HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF TELECOMMUNICATIONS ENGINEERING
_____ * _____

ANALOG SIGNAL PROCESSING
PROJECT REPORT
MATLAB SOLVER PROGRAM FOR OPERATION AMPLIFIER CIRCUITS

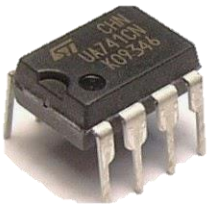
Lecturer: Assoc. Prof. Hà Hoàng Kha
Class: TT04
Semester: 211
Members: Lương Triển Thắng 2051194
Nguyễn Ngọc Minh Anh 2051033
Phạm Nguyễn Trung Tín 2051203

Hồ Chí Minh City – 7th November, 2021

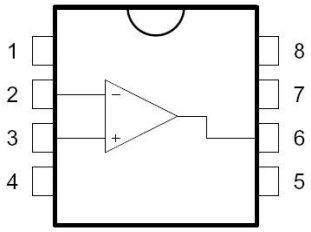
Table of Contents

I. Introduction to OpAmp

1. A brief look at OpAmp



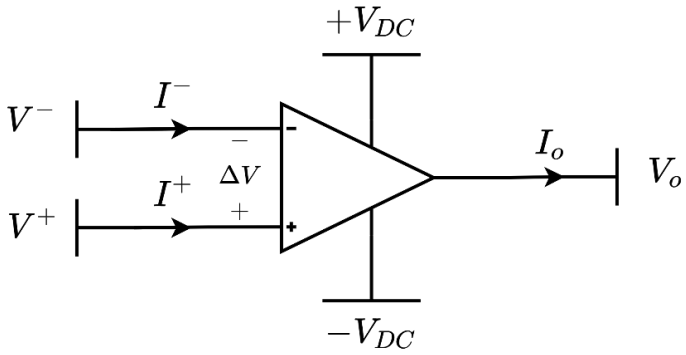
UA741CN IC



UA741CN Pinout

Figure I-1: UA741CN IC and pinout

Operational Amplifiers, also known as Op-amps or OpAmps, are basically a voltage amplifying device designed to be used with components like capacitors and resistors, between its in and out terminals. They are essentially a core part of analog devices. OpAmp has positive and negative inputs which allow circuits that use feedback to achieve a wide range of functions. Feedback components like these are used to determine the operation of the amplifier. The amplifier can perform many different operations (resistive, capacitive, or both), giving it the name *Operational Amplifier*.



Terminology
$\boldsymbol{V^+}$: non-inverting input voltage
$\boldsymbol{V^-}$: inverting input voltage
$\boldsymbol{V_o}$: output voltage
$\boldsymbol{I_o}$: non-inverting input voltage
$\boldsymbol{I^+}$: non-inverting input current
$\boldsymbol{I^-}$: inverting input current
$\pm\boldsymbol{V_{DC}}$: positive and negative DC supply voltages used to power the OpAmp ($\pm$5V~$\pm$30V)
$\boldsymbol{\Delta V} = V^{\pm}V^{-}$: difference voltage

Figure I-2: OpAmp Schematic

Using OpAmp, it's easy to make amplifiers, comparators, log amps, filters, oscillators, data converters, level translators, references, and more. Mathematical functions like addition, subtraction, multiplication, and integration can be easily accomplished.

The operational amplifier is arguably the most useful single device in analog electronic circuitry. With only a handful of external components, it can be made to perform a wide variety of analog signal processing tasks. It is also quite affordable, most general-purpose amplifiers selling for under a dollar apiece. Modern designs have been engineered with durability in mind as well: several “op-amps” are manufactured that can sustain direct short-circuits on their outputs without damage.

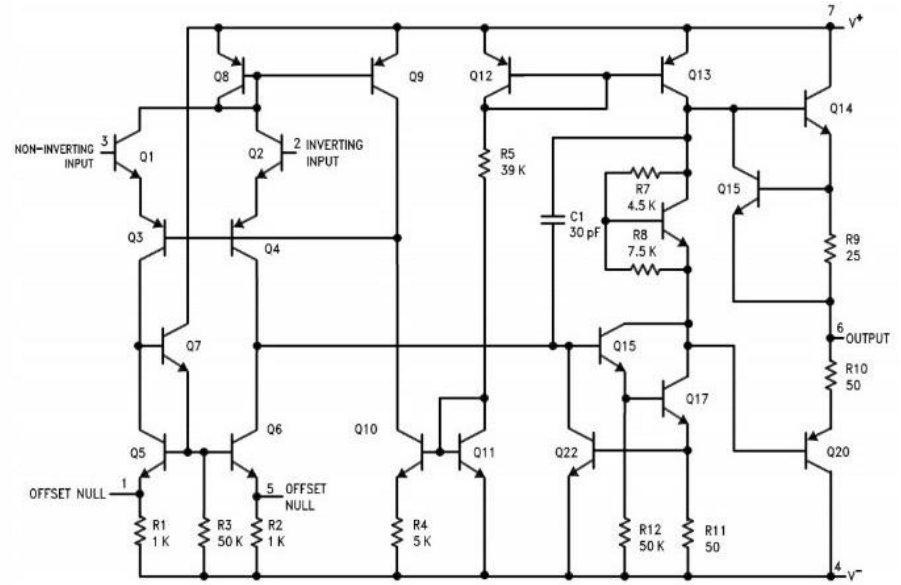


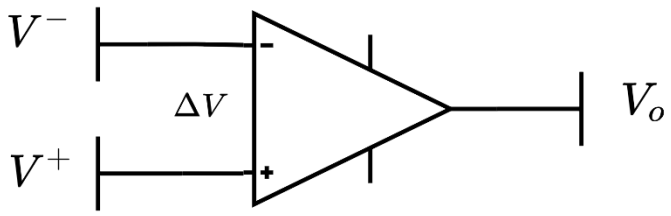
Figure I-3: Schematic of LM741 from its datasheet of Texas Instruments

One key to the usefulness of these little circuits is in the engineering principle of feedback, particularly negative feedback, which constitutes the foundation of almost all automatic control processes. Its numerous practical applications include instrumentation amplifiers, digital-to-analog converters, analog computers, level shifters, filters, calibration circuits, inverters, summers, integrators, differentiators, subtractors, logarithmic amplifiers, comparators, gyrators, oscillators, rectifiers, regulators, voltage-to-current converters, current-to-voltage converters, and clippers.

2. Open-loop vs. closed-loop operation

a. Open-loop

- Rarely used.
- OpAmp specification may be important.



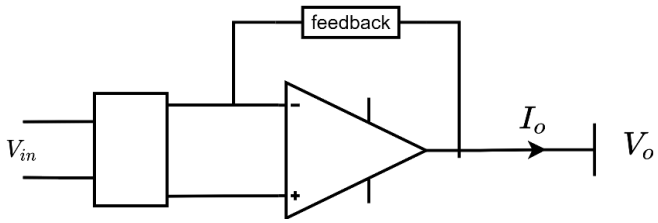
Properties
$\boldsymbol{\Delta V}$: differential input voltage
$\boldsymbol{A_{OL}}$: open-loop gain
(typical value: $A_{OL} = 100{,}000$)
In general:

$$V_o = A_{OL} \text{ or } A_{OL} = \frac{V_o}{\Delta V}$$

Figure I-4: Open-loop circuit and its properties

b. Closed-loop

- Most used.
- Some sort of feedback from output to input exists.
- The input voltage V_{in} is defined according to the application.



Properties

$\boldsymbol{A_{CL}} = \frac{V_o}{V_{in}}$: closed-loop gain

Figure I-5: Closed-loop circuit and its properties

II. General Knowledge for several ideal OpAmp circuits

The ideal op-amp is an amplifier with infinite input impedance, infinite open-loop gain, zero out-put impedance, infinite bandwidth, and zero noise. It has positive and negative inputs which allow circuits that use feedback to achieve a wide range of functions.

Assumptions for ideal OpAmp:

- Assume that negative feedback: $V^+ = V^-$.
- Assume the input(s) resistance is finite, so $I^+ = I^- = 0$.

