix formed in two steps

1. Each element in the diagonal matrix is equal to the sum of the conductance (one over the resistance) of each element connected to the corresponding node. So the first diagonal element is the sum of conductances connected to node 1, the second diagonal element is the sum of conductances connected to node 2, and so on.
2. The off diagonal elements are the negative conductance of the element connected to the pair of corresponding node. Therefore a resistor between nodes 1 and 2 goes into the **G** matrix at location (1,2) and locations (2,1)

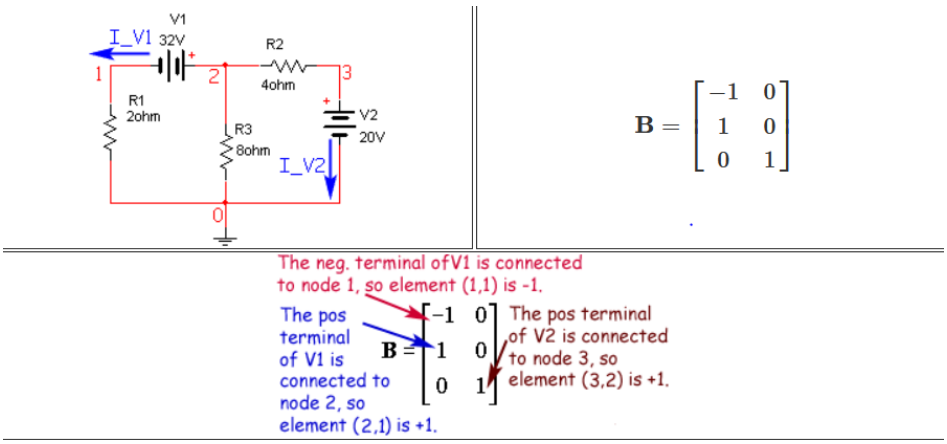


$$\mathbf{G} = \begin{bmatrix} \frac{1}{R_1} & 0 & 0 \\ 0 & \frac{1}{R_2} + \frac{1}{R_3} & -\frac{1}{R_2} \\ 0 & -\frac{1}{R_2} & \frac{1}{R_2} \end{bmatrix}$$

If an element is grounded, it will only have contribute to one entry in the **G** matrix -- at the appropriate location on the diagonal. If it is ungrounded it will contribute to four entries in the matrix -- two diagonal entries (corresponding to the two nodes) and two off-diagonal entries.

Rules for making the B matrix

The **B** matrix is an $n \times m$ matrix with only 0, 1 and -1 elements. Each location in the matrix corresponds to a particular voltage source (first dimension) or a node (second dimension). If the positive terminal of the ith voltage source is connected to node k, then the element (i,k) in the **B** matrix is a 1. If the negative terminal of the ith voltage source is connected to node k, then the element (i,k) in the **B** matrix is a -1. Otherwise, elements of the **B** matrix are zero.



f a voltage source is ungrounded, it will have two elements in the **B** matrix (a 1 and a -1 in the same column). If it is grounded it will only have one element in the matrix.

Rules for making the C matrix

The **C** matrix is an $m \times n$ matrix with only 0, 1 and -1 elements. Each location in the matrix corresponds to a particular node (first dimension) or voltage source (second dimension). If the positive terminal of the ith voltage source is connected to node k, then the element (k,i) in the **C** matrix is a 1. If the negative terminal of the ith voltage source is connected to node k, then the element (k,i) in the **C** matrix is a -1. Otherwise, elements of the **C** matrix are zero.

In other words, the **C** matrix is the transpose of the **B** matrix. (This is not the case when dependent sources are present.)

Rules for making the D matrix

The **D** matrix is an $m \times m$ matrix that is composed entirely of zeros.

The x matrix

The **x** matrix holds our unknown quantities and will be developed as the combination of 2 smaller matrices **v** and **j**. It is considerably easier to define than the **A** matrix.

x=[v ; j]

The **x** matrix is $(m+n) \times 1$ (n is the number of nodes, and m is the number of independent voltage sources) and:

- the **v** matrix is $n \times 1$ and hold the unknown voltages
- the **j** matrix is $m \times 1$ and holds the unknown currents through the voltage sources

The z matrix

The **z** matrix holds our independent voltage and current sources and will be developed as the combination of 2 smaller matrices **i** and **e**. It is quite easy to formulate.

z=[i ;e]

The **z** matrix is $(m+n) \times 1$ (n is the number of nodes, and m is the number of independent voltage sources) and:

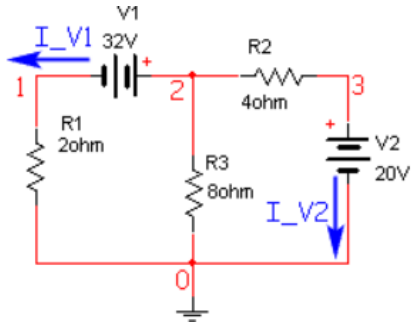
- the **i** matrix is *n*×1 and contains the sum of the currents through the passive elements into the corresponding node (either zero, or the sum of independent current sources).
- the **e** matrix is *m*×1 and holds the values of the independent voltage sources.

Rules for making the i matrix

The **i** matrix is an *n*×1 matrix with each element of the matrix corresponding to a particular node. The value of each element of **i** is determined by the sum of current sources into the corresponding node. If there are no current sources connected to the node, the value is zero.

Rules for making the e matrix

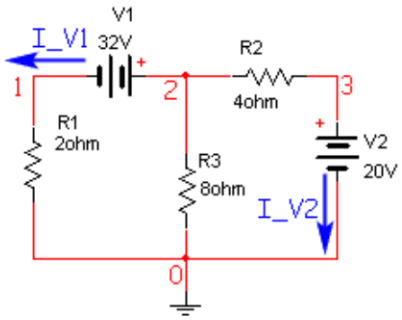
The **e** matrix is an *m*×1 matrix with each element of the matrix equal in value to the corresponding independent voltage source.



$$\mathbf{i} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} V1 \\ V2 \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ V1 \\ V2 \end{bmatrix}$$

Putting it Together

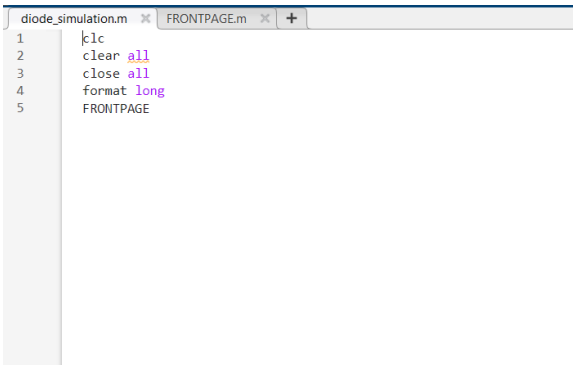
We can now write out the full matrix solutions for both cases that we have been developing.



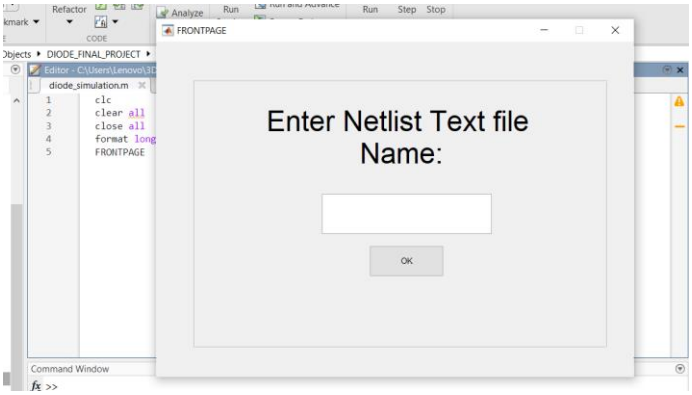
$$\begin{bmatrix} \frac{1}{R_1} & 0 & 0 & -1 & 0 \\ 0 & \frac{1}{R_2}+\frac{1}{R_3} & -\frac{1}{R_2} & 1 & 0 \\ 0 & -\frac{1}{R_2} & \frac{1}{R_2} & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v\_1 \\ v\_2 \\ v\_3 \\ i\_V1 \\ i\_V2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ V1 \\ V2 \end{bmatrix}$$

Procedure and code explanation:

Initialization and taking input:



As we run the function diode_simulation, the ‘FRONTPAGE’ gui is called.



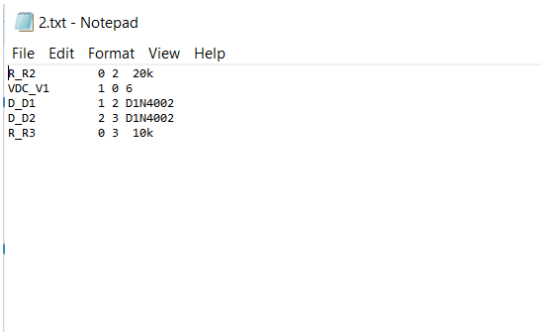
Here the name of the text file containing the netlist of the circuit is given as input.

In the netlist ,

- to represent resistance we use ‘R’ a
- to represent diodes we use ‘D’
- to represent DC voltage source we use ‘VDC_V’

- to represent AC voltage source we use ‘VSIN_V’

An example of netlist:



Functionalities of Frontpage:

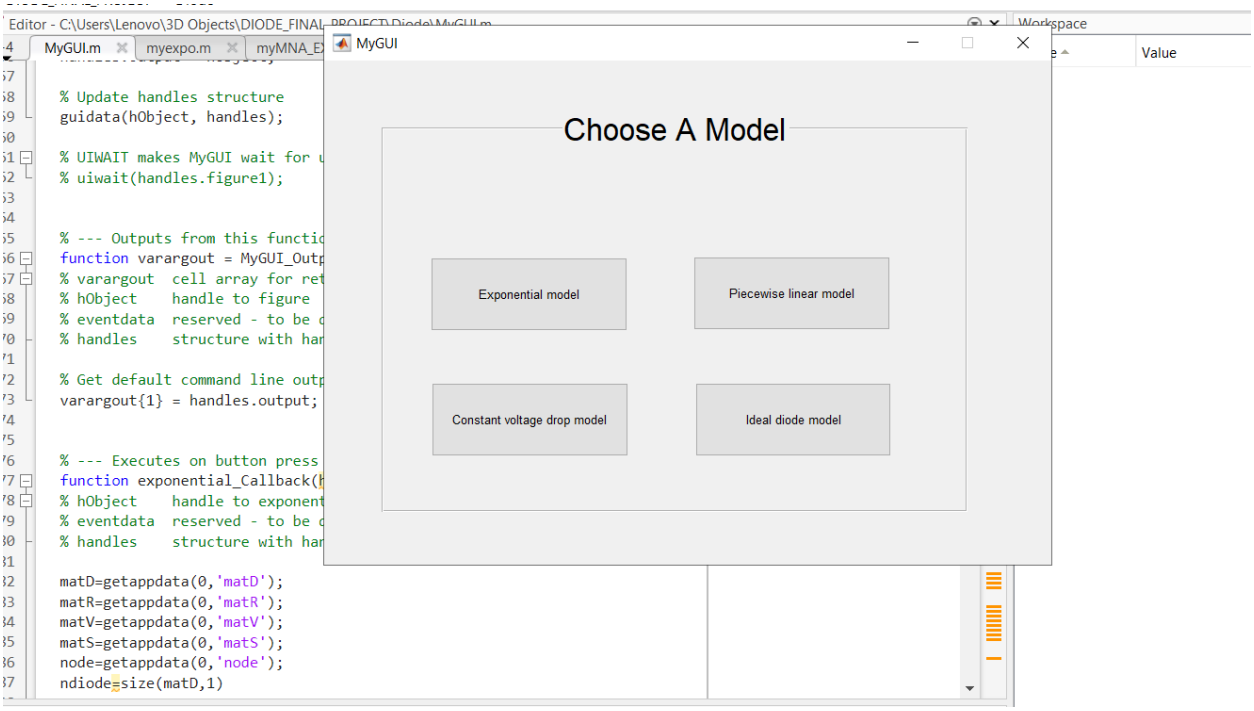
1. At first, the input was taken through netlist_file_reader1
2. The text file was read
3. matV, matS, matD ,matR are formed here.
4. In the netlist , the following replacements are made:

k	10^3
K	10^3
M	10^6
m	10^-3
u	10^-6
n	10^-9

```
%% my attempt on reading INPUT

fname=get(handles.netlist_file_reader1,'string');
    fname=strcat(fname,'.txt');
    fid = fopen(char(fname),'r');
    %-----------------------------------
    formatSpec = '%s';
    letterline = fgetl(fid);
    matV = zeros(1,4);
    matR = zeros(1,4);
    matD = zeros(1,3);
    matS = zeros(1,5); % ac er jonno matS=zeros(1,5)
    %-----------------------------------
    while ischar(letterline)
        if letterline(1)=='V'&& letterline(2)=='D'&& letterline(3)=='C'

            line = strrep(letterline,'VDC_V',' ');
            line = strrep(line,'k','e03');
            line = strrep(line,'M','e06');
            line = strrep(line,'m','e-03');
            line = strrep(line,'u','e-06');
            line = strrep(line,'K','e03');
            line = strrep(line,'u','e-09');
            val_line = str2num(line);
            matV = [matV ; val_line];
        elseif letterline(1)=='V'&& letterline(2)=='S'&& letterline(3)=='I'&& letterline(4)=='N'
            line = strrep(letterline,'VSIN_V',' ');
            line = strrep(line,'k','e3');
            line = strrep(line,'M','e6');
            line = strrep(line,'m','e-3');
             line = strrep(line,'u','e-06');
            line = strrep(line,'K','e03');
            line = strrep(line,'u','e-09');
            val_line = str2num(line);
```

From pushbutton_netlist_input1_Callback, MyGUI is called.

Functionalities of MyGUI:



There are 4 pushbuttons in this GUI and these pushbuttons enable the user to select a specific model of operation for the diodes. Here in each callback function, necessary commands are written to get the value of node voltages and current through voltage sources and the diodes.

```
% --- Executes on button press in exponential.
function exponential_Callback(hObject, eventdata, handles)
% hObject    handle to exponential (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

matD=getappdata(0,'matD');
matR=getappdata(0,'matR');
matV=getappdata(0,'matV');
matS=getappdata(0,'matS');
node=getappdata(0,'node');
ndiode=size(matD,1)

check=nnz(matS)

if check==0


[stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=myexpo(matD,matR,matV,node)
% ans=myMNA_EXP(matD,matR,matV,node)
if ansnew==[0]
    ansnewgui=ans_from_func
else
    ansnewgui=ansnew

end


setappdata(0,'strI',stringI);
setappdata(0,'strV',stringV);
setappdata(0,'strErrI',stringErrI);
setappdata(0,'strErrV',stringErrV);
setappdata(0,'ansnewgui',ansnewgui);
setappdata(0,'node',node);
setappdata(0,'matD',matD);
setappdata(0,'matR',matR);
setappdata(0,'matV',matV);
setappdata(0,'diodecurr',diodecurr);
GUIoutDC

else %AC part
```

```
else %AC part

amp=matS(4);
w=matS(5);
period=2*3.1415/w
per=5*period
t=linspace(0,per,100)
matAC=amp*sin(w*t)

len=length(t)

ACvolts=zeros(len,node)
ACsourceCurr=zeros(len,1)
diodecurrent=zeros(len,ndiode)

for j=1:len
    matS(4)=matAC(j)
    [stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=myexpo(matD,matR,matS,node)
    if ansnew==[0]
    ansnewgui=ans_from_func
    else
    ansnewgui=ansnew

    end

    ACvolts(j,:)=ansnewgui(1:node)
    vnode=node+1
%       ACsourceCurr(i)=ansnewgui(vnode,1)

ACsourceCurrval=ansnewgui(node+1)
ACsourceCurr(j)=ACsourceCurrval
diodecurrent(j,:)=diodecurr(1:ndiode)
end
ACvolts
ACsourceCurr
diodecurrent

setappdata(0,'ACvolts',ACvolts);
setappdata(0,'t',t);
setappdata(0,'node',node);
setappdata(0,'matD',matD);
setappdata(0,'matR',matR);
setappdata(0,'matV',matV);
 setappdata(0,'ACsourceCurr',ACsourceCurr);
 setappdata(0,'diodecurrent',diodecurrent);
GUIoutAC
```

Here the function myexpo.m is called which determines the necessary values of voltage and current and these values are later used in other GUIs to show output.

Similarly the commands are written in the block of other model's pushbutton callback.

```
function piecewiselinear_Callback(hObject, eventdata, handles)
% hObject    handle to piecewiselinear (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

matD=getappdata(0,'matD');
matR=getappdata(0,'matR');
matV=getappdata(0,'matV');
node=getappdata(0,'node');
matS=getappdata(0,'matS');
ndiode=size(matD,1)


check=nnz(matS)

if check==0
[stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=mypwl(matD,matR,matV,node)
% ans=myMNA_PWL(matD,matR,matV)
if ansnew==[0]
    ansnewgui=ans_from_func
else
    ansnewgui=ansnew

end


setappdata(0,'strI',stringI);
setappdata(0,'strV',stringV);
setappdata(0,'strErrI',stringErrI);
setappdata(0,'strErrV',stringErrV);
setappdata(0,'ansnewgui',ansnewgui);
setappdata(0,'node',node);
setappdata(0,'matD',matD);
setappdata(0,'matR',matR);
setappdata(0,'matV',matV);
setappdata(0,'diodecurr',diodecurr);

GUIoutDC
```

```
else %AC part

amp=matS(4);
w=matS(5);
period=2*3.1415/w
per=5*period
t=linspace(0,per,100)
matAC=amp*sin(w*t)

len=length(t)

ACvolts=zeros(len,node)
ACsourceCurr=zeros(len,1)
diodecurrent=zeros(len,ndiode)

for j=1:len
    matS(4)=matAC(j)
    [stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=mypwl(matD,matR,matS,node)
    if ansnew==[0]
    ansnewgui=ans_from_func
    else
    ansnewgui=ansnew

    end

    ACvolts(j,:)=ansnewgui(1:node)
     ACsourceCurr(j)=ansnewgui(node+1)
     diodecurrent(j,:)=diodecurr(1:ndiode)


end
ACvolts
 ACsourceCurr
 diodecurrent

setappdata(0,'ACvolts',ACvolts);
setappdata(0,'t',t);
setappdata(0,'node',node);
setappdata(0,'matD',matD);
setappdata(0,'matR',matR);
setappdata(0,'matV',matV);
setappdata(0,'ACsourceCurr',ACsourceCurr);
setappdata(0,'diodecurrent',diodecurrent);
GUIoutAC


end
```

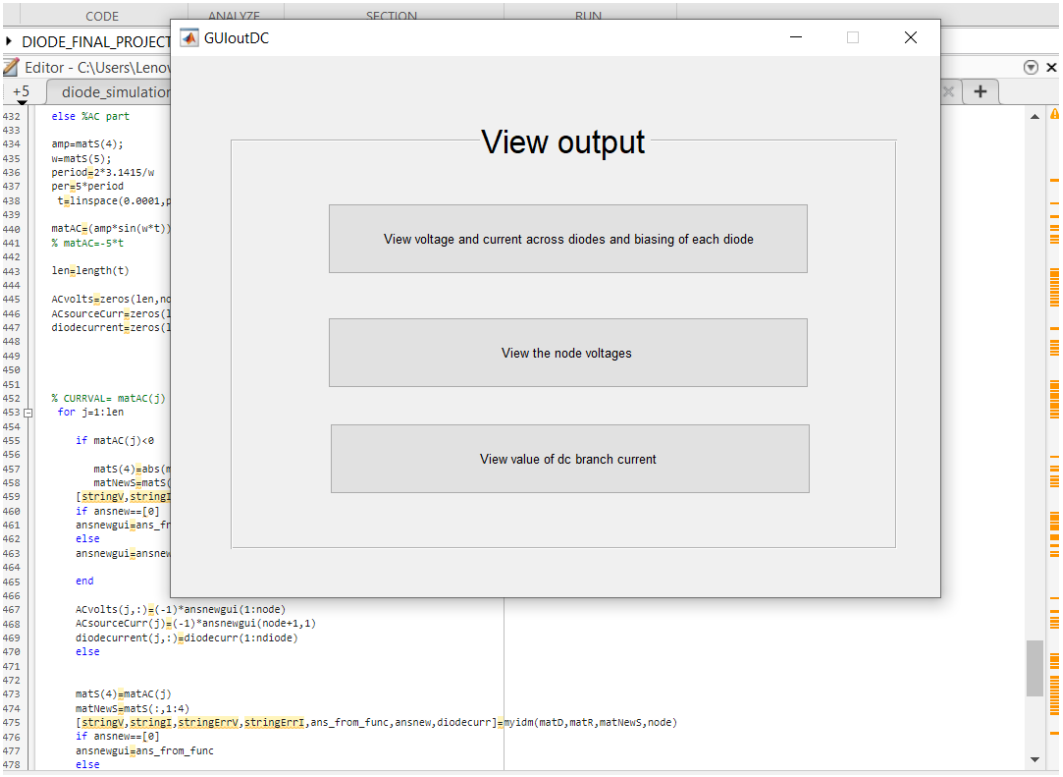
```
% --- Executes on button press in constantvoltagedrop.
function constantvoltagedrop_Callback(hObject, eventdata, handles)
% hObject    handle to constantvoltagedrop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
matD=getappdata(0,'matD');
matR=getappdata(0,'matR');
matV=getappdata(0,'matV');
node=getappdata(0,'node');
ndiode=size(matD,1)


matS=getappdata(0,'matS');

check=nnz(matS)

if check==0


[stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=mycvd(matD,matR,matV,node)
% ans=myMNA_CVD(matD,matR,matV)
% ansofcvd=myMNA_CVD(matD,matR,matV)
if ansnew==[0]
    ansnewgui=ans_from_func
else
    ansnewgui=ansnew

end


setappdata(0,'strI',stringI);
setappdata(0,'strV',stringV);
setappdata(0,'strErrI',stringErrI);
setappdata(0,'strErrV',stringErrV);
setappdata(0,'ansnewgui',ansnewgui);
setappdata(0,'node',node);
setappdata(0,'matD',matD);
setappdata(0,'matR',matR);
setappdata(0,'matV',matV);
setappdata(0,'diodecurr',diodecurr);

GUIoutDC

else %AC part
```

```
else %AC part



amp=matS(4);
w=matS(5);
period=2*3.1415/w
per=5*period
t=linspace(0,per,100)
matAC=amp*sin(w*t)

len=length(t)

ACvolts=zeros(len,node)
ACsourceCurr=zeros(len,1)
diodecurrent=zeros(len,ndiode)

for j=1:len
    matS(4)=matAC(j)
    matNewS=matS(:,1:4)



    [stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=mycvd(matD,matR,matNewS,node)
    if ansnew==[0]
    ansnewgui=ans_from_func
    else
    ansnewgui=ansnew

    end

    ACvolts(j,:)=ansnewgui(1:node)
     ACsourceCurr(j)=ansnewgui(node+1)
     diodecurrent(j,:)=diodecurr(1:ndiode)


end
ACvolts
ACsourceCurr
diodecurrent

setappdata(0,'ACvolts',ACvolts);
setappdata(0,'t',t);
setappdata(0,'node',node);
setappdata(0,'matD',matD);
setappdata(0,'matR',matR);
setappdata(0,'matV',matV);
setappdata(0,'ACsourceCurr',ACsourceCurr);
setappdata(0,'diodecurrent',diodecurrent);
GUIoutAC
```

```
% --- Executes on button press in ideal.
function ideal_Callback(hObject, eventdata, handles)
% hObject    handle to ideal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


matD=getappdata(0,'matD');
matR=getappdata(0,'matR');
matV=getappdata(0,'matV');
node=getappdata(0,'node');
ndiode=size(matD,1)

matS=getappdata(0,'matS');

check=nnz(matS)

if check==0

[stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=myidm(matD,matR,matV,node)
% ans=myMNA_IDM(matD,matR,matV)
if ansnew==[0]
    ansnewgui=ans_from_func
else
    ansnewgui=ansnew

end


setappdata(0,'strI',stringI);
setappdata(0,'strV',stringV);
setappdata(0,'strErrI',stringErrI);
setappdata(0,'strErrV',stringErrV);
setappdata(0,'ansnewgui',ansnewgui);
setappdata(0,'node',node);
setappdata(0,'matD',matD);
setappdata(0,'matR',matR);
setappdata(0,'matV',matV);
setappdata(0,'diodecurr',diodecurr);
GUIoutDC
```

```
else %AC part

amp=matS(4);
w=matS(5);
period=2*3.1415/w
per=5*period
 t=linspace(0.0001,per,100)

matAC=(amp*sin(w*t))
% matAC=-5*t

len=length(t)

ACvolts=zeros(len,node)
ACsourceCurr=zeros(len,1)
diodecurrent=zeros(len,ndiode)


% CURRVAL= matAC(j)
 for j=1:len

    if matAC(j)<0

       matS(4)=abs(matAC(j))
       matNewS=matS(:,1:4)
    [stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=myidmAC(matD,matR,matNewS,node)
    if ansnew==[0]
    ansnewgui=ans_from_func
    else
    ansnewgui=ansnew

    end

    ACvolts(j,:)=(-1)*ansnewgui(1:node)
    ACsourceCurr(j)=(-1)*ansnewgui(node+1,1)
    diodecurrent(j,:)=diodecurr(1:ndiode)
    else


    matS(4)=matAC(j)
    matNewS=matS(:,1:4)
    [stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=myidm(matD,matR,matNewS,node)
    if ansnew==[0]
    ansnewgui=ans_from_func
    else
    ansnewgui=ansnew

    end
```

Functionalities of GUIoutDC:



As for DC circuits, we will show the value of node voltages and branch currents. Hence, GUIoutDC will be called in case of DC input, and it will provide the user with option to:

- View voltage and current across diodes and biasing of each diode:

  For this ,MyGUI3 will be called in pushbutton callback

- View the node voltages

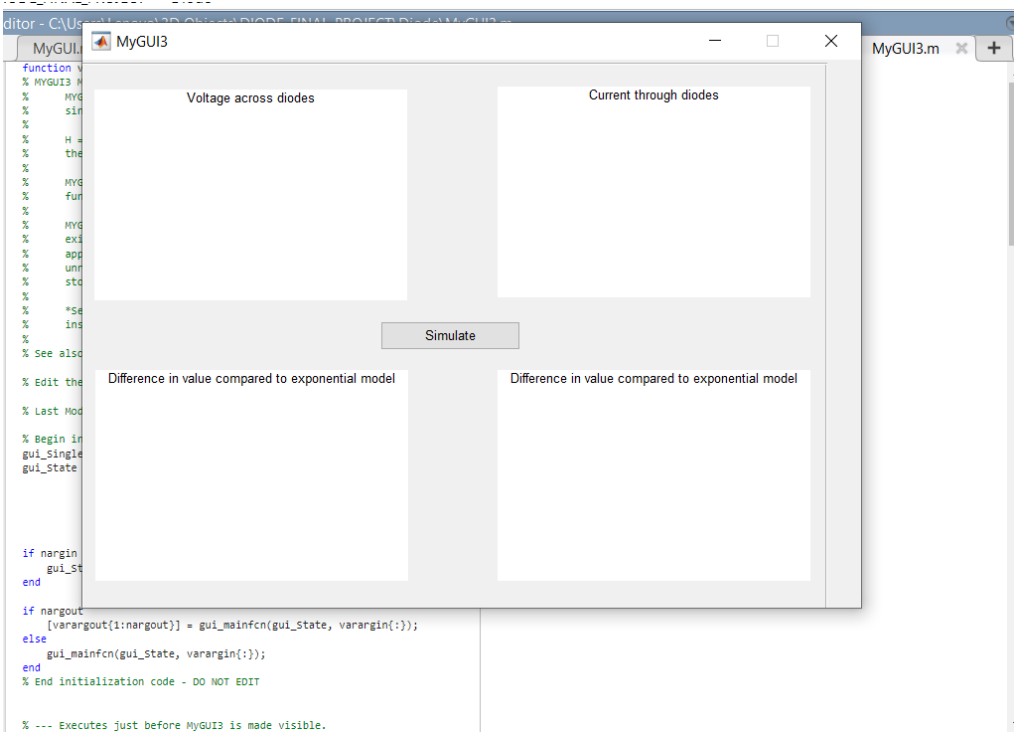
  For this , MyGUInodeDC will be called in pushbutton callback

