Project:

Objective:

Solving an AC or DC circuit

- Input is the value of each element (R, L, C voltage source etc.)
- Circuit may contain both the independent and dependent current or voltage source.
- Show the schematic of the given circuit
- Also show the current, voltage in each node and branch respectively, power dissipation or supply for each element
- Make a suitable GUI for this project.
- Show result and analysis for at least 10 test cases

GUI Code:

```
function varargout = proj_gui(varargin)
% PROJ_GUI MATLAB code for proj_gui.fig
%      PROJ_GUI, by itself, creates a new PROJ_GUI or raises the existing
%      singleton*.
%
%      H = PROJ_GUI returns the handle to a new PROJ_GUI or the handle to
%      the existing singleton*.
%
%      PROJ_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PROJ_GUI.M with the given input arguments.
%
%      PROJ_GUI('Property','Value',...) creates a new PROJ_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before proj_gui_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to proj_gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help proj_gui

% Last Modified by GUIDE v2.5 21-Jul-2021 11:38:33
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @proj_gui_OpeningFcn, ...
                   'gui_OutputFcn',  @proj_gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before proj_gui is made visible.
function proj_gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to proj_gui (see VARARGIN)

% Choose default command line output for proj_gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes proj_gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = proj_gui_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;
```

```
function net_input_Callback(hObject, eventdata, handles)
% hObject    handle to net_input (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 net_input as text
%        str2double(get(hObject,'String')) returns contents of net_input as a double
x=get(hObject,"String");
Circuit_image=0;
if x(1,1)=="c"

   Circuit_Image=x(1,:);

   x=x(2:end,:);
end

[node_voltage_arra,current_branch_arra,power_arra]=project_main(x);
set(handles.node_voltage,"String",node_voltage_arra);
set(handles.current_branch,"String",current_branch_arra);
set(handles.power_arr,"String",power_arra);

I=imread(Circuit_Image);
imshow(Circuit_Image);




% --- Executes during object creation, after setting all properties.
function net_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to net_input (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)
```

```
function node_voltage_Callback(hObject, eventdata, handles)
% hObject    handle to node_voltage (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 node_voltage as text
%        str2double(get(hObject,'String')) returns contents of node_voltage as a double


% --- Executes during object creation, after setting all properties.
function node_voltage_CreateFcn(hObject, eventdata, handles)
% hObject    handle to node_voltage (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



function current_branch_Callback(hObject, eventdata, handles)
% hObject    handle to current_branch (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 current_branch as text
%        str2double(get(hObject,'String')) returns contents of current_branch as a double


% --- Executes during object creation, after setting all properties.
function current_branch_CreateFcn(hObject, eventdata, handles)
% hObject    handle to current_branch (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



function power_arr_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to power_arr (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 power_arr as text
%        str2double(get(hObject,'String')) returns contents of power_arr as a double


% --- Executes during object creation, after setting all properties.
function power_arr_CreateFcn(hObject, eventdata, handles)
% hObject    handle to power_arr (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
```