1. Inverting amplifier circuit

An Inverting Amplifier is a type of Operational Amplifier circuit which produces an output which is out of phase with respect to its input by 180°.

This mean that if the input pulse is positive, then the output pulse will be negative and vice versa.

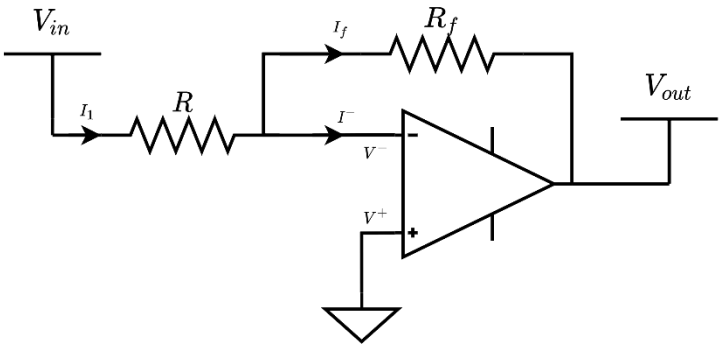


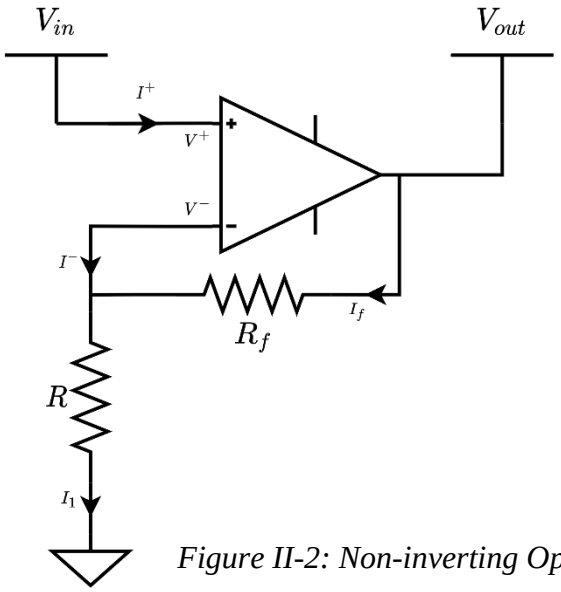
Figure II-1: Inverting OpAmp Circuit

- Since V^+ is connected to the ground, then $V^+ = V^- = 0$
- Apply KCL:

$$I_1 = I^- + I_f$$
$$\Leftrightarrow I_1 = I_f$$
$$\Leftrightarrow \frac{V_{in} - V^-}{R} = \frac{V^- - V_{out}}{R_f}$$
$$\Leftrightarrow \frac{V_{in}}{R} = \frac{-V_{out}}{R_f}$$
$$\Rightarrow \boxed{V_{out} = -\frac{R_f}{R} V_{in}}$$

2. Non-inverting amplifier circuit

A non-inverting Amplifier is an OpAmp circuit configuration that produces an amplified output signal which is in phase with the applied input signal. In other words, a non-inverting amplifier behaves like a voltage follower circuit. A non-inverting amplifier also uses a negative feedback connection, but instead of feeding the entire output signal to the input, only a part of the output signal voltage is fed back as input to the inverting input terminal of the OpAmp.



Apply KCL:

$$I_1 = I^- + I_f$$
$$\Leftrightarrow I_1 = I_f$$
$$\Leftrightarrow \frac{V^-}{R} = \frac{V_{out} - V^-}{R_f}$$
$$\Rightarrow \boxed{V_{out} = \left(1 + \frac{R_f}{R}\right) V_{in}}$$

Figure II-2: Non-inverting OpAmp Circuit

3. Summing inverting amplifier circuit

A summing amplifier is an example of an inverting amplifier with multiple inputs, enabling to effectively add several individual input signals, which proves to be useful in audio mixing applications. Its output is the sum of each input scaled by the corresponding resistor ratio.

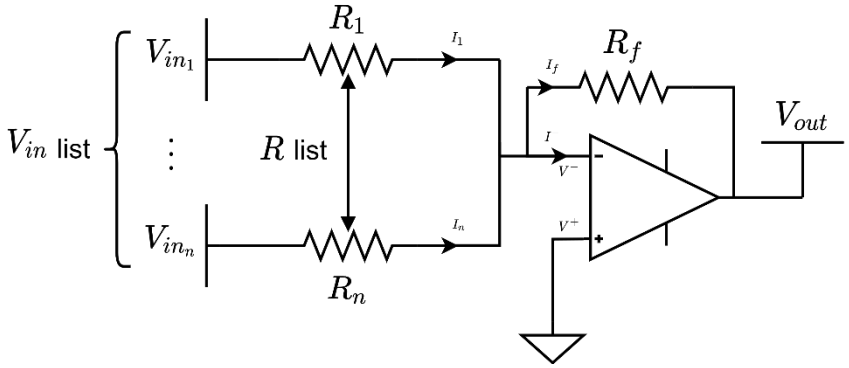


Figure II-3: Summing Inverting OpAmp Circuit

- Since V^+ is connected to the ground, then $V^+ = V^- = 0$
- Apply KCL:

$$I_1 + I_2 + \cdots + I_n = I^- + I_f$$
$$\Leftrightarrow I_1 + I_2 + \cdots + I_n = I_f$$
$$\Leftrightarrow \frac{V_{in_1} - V^-}{R_1} + \frac{V_{in_2} - V^-}{R_2} + \cdots + \frac{V_{in_n} - V^-}{R_n} = \frac{V^- - V_{out}}{R_f}$$
$$\Leftrightarrow \frac{V_{in_1}}{R_1} + \frac{V_{in_2}}{R_2} + \cdots + \frac{V_{in_n}}{R_n} = \frac{-V_{out}}{R_f}$$
$$\Rightarrow \boxed{V_{out} = -\left(\frac{V_{in_1}}{R_1} + \frac{V_{in_2}}{R_2} + \cdots + \frac{V_{in_n}}{R_n}\right) R_f}$$

4. Summing non-inverting amplifier circuit

The non-inverting summing amplifier is based on around the configuration of a non-inverting operational amplifier circuit in that the input (either DC or AC) is applied to the non-inverting $(+)$ terminal, while the required negative feedback and gain is achieved by feeding back some portion of the output signal (V_{out}) to the inverting $(-)$ terminal.

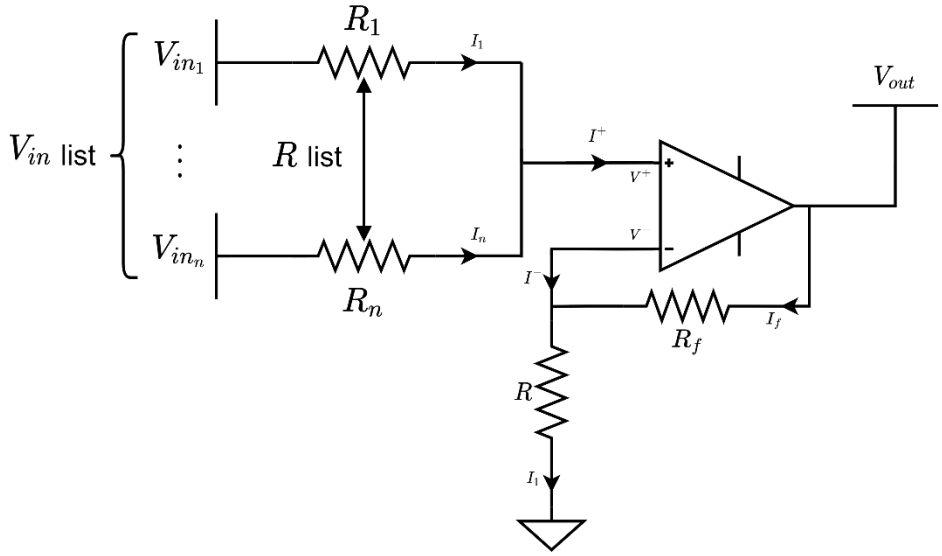


Figure II-4: Summing Non-inverting OpAmp Circuit

Apply KCL:

