MATLAB GUI Application for Teaching Electronics

Ali H. Assi, Maitha H. Al Shamisi and Hassan A. N. Hejase
United Arab Emirates University
United Arab Emirates

1. Introduction

The Electrical Engineering (EE) Department at the University Arab Emirates (UAE) University incorporates numerous software tools in teaching the diverse electrical engineering (EE) courses imparted at undergraduate and graduate level. The most common being the use of MATLAB, Simulink and LabView, in addition to standard circuits, electronics, and power systems software packages such as OrCAD, MultiSim and PSCAD. Instructors also make use of free available online JAVA applets that apply to specific advanced EE courses such as signals and systems, electromagnetics, antenna engineering, among others. The variety of tools used in each EE course makes it difficult for students to learn a new tool or program for each course. This suggests that MATLAB can be used as a common platform for all courses given its rich library and available tools. Student evaluations over the past years have reflected favourably on the use of MATLAB tools as a valuable support in graphical visualization, numerical evaluation and modelling tasks in the diverse EE course. Most books published nowadays in the various EE subjects include MATLAB exercises and applications in each chapter. The use of these software tools is intended to enhance student appreciation of theoretical concepts and as support tools for hands-on analysis and design experience. Most EE students use the limited MATLAB/Simulink Student Version which does not include many of the needed MATLAB toolboxes. As a result students have to work on campus in order to access the specialized toolboxes. Developing GUI-based applets offers the advantage of providing more independent MATLAB-based tools for use by students on their own Laptop anywhere.
Numerous educators have been developing software applets in different electrical engineering subjects. Such tools are indispensable in helping students better understand basic scientific and engineering concepts through a user-friendly interactive environment that also counts with an adequate help menu to guide students through the application. (Azemi & Stook, 1996) utilized MATLAB in undergraduate electric circuit courses. They focused on features of MATLAB that have not been adapted by other educators before. They worked on generating analytical solutions with the Symbolic Math toolbox, creating interactive simulations with user interface control, and the use of MATLAB Compiler and MATLAB C Library to produce stand-alone applications. They presented examples illustrating the above mentioned features and made the code available on their website.

They also discussed student response to use of the developed MATLAB software package in circuits analysis. (Andreatos & Michalareas, 2008) developed a MATLAB-based application for e-assessment in an introductory analog electronic design course. The application included separate MATLAB GUI interfaces for students and instructor. The applet was intended to help students design a transistor amplifier and in parallel provide an automated qualitative and quantitative assessment tool for instructors. The added assessment tool aimed at ensuring that students engage in actual circuit evaluation rather than making random guesses. (Andreatos & Zagorianos, 2009) also presented a MATLAB-based GUI tool for teaching Automatic Control Systems. The tool is demonstrated using a step-by-step exercise on a typical aircraft control system. (Attia, 1995, 1996) designed AC circuits and electronics teaching tools using MATLAB to teach circuit theory, filter design, random processes, control system and communication theory. The tools employ matrix functions for experimental data analysis as well as graphical features to display the frequency response of amplifiers and illustrate the principles and concepts of semiconductor physics. The circuits MATLAB exercises cover sinusoidal ac analysis, network characteristics and frequency response. The interactive programming and versatile graphics of MATLAB are especially effective in exploring some of the characteristics of devices and electronic circuits. (Rajashekar & Bovik, 2000) presented a suite of user-friendly interactive Digital Signal Processing (DSP) demonstration modules using MATLAB. Their focus was on providing visualization tools that emphasize the intuitive aspects of DSP algorithms. A MATLAB/GUI based educational tool was developed by (Koç & Aydoğmus, 2009) for power system fault calculations. This software provides a user-friendly interface to help the student understand the symmetrical components and fault calculations. The tool allows students to choose one of four fault options for which fault current and voltage calculations are performed. The GUI provides a graphical output representation of currents and voltages. For such application, the instructor expects students to check their answers with hand calculations.

The EE Department at UAE University offers two circuits courses, namely, Electric Circuits I (ELEC 320) and Electric Circuits II (ELEC 325). The Electric Circuits I course runs through both semesters of the academic year. It focuses on the analysis of basic DC and AC electric circuits. Among the topics covered in this introductory course are operational amplifier (OP-AMP) circuits. The average student population each year ranges from 25 to 30 students. At the beginning of the course, students have background knowledge of basic mathematics, physics, and MATLAB programming skills needed throughout the course. One of the most significant course design objectives is the development of a tool for achieving improved learning process. Today, during the teaching process of the fundamentals principles of Electronics, the emphasis is not given on tedious calculations, but rather on offering engineering education, by utilizing efficient software tools. Computer-aided applications are appropriate tools, because they improve the efficiency of learning. In this chapter, basic electronic and electric circuits are investigated using an interactive MATLAB GUI program applet (MATLAB, 2010). The developed comprehensive and user-friendly tool called Electronics Teaching Assistant (abbreviated herein as ETA) can perform typical operational amplifier (OP-AMP) gain calculations and displays analog graphs for input and output currents and voltages in a user friendly MATLAB environment.

Section 2 will address the operational amplifier and basic configurations used in teaching OP-AMP circuits. Section 3 briefly discusses the voltage and current divider circuits. Section 4 explains in detail the development of the GUI tool including code used and input and output windows. Conclusions are then presented in Section 5.

The authors intend to develop more GUI-based applets for numerous circuits and electronics subjects with the objective of making student learning of basic electrical topics fun and interactive. Proper assessment of student learning is followed each semester in line with the ABET outcome assessment process.

2. Operational amplifiers

Operational amplifiers (op amps) typically have 2 inputs, a positive (non-inverting) input and a negative (inverting) input. A signal fed into the positive (non-inverting) input will produce an output signal which is in phase with the input. If the signal is fed into the negative (inverting) input, the output will be 180 degrees out of phase when compared to the input.
The following sub-sections represent an attempt to give you the basic understanding of OP-AMP configurations. None of the power supply connections are shown. Most OP-AMP circuits used in audio applications use a ±15 volts power supply. They can also be used with a single ended supply (no negative voltage).
The diagram below (Fig. 1) shows the OP-AMP symbol.

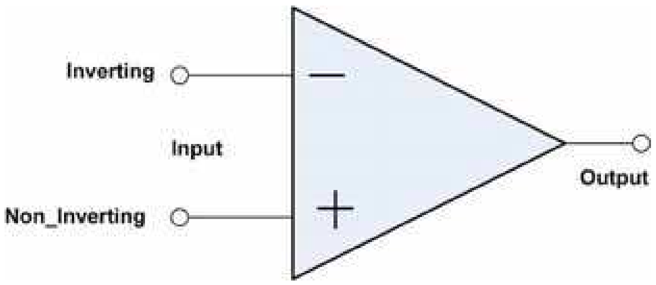


Fig. 1. The OP-AMP symbol

2.1 Inverting amplifier

This is a fundamental OP-AMP configuration whose schematic diagram depicted in Fig. 2 shows the basic circuit configuration.

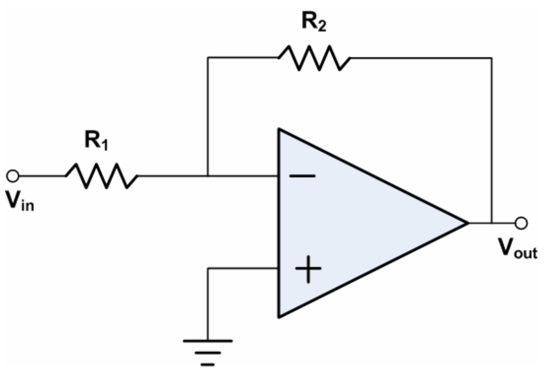


Fig. 2. The inverting configuration

An input voltage, V_{in} is applied to the input resistor, R_1. The OP-AMP amplifies the input voltage it receives and inverts its polarity, producing an output voltage, V_{out}. This same output voltage is also applied to a feedback resistor, R_2, which is connected to the amplifier input along with R_1.
The OP-AMP itself has a very high voltage gain. As a result, the junction of the two resistors, which is also the OP-AMP input, must be virtually at ground potential. A non-zero input voltage will be amplified so that the output voltage would try to exceed its electronic limits. At the same time, the OP-AMP requires an extremely small input current to operate. Therefore, the input current (Vin/R_1) must be the same as the feedback current (V_{out}/R_2). This implies that the effective gain of the circuit with feedback in place is simply the resistance ratio, R_2/R_1. With such configuration, we can obtain accurate results if we use precision resistors, and yielding a gain of:

$$Gain = \frac{V_{out}}{V_{in}} = \frac{R_2}{R_1} \tag{1}$$

2.2 Non-inverting amplifier

Fig. 3 shows a non-inverting OP-AMP circuit. In this circuit, the input signal is effectively used as the reference voltage at the "+" input, while the "-" input is indirectly referenced to ground. In order to keep the two input voltages the same, the OP-AMP must set V_{out} to whatever voltage is required to make the feedback voltage to the "-" input match the input voltage to the "+" input.

