Appendix I: Matlab Code

Main page

```
% Initialize menu
function main()

%Cleaning previous graphs
close all;

% Buttons
interficie = uifigure;
interficie.Name = 'NanoBIA4Wire User Interface';

generate = uibutton(interficie, 'Text', 'Generate Signal',
'VerticalAlignment', 'top', 'ButtonPushedFcn', @(generate,event)
click_generate(generate, interficie));
calibrate = uibutton(interficie, 'Text', 'Calibrate Signal',
'Position', [100 150 100 22], 'VerticalAlignment', 'top',
'ButtonPushedFcn', @(calibrate,event) click_calibrate(calibrate,
interficie));
adjust = uibutton(interficie, 'Text', 'Adjust Model', 'Position', [100
200 100 22], 'VerticalAlignment', 'top', 'ButtonPushedFcn', @(adjust,
event) click_adjust(adjust, interficie));
end

% Callback // Click generate
function click_generate(generate, interficie)
generate_signal;
end

% Callback // Click calibrate
function click_calibrate(calibrate, interficie)
calibrate_signal;
end

% Callback // Click adjust
function click_adjust(adjust, interficie)
model_adjusting;
end
```

Generate Signal

```
% Initializing GUI and cleaning axis
function GUI_Initialization()

% Global variables to enter on handles
global ax_up;
global ax_down;
global devices_list;
global start_freq;
global stop_freq;
global sweep_points;
global linear;
global logarithmic;
global run_time;
```

```
global meas_freq;
global single_freq;
global sample_freq;
global freq_sweep;
global import_file;

% Cleaning previous graphs
close all

% Graphs
axis = uifigure;
screen_height = 700;
screen_width = 1500;
axis.Name = 'Generate Signal User Interface';
axis.Position = [0 40 screen_width screen_height];

% Impedance Magnitude
ax_up = uiaxes(axis);
ax_up.OuterPosition = [(screen_width/5)+10 screen_height/2
2*(screen_width/3) screen_height/2];
ax_up.PositionConstraint = 'outerposition';
ax_up.Title.String = 'Impedance Magnitude';
ax_up.XLabel.String = 'Time (s)';
ax_up.YLabel.String = 'Magnitude (?)';

% Impedance Phase
ax_down = uiaxes(axis);
ax_down.Position = [(screen_width/5)+10 0 2*(screen_width/3)
screen_height/2];
ax_down.PositionConstraint = 'outerposition';
ax_down.Title.String = 'Impedance Phase';
ax_down.XLabel.String = 'Time (s)';
ax_down.YLabel.String = 'Phase (°)';

%Buttons

screen_partitions = 43;
partition_height = screen_height/screen_partitions;
buttons_font_size = 11;
sangria = 10;
buttons_width = (screen_width/5)-10;
buttons_width_real = (screen_width/5)-30;

%Scan
scan = uibutton(axis, 'Position', [10 2+33*partition_height
buttons_width_real partition_height+1], 'Text', 'Scan', 'FontSize',
buttons_font_size, 'VerticalAlignment', 'top', 'ButtonPushedFcn',
@(scan,event) click_scan(scan,axis));

%Connect
connect = uibutton(axis, 'Position', [10 2+32*partition_height
buttons_width_real partition_height+1], 'Text', 'Connect', 'FontSize',
buttons_font_size, 'VerticalAlignment', 'top', 'ButtonPushedFcn',
@(connect,event) click_connect(connect,axis));

%Save to file
save = uibutton(axis, 'Position', [10 2+31*partition_height
buttons_width_real partition_height+1], 'Text', 'Save to file',
'FontSize', buttons_font_size, 'VerticalAlignment', 'top',
'ButtonPushedFcn', @(save,event) click_save(save,axis));
```

```
%Quit
quit = uibutton(axis, 'Position', [10 2+30*partition_height
buttons_width_real partition_height+1], 'Text', 'Quit', 'FontSize',
buttons_font_size, 'VerticalAlignment', 'top', 'ButtonPushedFcn',
@(quit,event) click_quit(quit,axis));

%Left menu
uilabel(axis,'Text','Bluetooth Devices','Position',[0
42*partition_height screen_width/5
partition_height],'HorizontalAlignment','center','VerticalAlignment','
top','FontWeight','bold','FontSize',12);
devices_list =
uilistbox(axis,'Items',{},'Value',{},'Position',[sangria
34.5*partition_height buttons_width
7.5*partition_height],'FontSize',12);
uilabel(axis,'Text','Analysis Options','Position',[0