$$I_1 + I_2 + \cdots + I_n = I^+ = 0$$

$$\Leftrightarrow \frac{V_{in_1} - V^+}{R_1} + \frac{V_{in_2} - V^+}{R_2} + \cdots + \frac{V_{in_n} - V^+}{R_n} = 0$$

$$\Leftrightarrow \boxed{V^+ = V^- = \frac{\frac{V_{in_1}}{R_1} + \frac{V_{in_2}}{R_2} + \cdots + \frac{V_{in_n}}{R_n}}{\frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n}}}$$

$$I = I^- + I_f$$

$$\Leftrightarrow I = I_f$$

$$\Leftrightarrow \frac{V^-}{R} = \frac{V_{out} - V^-}{R_f}$$

$$\Rightarrow \boxed{V_{out} = \left(1 + \frac{R_f}{R}\right) V^-}$$

$$\Leftrightarrow \boxed{V_{out} = \left(1 + \frac{R_f}{R}\right) \frac{\frac{V_{in_1}}{R_1} + \frac{V_{in_2}}{R_2} + \cdots + \frac{V_{in_n}}{R_n}}{\frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n}}}$$

III. MATLAB Solving Program

1. Features and instructions

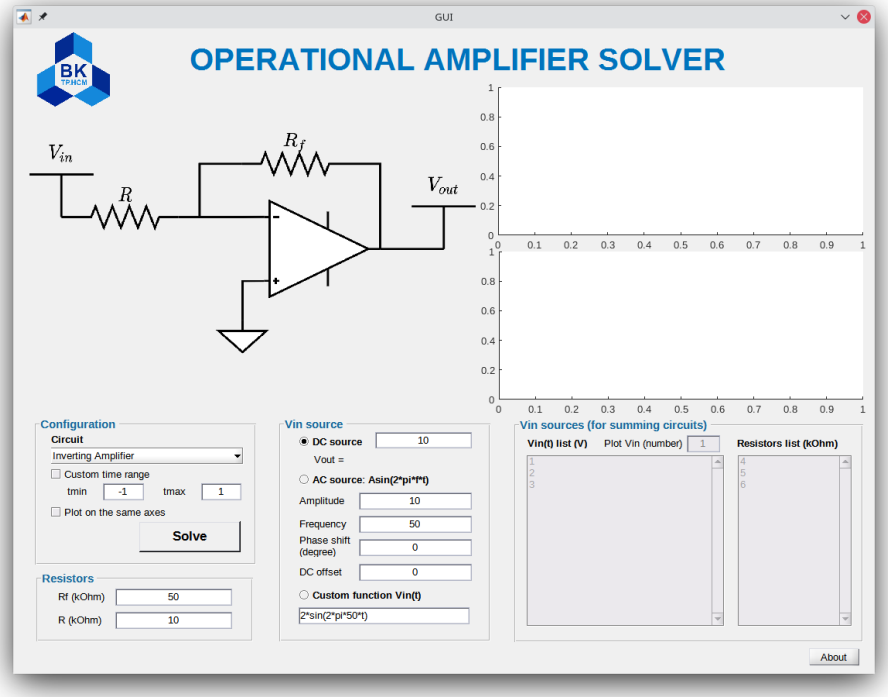


Figure III-1: GUI for Operational Amplifier Solver

a. Features

- Includes four different circuits:
  + Inverting Amplifier
  + Non-inverting Amplifier
  + Summing Inverting Amplifier
  + Summing Non-inverting Amplifier
- Three different types of source signals:
  + DC source
  + AC source
  + Custom function $V_{in}(t)$
- For Summing circuits
  + Input can be maximum of 100 sources.
  + Plot all input signal on the same axes.
- Time domain can be adjusted for the graphs.
- Input and output can be plotted on a single axes.
- Resistor (potentiometer) can be a function that changes over time (must not be negative).
- All inputs accept basic operations, e.g., $+,\ -,\ *,\ /,$ ^, abs, mod, log, ...

b. Instructions

- **Step 1:** Choose the circuit.
- **Step 2:**
  + If non-summing circuit is chosen:
    Choose the source in "Vin source" panel then fill the value.
  + If summing circuit is chosen:
    Fill in the value of the source and resistors in "Vin sources (for summing circuits)" panel in the exact order.
    Note: The number of the Vin and the number of resistors must be equal.
- **Step 3:** Fill in the value R_f and R (if needed) in "Resistors" panel.
- **Step 4:** (optional)
  + Custom the time range (tmin, tmax) to plot Vin and Vout in that range only.
  + Tick "Plot on the same axes" to plot Vin and Vout on the same axes.
- **Step 5:** Press the "Solve" button and the results will shown up.[1]

2. Examples

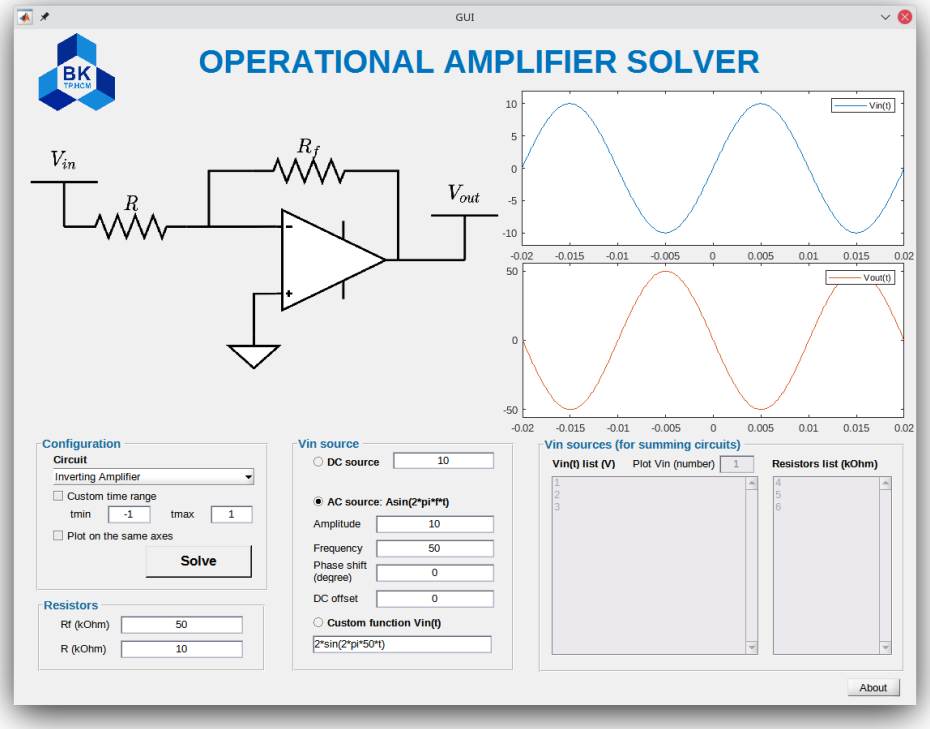


Figure III-2: Inverting Amplifier with AC source

[1] Solving speed depends on your computer specs.