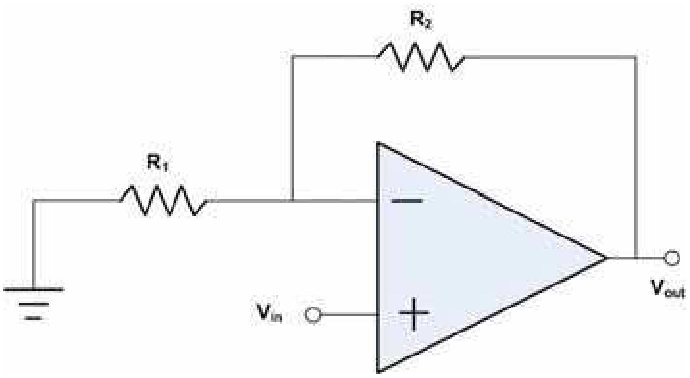


Fig. 3. The non-inverting configuration

Since R_2 and R_1 form a voltage divider, the feedback voltage will be:

$$V_{out} \times \frac{R_1}{R_2 + R_1} \tag{2}$$

The gain of this circuit becomes:

$$Gain = \frac{V_{out}}{V_{in}} = 1 + \frac{R_2}{R_1} \tag{3}$$

2.3 Voltage follower

This is a special case of the non-inverting amplifier with $R_1 = \infty$ and $R_2 = 0$. Fig. 4 shows the voltage follower circuit.

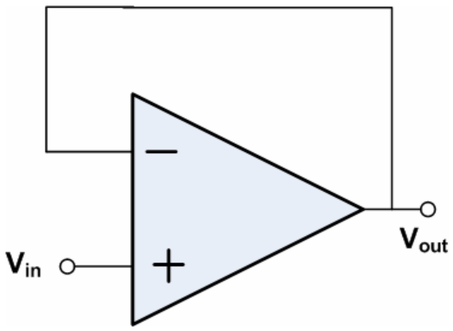


Fig. 4. The voltage follower

For the circuit shown in Fig. 4, and using equation (3), one can easily find:

$$V_{out} = V_{in} \tag{4}$$

This is a very useful circuit, because the input impedance of the OP-AMP is very high, giving effective isolation of the output from the signal source. The circuit draws very little power from the signal source, avoiding "loading" effects. This circuit in general is a useful first stage. The voltage follower is often used for the construction of buffers for logic circuits.

2.4 Summing amplifier

This is special case of the inverting configuration with more than one input as shown in Fig. 5.

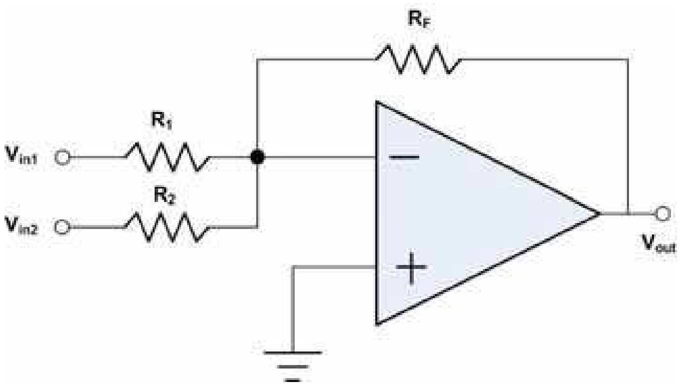


Fig. 5. The summing amplifier

This circuit will amplify each individual input voltage and produce an output voltage signal that is proportional to the algebraic "SUM" of the two individual input voltages V_{in1} and V_{in2}. We can also add more inputs if required. The point of using an OP-AMP to add

multiple input signals is to avoid interaction between them, so that any change in one input voltage will not have any effect on the other input. This is because the input signals are effectively isolated from each other by the "virtual earth" node at the inverting input of the OP-AMP.
For the circuit shown in Fig. 5, the voltage at the output is given by:

$$V_{out} = -R_F\left(\frac{V_1}{R_1} + \frac{V_2}{R_2}\right) \tag{5}$$

A direct voltage addition can also be obtained when all the resistances are of equal value (i.e. $R_F = R_1 = R_2$):

$$V_{out} = -(V_1 + V_2) \tag{6}$$

2.5 Differential amplifier

The circuit of a differential amplifier is shown in Fig. 6. Apply the superposition principle to obtain the gain expression:

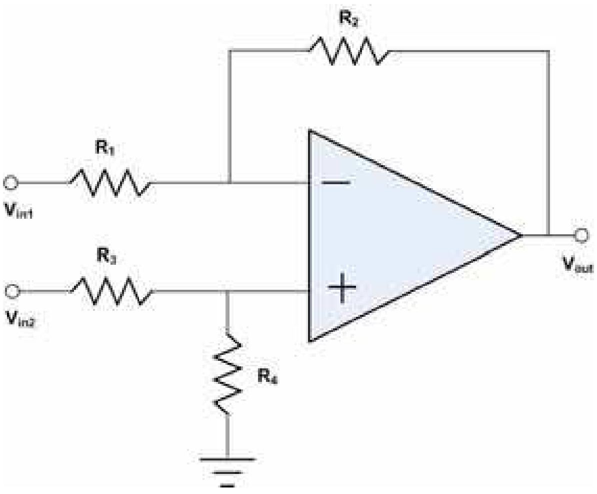


Fig. 6. The differential amplifier

$$V_{out} = -\frac{R_2}{R_1}V_{in1} + \frac{R_4}{R_3 + R_4}\left(1 + \frac{R_2}{R_1}\right)V_{in2} \tag{7}$$

For $R_2 = R_4$, the output will be:

$$V_{out} = -\frac{R_2}{R_1(V_{in2} - V_{in1})} \tag{8}$$

Finally, for $R_2 = R_1$ one can obtain the exact difference of V_{in2} and V_{in1}:

$$V_{out} = V_{in2} - V_{in1} \tag{9}$$

3. Basic electric circuits

This section explains briefly two basic circuit configurations of general use in electronic circuit analysis.

3.1 Voltage divider

The two resistor voltage divider, shown in Fig. 7, is used often to supply a voltage different from that of an available battery or power supply. In practice, the output voltage depends upon the resistance of the load it drives. Note here that R_2 includes also the load resistance.

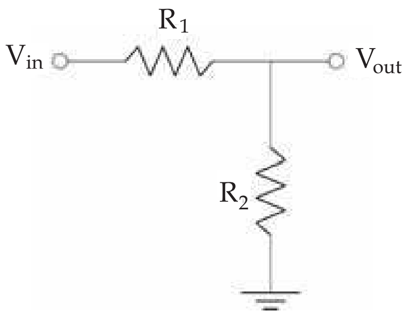


Fig. 7. The voltage divider

$$V_{out} = V_{in} \times \frac{R_2}{R_2 + R_1} \tag{10}$$

3.1 Current divider

For the circuit shown in Fig. 8, one can easily derive the following relation:

$$I_1 = I_3 \times \frac{R_2}{R_2 + R_1} \tag{11}$$

and

$$I_2 = I_3 \times \frac{R_1}{R_2 + R_1} \tag{12}$$

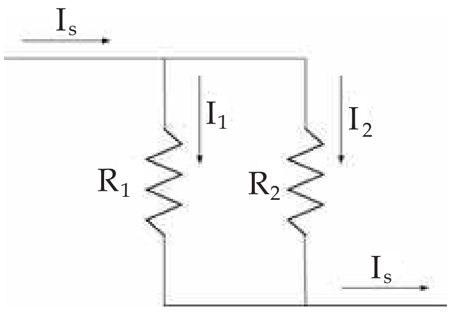


Fig. 8. The current divider

4. Graphical MATLAB-based tool

A graphical user interface tool was designed using the Matlab **GUIDE** environment which greatly simplifies the process of building and developing GUIs. **GUIDE Layout Editor** allows the user to populate a GUI by clicking and dragging GUI components namely, axes, panels, buttons, text fields, sliders into the layout area. Moreover, from the Layout Editor, the user can size the GUI, modify component look and feel, align components, set tab order, view a hierarchical list of the component objects, and set GUI options.
GUIDE automatically generates a program file containing MATLAB functions that controls how the GUI operates. This code file helps initialize the GUI and contains a framework for the GUI callbacks; the routines that execute when a user interacts with a GUI component. The MATLAB Editor should be used to add code to callbacks in order to perform the required actions [MATLAB Creating Graphical User Interfaces, 2004].