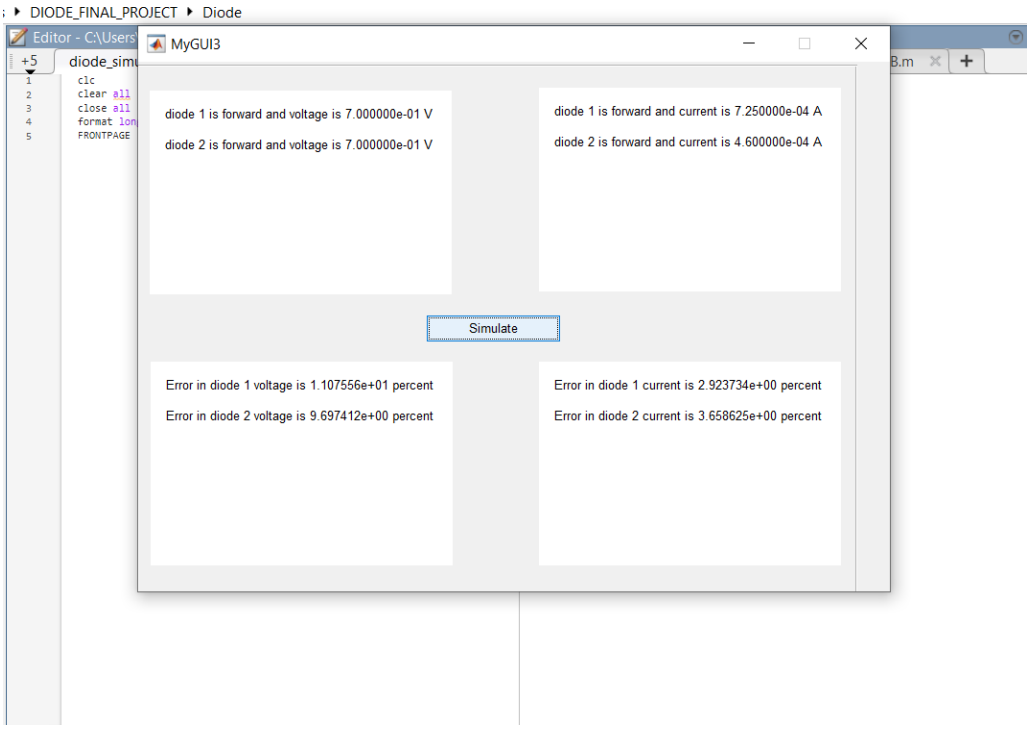
- View the value of DC branch current

  For this , MyGUIcurrentDC will be called in pushbutton callback

```
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUIoutDC_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

MyGUI3




% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

MyGUInodeDC



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

MyGUIcurrentDC
```

Functionalities of MyGUI3:

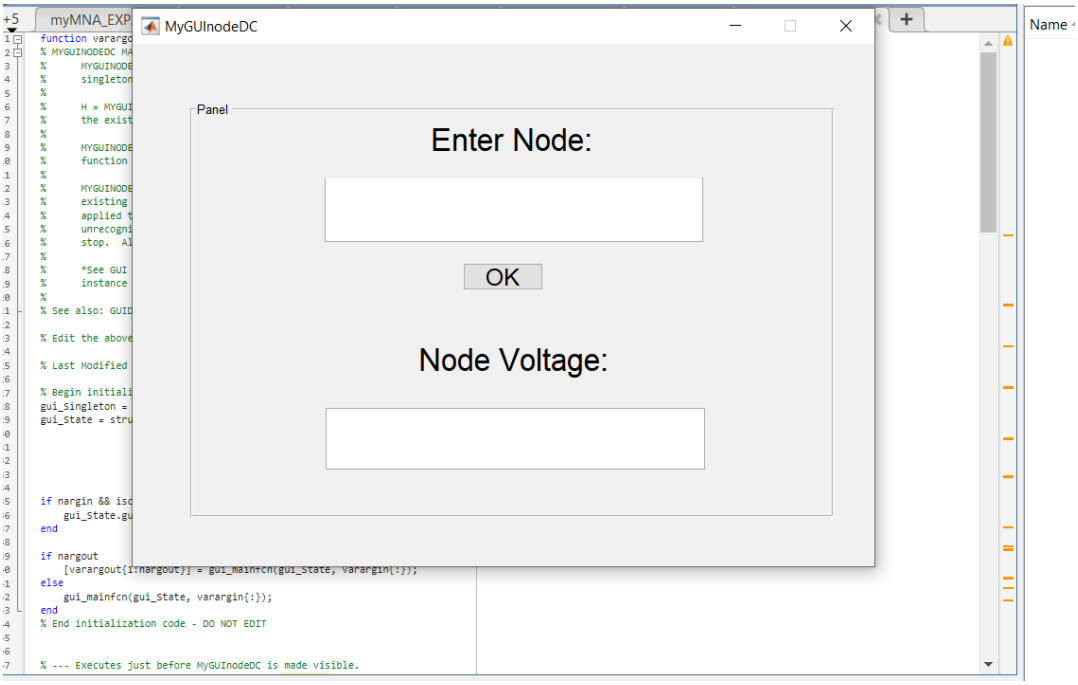

MyGUI3
Voltage across diodes
Current through diodes
Simulate
Difference in value compared to exponential model
Difference in value compared to exponential model
MyGUI3.m

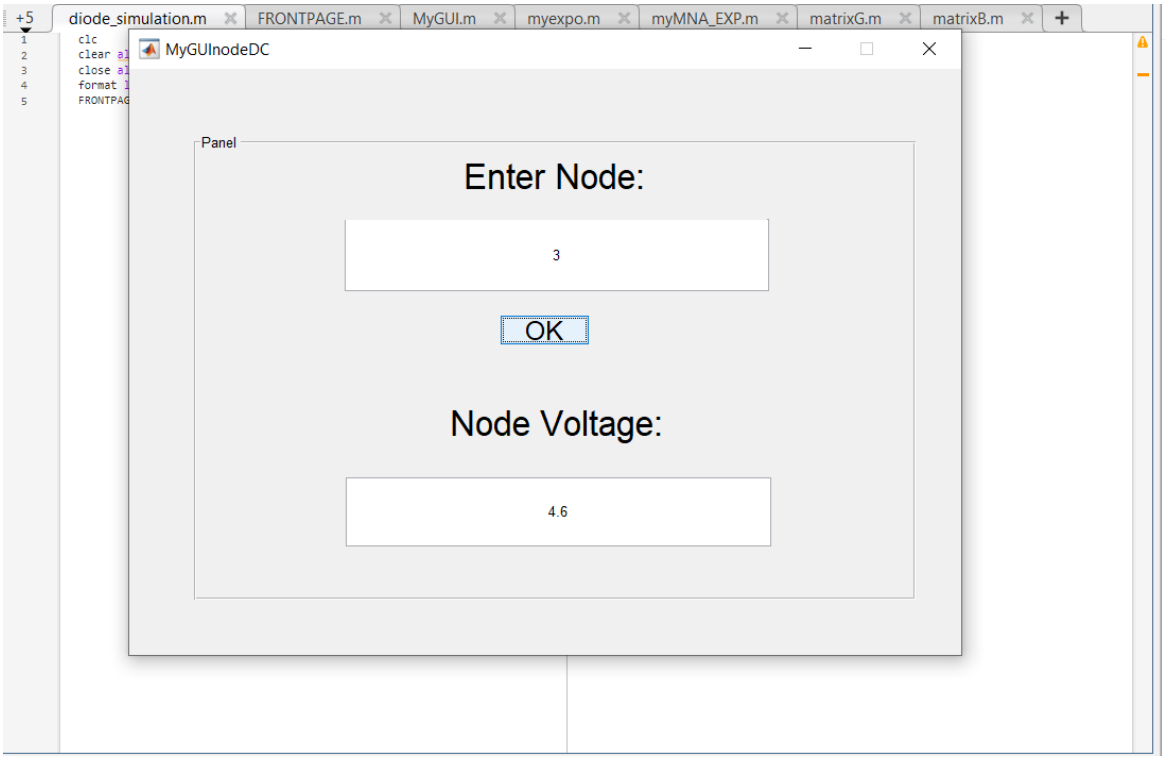

DIODE_FINAL_PROJECT ▸ Diode
MyGUI3
diode 1 is forward and voltage is 7.000000e-01 V
diode 2 is forward and voltage is 7.000000e-01 V
diode 1 is forward and current is 7.250000e-04 A
diode 2 is forward and current is 4.600000e-04 A
Simulate
Error in diode 1 voltage is 1.107556e+01 percent
Error in diode 2 voltage is 9.697412e+00 percent
Error in diode 1 current is 2.923734e+00 percent
Error in diode 2 current is 3.658625e+00 percent

```
function currenterr_Callback(hObject, eventdata, handles)
% hObject    handle to currenterr (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of currenterr as text
%        str2double(get(hObject,'String')) returns contents of currenterr as a double


% --- Executes during object creation, after setting all properties.
function currenterr_CreateFcn(hObject, eventdata, handles)
% hObject    handle to currenterr (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in simulate_pushbutton2.
function simulate_pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to simulate_pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


strV=getappdata(0,'strV');
strI=getappdata(0,'strI');
strErrV=getappdata(0,'strErrV');
strErrI=getappdata(0,'strErrI');
set(handles.outputvolt,'string',strV);
set(handles.outputcurr,'string',strI);
set(handles.volterr,'string',strErrV);
set(handles.currenterr,'string',strErrI);
```

Here in MyGUI3, 4 strings are shown as output, which are: voltage across diodes, current through diodes, difference (error) in value of voltages across diodes compared to exponential model and difference (error) in value of diode currents compared to exponential model.

Functionalities of MyGUInodeDC:

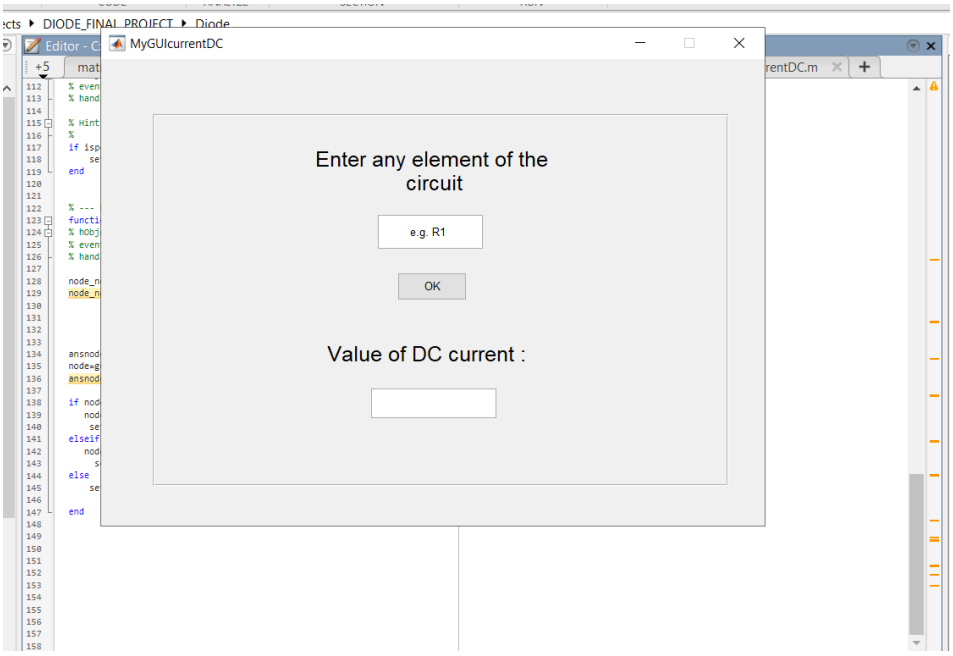


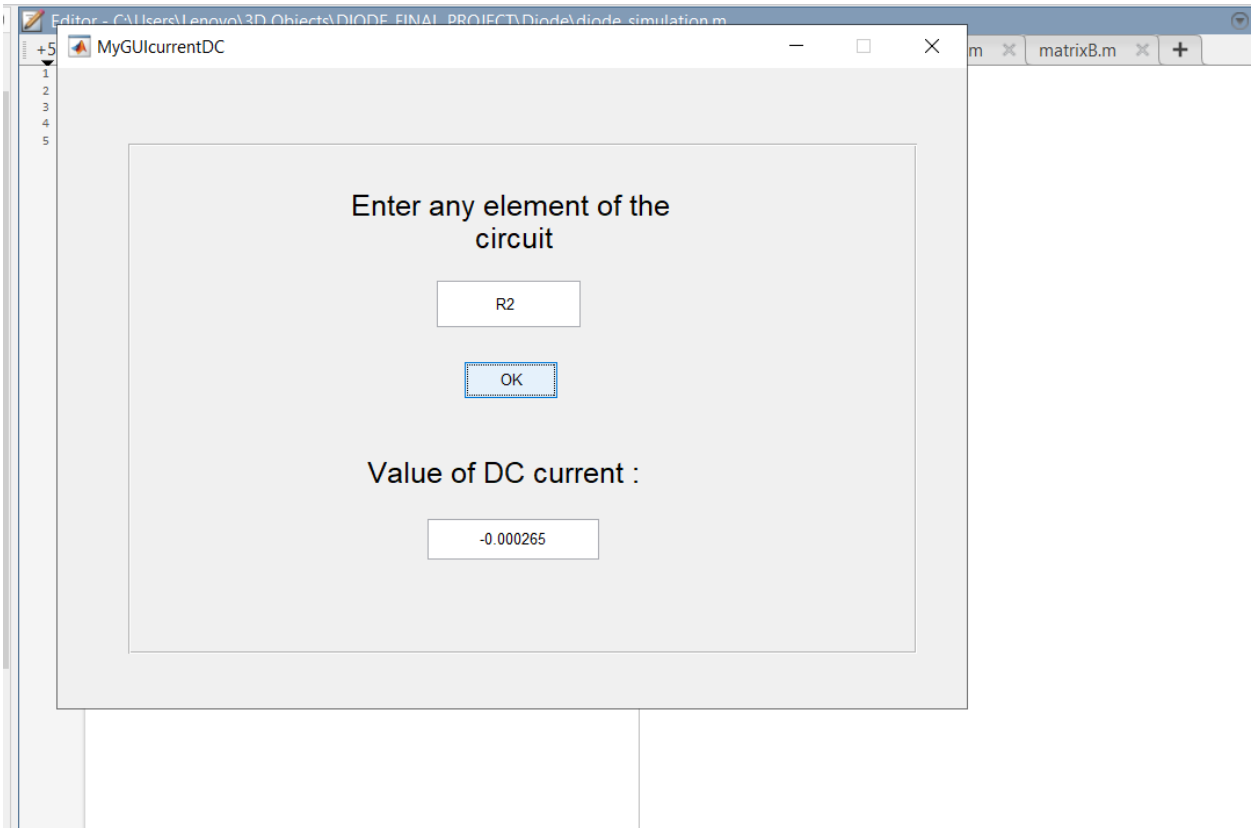


```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

node_no=str2num(get(handles.edit1,'string'));
node_no



ansnode=getappdata(0,'ansnewgui');
node=getappdata(0,'node');
ansnode

if node_no==0
   node_voltage=0
    set(handles.edit2,'string',node_voltage);
elseif (node_no>0 && node_no<=node)
   node_voltage=ansnode(node_no,1)
     set(handles.edit2,'string',node_voltage);
else
    set(handles.edit2,'string','error!!');

end
```

Here, 'in pushbutton1_Callback', the node the user wants to find the voltage is taken as input and then the voltage of that respective node is shown as output.

Functionalities of MyGUIcurrentDC:



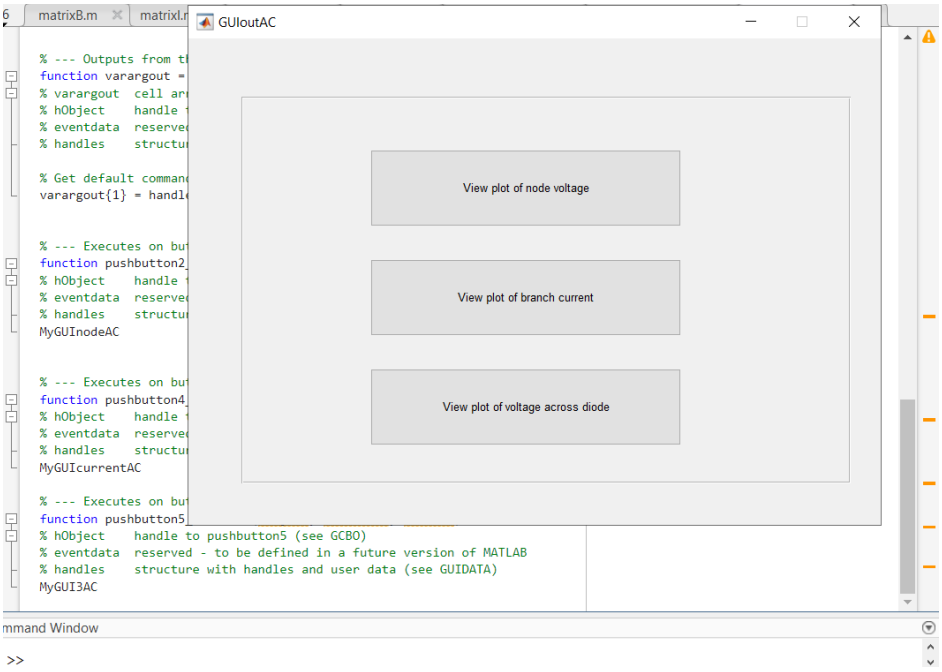


```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

element=get(handles.edit1,'string');
element_name=element(1)
element_id=str2num(element(2))

ansnode=getappdata(0,'ansnewgui');
node=getappdata(0,'node');
ansnode
matD=getappdata(0,'matD')
matR=getappdata(0,'matR')
matV=getappdata(0,'matV')
diodecurr=getappdata(0,'diodecurr')
% noR=size(matR,1)
node1=0
node2=0


 if element_name=='R'
        for i=1:size(matR,1)
            if matR(i,1)==element_id
                node1=matR(i,2)
                node2=matR(i,3)
                res=matR(i,4)
            end
         end
        if node1==0
            nodev1=0
            nodev2=ansnode(node2)
        end
        if node2==0
            nodev2=0
            nodev1=ansnode(node1)
        end
```

```
if element_name=='R'
        for i=1:size(matR,1)
            if matR(i,1)==element_id
                node1=matR(i,2)
                node2=matR(i,3)
                res=matR(i,4)
            end
        end
       if node1==0
           nodev1=0
           nodev2=ansnode(node2)
       end
       if node2==0
           nodev2=0
           nodev1=ansnode(node1)
       end
       if node1~=0 && node2~=0
           nodev1=ansnode(node1)
           nodev2=ansnode(node2)
       end
      current=(nodev1-nodev2)/res
      set(handles.edit2,'string',current);

elseif element_name=='V'

    current=(-1*ansnode(node+1,1));
    set(handles.edit2,'string',current);

elseif element_name=='D'
    current=diodecurr(element_id)
    set(handles.edit2,'string',current);
else
    set(handles.edit2,'string','ERROR!');


end
```

In this way, current across any element can be shown as output.

Functionalities of GUIoutAC:



As for AC circuits, we will show the plots of node voltages and branch currents. Hence, GUIoutAC will be called in case of AC input, and it will provide the user with option to:

- View plot of voltage difference across diodes

  For this ,MyGUI3AC will be called in pushbutton callback

- View the plots of node voltages

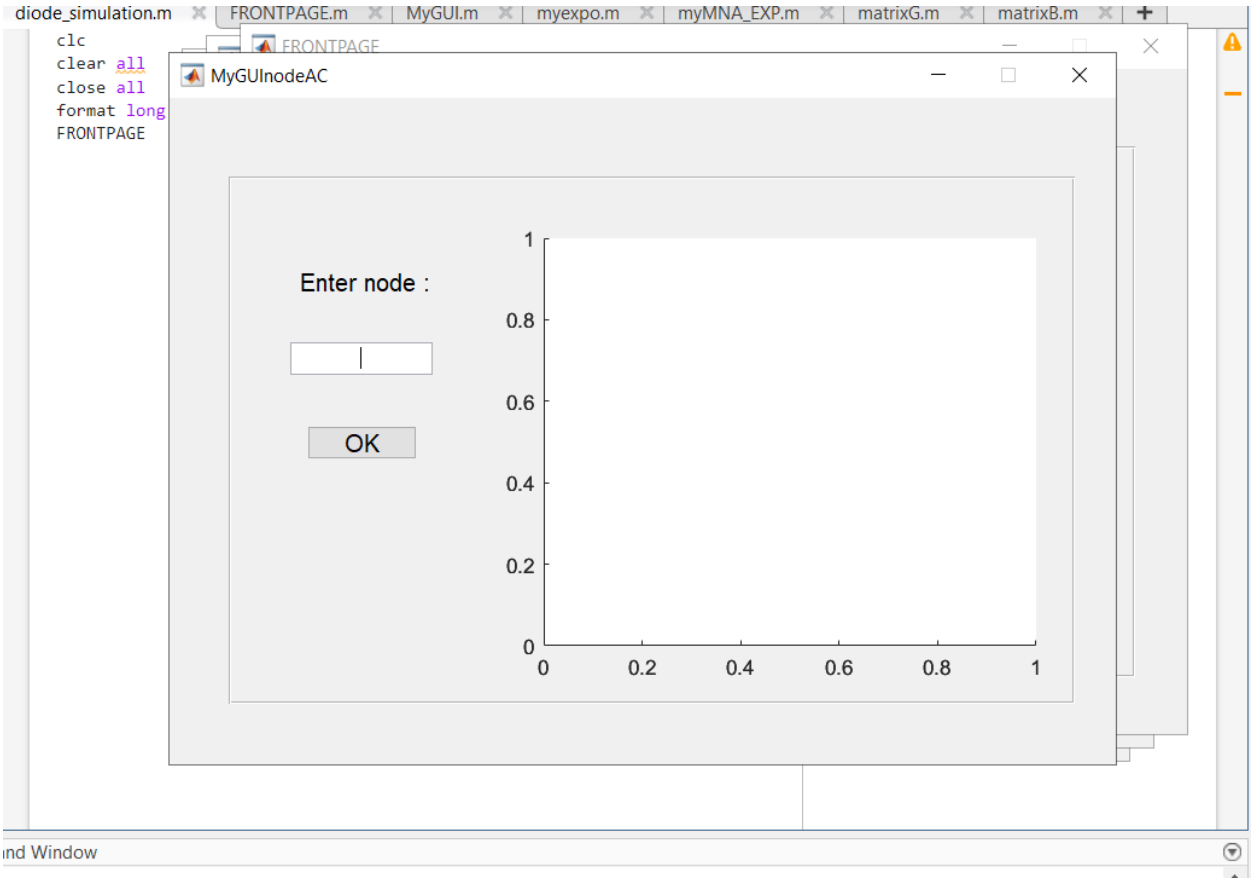
  For this , MyGUInodeAC will be called in pushbutton callback

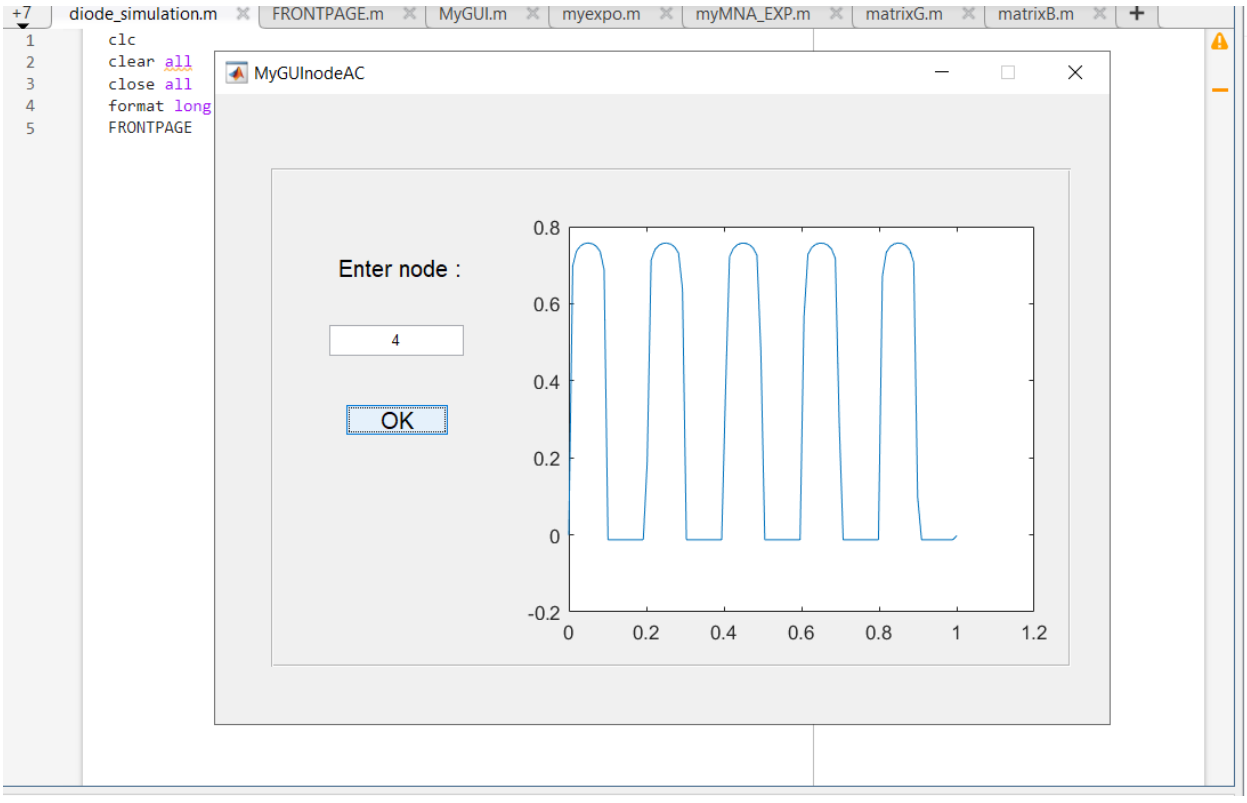
- View the plot of branch current, which means current through any element.

  For this , MyGUIcurrentAC will be called in pushbutton callback

```
% --- Outputs from this function are returned to the command line.
function varargout = GUIoutAC_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
MyGUInodeAC


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
MyGUIcurrentAC

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
MyGUI3AC
```

Functionalities of MyGUInodeAC:

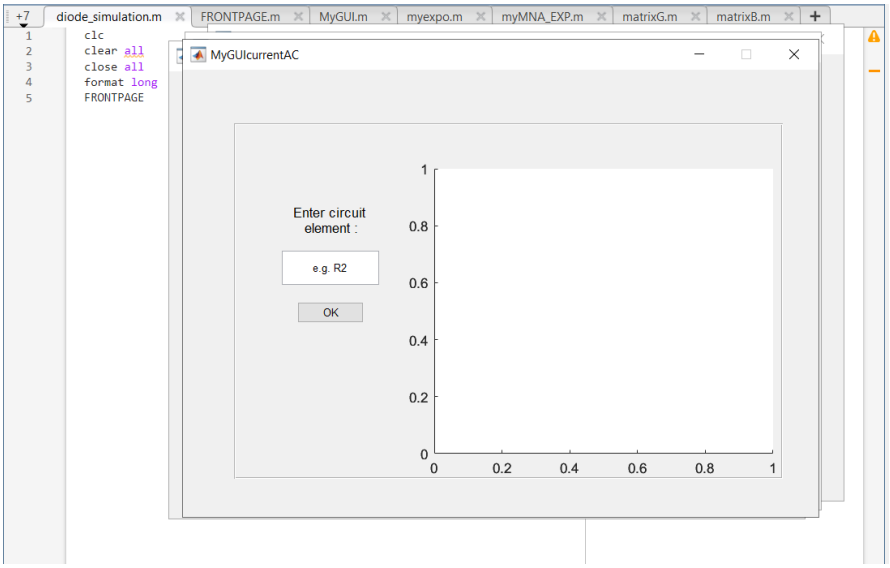

diode_simulation.m
FRONTPAGE.m
MyGUI.m
myexpo.m
myMNA_EXP.m
matrixG.m
matrixB.m
clc
clear all
close all
format long
FRONTPAGE
MyGUInodeAC
Enter node :
OK
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1

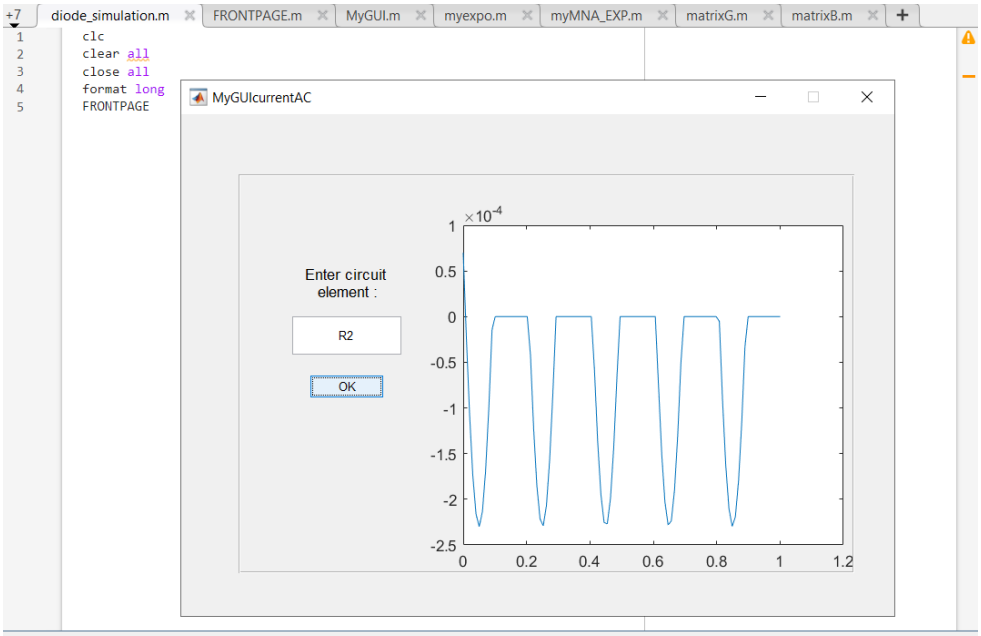

diode_simulation.m
FRONTPAGE.m
MyGUI.m
myexpo.m
myMNA_EXP.m
matrixG.m
matrixB.m
clc
clear all
close all
format long
FRONTPAGE
MyGUInodeAC
Enter node :
4
OK
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t=getappdata(0,'t')
ACvolts=getappdata(0,'ACvolts')
node_no=str2num(get(handles.edit1,'string'))
ACvolts(:,node_no)
plot(t,ACvolts(:,node_no))
```

Here, 'in pushbutton1_Callback', the node the user wants to find the voltage is taken as input and then the plot of node voltage of that respective node is shown as output.

Functionalities of MyGUIcurrentAC:


diode_simulation.m
FRONTPAGE.m
MyGUI.m
myexpo.m
myMNA_EXP.m
matrixG.m
matrixB.m
clc
clear all
close all
format long
FRONTPAGE
MyGUIcurrentAC
Enter circuit element :
e.g. R2
OK
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1


diode_simulation.m
FRONTPAGE.m
MyGUI.m
myexpo.m
myMNA_EXP.m
matrixG.m
matrixB.m
clc
clear all
close all
format long
FRONTPAGE
MyGUIcurrentAC
Enter circuit element :
R2
OK
×10^-4
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
0.2
0.4
0.6
0.8
1
1.2

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
element=get(handles.edit1,'string');
element_name=element(1)
element_id=str2num(element(2))
t=getappdata(0,'t')
% y=5*t

matD=getappdata(0,'matD')
matR=getappdata(0,'matR')
matV=getappdata(0,'matV')
node=getappdata(0,'node');
ACvolts=getappdata(0,'ACvolts');
 ACsourceCurr=getappdata(0,'ACsourceCurr');
diodecurrent=getappdata(0,'diodecurrent');

node1=zeros(length(t),1)
node2=zeros(length(t),1)

if element_name=='R'
        for i=1:size(matR,1)
            if matR(i,1)==element_id
                node1=matR(i,2)
                node2=matR(i,3)
                res=matR(i,4)
            end
         end
        if node1==0
            nodev1=0
            nodev2=ACvolts(:,node2)
        end
```

```
if element_name=='R'
        for i=1:size(matR,1)
            if matR(i,1)==element_id
                node1=matR(i,2)
                node2=matR(i,3)
                res=matR(i,4)
            end
         end
        if node1==0
            nodev1=0
            nodev2=ACvolts(:,node2)
        end
        if node2==0
            nodev2=0
            nodev1=ACvolts(:,node1)
        end
        if node1~=0 && node2~=0
            nodev1=ACvolts(:,node1)
            nodev2=ACvolts(:,node2)
        end
       current=(nodev1-nodev2)/res

plot(t,current);

elseif element_name=='V'
    ACsourceCurr=(-1)*ACsourceCurr
     plot(t,ACsourceCurr)

elseif element_name=='D'
    plot(t,diodecurrent(:,element_id))
end
```

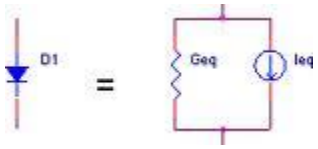
In this way, plot of current across any element can be shown as output.