Function code(project_main.m)

```
function [a,b,c]=project_main(x)

%% Creating a datastructure from the netlist

DC=0;
V_source_array=[];
I_source_array=[];
R_array=[];
C_array=[];
L_array=[];
VCVS_array=[];
VCCS_array=[];
CCCS_array=[];
CCVS_array=[];
branch_current_array=[];
node_voltage_array=[];
power_array=[];
number_of_branches=0;
number_of_nodes=0;
node_zero=0;

while 1
    name=x(number_of_branches+1,:);
```

```
name=strtrim(name);
if name=="END"
   break;
end
number_of_branches=number_of_branches+1;
if name(1,1)=='V'

   name=split(name);
   name=string(name');
   if name(1,4)=="DC"
      DC=1;
   else
      freq=str2double(name(1,7));
   end
   V_source_array=[V_source_array; name];

elseif name(1,1) =='I'
   name=split(name);
   name=string(name');
   if name(1,4)=="DC"
      DC=1;
   else
      freq=str2double(name(1,7));
   end
   I_source_array=[I_source_array; name];

elseif name(1,1)=='R'
   name=split(name);
   name=string(name');
   R_array=[R_array; name];

elseif name(1,1)=='C'
   name=split(name);
   name=string(name');
   C_array=[C_array; name];

elseif name(1,1)=='L'
   name=split(name);
   name=string(name');
   L_array=[L_array; name];

elseif name(1,1)=='E'
   name=split(name);
   name=string(name');
   VCVS_array=[VCVS_array; name];

elseif name(1,1)=='F'
   name=split(name);
```

```
        name=string(name');
        CCCS_array=[CCCS_array; name];

    elseif name(1,1)=='G'
        name=split(name);
        name=string(name');
        VCCS_array=[VCCS_array; name];

    elseif name(1,1)=='H'
        name=split(name);
        name=string(name');
        CCVS_array=[CCVS_array; name];
    end

    if(number_of_nodes<str2double(name(1,2)))
        number_of_nodes=str2double(name(1,2));
    end
    if(number_of_nodes<str2double(name(1,3)))
        number_of_nodes=str2double(name(1,3));
    end
end

branch_current_array=zeros(number_of_branches,1);
power_array=zeros(number_of_branches,1);



%% Finding the sizes of arrays


size_V_source_array=size(V_source_array);
size_I_source_array=size(I_source_array);
size_VCCS_array=size(VCCS_array);
size_VCVS_array=size(VCVS_array);
size_CCVS_array=size(CCVS_array);
size_CCCS_array=size(CCCS_array);
size_L_array=size(L_array);
size_C_array=size(C_array);
size_R_array=size(R_array);
total_length=size_V_source_array(1,1)+size_VCVS_array(1,1)+size_CCVS_array(1,1);

%% to keep track of voltageSources
if (size_V_source_array)
    tracker=[];
end
%% (A B; C D)*X= S
S=zeros(number_of_nodes+total_length,1);
A=zeros(number_of_nodes, number_of_nodes);
```

```
B=zeros(number_of_nodes,total_length);
C=zeros(total_length,number_of_nodes);
D=zeros(total_length,total_length);


%% Stamping(for modified nodal analysis)

%for resistors
for i=1:size_R_array(1,1)
   pos_node=str2double(R_array(i,2));
   neg_node=str2double(R_array(i,3));
   res_mag=str2double(R_array(i,4));
   if(pos_node==0)
      A(neg_node,neg_node)=A(neg_node,neg_node)+(1/res_mag);
   elseif(neg_node==0)
      A(pos_node,pos_node)=A(pos_node,pos_node)+(1/res_mag);
   else
      A(pos_node,pos_node)=A(pos_node,pos_node)+(1/res_mag);
      A(neg_node,neg_node)=A(neg_node,neg_node)+(1/res_mag);
      A(pos_node,neg_node)=A(pos_node,neg_node)-(1/res_mag);
      A(neg_node,pos_node)=A(neg_node,pos_node)-(1/res_mag);
   end
end


%for VCCS
size_VCCS_array=size(VCCS_array);
for i=1:size_VCCS_array(1,1)
   pos_node=str2double(VCCS_array(i,2));
   neg_node=str2double(VCCS_array(i,3));
   pos_source_node=str2double(VCCS_array(i,4));
   neg_source_node=str2double(VCCS_array(i,5));
   amp_G=str2double(VCCS_array(i,6));
   if(neg_node~=0)
      if(pos_source_node==0)
         A(neg_node,neg_source_node)=A(neg_node,neg_source_node)+amp_G;
      elseif(neg_source_node==0)
         A(neg_node,pos_source_node)=A(neg_node,pos_source_node)-amp_G;
      else
         A(neg_node,neg_source_node)=A(neg_node,neg_source_node)+amp_G;
         A(neg_node,pos_source_node)=A(neg_node,pos_source_node)-amp_G;
      end
   end
   if(pos_node~=0)
      if(pos_source_node==0)
         A(pos_node,neg_source_node)=A(pos_node,neg_source_node)-amp_G;
      elseif(neg_source_node==0)
         A(pos_node,pos_source_node)=A(pos_node,pos_source_node)+amp_G;
```

```
        else
            A(pos_node,neg_source_node)=A(pos_node,neg_source_node)-amp_G;
            A(pos_node,pos_source_node)=A(pos_node,pos_source_node)+amp_G;
        end

    end
end


%for current source

size_I_source_array=size(I_source_array);
for i=1:size_I_source_array
    pos_node=str2double(I_source_array(i,2));
    neg_node=str2double(I_source_array(i,3));
    if(DC)
        cur_mag=str2double(I_source_array(i,5));
    else
        mag=str2double(I_source_array(i,5));

cur_mag=mag*cos(str2double(I_source_array(i,6)))+mag*sin(str2double(I_source_array(i,6))
)*1j;
    end
    if(pos_node==0)
        S(neg_node,1)=S(neg_node,1)+cur_mag;
    elseif(neg_node==0)
        S(pos_node,1)=S(pos_node,1)-cur_mag;
    else
        S(neg_node,1)=S(neg_node,1)+cur_mag;
        S(pos_node,1)=S(pos_node,1)-cur_mag;
    end
end

B_new=1;
C_new=1;

%for Vsource
for i=1:size_V_source_array(1,1)
    pos_node=str2double(V_source_array(i,2));
    neg_node=str2double(V_source_array(i,3));
    if(DC)
        volt_mag=str2double(V_source_array(i,5));
    else
        mag=str2double(V_source_array(i,5));

volt_mag=mag*cos(str2double(V_source_array(i,6)))+mag*sin(str2double(V_source_array(i,
6)))*1j;
    end
```

```
   if(pos_node==0)
      B(neg_node,i)=B(neg_node,i)-1;
      C(i,neg_node)=C(i,neg_node)-1;
   elseif(neg_node==0)
      B(pos_node,i)=B(pos_node,i)+1;
      C(i,pos_node)=C(i,pos_node)+1;
   else
      B(neg_node,i)=B(neg_node,i)-1;
      C(i,neg_node)=C(i,neg_node)-1;
      B(pos_node,i)=B(pos_node,i)+1;
      C(i,pos_node)=C(i,pos_node)+1;
   end
   S(i+number_of_nodes,1)=volt_mag;
   tracker=[tracker V_source_array(i,1)];
end

B_new=B_new+size_V_source_array(1,1);
C_new=C_new+size_V_source_array(1,1);

%for CCCS

for i=1:size_CCCS_array(1,1)

   for j=1:size_V_source_array(1,1)
      if tracker(1,j)==CCCS_array(i,4)
         break;
      end
   end
   pos_node=str2double(CCCS_array(i,2));
   neg_node=str2double(CCCS_array(i,3));
   amp_F=str2double(CCCS_array(i,5));
   if(pos_node==0)
      B(neg_node,j)=B(neg_node,j)-amp_F;
   elseif(neg_node==0)
      B(pos_node,j)=B(pos_node,j)+amp_F;
   else
     B(neg_node,j)=B(neg_node,j)-amp_F;
     B(pos_node,j)=B(pos_node,j)+amp_F;
   end

end

%for VCVS

for i=1:size_VCVS_array(1,1)
   pos_node=str2double(VCVS_array(i,2));
   neg_node=str2double(VCVS_array(i,3));
   pos_source_node=str2double(VCVS_array(i,4));
   neg_source_node=str2double(VCVS_array(i,5));
   amp_E=str2double(VCVS_array(i,6));
   if(neg_node~=0)
      B(neg_node,B_new)=B(neg_node,B_new)-1;
      C(C_new,neg_node)=C(C_new,neg_node)-1;
   end
   if(pos_node~=0)
      B(pos_node,B_new)=B(pos_node,B_new)+1;
      C(C_new,pos_node)=C(C_new,pos_node)+1;
   end
   if(neg_source_node~=0)
      C(C_new,neg_source_node)=C(C_new,neg_source_node)+amp_E;
   end
   if(pos_source_node~=0)
      C(C_new,pos_source_node)=C(C_new,pos_source_node)-amp_E;
   end
   B_new=B_new+1;
   C_new=C_new+1;
end


%For CCVS

for i=1:size_CCVS_array(1,1)
   for j=1:size_V_source_array(1,1)
      if tracker(1,j)==CCVS_array(i,4)
         break;
      end
   end
   pos_node=str2double(CCVS_array(i,2));
   neg_node=str2double(CCVS_array(i,3));
   amp_H=str2double(CCVS_array(i,5));
   if(neg_node~=0)
      B(neg_node,B_new)=B(neg_node,B_new)-1;
      C(C_new,neg_node)=C(C_new,neg_node)-1;
   end
   if(pos_node~=0)
      B(pos_node,B_new)=B(pos_node,B_new)+1;
      C(C_new,pos_node)=C(C_new,pos_node)+1;
   end
   D(C_new,j)=D(C_new,j)-amp_H;
   B_new=B_new+1;
   C_new=C_new+1;
end



%for capacitors
```

```
for i=1:size_C_array(1,1)
   pos_node=str2double(C_array(i,2));
   neg_node=str2double(C_array(i,3));
   if(DC)
      C_mag=inf;
   else
      C_mag=1/(2j*pi*freq*str2double(C_array(i,4)));
   end
   if(pos_node==0)
      A(neg_node,neg_node)=A(neg_node,neg_node)+(1/C_mag);
   elseif(neg_node==0)
      A(pos_node,pos_node)=A(pos_node,pos_node)+(1/C_mag);
   else
      A(pos_node,pos_node)=A(pos_node,pos_node)+(1/C_mag);
      A(neg_node,neg_node)=A(neg_node,neg_node)+(1/C_mag);
      A(pos_node,neg_node)=A(pos_node,neg_node)-(1/C_mag);
      A(neg_node,pos_node)=A(neg_node,pos_node)-(1/C_mag);
   end
end



%for inductors
for i=1:size_L_array(1,1)
   pos_node=str2double(L_array(i,2));
   neg_node=str2double(L_array(i,3));
   if(DC)
      L_mag=0;
   else
      L_mag=(2j*pi*freq*str2double(L_array(i,4)));
   end
   if(pos_node==0)
      A(neg_node,neg_node)=A(neg_node,neg_node)+(1/L_mag);
   elseif(neg_node==0)
      A(pos_node,pos_node)=A(pos_node,pos_node)+(1/L_mag);
   else
      A(pos_node,pos_node)=A(pos_node,pos_node)+(1/L_mag);
      A(neg_node,neg_node)=A(neg_node,neg_node)+(1/L_mag);
      A(pos_node,neg_node)=A(pos_node,neg_node)-(1/L_mag);
      A(neg_node,pos_node)=A(neg_node,pos_node)-(1/L_mag);
   end
end

%% Solving the matrix to find the node voltages
M=[A B;C D];
S;
```

```
x=M\S;

node_voltage_array=x(1:number_of_nodes,:);


%% Solving for branch_Currents


volt_count=0;

%for Vsource
for i=1:size_V_source_array(1,1)
   volt_count=volt_count+1;
   branch_current_array(i,1)=x(number_of_nodes+volt_count,1);
end

%Solving for Isource
I_count=0;
for i=1:size_I_source_array(1,1)
   if(DC)
      I_count=I_count+1;
      branch_current_array(i+volt_count,1)=I_source_array(i,5);
   else
      I_count=I_count+1;
      temp=str2double(I_source_array(i,5));

temp=temp*cos(str2double(I_source_array(i,6)))+temp*sin(str2double(I_source_array(1,6)))*
1j;
      branch_current_array(i+volt_count,1)=temp;
   end
end
I_count=I_count+volt_count;

%for cccs
F_count=0;
for i=1:size_CCCS_array(1,1)
   for j=1:size_V_source_array(1,1)
      if tracker(1,j)==CCCS_array(i,4)
         break;
      end
   end
   F_count=F_count+1;

branch_current_array(i+I_count,1)=str2double(CCCS_array(i,5))*branch_current_array(j,1);
end
F_count=F_count+I_count;

%for vcvs
```

```
E_count=0;
for i=1:size_VCVS_array(1,1)
   E_count=E_count+1;
   branch_current_array(i+F_count,1)=x(number_of_nodes+volt_count+E_count,1);
end
new_E_count=E_count+F_count;

%for CCVS
H_count=0;
for i=1:size_CCVS_array(1,1)

   H_count=H_count+1;

branch_current_array(i+new_E_count,1)=x(number_of_nodes+volt_count+E_count+H_coun
t,1);

end
new_H_count=H_count+new_E_count;


%for VCCS
G_count=0;
for i=1:size_VCCS_array(1,1)
   G_count=G_count+1;
   if(str2double(VCCS_array(i,4))==0)
      a=0;
      b=node_voltage_array(str2double(VCCS_array(i,5)));
   elseif(str2double(VCCS_array(i,5))==0)
      b=0;
      a=node_voltage_array(str2double(VCCS_array(i,4)));
   else
      a=node_voltage_array(str2double(VCCS_array(i,4)));
      b=node_voltage_array(str2double(VCCS_array(i,5)));
   end
   temp=a-b;
   branch_current_array(i+new_H_count,1)=temp*str2double(I_source_array(i,6));
end
G_count=G_count+new_H_count;

%for R
R_count=0;
for i=1:size_R_array(1,1)
   R_count=R_count+1;
   if(str2double(R_array(i,2))==0)
      a=0;
      b=node_voltage_array(str2double(R_array(i,3)));
   elseif(str2double(R_array(i,3))==0)
      b=0;
      a=node_voltage_array(str2double(R_array(i,2)));
   else
      b=node_voltage_array(str2double(R_array(i,3)));
      a=node_voltage_array(str2double(R_array(i,2)));
   end
   temp=a-b;
   branch_current_array(i+G_count,1)=temp/str2double(R_array(i,4));
end
R_count=R_count+G_count;

%for C

C_count=0;
for i=1:size_C_array(1,1)

   if(DC)
      C_mag=inf;
   else
      C_mag=1/(2j*pi*freq*str2double(C_array(i,4)));
   end
   C_count=C_count+1;
   if(str2double(C_array(i,2))==0)
      a=0;
      b=node_voltage_array(str2double(C_array(i,3)));
   elseif(str2double(C_array(i,3))==0)
      b=0;
      a=node_voltage_array(str2double(C_array(i,2)));
   else
     a=node_voltage_array(str2double(C_array(i,2)));
     b=node_voltage_array(str2double(C_array(i,3)));
   end
   temp=a-b;
   branch_current_array(i+R_count,1)=temp/C_mag;
end
C_count=C_count+R_count;


%for L
for i=1:size_L_array(1,1)

   if(DC)
      L_mag=0;
   else
      L_mag=2j*pi*freq*str2double(L_array(i,4));
   end
   if(str2double(L_array(i,2))==0)
      a=0;
      b=node_voltage_array(str2double(L_array(i,3)));
```