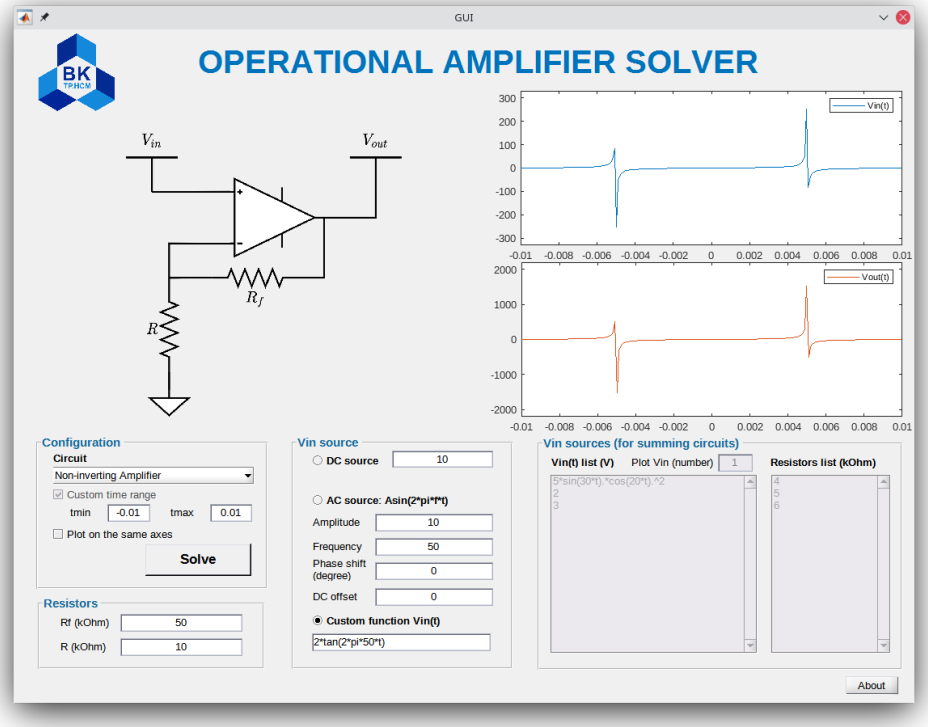


Figure III-3: Non-inverting Amplifier with custom function

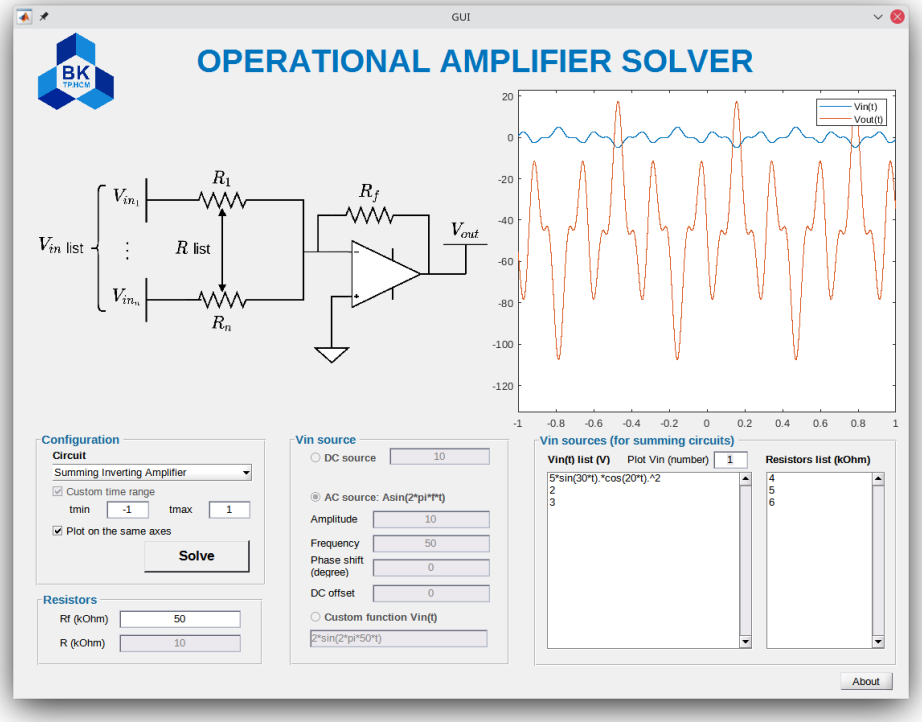


Figure III-4: Summing Inverting Amplifier with a list of 3 Vins, 3 resistors

3. GUI designing and coding

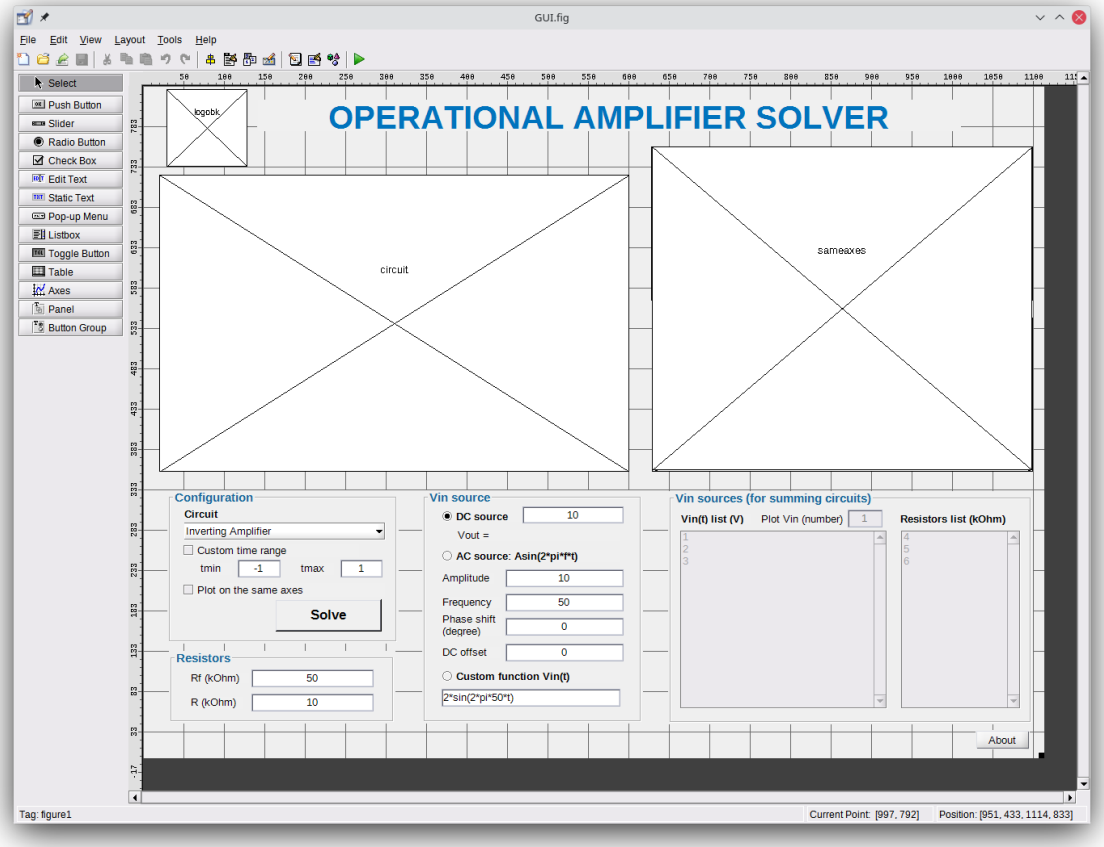


Figure III-5: GUI designing in GUIDE

To achieve “Plot on the same axes” feature, a new axes is put on top of vin and vout graph.

After finish designing the GUI, it’s time for the code.

When looking closely into the code below, the first function is GUI_OpeningFcn which clarifies the initial user interface.

Then, code lines are all “Callback” functions which are the procedures that will be performed when the users interact.