Functionalities of myMNA_EXP.m :

```
function ans_from_func=myMNA_EXP(matD,matR,matV,node);

%% diode parameters

V_t=26e-3;

Isat=5e-17;

Idiode=0.1;

Vdiode=V_t*log(1 + Idiode/Isat);

eqG=Isat/V_t*exp(Vdiode/V_t);

eqI=Idiode-eqG*Vdiode;

Vd_temp = 1;

no_V=size(matV,1);

%% Initialising diode values

ndiode = size(matD, 1);

Idiode(1 : ndiode)=0.1;

Vdiode(1 : ndiode)=V_t*log(1 + Idiode/Isat);

eqG(1 : ndiode)=Isat/V_t*exp(Vdiode/V_t);

eqI(1 : ndiode)=Idiode- eqG.*Vdiode;

Vd_temp(1 : ndiode) = 1;

itr=1;
```

```
%% iterations

while(max(abs(Vd_temp-Vdiode))>1e-6)

        G1 = matrixG(matR, node);

        matR2=[matD 1./eqG'];

        G2 = matrixG(matR2, node);

        G = G1+G2;

        B = matrixB(node,matV);

        Bp = B';

        D = zeros(no_V);

        MNA = [G B;Bp D];

        I = [matD eqI'];% diodes are treated as current sources

        i = matrixI(I,node);

        z = [i;matV(:,4)];

        ans_from_func = inv(MNA)*z;

        v = ans_from_func(1 : node);

        curr = ans_from_func(node+1 : end);

        Vd_temp(1 : end) = Vdiode; % stores Vd from previous iteration


         % calculating the diode node voltage

         for i = 1 : size(matD, 1)

             positive = matD(i, 2);

             negative = matD(i, 3);

             if negative == 0

                 Vdiode(i) = ans_from_func(positive);

             elseif positive == 0

                 Vdiode(i) = -ans_from_func(negative);

             else

                 Vdiode(i) = ans_from_func(positive) - ans_from_func(negative);

             end
         end

         Idiode = Isat*(exp(Vdiode/V_t) - 1);

         eqG = Isat/V_t*exp(Vdiode/V_t);

         eqI = Idiode - eqG.*Vdiode;

         itr = itr + 1;
end
```

In this function:

- 

Where *Ieq* and G_{eq} are the current source and conductance of the diode companion model respectively.
- Started with an initial guess for the current through the diode.
- Found its conductance at this operating point from the Shockley equation (or known as the exponential model). Used those values for *Ieq* and *Geq*, and solved the circuit.
- Found that there is a significant difference between Vd_temp and Vdiode
- So updated *Ieq* to that total current, and find a new *Geq* for this new estimate of the operating point. Solved the circuit again.
- Again, got a slightly different total voltage across the diode. Made a new equivalent circuit. Solved again.
- Kept doing this the solution is close enough to the correct answer. This could be when the current through the conductance arm gets very small. Or it could be when the difference in the solutions between one iteration and the next gets very small. The latter is implemented in the code.

Functionalities of myexpo.m :

```
function [stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=myexpo(matD,matR,matV,node);
ans_from_func=myMNA_EXP(matD,matR,matV,node)
matD
matD=sortrows(matD)

ansnew=[0];


Isat=5e-17;

V_t=26e-3;

stringV='';
stringI='';
stringErrV='';
stringErrI='';

temp_I='';
diodecurr=zeros(size(matD,1),1);

for m=1:size(matD,1)

    if matD(m,3)==0

        V_d=ans_from_func(matD(m,2));

    elseif matD(m,2)==0
        V_d=-ans_from_func(matD(m,3));

    else
        V_d=ans_from_func(matD(m,2))-ans_from_func(matD(m,3));
    end
    if V_d>0

        temp_V=sprintf('diode %d is forward biased and voltage is %d V\n',...
            matD(m,1),V_d);
```

```
if V_d>0

    temp_V=sprintf('diode %d is forward biased and voltage is %d V\n',...
        matD(m,1),V_d);

    stringV=char({stringV;temp_V});

    diodecurr(m,1)=Isat*(exp(V_d/V_t) - 1)

    temp_I=sprintf('diode %d current is %d A\n',...
        matD(m,1),Isat*(exp(V_d/V_t) - 1));

    stringI=char({stringI;temp_I});

    temp_ErrV=sprintf('Error in diode %d voltage is %d percent\n',...
        matD(m,1),0);

    stringErrV=char({stringErrV;temp_ErrV});

    temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
        matD(m,1),0);


    stringErrI=char({stringErrI;temp_ErrI});


else temp_V=sprintf('diode %d is reverse biased and voltage is %d V\n',...
        matD(m,1),V_d);

    stringV=char({stringV;temp_V});

    diodecurr(m,1)=-Isat

    temp_I=sprintf('diode %d is reverse biased and current is %d A\n',...
        matD(m,1),-Isat);

    stringI=char({stringI;temp_I});
```

```
        stringErrV=char({stringErrV;temp_ErrV});

        temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
            matD(m,1),0);


        stringErrI=char({stringErrI;temp_ErrI});


    else temp_V=sprintf('diode %d is reverse biased and voltage is %d V\n',...
            matD(m,1),V_d);

        stringV=char({stringV;temp_V});

        diodecurr(m,1)=-Isat

        temp_I=sprintf('diode %d is reverse biased and current is %d A\n',...
            matD(m,1),-Isat);

        stringI=char({stringI;temp_I});

        temp_ErrV=sprintf('Error in diode %d voltage is %d percent\n',...
            matD(m,1),0);

        stringErrV=char({stringErrV;temp_ErrV});

        temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
            matD(m,1),0);

        stringErrI=char({stringErrI;temp_ErrI});

    end
end

ans_from_func
```

- myMNA_EXP.m is called in the beginning
- Finally, voltage difference of diodes are determined and the biasing is also determined according to that.
- Diodecurr is a matrix containing current values through voltage sources and diodes

Functionalities of myMNA_PWL.m :

```
function ans_from_func=myMNA_PWL(matD,matR,matV)

node = max([max(matD(:,2:3)) max(matR(:,2:3)) max(matV(:,2:3))]);

no_V=size(matV,1);

matR=[matR;matD 20*ones(size(matD,1),1)];

G = matrixG(matR,node);

B = matrixB(node,matV);

Bp=B';

D=zeros(no_V);
MNA=[G B;Bp D];

I=[matD -0.65/20*ones(size(matD,1),1)];

z=[matrixI(I,node);matV(:,4)];

ans_from_func=inv(MNA)*z
```

Here each diode is considered as a combination of 20 ohm resistor and (-0.65/20) A current source. Hence, the matrix ans_from_func will contain all the node voltages and the current through input voltage source.

Functionalities of mypwl.m :

Code:

function

[stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=mypwl(matD,matR,matV,

node);

matVnew=matV;

```
ans_from_func=myMNA_PWL(matD,matR,matV)
ansnew=[0]

ansexp=myMNA_EXP(matD,matR,matV,node);

matEv=matD;

stringV='';
stringI='';
stringErrV='';
stringErrI='';

temp_I='';

diodecurr=zeros(size(matD,1),1);

for m=1:size(matD)

   if matD(m,3)==0

      dVolt=ans_from_func(matD(m,2));

   elseif matD(m,2)==0

      dVolt=-ans_from_func(matD(m,3));

   else
      dVolt=ans_from_func(matD(m,2))-ans_from_func(matD(m,3));
   end
```

```
    if (-0.65/20+dVolt/20)>0
        matEv(m,:)=0;

    else matD(m,:)=0;
    end

end
matD(all(~matD,2),:)=[];

matEv(all(~matEv,2),:)=[];

if isempty(matEv)
    for m=1:size(matD,1)

        if matD(m,3)==0

          dVolt=ans_from_func(matD(m,2));

          dVolt1=ansexp(matD(m,2));
        elseif matD(m,2)==0

           dVolt=-ans_from_func(matD(m,3));

           dVolt1=-ansexp(matD(m,3));
        else
           dVolt=ans_from_func(matD(m,2))-ans_from_func(matD(m,3));

           dVolt1=ansexp(matD(m,2))-ansexp(matD(m,3));
        end
```

```
        temp_V=sprintf('diode %d is forward biased and voltage is %d V\n',...
            matD(m,1),dVolt);

        stringV=char({stringV;temp_V});

         diodecurr(matD(m,1),1)=-0.65/20+dVolt/20

        temp_I=sprintf('diode %d is forward biased and current is %d A\n',...
            matD(m,1),-0.65/20+dVolt/20);

        stringI=char({stringI;temp_I});

        Idiode=5e-17*(exp(dVolt1/26e-3)-1);

        d_I=-0.65/20+dVolt/20;

        temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
            matD(m,1),abs(Idiode-d_I)*100/abs(Idiode));

        stringErrI=char({stringErrI;temp_ErrI});

        tempErrV=sprintf('Error in diode %d voltage is %d percent\n',...
            matD(m,1),abs(dVolt-dVolt1)*100/abs(dVolt1));

        stringErrV=char({stringErrV;tempErrV});
    end
else matEv=[matEv 1e15*ones(size(matEv,1),1)];
```

```
matR=[matR; matEv];

ansnew=myMNA_PWL(matD,matR,matVnew)

for m=1:size(matEv,1)

  if matEv(m,3)==0

      dVolt=ansnew(matEv(m,2));

      dVolt1=ansexp(matEv(m,2));

   elseif matEv(m,2)==0
      dVolt=-ansnew(matEv(m,3));

      dVolt1=-ansexp(matEv(m,3));
   else
      dVolt=ansnew(matEv(m,2))-ansnew(matEv(m,3));

      dVolt1=ansexp(matEv(m,2))-ansexp(matEv(m,3));
  end

   temp_V=sprintf('diode %d is reverse biased and voltage is %d V\n',...
      matEv(m,1),dVolt);

   stringV=char({stringV;temp_V});

   diodecurr(matEv(m,1),1)=dVolt/1e16
```

```
    temp_I=sprintf('diode %d is reverse biased and current is %d A\n',...
        matEv(m,1),dVolt/1e16);

    stringI=char({stringI;temp_I});

    tempErrV=sprintf('Error in diode %d voltage is %d percent\n',...
        matEv(m,1),abs(dVolt-dVolt1)*100/abs(dVolt1));

    stringErrV=char({stringErrV;tempErrV});


    d_I=dVolt/1e16;

    Idiode=5e-17*(exp(dVolt1/26e-3)-1);

    temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
        matEv(m,1),abs(Idiode-d_I)*100/abs(Idiode));

    stringErrI=char({stringErrI;temp_ErrI});


end
for m=1:size(matD,1)

    if matD(m,3)==0

        dVolt=ansnew(matD(m,2));

        dVolt1=ansexp(matD(m,2));
```

```
elseif matD(m,2)==0

   dVolt=-ansnew(matD(m,3));

   dVolt1=-ansexp(matD(m,3));
else
   dVolt=ansnew(matD(m,2))-ansnew(matD(m,3));

   dVolt1=ansexp(matD(m,2))-ansexp(matD(m,3));
end

temp_V=sprintf('diode %d is forward biased and voltage is %d V\n',...
   matD(m,1),dVolt);

stringV=char({stringV;temp_V});

 diodecurr(matD(m,1),1)=-0.65/20+dVolt/20

temp_I=sprintf('diode %d is forward biased and current is %d A\n',...
   matD(m,1),-0.65/20+dVolt/20);

stringI=char({stringI;temp_I});

tempErrV=sprintf('Error in diode %d voltage is %d percent\n',...
   matD(m,1),abs(dVolt-dVolt1)*100/abs(dVolt1));

stringErrV=char({stringErrV;tempErrV});
```

```
        d_I=(dVolt-0.65)/20;

        Idiode=5e-17*(exp(dVolt1/26e-3)-1);

        temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
            matD(m,1),abs(Idiode-d_I)*100/abs(Idiode));

        stringErrI=char({stringErrI;temp_ErrI});


    end
end
```

In this function we get the values : stringV,stringI,stringErrV,stringErrI,ans_from_func and diodecurr.

These values will be used in piecewise linear callback in MyGUI function.

Functionalities of myMNA_CVD.m :

```
function ans_from_func=myMNA_CVD(matD,matR,matV)

node = max([max(matD(:,2:3)) max(matR(:,2:3)) max(matV(:,2:3))]);

%considering each diode as a voltage source

ndiode = 0.7*ones(size(matD,1),1)

matDSOURCE=matD(:,1)+size(matV,1)


matV  = [matV;matDSOURCE matD(:,2:3) ndiode]

matV=sortrows(matV);

no_V=size(matV,1);

G = matrixG(matR,node);

B = matrixB(node,matV);

Bp=B';

D=zeros(no_V);

MNA=[G B;Bp D];

z=[zeros(node,1);matV(:,4)];

ans_from_func=inv(MNA)*z;
```

Here each diode is considered as a 0.7V voltage source. Hence, the matrix ans_from_func will contain all the node voltages and the currents through input voltage source as well as the diodes.

Functionalities of mycvd.m :

Code:

```
function
[stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=mycvd(matD,matR,matV,
node)
matVnew=matV;

ans_from_func=myMNA_CVD(matD,matR,matV)
ansnew=[0]

ansexp=myMNA_EXP(matD,matR,matV,node);

matEv=matD;

matEv=sortrows(matEv);

matD=sortrows(matD);

stringV='';

stringI='';
stringErrV='';
stringErrI='';
temp_I='';

diodecurr=zeros(size(matD,1),1);

for m=1:size(matD)
```

```
    if ans_from_func(node+size(matV,1)+m)>0
        matEv(m,:)=0
    else matD(m,:)=0
    end
end
matD(all(~matD,2),:)=[]
matEv(all(~matEv,2),:)=[]
if isempty(matEv)
    for m=1:size(matD,1)
        if matD(m,3)==0
            d_V=ans_from_func(matD(m,2));
            d_V1=ansexp(matD(m,2));
        elseif matD(m,2)==0
            d_V=-ans_from_func(matD(m,3));
            d_V1=-ansexp(matD(m,3));
        else
```

```
    d_V=ans_from_func(matD(m,2))-ans_from_func(matD(m,3));

    d_V1=ansexp(matD(m,2))-ansexp(matD(m,3));

end
temp_V=sprintf('diode %d is forward and voltage is %d V\n',...
    matD(m,1),d_V);

stringV=char({stringV;temp_V});

d_I=ans_from_func(node+size(matV,1)+m);

diodecurr(matD(m,1),1)=d_I

temp_I=sprintf('diode %d is forward and current is %d A\n',...
    matD(m,1),d_I);

stringI=char({stringI;temp_I});

Idiode=5e-17*(exp(d_V1/26e-3)-1)

temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
    matD(m,1),abs(Idiode-d_I)*100/abs(Idiode));

stringErrI=char({stringErrI;temp_ErrI});

temp_ErrV=sprintf('Error in diode %d voltage is %d percent\n',...
    matD(m,1),abs(d_V-d_V1)*100/abs(d_V1));
```

```
        stringErrV=char({stringErrV;temp_ErrV});
    end
else matEv=[matEv 1e15*ones(size(matEv,1),1)]

    matR=[matR; matEv]

    ansnew=myMNA_CVD(matD,matR,matVnew)

    for m=1:size(matEv,1)

        if matEv(m,3)==0
            d_V=ansnew(matEv(m,2));



            d_V1=ansexp(matEv(m,2));

        elseif matEv(m,2)==0

            d_V=-ansnew(matEv(m,3));

            d_V1=-ansexp(matEv(m,3));
        else

            d_V=ansnew(matEv(m,2))-ansnew(matEv(m,3));

            d_V1=ansexp(matEv(m,2))-ansexp(matEv(m,3));
        end
        temp_V=sprintf('diode %d is reverse biased and voltage is %d V\n',...
            matEv(m,1),d_V);
```

```
    stringV=char({stringV;temp_V});

     diodecurr(matEv(m,1),1)=d_V/1e12

    temp_I=sprintf('diode %d is reverse biased and current is %d A\n',...
       matEv(m,1),d_V/1e12);

    stringI=char({stringI;temp_I});

    temp_ErrV=sprintf('Error in diode %d voltage is %d percent\n',...
       matEv(m,1),abs(d_V-d_V1)*100/abs(d_V1));

    stringErrV=char({stringErrV;temp_ErrV});


    d_I=d_V/1e16;

    Idiode=5e-17*(exp(d_V1/26e-3)-1);

    temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
       matEv(m,1),abs(Idiode-d_I)*100/abs(Idiode));

    stringErrI=char({stringErrI;temp_ErrI});

end
for m=1:size(matD,1)

    if matD(m,3)==0
```

```
    d_V=ansnew(matD(m,2));

    d_V1=ansexp(matD(m,2));
elseif matD(m,2)==0

    d_V=-ansnew(matD(m,3));

    d_V1=-ansexp(matD(m,3));
else
    d_V=ansnew(matD(m,2))-ansnew(matD(m,3));

    d_V1=ansexp(matD(m,2))-ansexp(matD(m,3));
end
temp_V=sprintf('diode %d is forward and voltage is %d V\n',...
    matD(m,1),d_V);

stringV=char({stringV;temp_V});

diodecurr(matD(m,1),1)=ansnew(node+size(matV,1)+m)

temp_I=sprintf('diode %d is forward and current is %d A\n',...
    matD(m,1),ansnew(node+size(matV,1)+m));

stringI=char({stringI;temp_I});

temp_ErrV=sprintf('Error in diode %d voltage is %d percent\n',...
    matD(m,1),abs(d_V-d_V1)*100/abs(d_V1));
```

```
        stringErrV=char({stringErrV;temp_ErrV});

        d_I=ansnew(node+size(matV,1)+m);

        Idiode=5e-17*(exp(d_V1/26e-3)-1);

        temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
            matD(m,1),abs(Idiode-d_I)*100/abs(Idiode));

        stringErrI=char({stringErrI;temp_ErrI});


    end
end
```

In this function we get the values : stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew and diodecurr.

These values will be used in constant voltage callback in MyGUI function.

ans_from_func is used only when no diode is in reverse bias, in all other cases the ansnew matrix will be used. This 'ansnew' matrix is created in this function because if a diode in reverse bias we replace it with a 10^15 ohm resistor so that branch current can become almost 0.

Functionalities of myMNA_IDM.m :

```
function ans_from_func=myMNA_IDM(matD,matR,matV)
node = max([max(matD(:,2:3)) max(matR(:,2:3)) max(matV(:,2:3))]);
n = size(matD);
ndiode = 1e-4*ones(n(1),1);
n1=size(matV);
matV  = [matV;matD(:,1)+n1(1) matD(:,2:3) ndiode];
matV=sortrows(matV);
nV=size(matV,1);
G = matrixG(matR,node);
B = matrixB(node,matV);
Bp=B';
D=zeros(nV);
MNA=[G B;Bp D];
z=[zeros(node,1);matV(:,4)];
ans_from_func=inv(MNA)*z
```

Here each diode is considered as a 10^-4 V voltage source. Hence, the matrix ans_from_func will contain all the node voltages and the currents through input voltage source as well as the diodes.

Functionalities of myidm.m :

Code:

function

[stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew,diodecurr]=myidm(matD,matR,matV,node)

matVnew=matV;

ans_from_func=myMNA_IDM(matD,matR,matV)

ansnew=[0]

ansexp=myMNA_EXP(matD,matR,matV,node)

```
matEv=matD;

matEv=sortrows(matEv);

matD=sortrows(matD);

stringV='';
stringI='';
stringErrV='';
stringErrI='';

temp_I='';

diodecurr=zeros(size(matD,1),1);
for m=1:size(matD)

   if ans_from_func(node+size(matV,1)+m)>0

      matEv(m,:)=0;

   else matD(m,:)=0;
   end
end
matD(all(~matD,2),:)=[];

matEv(all(~matEv,2),:)=[];

if isempty(matEv)
```

```
for m=1:size(matD,1)

    temp_V=sprintf('diode %d is forward and it is short circuited\n',...
        matD(m,1));

    stringV=char({stringV;temp_V});

    diodecurr(matD(m,1),1)=ans_from_func(node+size(matV,1)+m)

    temp_I=sprintf('diode %d current is %d A\n',...
        matD(m,1),ans_from_func(node+size(matV,1)+m));

    stringI=char({stringI;temp_I});

    if matD(m,3)==0

        d_V1=ansexp(matD(m,2));
    elseif matD(m,2)==0

        d_V1=-ansexp(matD(m,3));
    else

        d_V1=ansexp(matD(m,2))-ansexp(matD(m,3));
    end

    d_I=ans_from_func(node+size(matV,1)+m);

    Idiode=5e-17*(exp(d_V1/26e-3)-1);
```

```
        temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
            matD(m,1),abs(Idiode-d_I)*100/abs(Idiode));

        stringErrI=char({stringErrI;temp_ErrI});

        temp_ErrV=sprintf('Error in diode %d voltage is %d percent\n',...
            matD(m,1),100);
        stringErrV=char({stringErrV;temp_ErrV});

    end
else matEv=[matEv 1e15*ones(size(matEv,1),1)];

    matR=[matR; matEv];

    ansnew=myMNA_IDM(matD,matR,matVnew)

    size(matEv,1)
    for m=1:size(matEv,1)
        loopinside=m
        if matD(m,3)==0

            d_V=ansnew(matD(m,2))

            d_V1=ansexp(matD(m,2));
        elseif matD(m,2)==0
            d_V=-ansnew(matD(m,3));

            d_V1=-ansexp(matD(m,3));
        else
```

```
    d_V=ansnew(matD(m,2))-ansnew(matD(m,3));

    d_V1=ansexp(matD(m,2))-ansexp(matD(m,3));
end


temp_V=sprintf('diode %d is reverse and voltage is %d V\n',...
    matEv(m,1),d_V);

stringV=char({stringV;temp_V});

diodecurr(matEv(m,1),1)=0

temp_I=sprintf('diode %d is reverse and it is open circuited \n',...
    matEv(m,1));

stringI=char({stringI;temp_I});


Idiode=5e-17*(exp(d_V1/26e-3)-1)

temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
    matEv(m,1),100);

stringErrI=char({stringErrI;temp_ErrI});

temp_ErrV=sprintf('Error in diode %d voltage is %d percent\n',...
    matEv(m,1),abs(d_V-d_V1)*100/abs(d_V1));
```

```
    stringErrV=char({stringErrV;temp_ErrV});


end
for m=1:size(matD,1)

    temp_V=sprintf('diode %d is forward and it is short circuited\n',...
        matD(m,1));

    stringV=char({stringV;temp_V});

     diodecurr(matD(m,1),1)=ansnew(node+size(matV,1)+m)

    temp_I=sprintf('diode %d current is %d A\n',...
        matD(m,1),ansnew(node+size(matV,1)+m));

    stringI=char({stringI;temp_I});

    if matD(m,3)==0

        d_V1=ansexp(matD(m,2));
    elseif matD(m,2)==0

        d_V1=-ansexp(matD(m,3));
    else

        d_V1=ansexp(matD(m,2))-ansexp(matD(m,3));
    end
```

```
        d_I=ansnew(node+size(matV,1)+m);

        Idiode=5e-17*(exp(d_V1/26e-3)-1);

        temp_ErrI=sprintf('Error in diode %d current is %d percent\n',...
            matD(m,1),abs(Idiode-d_I)*100/abs(Idiode));

        stringErrI=char({stringErrI;temp_ErrI});

        temp_ErrV=sprintf('Error in diode %d voltage is %d percent\n',...
            matD(m,1),100);

        stringErrV=char({stringErrV;temp_ErrV});



    end
end
```

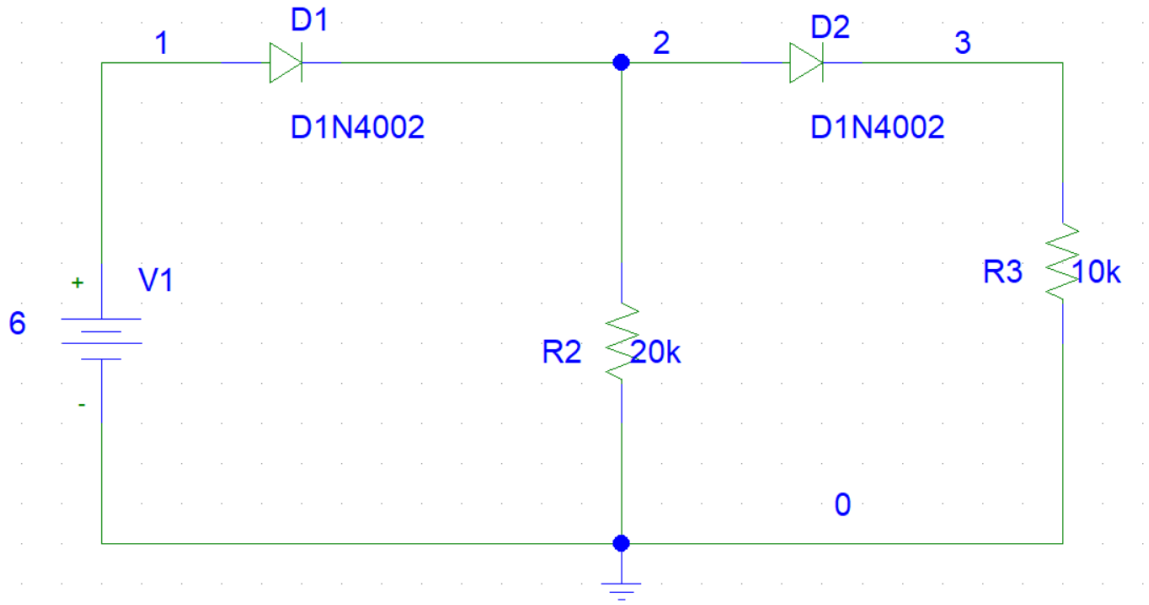
In this function we get the values : stringV,stringI,stringErrV,stringErrI,ans_from_func,ansnew and diodecurr.

