BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Electrical and Electronic Engineering

Course No: EEE 212

Course Title: Numerical Technique Laboratory

Project Name—

Basic Netlist (Circuit Solver)

Submitted By—

1906176 - Tanvir Rahman

1906177- Ameer Hamja Ibne Jamal

Group 7

Section C

Level 2 , Term 1

Table of Contents

Objective

The main objectives of the project is to—

- Solve DC & AC circuit , to get the all node voltages and branch currents.
- Calculate Thevenin and Norton characteristics of particular nodes.
- Plot any branch or node characteristics along with source sweeping.
- Calculate resonant frequency, band width & Quality Factor of any simple series RLC ac circuit .
- Plot the voltage curve of elements in RLC circuit with frequency changing.

Introduction

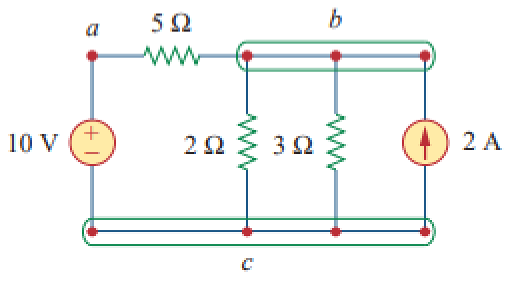
Circuit solver is a very useful tool for an electrical engineer. Numerical calculations can be solved easily by this kind of tools. Here in the same way, this project can help us to solve any circuits. If we are to calculate normal DC or AC circuit by conventional nodal or mesh analysis , it is time consuming process. This project needs the netlist code of the circuit , and it is able to give all the node voltages .Also it can deal with dependent sources as well. Moreover, Thevenin and Norton analysis, sweeping between two particular nodes are also be known and shown by this app. From the second part, we will be able to know resonant frequency, band width and quality factor of a particular RLC series circuit. Here all the codes are implemented by MATLAB software. We have made two GUI's using MATLAB Appdesigner and finally merged them in a single application.

Theory

This project will enlighten us to solve circuits easily.For implementing the project properly , we had to use some basic property of electrical matters as follows :

✓ Node & Branches—

A branch represents a single element such as a voltage source or a resistor. A node is the point of connection between two or more branches.



Basically, a branch represents any two-terminal element. In this circuit,it has five branches, namely, the 10-V voltage source, the 2-A current source, and the three resistors. And a,b & c are the nodes.

- ✓ KCL –

Kirchhoff's current law (KCL) states that the algebraic sum of currents entering a node (or a closed boundary) is zero.Mathematically , KCL implies that,

$$\sum_{n=1}^{N} i_n = 0$$

where N is the number of branches connected to the node and is the nth current entering (or leaving) the node. By this law, currents entering a node may be regarded as positive, while currents leaving the node may be taken as negative or vice versa.

- ✓ KVL—

Kirchhoff's voltage law (KVL) states that the algebraic sum of all voltages around a closed path (or loop) is zero . Expressed mathematically, KVL states that,

$$\sum_{m=1}^{M} v_m = 0$$

where M is the number of voltages in the loop (or the number of branches in the loop) and is the mth voltage.

- ✓ Nodal Analysis—

Nodal analysis provides a general procedure for analyzing circuits using node voltages as the circuit variables. Steps to Determine Node Voltages:

1. Select a node as the reference node. Assign voltages v_1 , v_2 ,..... v_{n-1} to the remaining (n-1) nodes. The voltages are referenced with respect to the reference node.

2. Apply KCL to each of the (n-1) nonreference nodes. Use Ohm's law to express the branch currents in terms of node voltages.

3. Solve the resulting simultaneous equations to obtain the unknown node voltages.

A supernode is formed by enclosing a (dependent or independent) voltage source connected between two nonreference nodes and any elements connected in parallel with it.

For the case of Supernode ,we had to deal with the code more technically.

- ✓ Thevenin's Theorem—

Thevenin's theorem states that a linear two-terminal circuit can be replaced by an equivalent circuit consisting of a voltage source V_{Th} in series with a resistor R_{Th} , where V_{Th} is the open-circuit voltage at the terminals and R_{Th} is the input or equivalent resistance at the terminals when the independent sources are turned off.

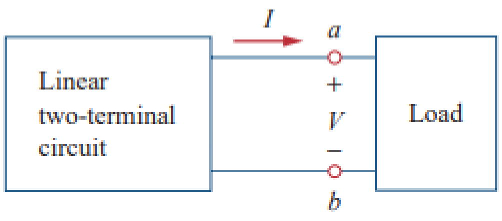


Fig: Original Circuit

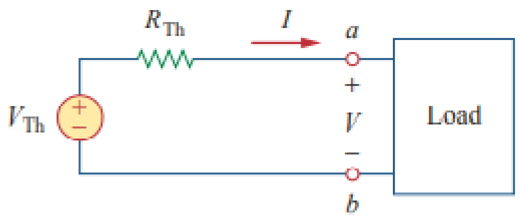


Fig : Thevenin Equivalent Circuit

- ✓ Norton's Theorem—

Norton's theorem states that a linear two-terminal circuit can be replaced by an equivalent circuit consisting of a current source I_N in parallel with a resistor R_N , where I_N is the short-circuit current through the terminals and R_N is the input or equivalent resistance at the terminals when the independent sources are turned off.

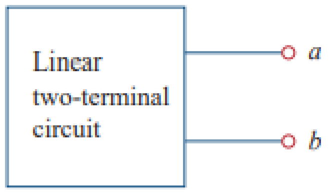


Fig : Original Circuit

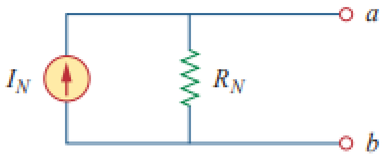


Fig : Norton Equivalent Circuit

- ✓ Linear System of Equations—

Let the equations be:

$a_1x+a_2y+a_3z=d_1$

$b_1x+b_2y+b_3z=d_2$

$c_1x+c_2y+c_3z=d_3$

Now we can write the above equations in the matrix form as follows

$$\begin{bmatrix} a_1x+a_2y+a_3z \\ b_1x+b_2y+b_3z \\ c_1x+c_2y+c_3z \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \Rightarrow \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \Rightarrow AX = B$$

… (i)

Where, A = $= \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}, X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, B = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}.$

Then X= A^{-1} B . In Matlab , there is built in function to calculate inverse matrix.

- ✓ Series Resonance—

Resonance is a condition in an RLC circuit in which the capacitive and inductive reactances are equal in magnitude, thereby resulting in a purely resistive impedance. The input impedance of an RLC circuit is—

$$\mathbf{Z} = R + j\left(\omega L - \frac{1}{\omega C}\right)$$

Resonance results when the imaginary part of this is zero. Then we will get—

$$\omega_0 = \frac{1}{\sqrt{LC}} \text{ rad/s}$$

This is resonant frequency . At this , inductance and capacitance will nullify each other.

The quality factor (Q) relates the maximum or peak energy stored to the energy dissipated in the circuit per cycle of oscillation.

$$Q = \frac{\omega_0 L}{R} = \frac{1}{\omega_0 CR}$$

Bandwidth (B) is the range (f1to f2) of frequencies for which the current is greater than 70.7% of the resonant value.

$$B = \frac{R}{L} = \frac{\omega_0}{Q}$$

Algorithms used in the project

Formulas/techniques used in the project:

We used nodal analysis, to be specific, modified nodal analysis, here. In nodal analysis the variables are the node voltages. But in modified nodal analysis we take the currents through the voltage sources (whether dependent or independent) as equation variables. That's why when we insert the voltage sources, we a row and a column in the coefficient matrix and a new row in the right-hand side column vector. To do these we have to change the values as mentioned in the theory part. Values to insert these values in the matrix is pictorially shown below.

1. **Resistance:**

SPICE Netlist Format (R)

R_k N_+ $N-$ $value\_of\_R_k$

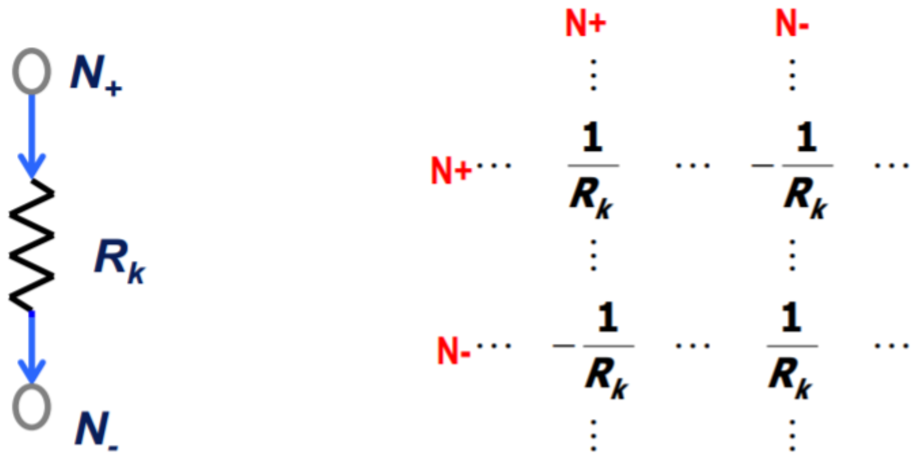


Fig: Values to be inserted for resistance

The picture shows that, for resistance only 4 values should be changed and only in the co-efficient matrix.

2. **Independent Voltage source:**

SPICE Netlist Format (Floating voltage source)

V_K N+ N- value_of_Vk

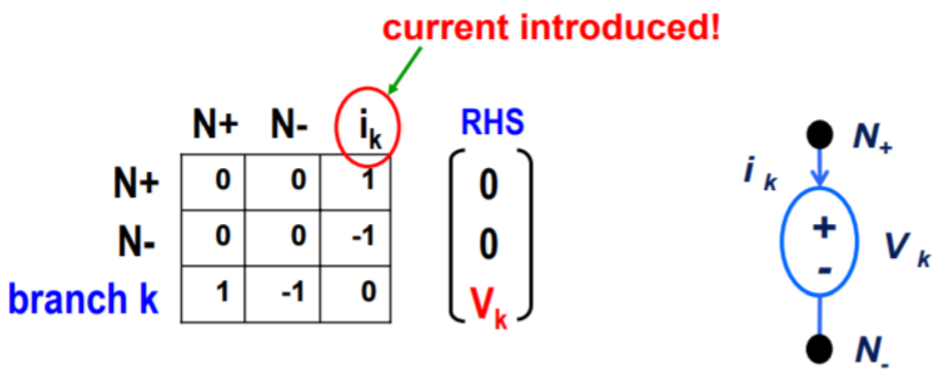
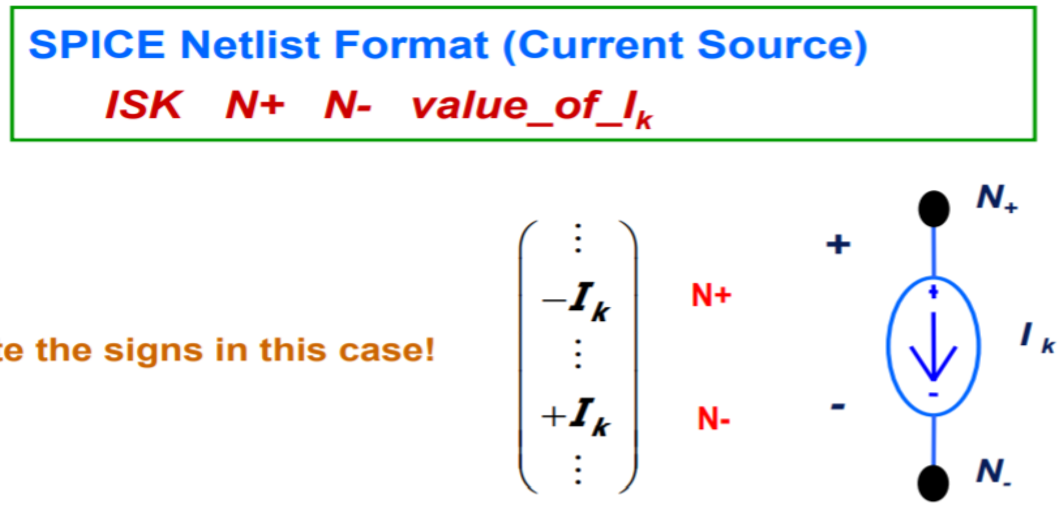


Fig: Values to be inserted for Voltage source

Here we can see that 'branch k' row and 'i_k' column has been added to the co-efficient matrix and V_k is added in right hand side vector. Here i_k is the current flowing from positive end to the negative end of voltage source.