4.1 GUI layout and programming

The main window (Electronics Teaching Assistant) is designed to allow the user choose between Operational Amplifier circuits and Electric Circuits and exit the tool as shown in Fig. 9. It consists of two Axes, text and three push buttons namely, **OP AMP Circuits**, **Electric Circuits** and **Close**. The two axes are used for presenting images: one for logo and the other for background. The text displays the tool's name. OP AMP Circuits button will allow the user to analyze different types of OP AMP Circuits. Electric Circuits button will let the user analyze different types of electric circuits (voltage and current dividers). Close button will simply close the whole program.
The following code blocks show how the three buttons are programmed. The set function **set(handle, 'PropertyName', value)** is used to set a property value of buttons.

```
% --- Programming theOP_AMP_Circuits_Button.
functionOP_AMP_Circuits_Button_Callback(hObject, eventdata, handles)
% hObject=handle to OP_AMP_Circuits_Button (see GCBO)
% eventdata= reserved to be defined in a future version of MATLAB
% handles = structure with handles and user data (see GUIDATA)
%---- To open Electric Circuits window ---------%
set(ETA_OP_AMP_Circuits,'Visible','on')
 %---- To Close Electronics_Teaching_Assistant window ------%
set(Electronics_Teaching_Assistant,'Visible','off')
functionElectronic_Circuit_Button_Callback(hObject, eventdata, handles)
%---- To open Electric Circuits window ---------%
set(ETA_Electric_Circuits,'Visible','on')
%---- To close main window window ------------%
set(Electronics_Teaching_Assistant,'Visible','off')
functionClose_Button_Callback(hObject, eventdata, handles)
%---- To terminate the program ---------%
delete(get(0,'Children'));
```

In order to show the logo and background images and their axes, the code is written under **Opening Function**. **Axes function** is used to determine which axes the image should display followed by **imshowfunction**.

functionElectronics_Teaching_Assistant_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output arguments. 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 Electronics_Teaching_Assistant (see VARARGIN)
% Choose default command line output for Electronics_Teaching_Assistant
handles.output = hObject;
%------------- Add logo---------------------------%
axes(handles.Axes_UAEU_Logo);
imshow('uaeu_logo.png');
%------------- Add Background --------------------%
axes(handles.Axes_Background)
imshow('Electronic_Circuit.jpg');
% Update handles structure
guidata(hObject, handles);

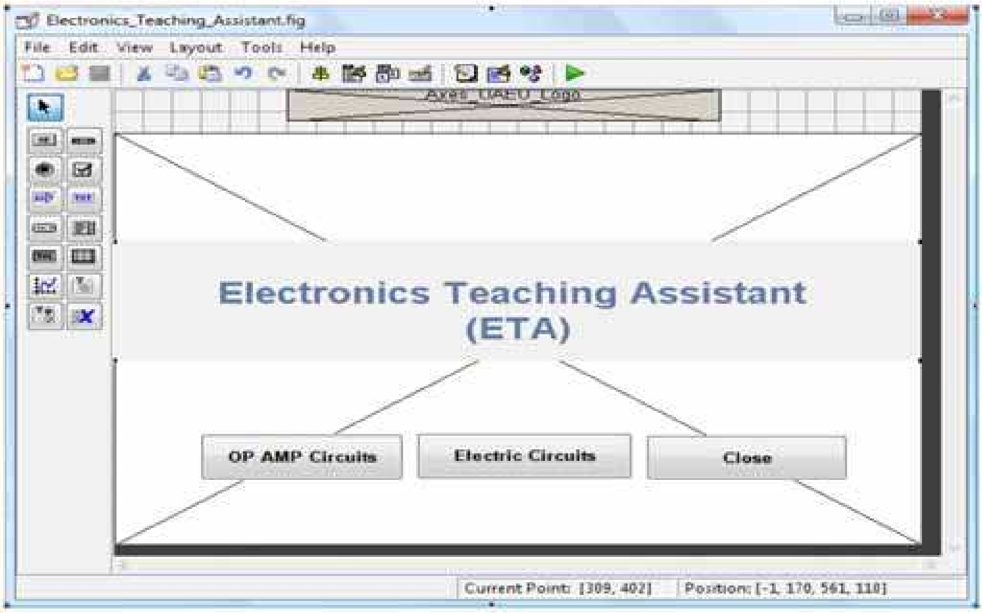

Electronics_Teaching_Assistant.fig
File Edit View Layout Tools Help
Axes_UAEU_Logo
Electronics Teaching Assistant (ETA)
OP AMP Circuits
Electric Circuits
Close
Current Point: [309, 402]
Position: [-1, 170, 561, 110]

Fig. 9. Electronics Teaching Assistant

The ETA_Op_AMP_Circuits window shown in Fig. 10 is designed to allow the user choose between different types of circuits through a pop-up menu. The user can also visualize the circuit diagram when updated according to the user choices. This diagram is presented on Circuit Axes which is located on the top right hand. Input and output parameters vary according to circuit types. All circuit components are first laid out then their values are defined. For example, the R_1 and R_3 text located in the background of R_1 has its visibility property set to off. Once the user selects the differential amplifier circuit, the text becomes visible and R_1 text will become invisible.The relationship between input and output voltage (i.e. the gain) is plotted on Vin_Vout_Axes which is located underneath Circuit Axes. Three push buttons are presented namely; **Calculate**, **Reset** and **Main**. The calculate button computes gain(s) and plots input and output voltages. The Reset button clears the contents of input and output texts and axes. The Main button opens the main window and closes current window.