All green code lines are “comments”, are also the notes about how these code’s procedure is employed for and then follow pre-programming sequence. The page will be rotated to better showing the comments.

```
function varargout = GUI(varargin)
% GUI MATLAB code for GUI.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to 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 GUI

% Last Modified by GUIDE v2.5 06-Nov-2021 22:29:48

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @GUI_OpeningFcn, ...
    'gui_OutputFcn',  @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 GUI is made visible.
function GUI_OpeningFcn(hObject, ~, 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 GUI (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);
axes(handles.circuit);
[im, ~, alpha] = imread('invert.png');
f = imshow(im);
set(f, 'AlphaData', alpha);

axes(handles.logobk);
[im, ~, alpha] = imread('logobk.png');
f = imshow(im);
set(f, 'AlphaData', alpha);

% --- Outputs from this function are returned to the command line.
function varargout = GUI_OutputFcn(~, ~, 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;

% % % START HERE !!!

% ---------------------- CONFIGURATION PANEL ----------------------

% CIRCUIT SELECTING MENU
% When changing the circuit
% - Change the circuit diagram (png transparent).
% - Change back to AC source to avoid bugs.
% - If the summing circuit is selected, custom time range must be on.
function circuitselect_Callback(hObject, ~, handles)
axes(handles.circuit);
if(get(hObject,'Value') == 1)
    set(handles.customtimechkbox, 'Enable','on');
    set(handles.rtxtbox, 'Enable','on');
    set(handles.acsourcebtn, 'Value', 1);

    [im, ~, alpha] = imread('invert.png');

    summingon(false,handles);
elseif(get(hObject,'Value') == 2)
    set(handles.customtimechkbox, 'Enable','on');
    set(handles.rtxtbox, 'Enable','on');
    set(handles.acsourcebtn, 'Value', 1);

    [im, ~, alpha] = imread('noninvert.png');

    summingon(false,handles);
elseif(get(hObject,'Value') == 3)
    set(handles.customtimechkbox, 'Value',1);
    set(handles.customtimechkbox, 'Enable','off');
    set(handles.rtxtbox, 'Enable','off');
    set(handles.acsourcebtn, 'Value', 1);

    [im, ~, alpha] = imread('invert_sum.png');

    summingon(true,handles);
elseif(get(hObject,'Value') == 4)
    set(handles.customtimechkbox, 'Value',1);
    set(handles.customtimechkbox, 'Enable','off');
```

```
    set(handles.rtxtbox, 'Enable','on');
    set(handles.acsourcebtn, 'Value', 1);

    [im, ~, alpha] = imread('noninvert_sum.png');

    summingon(true,handles);
end

f = imshow(im);
set(f, 'AlphaData', alpha);

function circuitselect_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% CUSTOM TIME RANGE CHECKBOX
function customtimechkbox_CreateFcn(~, ~, ~)
function customtimechkbox_Callback(~, ~, ~)

% TMIN TEXTBOX

% FEATURES:
% - Can use some simple mathematical operation, eg. +, -, *, /, ^, ...
% - Avoid some unexpected input, eg. text

% IDEAS:
% - Create of a list of text that need to be ignore.
% - If one or more ignore text contained in the text:
%    + Try to evaluate it.
%    + If the evaluation failed, set the text to '0'.
% - If there is no ignore text:
%    + Try to convert it into (double).
%    + If the conversion failed, set the text to '0'.
function tmintxtbox_Callback(hObject, ~, handles)
value = get(hObject, 'String');
ignore = ["+","-","*","/","pi","^","exp","log","abs","mod"];
if(contains(value,ignore))
    try
        value = eval(value);
        set(handles.tmintxtbox, 'String', string(value));
    catch ME
        set(handles.tmintxtbox, 'String', '0');
    end
else
    value = str2double(value);
    if(isnan(value))
        set(handles.tmintxtbox, 'String', '0');
    end
end

function tmintxtbox_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% TMAX TEXTBOX

% FEATURES:
% - Can use some simple mathematical operation, eg. +, -, *, /, ^, ...
```

```
% - Avoid some unexpected input, eg. text

% IDEAS:
% - Create of a list of text that need to be ignore.
% - If one or more ignore text contained in the text:
%    + Try to evaluate it.
%    + If the evaluation failed, set the text to '0'.
% - If there is no ignore text:
%    + Try to convert it into (double).
%    + If the conversion failed, set the text to '0'.
function tmaxtxtbox_Callback(hObject, ~, handles)
value = get(hObject, 'String');
ignore = ["+","-","*","/","pi","^","exp","log","abs","mod"];
if(contains(value,ignore))
    try
        value = eval(value);
        set(handles.tmaxtxtbox, 'String', string(value));
    catch ME
        set(handles.tmaxtxtbox, 'String', '0');
    end
else
    value = str2double(value);
    if(isnan(value))
        set(handles.tmaxtxtbox, 'String', '0');
    end
end

function tmaxtxtbox_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% PLOT ON THE SAME AXES CHECKBOX
function sameaxeschkbox_Callback(~, ~, ~)

% ---------------------- RESISTORS PANEL ----------------------

% RF TEXTBOX

% FEATURES:
% - Can use some simple mathematical operation, eg. +, -, *, /, ^, ...
% - Avoid some unexpected input, eg. text (except "t")

% IDEAS:
% - Create of a list of text that need to be ignore.
% - If one or more ignore text contained in the text:
%    + Try to evaluate it.
function rftxtbox_Callback(hObject, ~, handles)
value = get(hObject, 'String');
ignore = ["+","-","*","/","pi","^","exp","log","abs","mod"];
if(contains(value,ignore))
    try
        value = eval(value);
        set(handles.rftxtbox, 'String', string(value));
    catch ME

    end
end

function rftxtbox_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end

% R TEXTBOX

% FEATURES:
% - Can use some simple mathematical operation, eg. +, -, *, /, ^, ...
% - Avoid some unexpected input, eg. text (except "t")

% IDEAS:
% - Create of a list of text that need to be ignore.
% - If one or more ignore text contained in the text:
%    + Try to evaluate it.
function rtxtbox_Callback(hObject, ~, handles)
value = get(hObject, 'String');
ignore = ["+","-","*","/","pi","^","exp","log","abs","mod"];
if(contains(value,ignore))
    try
        value = eval(value);
        set(handles.rtxtbox, 'String', string(value));
    catch ME

    end
end

function rtxtbox_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% ---------------------- VIN SOURCE PANEL ----------------------

% DC SOURCE TEXTBOX

% FEATURES:
% - Can use some simple mathematical operation, eg. +, -, *, /, ^, ...
% - Avoid some unexpected input, eg. text

% IDEAS:
% - Create of a list of text that need to be ignore.
% - If one or more ignore text contained in the text:
%    + Try to evaluate it.
%    + If the evaluation failed, set the text to '0'.
% - If there is no ignore text:
%    + Try to convert it into (double).
%    + If the conversion failed, set the text to '0'.
function dcsourcetxtbox_Callback(hObject, ~, handles)
value = get(hObject, 'String');
ignore = ["+","-","*","/","pi","^","exp","log","abs","mod"];
if(contains(value,ignore))
    try
        value = eval(value);
        set(handles.dcsourcetxtbox, 'String', string(value));
    catch ME
        set(handles.dcsourcetxtbox, 'String', '0');
    end
else
    value = str2double(value);
    if(isnan(value))
        set(handles.dcsourcetxtbox, 'String', '0');
    end
```

```
end

function dcsourcetxtbox_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% DC SOURCE RADIO BUTTON
% If selected, custom time range must be using and does not allow turning
% it off and vice versa.
function dcsourcebtn_Callback(hObject, ~, handles)
if(get(hObject,'Value') == 1)
    set(handles.customtimechkbox, 'Value',1)
    set(handles.customtimechkbox, 'Enable','off')
else
    set(handles.customtimechkbox, 'Enable','on')
end

% AC SOURCE RADIO BUTTON
% If selected, custom time range can be or not be using and allow turning
% it on or off and vice versa.
function acsourcebtn_Callback(hObject, ~, handles)
if(get(hObject,'Value') == 0)
    set(handles.customtimechkbox, 'Value',1)
    set(handles.customtimechkbox, 'Enable','off')
else
    set(handles.customtimechkbox, 'Enable','on')
end

% AMPLITUDE TEXTBOX

% FEATURES:
% - Can use some simple mathematical operation, eg. +, -, *, /, ^, ...
% - Avoid some unexpected input, eg. text, negative value

% IDEAS:
% - Create of a list of text that need to be ignore.
% - If one or more ignore text contained in the text:
%    + Try to evaluate it, get the absolute value of it's negative.
%    + If the evaluation failed, set the text to '0'.
% - If there is no ignore text:
%    + Try to convert it into (double), get the absolute value of it's negative.
%    + If the conversion failed, set the text to '0'.
function atxtbox_Callback(hObject, ~, handles)
value = get(hObject, 'String');
ignore = ["+","-","*","/","pi","^","exp","log","abs","mod"];
if(contains(value,ignore))
    try
        value = eval(value);
        if(value < 0)
            set(handles.atxtbox, 'String', abs(value));
        else
            set(handles.atxtbox, 'String', string(value));
        end
    catch ME
        set(handles.atxtbox, 'String', '0');
    end
else
    value = str2double(value);
    if(isnan(value))
        set(handles.atxtbox, 'String', '0');
```

```
    end

    if(value < 0)
        set(handles.atxtbox, 'String', abs(value));
    end
end

function atxtbox_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% FREQUENCY TEXTBOX

% FEATURES:
% - Can use some simple mathematical operation, eg. +, -, *, /, ^, ...
% - Avoid some unexpected input, eg. text, negative value, zero

% IDEAS:
% - Create of a list of text that need to be ignore.
% - If one or more ignore text contained in the text:
%    + Try to evaluate it, get the absolute value of it's negative.
%    + If it's  0 or the evaluation failed, set the text to '1'.
% - If there is no ignore text:
%    + Try to convert it into (double), get the absolute value of it's negative.
%    + If it's 0 or the conversion failed, set the text to '1'.
function freqtxtbox_Callback(hObject, ~, handles)
value = get(hObject, 'String');
ignore = ["+","-","*","/","pi","^","exp","log","abs","mod"];
if(contains(value,ignore))
    try
        value = eval(value);
        if(value < 0)
            set(handles.freqtxtbox, 'String', abs(value));
        elseif(value == 0)
            set(handles.freqtxtbox, 'String', '1');
        else
            set(handles.freqtxtbox, 'String', string(value));
        end
    catch ME
        set(handles.freqtxtbox, 'String', '1');
    end
else
    value = str2double(value);
    if(isnan(value) || value == 0)
        set(handles.freqtxtbox, 'String', '1');
    end

    if(value < 0)
        set(handles.freqtxtbox, 'String', abs(value));
    end
end

function freqtxtbox_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% PHASE SHIFT TEXTBOX

% FEATURES:
```

```
% - Can use some simple mathematical operation, eg. +, -, *, /, ^, ...
% - Avoid some unexpected input, eg. text

% IDEAS:
% - Create of a list of text that need to be ignore.
% - If one or more ignore text contained in the text:
%    + Try to evaluate it.
%    + If the evaluation failed, set the text to '0'.
% - If there is no ignore text:
%    + Try to convert it into (double).
%    + If the conversion failed, set the text to '0'.
function phasetxtbox_Callback(hObject, ~, handles)
value = get(hObject, 'String');
ignore = ["+","-","*","/","pi","^","exp","log","abs","mod"];
if(contains(value,ignore))
    try
        value = eval(value);
        if(value < 0)
            set(handles.phasetxtbox, 'String', abs(value));
        else
            set(handles.phasetxtbox, 'String', string(value));
        end
    catch ME
        set(handles.phasetxtbox, 'String', '0');
    end
else
    value = str2double(value);
    if(isnan(value))
        set(handles.phasetxtbox, 'String', '0');
    end

    if(value < 0)
        set(handles.phasetxtbox, 'String', abs(value));
    end
end

function phasetxtbox_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% DC OFFSET TEXTBOX

% FEATURES:
% - Can use some simple mathematical operation, eg. +, -, *, /, ^, ...
% - Avoid some unexpected input, eg. text

% IDEAS:
% - Create of a list of text that need to be ignore.
% - If one or more ignore text contained in the text:
%    + Try to evaluate it.
%    + If the evaluation failed, set the text to '0'.
% - If there is no ignore text:
%    + Try to convert it into (double).
%    + If the conversion failed, set the text to '0'.
function dctxtbox_Callback(hObject, ~, handles)
value = get(hObject, 'String');
ignore = ["+","-","*","/","pi","^","exp","log","abs","mod"];
if(contains(value,ignore))
    try
        value = eval(value);
```

```
        set(handles.dctxtbox, 'String', string(value));
    catch ME
        set(handles.dctxtbox, 'String', '0');
    end
else
    value = str2double(value);
    if(isnan(value))
        set(handles.dctxtbox, 'String', '0');
    end
end

function dctxtbox_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% CUSTOM FUNCTION RADIO BUTTON
% If selected, custom time range must be using and does not allow turning
% it off and vice versa.
function customfuncbtn_Callback(hObject, ~, handles)
if(get(hObject,'Value') == 1)
    set(handles.customtimechkbox, 'Value',1)
    set(handles.customtimechkbox, 'Enable','off')
else
    set(handles.customtimechkbox, 'Enable','on')
end

% CUSTOM FUNCTION TEXTBOX
function customfunctxtbox_Callback(~, ~, ~)

function customfunctxtbox_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% ---------------------- VINS AND RESISTORS (for summing circuit) PANEL ----------------------

% VIN SUM LIST TEXTBOX

% PROBLEMS:
% - If the list has 5 itmes, and plot the 5th vin is selected, when
% removing items, the vinplottxtbox should be updated as the new max.
function vinsumlist_Callback(hObject, ~, handles)
value = str2double(get(handles.vinplottxtbox,'String'));
vinlist = get(hObject,'String');
[row, ~] = size(vinlist);

if(isnan(value))
    set(handles.vinplottxtbox,'String',1);
elseif(value > row)
    set(handles.vinplottxtbox,'String',string(row));
elseif (value < 0)
    set(handles.vinplottxtbox,'String','0');
end

function vinsumlist_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% R SUM LIST TEXTBOX
```

```
function rsumlist_Callback(~, ~, ~)

function rsumlist_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% PLOT VIN SELECT TEXTBOX
% Selecting Vin to plot (summing circuit)

% FEATURES:
% - Only allow user input numbers that in the range of 1 to max of textbox
function vinplottxtbox_Callback(hObject, ~, handles)
value = str2double(get(hObject,'String'));
vinlist = get(handles.vinsumlist,'String');
[row, ~] = size(vinlist);

if(isnan(value))
    set(hObject,'String',1);
elseif(value > row)
    set(hObject,'String',string(row));
elseif (value < 0)
    set(hObject,'String','0');
end

function vinplottxtbox_CreateFcn(hObject, ~, ~)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% ---------------------- SELF-DEFINED FUNCTIONS ----------------------

    % IF THE PROGRAM IS SOLVING
    % Lock the "Solve" button and change the text into "Please wait..." and
    % revert if done.
    function isSolving(True, handles)
    if(True == false)
        set(handles.solvebtn, 'String', 'Solve');
        set(handles.solvebtn, 'Value', 0);
        set(handles.solvebtn, 'Enable', 'on');
    else
        set(handles.solvebtn, 'String', 'Please wait...');
        set(handles.solvebtn, 'Value', 1);
        set(handles.solvebtn, 'Enable', 'off');
    end

% IF SUMMING CIRCUIT IS SELECTED
% Turn off all items which don't belongs to summing circuit and turn on
% all items which belongs to summing circuit and vice versa.
function summingon(istrue,handles)
if(istrue == false)
    set(handles.vinsumlist,'Enable','off');
    set(handles.rsumlist,'Enable','off');
    set(handles.vinplottxtbox,'Enable','off');
    set(handles.dcsourcetxtbox,'Enable','on');
    set(handles.atxtbox,'Enable','on');
    set(handles.freqtxtbox,'Enable','on');
    set(handles.phasetxtbox,'Enable','on');
    set(handles.dctxtbox,'Enable','on');
    set(handles.customfunctxtbox,'Enable','on');
    set(handles.dcsourcebtn,'Enable','on');
```

```
    set(handles.acsourcebtn,'Enable','on');
    set(handles.customfuncbtn,'Enable','on');
else
    set(handles.vinsumlist,'Enable','on');
    set(handles.rsumlist,'Enable','on');
    set(handles.vinplottxtbox,'Enable','on');
    set(handles.dcsourcetxtbox,'Enable','off');
    set(handles.atxtbox,'Enable','off');
    set(handles.freqtxtbox,'Enable','off');
    set(handles.phasetxtbox,'Enable','off');
    set(handles.dctxtbox,'Enable','off');
    set(handles.customfunctxtbox,'Enable','off');
    set(handles.dcsourcebtn,'Enable','off');
    set(handles.acsourcebtn,'Enable','off');
    set(handles.customfuncbtn,'Enable','off');
end

% IF PLOTING ON THE SAME AXES
% Turn off vin and vout axes and turn on sameaxes and vice versa.
function isSameAxes(~, handles)
value = get(handles.sameaxeschkbox,'Value');
cla(handles.vin);
cla(handles.vout);
cla(handles.sameaxes);
if(value == 1)
    set(handles.vin,'Visible','Off');
    set(handles.vout,'Visible','Off');
    set(handles.sameaxes,'Visible','On');
else
    set(handles.vin,'Visible','On');
    set(handles.vout,'Visible','On');
    set(handles.sameaxes,'Visible','Off');
end

% ---------------------- CREDITS ----------------------

% ABOUT BUTTON
% Credits!!! Do not remove or modify. Thanks.
function aboutbtn_Callback(~, ~, ~)
CreateStruct.Interpreter = 'tex';
CreateStruct.WindowStyle = 'non-modal';
message = sprintf('{\\bfHo Chi Minh City University of Technology}\n{\\bfProject:} Operational Amplifier Solver v1.2\n{\\bfSubject:} Analog Signal
Processing\n{\\bfLecturer:} Assoc Prof. Ha Hoang Kha\n{\\bfClass:} TT04 - 211\n{\\bfMembers:}\n+ Luong Trien Thang - 2051194\n+ Nguyen Ngoc Minh Anh -
2051033\n+ Pham Nguyen Trung Tin - 2051203');
uiwait(msgbox(message,CreateStruct));

% ---------------------- MAIN PROGRAM ----------------------

% SOLVE BUTTON (MAIN)
function solvebtn_Callback(hObject, ~, handles)
if(get(hObject, 'Value') == 1)                                          % If the button is clicked
    isSolving(true, handles);                                           %    Triggger "isSolving(true)" function.

    freq = str2double(get(handles.freqtxtbox,'String'));                %    Get the value of "freq".

    if(get(handles.customtimechkbox,'Value') == 0)                      %    If "custom time range" is not checked
        period = 1/freq;                                                %        Get the period
        t = linspace(-period,period,round(exp(2*period)+1000));         %        exp... is used to get more percise on large interval of delta t.
    else                                                                %    Else
        tmin = str2double(get(handles.tmintxtbox,'String'));            %        Get tmin and tmax
        tmax = str2double(get(handles.tmaxtxtbox,'String'));
```

```
    if(tmin > tmax)                                                 %    If tmin > tmax
        set(handles.tmintxtbox,'String',string(tmax));              %        Swap tmin and tmax
        set(handles.tmaxtxtbox,'String',string(tmin));
        t = linspace(tmax,tmin,round(exp(abs(tmin-tmax))+1000));    %        t = linspace...
    elseif(tmin == tmax)                                            %    If tmin == tmax
        message = sprintf('tmin and tmax must not be equal.');      %        Showing error message
        uiwait(errordlg(message));
        isSolving(false, handles);
        return;
    else                                                            %    Else
        t = linspace(tmin,tmax,round(exp(abs(tmin-tmax))+1000));    %        t = linspace...
    end
end

if(get(handles.dcsourcebtn,'Value') == 1)                           %    If DC source is selected
    vin = eval(get(handles.dcsourcetxtbox,'String'));               %        vin = value of dctxtbox
elseif(get(handles.acsourcebtn,'Value') == 1)                       %    If AC source is selected
    dc = str2double(get(handles.dctxtbox,'String'));                %        Get amplitude, phase and dc offset in the textboxes
    amplitude = str2double(get(handles.atxtbox,'String'));
    phase = deg2rad(str2double(get(handles.phasetxtbox,'String')));

    vin = amplitude*sin(2*pi*freq*t + phase) + dc;                  %       vin = A*sin(2*pi*f*t + phase shift) + DC offset
elseif(get(handles.customfuncbtn,'Value') == 1)                     %    If custom function is selected
    try
        vin = eval(get(handles.customfunctxtbox,'String'));         %        Try to evaluate the function
    catch ME                                                        %        If failed, show the error message
        message = sprintf('Error in Vin custom function:\n%s', ME.message);
        uiwait(errordlg(message));
        isSolving(false, handles);                                  %    Triggger "isSolving(false)" function.
    end
end

try
    rf = eval([get(handles.rftxtbox,'String'),'*1000']);            %    Try to evalute Rf
catch ME                                                            %    If failed, show the error message.
    message = sprintf('Error in Rf:\n%s', ME.message);
    uiwait(errordlg(message));
    isSolving(false, handles);                                      %    Triggger "isSolving(false)" function.
end

try
    r = eval([get(handles.rtxtbox,'String'),'*1000']);              %    Try to evalute R
catch ME                                                            %    If failed, show the error message.
    message = sprintf('Error in R:\n%s', ME.message);
    uiwait(errordlg(message));
    isSolving(false, handles);                                      %    Triggger "isSolving(false)" function.
end

if(isempty(find(r<0, 1)) == 0 || isempty(find(rf<0, 1)) == 0)       %    If Rf or R is/contains negative value(s), show the error message.
    message = sprintf('Resistors must not be negative.\nTry to change the time range or resistor functions.');
    uiwait(errordlg(message));
    isSolving(false, handles);                                      %        Triggger "isSolving(false)" function.
    return;                                                         %        Stop the function.
end