These values will be used in ideal_callback in MyGUI function.

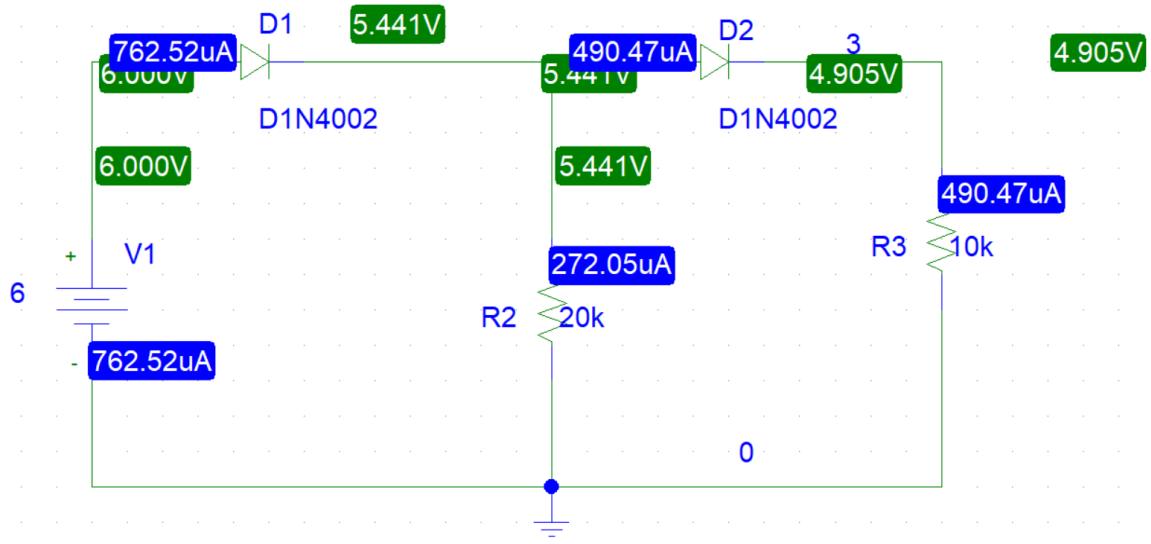
ans_from_func is used only when no diode is in reverse bias, in all other cases the ansnew matrix will be used. This 'ansnew' matrix is created in this function because if a diode in reverse bias we replace it with a 10^15 ohm resistor so that branch current can become almost 0.

Circuit Examples:

File name: 2.txt



Simulation output in PSPICE:



```
 NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE

(    1)    6.0000  (    2)    5.4410  (    3)    4.9047


    VOLTAGE SOURCE CURRENTS
    NAME         CURRENT

    V_V1        -7.625E-04
```

```
**** DIODES


NAME         D_D1        D_D2
MODEL        D1N4002     D1N4002
ID           7.63E-04    4.90E-04
VD           5.59E-01    5.36E-01
REQ          6.73E+01    1.05E+02
CAP          7.09E-08    4.56E-08
```

Simulation result obtained from MATLAB:

Exponential model:

ansnewgui =

6.000000000000000

5.212814856200693

4.437643271033519

-0.000704405069913

Piece_wise linear model:

ansnewgui =

6.000000000000000

5.335312765546735

4.675960843859017

-0.000734361722663

Constant Voltage drop Model:

ansnewgui =

6.000000000000000

5.300000000000000

4.600000000000000

-0.000725000000000

0.000725000000000

0.000460000000000

Ideal diode model:

ansnewgui =

6.000000000000000

5.999900000000000

5.999800000000000

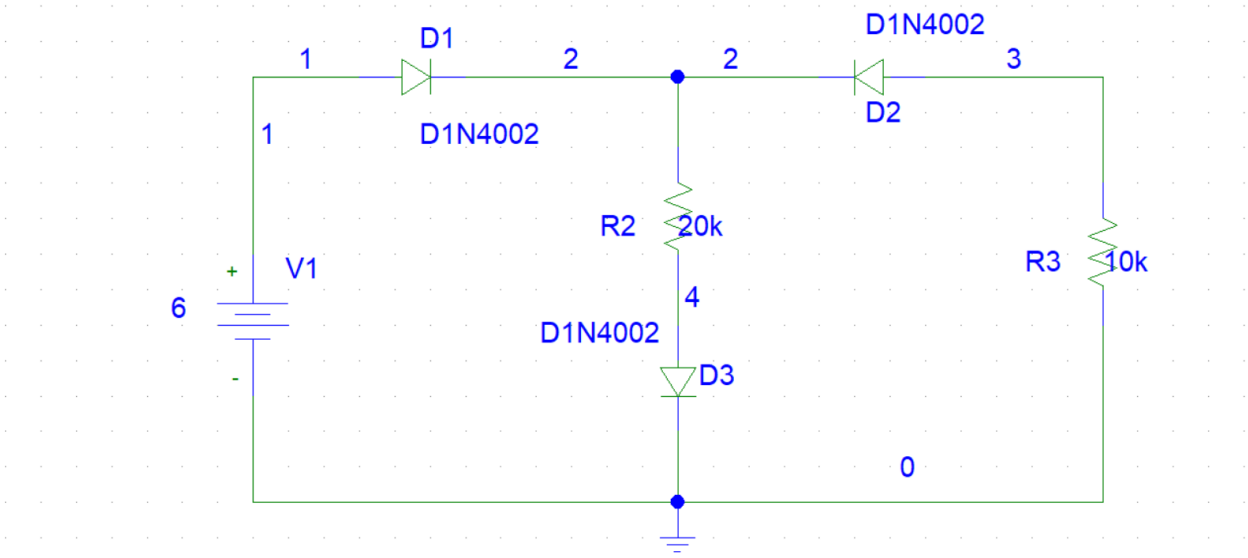
-0.000899975000000

0.000899975000000

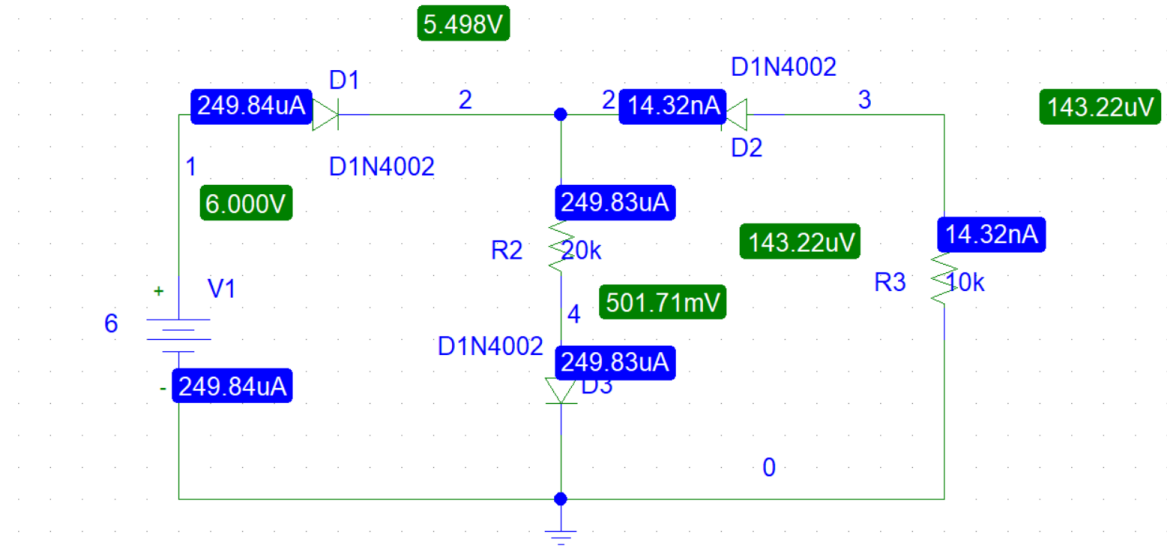
0.000599980000000

Here in the ansnewgui matrix the first 3 elements are respectively the node voltage of node 1,2 and 3. Then the consecutive element is the current through input DC source and in models where Diodes are considered as voltage source , the following elements represent current through diodes in respective order.

File name: 4.txt



Simulation output in PSPICE:



NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE
(1)	6.0000	(2)	5.4983	(3)	143.2E-06	(4)	.5017

VOLTAGE SOURCE CURRENTS

NAME	CURRENT
V_V1	-2.498E-04

**** DIODES

NAME	D_D2	D_D1	D_D3
MODEL	D1N4002	D1N4002	D1N4002
ID	-1.43E-08	2.50E-04	2.50E-04
VD	-5.50E+00	5.02E-01	5.02E-01
REQ	2.58E+11	2.05E+02	2.05E+02
CAP	2.42E-11	2.33E-08	2.33E-08

Simulation result obtained from MATLAB:

Exponential model:

ansnewgui =

6.000000000000000

5.242573399107166

0.000000000000500

0.757426600887672

-0.000224257339911

Piece_wise linear model:

ansnewgui =

6.000000000000000

5.345309381237419

0.000000000053453

0.654690618762475

-0.000234530938129

Constant Voltage drop Model:

ansnewgui =

6.000000000000000

5.300000000000000

0.000000000053000

0.700000000000000

-0.000230000000005

0.000230000000005

0.000230000000000

Ideal diode model:

ansnewgui =

6.000000000000000

5.999900000000000

0.000000000059999

0.000100000000000

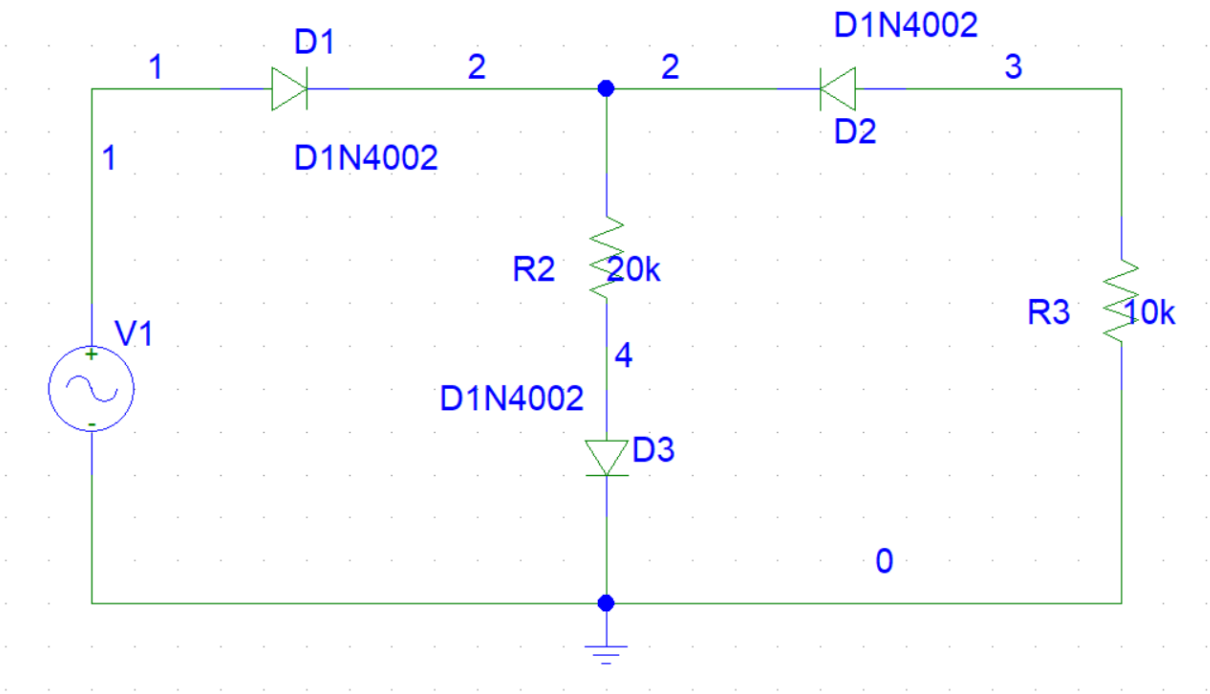
-0.000299990000006

0.000299990000006

0.000299990000000

Here in the ansnewgui matrix the first 4 elements are respectively the node voltage of node 1,2 and 3. Then the consecutive element is the current through input DC source and in models where Diodes are considered as voltage source , the following elements represent current through diodes in respective order.

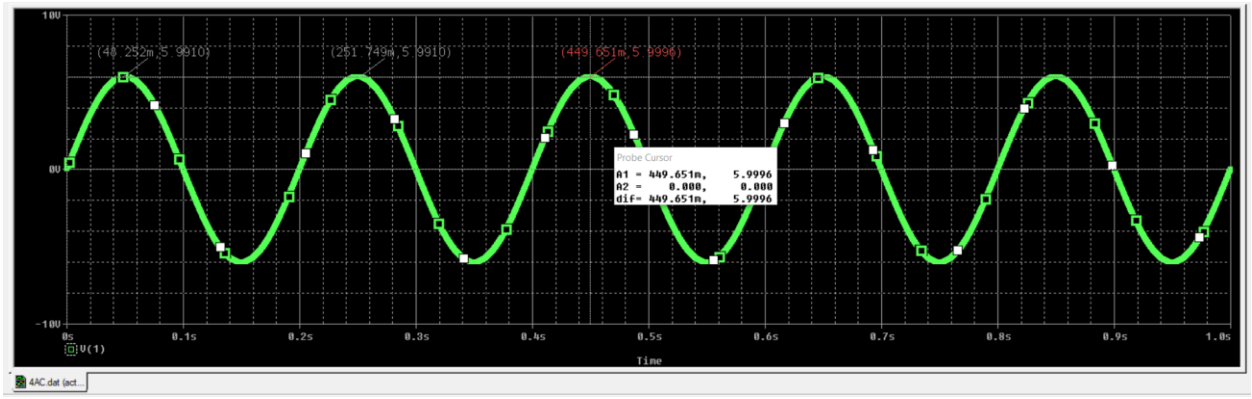
File name: 4AC.txt



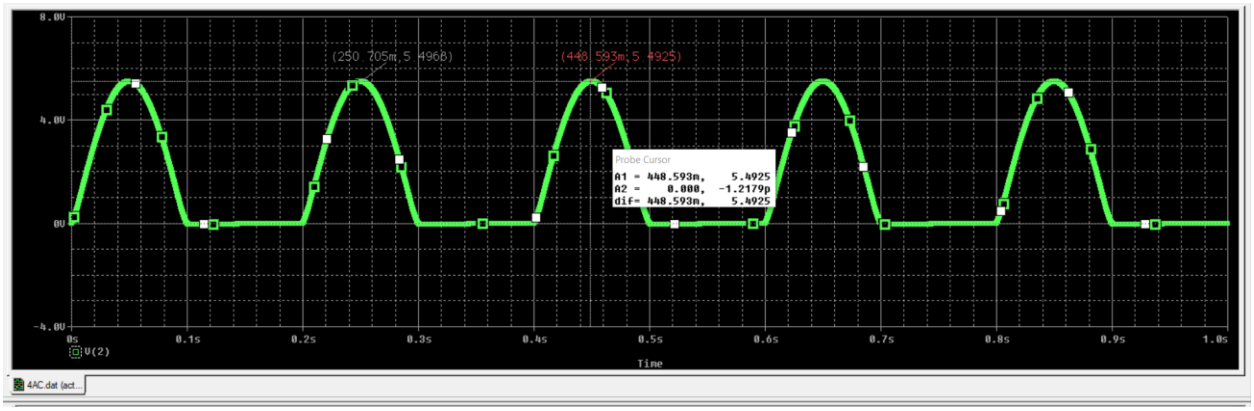
Simulation output in PSPICE:

PLOTS:

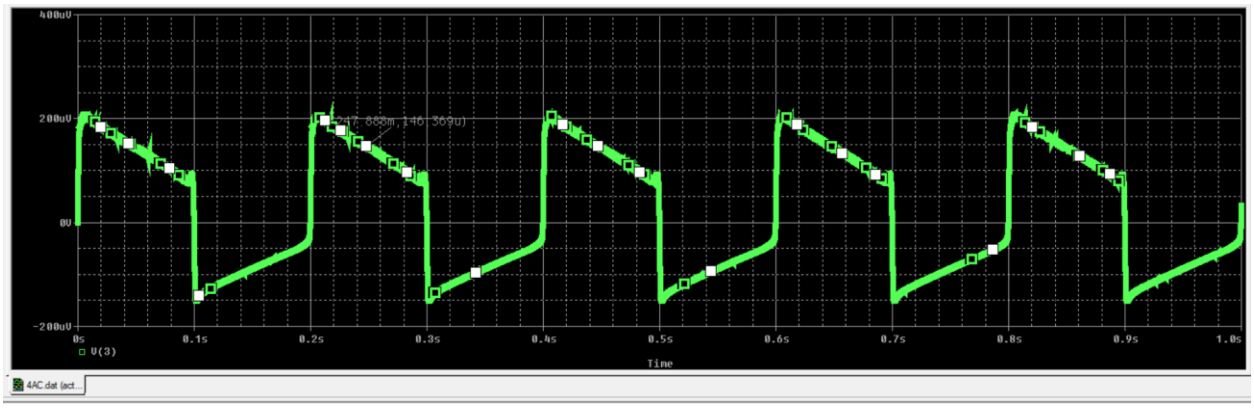
V(1)


10V
0V
-10V
(48.252m,5.9910)
(251.749m,5.9910)
(449.651m,5.9996)
Probe Cursor
A1 = 449.651m, 5.9996
A2 = 0.000, 0.000
dif= 449.651m, 5.9996
0s
0.1s
0.2s
0.3s
0.4s
0.5s
0.6s
0.7s
0.8s
0.9s
1.0s
V(1)
Time
4AC.dat (act...

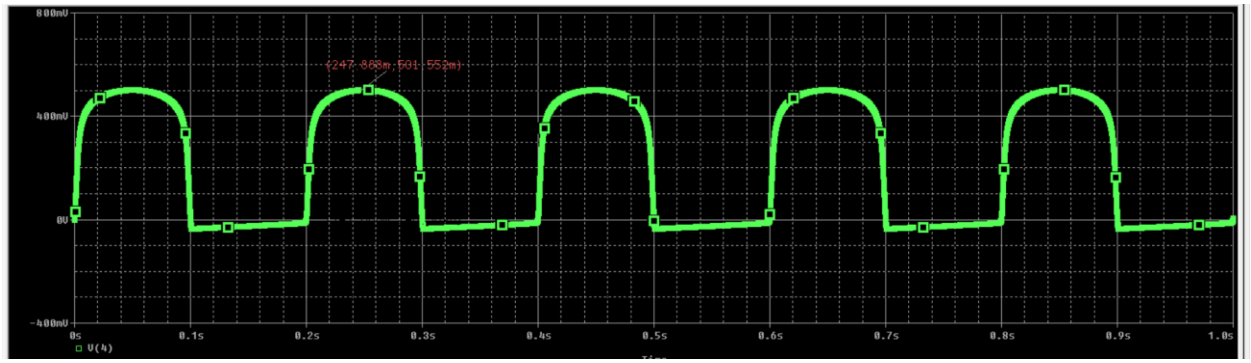
V(2)


8.0V
4.0V
0V
-4.0V
(250.705m,5.4968)
(448.593m,5.4925)
Probe Cursor
A1 = 448.593m, 5.4925
A2 = 0.000, -1.2179p
dif= 448.593m, 5.4925
0s
0.1s
0.2s
0.3s
0.4s
0.5s
0.6s
0.7s
0.8s
0.9s
1.0s
V(2)
Time
4AC.dat (act...

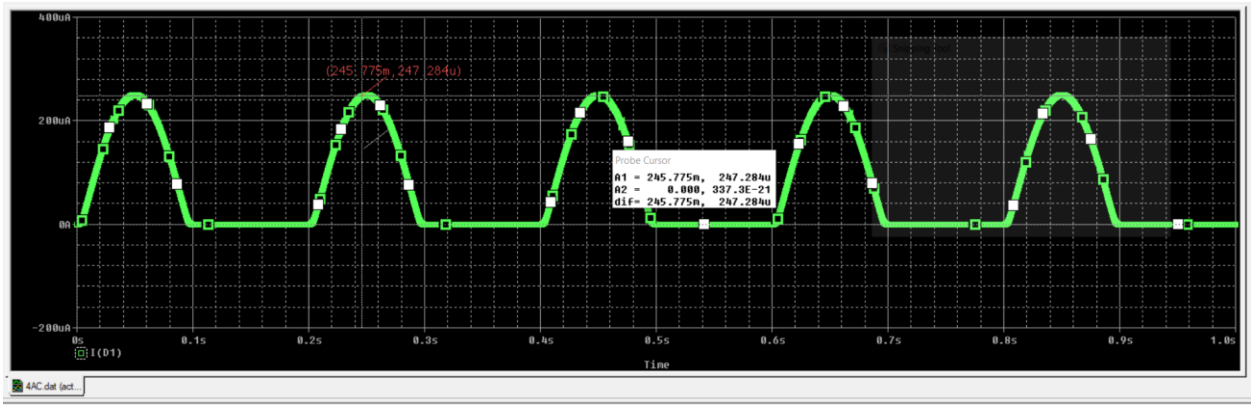
V(3)


400uV
200uV
0V
-200uV
(47.868m,146.369u)
0s
0.1s
0.2s
0.3s
0.4s
0.5s
0.6s
0.7s
0.8s
0.9s
1.0s
V(3)
Time
4AC.dat (act...

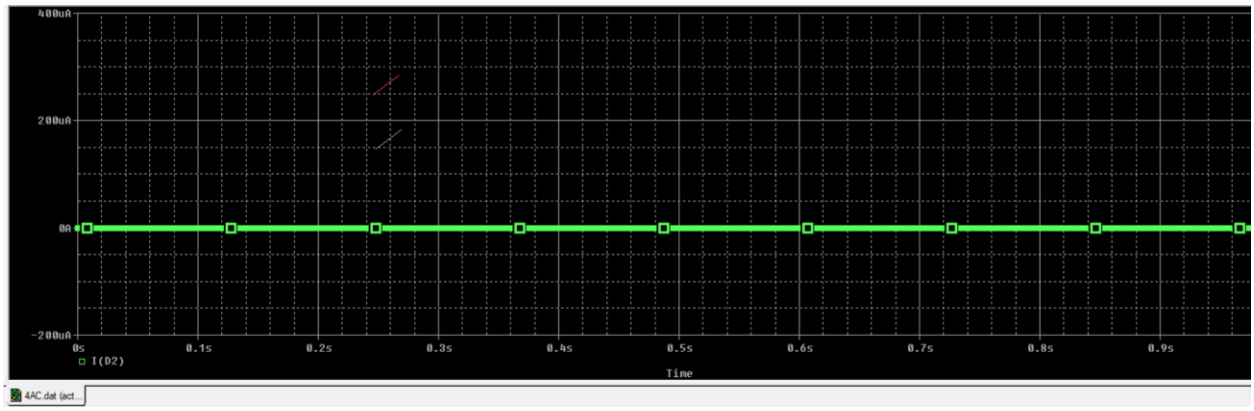
V(4)



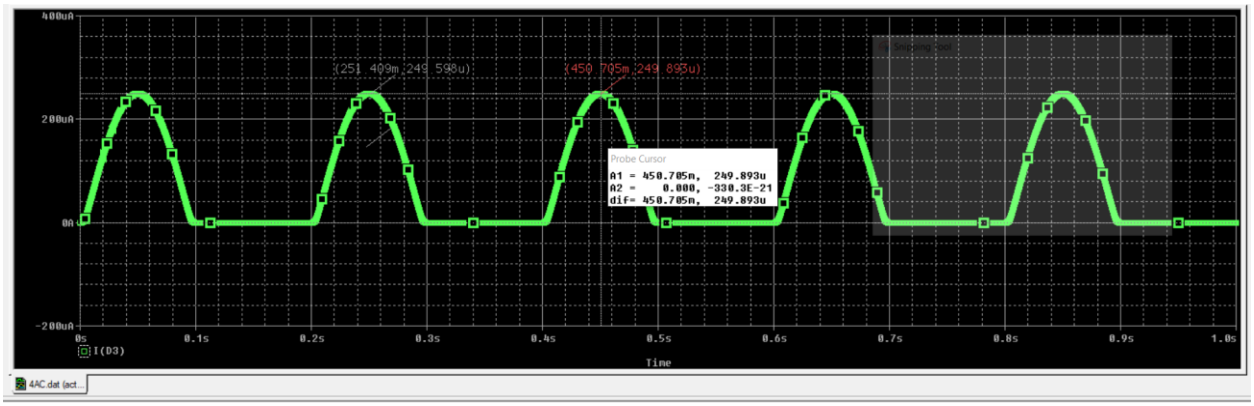
I(D1)



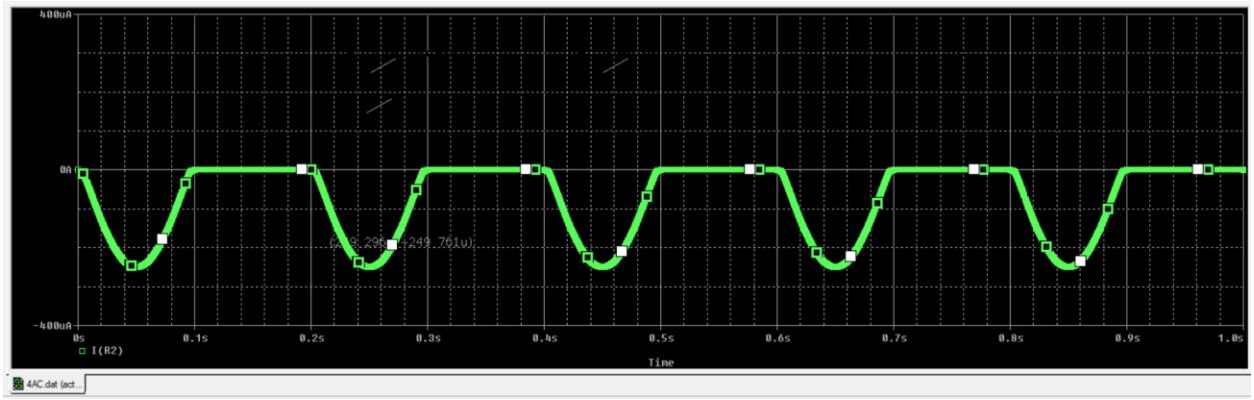
I(D2)



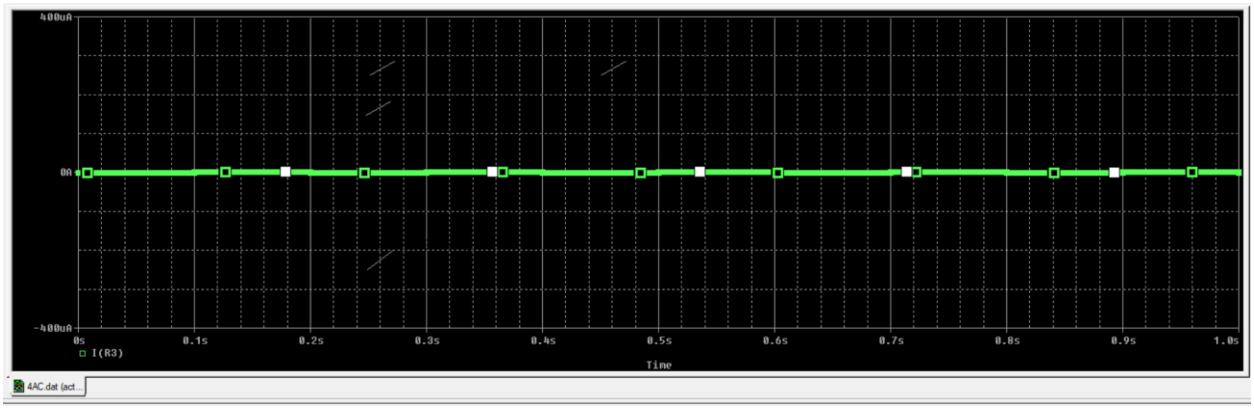
I(D3)


400uA
200uA
0A
-200uA
(251.409m,249.598u)
(450.705m,249.893u)
Probe Cursor
A1 = 450.705m, 249.893u
A2 = 0.000, -330.3E-21
dif= 450.705m, 249.893u
0s
0.1s
0.2s
0.3s
0.4s
0.5s
0.6s
0.7s
0.8s
0.9s
1.0s
I(D3)
Time
4AC.dat (act...