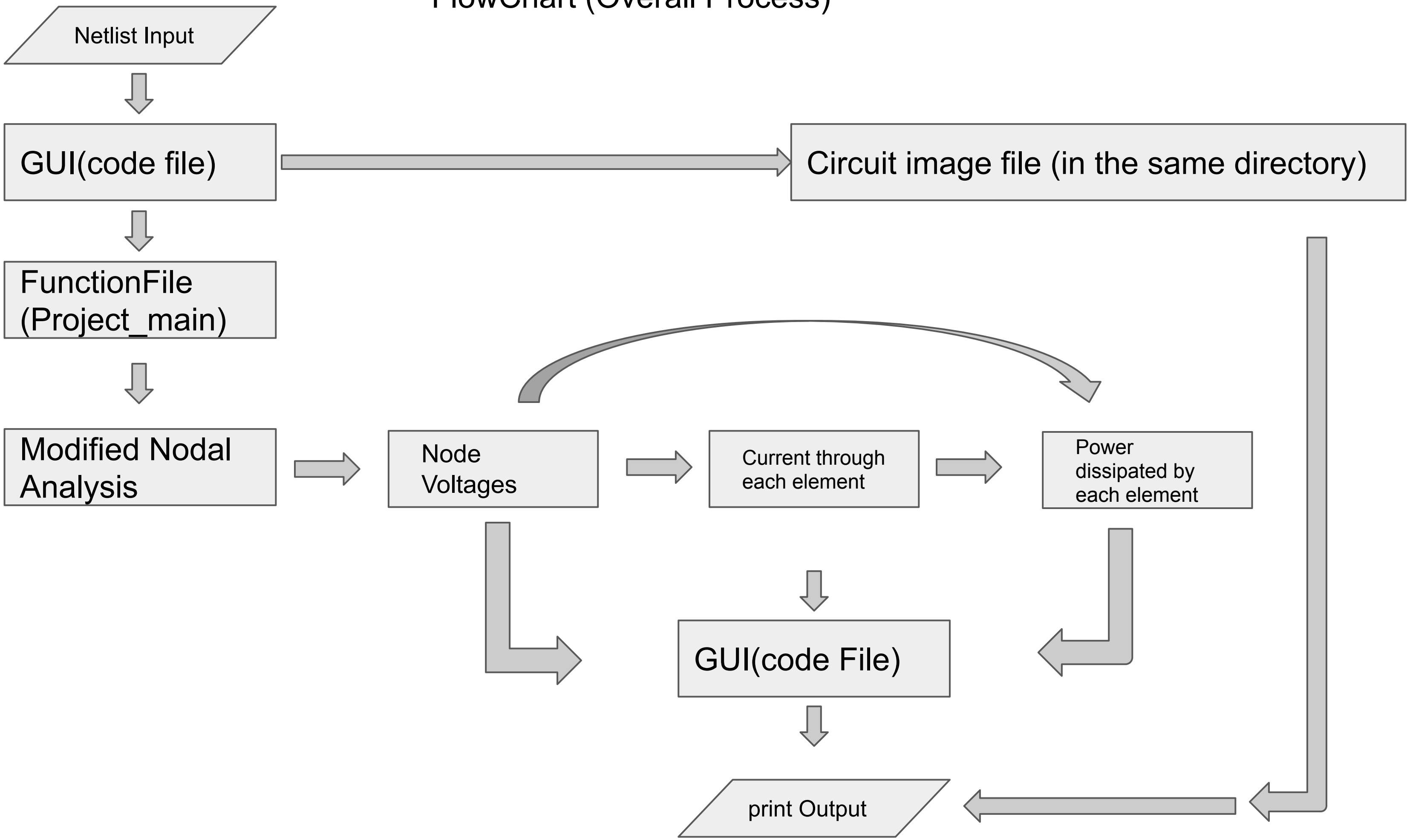
```
    elseif(str2double(L_array(i,3))==0)
        b=0;
        a=node_voltage_array(str2double(L_array(i,2)));
    else
        a=node_voltage_array(str2double(L_array(i,2)));
        b=node_voltage_array(str2double(L_array(i,3)));
    end
    temp=a-b;
    branch_current_array(i+C_count,1)=temp/L_mag;
end



branch_current_string="Current through,"+newline;
power_string="Power Dissipated by, "+newline;


%% Power disspated

new_I=0;
%for Vsources
for i=1:size_V_source_array(1,1)
    if(DC)
        power_array=str2double(V_source_array(i,5))*conj(branch_current_array(new_I+i,1));
        branch_current_string=branch_current_string+V_source_array(i,1)+":
"+branch_current_array(i,1)+" A"+newline;
        power_string=power_string+V_source_array(i,1)+": "+branch_current_array(i,1)+"
VA"+newline;
    else
        mag=str2double(V_source_array(i,5));

volt_mag=mag*cos(str2double(V_source_array(i,6)))+mag*sin(str2double(V_source_array(i,
6)))*1j;
        power_array(i,1)=0.5*volt_mag*conj(branch_current_array(new_I+i,1));
        branch_current_string=branch_current_string+V_source_array(i,1)+":
"+branch_current_array(i,1)+" A"+newline;
        power_string=power_string+V_source_array(i,1)+": "+branch_current_array(i,1)+"
VA"+newline;
    end
    new_I=new_I+1;
end

%for Isources
for i=1:size_I_source_array(1,1)
    pos_node=str2double(I_source_array(i,2));
    neg_node=str2double(I_source_array(i,3));
    if(pos_node==0)
        a=0;
```

```
      b=node_voltage_array(neg_node);
   elseif(neg_node)==0
      b=0;
      a=node_voltage_array(pos_node);
   else
      a=node_voltage_array(pos_node);
      b=node_voltage_array(neg_node);
   end

   voltage=a-b;
   if(DC)
      power_array(new_I+i,1)=voltage*conj(branch_current_array(new_I+i,1));
   else
      power_array(new_I+i,1)=0.5*voltage*conj(branch_current_array(new_I+i,1));
   end
   branch_current_string=branch_current_string+I_source_array(i,1)+":
"+branch_current_array(new_I+i,1)+" A"+newline;
   power_string=power_string+I_source_array(i,1)+": "+power_array(new_I+i,1)+"
VA"+newline;
end
new_I=new_I+size_I_source_array(1,1);

for i=1:size_CCCS_array(1,1)
   pos_node=str2double(CCCS_array(i,2));
   neg_node=str2double(CCCS_array(i,3));
   if(pos_node==0)
      a=0;
      b=node_voltage_array(neg_node);
   elseif(neg_node)==0
      b=0;
      a=node_voltage_array(pos_node);
   else
      a=node_voltage_array(pos_node);
      b=node_voltage_array(neg_node);
   end

   voltage=a-b;
   if(DC)
      power_array(new_I+i,1)=voltage*conj(branch_current_array(new_I+i,1));
   else
      power_array(new_I+i,1)=0.5*voltage*conj(branch_current_array(new_I+i,1));
   end
   branch_current_string=branch_current_string+CCCS_array(i,1)+":
"+branch_current_array(new_I+i,1)+" A"+newline;
   power_string=power_string+CCCS_array(i,1)+": "+power_array(new_I+i,1)+"
VA"+newline;
end
new_I=new_I+size_CCCS_array(1,1);
```

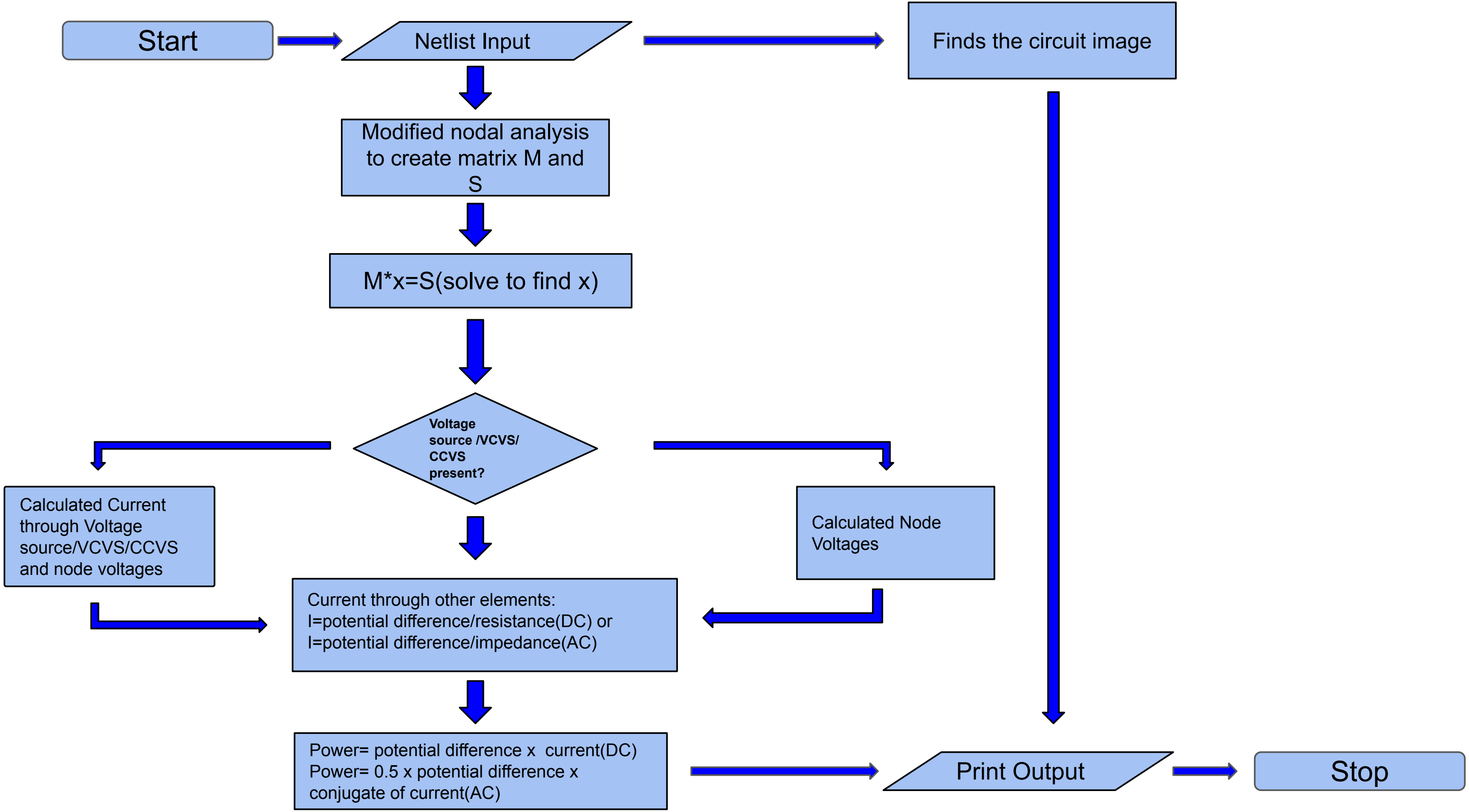
```
for i=1:size_VCVS_array(1,1)
   pos_node=str2double(VCVS_array(i,2));
   neg_node=str2double(VCVS_array(i,3));
   if(pos_node==0)
      a=0;
      b=node_voltage_array(neg_node);
   elseif(neg_node)==0
      b=0;
      a=node_voltage_array(pos_node);
   else
      a=node_voltage_array(pos_node);
      b=node_voltage_array(neg_node);
   end

   voltage=a-b;
   if(DC)
      power_array(new_l+i,1)=voltage*conj(branch_current_array(new_l+i,1));
   else
      power_array(new_l+i,1)=0.5*voltage*conj(branch_current_array(new_l+i,1));
   end
   branch_current_string=branch_current_string+VCVS_array(i,1)+":
"+branch_current_array(new_l+i,1)+" A"+newline;
   power_string=power_string+VCVS_array(i,1)+": "+power_array(new_l+i,1)+" VA"+newline;
end
new_l=new_l+size_VCVS_array(1,1);

for i=1:size_CCVS_array(1,1)
   pos_node=str2double(CCVS_array(i,2));
   neg_node=str2double(CCVS_array(i,3));
   if(pos_node==0)
      a=0;
      b=node_voltage_array(neg_node);
   elseif(neg_node)==0
      b=0;
      a=node_voltage_array(pos_node);
   else
      a=node_voltage_array(pos_node);
      b=node_voltage_array(neg_node);
   end

   voltage=a-b;
   if(DC)
      power_array(new_l+i,1)=voltage*conj(branch_current_array(new_l+i,1));
   else
      power_array(new_l+i,1)=0.5*voltage*conj(branch_current_array(new_l+i,1));
   end
   branch_current_string=branch_current_string+CCVS_array(i,1)+":
"+branch_current_array(new_I+i,1)+" A"+newline;
   power_string=power_string+CCVS_array(i,1)+": "+power_array(new_I+i,1)+"
VA"+newline;
end
new_I=new_I+size_CCVS_array(1,1);

for i=1:size_VCCS_array(1,1)
   pos_node=str2double(VCCS_array(i,2));
   neg_node=str2double(VCCS_array(i,3));
   if(pos_node==0)
      a=0;
      b=node_voltage_array(neg_node);
   elseif(neg_node)==0
      b=0;
      a=node_voltage_array(pos_node);
   else
      a=node_voltage_array(pos_node);
      b=node_voltage_array(neg_node);
   end

   voltage=a-b;
   if(DC)
      power_array(new_I+i,1)=voltage*conj(branch_current_array(new_I+i,1));
   else
      power_array(new_I+i,1)=0.5*voltage*conj(branch_current_array(new_I+i,1));
   end
   branch_current_string=branch_current_string+VCCS_array(i,1)+":
"+branch_current_array(new_I+i,1)+" A"+newline;
   power_string=power_string+VCCS_array(i,1)+": "+power_array(new_I+i,1)+"
VA"+newline;
end
new_I=new_I+size_VCCS_array(1,1);


%for R,L,C
for i=1:size_R_array(1,1)
   pos_node=str2double(R_array(i,2));
   neg_node=str2double(R_array(i,3));
   if(pos_node==0)
      a=0;
      b=node_voltage_array(neg_node);
   elseif(neg_node)==0
      b=0;
      a=node_voltage_array(pos_node);
   else
      a=node_voltage_array(pos_node);
      b=node_voltage_array(neg_node);
```

```
    end

    voltage=a-b;
    if(DC)
        power_array(new_I+i,1)=voltage*conj(branch_current_array(new_I+i,1));
    else
        power_array(new_I+i,1)=0.5*voltage*conj(branch_current_array(new_I+i,1));
    end
    branch_current_string=branch_current_string+R_array(i,1)+":
"+branch_current_array(new_I+i,1)+" A"+newline;
    power_string=power_string+R_array(i,1)+": "+power_array(new_I+i,1)+" VA"+newline;
end
new_I=new_I+size_R_array(1,1);

for i=1:size_C_array(1,1)
    pos_node=str2double(C_array(i,2));
    neg_node=str2double(C_array(i,3));
    if(pos_node==0)
        a=0;
        b=node_voltage_array(neg_node);
    elseif(neg_node)==0
        b=0;
        a=node_voltage_array(pos_node);
    else
        a=node_voltage_array(pos_node);
        b=node_voltage_array(neg_node);
    end

    voltage=a-b;
    if(DC)
        power_array(new_I+i,1)=voltage*conj(branch_current_array(new_I+i,1));
    else
        power_array(new_I+i,1)=0.5*voltage*conj(branch_current_array(new_I+i,1));
    end
    branch_current_string=branch_current_string+C_array(i,1)+":
"+branch_current_array(new_I+i,1)+" A"+newline;
    power_string=power_string+C_array(i,1)+": "+power_array(new_I+i,1)+" VA"+newline;
end
new_I=new_I+size_C_array(1,1);

for i=1:size_L_array(1,1)
    pos_node=str2double(L_array(i,2));
    neg_node=str2double(L_array(i,3));
    if(pos_node==0)
        a=0;
        b=node_voltage_array(neg_node);
    elseif(neg_node)==0
        b=0;
        a=node_voltage_array(pos_node);
    else
        a=node_voltage_array(pos_node);
        b=node_voltage_array(neg_node);
    end

    voltage=a-b;
    if(DC)
        power_array(new_I+i,1)=voltage*conj(branch_current_array(new_I+i,1));
    else
        power_array(new_I+i,1)=0.5*voltage*conj(branch_current_array(new_I+i,1));
    end
    branch_current_string=branch_current_string+L_array(i,1)+":
"+branch_current_array(new_I+i,1)+" A"+newline;
    power_string=power_string+L_array(i,1)+": "+power_array(new_I+i,1)+" VA"+newline;
end
new_I=new_I+size_L_array(1,1);


node_voltage_string="V0= 0V"+newline;
for i=1:number_of_nodes
    node_voltage_string=node_voltage_string+"V"+i+"= "+node_voltage_array(i,1)+newline;
end


%% sending info to GUI
a=node_voltage_string;
b=branch_current_string;
c=power_string;
```