3. **Independent Current Source:**



SPICE Netlist Format (Current Source)

ISK N+ N- value_of_I_k

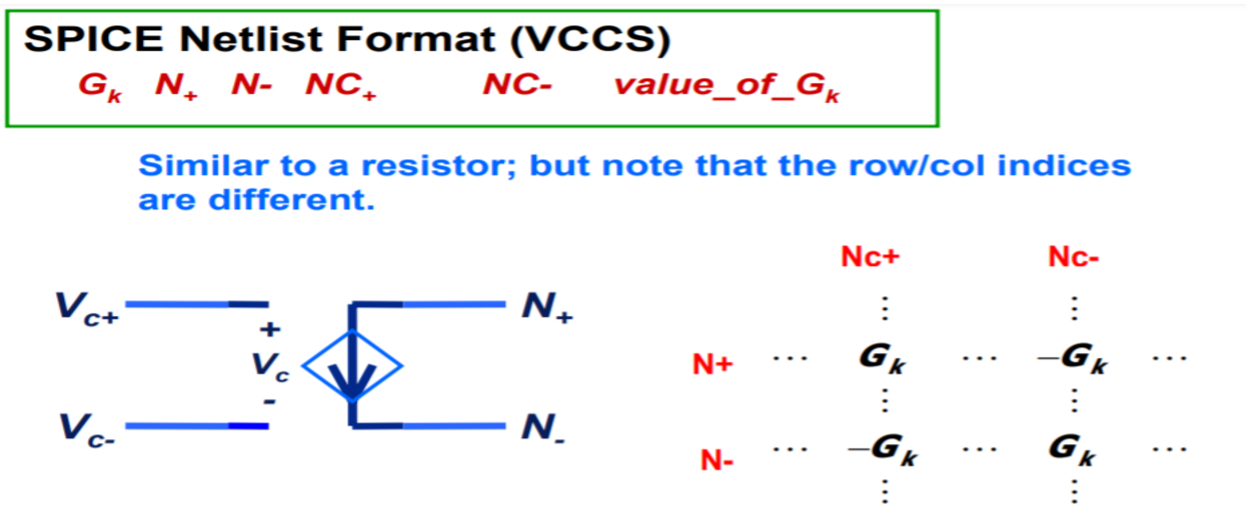
Note the signs in this case!

$$\begin{pmatrix}\vdots\\ -I_k\\ \vdots\\ +I_k\\ \vdots\end{pmatrix}\begin{array}{l} \\ N+\\ \\ N-\\ \\ \end{array}$$

Fig: Values to be inserted for Current source

For independent current source only right-hand side vector is modified as shown in the picture.

4. **VCCS**



SPICE Netlist Format (VCCS)

G_k N_+ N- NC_+ NC- value_of_G_k

Similar to a resistor; but note that the row/col indices are different.

$$\begin{array}{c|ccccccc} & & Nc+ & & Nc- & \\ & & \vdots & & \vdots & \\ N+ & \cdots & G_k & \cdots & -G_k & \cdots \\ & & \vdots & & \vdots & \\ N- & \cdots & -G_k & \cdots & G_k & \cdots \\ & & \vdots & & \vdots & \end{array}$$

Fig: Values to be inserted for VCCS

Among the 4 types of dependent sources VCCS has the simplest insertion technique. 4 values in the co-efficient matrix are modified only.

5. VCVS

SPICE Netlist Format (VCVS)

E_K N+ N- NC+ NC- value_of_E_K

$$
\begin{array}{c|c|c|c|c|c|}
 & N_+ & N- & NC_+ & NC- & i_k \\
\hline
N_+ & & & & & 1 \\
\hline
N- & & & & & -1 \\
\hline
NC_+ & & & & & \\
\hline
NC- & & & & & \\
\hline
br\ k & 1 & -1 & -E_k & E_k & \\
\hline
\end{array}
$$

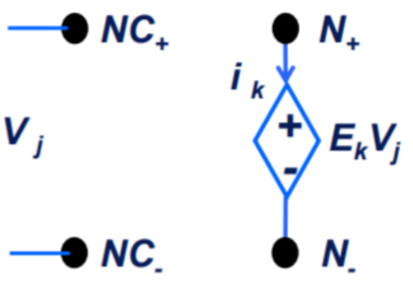


Fig: Values to be inserted for VCVS

As VCVS is a voltage source, a new variable (current through it) is added, which is named 'i_K', as we see in the picture.

6. CCCS

SPICE Netlist Format (CCCS)

F_K N+ N- Vname value_of_F_K

Vname NC+ NC- value

$$
\begin{array}{c|c|c|c|c|c|c}
 & N_+ & N- & NC_+ & NC- & i_c & RHS \\
\hline
N_+ & & & & & F_k & 0 \\
\hline
N- & & & & & -F_k & 0 \\
\hline
NC+ & & & & & 1 & 0 \\
\hline
NC- & & & & & -1 & 0 \\
\hline
br\ Vc & & & 1 & -1 & & V_c \\
\hline
\end{array}
$$

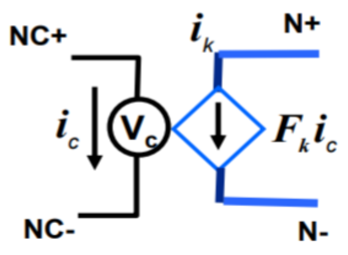
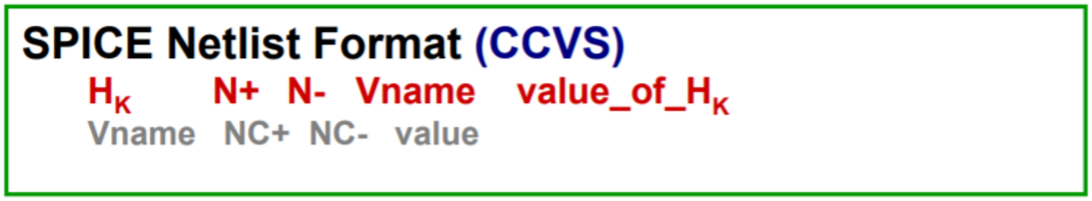


Fig: Values to be inserted for CCCS

Current controlled source insertion are a bit problematic. Here we can see that that the branch created for the controlling source are also needed.

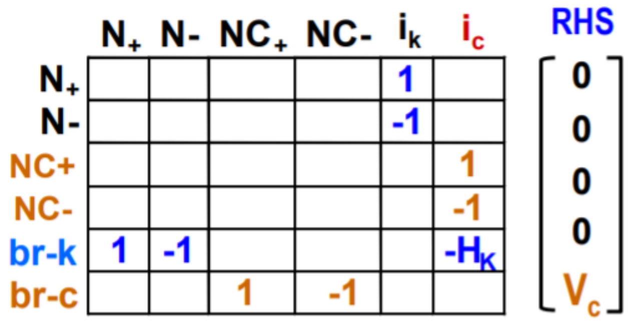
7. CCVS



SPICE Netlist Format (CCVS)

H_K N+ N- Vname value_of_H_K

Vname NC+ NC- value



	N_+	N-	NC_+	NC-	i_k	i_c	RHS
N_+					1		0
N-					-1		0
NC+						1	0
NC-						-1	0
br-k	1	-1				$-H_K$	0
br-c			1	-1			V_c

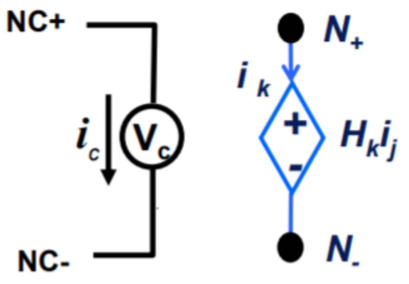


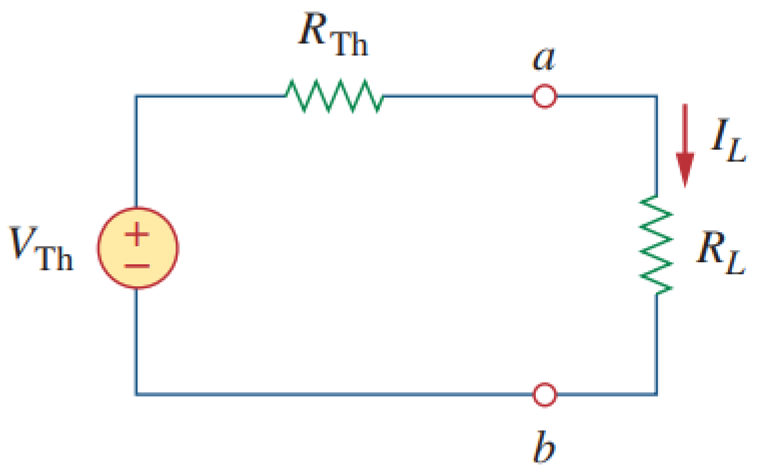
Fig: Values to be inserted for CCVS

Same problem arises here as for CCCS.

After taking input, we solve the system of linear equation (built in function available in MATLAB) and get the values of node voltages and branch currents.

8. Thevenin Voltage and Resistance—

Let’s assume thevenin equivalent for a complex circuit is given in the picture.



When R_L=R1, the voltage difference between a and b is

$$v1 = V_{TH} * \frac{R_L}{R_L+R_{TH}}$$

Or, $R1 * V_{TH} - V1 * R_{TH} = V1 * R1 \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (1)$

For R_L = R2, $R2 * V_{TH} - V2 * R_{TH} = V2 * R2 \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (2)$

Solving these 2 equations, we get V_{TH} and R_{TH}.

9. Series Resonance—

For series resonance we used the following formula.

$$|\mathbf{V}_L| = \frac{V_m}{R}\omega_0 L = QV_m$$

$$|\mathbf{V}_C| = \frac{V_m}{R}\frac{1}{\omega_0 C} = QV_m$$

And we calculated the other parameter using these formulae.

$$\omega_0 = \frac{1}{\sqrt{LC}}\ \mathrm{rad/s} \qquad Q = \frac{\omega_0 L}{R} = \frac{1}{\omega_0 CR} \qquad B = \frac{R}{L} = \frac{\omega_0}{Q}$$

And for the plot , we used simple ohm's law . Finding the current , we have put the voltages in a vector . And thus the plot was drawn.

Steps to execute the project

1. Run Home app
2. If you want to solve AC and DC circuit, go to step 10
3. If you want to analyze series RLC resonance hit the ‘Simple Series RLC Resonance’ button.
4. Input data.
5. Hit the ‘Show output’ button.
6. If you want to see plots hit ‘Show plot’ button.
7. If you want to analyze series RLC resonance, go to step 4.
8. If you want to solve AC and DC circuit, hit the ‘Home’ button and go to step 10.
9. If you want to terminate go to step 18.
10. Hit the ‘DC and AC circuit solver’ button.
11. Input node.
12. Input element values and hit the button ‘End’.
13. If you want to sweep DC source, fill the DC sweep panel and hit ‘Show plot’ button.
14. If you want to find thievenin resistance between two input nodes and hit the button show.
15. If you want to solve AC and DC circuit, go to step 10.
16. If you want to analyze series RLC resonance, hit the ‘Home’ button and go to step 5.
17. If you want to terminate go to step 18.
18. Quit program.