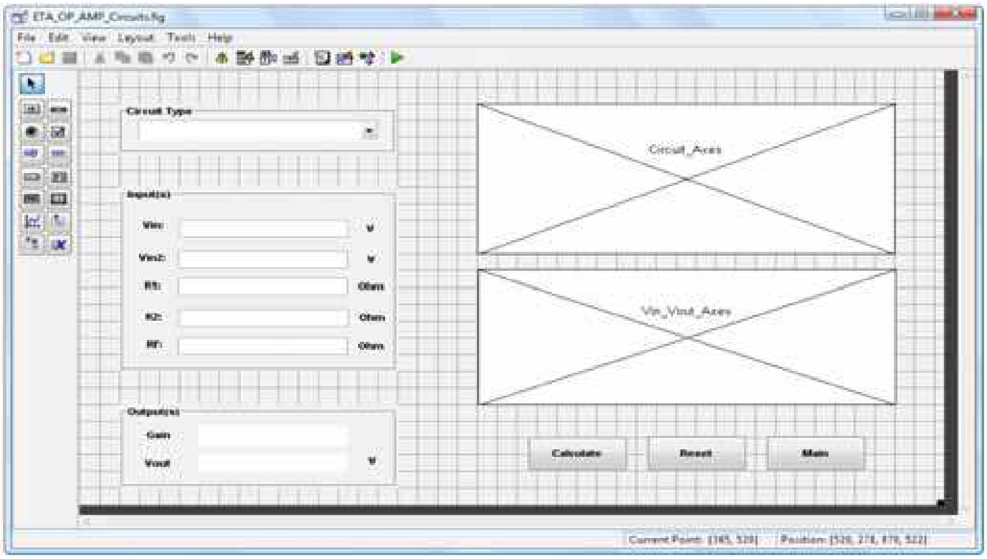


Fig. 10. ETA_OP_AMP_Circuits GUI Layout

The program code of pop-upmenu and Calculate_Buttoncall functions for case 4 (Summing Amplifier) are listed below.

```
functionCircuit_Type_Popupmenu_Callback(hObject, eventdata, handles)
% hObject handle theCircuit_Type_Popupmenu (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints:
% contents = get(hObject,'String') returns Circuit_Type_Popupmenu contents as cell array
% contents{get(hObject,'Value')} returns selected item from Circuit_Type_Popupmenu
switch get(handles.Circuit_Type_Popupmenu,'Value')
        %------------------ Summing Amplifier-------------------%
case 4
axes(handles.Circuit_Axes);
imshow('Summation.png');
set(handles.Gain_Text,'Visible','off');
set(handles.Gain_Out_Text,'Visible','off');
        %------------------ IF follower was chosen---------------%
        %----------------- Extra inputs -------------------------%
```

```
set(handles.Vin_2_Text,'Visible','on'); %Vin 2 label invisible
set(handles.R1_Edit,'Visible','on');
set(handles.R2_Edit,'Visible','on');
%************** R1 & R2 *****************%
set(handles.R1_Text,'Visible','on');
set(handles.R1_Edit,'Visible','on');
set(handles.R1_Unit_Text,'Visible','on');
set(handles.R2_Text,'Visible','on');
set(handles.R2_Edit,'Visible','on');
set(handles.R2_Unit_Text,'Visible','on');
        %----------------- Extra inputs -------------------------%
set(handles.Vin_2_Text,'Visible','on'); %Vin 2 label visiable
set(handles.Vin_2_Edit,'Visible','on'); %Vin 2 Edit text visiable
set(handles.V_Text,'Visible','on');   %Measure Unit of Vin2 visiable
set(handles.RF_Text,'Visible','on');   % Rf text visible
set(handles.R_FEdit,'Visible','on');  % Rf Edit visible
set(handles.RF_Unit_Text,'Visible','on');  % Rf unit text visible
set(handles.R1_R3_Text,'Visible','off');
set(handles.R2_R4_Text,'Visible','off');
        %------------------ Clear All the Edit Text -------------------------%
set(handles.Vin_Edit,'string', ' ')
set(handles.Vin_2_Edit,'string', ' ')
set(handles.R1_Edit,'string', ' ')
set(handles.R2_Edit,'string', ' ')
set(handles.R_FEdit,'string', ' ')
set(handles.Gain_Out_Text,'string', ' ')
set(handles.Vout_Out_Text,'string', ' ')
        %------------- Clear Vin-Vout axes before plotting, clear previous plot ---------------%
axes(handles.Vin_Vout_Axes)
cla
    functionCalculate_Button_Callback(hObject, eventdata, handles)
    % hObject  handle theCalculate_Button (see GCBO)
    % eventdata reserved - to be defined in a future version of MATLAB
    % handles  structure with handles and user data (see GUIDATA)
    %------------- Clear Vin-Vout axes before plotting, clear previous plot ---------------%
    axes(handles.Vin_Vout_Axes)
    cla%
    %---------------------- Basic inputs--------------------------------%
    Vin   = str2double(get(handles.Vin_Edit, 'string'))% Input voltage
    R1   = str2double(get(handles.R1_Edit, 'String'));  % input Resistor1
    R2   = str2double(get(handles.R2_Edit, 'String'));  % input Resistor2
    %--------------------------------------------------------------------%
    %Menu List :
    % 1 :Default setting
    % 2 : Inverting Amplifier
    % 3 : Non- Inverting Amplifier
    % 4 : Voltage Follower
    % 5 : Summing Amplifier
    % 6 : Differential Amplifier
    %--------------------------------------------------------------------%
    switch get(handles.Circuit_Type_Popupmenu,'Value')
    %---------------------------- Summing Amplifier---------------------------%
    case 4
          Vin2  = str2double(get(handles.Vin_2_Edit, 'string'))  % Input voltage 2
        Rf  = str2double(get(handles.R_FEdit, 'string'));  % input Resistor 3
        %--------------Summing Equation -------------%
    Gain_Result   = -Rf/R1  % Gain 1
    Gain_Result_2 = -Rf/R2  % Gain 2
        %-------output voltage---------%
        Vout_1 = Gain_Result * Vin
        Vout_2 = Gain_Result_2 * Vin2
    Vout = Vout_1+ Vout_2;
    set(handles.Vout_Out_Text,'String',num2str(Vout))
        n = 1; % one cycle
    t = 0 :pi/8 : 2*n*pi        % time domain
    Vin_Plot  = Vin * sin(t)
    Vout_Plot = Vout * sin(t)
    if (abs (Vout) <= abs(Power_Supply)) % Check if clipping problem is occurred
    plot(t, Vin_Plot,'RED' ,'linewidth',2)
    grid on
    axis([ 0 max(t) -20  20])
    hold on
    plot(t,  Vout_Plot,'GREEN' ,'linewidth',2)
    grid on
    xlabel('Time (t)','fontweight','bold')
    ylabel('Input - Output Voltage (V)','fontweight','bold')
    legend('Vin', 'Vout');
    else
    axes(handles.Vin_Vout_Axes)
    Vin_Plot  = Vin * sin(t)
    Vout_Plot = Vout * sin(t)
    for i = 1 : length(Vout_Plot)
    if (Vout_Plot(i)>Power_Supply)
    Vout_Plot(i)= Power_Supply
    elseif (Vout_Plot(i) < -Power_Supply)
    Vout_Plot(i)= -Power_Supply
    end
    end
    plot(t, Vin_Plot,'RED' ,'linewidth',2)
    grid on
    axis([ 0 max(t) -20  20])
    hold on
    plot(t,  Vout_Plot,'GREEN' ,'linewidth',2)
    grid on
```

```
xlabel('Time (t)','fontweight','bold')
ylabel('Input - Output Voltage (V)','fontweight','bold')
legend('Vin', 'Vout');
end
    %------------------------ Display warning message (Clipping)-------------%
if abs(Vout) > 15
msgboxText{1} =  'Clipping!';
msgbox(msgboxText,'Clipping Phenomena', 'warn');
end
% --- ExecuteReset_Button.
functionReset_Button_Callback(hObject, eventdata, handles)
% hObject    handle to Reset_Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%---------- Clear the contents----------------%
set(handles.Vin_Edit,'string', ' ')
set(handles.Vin_2_Edit,'string', ' ')
set(handles.R1_Edit,'string', ' ')
set(handles.R2_Edit,'string', ' ')
set(handles.R_FEdit,'string', ' ')
set(handles.Gain_Out_Text,'string', ' ')
set(handles.Vout_Out_Text,'string', ' ')
set(handles.Circuit_Type_Popupmenu, 'value', 1)%popup menu go to default
axes(handles.Circuit_Axes);
imshow('White_Background.jpg');
axes(handles.Vin_Vout_Axes)
cla % Clear current axis
% --- Execute Main_Button.
functionMain_Button_Callback(hObject, eventdata, handles)
% hObject    handle theMain_Button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%---- To open Main Window window ------%
set(Electronics_Teaching_Assistant,'Visible','on')
%---- To close Main Window window ------%
set(ETA_OP_AMP_Circuits,'Visible','off')
```

The Electric Circuits window, shown in Fig. 11, is designed to allow users analyze voltage and current dividers. This window consists of a pop-up menu where the user can choose between voltage and current dividers. The Electric Circuit Axes is updated accordingly. The inputs and outputs are varied between the different circuit types, with all components laid out first then their values specified. Three buttons are used namely; Calculate, Reset and Main, to perform the following functions: compute voltage and current, clear input and output text, and navigate to education window, respectively.

The codes shown below define the pop-up menu of the callback function and calculated push button callback function for case 3 (Current Divider).

```
functionElectric_Circuit_Popupmenu_Callback(hObject, eventdata, handles)
```