FlowChart (Overall Process)


Netlist Input
GUI(code file)
Circuit image file (in the same directory)
FunctionFile (Project_main)
Modified Nodal Analysis
Node Voltages
Current through each element
Power dissipated by each element
GUI(code File)
print Output

FlowChart (Algorithm)



Start

Netlist Input

Finds the circuit image

Modified nodal analysis to create matrix M and S

M*x=S(solve to find x)

Voltage source /VCVS/ CCVS present?

Calculated Current through Voltage source/VCVS/CCVS and node voltages

Calculated Node Voltages

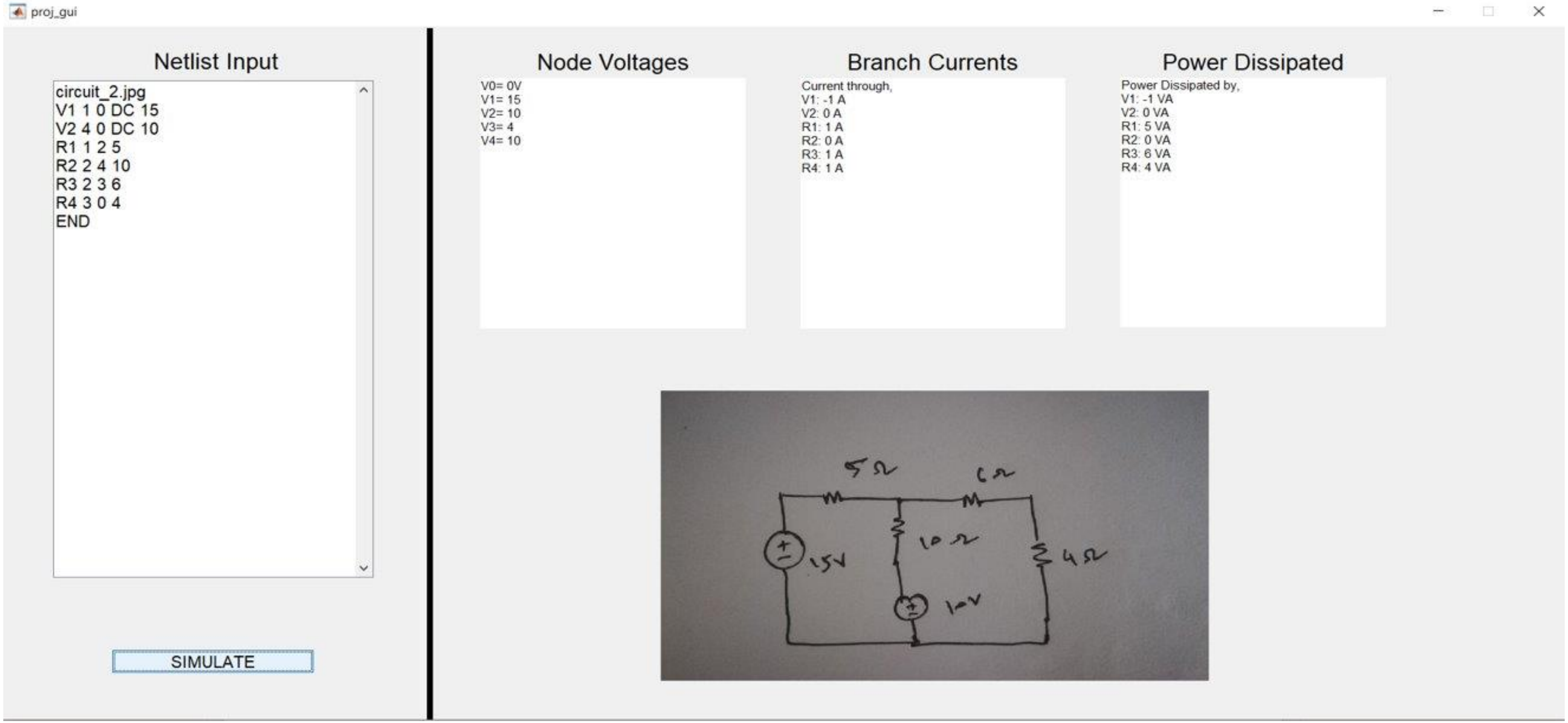
Current through other elements:
I=potential difference/resistance(DC) or
I=potential difference/impedance(AC)

Power= potential difference x current(DC)
Power= 0.5 x potential difference x conjugate of current(AC)