Codes

Working code of the circuitAnalyze app:

```
properties (Access = private)
        n=0;
        lhs=[];
        rhs=[];
        nodesofVDC=[];
        branch=[];
    end
```

```
% Callback function
function SaveAttributeButtonPushed(app, event)
    st=app.EnterElementNetlistCodeEditField.Value;
    st=strip(st);

    if ~isempty(st)
        [elementName, attributes, app.nodesofVDC,
app.branch]=handleInput(app.nodesofVDC, app.branch, st);

        [app.lhs, app.rhs]=callStampers(elementName, attributes,app.lhs, app.rhs);
    end
    app.EnterElementNetlistCodeEditField.Value='';
end


% Button pushed function: InputNodeButton
function InputNodeButtonPushed(app, event)
            app.n=app.EnterNumberofnodesSpinner.Value;
            [app.lhs, app.rhs, app.nodesofVDC, app.branch]=initiate(app.n);
            app.TextArea.Value='';
            app.InputNodeButton.Enable='off';
end

% Button pushed function: ENDButton
function ENDButtonPushed(app, event)
    val=app.TextArea_2.Value;
    [app.lhs, app.rhs, app.nodesofVDC, app.branch] = handleInput_0(app.nodesofVDC,
app.branch, app.lhs, app.rhs, val);
    st=solve_it(app.lhs, app.rhs, app.branch, app.n);
    disp(st);
    app.TextArea.Value=st;
end

% Button pushed function: ShowButton
function ShowButtonPushed(app, event)
    n1=app.NODE_1Spinner.Value;
    n2=app.NODE_2Spinner.Value;
    [vth, rth]=thievenin(app.lhs, app.rhs, n1, n2);
    app.TheveninVoltageVEditField.Value=vth;
    app.TheieveninOrNortonResistanceEditField.Value=rth;
    app.NortonCurrentIEditField.Value=vth/rth;

end

% Button pushed function: ResetButton
function ResetButtonPushed(app, event)
    app.NODE_1Spinner.Value=0;
    app.NODE_2Spinner.Value=0;
    app.TheveninVoltageVEditField.Value=0;
    app.TheieveninOrNortonResistanceEditField.Value=0;
    app.NortonCurrentIEditField.Value=0;
end

% Button pushed function: ShowPlotButton
```

```
function ShowPlotButtonPushed(app, event)
    n1=app.Node_1Spinner.Value;
    n2=app.Node_2Spinner.Value;
    vname=app.XaxisEditField.Value;
    range1=app.FromEditField.Value;
    range2=app.ToEditField.Value;
    [pltx, plty]=dcSweep(app.lhs, app.rhs, vname, n1, n2, range1, range2,
app.branch);
    plot(app.UIAxes,pltx,plty);
     xlabel(app.UIAxes, upper(vname));
    ylabel(app.UIAxes, "V("+num2str(n1)+":"+num2str(n2)+")")

end

% Button pushed function: HomeButton
function HomeButtonPushed(app, event)
    circuitAnalyze.app.callingapp=Home;
    app.delete;
end
```

Additional functions are:

initiate.m

```
% function to n=initiate all the values
function [lhs, rhs, nodesofVDC, branch]=initiate(n)
    % initiate left hand side matrix
    % also called co-efficient matrix
    lhs=zeros(n,n);

    % initiates right hand side vector
    rhs=zeros(n,1);

    % matrix to keep track of voltage sources
    nodesofVDC=zeros(n,3);

    % vector to keep track of variable names
    branch=["1"];

    % variable name assigning loop
    for i=1:n
        branch(i)=num2str(i);
    end
end
```

handleInput_0.m

```
% takes single line input from multiline input
% calls the function to call stampers
```

```
function [lhs, rhs, nodesofVDC, branch] = handleInput_0(nodesofVDC, branch,lhs,rhs,
val)
    n=length(val);
    for i=1:n
        st=char(val(i));
        [elementName, attributes, nodesofVDC, branch]=handleInput(nodesofVDC, branch,
st);
        [lhs, rhs]=callStampers(elementName, attributes, lhs, rhs);
    end
end
```

handleInput.m

```
% from singleline input extracts attributes of an element
function [elementName, attributes, nodesofVDC, branch]=handleInput(nodesofVDC,
branch, st)

    st=split(st);
    elementName=upper(char(st(1)));
    attributes=[];

    %for independent Voltage source

        if elementName(1)=='V'
            l=length(branch);
            branch(l+1)=string(elementName);        % (l+1)th variable is the current
though Voltage source
            nodesofVDC(l+1,1)=str2num(char(st(2)));
            nodesofVDC(l+1,2)=str2num(char(st(3)));
            nodesofVDC(l+1,3)=str2num(char(st(4)));
            attributes=[attributes str2num(char(st(2:length(st))))];
        end


    %For dependent voltage sources (CCVS and VCVS)
    if  elementName(1)=='E' || elementName(1)=='H'
        l=length(branch);
        branch(l+1)=string(elementName);        % (l+1)th variable is the current
though Voltage source
    end


    %for CCCS and CCVS
    if elementName(1)=='F' || elementName(1)=='H'
        val=str2num(char(st(5)));
        N1=str2num(char(st(2)));
        N2=str2num(char(st(3)));
        attributes=[attributes N1 N2];

        branchName = string(upper(char(st(4))));
        for i=1:length(branch)

            if branchName==branch(i)
                break;
            end
        end
        attributes=[attributes nodesofVDC(i,1) nodesofVDC(i,2) nodesofVDC(i,3) i val
];


    end

    % for current source, resistor, VCVS and VCCS
    if elementName(1)=='I' || elementName(1)=='R' || elementName(1)=='G'
||elementName(1)=='E'
        attributes=[attributes str2num(char(st(2:length(st))))];
    end


end
```

callStampers.m

```
function [lhs, rhs]=callStampers(elementName, attributes,lhs, rhs)

    % Determining the element type and call the respective stamping function
    if elementName(1)=="V"
        [lhs, rhs]=voltageStamp(attributes,lhs,rhs);
    elseif elementName(1)=="R"
        [lhs, rhs]=resistanceStamp(attributes,lhs,rhs);
    elseif elementName(1)=="I"
        [lhs, rhs]=currentStamp(attributes,lhs,rhs);
    elseif elementName(1)=="G"
        [lhs, rhs]=VCCSstamp(attributes,lhs,rhs);
    elseif elementName(1)=="E"
        [lhs, rhs]=VCVSstamp(attributes,lhs,rhs);
    elseif elementName(1)=="F"
        [lhs, rhs]=CCCSstamp(attributes,lhs,rhs);
    elseif elementName(1)=="H"
        [lhs, rhs]=CCVSstamp(attributes,lhs,rhs);
    end


end
```

voltageStamp.m

```
% function to stamp voltage source in the matrix
function [lhs, rhs]=voltageStamp(attributes, lhs, rhs)

    % N1 is positive node
    % N2 is negative node
    N1=attributes(1); N2=attributes(2); value=attributes(3);
    % n is the current length of matrixes
    n=size(rhs);
    n=n(1);

    % stamping in lhs
    if N1~=0
    lhs(n+1,N1)=1; lhs(N1,n+1)=1;
    end
    if N2~=0
    lhs(n+1,N2)=-1; lhs(N2,n+1)=-1;
    end

    % stamping in rhs
    rhs(n+1,1)=value;

end
```

currentStamp.m

```
% inserts current source attributes in matrix
function [lhs, rhs]=currentStamp(attributes, lhs, rhs)

    N1=attributes(1);   % positive node
    N2=attributes(2);   % negative node
    value=attributes(3);    % value of source

    if N1~=0
        rhs(N1,1)=rhs(N1,1)-value;
    end
    if N2~=0
        rhs(N2,1)=rhs(N2,1)+value;
    end
end
```

resistanceStamp.m

```
function [lhs, rhs]=resistanceStamp(attributes, lhs, rhs)
    N1=attributes(1); N2=attributes(2); value=attributes(3);
    if N1~=0
        lhs(N1,N1)=lhs(N1,N1)+1/value;
    end
    if N2~=0
        lhs(N2,N2)=lhs(N2,N2)+1/value;
    end

    if N1~=0 && N2~=0
        lhs(N1,N2)=lhs(N1,N2)-1/value;
        lhs(N2,N1)=lhs(N2,N1)-1/value;
end
```

CCCSstamp.m

```
function [lhs, rhs]=CCCSstamp(attributes, lhs, rhs)
```

```
    % naming the values of the element.

    N1=attributes(1);   % the positve node of CCCS
    N2=attributes(2);   % the negative node of CCCS
    NC1=attributes(3);  % posive node of controllinh current
    NC2=attributes(4);  % negative node of controlling current
    VC=attributes(5);   % value of controlling voltage
    ic=attributes(6);   % branch current number of controlling source
    val=attributes(7);  %value of controlled voltage

    % stamping values in the matrix

    % stamping in lhs
    if N1~=0
        lhs(N1,ic)=val;
    end
    if N2~=0
        lhs(N2,ic)=-val;
    end

    if NC1~=0
        lhs(NC1,ic)=+1;
        lhs(ic,NC1)=+1;
    end
    if NC2~=0
        lhs(NC2,ic)=-1;
        lhs(ic,NC2)=-1;
    end


    % stamping in rhs
    rhs(ic,1)=VC;

end
```

CCVSstamp.m

```
function [lhs, rhs]=CCVSstamp(attributes, lhs, rhs)

    % naming the values of the element.
    N1=attributes(1);   % the positive node of CCCS
    N2=attributes(2);   % the negative node of CCCS
    NC1=attributes(3);  % positive node of controlling current
    NC2=attributes(4);  % negative node of controlling current
    VC=attributes(5);
    ic=attributes(6);
    val=attributes(7);

    % ik is the current in CCVS branch. new varible.
    ik=length(rhs)+1;
    lhs(ik,ik)=0;
    rhs(ik,1)=0;

     % stamping values in the matrix
     %stamping in lhs
     if N1~=0
        lhs(N1,ik)=+1;
        lhs(ik,N1)=+1;
     end

     if N2~=0
        lhs(N2,ik)=-1;
        lhs(ik,N2)=-1;
     end

     if NC1~=0
        lhs(NC1,ic)=+1;
        lhs(ic,NC1)=+1;
     end

     if NC2~=0
        lhs(NC2,ic)=-1;
        lhs(ic,NC2)=-1;
     end

     lhs(ik,ic)=-val;



     %stamping in rhs
     rhs(ic,1)=rhs(ic,1)+VC;

end
```

VCCSstamp.m

```
function [lhs, rhs,usedNodes]=VCCSstamp(attributes, lhs, rhs,usedNodes)
    N1=attributes(1);
    N2=attributes(2);
    NC1=attributes(3);
    NC2=attributes(4);
    val=attributes(5);

    %stamping values in matrix

    %stamping in lhs
    if N1~=0 && NC1~=0
        lhs(N1,NC1)=lhs(N1,NC1)+val;
    end
    if N1~=0 && NC2~=0
        lhs(N1,NC2)=lhs(N1,NC2)-val;
    end
    if N2~=0 && NC1~=0
        lhs(N2,NC1)=lhs(N2,NC1)-val;
    end
    if N2~=0 && NC2~=0
        lhs(N2,NC2)=lhs(N2,NC2)+val;
    end
```

```
end
```

VCVSstamp.m

```
function [lhs, rhs]=VCVSstamp(attributes, lhs, rhs)
    N1=attributes(1);
    N2=attributes(2);
    NC1=attributes(3);
    NC2=attributes(4);
    val=attributes(5);

    %
    %n=length(rhs)+1;
    n=length(rhs);
    lhs(n+1,n+1)=0;
    rhs(n+1,1)=0;

    % stamping values in the matrix
    if N1~=0
        lhs(N1,n+1)=lhs(N1,n+1)+1;
        lhs(n+1,N1)=lhs(n+1,N1)+1;
    end
    if N2~=0
        lhs(N2,n+1)=lhs(N2,n+1)-1;
        lhs(n+1,N2)=lhs(n+1,N2)-1;
    end

    if NC1~=0
        lhs(n+1,NC1)=lhs(n+1,NC1)-val;
    end
    if NC2~=0
        lhs(n+1,NC2)=lhs(n+1,NC2)+val;
    end
end
```

solve_it.m

```
function ST= solve_it(lhs, rhs, branch, n)
    res=lhs\rhs;
    fprintf('\n');
    ST={};
    for i=1:length(res)
        if i>n
            st= "I(" + branch(i) + "+:" + branch(i) + "-)=    " + num2str(res(i)) + "
A";
        else
            st = "V(" + branch(i) + ")=    " + num2str(res(i)) + " V";
        end
        ST{i,1}=char(st);
    end

end
```

thevenin.m

```
function [vth, rth]=thevenin(lhs, rhs, n1, n2)
    r1=1;
    attributes=[n1 n2 r1];
    [Tlhs, Trhs]=resistanceStamp(attributes, lhs, rhs);
    Tres=Tlhs\Trhs;
    v1=0;
    if n1
        v1= v1+Tres(n1);
    end

    if n2
        v1=v1-Tres(n2);
    end

    r2=2;
    attributes=[n1 n2 r2];
    [Tlhs, Trhs]=resistanceStamp(attributes, lhs, rhs);
    Tres=Tlhs\Trhs;
    v2=0;
    if n1
        v2= v2+Tres(n1);
    end

    if n2
        v2=v2-Tres(n2);
    end

    res=[r1 -v1;r2 -v2]\[v1*r1; v2*r2];
    vth=res(1);
    rth=res(2);




end
```

dcSweep.m

```
function [pltx, plty]=dcSweep(lhs, rhs, vname, n1, n2, range1, range2, branch)

    vname=upper(string(vname)); % name of the source to sweep

    for i=1:length(branch)      %seek which branch current corresponds to the source
        if vname==branch(i)
            br=i;
            break;
        end
    end

    pltx=linspace(range1, range2, 1000);    % x axis variable
    plty=pltx;  %dummy y axis variable
    for i=1:1000
        Trhs=rhs;
        Trhs(br)=pltx(i);
        res=lhs\Trhs;
        v=0;
        if n1
            v= v+res(n1);
        end

        if n2
            v=v-res(n2);
        end
        plty(i)=v;
    end


end
```