(29*partition_height)-5 screen_width/5
partition_height],'HorizontalAlignment','center','VerticalAlignment','
top','FontWeight','bold','FontSize',12);
uilabel(axis,'Text','Run time','Position',[sangria
(27.5*partition_height)-5 screen_width/5
partition_height+4],'HorizontalAlignment','left','VerticalAlignment','
center','FontSize',12);
run_time =
uitextarea(axis,'Value',{'10'},'Position',[((screen_width/5)+10)/2
(27.5*partition_height)-5 buttons_width/2
partition_height+4],'FontSize',12);
uilabel(axis,'Text','Sample frequency','Position',[sangria
(26.3*partition_height)-5 screen_width/5
partition_height+4],'HorizontalAlignment','left','VerticalAlignment','
center','FontSize',12);
sample_freq =
uitextarea(axis,'Value',{'1'},'Position',[((screen_width/5)+10)/2
(26.3*partition_height)-5 buttons_width/2
partition_height+4],'FontSize',12);
uilabel(axis,'Text','Mode','Position',[0 (25*partition_height)-5
screen_width/5
partition_height],'HorizontalAlignment','center','VerticalAlignment','
top','FontWeight','bold','FontSize',12);

mode = uibuttongroup(axis,'Position',[10 (24*partition_height)-5
screen_width/5 partition_height], 'SelectionChangedFcn',
@click_single_freq );
mode.BorderType = 'none';
single_freq = uiradiobutton(mode,'Position',[10 0 screen_width/5
partition_height],'Text','Single Frequency','FontSize', 11);
freq_sweep = uiradiobutton(mode,'Position',[10+(screen_width/5)/2 0
screen_width/5 partition_height],'Text','Frequency Sweep','FontSize',
11);

uilabel(axis,'Text','Measurement frequency','Position',[sangria
(22.3*partition_height)-5 screen_width/5
partition_height+4],'HorizontalAlignment','left','VerticalAlignment','
center','FontSize',12);
meas_freq =
uitextarea(axis,'Value',{'50000'},'Position',[((screen_width/5)+10)/2
(22.3*partition_height)-5 buttons_width/2
partition_height+4],'FontSize',12);
uilabel(axis,'Text','Sweep Type','Position',[0 (21*partition_height)-6
screen_width/5
```

```
partition_height],'HorizontalAlignment','center','VerticalAlignment','
top','FontWeight','bold','FontSize',12);

type = uibuttongroup(axis,'Position',[10 (20*partition_height)-6
screen_width/5 partition_height]);
type.BorderType = 'none';
linear = uiradiobutton(type,'Position',[10 0 screen_width/5
partition_height],'Text','Linear', 'Enable', 'off');
logarithmic = uiradiobutton(type,'Position',[10+(screen_width/5)/2 0
screen_width/5 partition_height],'Text','Logarithmic','FontSize',11,
'Enable', 'off');

uilabel(axis,'Text','Start frequency','Position',[sangria
(18.5*partition_height)-6 screen_width/5
partition_height+4],'HorizontalAlignment','left','VerticalAlignment','
center','FontSize',12);
start_freq =
uitextarea(axis,'Value',{'500'},'Position',[((screen_width/5)+10)/2
(18.5*partition_height)-6 buttons_width/2
partition_height+4],'FontSize',12, 'Enable', 'off', 'Editable',
'off');
uilabel(axis,'Text','Stop frequency','Position',[sangria
(17.3*partition_height)-6 screen_width/5
partition_height+4],'HorizontalAlignment','left','VerticalAlignment','
center','FontSize',12);
stop_freq =
uitextarea(axis,'Value',{'50000'},'Position',[((screen_width/5)+10)/2
(17.3*partition_height)-6 buttons_width/2
partition_height+4],'FontSize',12, 'Enable', 'off', 'Editable',
'off');
uilabel(axis,'Text','Sweep points','Position',[sangria
(16.1*partition_height)-6 screen_width/5
partition_height+4],'HorizontalAlignment','left','VerticalAlignment','
center','FontSize',12);
sweep_points =
uitextarea(axis,'Value',{'10'},'Position',[((screen_width/5)+10)/2
(16.1*partition_height)-6 buttons_width/2
partition_height+4],'FontSize',12, 'Enable', 'off', 'Editable',
'off');

uilabel(axis,'Text','Visualize Signal','Position',[sangria
(15*partition_height)-12 screen_width/5
partition_height],'HorizontalAlignment','center','VerticalAlignment','
top','FontWeight','bold','FontSize',12);