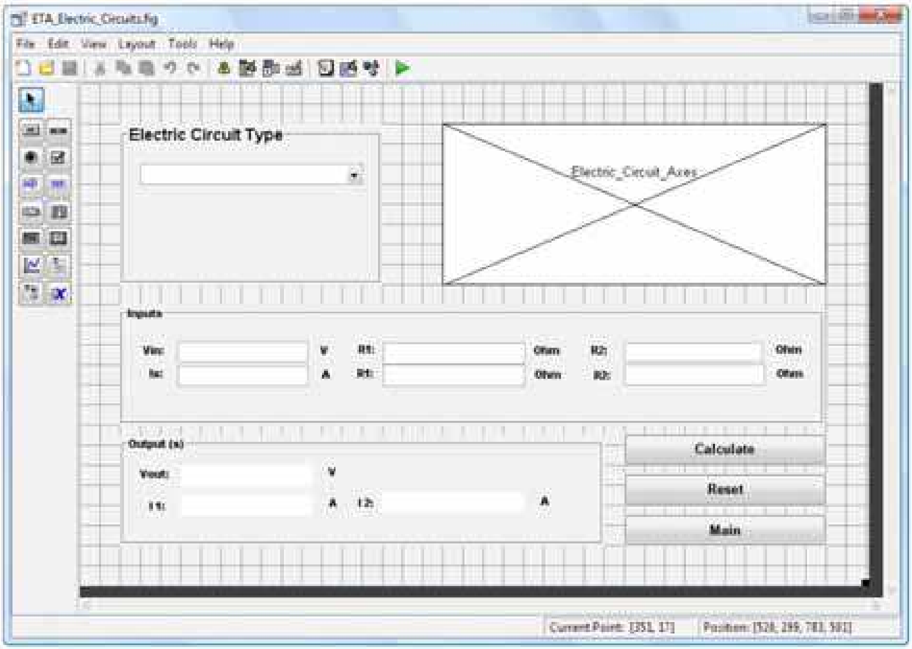


Fig. 11. ETA_Electric_Circuits GUI Layout

```
% hObject= handle theElectric_Circuit_Popupmenu (see GCBO)
% eventdata= reserved to be defined in a future version of MATLAB
% handles = structure with handles and user data (see GUIDATA)
switch get(handles.Electric_Circuit_Popupmenu,'Value')
%---------------- current divider ----------------------%
case 3
axes(handles.Electric_Circuit_Axes);
imshow('Current_Divider.png');
      %----------- Current divider inputs are visible------------%
set(handles.Is_Text,'Visible','on');
set(handles.Is_Edit,'Visible','on');
set(handles.Is_Unit_Text,'Visible','on');
set(handles.R1_Current_Edit,'Visible','on');
set(handles.R1_Current_text,'Visible','on');
set(handles.R1_Unit_Text,'Visible','on');
set(handles.IR2_Text,'Visible','on');
set(handles.R2_Current_Edit,'Visible','on');
set(handles.R2_Unit_Text,'Visible','on');
    %----------- Current divider outputs are visible------------%
set(handles.I1_Text,'Visible','on');
set(handles.I_1_Edit,'Visible','on');
set(handles.I1_Unit_Text,'Visible','on');
set(handles.I2_Text,'Visible','on');
set(handles.I_2_Edit,'Visible','on');
set(handles.I2_Unit_Text,'Visible','on');
    %----------- Voltage divider inputs are invisible------------%
set(handles.Vin_Text,'Visible','off');
set(handles.Vin_Edit,'Visible','off');
set(handles.Vin_Unit_Text,'Visible','off');
set(handles.R1_Text,'Visible','off');
set(handles.R1_Voltage_Edit,'Visible','off');
set(handles.Voltage_R1_Unit_Text,'Visible','off');
set(handles.Rb_Text,'Visible','off');
set(handles.R2_Voltage_Edit,'Visible','off');
set(handles.Rb_Unit_Text,'Visible','off');
     %----------- Voltage divider output is invisible------------%
set(handles.Vout_Text,'Visible','off');
set(handles.Vout_Edit,'Visible','off');
set(handles.Vout_Unit_Text,'Visible','off');
end
functionCalculate_Button_Callback(hObject, eventdata, handles)
% hObject= handle theCalculate_Button (see GCBO)
% eventdata=reserved to be defined in a future version of MATLAB
% handles =  structure with handles and user data (see GUIDATA)
switch get(handles.Electric_Circuit_Popupmenu,'Value')
case 3
     Is = str2double(get(handles.Is_Edit, 'string'))% Input current
     R1 = str2double(get(handles.R1_Current_Edit,  'String'));   % input Resistor
     R2 = str2double(get(handles.R2_Current_Edit,  'String'));   % current drain
     I1 = (R2/(R1+R2)) * Is
     I2 = (R1/(R1+R2)) * Is
set(handles.I_1_Edit,'Visible','on');
set(handles.I_2_Edit,'Visible','on');
set(handles.I_1_Edit,'String',num2str(I1))
set(handles.I_2_Edit,'String',num2str(I2))
end
```

4.2 Running the GUI

When the program is running, the main window appears as shown in Fig. 12. As mentioned earlier, this window allows the user to open the "OP AMP circuits" window by clicking the 'OP AMP Circuits' button. One can also open "Electric Circuits window" by clicking the 'Electric Circuits' button. Finally, the program can be closed by clicking the 'Close' button.
When the 'OP AMP Circuits' button is clicked, the OP_AMP_Circuits window opens while the main window (Education) disappears. The user can choose one of the following basic OP-AMP circuits types: Inverting Amplifier, Non Inverting Amplifier, Summing Amplifier, Voltage Follower and Differential Amplifier (Fig. 13). By selecting the circuit type from the menu; the schematic of the selected circuit will display on the upper axes and the inputs and outputs will change accordingly. The **Calculate**, **Reset**, and **Main** buttons perform functions as described earlier.

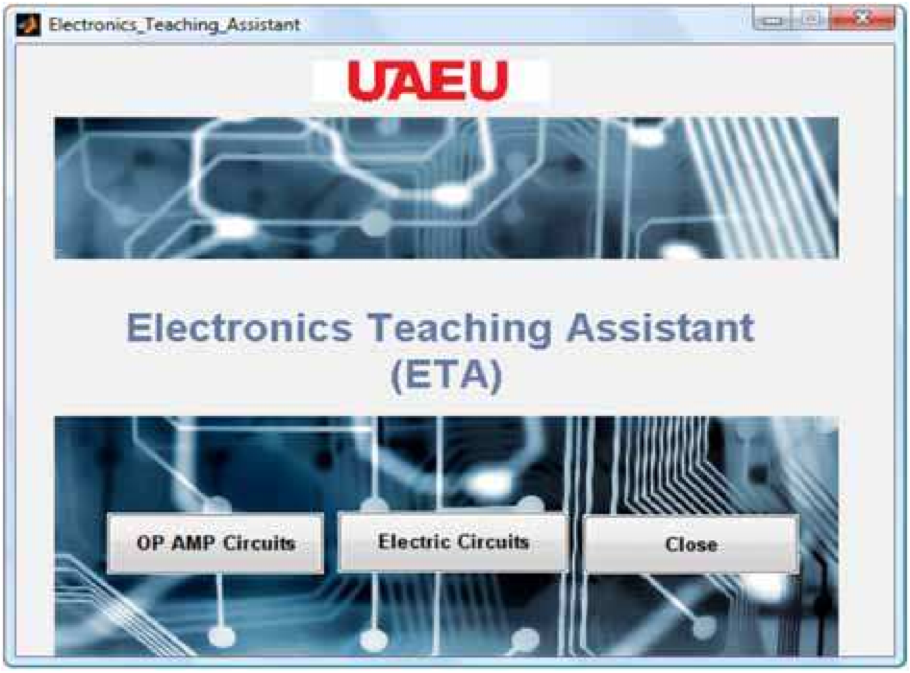

Electronics_Teaching_Assistant
UAEU
Electronics Teaching Assistant
(ETA)
OP AMP Circuits
Electric Circuits
Close

Fig. 12. Running Electronics Teaching Assistant GUI

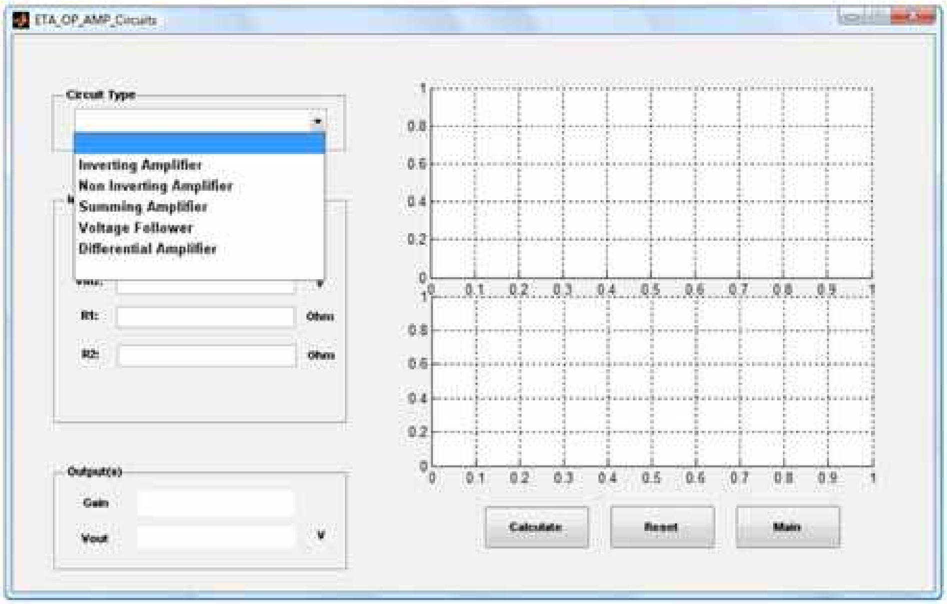
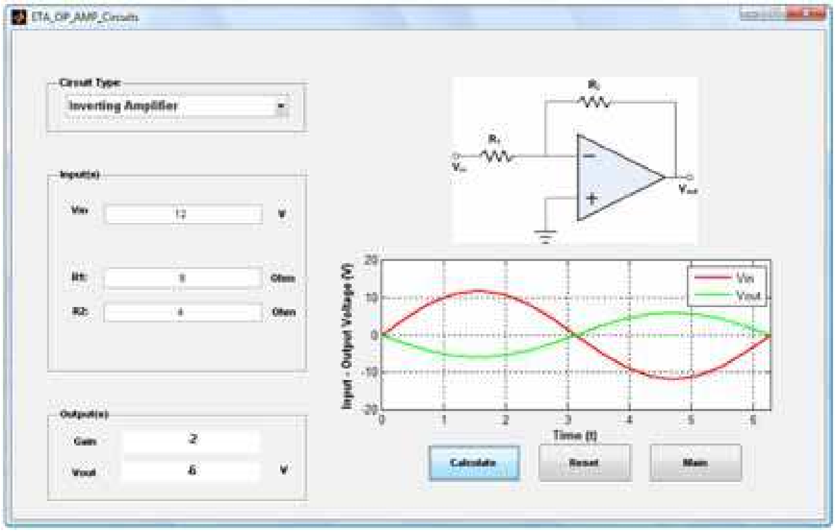


Fig. 13. Running the Operational Amplifier GUI

Figures 14 and 15 show the functions of the inverting and non- inverting amplifier circuits, respectively. The inputs for these circuits are named V_{in} (V), R_1 (Ohm) and R_2 (Ohm). The inputs and outputs are represented as sinusoidal waves and the relationship between them (i.e. gain) can be plotted. The red waveform represents the input of the amplifier while the green waveform represents the output. The value of the gain and V_{out} are displayed numerically.