Working code of the app_seriesRLC_res app:

```
% Button pushed function: ShowOutputButton
function ShowOutputButtonPushed(app, event)
    r=app.InputRohmEditField.Value;
    l=app.InputLHEditField.Value;
    c=app.InputCFEditField.Value;
    amp=app.AmplitudeVEditField.Value;
    del=app.InitialPhaseDegreeEditField.Value;
    resw=1/sqrt(l*c);
    resf=resw/(2*pi);
    app.EditField.Value=resf;
    BW=r/l;
    QF=resw/BW;
    app.EditField_2.Value=BW;
    app.EditField2.Value=QF;
end

% Button pushed function: ShowPlotButton
function ShowPlotButtonPushed(app, event)
  r=app.InputRohmEditField.Value;
    l=app.InputLHEditField.Value;
    c=app.InputCFEditField.Value;
    amp=app.AmplitudeVEditField.Value;
    del=app.InitialPhaseDegreeEditField.Value;
    resw=1/sqrt(l*c);
    resf=resw/(2*pi);
    f=0:1:(2*resf);
    j=sqrt(-1);
    i=(amp*(cos(del)+j*sin(del)))./(r+(2*pi.*f).*l+1./((2*pi.*f).*c));
    Vr=i.*r;
    Vl=i.*(2*pi.*f).*l;
    Vc=i.*1./((2*pi.*f).*c);
    plot(app.UIAxes,f,abs(Vr),f,abs(Vl),f,abs(Vc));
    legend(app.UIAxes,'VR','VL','VC');
end

% Button pushed function: ClearDataButton
function ClearDataButtonPushed(app, event)
    app.InputRohmEditField.Value=0;
    app.InputLHEditField.Value=0;
    app.InputCFEditField.Value=0;
    app.AmplitudeVEditField.Value=0;
    app.InitialPhaseDegreeEditField.Value=0;
    app.EditField.Value=0;
    app.EditField_2.Value=0;
    app.EditField2.Value=0;
    f=0:.1:1;
    Vr = ones(11);
    plot(app.UIAxes,f,Vr,"Color",[0.80,0.80,0.80]);
    legend(app.UIAxes,'hide');
end

% Button pushed function: HomeButton
function HomeButtonPushed(app, event)
    circuitAnalyze.app.callingapp=Home;
    app.delete;
end
```

Test Cases

Case for DC & AC Circuit Solver --

- ✓ Test case 1:(Only with independent sources)

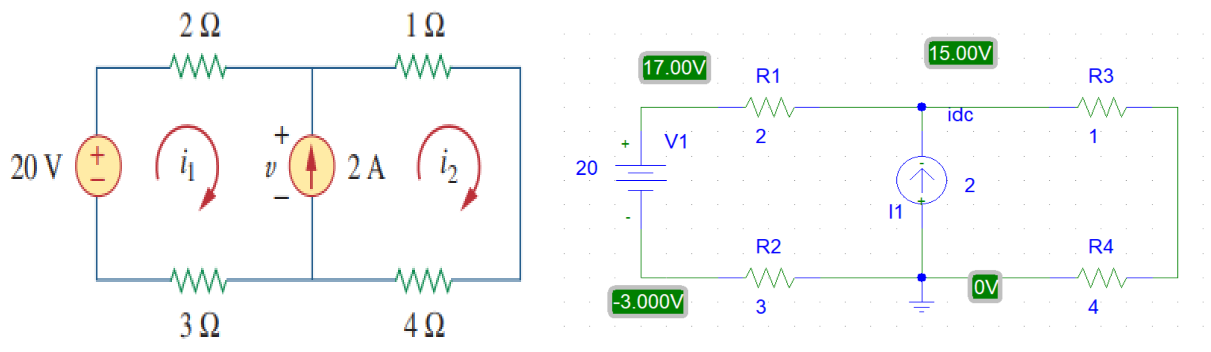
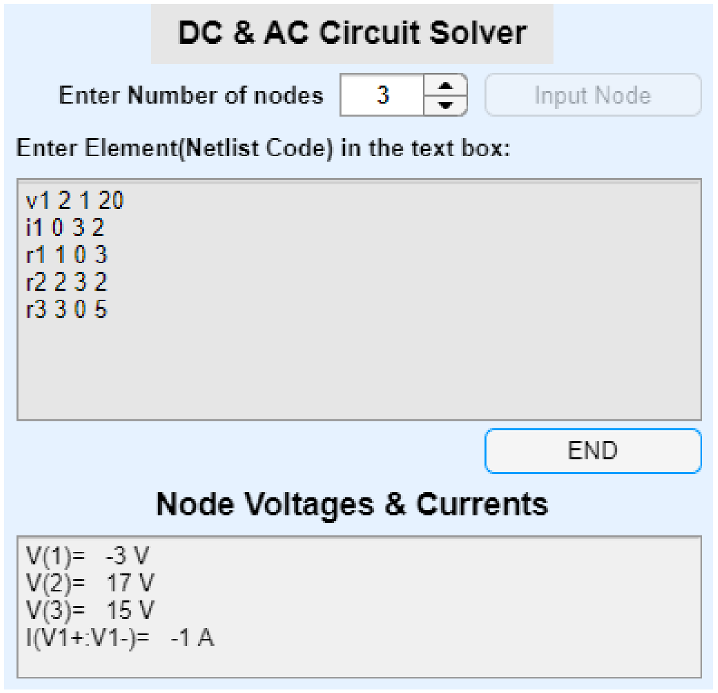


Fig: Circuit and PS piece simulation.


DC & AC Circuit Solver

Enter Number of nodes 3 Input Node

Enter Element(Netlist Code) in the text box:

```
v1 2 1 20
i1 0 3 2
r1 1 0 3
r2 2 3 2
r3 3 0 5
```

END

Node Voltages & Currents

```
V(1)=  -3 V
V(2)=  17 V
V(3)=  15 V
I(V1+:V1-)=  -1 A
```

Fig: Simulation result in the circuitAnalyze app

✓ <u>Test case 2: (with CCVS)</u>

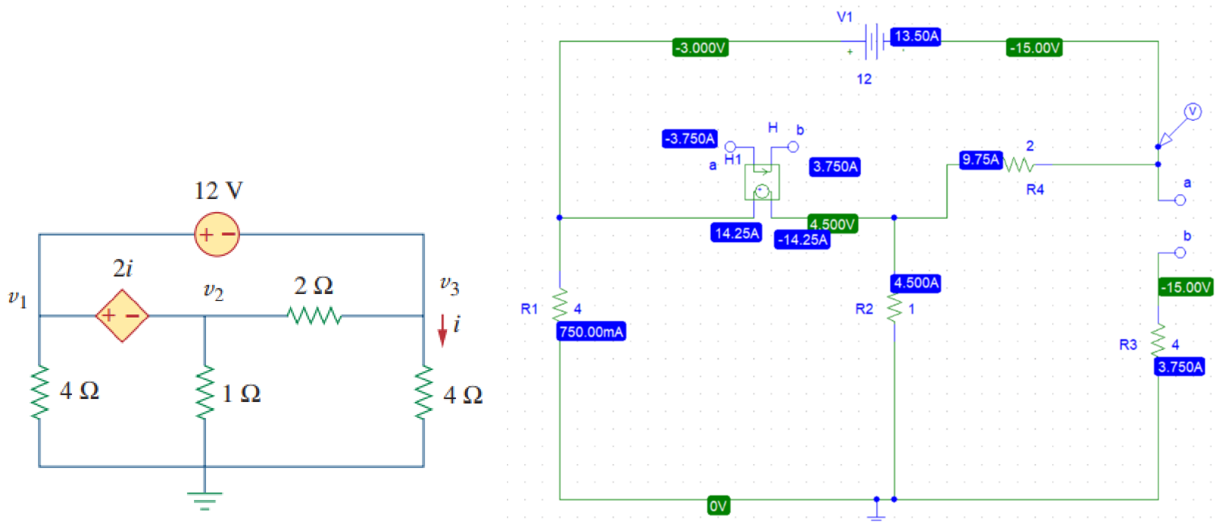
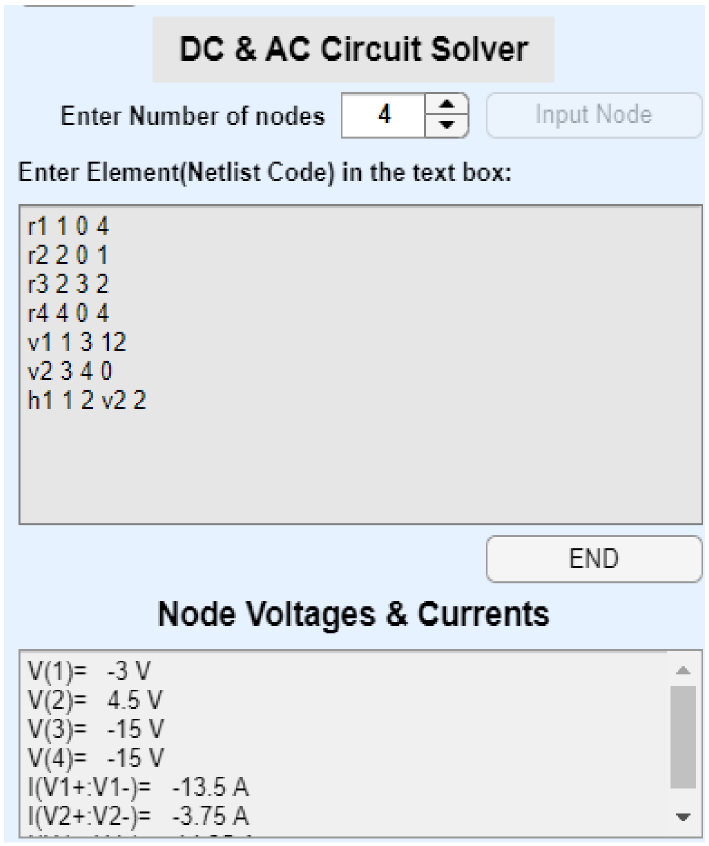


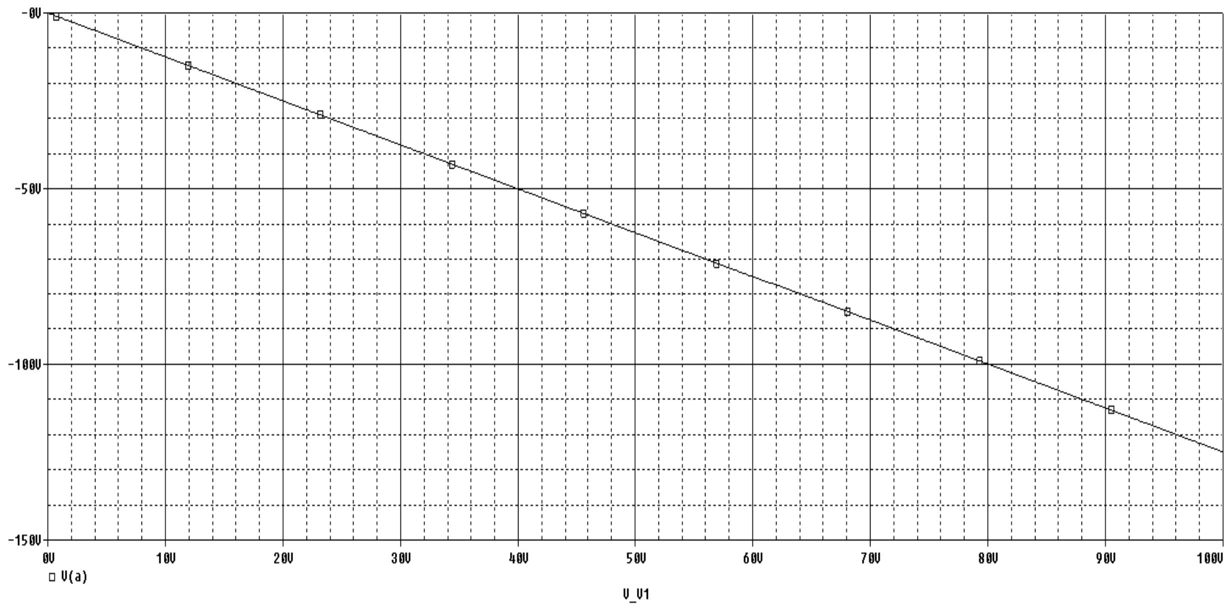
Fig: Circuit and PS piece simulation.