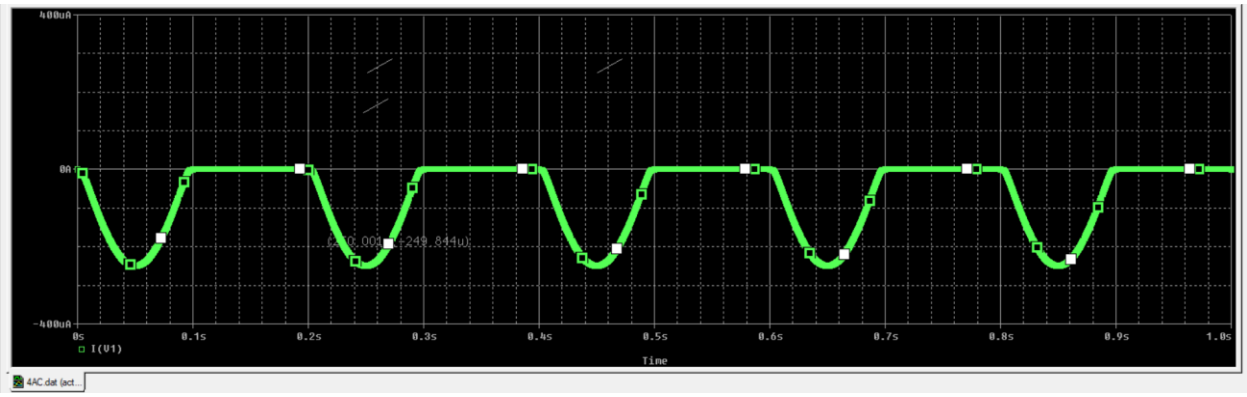
I(R2)


400uA
0A
-400uA
0s
0.1s
0.2s
0.3s
0.4s
0.5s
0.6s
0.7s
0.8s
0.9s
1.0s
I(R2)
Time
4AC.dat (act...

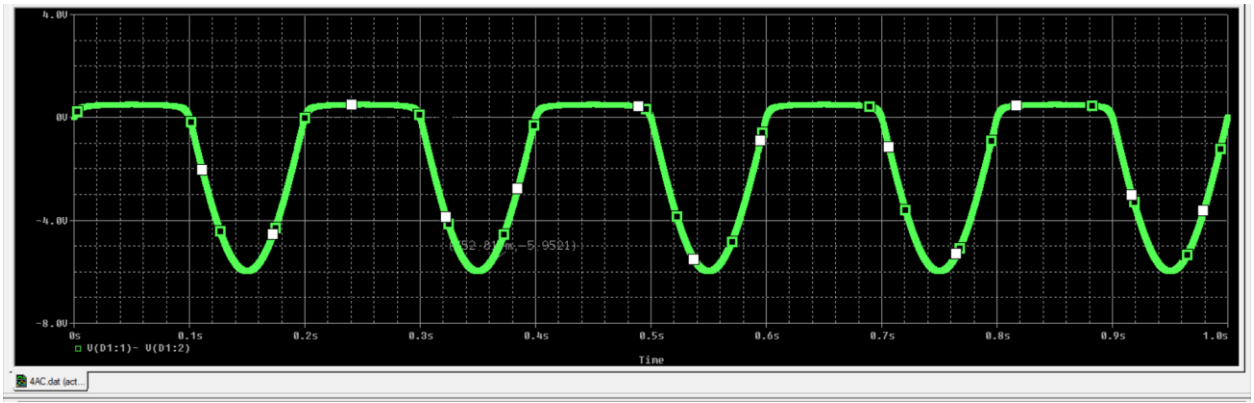
I(R3)


400uA
0A
-400uA
0s
0.1s
0.2s
0.3s
0.4s
0.5s
0.6s
0.7s
0.8s
0.9s
1.0s
I(R3)
Time
4AC.dat (act...

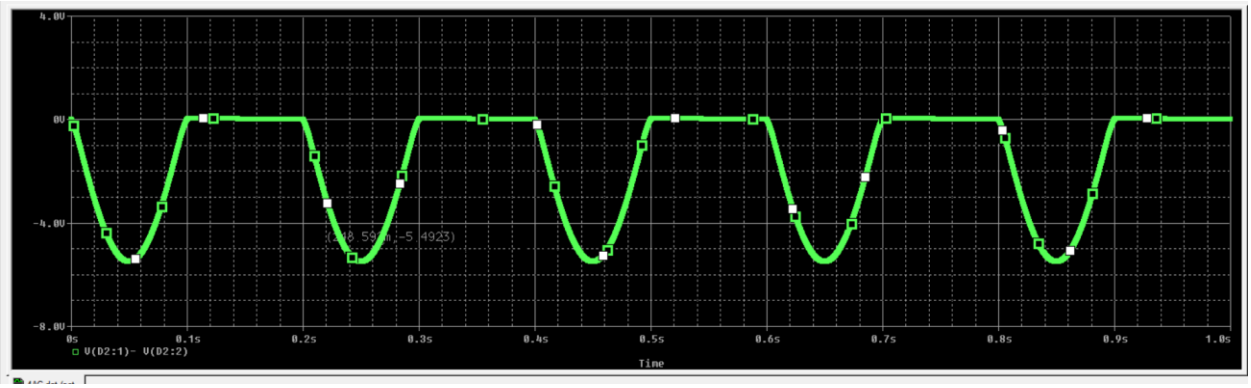
I(V1)



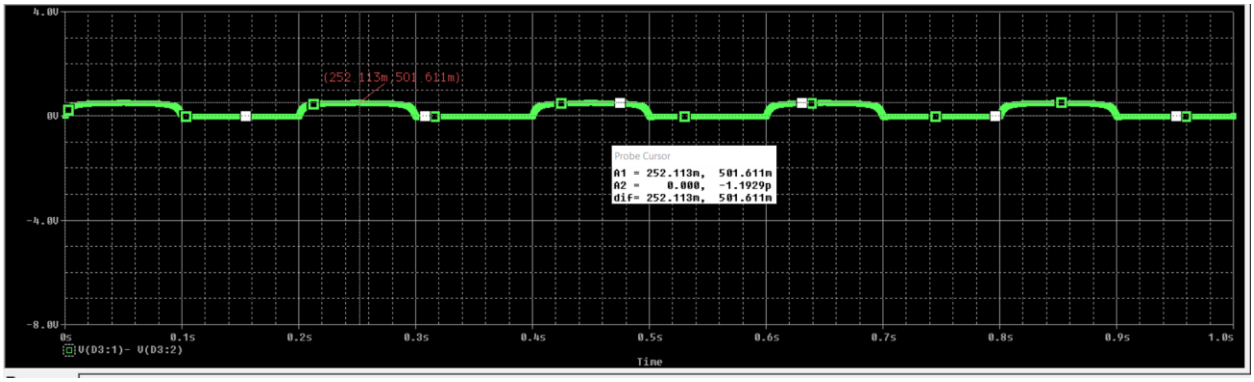
V(D1:1)- V(D1:2)



V(D2:1)- V(D2:2)

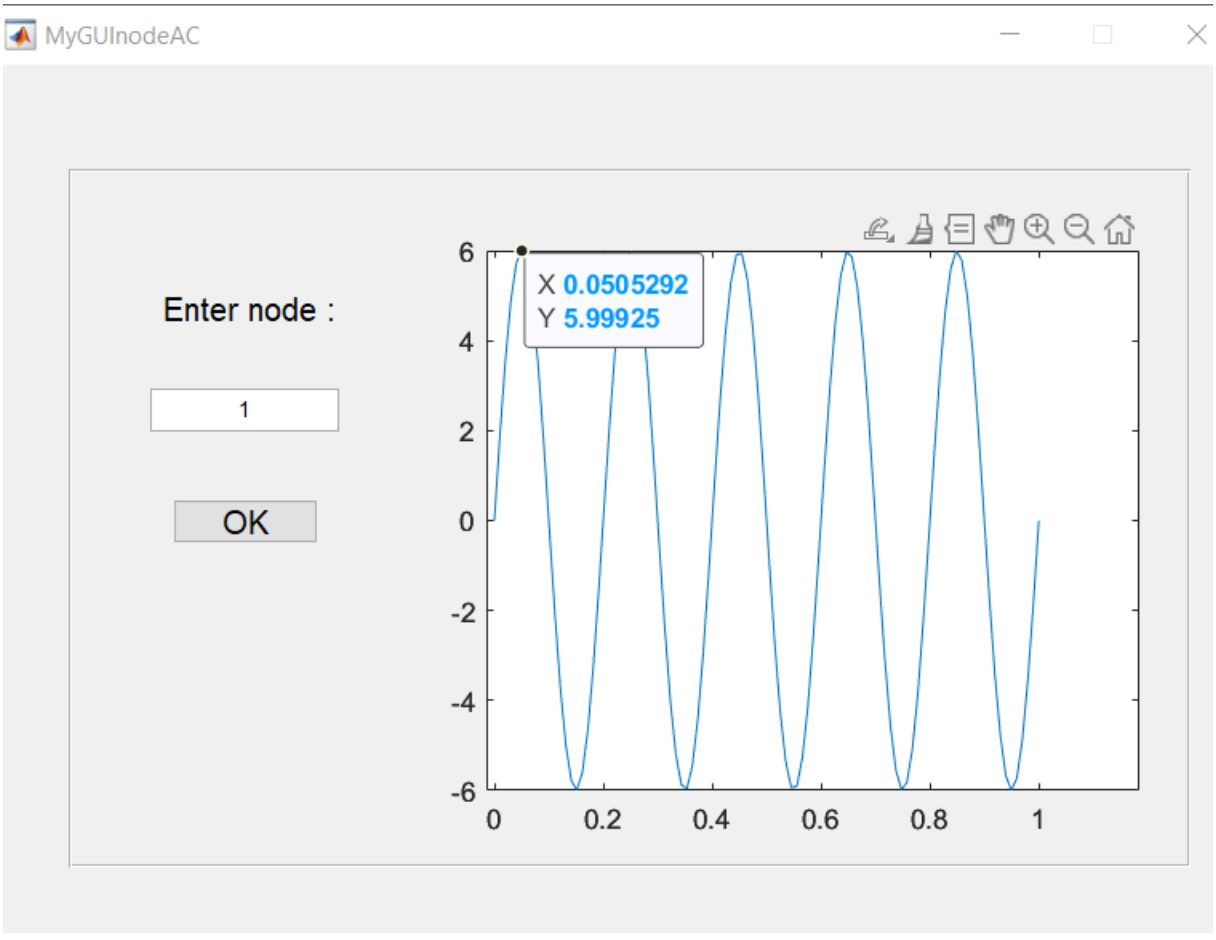


V(D3:1)- V(D3:2)

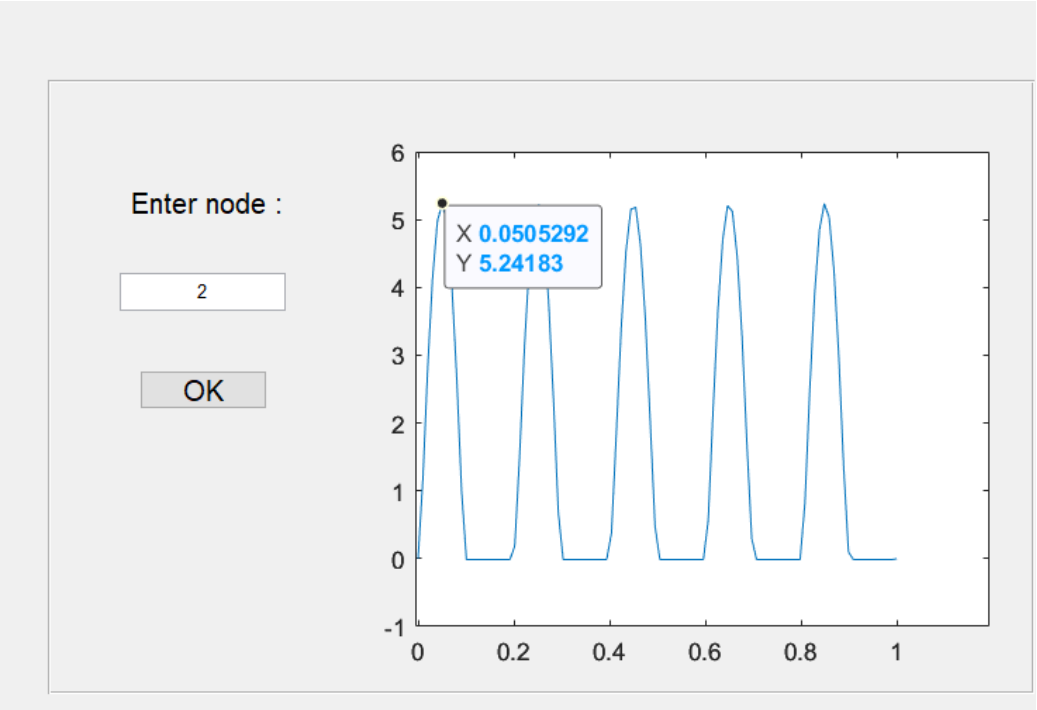


Simulation PLOT obtained in MATLAB:

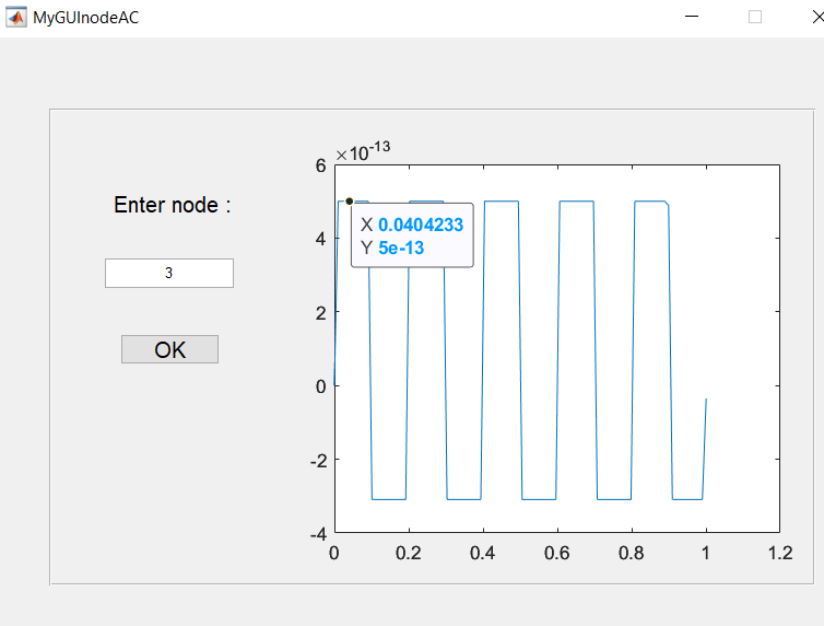
1. Exponential model
    - V(1)



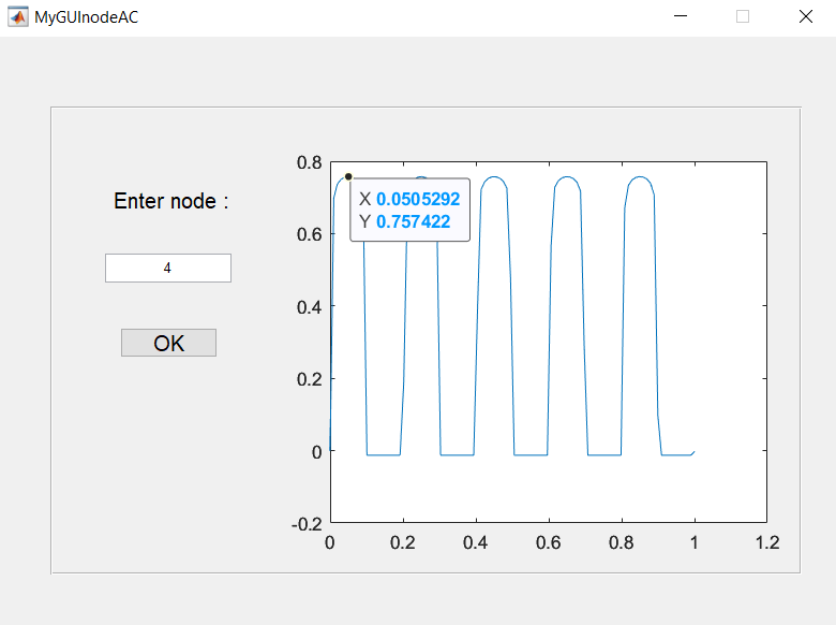
- V(2)


Enter node :
2
OK
X 0.0505292
Y 5.24183
6
5
4
3
2
1
0
-1
0
0.2
0.4
0.6
0.8
1

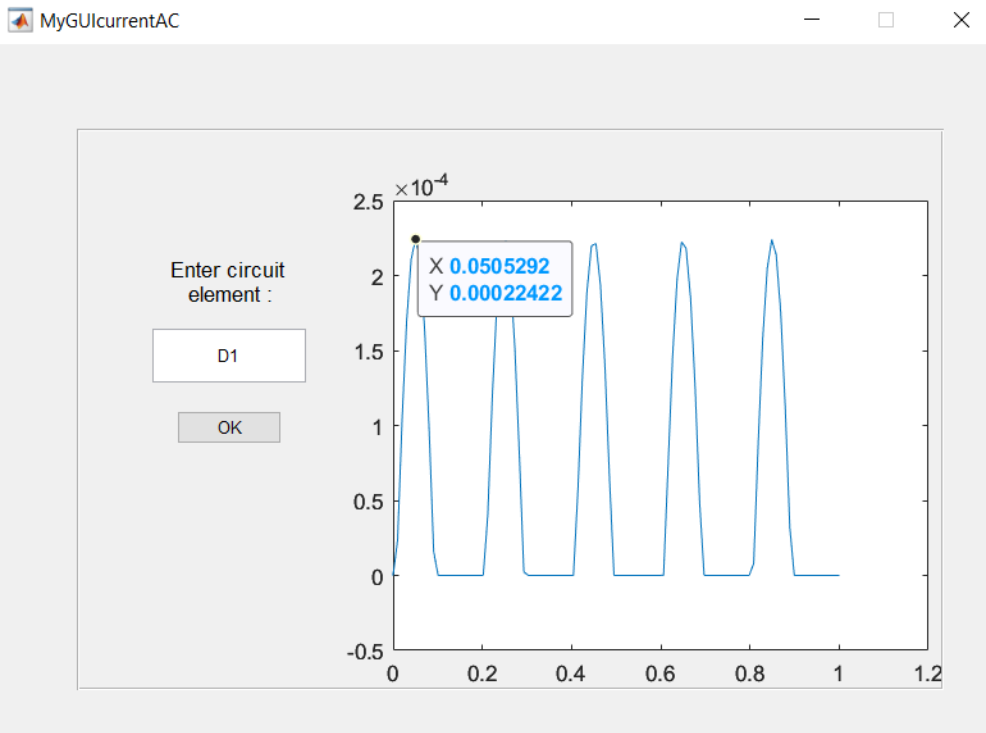
- V(3)


MyGUInodeAC
Enter node :
3
OK
×10-13
X 0.0404233
Y 5e-13
6
4
2
0
-2
-4
0
0.2
0.4
0.6
0.8
1
1.2

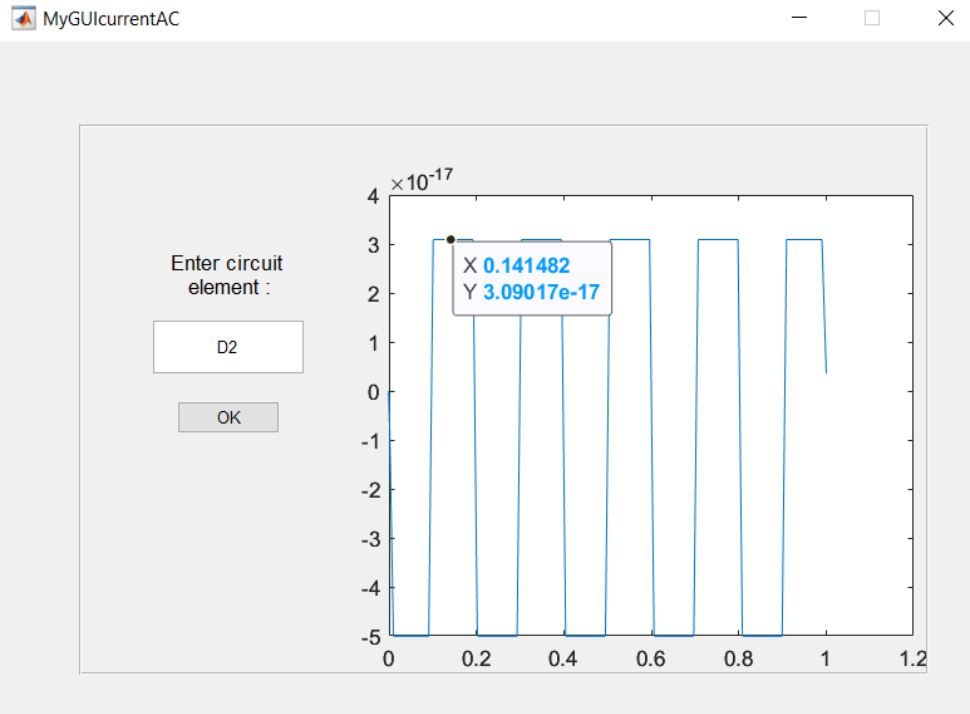
- V(4)


MyGUInodeAC
Enter node :
4
OK
X 0.0505292
Y 0.757422
0.8
0.6
0.4
0.2
0
-0.2
0
0.2
0.4
0.6
0.8
1
1.2

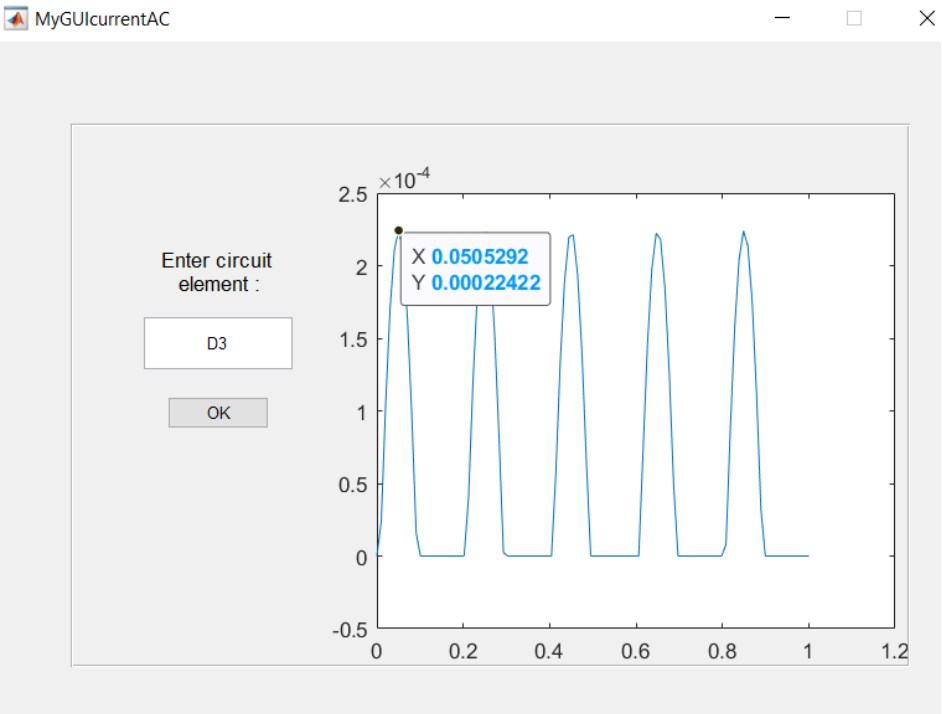
- I(D1)


MyGUIcurrentAC
Enter circuit element :
D1
OK
×10^-4
X 0.0505292
Y 0.00022422
2.5
2
1.5
1
0.5
0
-0.5
0
0.2
0.4
0.6
0.8
1
1.2

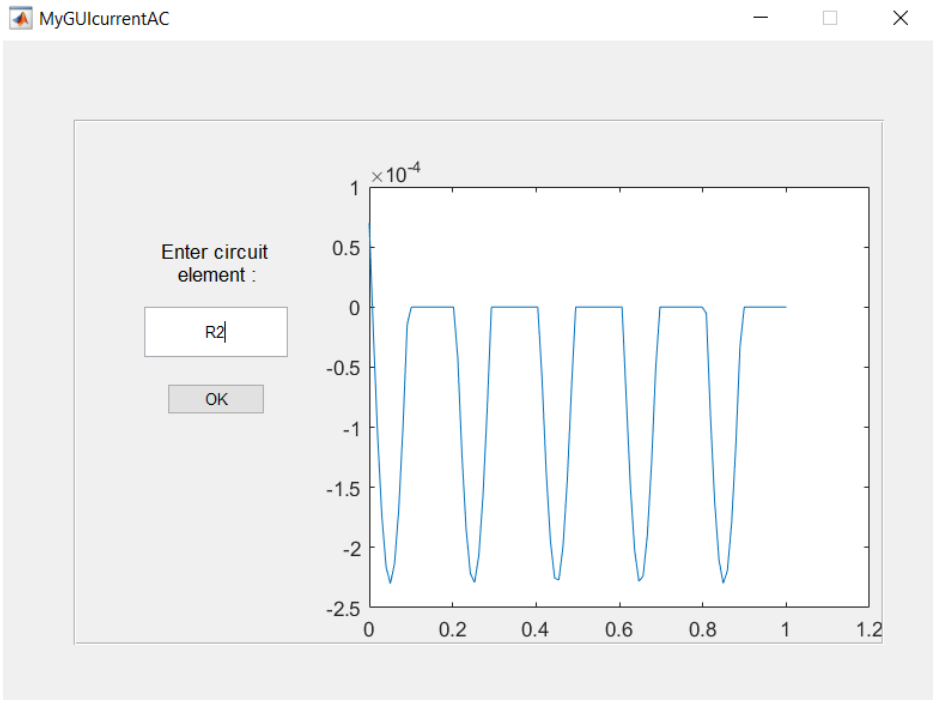
- I(D2)


MyGUIcurrentAC
Enter circuit element :
D2
OK
×10^-17
4
3
2
1
0
-1
-2
-3
-4
-5
X 0.141482
Y 3.09017e-17
0
0.2
0.4
0.6
0.8
1
1.2

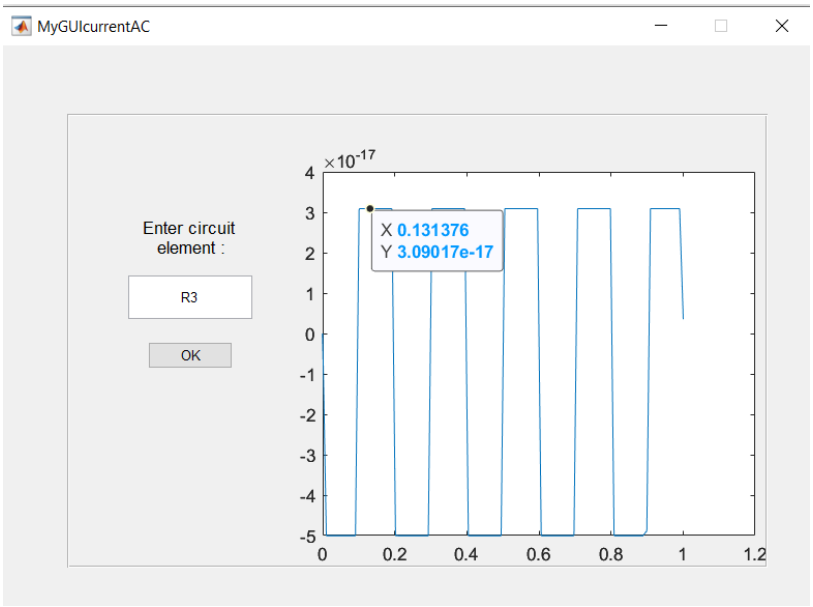
- I(D3)


MyGUIcurrentAC
Enter circuit element :
D3
OK
×10^-4
2.5
2
1.5
1
0.5
0
-0.5
X 0.0505292
Y 0.00022422
0
0.2
0.4
0.6
0.8
1
1.2

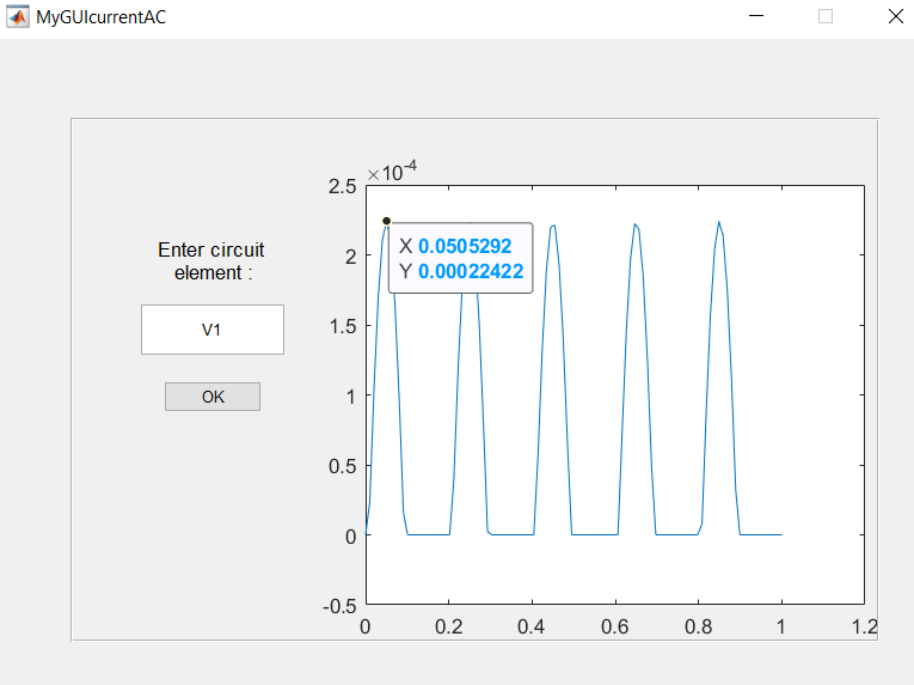
- I(R2)