Circuit Type: Inverting Amplifier

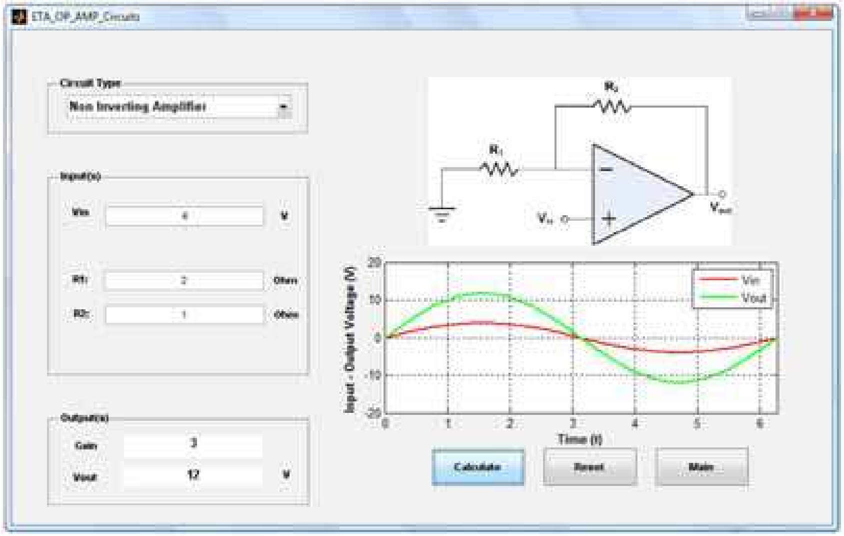
Input(s)

Vin:	12	V
R1:	8	Ohm
R2:	4	Ohm

Output(s)

Gain	2	
Vout	-6	V

Fig. 14. Running the Inverting Amplifier GUI



Circuit Type: Non Inverting Amplifier

Input(s)

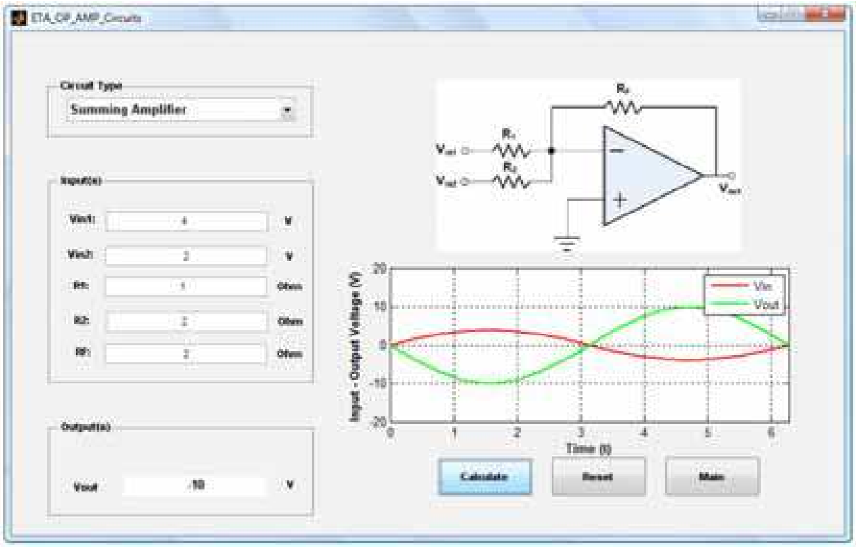
Vin:	4	V
R1:	2	Ohm
R2:	1	Ohm

Output(s)

Gain	3	
Vout	12	V

Fig. 15. Running the Non Inverting Amplifier GUI

The Summing Amplifier function is shown in Fig. 16. The inputs areV_{in1} (V), V_{in2} (V), R_1 (Ohm), R_2 (Ohm) and R_F (Ohm) while the output is V_{out} (V).


ETA_OP_AMP_Circuits

Circuit Type: Summing Amplifier

Input(s)

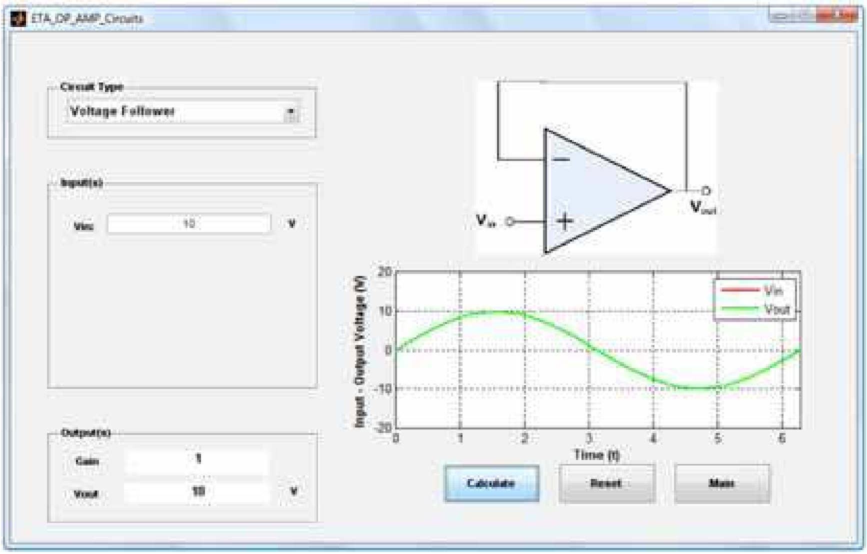
Vin1:	4	V
Vin2:	2	V
R1:	1	Ohm
R2:	2	Ohm
Rf:	2	Ohm

Output(s)

Vout	-10	V

Fig. 16. Running the Summing Amplifier GUI

TheVoltage Follower function is shown in Fig. 17. This circuit has only one input V_{in} (V) with unity gain implying that the input and output values are equal. The V_{in} waveform does not appear because $V_{out} = V_{in}$.


ETA_OP_AMP_Circuits

Circuit Type: Voltage Follower

Input(s)

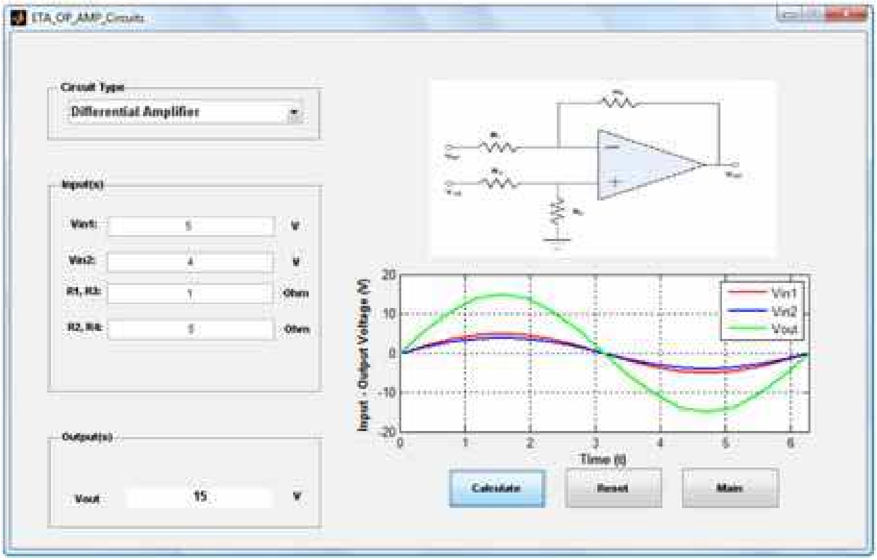
Vin:	10	V

Output(s)

Gain	1	
Vout	10	V

Fig. 17. Running the Voltage Follower GUI

The Differential Amplifier function is shown in Fig. 18. Parameters V_{in1}, V_{in2}, (R_1, R_3), and (R_2, R_4) are the inputs of the differential amplifier circuit, while V_{out} is the output.