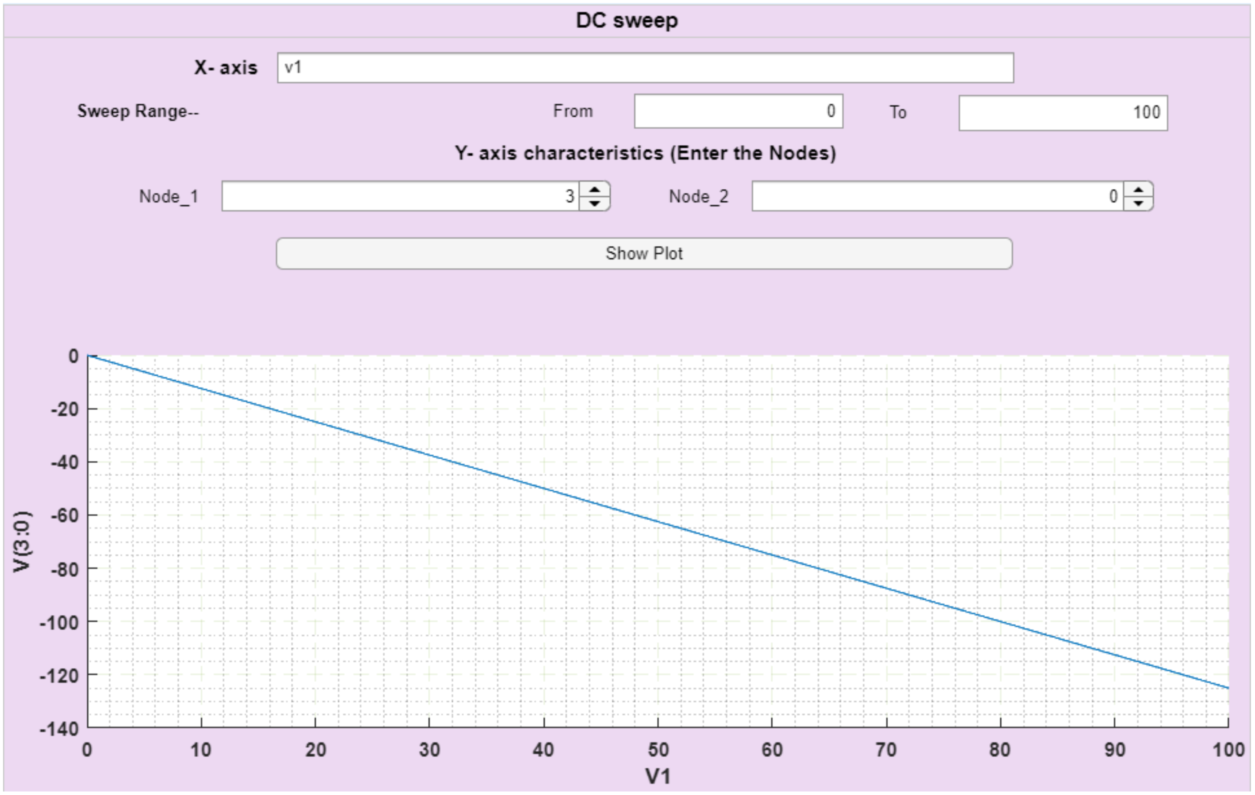


```
r1 1 0 4
r2 2 0 1
r3 2 3 2
r4 4 0 4
v1 1 3 12
v2 3 4 0
h1 1 2 v2 2
```

```
V(1)=   -3 V
V(2)=   4.5 V
V(3)=   -15 V
V(4)=   -15 V
I(V1+:V1-)=   -13.5 A
I(V2+:V2-)=   -3.75 A
```

Fig: Simulation result in the circuitAnalyze app




DC sweep

X- axis v1

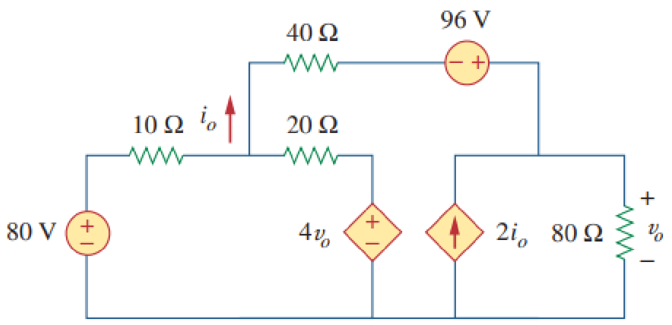
Sweep Range-- From 0 To 100

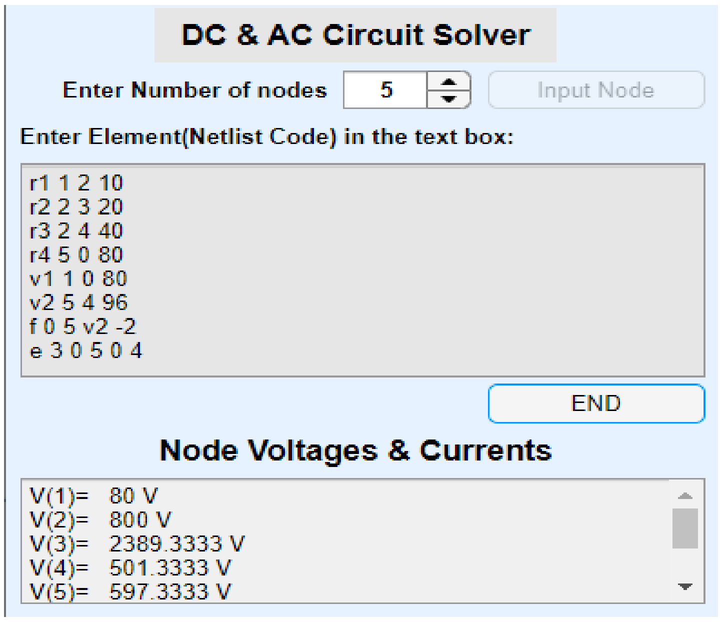
Y- axis characteristics (Enter the Nodes)

Node_1 3 Node_2 0

Fig: DC sweep in PS piece and ciruitAnalyze app.

✓ <u>Test case 3: (with VCVS and CCCS)</u>





```
r1 1 2 10
r2 2 3 20
r3 2 4 40
r4 5 0 80
v1 1 0 80
v2 5 4 96
f 0 5 v2 -2
e 3 0 5 0 4
```

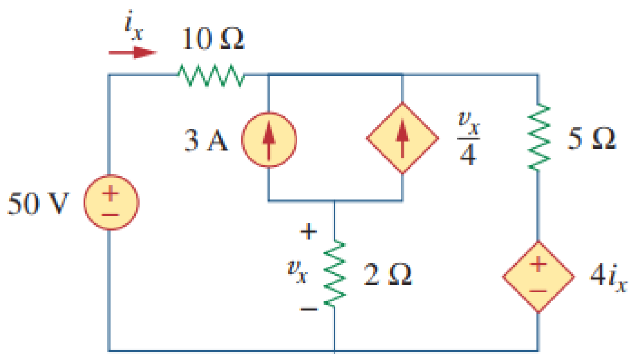
```
V(1)=  80 V
V(2)=  800 V
V(3)=  2389.3333 V
V(4)=  501.3333 V
V(5)=  597.3333 V
```

Fig: circuit and the result after simulation in circuitAnalyze app.

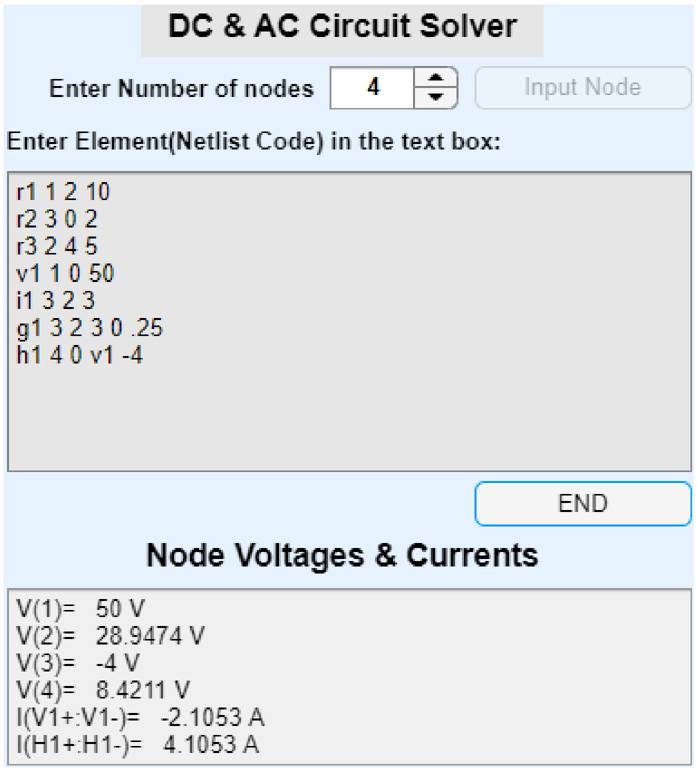
The full output was:

```
V(1)=   80 V
V(2)=   800 V
V(3)=   2389.3333 V
V(4)=   501.3333 V
V(5)=   597.3333 V
I(V1+:V1-)=   72 A
I(V2+:V2-)=   -7.4667 A
I(E+:E-)=   -79.4667 A
```

- ✓ Test case 4: (with VCCS and CCVS)



The value of Vx and ix is $v_x = 2(i_1 - i_2) = $ **−4 volts** and $i_x = i_2 - 2 = $ **2.105 amp**



```
r1 1 2 10
r2 3 0 2
r3 2 4 5
v1 1 0 50
i1 3 2 3
g1 3 2 3 0 .25
h1 4 0 v1 -4
```

```
V(1)=   50 V
V(2)=   28.9474 V
V(3)=   -4 V
V(4)=   8.4211 V
I(V1+:V1-)=   -2.1053 A
I(H1+:H1-)=   4.1053 A
```

Fig: Output in circuitAnlayze app

✓ <u>Test case 5: (finding Thevenin equivalents)</u>

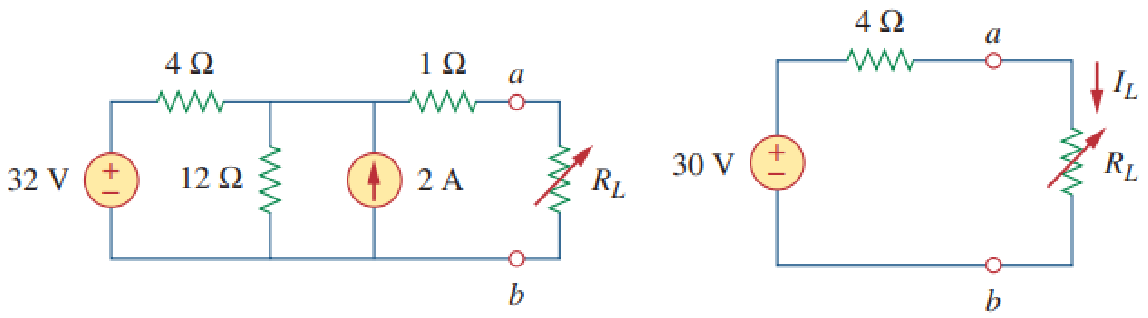
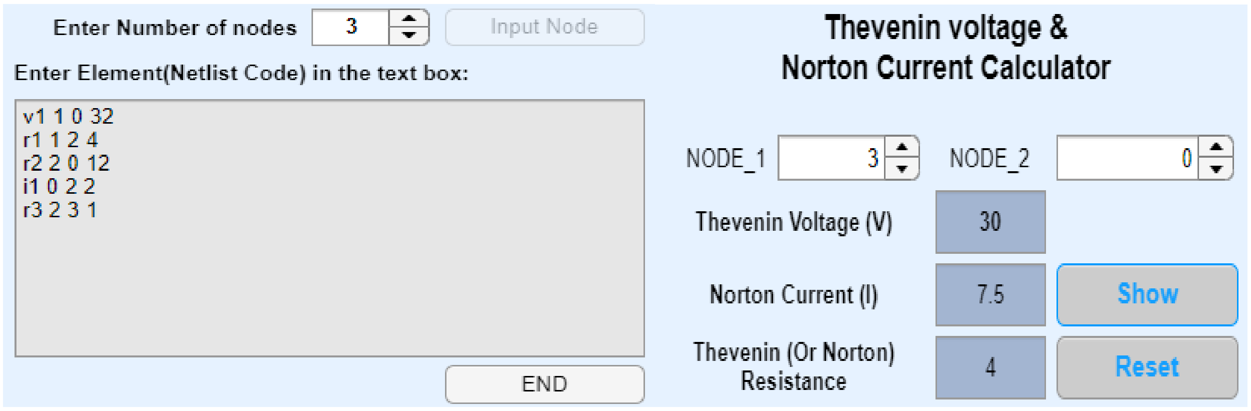


Fig: circuit and the thievenin equivalent



Enter Number of nodes 3

Input Node

Enter Element(Netlist Code) in the text box:

```
v1 1 0 32
r1 1 2 4
r2 2 0 12
i1 0 2 2
r3 2 3 1
```

END

Thevenin voltage &
Norton Current Calculator

NODE_1 3 NODE_2 0

Thevenin Voltage (V) 30

Norton Current (I) 7.5

Thevenin (Or Norton) Resistance 4

Fig: Thevenin voltage and resistance value measuring.

