**RC CIRCUIT WITH LOW-PASS FILTER - MATLAB PROJECT**

**Bria Raul**

**MATLAB Files Explained**

***Main file: circuit\_rc\_ftj\_main.m***

The main file deletes all data and cleans the figure, after which it initializes the graphical interface. It manages the general flow of the application and calls the necessary functions for simulation and display.

------------------------------------------------------------------------------------------------------------ clear all;

close all;

circuit\_rc\_ftj\_gui();

------------------------------------------------------------------------------------------------------------

***Graphic Component: create\_gui\_components.m***

This file creates the components that are part of the graphical user interface.

-The signal adjustment panel is created using the uipanel function

-The slider, respectively the edit buttons for amplitude, period, resistance and capacitance are created inside the panel using the uilabel, uislider, respectively uiedit functions

-A group of radio buttons is added for selecting the type of signal: sinusoidal, rectangular or triangular, using the uiradiobutton function

-The Start button is added with the help of the uibutton function

-The image is present with the help of the axes and the uiaxes, imread and imshow functions

-With the help of a label, the frequency of cutting also appears

-One of the most important parts of this file is the creation of structures for components and axes, using the struct function

------------------------------------------------------------------------------------------------------------ function [components, axesHandles] = create\_gui\_components(fig)

signalPanel = uipanel(fig, 'Title', 'Ajustare Semnal ?i Circuit', 'Position', [20, 370, 550, 300]);

uilabel(signalPanel, 'Text', 'Amplitudine [V]:', 'Position', [20, 215, 150, 22]);

amplitudeSlider = uislider(signalPanel, 'Position', [150, 225, 200, 3], 'Limits', [0, 10]);

uilabel(signalPanel, 'Text', 'Perioada [s]:', 'Position', [20, 145, 150, 22]);

periodEdit = uieditfield(signalPanel, 'numeric', 'Position', [150, 145, 70, 22]);

periodEdit.Value = 0.1;

uilabel(signalPanel, 'Text', 'Rezistenta [Ohm]:', 'Position', [20, 95, 150, 22]);

resistorEdit = uieditfield(signalPanel, 'numeric', 'Position', [150, 95, 70, 22]);

resistorEdit.Value = 100;

uilabel(signalPanel, 'Text', 'Capacitate [F]:', 'Position', [20, 45, 150, 22]);

capacitorEdit = uieditfield(signalPanel, 'numeric', 'Position', [150, 45, 70, 22]);

capacitorEdit.Value = 0.001;

signalGroup = uibuttongroup(signalPanel, 'Title', 'Tip Semnal', 'Position', [270, 20, 200, 150]);

sinusRadio = uiradiobutton(signalGroup, 'Position', [50, 80, 150, 22], 'Text', 'Sinusoidal', 'Value', true);

dreptRadio = uiradiobutton(signalGroup, 'Position', [50, 50, 150, 22], 'Text', 'Dreptunghiular', 'Value', false);

triRadio = uiradiobutton(signalGroup, 'Position', [50, 20, 150, 22], 'Text', 'Triunghiular', 'Value', false);

simulateButton = uibutton(signalPanel, 'Text', 'Start', 'Position', [400, 200, 100, 40]);

ax1 = uiaxes(fig, 'Position', [20, 50, 460, 250]);

title(ax1, 'Input signal');

xlabel(ax1, 'Timp [s]');

ylabel(ax1, 'Amplitudine [V]');

ax2 = uiaxes(fig, 'Position', [500, 50, 460, 250]);

title(ax2, 'R?spuns Circuit RC');

xlabel(ax2, 'Timp [s]');

ylabel(ax2, 'Amplitudine [V]');

img\_ax = uiaxes(fig, 'Position', [570, 320, 400, 350]);

img = imread('rc\_ftj. PNG');

imshow(img, 'Parent', img\_ax);

ftLabel = uilabel(fig, ...

'Text', 'Trimming frequency: N/A', ...

'Position', [400, 320, 250, 22], ...

'FontSize', 15);

components = struct('amplitudeSlider', amplitudeSlider, ...

'periodEdit', periodEdit, ...

'resistorEdit', resistorEdit, ...

'capacitorEdit', capacitorEdit, ...

'sinusRadio', sinusRadio, ...

'killedRadio', killedRadio,...

'triRadio', triRadio, ...

'simulateButton', simulateButton, ...

'ftLabel', ftLabel);

axesHandles = struct('ax1', ax1, 'ax2', ax2, 'img\_ax', img\_ax);

end

------------------------------------------------------------------------------------------------------------

***Graphic interface: circuit\_rc\_ftj\_gui.m***

This file creates the main GUI window with the help of the uifigure function. After that, the components and axes are created and brought into this file with the help of calling the function of the previous file (create\_gui\_components.m).

A callback is set for the simulation button (Start), and the callback function called simulateCallback collects the values of the components selected by the user and defines the time interval and frequency. The function then generates the input signal depending on the selected signal type. Next, the callback function calculates the product of resistance and capacity and the cutting frequency, after which it updates the label using the sprinf function.

After simulating the output response with the help of calling the simulate\_rc\_response function with the parameters of the input signal, the time interval and the RC product, the graphs are updated with the help of calling another function called update\_plots with the parameters of the axes, the time interval, the input signal, and the output signal.