MyGUIcurrentAC
Enter circuit element :
R2
OK
×10^-4
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
0.2
0.4
0.6
0.8
1
1.2

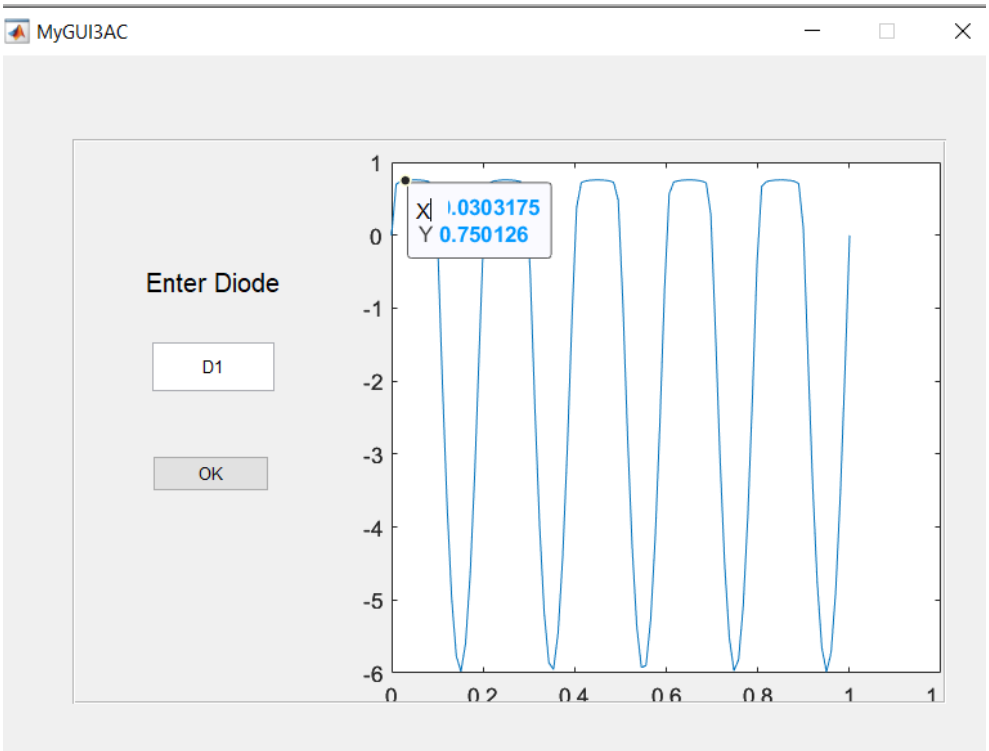
- I(R3)


MyGUIcurrentAC
Enter circuit element :
R3
OK
×10^-17
X 0.131376
Y 3.09017e-17
4
3
2
1
0
-1
-2
-3
-4
-5
0
0.2
0.4
0.6
0.8
1
1.2

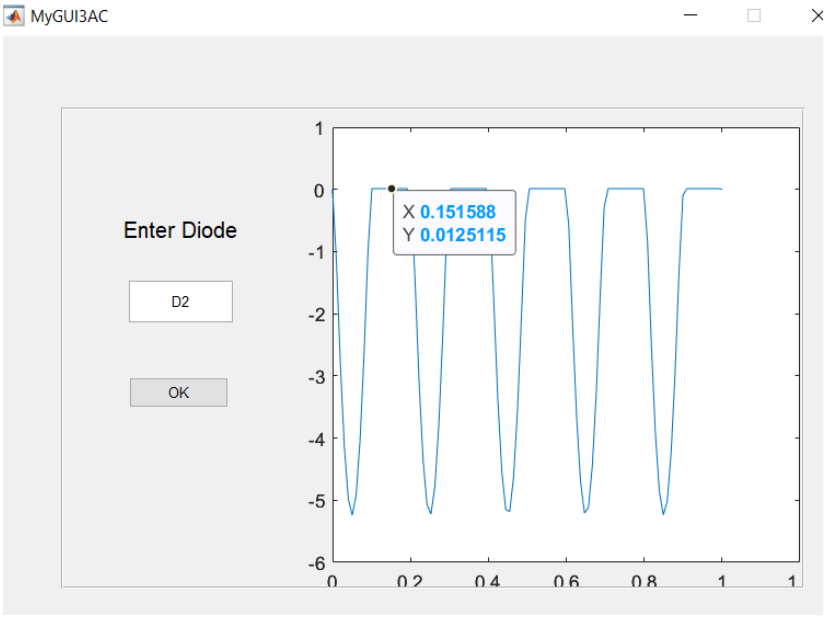
- I(V1)



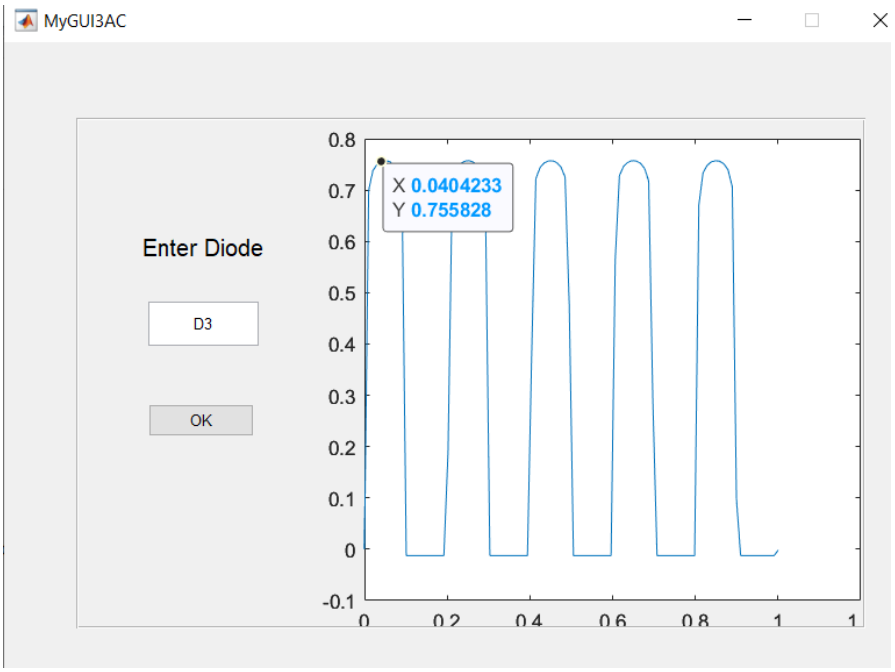
- V(D1:1)- V(D1:2)



- V(D2:1)- V(D2:2)

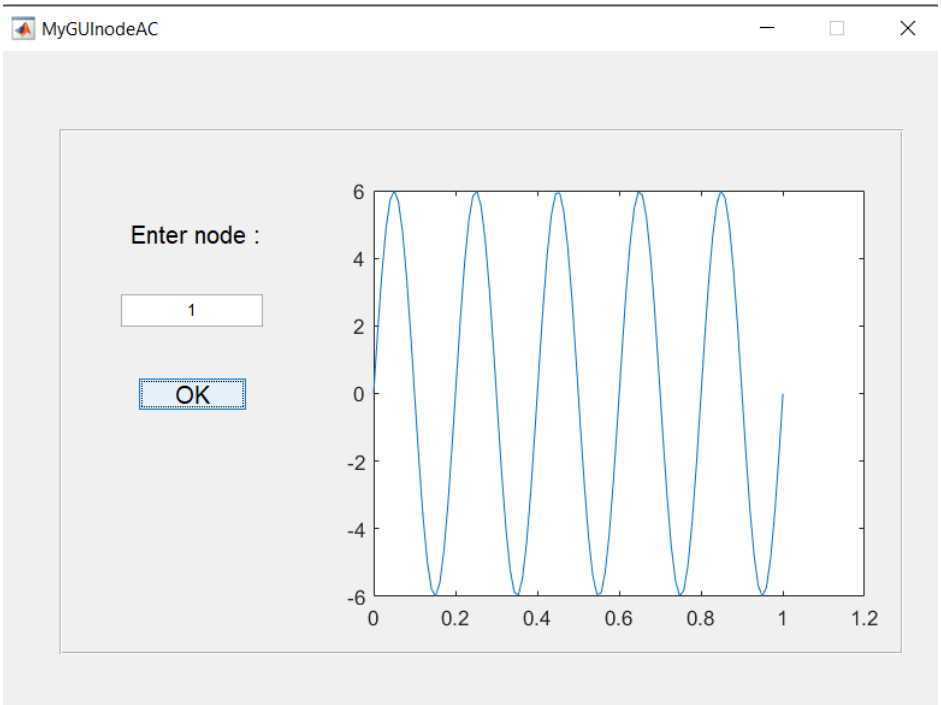


- V(D3:1)- V(D3:2)

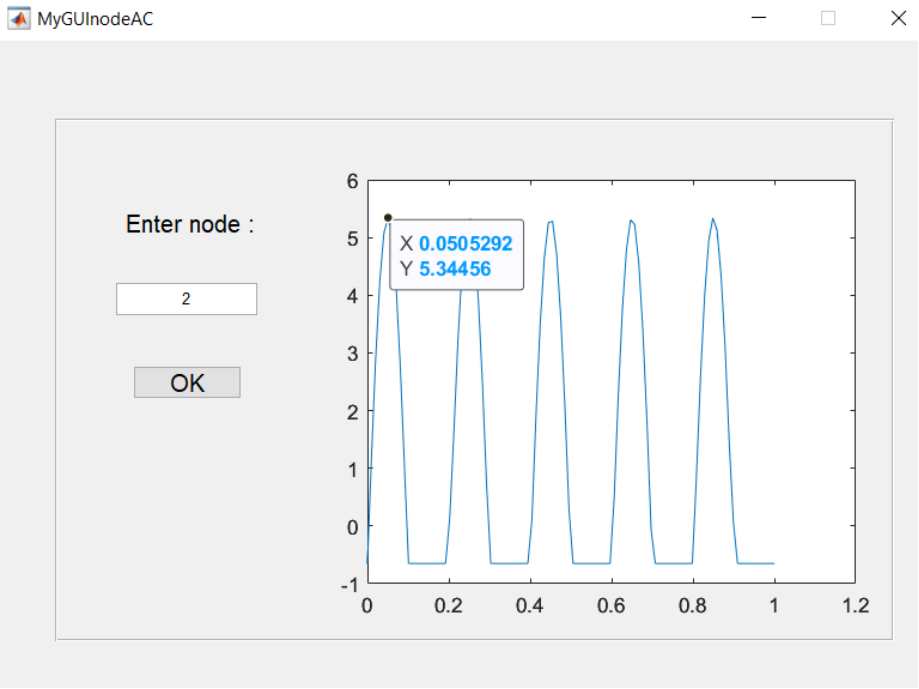


2. Piecewise linear

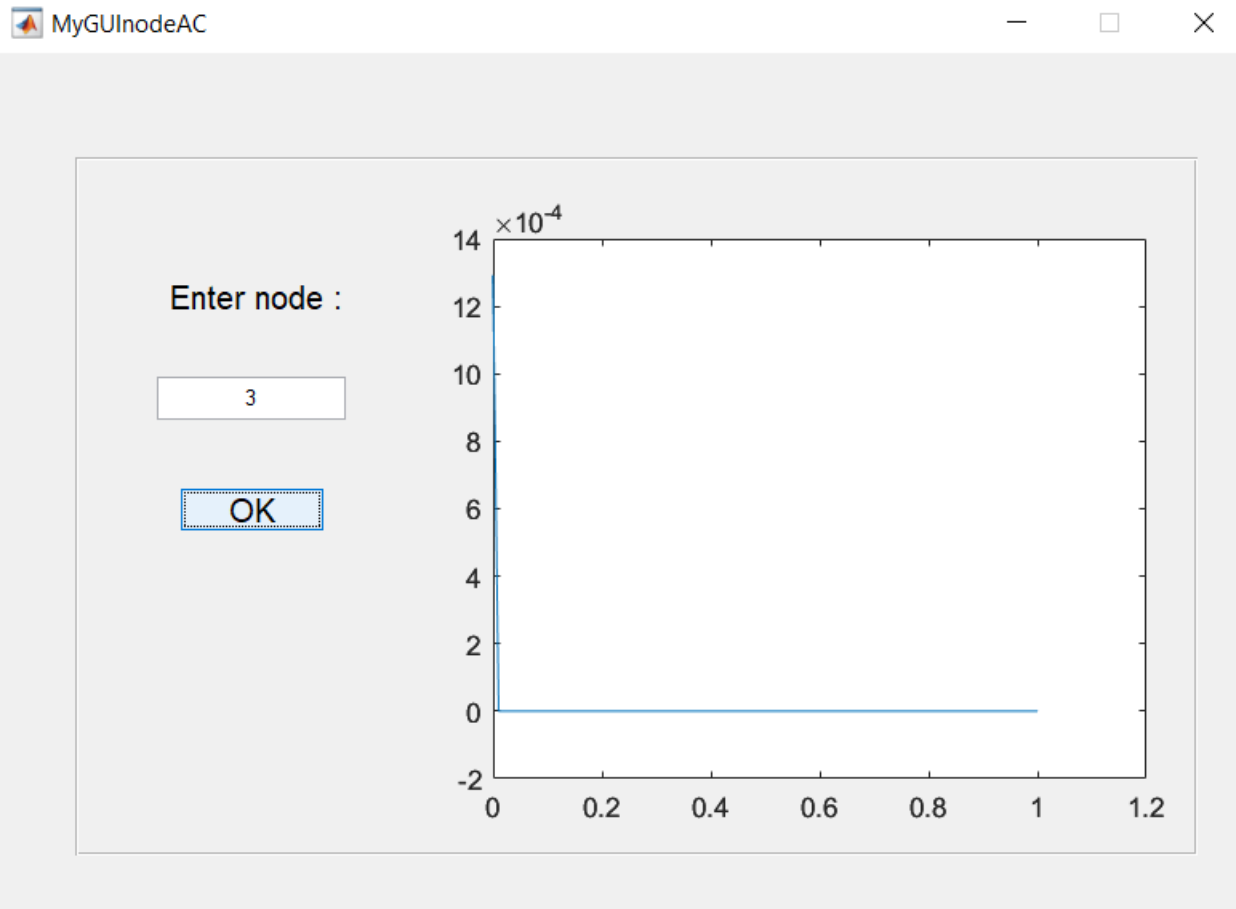
- V(1)



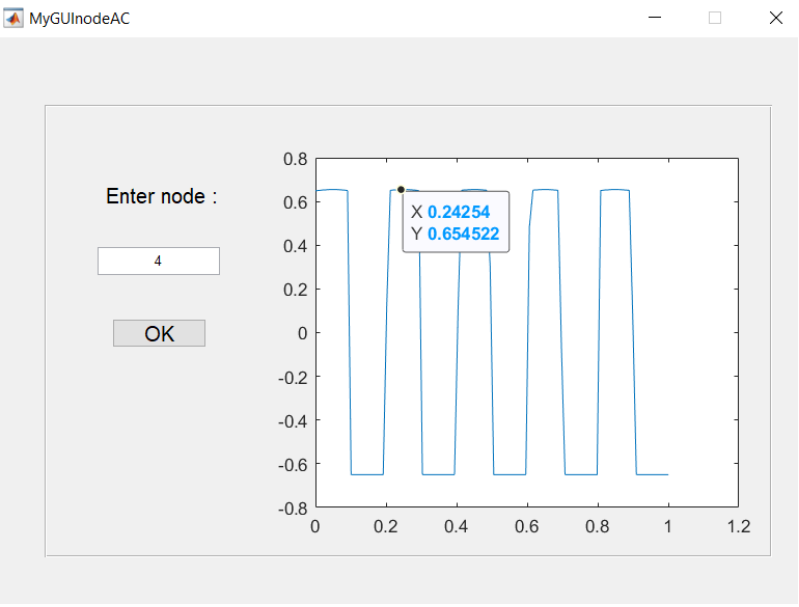
- V(2)



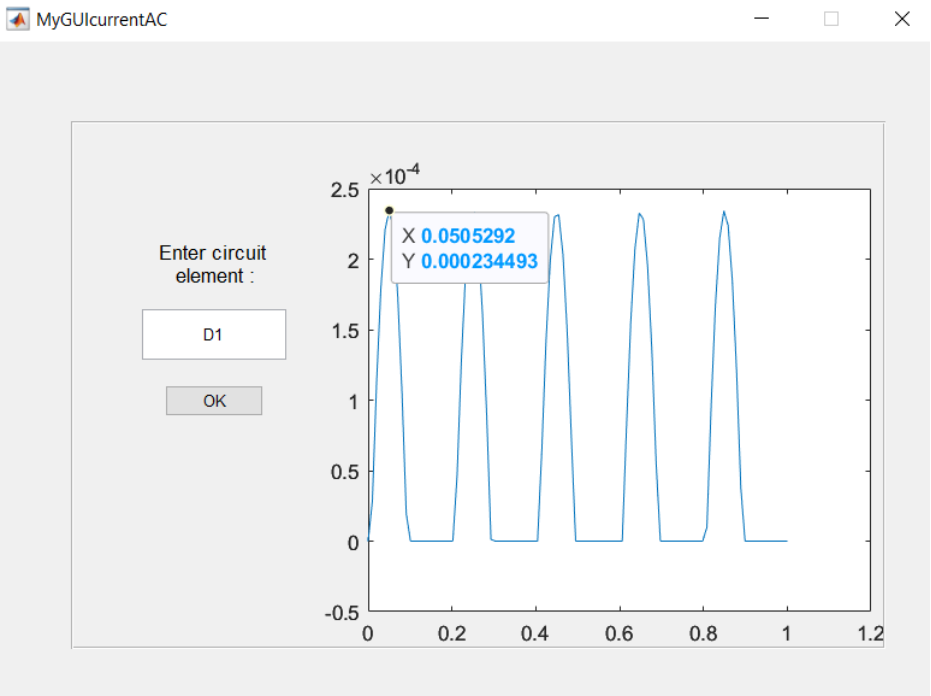
- V(3)


MyGUInodeAC
Enter node :
3
OK
×10^-4
14
12
10
8
6
4
2
0
-2
0
0.2
0.4
0.6
0.8
1
1.2

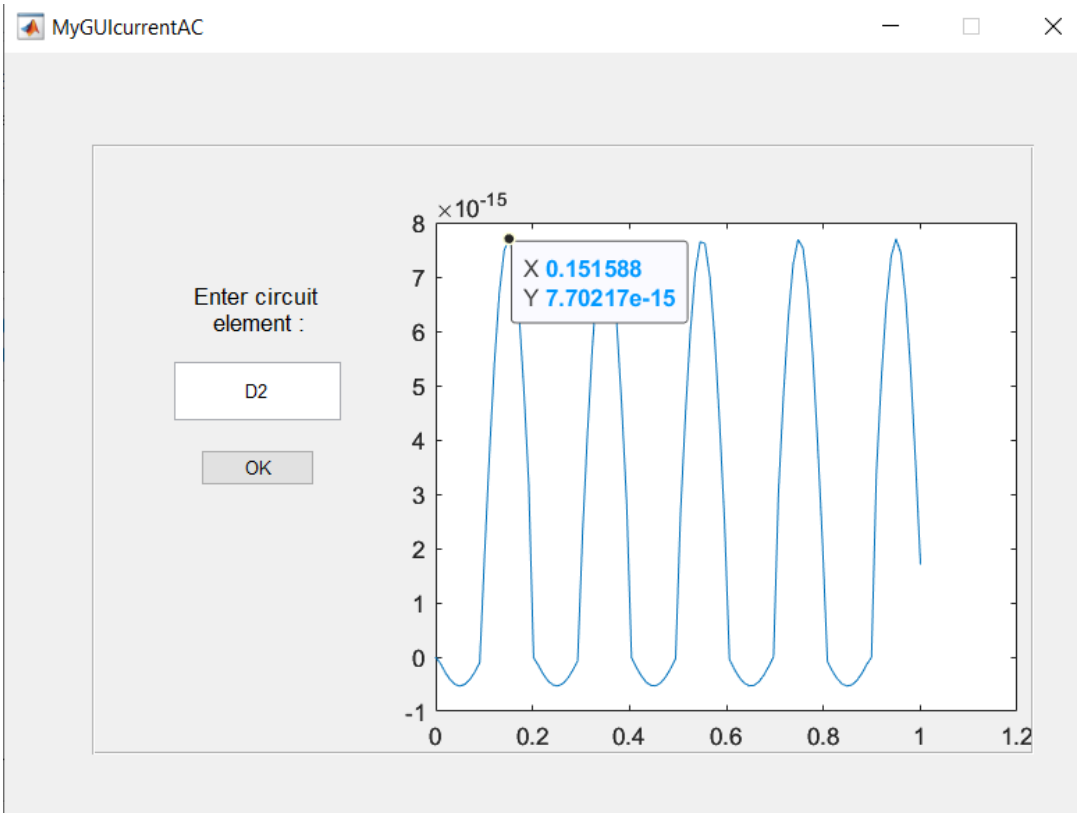
- V(4)


MyGUInodeAC
Enter node :
4
OK
X 0.24254
Y 0.654522
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.2
0.4
0.6
0.8
1
1.2

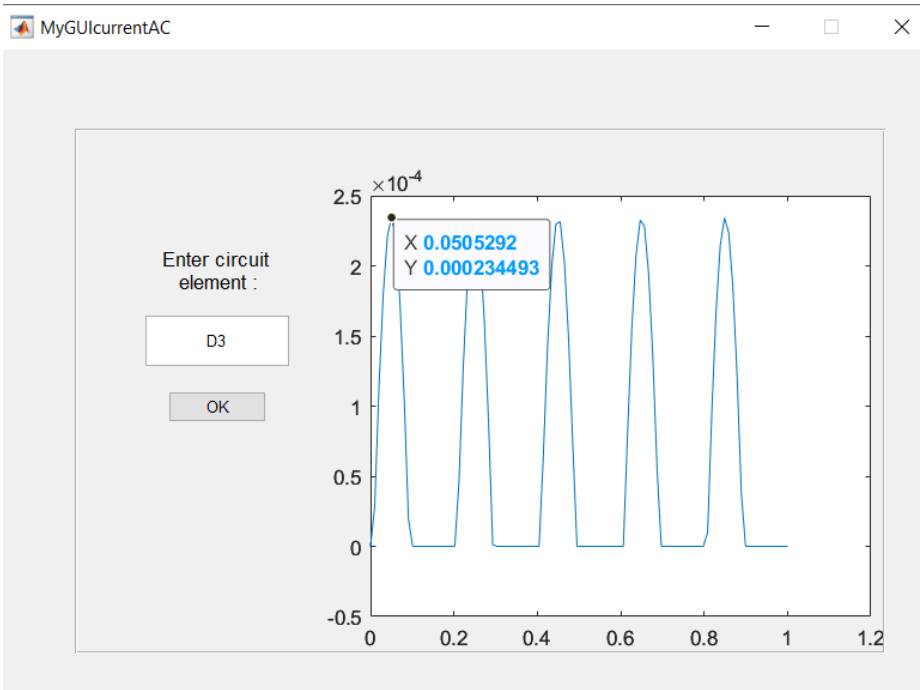
- I(D1)


MyGUIcurrentAC
Enter circuit element :
D1
OK
×10^-4
2.5
2
1.5
1
0.5
0
-0.5
0
0.2
0.4
0.6
0.8
1
1.2
X 0.0505292
Y 0.000234493

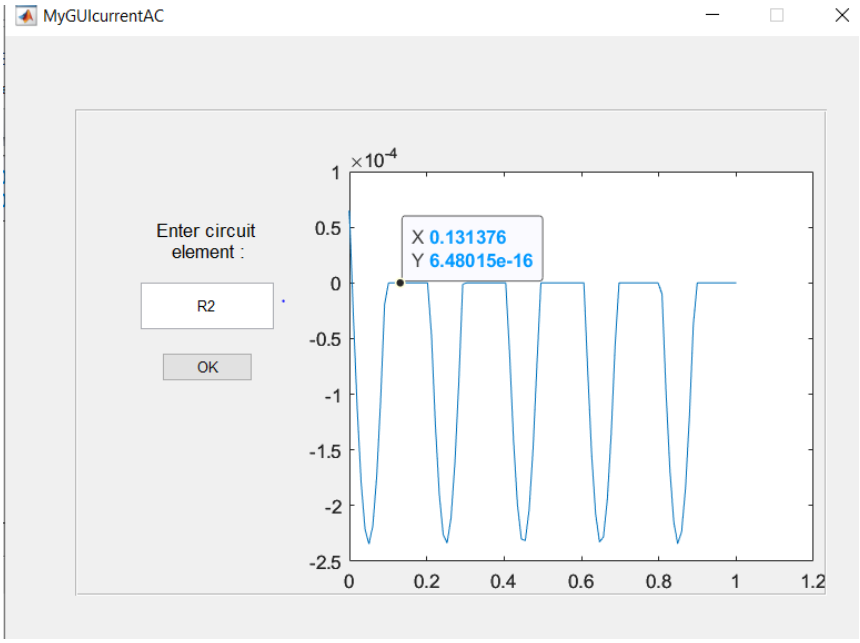
- I(D2)


MyGUIcurrentAC
Enter circuit element :
D2
OK
×10^-15
8
7
6
5
4
3
2
1
0
-1
0
0.2
0.4
0.6
0.8
1
1.2
X 0.151588
Y 7.70217e-15

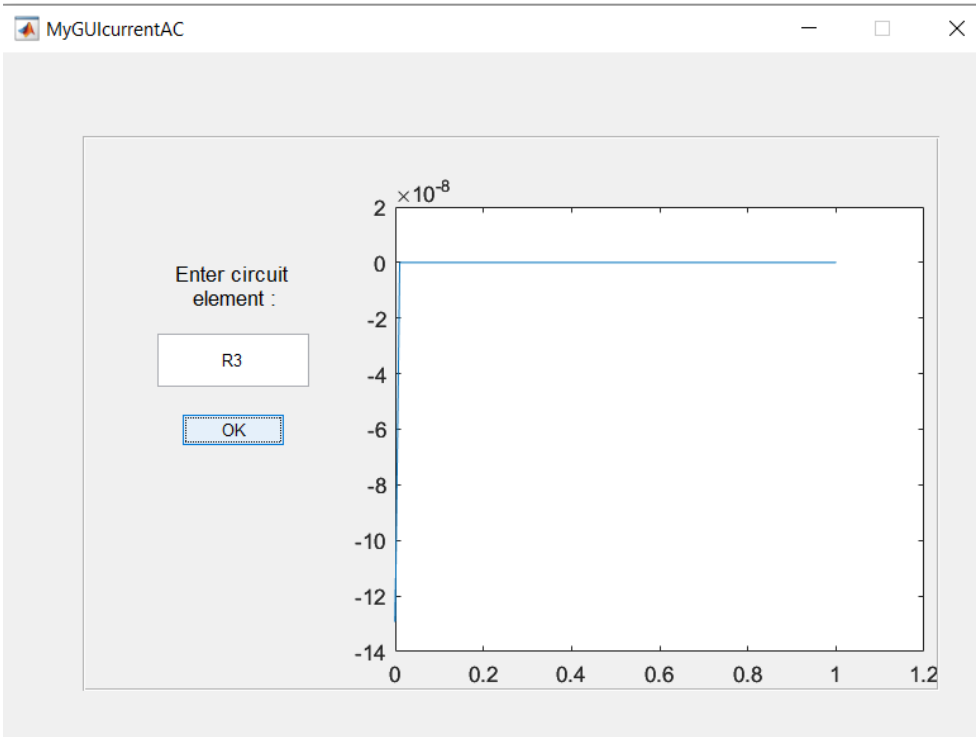
- I(D3)


MyGUIcurrentAC
Enter circuit element :
D3
OK
×10^-4
X 0.0505292
Y 0.000234493
2.5
2
1.5
1
0.5
0
-0.5
0
0.2
0.4
0.6
0.8
1
1.2