Test case 6: (finding Thevenin equivalents with dependent source)

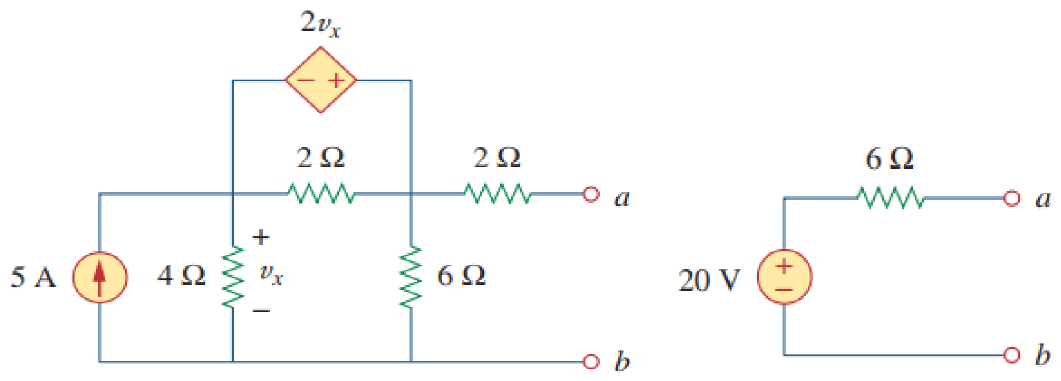
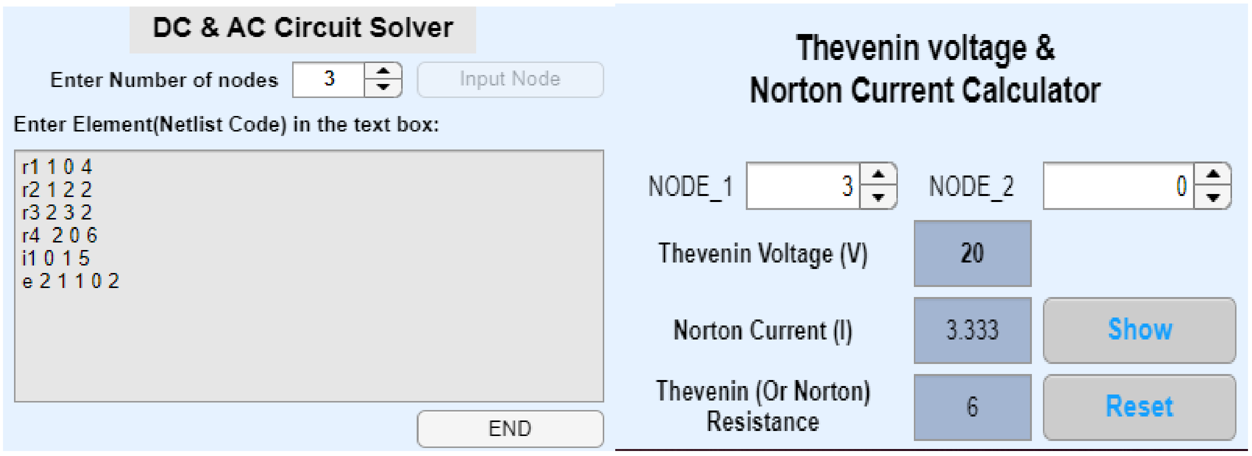


Fig: circuit and the thevenin equivalent



DC & AC Circuit Solver

Enter Number of nodes 3

Input Node

Enter Element(Netlist Code) in the text box:

```
r1 1 0 4
r2 1 2 2
r3 2 3 2
r4  2 0 6
i1 0 1 5
e 2 1 1 0 2
```

END

Thevenin voltage &
Norton Current Calculator

NODE_1 3 NODE_2 0

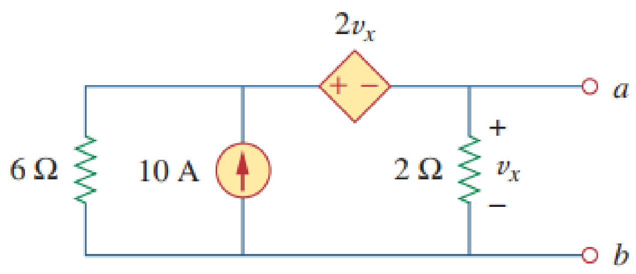
Thevenin Voltage (V) 20

Norton Current (I) 3.333

Thevenin (Or Norton) Resistance 6

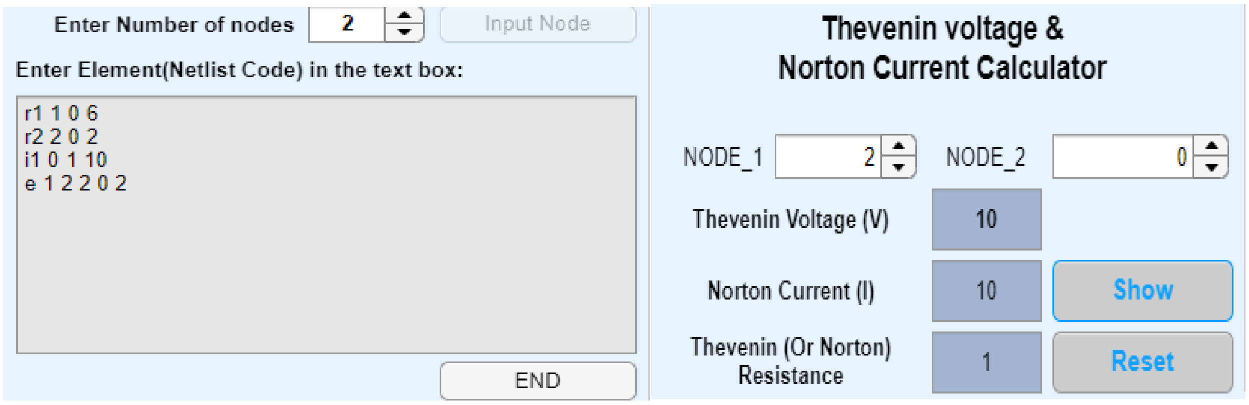
Fig: Thevenin voltage and resistance value measuring.

✓ Test case 7: (finding Norton equivalents)



Answer: $R_N = 1\ \Omega$, $I_N = 10$ A.

Fig: circuit and Norton equivalent values


Enter Number of nodes
Input Node
Enter Element(Netlist Code) in the text box:
END
Thevenin voltage &
Norton Current Calculator
Show
Reset

Enter Number of nodes: 2

```
r1 1 0 6
r2 2 0 2
i1 0 1 10
e 1 2 2 0 2
```

NODE_1: 2   NODE_2: 0

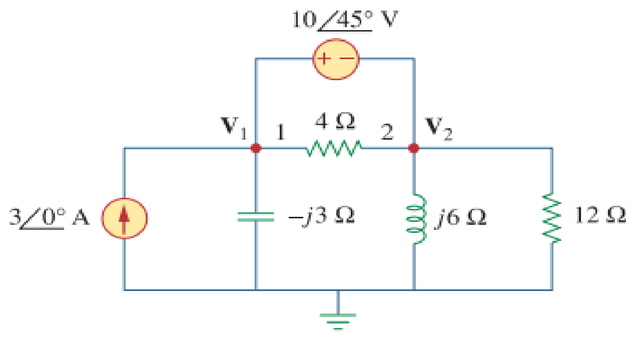
Thevenin Voltage (V): 10

Norton Current (I): 10

Thevenin (Or Norton) Resistance: 1

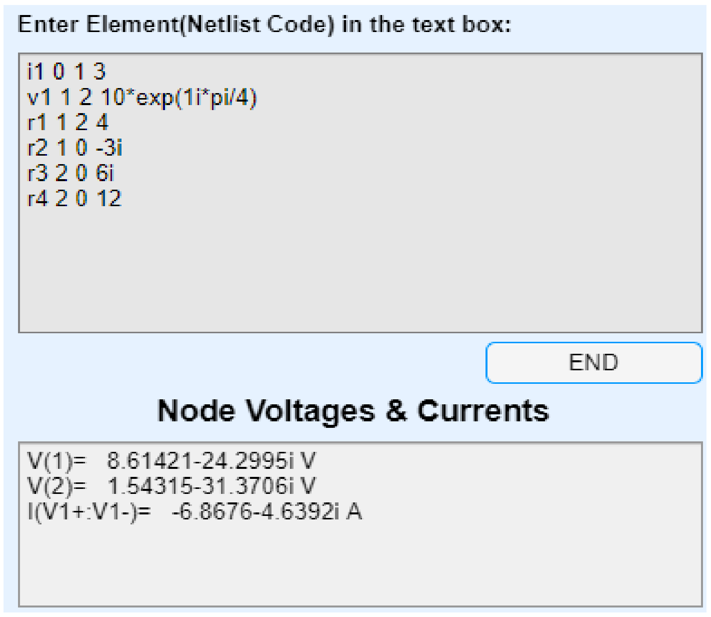
Output in the app.

✓ Test case 8: solving AC circuit



$$\mathbf{V}_1 = \mathbf{V}_2 + 10\angle 45^\circ = 25.78\angle -70.48^\circ \text{ V} \qquad \mathbf{V}_2 = 31.41\angle -87.18^\circ \text{ V}$$

Fig: AC circuit and the node voltages.



Enter Element(Netlist Code) in the text box:

```
i1 0 1 3
v1 1 2 10*exp(1i*pi/4)
r1 1 2 4
r2 1 0 -3i
r3 2 0 6i
r4 2 0 12
```

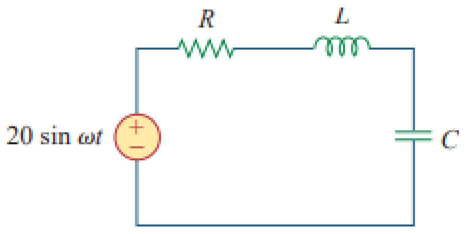
Node Voltages & Currents

```
V(1)=   8.61421-24.2995i V
V(2)=   1.54315-31.3706i V
I(V1+:V1-)=   -6.8676-4.6392i A
```

Fig: Output in the app.

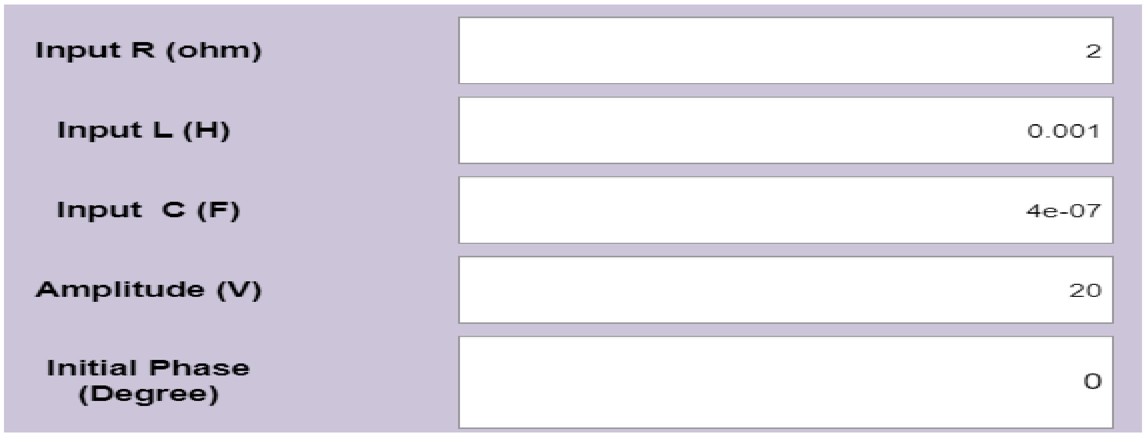
Case for Series RLC circuit—

✓ Case 1 -- In this circuit , R=2 Ω ,L = 1mH , C=0.4μF.