uilabel(axis,'Text','File to visualize','Position',[sangria
(13.8*partition_height)-15 screen_width/5
partition_height+4],'HorizontalAlignment','left','VerticalAlignment','
center','FontSize',12);
import_file = uitextarea(axis,'Position',[((screen_width/5)+10)/2
(13.8*partition_height)-15 buttons_width/2
partition_height+4],'FontSize',12);
select_file = uibutton(axis, 'Text', 'Select File', 'Position',
[sangria (12.6*partition_height)-17 buttons_width_real/2
partition_height+4], 'ButtonPushedFcn', @(import_file,event)
click_import_file(import_file,ax_up));

end

% Callback // Click Select File
function click_import_file(import_file,ax_up)
```

```
global import_file;
global ax_up;
global ax_down;

[filename, path] = uigetfile();
leer = [path, filename];
import_file.Value = leer;

try
    datos = readtable(leer);
    tiempo = datos.Time;
    frecuencia = datos.Frequency;
    magnitud = datos.Magnitude;
    fase = datos.Phase;

catch excepcio
    uiconfirm(axis, 'Incorrect Format File', 'Error', 'Options',
{'OK'}, 'icon', 'error');
end

semilogx(ax_up, frecuencia, magnitud);
semilogx(ax_down, frecuencia, fase);
end

% Callback // Click scan
function click_scan(scan, axis)

global devices_list;
list = blelist;
sublist = table2array(list(:, {'Address'}));
sublist2 = table2array(list(:, {'Name'}));

for i = 1:sublist.length()
    if eq("", sublist2(i))
        devices_list.Items(i) = cellstr(sublist(i));
    else
        devices_list.Items(i) = cellstr(sublist2(i));
    end
end

end

% Callback // Click connect
function click_connect(connect, axis)

clear global caracteristica;
clear global bio_device;

global bio_device;
global caracteristica;
global end_time;
global xtime;
global freq;
global mag;
global phase;
global sample_period;
global run_time;
global sample_freq;
global devices_list;
```

```
global start_freq;
global stop_freq;
global sweep_points;
global linear;
global meas_freq;
global freq_sweep;

xtime = [];
freq = [];
mag = [];
phase = [];
end_time =
uint32(str2double(run_time.Value)*str2double(sample_freq.Value));
sample_period = 1./str2double(sample_freq.Value);

% ConfigData creation with the format corresponding to the ConfigData
sample_freq_float = typecast((single(str2double(sample_freq.Value))),
'uint32');
time_int =
typecast((uint32(str2double(run_time.Value)*str2double(sample_freq.Val
ue))), 'uint32');
meas_freq_float = typecast((single(str2double(meas_freq.Value))),
'uint32');
if freq_sweep.Value
    freq_sweep_type = 1;
else
    freq_sweep_type = 0;
end
freq_sweep_int = typecast((uint32(freq_sweep_type)), 'uint32');
start_freq_float = typecast((single(str2double(start_freq.Value))),
'uint32');
stop_freq_float = typecast((single(str2double(stop_freq.Value))),
'uint32');
sweep_points_int = typecast((uint32(str2double(sweep_points.Value))),
'uint32');
if linear.Value
    type = 0;
else
    type = 1;
end
type_int = typecast((uint32(type)), 'uint32');

config_data = [typecast(sample_freq_float, 'uint8') typecast(time_int,
'uint8') typecast(meas_freq_float, 'uint8') typecast(freq_sweep_int,
'uint8') typecast(start_freq_float, 'uint8') typecast(stop_freq_float,
'uint8') typecast(sweep_points_int, 'uint8') typecast(type_int,
'uint8')];

bio_device = ble(devices_list.Value);
disp("Connected to " + devices_list.Value + ": " +
bio_device.Connected);

caracteristica = characteristic(bio_device, "181B", "2A9C");
subscribe(caracteristica, "notification");
caracteristica.DataAvailableFcn = @plot_update;
c = characteristic(bio_device, "181B", "c6b9f11b-f7bf-4f7c-84e0-
ff810b38c248");
write(c, config_data);

end
```

```
% Callback // Click save
function click_save(save, axis)
global freq;
global mag;
global phase;
global xtime;

try
    Time = transpose(xtime);
    Frequency = transpose(freq);
    Magnitude = transpose(mag);