if (isequal(size(vin),[1 1]))                                       %    If vin is DC
    vin = vin*ones(size(t));                                        %        Create an array with all are the same value DC with size t.
end

if(get(handles.circuitselect, 'Value') == 1)                        %    If inverting circuit is selected
    vout = -rf./vin/r;                                              %        Calculate vout.
```

```
elseif(get(handles.circuitselect, 'Value') == 2)                        %    If non-inverting circuit is selected
    vout = (rf/r+1).*vin;                                               %        Calculate vout.
elseif(get(handles.circuitselect, 'Value') == 3)                        %    If summing inverting circuit is selected
    vinplot = str2double(get(handles.vinplottxtbox,'String'));          %        Get value of vinplottxtbox.

    vinlist = get(handles.vinsumlist,'String');                         %        Get values of vinlist and rlist.
    rlist = get(handles.rsumlist,'String');
    [row, ~] = size(vinlist);                                           %        Get the number of rows of two of them
    [row2, ~] = size(rlist);
    if(row ~= row2)                                                     %        If row of vinlist is not equal to row of rlist, show the error message.
        message = sprintf('The number of Vins and the number of resistors must be equal.');
        uiwait(errordlg(message));
        isSolving(false, handles);                                      %            Triggger "isSolving(false)" function.
        return;                                                         %            Stop the function.
    end

    sum = 0;                                                            %        Initiate sum.
    for i=1:row                                                         %        For each row
        try
            r_ = eval([rlist{i,:},'*1000']);                            %            Try to evaluate r line i
        catch ME                                                        %            If failed, show the error message
            message = sprintf('Error in Resistors list line %d:\n%s', i, ME.message);
            uiwait(errordlg(message));
            isSolving(false, handles);                                  %            Triggger "isSolving(false)" function.
        end

        if(isempty(find(eval(rlist{i,:})<0, 1)) == 0)                   %        If R line i is/contains negative value, show the error message.
            message = sprintf('Resistors must not be negative.\nTry to change the time range or resistor functions.');
            uiwait(errordlg(message));
            isSolving(false, handles);                                  %            Triggger "isSolving(false)" function.
            return;                                                     %            Stop the function.
        end

        try
            vin_ = eval(vinlist{i,:});                                  %        Try to evaluate vin line i
        catch ME                                                        %        If failed, show the error message
            message = sprintf('Error in Vin list line %d:\n%s', i, ME.message);
            uiwait(errordlg(message));
            isSolving(false, handles);                                  %            Triggger "isSolving(false)" function.
        end

        sum = sum + vin_./r_;                                           %        sum = sum + vin_line_i/r_line_i
    end

    if(vinplot == 0)                                                    %        If plot vin textbox == 0
        vin = [];                                                       %            Create a blank array.
        for i=1:row                                                     %            For each vin from vinlist, add to that new array
            value = eval(vinlist{i,:});                                 %                this will create a [vin x t] array that contains
            if (isequal(size(value),[1 1]))                             %                all of the vins.
                vin = [vin;value*ones(size(t))];
            else
                vin = [vin;value];
            end
        end
    else                                                                %        Else
        vin = eval(vinlist{vinplot,:});                                 %            Plot selected vin.
        if (isequal(size(vin),[1 1]))
            vin = vin*ones(size(t));
        end
    end
```

```
        vout = -sum.*rf;                                                    %       vout = -sum*rf
        if (isequal(size(vout),[1 1]))
            vout = vout*ones(size(t));
        end
    elseif(get(handles.circuitselect, 'Value') == 4)                        %   If summing non-inverting circuit is selected
        vinplot = str2double(get(handles.vinplottxtbox,'String'));          %       Same as above.

        vinlist = get(handles.vinsumlist,'String');
        rlist = get(handles.rsumlist,'String');
        [row, ~] = size(vinlist);
        [row2, ~] = size(rlist);
        if(row ~= row2)
            message = sprintf('The number of Vins and the number of resistors must be equal.');
            uiwait(errordlg(message));
            isSolving(false, handles);
            return;
        end

        sum1 = 0;                                                           %       Initiate sum1 and sum2.
        sum2 = 0;
        for i=1:row
            try
                r_ = eval([rlist{i,:},'*1000']);                            %           Try to evaluate r line i
            catch ME                                                        %           If failed, show the error message
                message = sprintf('Error in Resistors list line %d:\n%s', i, ME.message);
                uiwait(errordlg(message));
                isSolving(false, handles);                                  %           Triggger "isSolving(false)" function.
            end

            if(isempty(find(eval(rlist{i,:})<0, 1)) == 0)                   %       If R line i is/contains negative value, show the error message.
                message = sprintf('Resistors must not be negative.\nTry to change the time range or resistor functions.');
                uiwait(errordlg(message));
                isSolving(false, handles);                                  %           Triggger "isSolving(false)" function.
                return;                                                     %           Stop the function.
            end

            try
                vin_ = eval(vinlist{i,:});                                  %       Try to evaluate vin line i
            catch ME                                                        %       If failed, show the error message
                message = sprintf('Error in Vin list line %d:\n%s', i, ME.message);
                uiwait(errordlg(message));
                isSolving(false, handles);                                  %           Triggger "isSolving(false)" function.
            end

            sum1 = sum1 + vin_./r_;                                         %       sum1 = sum1 + vin1/r1 + vin2/r2 + ...
            sum2 = sum2 + 1./r_;                                            %       sum2 = sum2 + 1/r1 + 1/r2
        end

        if(vinplot == 0)                                                    %       Same as summing inverting circuit
            vin = [];
            for i=1:row
                value = eval(vinlist{i,:});
                if (isequal(size(value),[1 1]))
                    vin = [vin;value*ones(size(t))];
                else
                    vin = [vin;value];
                end
            end
        else
            vin = eval(vinlist{vinplot,:});
            if (isequal(size(vin),[1 1]))
```

```
                vin = vin*ones(size(t));
            end
        end

        vout = (rf/r + 1).*sum1./sum2;                                  %       Calculate vout.
        if (isequal(size(vout),[1 1]))
            vout = vout*ones(size(t));
        end
    end

    sameAxes = get(handles.sameaxeschkbox,'Value');                     %   Get "plot on the same axes" value.
    isSameAxes(sameAxes,handles);                                       %   Trigger isSameAxes(sameAxes)
    if(sameAxes == true)                                                %   If plot on same axes is true
        axes(handles.sameaxes);                                         %       Plot vin with blue color, vout with red color on a same axes.
        vinplot = plot(t,vin,'Color',[0.000,  0.447,  0.741]);
        hold on;
        voutplot = plot(t,vout,'Color',[0.8500 0.3250 0.0980]);
        hold off;
        legend([vinplot(1),voutplot(1)],'Vin(t)','Vout(t)');            %       Add the legends.
    else                                                                %   Else
        axes(handles.vin);                                              %       Plot vin with blue color on vin axes, vout with red color on vout axes.
        plot(t,vin,'Color',[0.000,  0.447,  0.741]);                    %           then add the legends.
        legend('Vin(t)');
        axes(handles.vout);
        plot(t,vout,'Color',[0.8500 0.3250 0.0980]);
        legend('Vout(t)');
    end

    gapsRatio = 0.1;                                                    % Gaps for 2 bound of y-axis.
    if(sameAxes == true)                                                % If sameaxes is selected
        set(handles.sameaxes, 'XLim', t(size(t)));                      %   Set x-axis limit = size of t.
        ylimsameaxes = get(handles.sameaxes, 'YLim');                   %   Get y-axis limit then add a bit and set it to the axes.
        minsameaxes = ylimsameaxes(1);
        maxsameaxes = ylimsameaxes(2);
        set(handles.sameaxes, 'YLim', [minsameaxes-gapsRatio*abs(minsameaxes) maxsameaxes+gapsRatio*abs(maxsameaxes)]);
    else                                                                % Else
        set(handles.vin, 'XLim', t(size(t)));                           %   Same as above but for vin and vout.
        set(handles.vout, 'XLim', t(size(t)));

        ylimVin = get(handles.vin, 'YLim');
        ylimVout = get(handles.vout, 'YLim');
        minVin = ylimVin(1);
        maxVin = ylimVin(2);
        minVout = ylimVout(1);
        maxVout = ylimVout(2);
        set(handles.vin, 'YLim', [minVin-gapsRatio*abs(minVin) maxVin+gapsRatio*abs(maxVin)]);
        set(handles.vout, 'YLim', [minVout-gapsRatio*abs(minVout) maxVout+gapsRatio*abs(maxVout)]);
    end

    if(get(handles.dcsourcebtn, 'Value') == 1 && isequal(size(find(vout==vout(1))),size(vout)))% If DC source select and vout is a constant
        set(handles.voutdctxt,'String', 'Vout = ' + string(vout));      %   Show the vout constant value.
    else                                                                % Else
        set(handles.voutdctxt,'String', '');                            %   Show nothing.
    end
end
isSolving(false, handles);                                              % Solving done! Triggger "isSolving(false)" function.
```