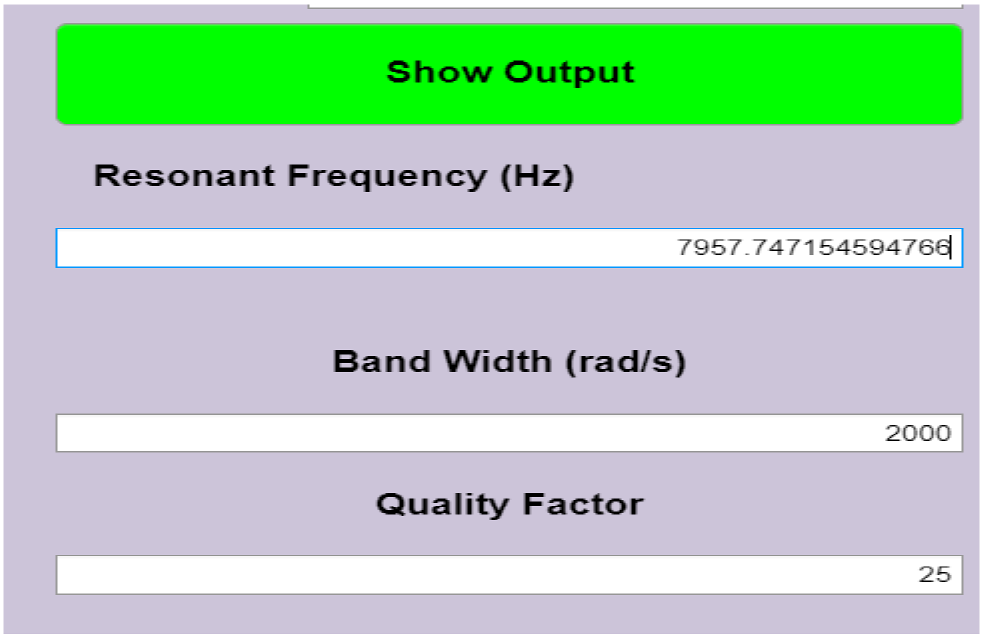
Calculate resonant frequency , Band Width and Quality Factor.

Input—



Input R (ohm)	2
Input L (H)	0.001
Input C (F)	4e-07
Amplitude (V)	20
Initial Phase (Degree)	0

Output—



Show Output

Resonant Frequency (Hz)

7957.747154594766

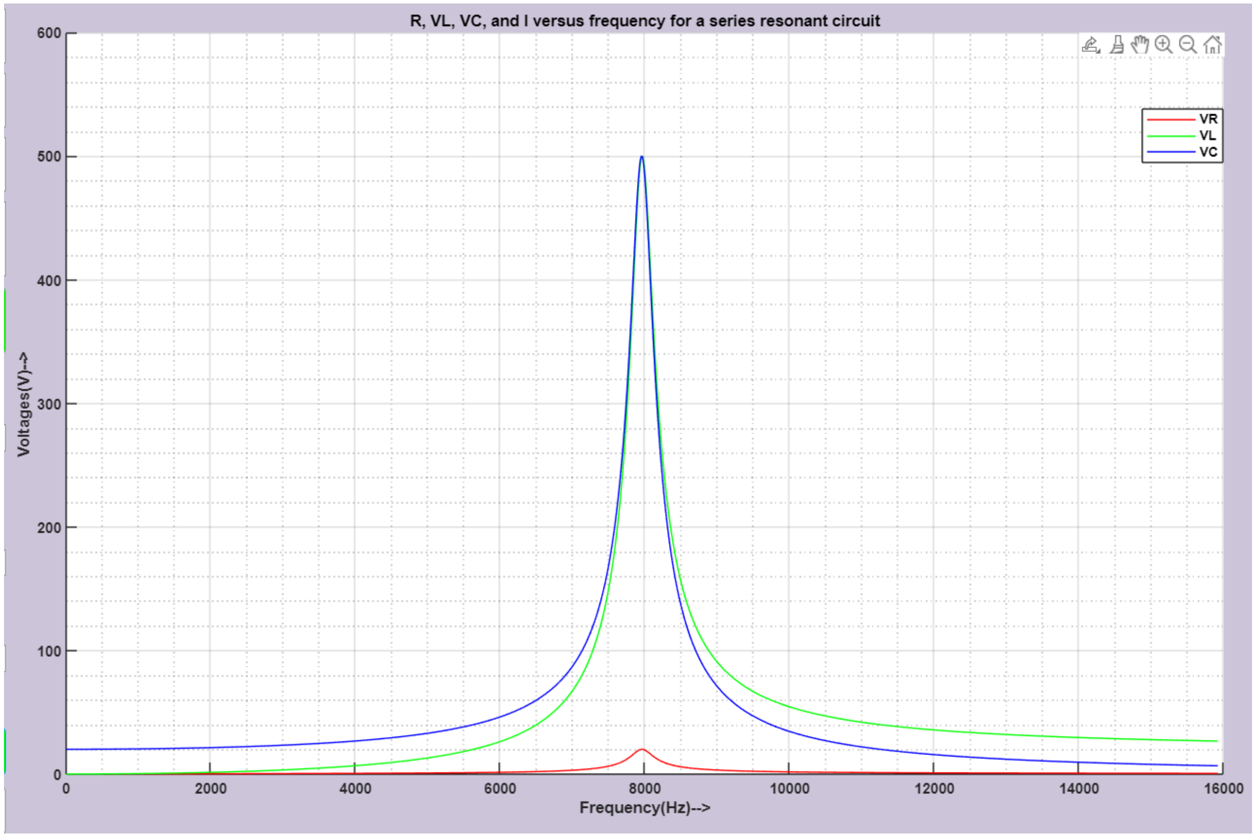
Band Width (rad/s)