ETA_OP_AMP_Circuits

Circuit Type: Differential Amplifier

Input(s)

Vin1: 5 V

Vin2: 4 V

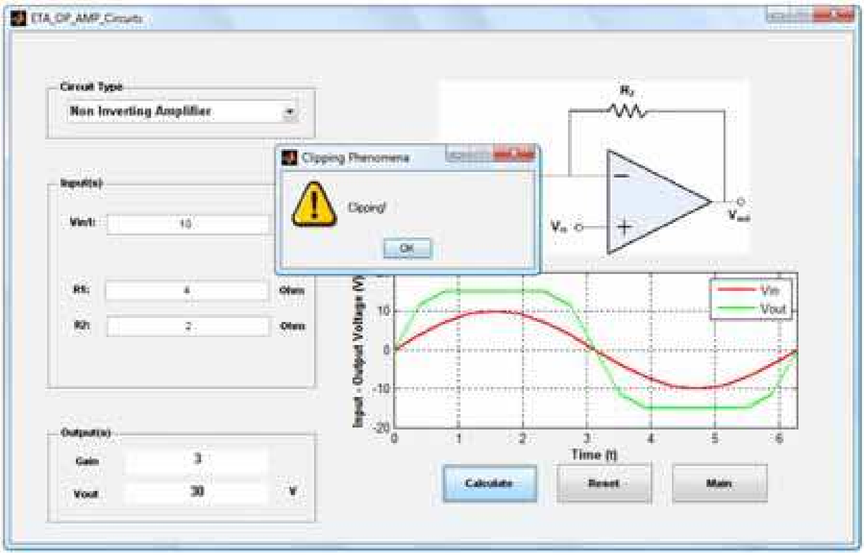
R1, R3: 1 Ohm

R2, R4: 5 Ohm

Output(s)

Vout 15 V

Fig. 18. Running the Differential Amplifier GUI


ETA_OP_AMP_Circuits

Circuit Type: Non Inverting Amplifier

Input(s)

Vin1: 10

R1: 4 Ohm

R2: 2 Ohm

Output(s)

Gain 3

Vout 30 V

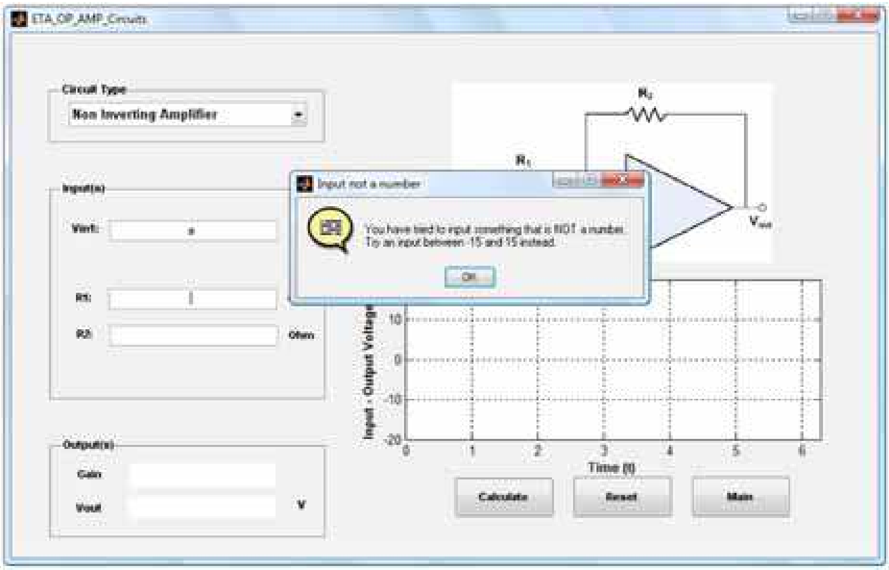
Clipping Phenomena

Clipping!

OK

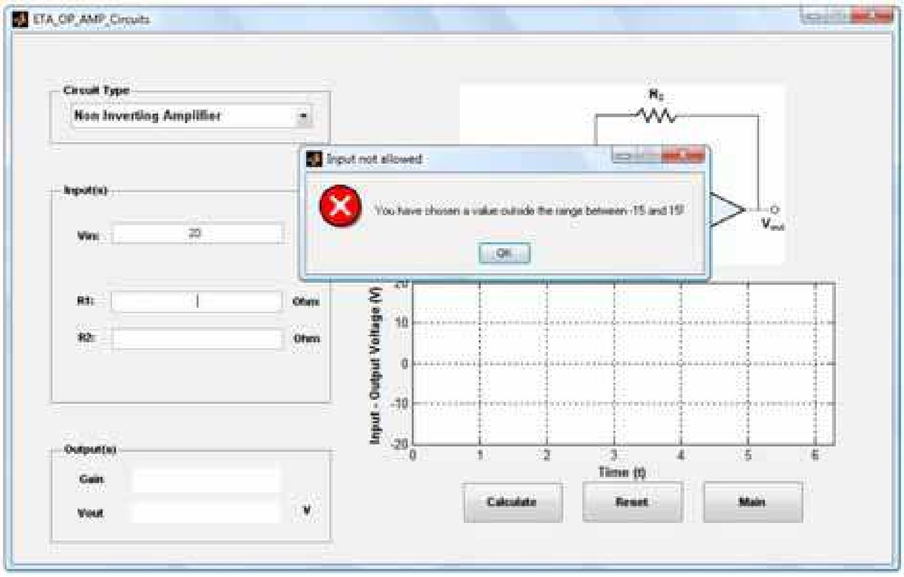
Fig. 19. Clipping Phenomena Warning Message

This ETA program has the ability to check if the clipping phenomena is occurring and notifies the user by displaying a warning message as illustrated in Fig. 19. Moreover, it checks user inputs; if the user unintentionally enters a non-numeric value, the error message will be shown as Fig. 20. The error message will be display if the user attempts to enter a voltage input that exceeds power supply voltage values (-15 V, 15V) as shown in Fig. 21. In case the user enters maximum input voltage, a warning message will be shown as shown in Fig. 22.


Input not a number

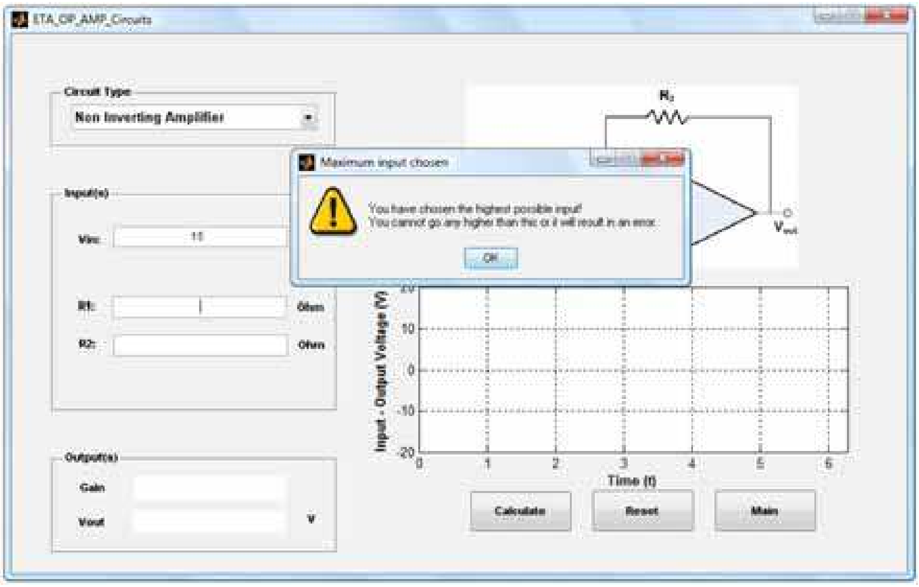
You have tried to input something that is NOT a number.
Try an input between -15 and 15 instead.

Fig. 20. The Input is not a Number Warning Message


Input not allowed

You have chosen a value outside the range between -15 and 15!

Fig. 21. The Input is out of the Range Error Message



Maximum input chosen

You have chosen the highest possible input!
You cannot go any higher than this or it will result in an error.