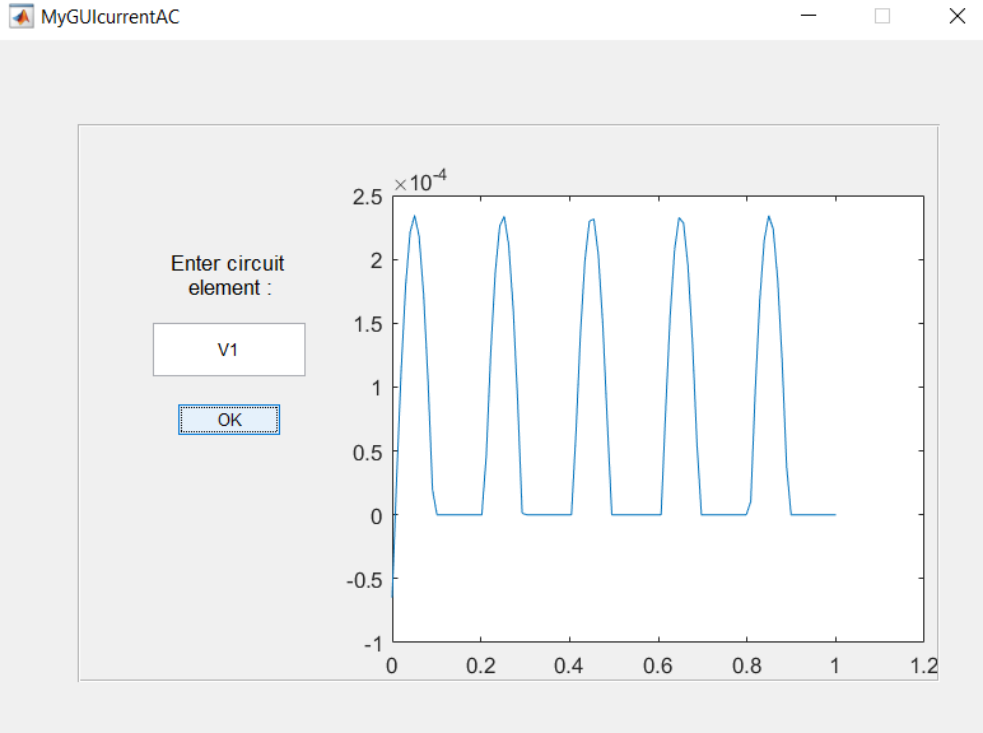
- I(R2)


MyGUIcurrentAC
Enter circuit element :
R2
OK
×10^-4
X 0.131376
Y 6.48015e-16
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
0.2
0.4
0.6
0.8
1
1.2

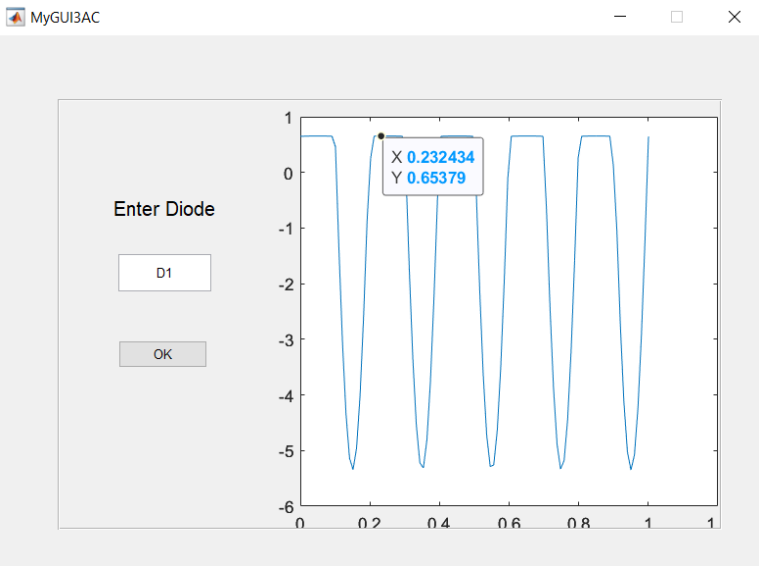
- I(R3)


MyGUIcurrentAC
Enter circuit element :
R3
OK
×10^-8
2
0
-2
-4
-6
-8
-10
-12
-14
0
0.2
0.4
0.6
0.8
1
1.2

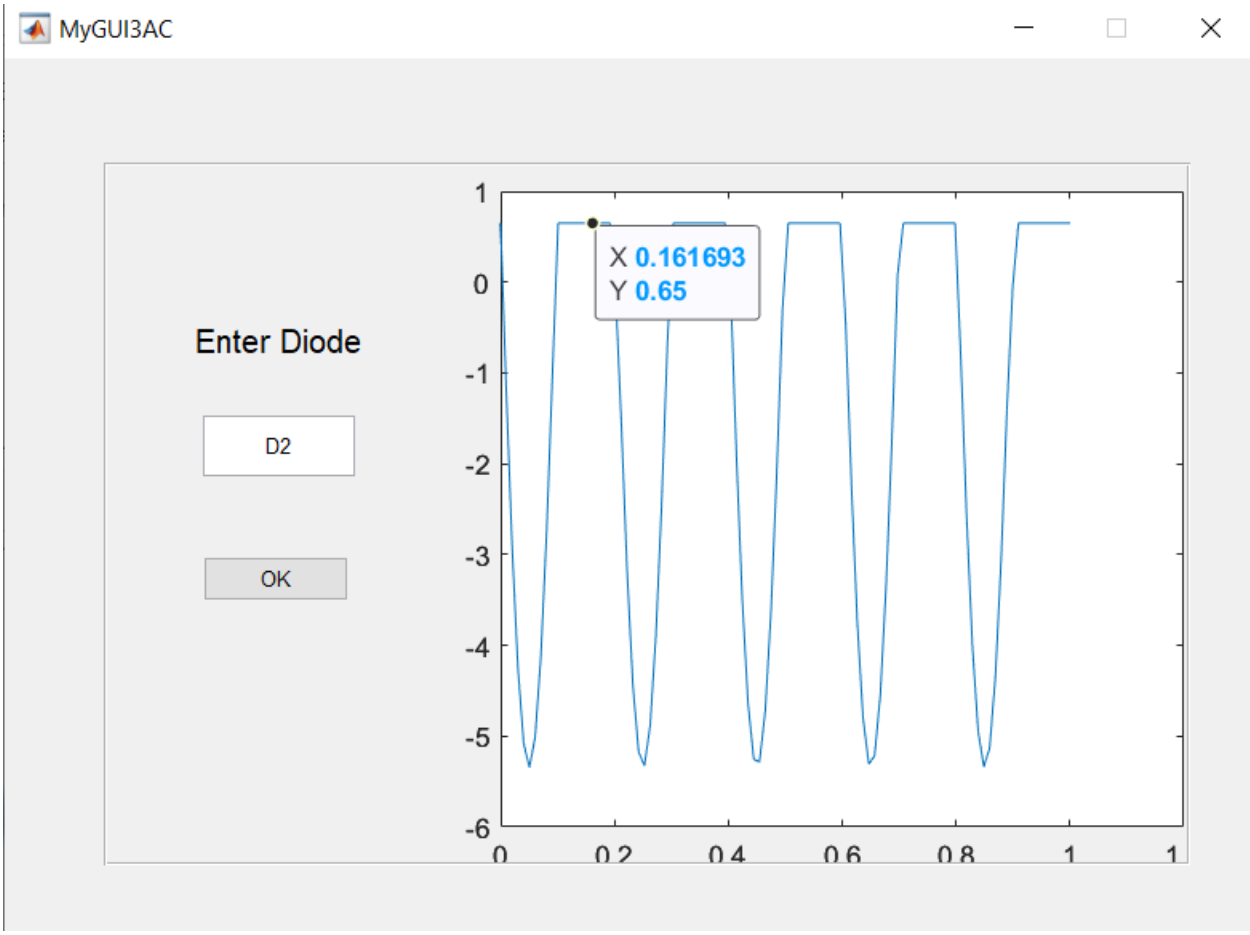
- I(V1)


MyGUIcurrentAC
Enter circuit element :
V1
OK
×10^-4
2.5
2
1.5
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1
1.2

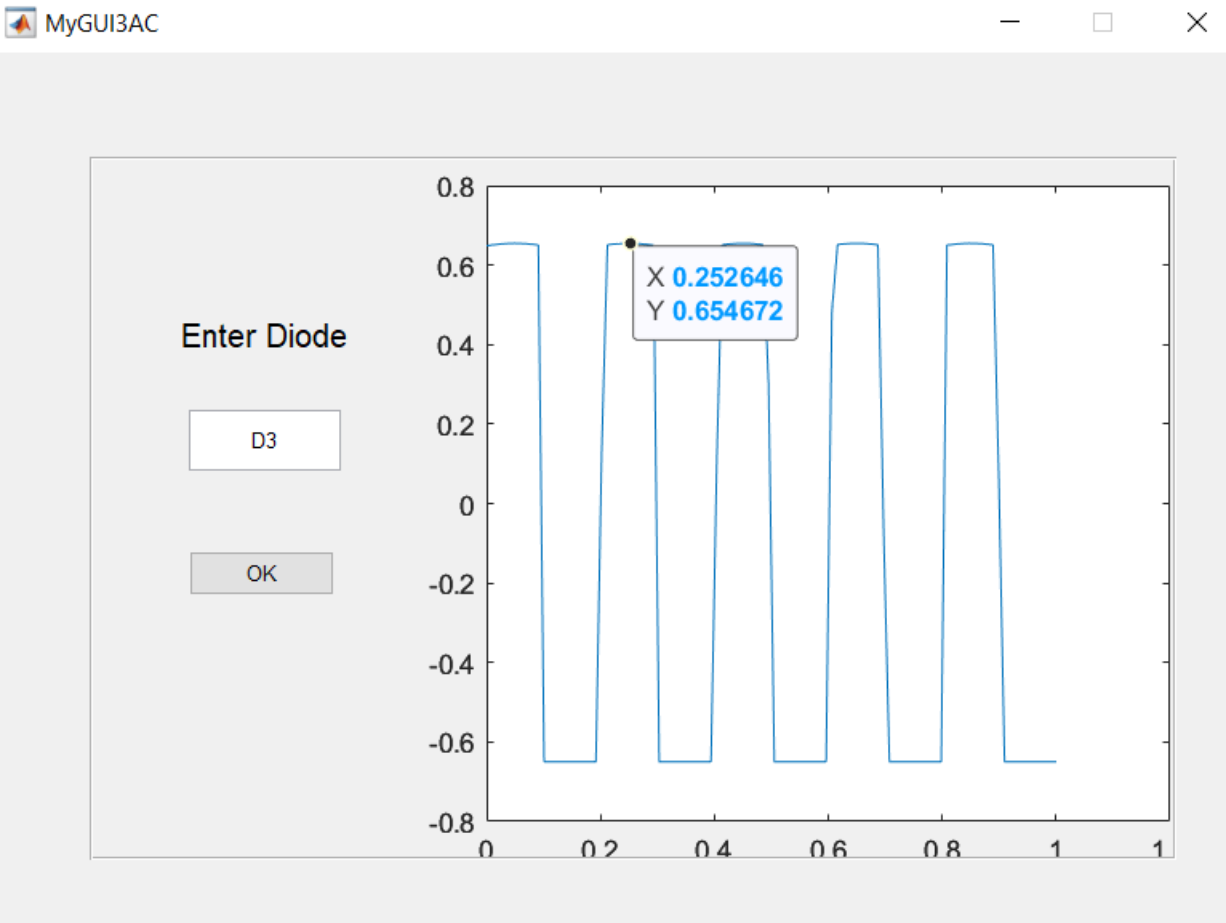
- V(D1:1)- V(D1:2)



- V(D2:1)- V(D2:2)

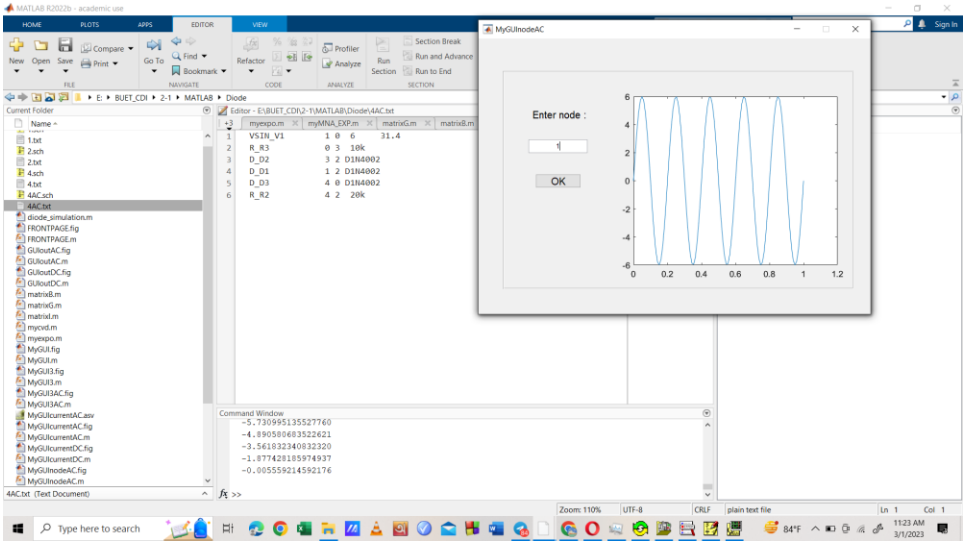


- V(D3:1)- V(D3:2)

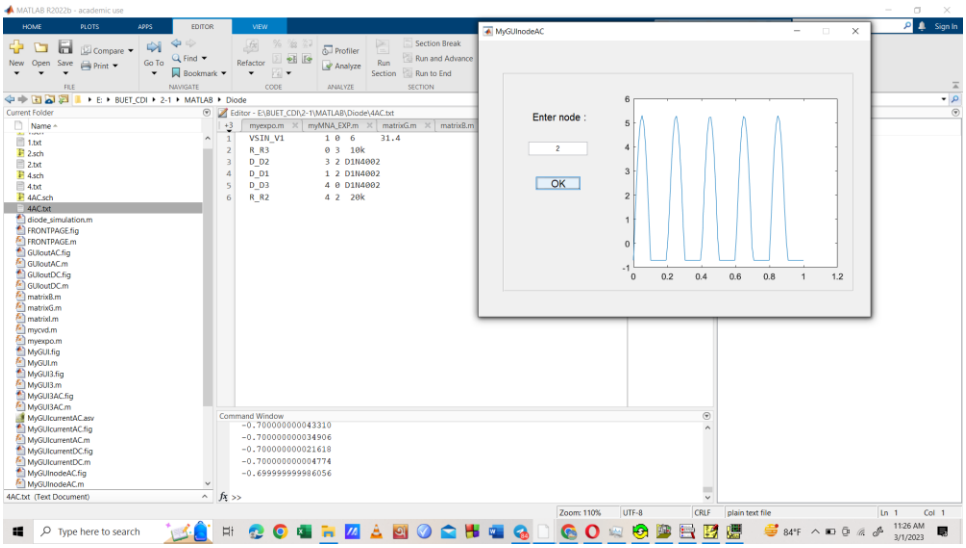


3. Constant voltage drop

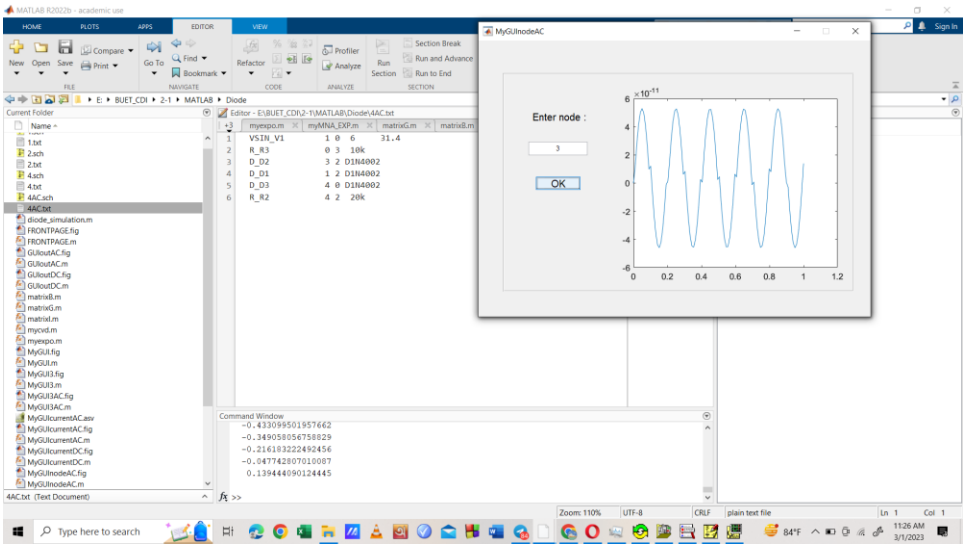
- V(1)


MyGUInodeAC
Enter node :
1
OK
Command Window
-5.73095135527760
-4.89058603522621
-3.56103234083232O
-1.87742815974937
-0.00555921459217

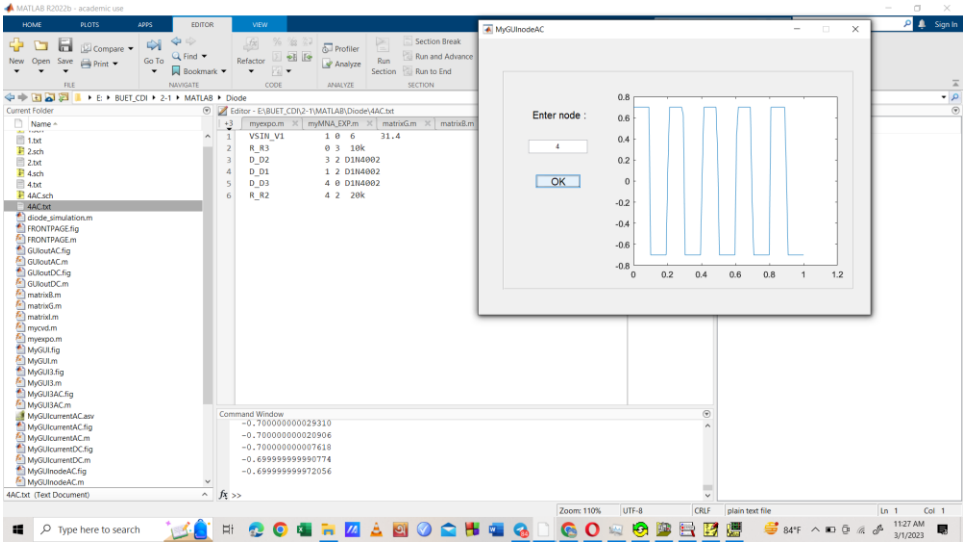
- V(2)


MyGUInodeAC
Enter node :
2
OK
Command Window
-0.700000000043310
-0.700000000034906
-0.700000000021610
-0.700000000004774
-0.699999999986056

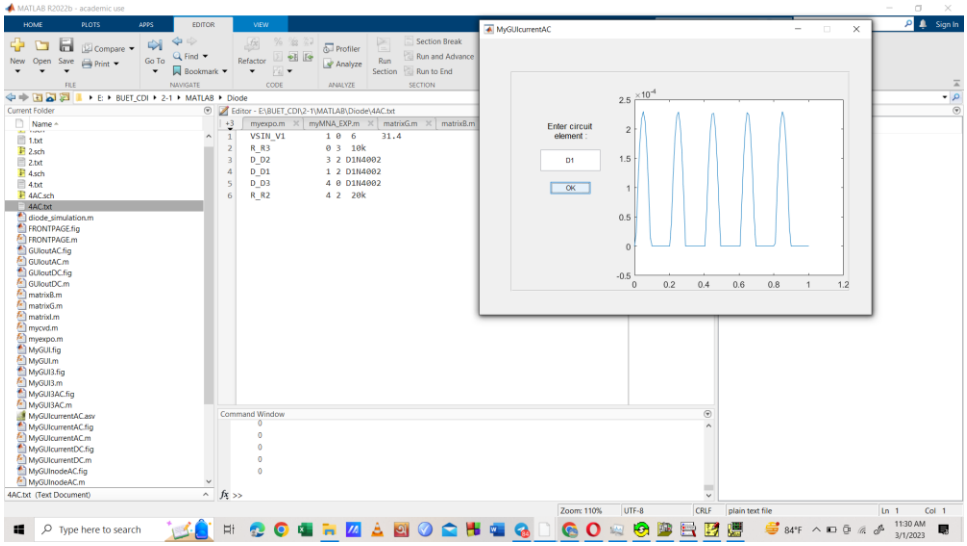
- V(3)


MyGUInodeAC
Enter node :
3
OK
×10^-11
6
4
2
0
-2
-4
-6
0
0.2
0.4
0.6
0.8
1
1.2

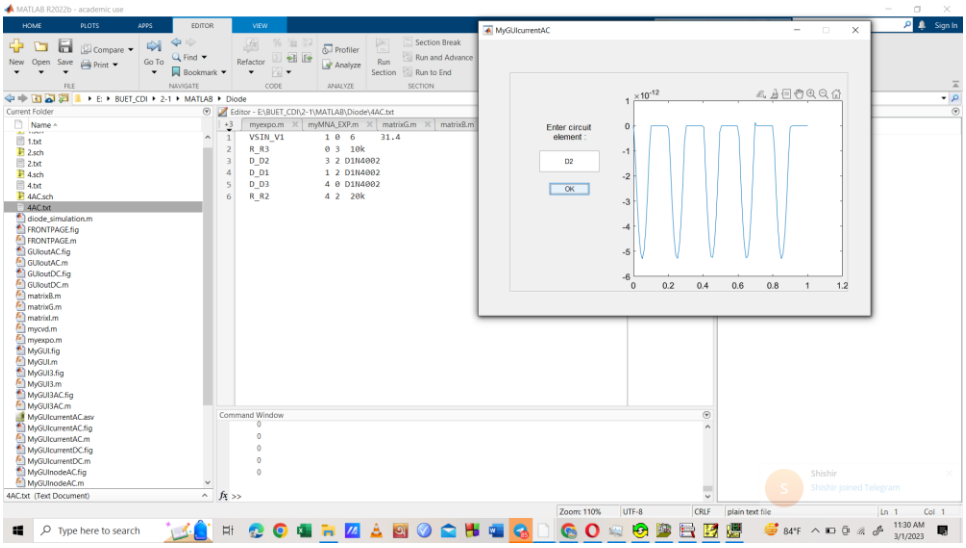
- V(4)


MyGUInodeAC
Enter node :
4
OK
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.2
0.4
0.6
0.8
1
1.2

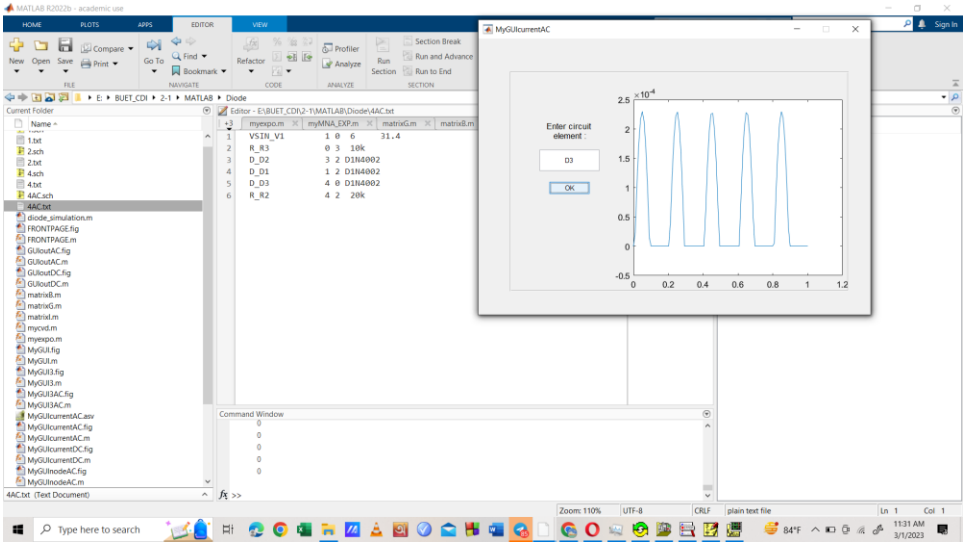
- I(D1)


MyGUIcurrentAC
Enter circuit element :
D1
OK

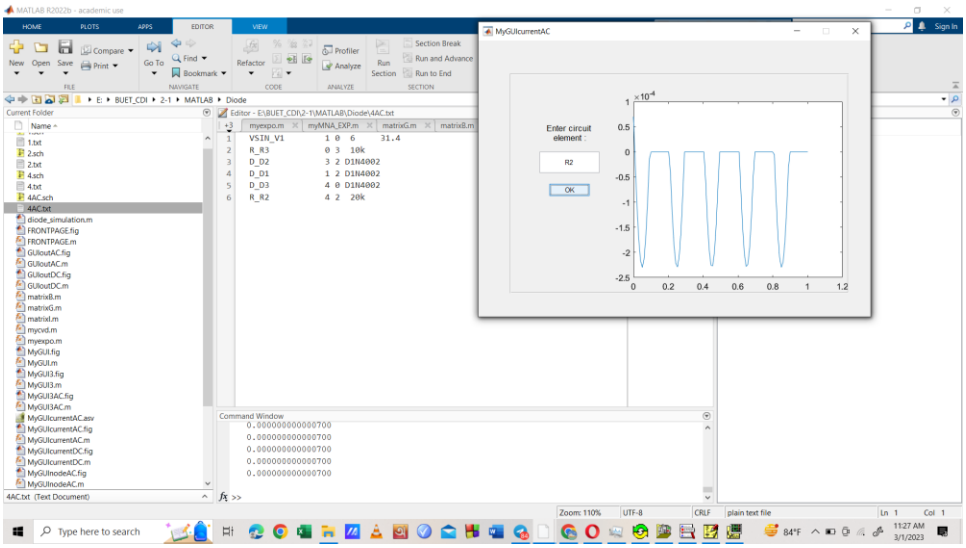
- I(D2)


MyGUIcurrentAC
Enter circuit element :
D2
OK

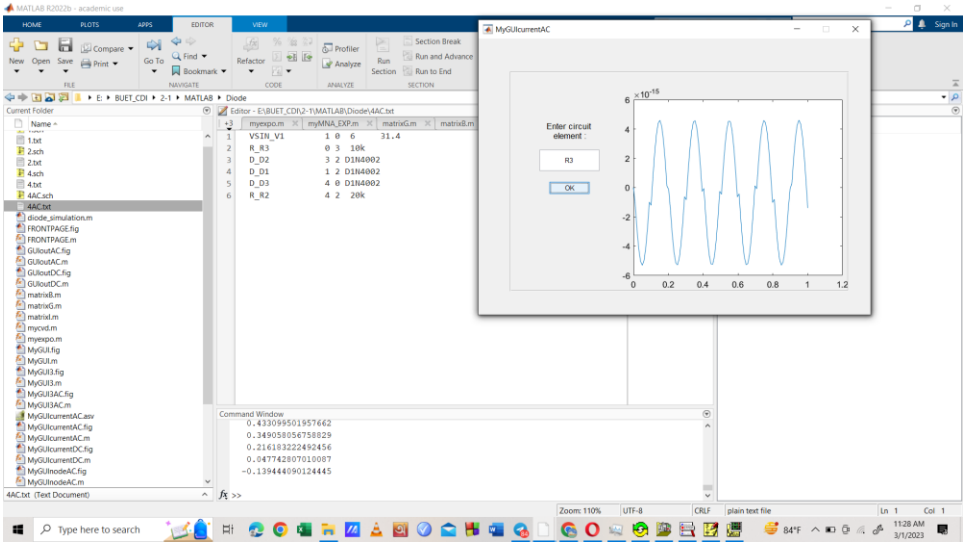
- I(D3)


MyGUIcurrentAC
Enter circuit element :
D3
OK
Command Window

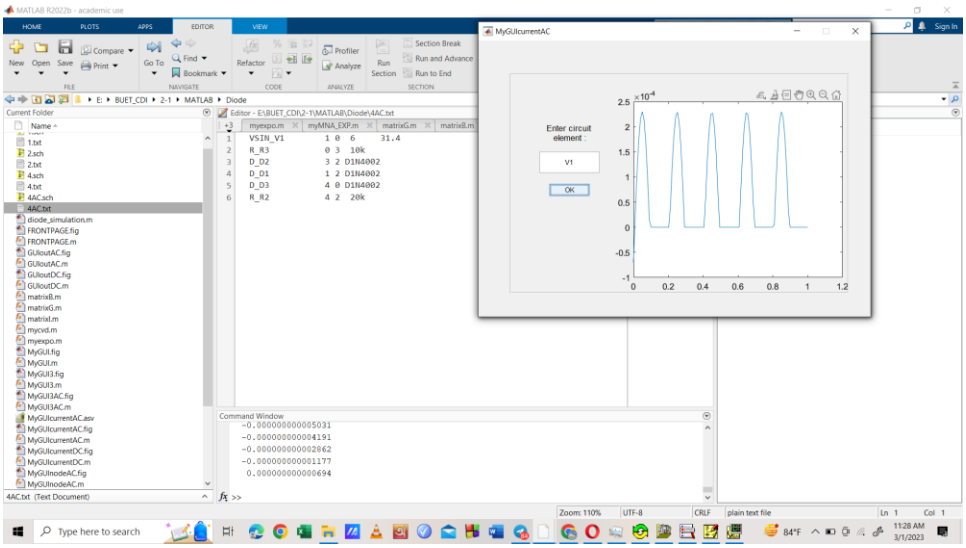
- I(R2)