Print Output

Stop

Test Case 1: Voltage Sources and resistors



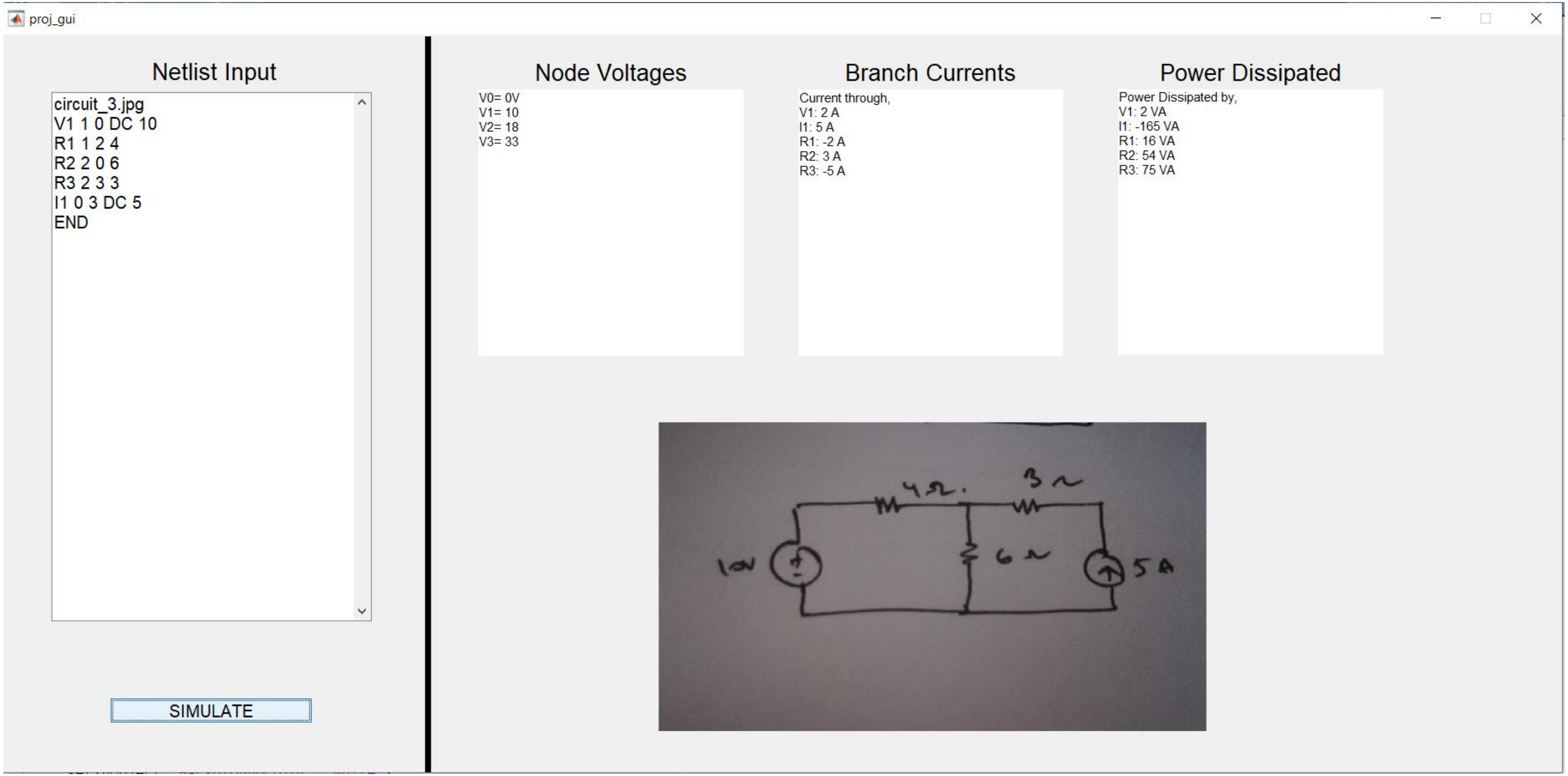
```
circuit_2.jpg
V1 1 0 DC 15
V2 4 0 DC 10
R1 1 2 5
R2 2 4 10
R3 2 3 6
R4 3 0 4
END
```

```
V0= 0V
V1= 15
V2= 10
V3= 4
V4= 10
```

```
Current through,
V1: -1 A
V2: 0 A
R1: 1 A
R2: 0 A
R3: 1 A
R4: 1 A
```

```
Power Dissipated by,
V1: -1 VA
V2: 0 VA
R1: 5 VA
R2: 0 VA
R3: 6 VA
R4: 4 VA
```

Test Case 2:(Voltage Source, Current Source and resistor)



Netlist Input

```
circuit_3.jpg
V1 1 0 DC 10
R1 1 2 4
R2 2 0 6
R3 2 3 3
I1 0 3 DC 5
END
```

Node Voltages

```
V0= 0V
V1= 10
V2= 18
V3= 33
```

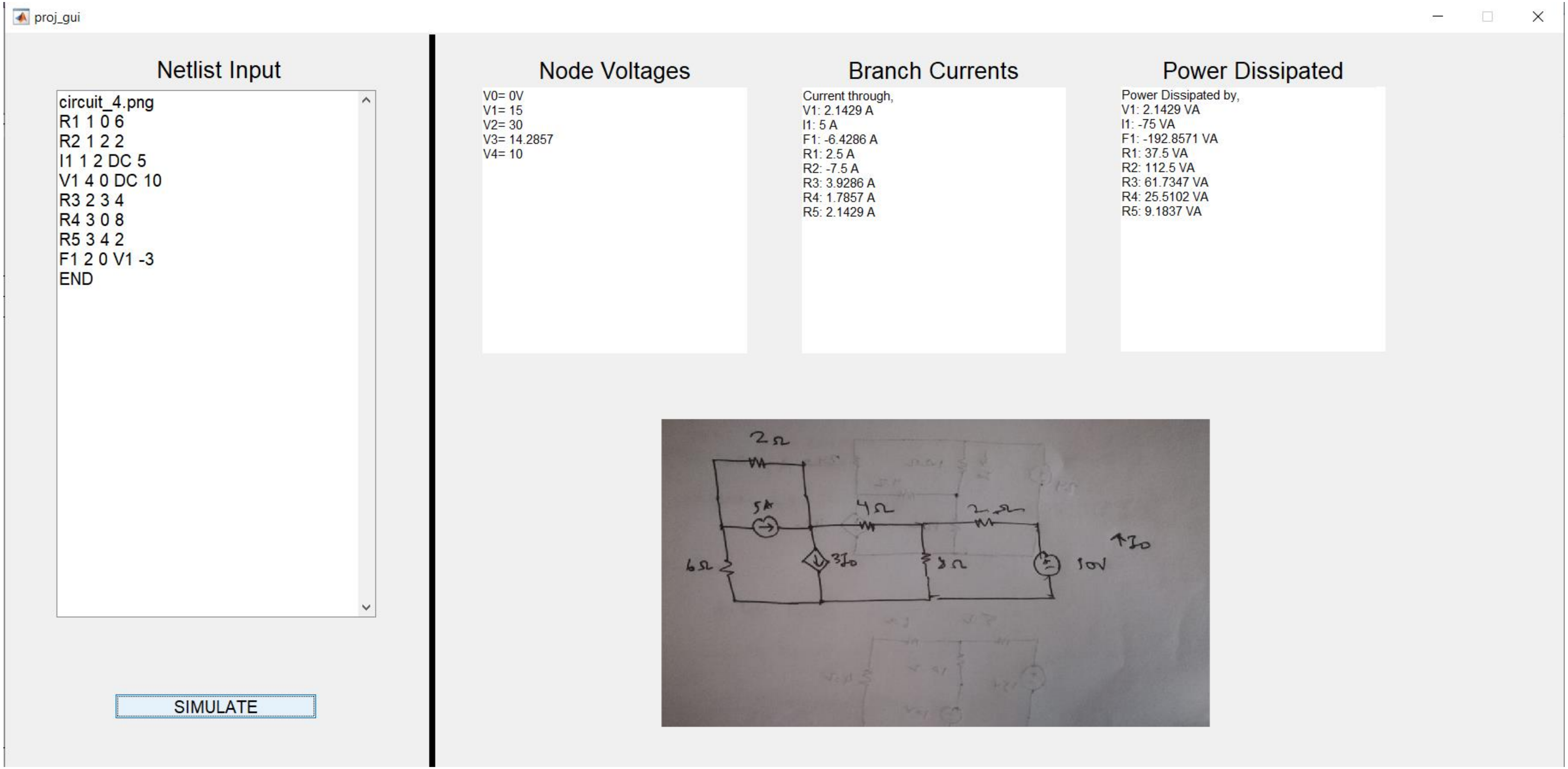
Branch Currents

```
Current through,
V1: 2 A
I1: 5 A
R1: -2 A
R2: 3 A
R3: -5 A
```

Power Dissipated

```
Power Dissipated by,
V1: 2 VA
I1: -165 VA
R1: 16 VA
R2: 54 VA
R3: 75 VA
```

Test Case 3: CCCS with sources and resistor(DC)



Netlist Input

```
circuit_4.png
R1 1 0 6
R2 1 2 2
I1 1 2 DC 5
V1 4 0 DC 10
R3 2 3 4
R4 3 0 8
R5 3 4 2
F1 2 0 V1 -3
END
```

Node Voltages

```
V0= 0V
V1= 15
V2= 30
V3= 14.2857
V4= 10
```

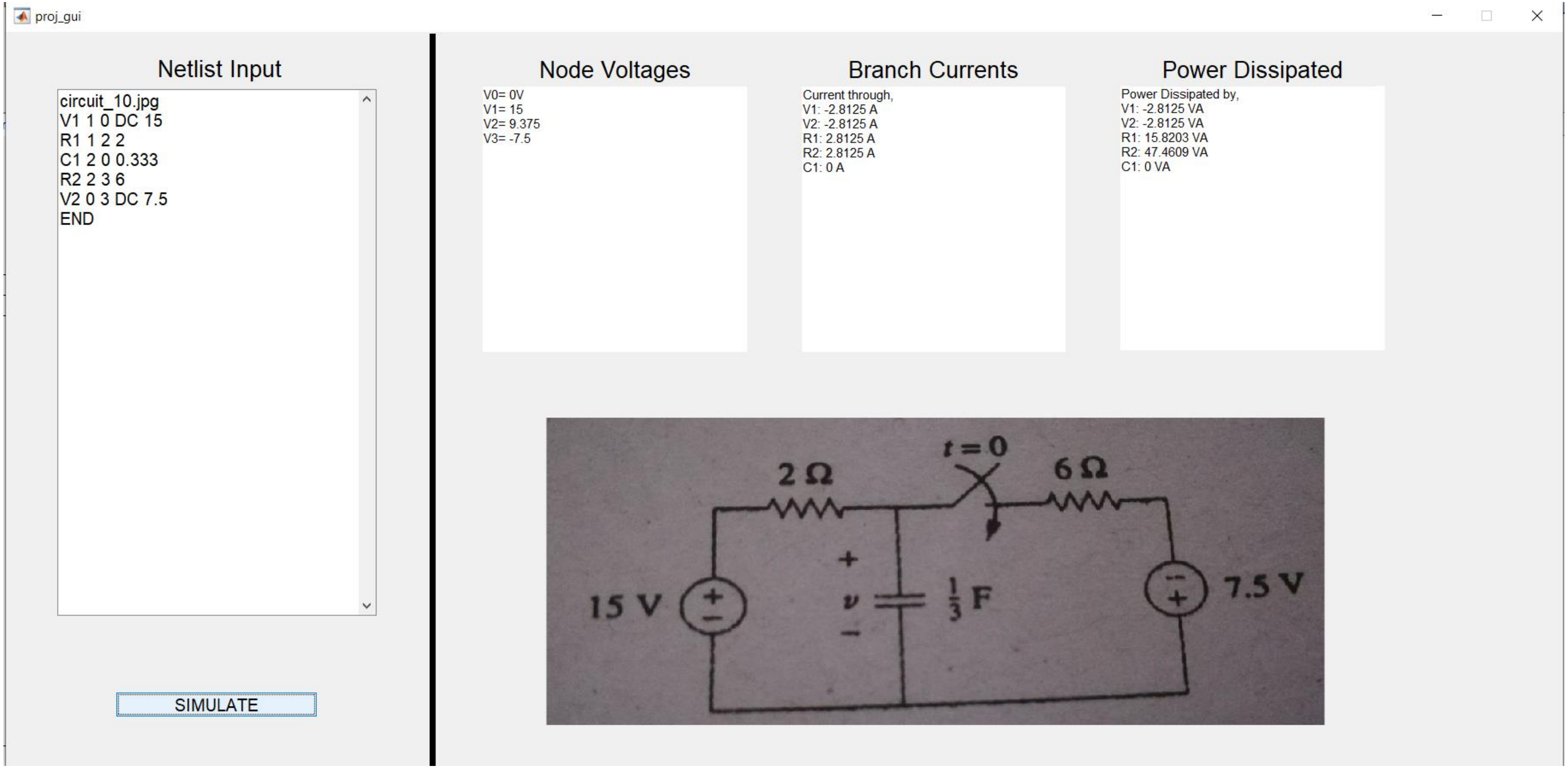
Branch Currents

```
Current through,
V1: 2.1429 A
I1: 5 A
F1: -6.4286 A
R1: 2.5 A
R2: -7.5 A
R3: 3.9286 A
R4: 1.7857 A
R5: 2.1429 A
```