Fig. 22. The Maximum input chosen Warning Message

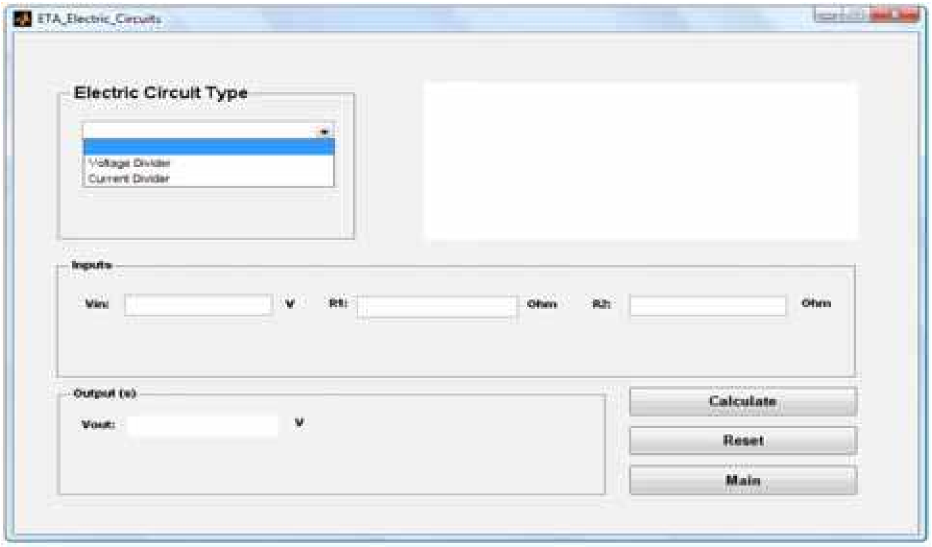
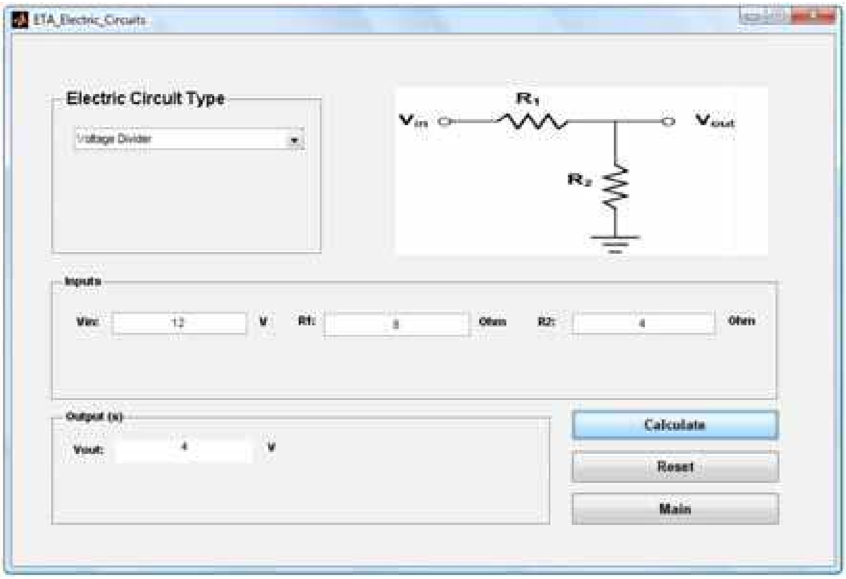


Fig. 23. Running the Electric Circuit GUI

When the user clicks the 'Electric Circuits' button, the ETA_Electric_Circuits window opens while the main window (Electronics Teaching Assistant) disappears. For this version of the ETA tool, the user can choose the electric circuit type (i.e. Voltage divider or Current divider) from the menu as shown in Fig. 23. By selecting the circuit from the menu; the schematic of the selected circuit will be shown on the upper axes and the inputs and outputs will change accordingly. The **Calculate**, **Reset**, and **Main** buttons perform functions as described earlier.

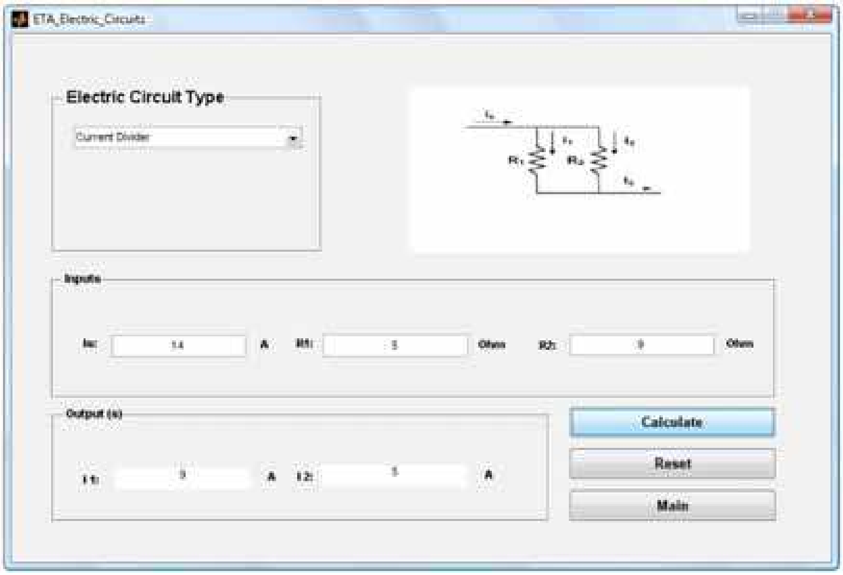


Voltage Divider

Vin: 12 V R1: 8 Ohm R2: 4 Ohm

Vout: 4 V

Fig. 24. Runningthe Voltage Divider GUI



Current Divider

Iin: 14 A R1: 5 Ohm R2: 9 Ohm

I 1: 9 A I 2: 5 A

Fig. 25. Running the Voltage Divider GUI

Fig 24 shows the function of the basic voltage divider circuit. When the voltage divider circuit is selected from the menu, the circuit's schematic is displayed on axes. The inputs of this circuit are: Vin (V), R1 (Ohm) and R2 (Ohm).The Calculate, Reset, and Main buttons perform functions as described earlier.
Fig. 25 shows the function of the current divider circuit. When the current divider circuit is selected from the menu, the circuit's schematic is displayed on axes. The inputs of this circuit are: Is (A), R1 (Ohm) and R2 (Ohm). The values of I1 and I2 are displayed. The Calculate, Reset, and Main buttons perform functions as described earlier.

5. Conclusion

This chapter presented a user-friendly interactive MATLAB-based GUI tool for teaching basic electrical OP-AMP circuits. The programming code for the GUI tool development was also addressed in addition to explanation of input and output parameters needed for different types of OP-AMPs. In addition, MATLAB has a rich collection of mathematical functions and tools to compute and visualize data for different circuit applications. Students can use OrCAD PSpice to compare with the developed applet results. The interactive and friendly nature of MATLAB and its immediate graphing tools are indispensable for helping electrical engineering students achieve a better understanding of basic concepts and principles of semiconductor fundamentals and other electrical topics.

6. References

Andreatos A.S. & Michalareas G. (2008).Engineering education e-assessment with Matlab; Case study in electronic design, *Proceedings of the 5th WSEAS /IASME International Conference on ENGINEERING EDUCATION (EE'08)*, pp. 172-177, Heraklion, Greece, July 22-24, 2008.

Andreatos A.S. & Zagorianos A. (2009).Matlab GUI Application for Teaching Control Systems, *Proceedings of the 6th WSEAS International Conference on ENGINEERING EDUCATION*, pp. 208-211, Rodos (Rhodes) Island, Greece, July 22-24, 2009.

AttiaJ. O. (1995). Teaching AC circuit analysis with MATLAB.*Proceedings of the 25th Frontiers in EducationConference*, Vol. 1, pp.2c6.9-2c612, Atlanta, GA, USA, November 01- 04, 1995.

AttiaJ. O. (1996).Teaching Electronics with MATLAB. *Proceedings of the 26th Frontiers in Education Conference*, Vol. 2, pp. 609-611, Salt Lake City, UT, USA, November 06- 09, 1996.

Azemi A. & Stook C. (1996). Utilizing MATLAB in undergraduate electric circuits courses, *Proceedings of the 26th Annual Conference Frontiers in Education Conference (FIE '96)*, Vol. 2, pp. 599-602, Salt Lake City, Utah, USA, November 06-09, 1996.

Azemi A. & Yaz E. (1994). PSpice and MATLAB in Undergraduate and Graduate Electrical Engineering Courses. *Proceedings of the 24th Frontiers in Education Conference*, pp. 456-459, San Jose, CA, USA, November 02-06, 1994.

Koç S. & AydoğmusZ. (2009).A MATLAB/GUI Based Fault Simulation Tool for Power System Education, *Mathematical and Computational Applications*, Vol. 14, No. 3, pp. 207-217.

MATLAB (2010). MATLAB Creating Graphical User Interfaces, revision of March 2010 for MATLAB 7.10 (Release 2010a), The MathWorks Inc., Natick, MA, USA.

Rajashekar U. &Bovik A.C. (2000).Interactive DSP education using MATLAB demos, *IEEE SignalProcessing Education Workshop*, Hunt, Texas, USA, October 15-18, 2000.