IV. Appendix

- The source code will be available on Github after 7th November, 2021 via this link.
- Ability to plot resistor function will also be added soon.

V. References

[1] Assoc. Prof. H. H. Kha, HCMC Univerity of Technology, "Operational Amplifier".

[2] Assoc. Prof. H. H. Kha, HCMC Univerity of Technology, "Project OpAmp Sample".

[3] R. Teja, "Summing Amplifier," Electronics Hub, 22 April 2021, https://www.electronicshub.org/summing-amplifier/.

[4] M. Safwat, "How to draw any function using GUI MATLAB.," YouTube, 12 April 2017, https://www.youtube.com/watch?v=ZjH1i9SfNa0.

[5] user3797886, rayryeng, "How do I make a png with transparency appear transparent in MatLab?," Stack Overflow, 07 August 2014, https://stackoverflow.com/questions/25172389/how-do-i-make-a-png-with-transparency-appear-transparent-in-matlab.

[6] MATLAB, "MATLAB Documentation," MathWorks, https://www.mathworks.com/help/matlab/.

[7] Maxim Integrated, "GLOSSARY DEFINITION FOR OP AMP," Maxim Integrated, 2020, https://www.maximintegrated.com/en/glossary/definitions.mvp/term/Op%20amp/gpk/883.

[8] All About Circuits, "Introduction to Operational Amplifiers (Op-amps)," EETech Media, LLC., https://www.allaboutcircuits.com/textbook/semiconductors/chpt-8/introduction-operational-amplifiers/.

VI. Table of Figures