Power Dissipated

```
Power Dissipated by,
V1: 2.1429 VA
I1: -75 VA
F1: -192.8571 VA
R1: 37.5 VA
R2: 112.5 VA
R3: 61.7347 VA
R4: 25.5102 VA
R5: 9.1837 VA
```

Test Case 4: Capacitor (DC circuit)



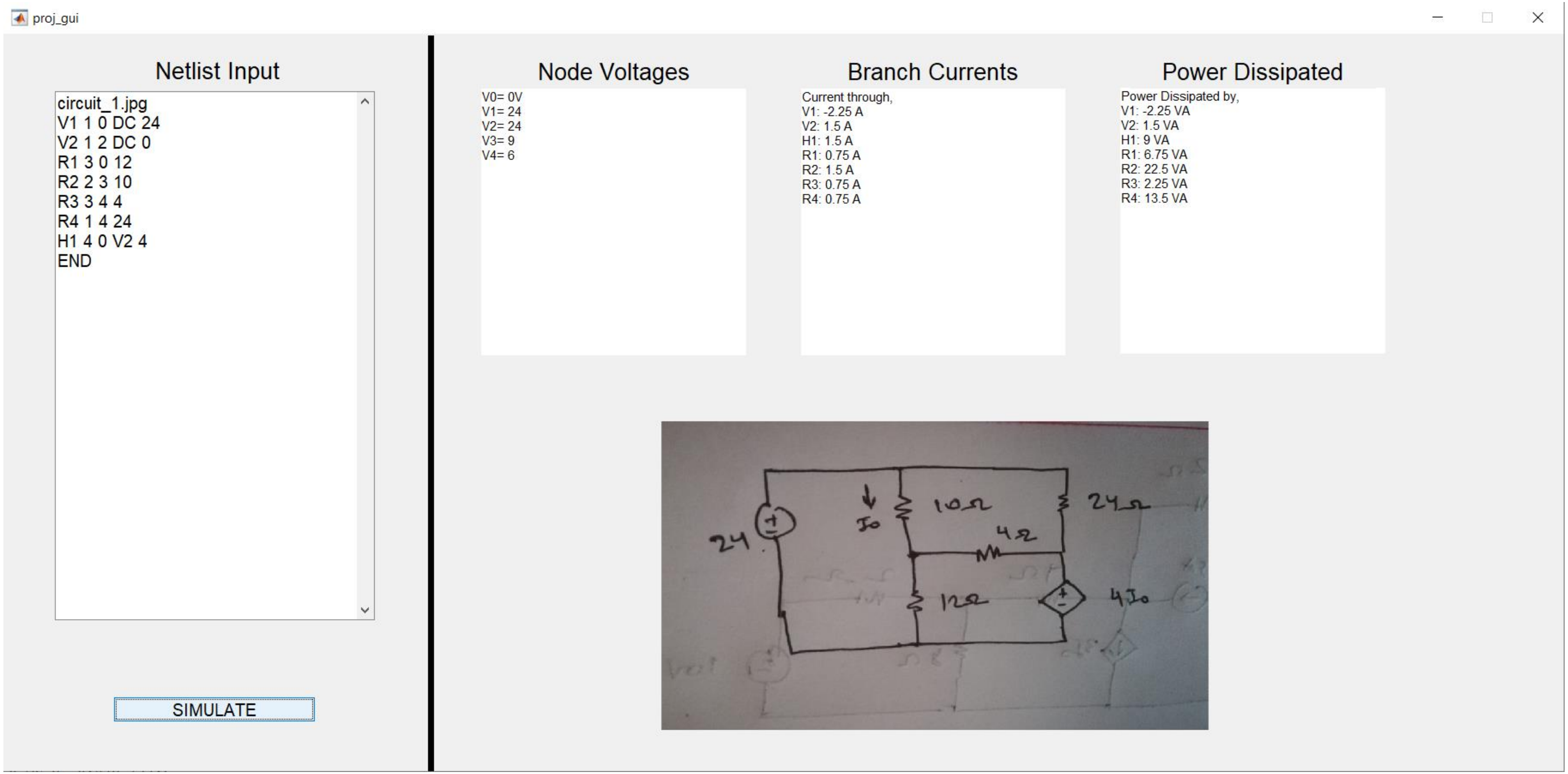
```
circuit_10.jpg
V1 1 0 DC 15
R1 1 2 2
C1 2 0 0.333
R2 2 3 6
V2 0 3 DC 7.5
END
```

```
V0= 0V
V1= 15
V2= 9.375
V3= -7.5
```

```
Current through,
V1: -2.8125 A
V2: -2.8125 A
R1: 2.8125 A
R2: 2.8125 A
C1: 0 A
```

```
Power Dissipated by,
V1: -2.8125 VA
V2: -2.8125 VA
R1: 15.8203 VA
R2: 47.4609 VA
C1: 0 VA
```

Test Case 5: CCVS(DC)



Netlist Input

```
circuit_1.jpg
V1 1 0 DC 24
V2 1 2 DC 0
R1 3 0 12
R2 2 3 10
R3 3 4 4
R4 1 4 24
H1 4 0 V2 4
END
```

Node Voltages

```
V0= 0V
V1= 24
V2= 24
V3= 9
V4= 6
```

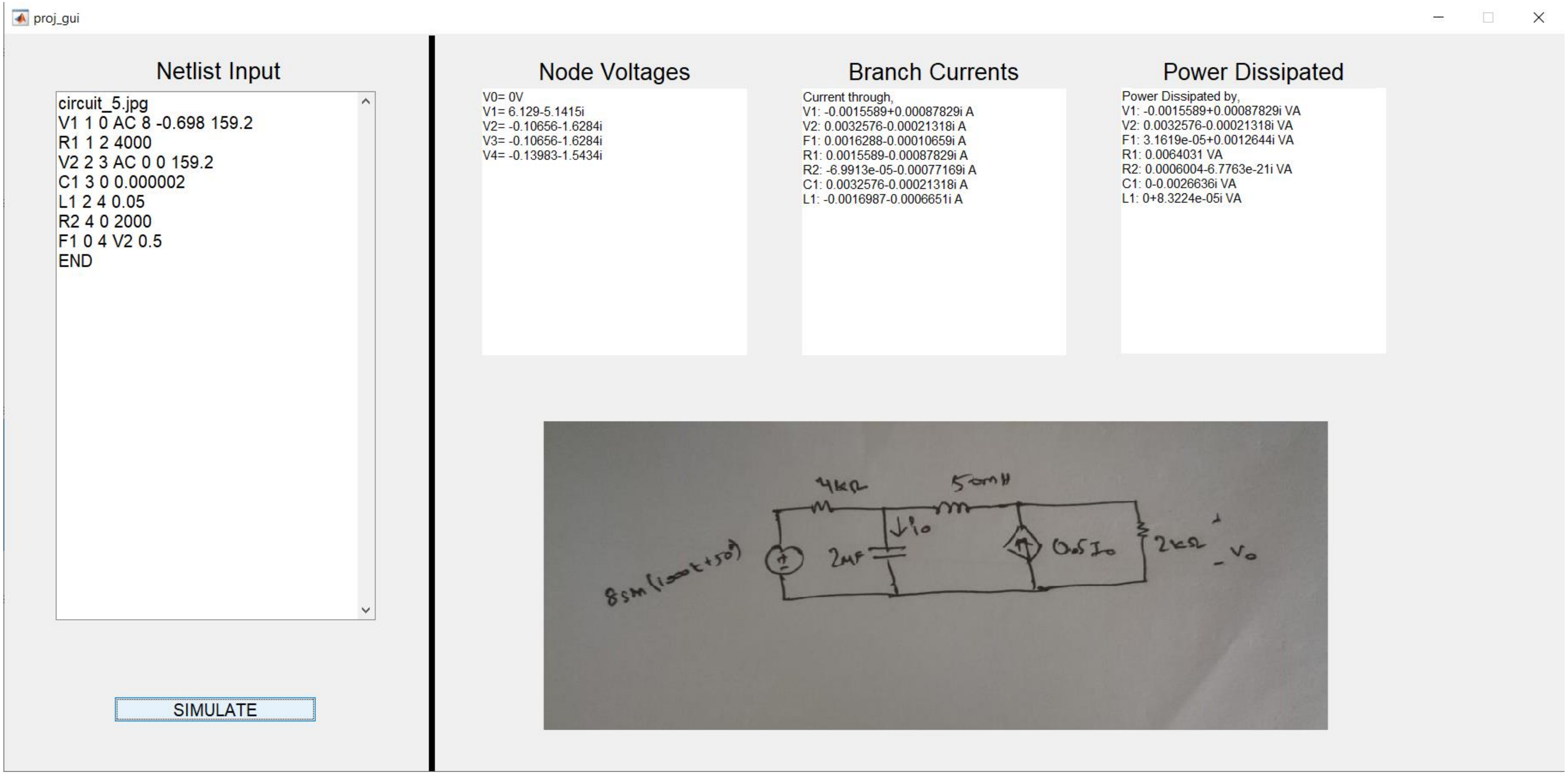
Branch Currents

```
Current through,
V1: -2.25 A
V2: 1.5 A
H1: 1.5 A
R1: 0.75 A
R2: 1.5 A
R3: 0.75 A
R4: 0.75 A
```

Power Dissipated

```
Power Dissipated by,
V1: -2.25 VA
V2: 1.5 VA
H1: 9 VA
R1: 6.75 VA
R2: 22.5 VA
R3: 2.25 VA
R4: 13.5 VA
```

Test Case 6: CCCS (AC circuit)



Netlist Input

```
circuit_5.jpg
V1 1 0 AC 8 -0.698 159.2
R1 1 2 4000
V2 2 3 AC 0 0 159.2
C1 3 0 0.000002
L1 2 4 0.05
R2 4 0 2000
F1 0 4 V2 0.5
END
```

Node Voltages

```
V0= 0V
V1= 6.129-5.1415i
V2= -0.10656-1.6284i
V3= -0.10656-1.6284i
V4= -0.13983-1.5434i
```