MyGUIcurrentAC
Enter circuit element :
R2
OK
Command Window

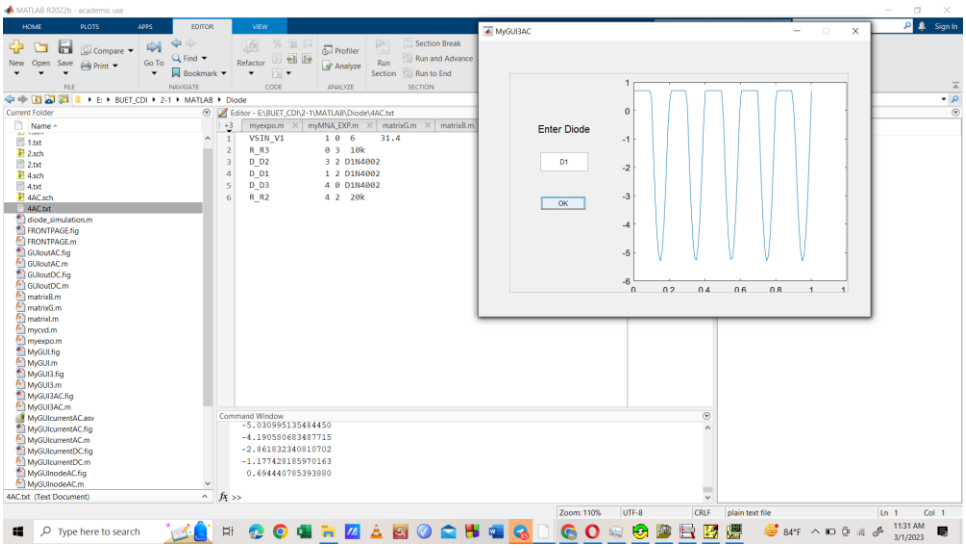
- I(R3)


MyGUIcurrentAC
Enter circuit element :
R3
OK
×10^-15
0.43309950195766 2
0.34905805675882 9
0.21610322249245 6
0.04774280701000 7
-0.13944409012444 5

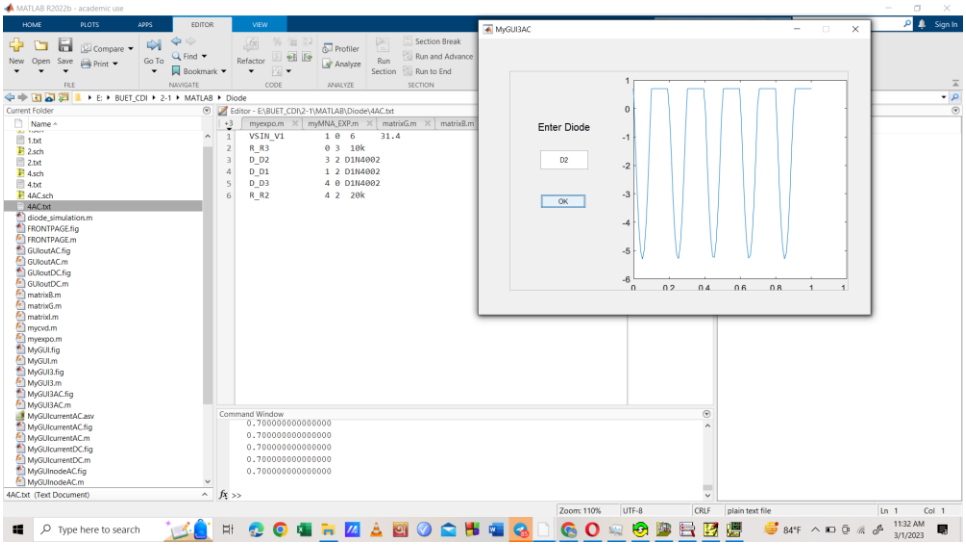
- I(V1)


MyGUIcurrentAC
Enter circuit element :
V1
OK
×10^-4
-0.00000000005031
-0.00000000004191
-0.00000000002862
-0.00000000001177
0.00000000000694

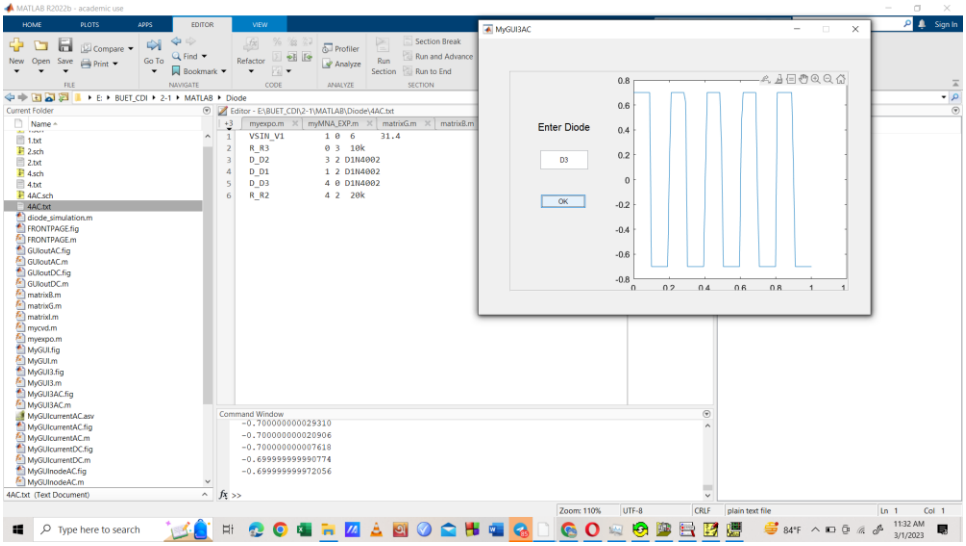
- V(D1:1)- V(D1:2)



- V(D2:1)- V(D2:2)

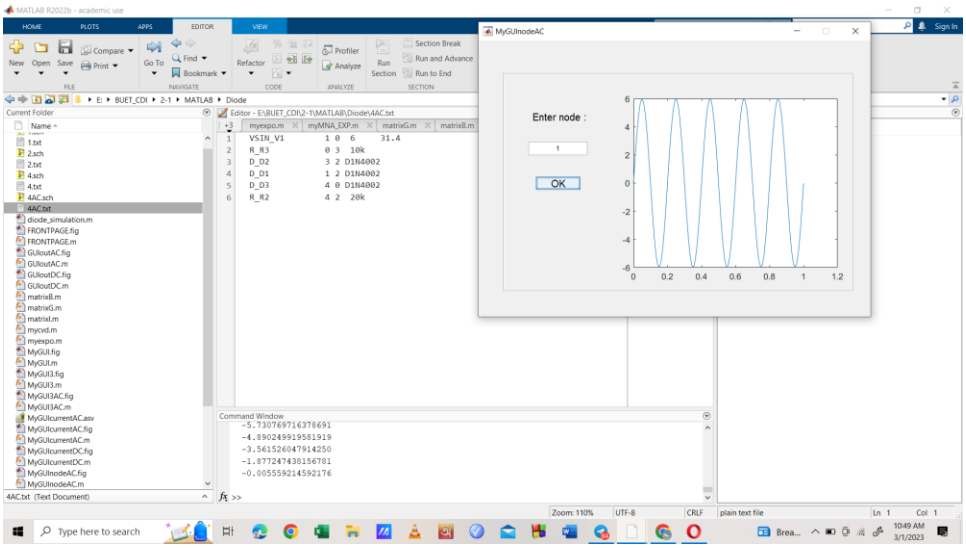


- V(D3:1)- V(D3:2)

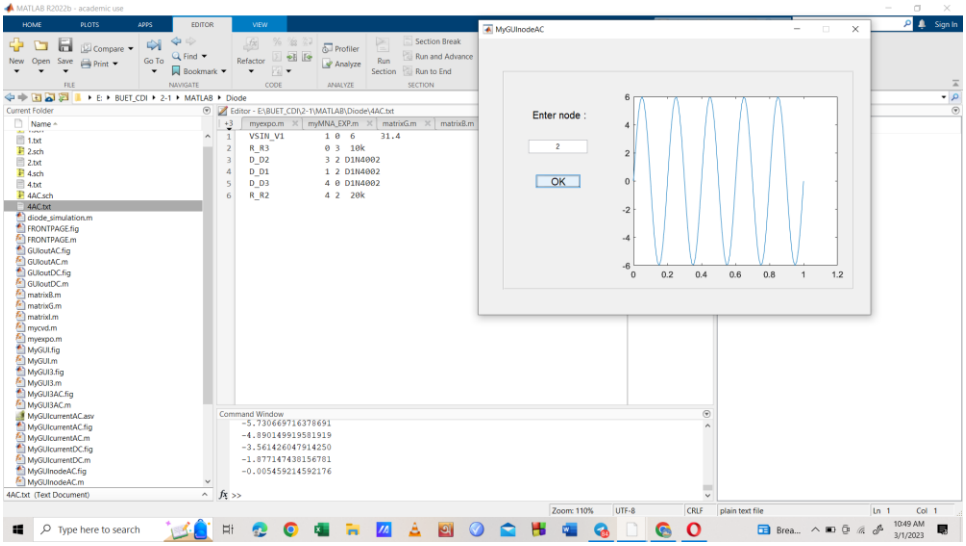


4. Ideal model

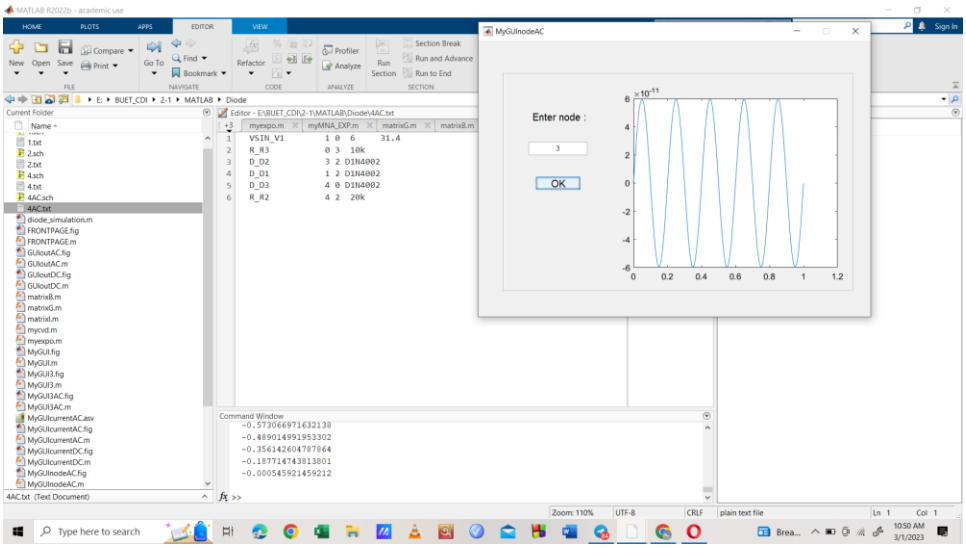
- V(1)



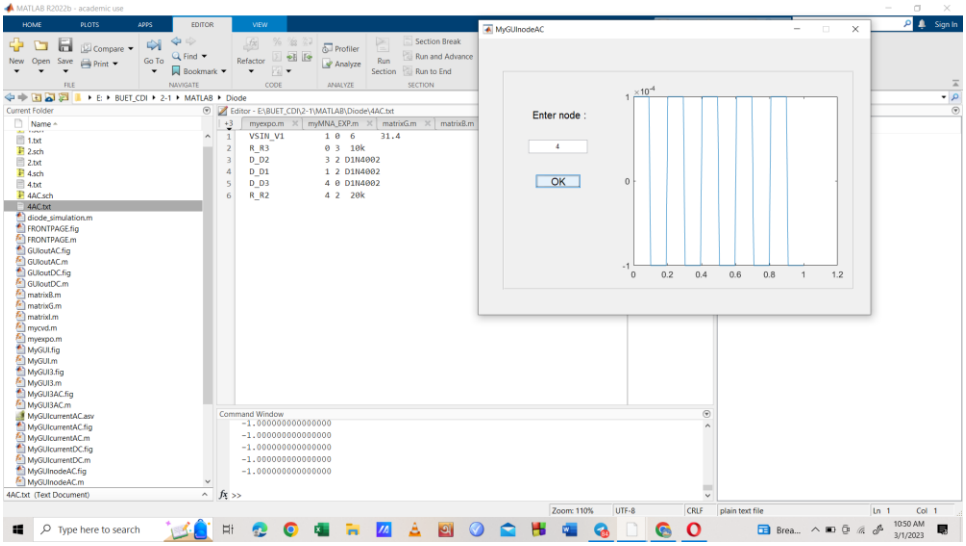
- V(2)


MyGUInodeAC
Enter node :
2
OK
Command Window
-5.730669716378691
-4.890149919581919
-3.561426047914250
-1.877147430156781
-0.005459214592176

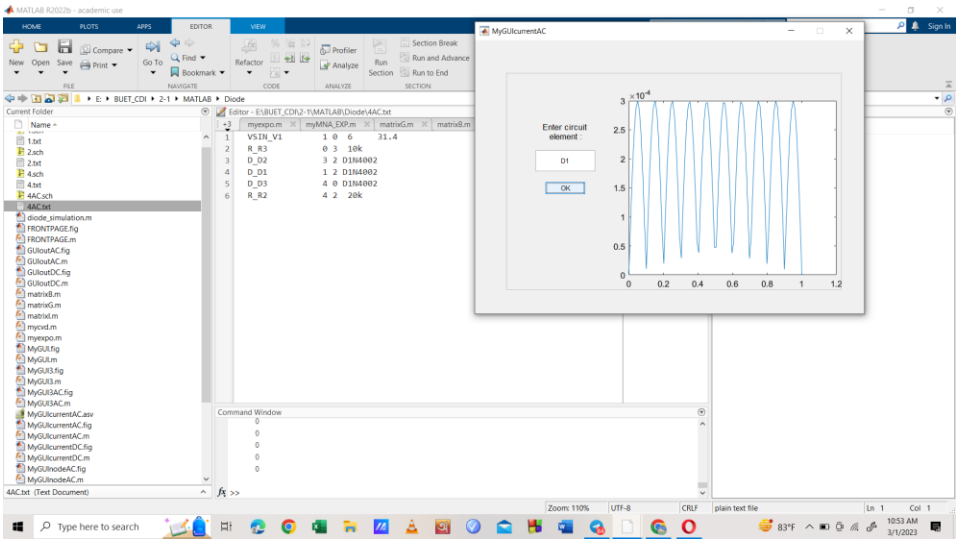
- V(3)


MyGUInodeAC
Enter node :
3
OK
Command Window
-0.573066971632138
-0.489014991953302
-0.356142604787864
-0.187714743813801
-0.000545921459212

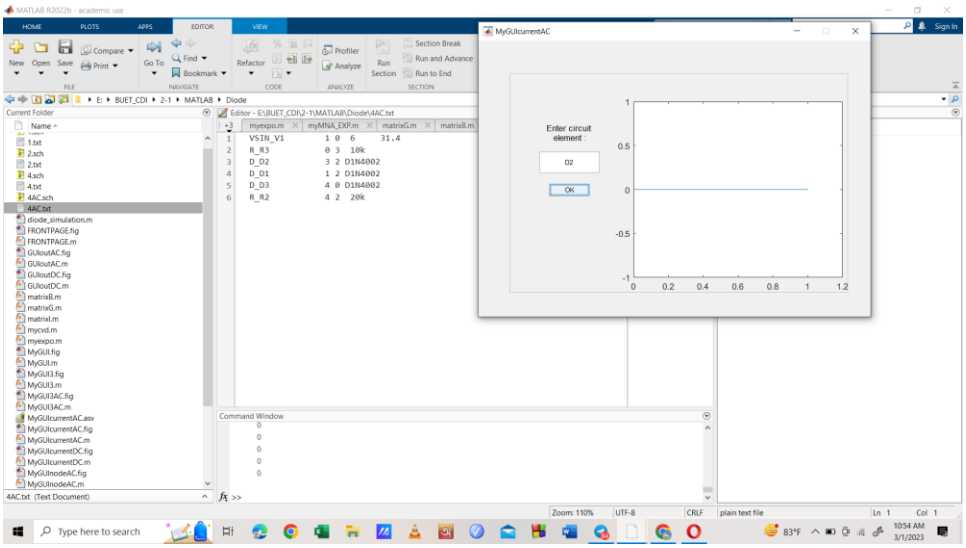
- V(4)


MyGUInodeAC
Enter node :
4
OK
VSIN_V1 1 0 6 31.4
R_R3 0 3 10k
D_D2 3 2 D1N4002
D_D1 1 2 D1N4002
D_D3 4 0 D1N4002
R_R2 4 2 20k

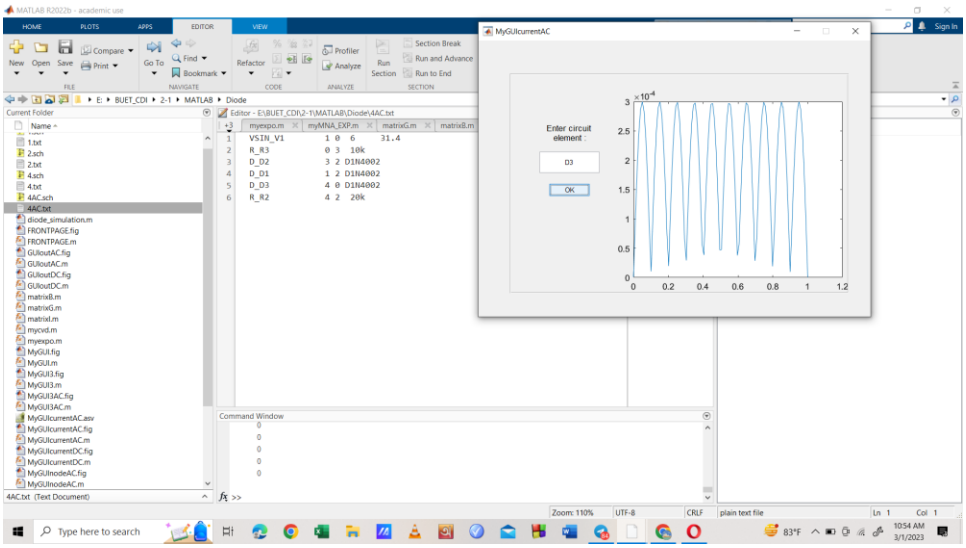
- I(D1)


MyGUIcurrentAC
Enter circuit element :
D1
OK
VSIN_V1 1 0 6 31.4
R_R3 0 3 10k
D_D2 3 2 D1N4002
D_D1 1 2 D1N4002
D_D3 4 0 D1N4002
R_R2 4 2 20k

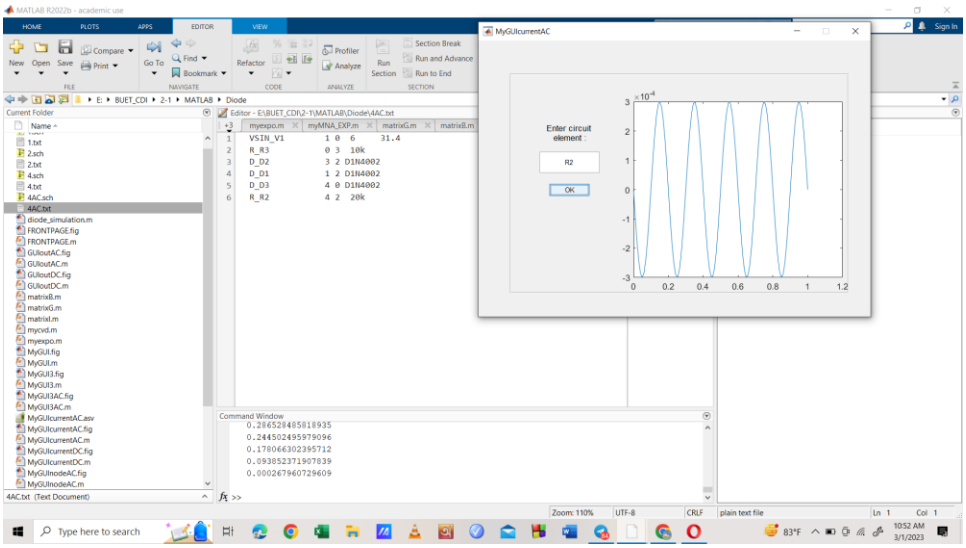
- I(D2)


MyGUIcurrentAC
Enter circuit element :
D2
OK

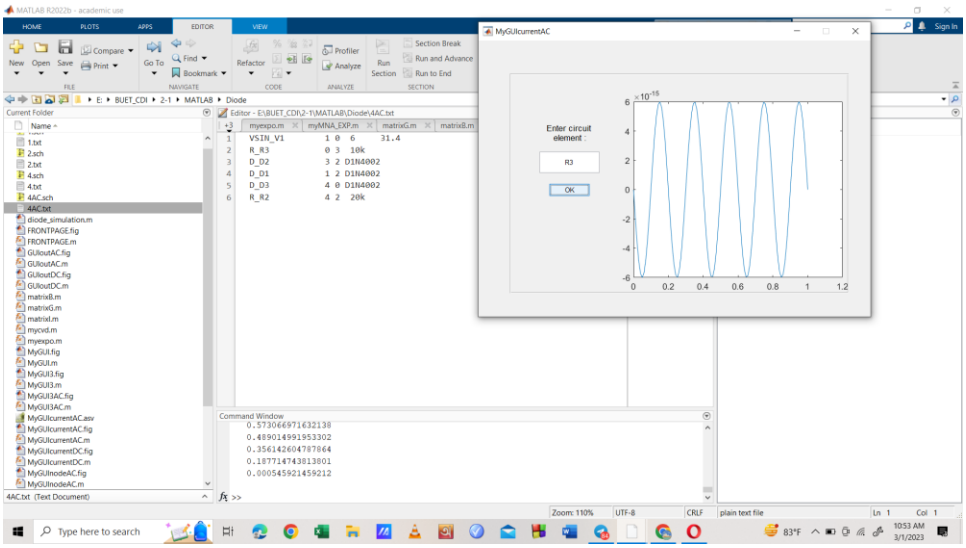
- I(D3)


MyGUIcurrentAC
Enter circuit element :
D3
OK

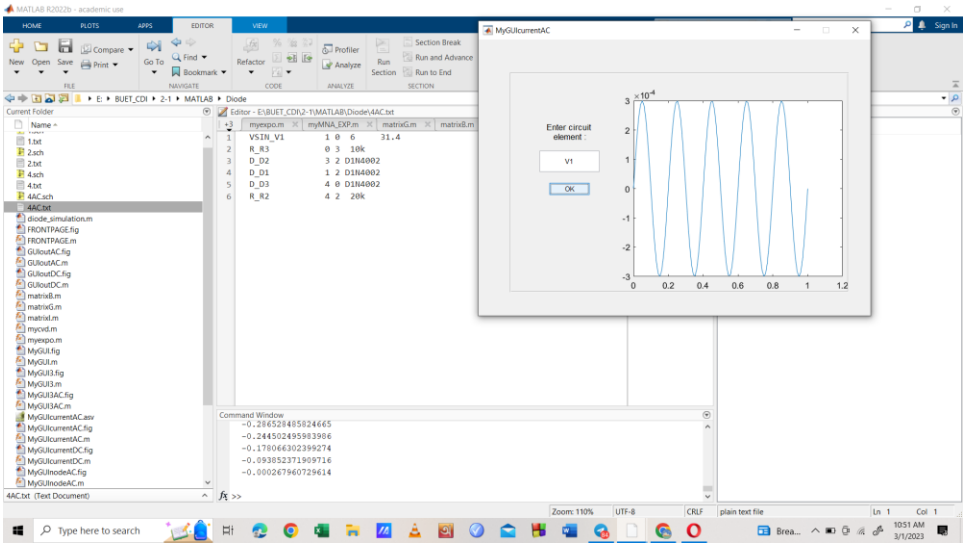
- I(R2)


MyGUIcurrentAC
Enter circuit element :
R2
OK

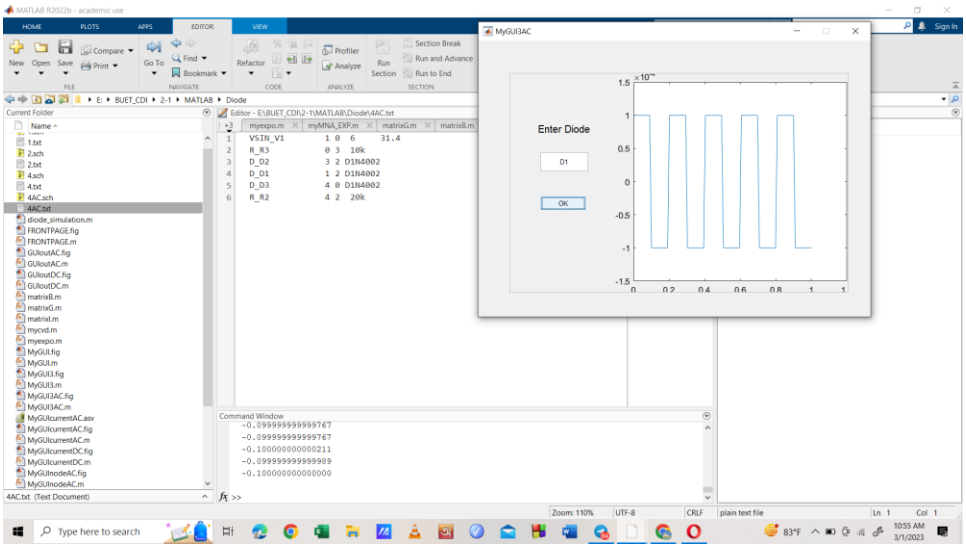
- I(R3)


MyGUIcurrentAC
Enter circuit element :
R3
OK

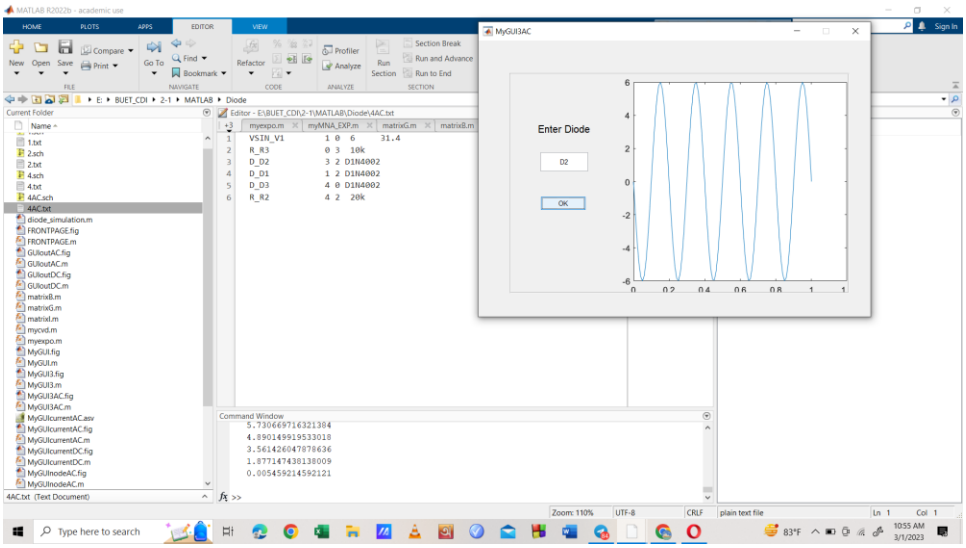
- I(V1)



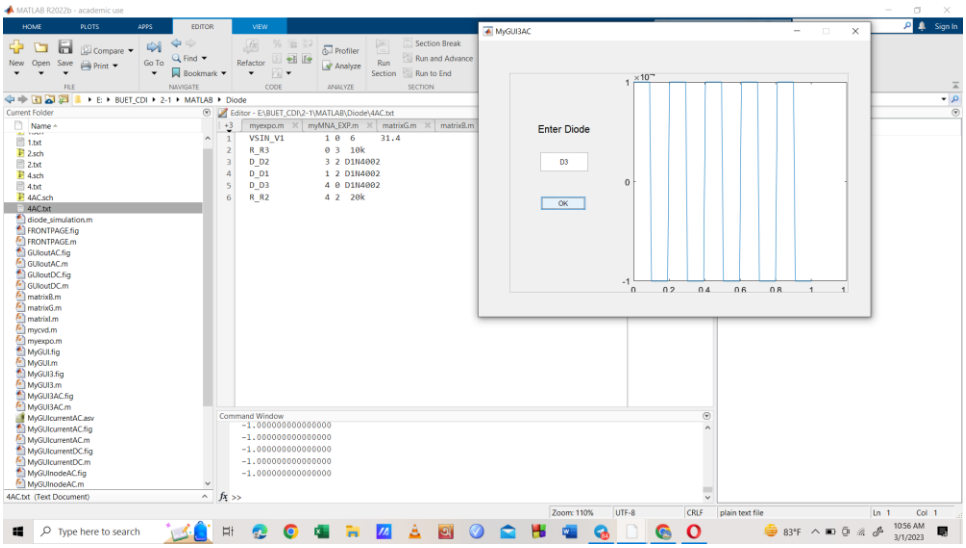
- V(D1:1)- V(D1:2)



- V(D2:1)- V(D2:2)



- V(D3:1)- V(D3:2)



Discussion

This project particularly aims to provide the user a better understanding on different models of diodes and how the results vary when we choose different models. For the graphic user interface, we chose to work with GUIDE, as the works of graphics in this project are kept fairly simple. As we compare the simulation output of PSPICE with the output of our project, we see that the results are quite similar and the waveshape of each plot also matches with PSPICE. However, there is minimal difference in the values of both case as PSPICE uses a very specific model of diode while completing its operation Nonetheless, the difference is negligible and we can reach the conclusion that our project is able to calculate the desired output.