2000

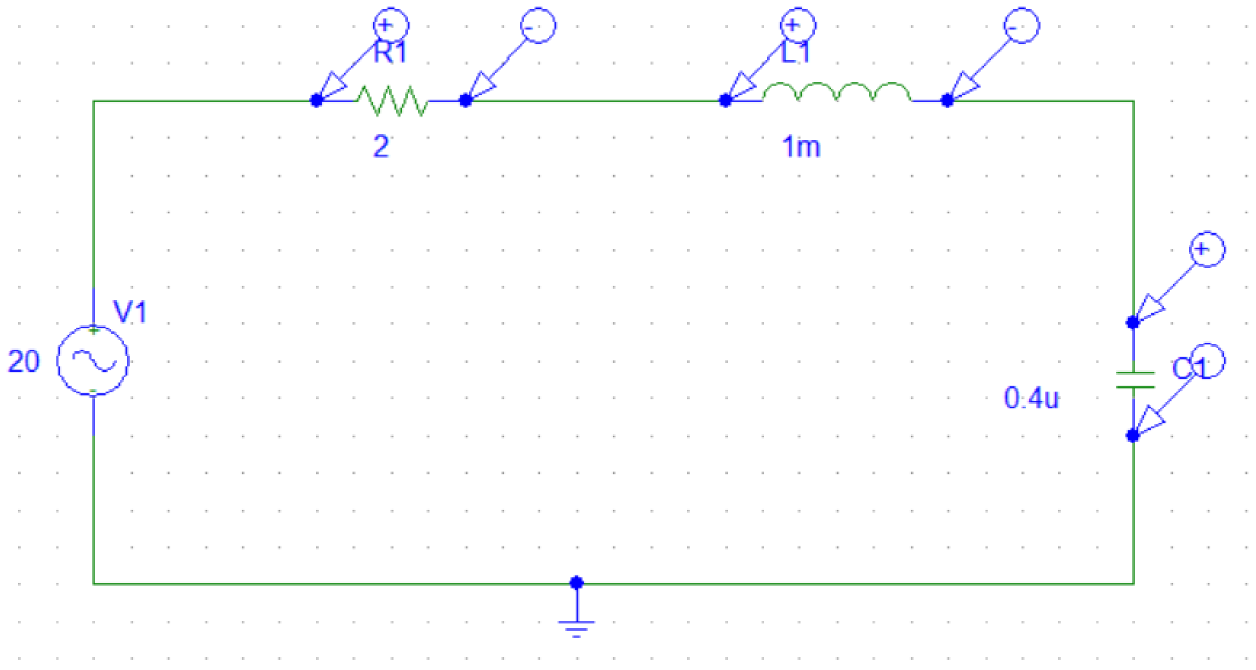
Quality Factor

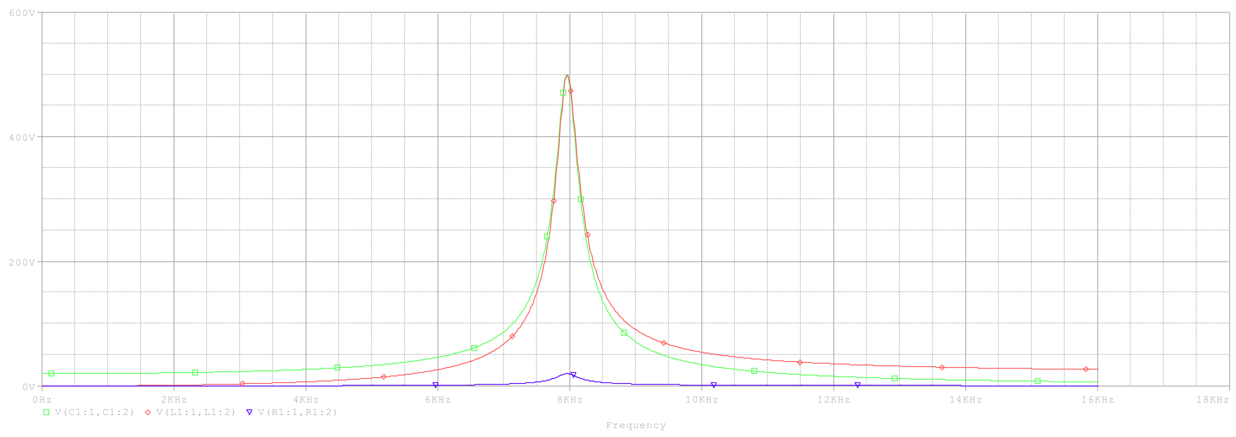
25

Graph—



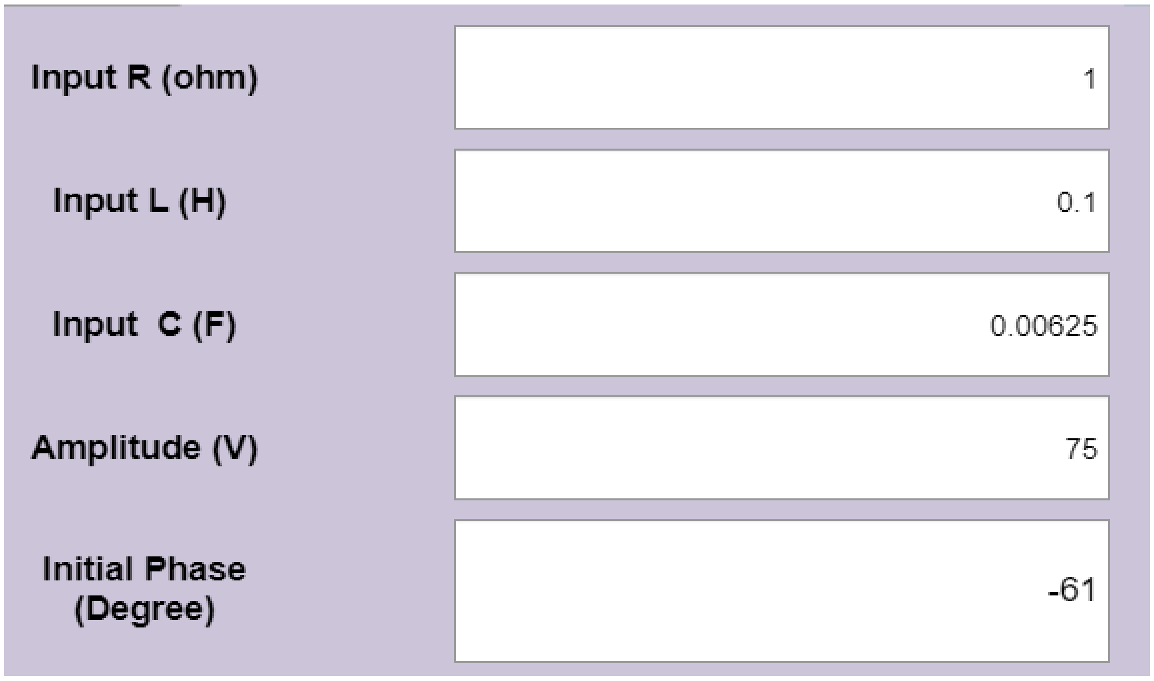
Verification with Schematic & Plot—





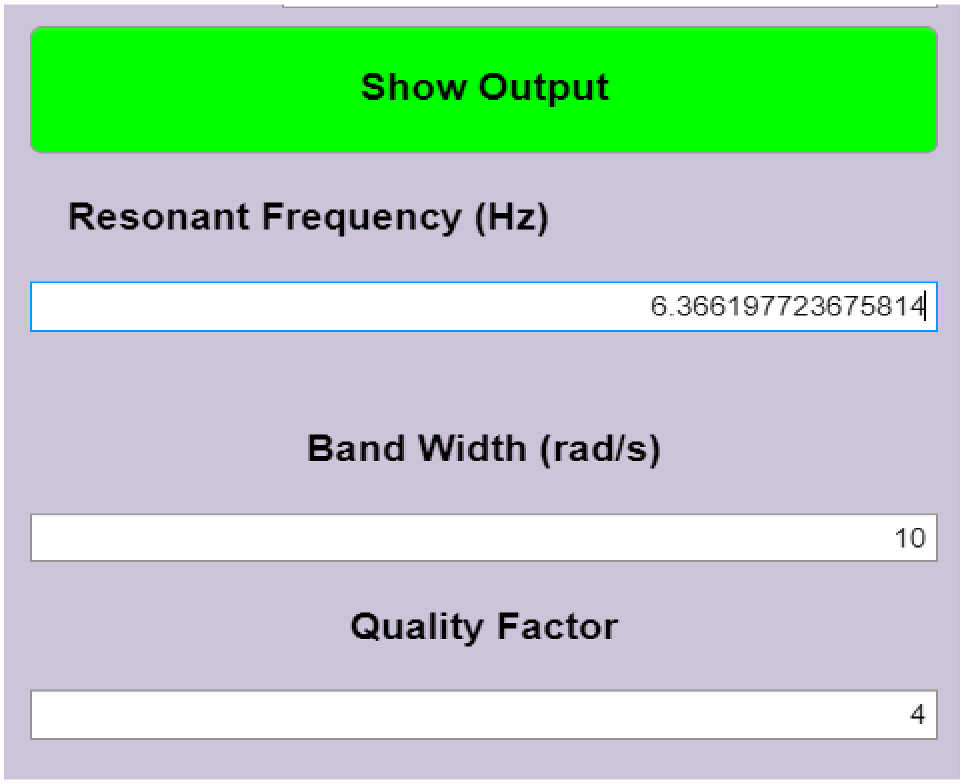
- ✓ <u>Case 2 --- Given that , R=1 Ω ,L = 100 mH , C= 6.25 mF. Also E= 75cos(wt-61).</u>
  <u>Calculate resonant frequency , Band Width and Quality Factor</u>.

Input—



Input R (ohm)	1
Input L (H)	0.1
Input C (F)	0.00625
Amplitude (V)	75
Initial Phase (Degree)	-61

Output—



Resonant Frequency (Hz)

6.366197723675814

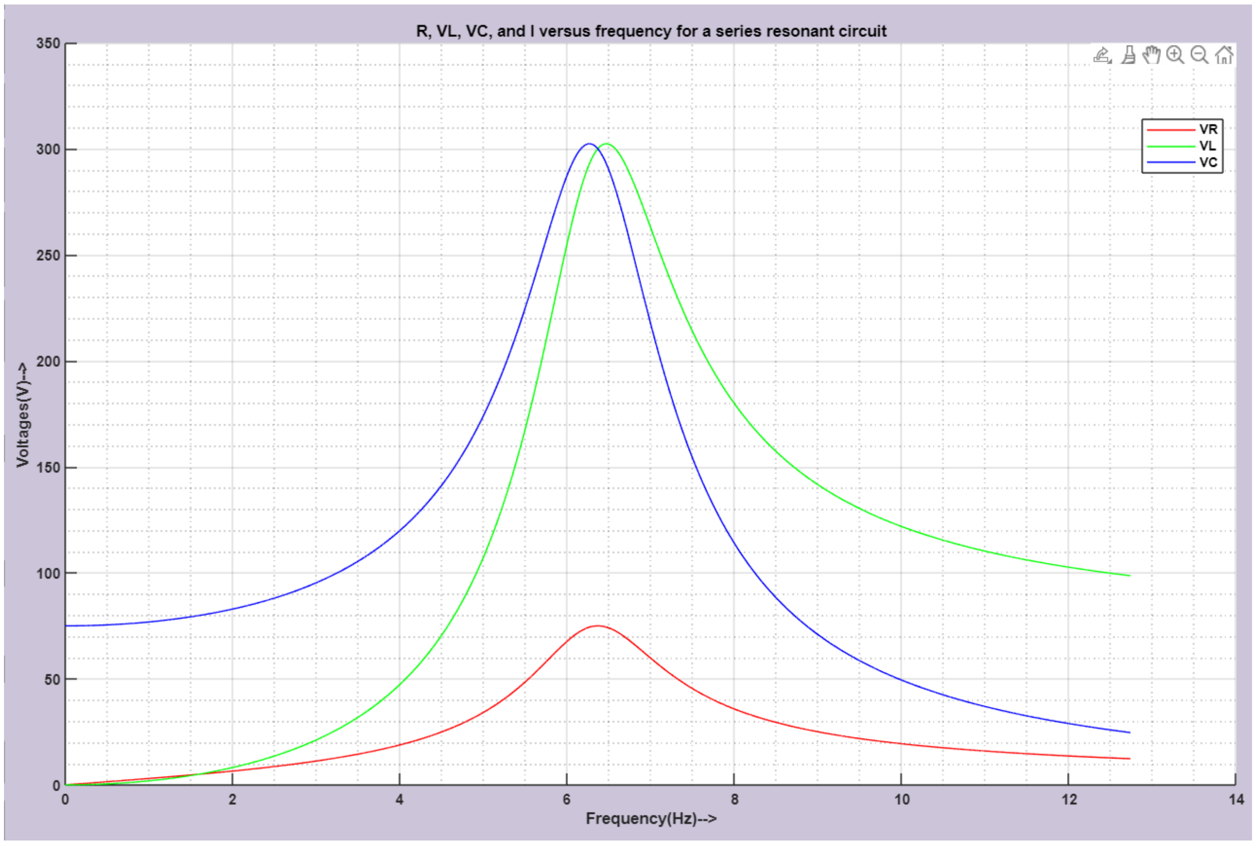
Band Width (rad/s)

10

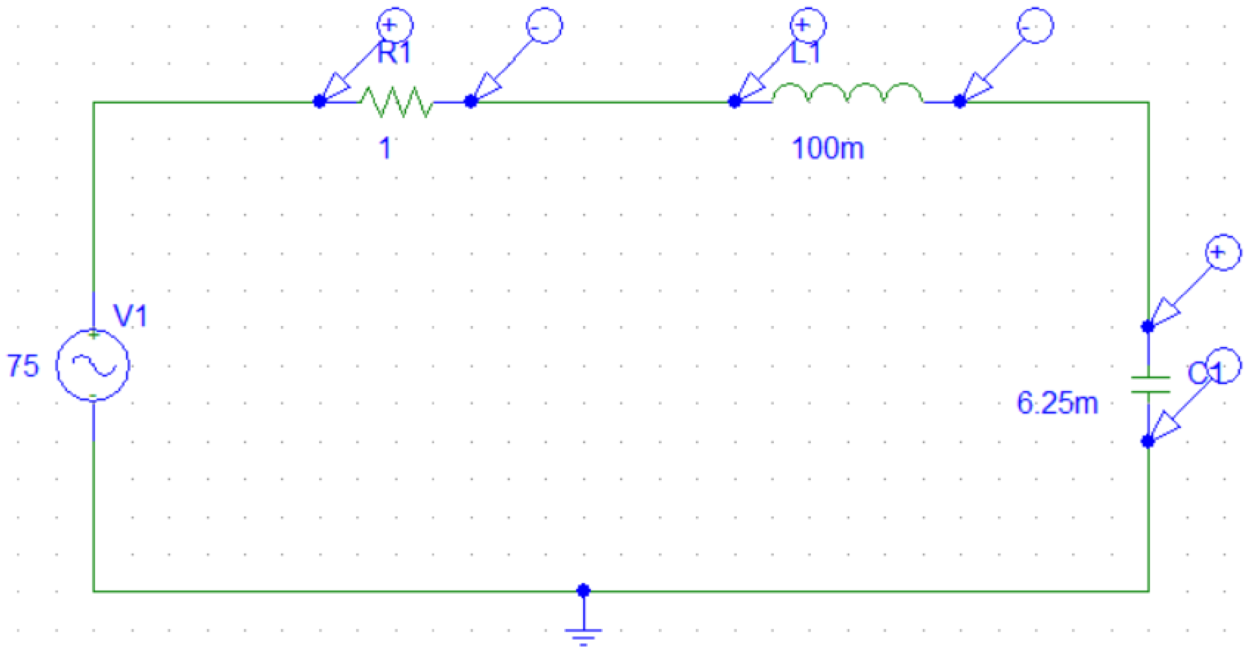
Quality Factor

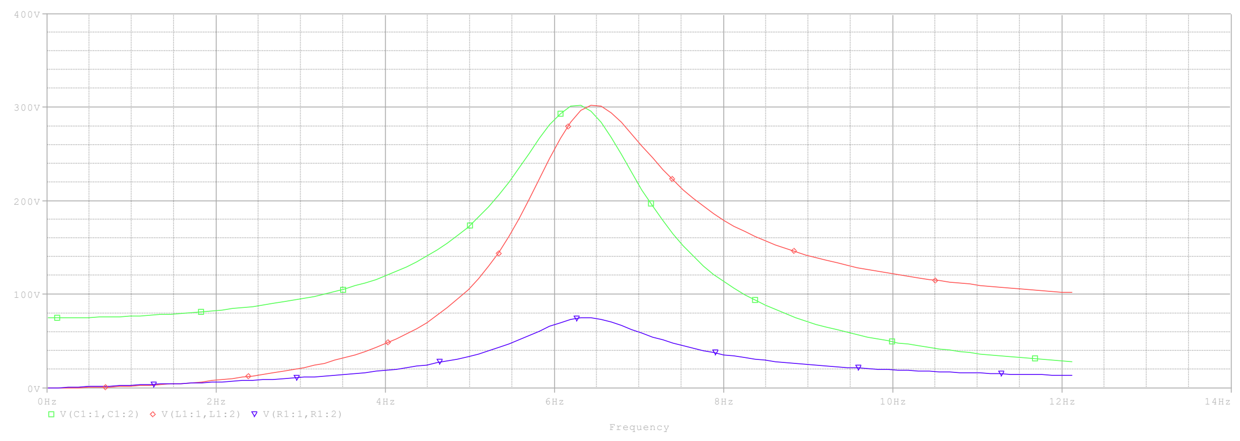
4

Graph—



Verification with Schematic & Plot—





<u>Application</u>

Some practical applications can be as follows—

1) If we make a circuit practically to understand the insight ,it takes time and it will cost also. But by using such kind of circuit solver ,one can easily calculate any circuit with proper understanding.
2) Also to implement big project , there are lots of issues . This type of software can save time to a great extent.
3) In communication systems ,there are lots of use of filter. And from the resonance part , one can get the idea about resonance characteristics of a circuit.

<u>Conclusion</u>

Every thing has limitations of it's own. In any experiment , there will be shortcoming. But we have to be conscious about this.

<u>Shortcoming –</u>

- This project can not deal with transient analysis.

- ➢ Unable to access inductance or capacitance directly given and time-varying resistance.
- ➢ Also sinusoidal components must be given by Phasor .
- ➢ Also built in function like dB(), P(), RMS() which we see in simulation software like Pspice are not included here.
- ➢ Also in RLC resonance ,it can not deal with complex circuit excluding series connection.
- ➢ Also unable to determine filter characteristics.

Shortcoming make the project ineffective. So always after project ,it is a good habit to build up and update the codes so that the project can be more appropriate and useful.

<u>Future Prospect of Improvement—</u>

➢ The GUI can be made more interactive.

➢ Plotting can be done with more details.

➢ Code can be lessen by using more built in function .

➢Transient analysis, filter ,3 phase , this type of operation can be included.

➢ Complexity, interaction and run time can be reduced by a further bugging.

<u>THE END</u>