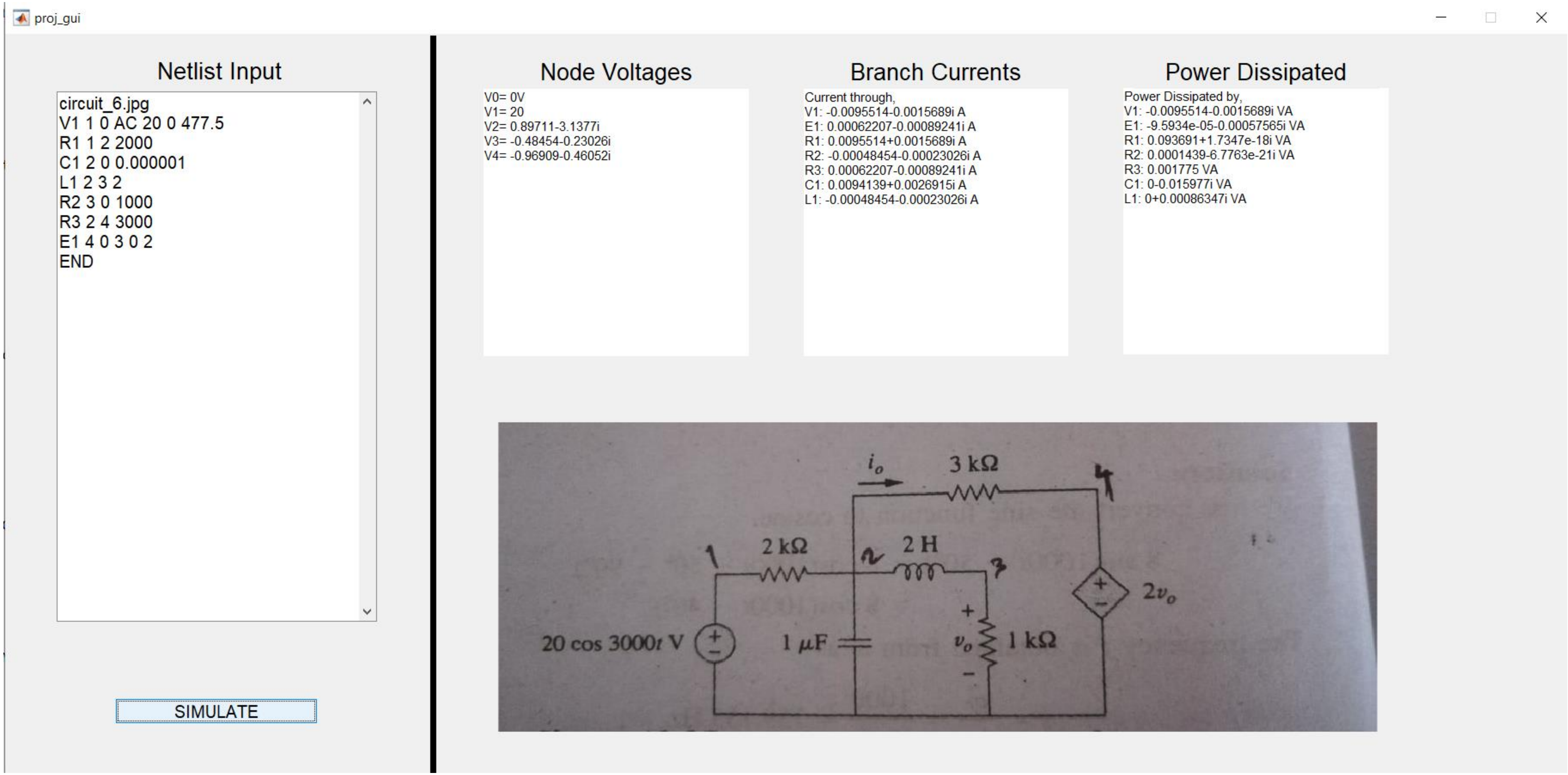
Branch Currents

```
Current through,
V1: -0.0015589+0.00087829i A
V2: 0.0032576-0.00021318i A
F1: 0.0016288-0.00010659i A
R1: 0.0015589-0.00087829i A
R2: -6.9913e-05-0.00077169i A
C1: 0.0032576-0.00021318i A
L1: -0.0016987-0.0006651i A
```

Power Dissipated

```
Power Dissipated by,
V1: -0.0015589+0.00087829i VA
V2: 0.0032576-0.00021318i VA
F1: 3.1619e-05+0.0012644i VA
R1: 0.0064031 VA
R2: 0.0006004-6.7763e-21i VA
C1: 0-0.0026636i VA
L1: 0+8.3224e-05i VA
```

Test Case 7: VCVS(AC)



Netlist Input

```
circuit_6.jpg
V1 1 0 AC 20 0 477.5
R1 1 2 2000
C1 2 0 0.000001
L1 2 3 2
R2 3 0 1000
R3 2 4 3000
E1 4 0 3 0 2
END
```

Node Voltages

```
V0= 0V
V1= 20
V2= 0.89711-3.1377i
V3= -0.48454-0.23026i
V4= -0.96909-0.46052i
```

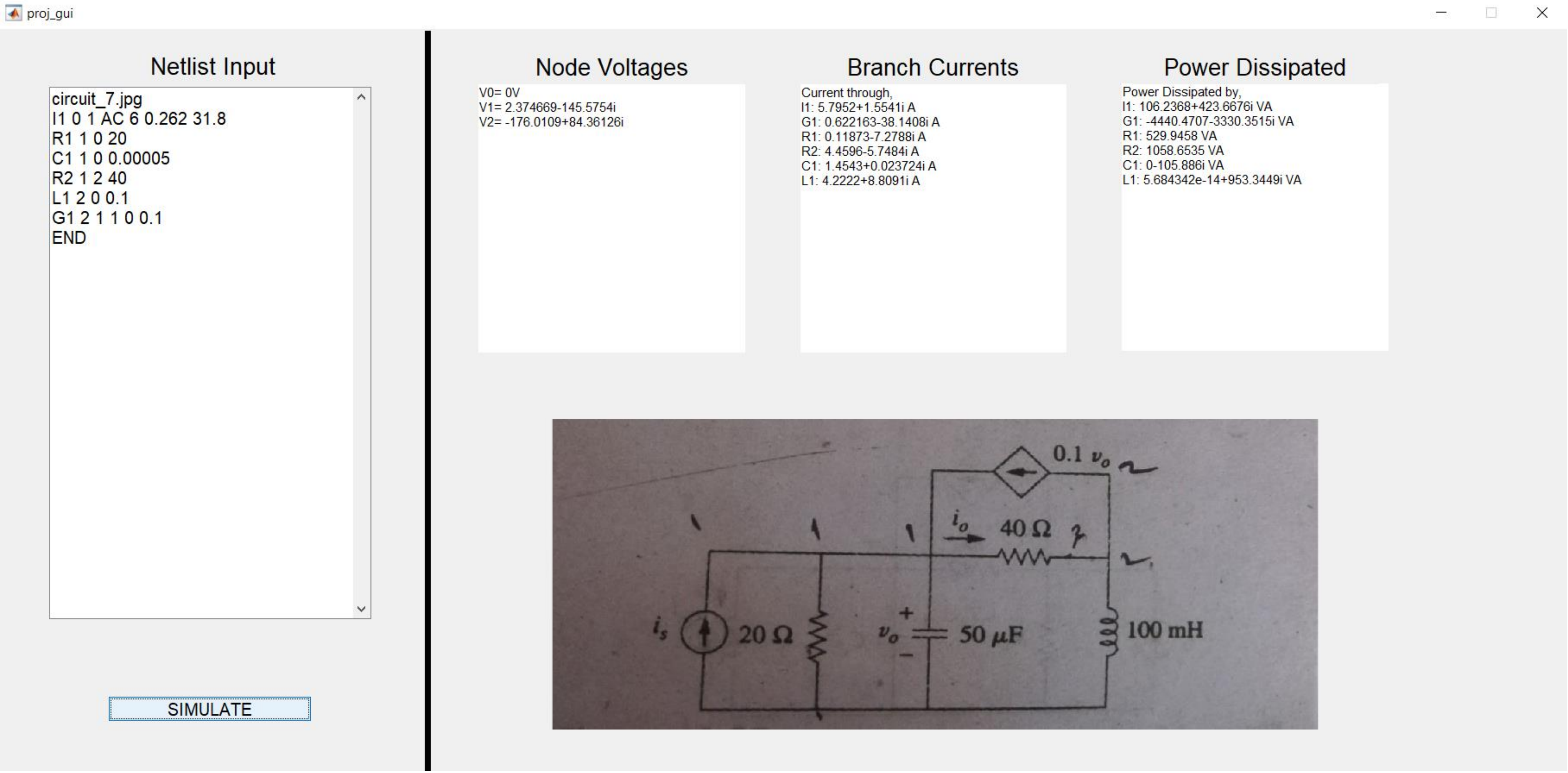
Branch Currents

```
Current through,
V1: -0.0095514-0.0015689i A
E1: 0.00062207-0.00089241i A
R1: 0.0095514+0.0015689i A
R2: -0.00048454-0.00023026i A
R3: 0.00062207-0.00089241i A
C1: 0.0094139+0.0026915i A
L1: -0.00048454-0.00023026i A
```

Power Dissipated

```
Power Dissipated by,
V1: -0.0095514-0.0015689i VA
E1: -9.5934e-05-0.00057565i VA
R1: 0.093691+1.7347e-18i VA
R2: 0.0001439-6.7763e-21i VA
R3: 0.001775 VA
C1: 0-0.015977i VA
L1: 0+0.00086347i VA
```

Test Case 8: VCCS(AC circuit)



Netlist Input

```
circuit_7.jpg
I1 0 1 AC 6 0.262 31.8
R1 1 0 20
C1 1 0 0.00005
R2 1 2 40
L1 2 0 0.1
G1 2 1 1 0 0.1
END
```