------------------------------------------------------------------------------------------------------------ fig = uifigure('Name', 'Circuit RC FTJ', 'Position', [50, 50, 1000, 700]);

[components, axesHandles] = create\_gui\_components(fig);

components.simulateButton.ButtonPushedFcn = @(src, event) simulateCallback();

function simulateCallback()

A = components.amplitudeSlider.Value;

T = components.periodEdit.Value;

sinus = components.sinusRadio.Value;

drept = components.dreptRadio.Value;

tri = components.triRadio.Value;

R = components.resistorEdit.Value;

C = components.capacitorEdit.Value;

t = 0:0.0000001:2 \* T;

f = 1 / T;

if sinus

inputSignal = A \* sin(2 \* pi \* f \* t);

elseif right

inputSignal = A \* square(2 \* pi \* f \* t);

elseif tri

inputSignal = A \* sawtooth(2 \* pi \* f \* t, 0.5);

end

RC = R \* C;

ft = 1 / (2 \* pi \* RC);

components.ftLabel.Text = sprintf('Trim frequency: %.5f Hz', ft);

outputSignal = simulate\_rc\_response(inputSignal, t, RC);

update\_plots(axesHandles, t, inputSignal, outputSignal);

end

end

------------------------------------------------------------------------------------------------------------

***Simulation Function: simulate\_rc\_response.m***

This function defines the transfer function (H) with the help of the tf function (transfer function) which has the value 1 as the numerator and the value sRC + 1 as the denominator.

After defining the transfer function, the output signal is calculated using the lsim function, with the parameters of the desired system (H), the input signal (inputSignal) and the time interval (t).

------------------------------------------------------------------------------------------------------------ function outputSignal = simulate\_rc\_response(inputSignal, t, RC)

H = tf(1, [RC 1]);

outputSignal = lsim(H, inputSignal, t);

end

------------------------------------------------------------------------------------------------------------

***Graphics update: update\_plots.m***

The update\_plots function is responsible for displaying the graphics of the input signal and the circuit response. It uses the axes defined in the GUI to visually render the simulation results.

The title function defines the title of each axis, and the xlabel and ylabel function names the x-axis and y-axis of each axis, respectively.

------------------------------------------------------------------------------------------------------------ function update\_plots(axesHandles, t, inputSignal, outputSignal)

plot(axesHandles.ax1, t, inputSignal);

title(axesHandles.ax1, 'Semnal de intrare');

xlabel(axesHandles.ax1, 'Timp [s]');

ylabel(axesHandles.ax1, 'Amplitudine [V]');

grid(axesHandles.ax1, 'on');

plot(axesHandles.ax2, t, outputSignal);

title(axesHandles.ax2, 'Raspuns Circuit RC');

xlabel(axesHandles.ax2, 'Timp [s]');

ylabel(axesHandles.ax2, 'Amplitudine [V]');

grid(axesHandles.ax2, 'on');

End

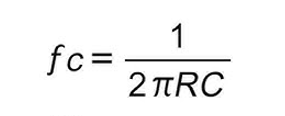
------------------------------------------------------------------------------------------------------------

**FTJ RC Circuit Components**

* **Resistor (R):** It is a passive element that limits the current in a circuit. Its resistance can be adjusted to influence the behavior of the circuit.
* **Capacitor (C):** It is another passive element that stores energy in the form of an electric field. Its ability to determine how the circuit responds to signals of different frequencies.
* **Input signal**: The signal that enters the circuit can be of several types. It is applied to the circuit input and is filtered by the behavior of the RC circuit.

**Cutting Frequency**

The cut-off frequency is the frequency at which the signal is reduced to half of its original value. This is an important feature of RC circuits because it establishes the separation point between low-frequency and high-frequency signals. The calculation formula for the cutting frequency is:



where:

* R is resistance in ohms
* C is the capacity in farazi
* fc is the cut-off frequency in hertz (Hz)

**Behavior of the RC FTJ circuit**

The RC FTJ circuit works as a filter that allows low-frequency signals to pass through, while high-frequency signals are attenuated. At frequencies higher than the cut-off frequency, the circuit begins to reduce the amplitude of the signal. This behavior is used in applications such as filtering audio signals, radio communication, and more.

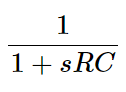
**Applications of the RC FTJ Circuit**

The FTJ RC circuit is used in numerous applications, including:

* **Audio Signal Filtering**: Allows low-frequency audio signals (e.g., low-pitched sounds) to pass through and attenuates high-frequency signals (e.g., noises)
* **Signal filtering in radio communications**: Can be used to separate useful signals from higher frequency signals
* **Filtering of signaling signals in electronic circuits**: These can be used to isolate specific signals that are used in various electronic applications

**The Transfer Function of an RC Circuit**

For an FTJ (Low Pass Filter) RC circuit, the transfer function can be expressed as follows:



where:

* R is resistance (Ohms)
* C is capacity (Farazi)
* s is the complex frequency operator, s = jω, where ω is the angular frequency and j is the imaginary part

**Conclusion**

The RC FTJ circuit is a classic example of an electronic filter that can be used to control electrical signals based on their frequency. In our project, we implemented this circuit to study how different signals are modified by this circuit, in order to demonstrate its applicability in various fields.

**Bibliography**

<https://sites.google.com/site/bazeleelectronicii/home/filtre/2-passive-low-pass-filter>

<https://www.allaboutcircuits.com/technical-articles/understanding-transfer-functions-for-low-pass-filters/>

<https://www.mathworks.com/help/>