Node Voltages

```
V0= 0V
V1= 2.374669-145.5754i
V2= -176.0109+84.36126i
```

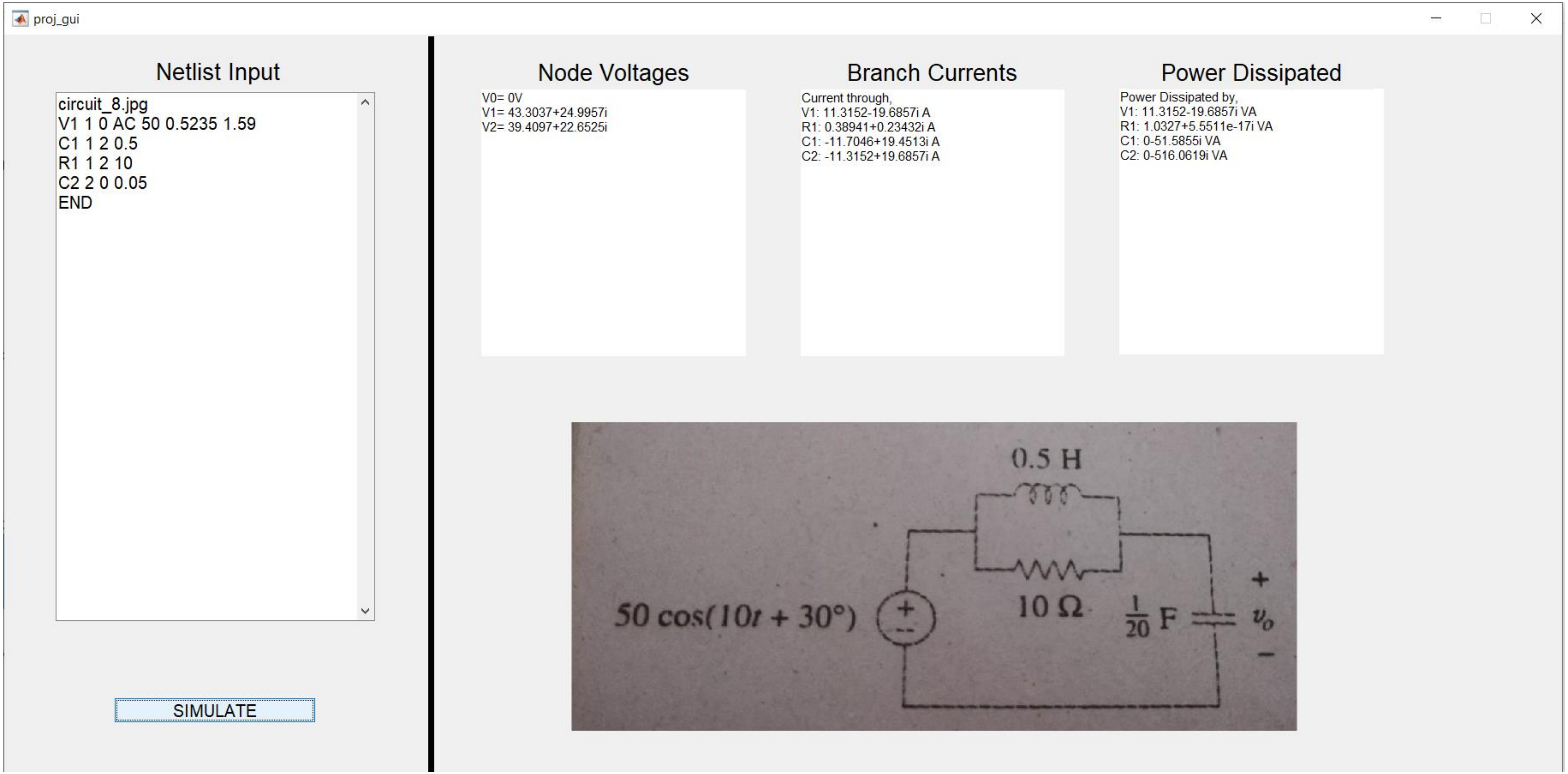
Branch Currents

```
Current through,
I1: 5.7952+1.5541i A
G1: 0.622163-38.1408i A
R1: 0.11873-7.2788i A
R2: 4.4596-5.7484i A
C1: 1.4543+0.023724i A
L1: 4.2222+8.8091i A
```

Power Dissipated

```
Power Dissipated by,
I1: 106.2368+423.6676i VA
G1: -4440.4707-3330.3515i VA
R1: 529.9458 VA
R2: 1058.6535 VA
C1: 0-105.886i VA
L1: 5.684342e-14+953.3449i VA
```

Test Case 9: Simple AC circuit without any dependent sources



Netlist Input

```
circuit_8.jpg
V1 1 0 AC 50 0.5235 1.59
C1 1 2 0.5
R1 1 2 10
C2 2 0 0.05
END
```

Node Voltages

```
V0= 0V
V1= 43.3037+24.9957i
V2= 39.4097+22.6525i
```

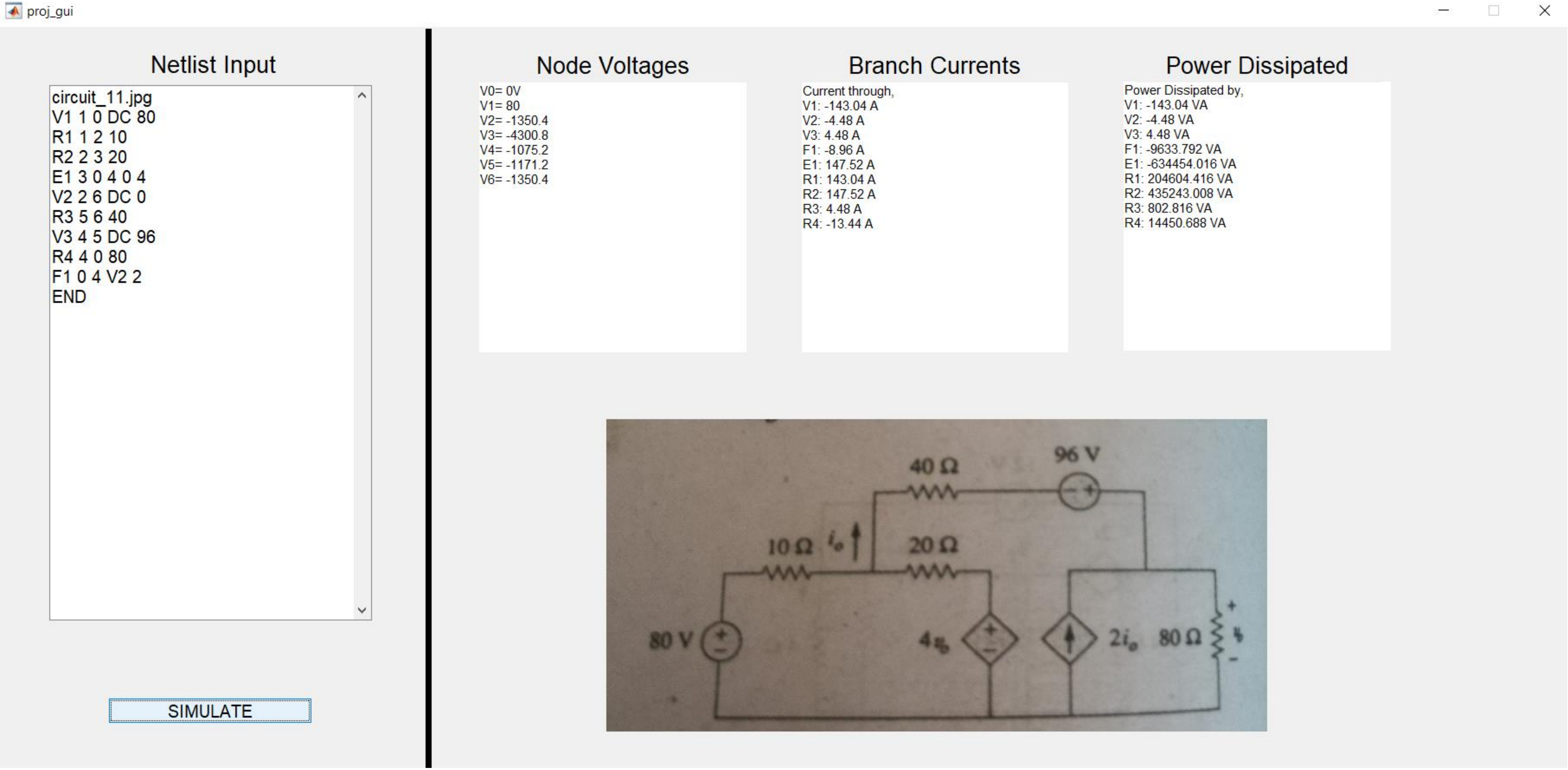
Branch Currents

```
Current through,
V1: 11.3152-19.6857i A
R1: 0.38941+0.23432i A
C1: -11.7046+19.4513i A
C2: -11.3152+19.6857i A
```

Power Dissipated

```
Power Dissipated by,
V1: 11.3152-19.6857i VA
R1: 1.0327+5.5511e-17i VA
C1: 0-51.5855i VA
C2: 0-516.0619i VA
```

Test Case 10:Mixed Dependent sources (DC)



```
circuit_11.jpg
V1 1 0 DC 80
R1 1 2 10
R2 2 3 20
E1 3 0 4 0 4
V2 2 6 DC 0
R3 5 6 40
V3 4 5 DC 96
R4 4 0 80
F1 0 4 V2 2
END
```

```
V0= 0V
V1= 80
V2= -1350.4
V3= -4300.8
V4= -1075.2
V5= -1171.2
V6= -1350.4
```

```
Current through,
V1: -143.04 A
V2: -4.48 A
V3: 4.48 A
F1: -8.96 A
E1: 147.52 A
R1: 143.04 A
R2: 147.52 A
R3: 4.48 A
R4: -13.44 A
```

```
Power Dissipated by,
V1: -143.04 VA
V2: -4.48 VA
V3: 4.48 VA
F1: -9633.792 VA
E1: -634454.016 VA
R1: 204604.416 VA
R2: 435243.008 VA
R3: 802.816 VA
R4: 14450.688 VA
```

Netlist Syntax used for this program:

- First Line is for the name of the image to be uploaded. Example-”circuit_1.jpg” and the name must be in small letters.(optional)
- Rest of the letters in netlist must be in caps lock.
- “END” will be the end of the netlist input and also has to be in caps lock.
- Numbers must be used to name nodes. Numbers should increase from 1,2,3….n by a difference from 1. No numbers can be skipped.
- Syntax is very similar to the one used for PSPICE.

Discussion:

Here the steady state results were found by simulating the netlist. In DC state the value of inductance was taken as zero (acting as a short circuit) and capacitor as infinity(open circuit). Power dissipation, current through each element and node voltages were calculated and printed in the output. The left hand side of the GUI is for input of the netlist while the right hand side is for output.

Future Prospects/Improvements:

Only steady state results were found. For DC circuit analysis transient expressions can also be found. Plots can also be made for the transient analysis and AC analysis by using matlab canvas.

The connection of nodes can be visualized using breadth first search algorithm and then the graph plotted. Schematic can be generated using code from the netlist input and then printed in the canvas.