    Phase = transpose(phase);
    fitxer = table(Time, Frequency, Magnitude, Phase);
    writetable(fitxer,'C:\Users\34638\Documents\UPC\TFG\Results.txt',
'WriteMode', 'Append');
    uiconfirm(axis, 'File saved!', 'Success', 'Options', {'OK'},
'icon', 'success');
catch excepcio
    uiconfirm(axis, 'Unable to save file', 'Error', 'Options', {'OK'},
'icon', 'error');
end
end

% Callback // Click quit
function click_quit(quit,axis)

global caracteristica;
global bio_device;
selection = uiconfirm(axis, 'Are you sure you want to close the
NanoBIA4Wire ui?', 'Exit ui?', 'Options', {'Yes', 'No'},
'DefaultOption', 'Yes', 'CancelOption', 'No', 'icon', 'warning');
if selection == "Yes"
    axis.Visible = "off";
end
if bio_device.Connected
    unsubscribe(caracteristica);
    caracteristica.DataAvailableFcn = [];
    clear global caracteristica;
    clear global bio_device;
end
end

% Callback // Click single frequency
function click_single_freq(src, evt)
global start_freq;
global stop_freq;
global sweep_points;
global linear;
global logarithmic;
global run_time;
global meas_freq;
global single_freq;
global ax_up;
global ax_down;

if evt.NewValue == single_freq
    start_freq.Editable = "off";
    start_freq.Enable = "off";
    stop_freq.Editable = "off";
```

```
    stop_freq.Enable = "off";
    sweep_points.Editable = "off";
    sweep_points.Enable = "off";
    linear.Enable = "off";
    logarithmic.Enable = "off";
    run_time.Editable = "on";
    run_time.Enable = "on";
    meas_freq.Editable = "on";
    meas_freq.Enable = "on";
    ax_up.XLabel.String = 'Time (s)';
    ax_down.XLabel.String = 'Time (s)';

else
    start_freq.Editable = "on";
    start_freq.Enable = "on";
    stop_freq.Editable = "on";
    stop_freq.Enable = "on";
    sweep_points.Editable = "on";
    sweep_points.Enable = "on";
    linear.Enable = "on";
    logarithmic.Enable = "on";
    run_time.Editable = "off";
    run_time.Enable = "off";
    meas_freq.Editable = "off";
    meas_freq.Enable = "off";
    ax_up.XLabel.String = 'Frequency (Hz)';
    ax_down.XLabel.String = 'Frequency (Hz)';
end
end

% Handle function to update parameters
function plot_update(src, evt)

%Global variables
global ax_up;
global ax_down;
global xtime;
global sample_period;
global freq;
global mag;
global phase;
global single_freq;
global logarithmic;

% Read data
[data, timestamp] = read(src, "oldest");

% Interpret data
data_freq = str2double(char(data(1:10)));
data_mag = str2double(char(data(11:23)));
data_phase = str2double(char(data(24:35)));


if isempty(xtime)
    xtime = [xtime 0];
else
    xtime = [xtime (xtime(end) + sample_period)];
end

% if data_phase > 180
```

```
%     data_phase = data_phase-360;
% end

freq = [freq data_freq];
mag = [mag data_mag];
phase = [phase unwrap(data_phase)];

if length(xtime) > 1
    if single_freq.Value == 1
        if logarithmic.Value == 1
            semilogx(ax_up, xtime, mag);
            semilogx(ax_down, xtime, phase);
        else
            plot(ax_up, xtime, mag);
            plot(ax_down, xtime, phase);
        end
    else
        if logarithmic.Value == 1
            semilogx(ax_up, freq, mag);
            semilogx(ax_down, freq, phase);
        else
            plot(ax_up, freq, mag);
            plot(ax_down, freq, phase);
        end
    end
end
end
```

Calibrate Signal

```
function GUI_Inicialization()
global arxiu_cal;
global rcal;
global arxiu_ref;
global axis;
global ax_up;
global ax_down;

% Cleaning previous graphs
close all;

% Graphs
axis = uifigure;
screen_height = 700;
screen_width = 1500;
axis.Name = 'Calibrate Signal Interface';
axis.Position = [0 40 screen_width screen_height];
screen_partitions = 43;
partition_height = screen_height/screen_partitions;
buttons_width = (screen_width/5)-10;
buttons_width_real = (screen_width/5)-30;

% Impedance Magnitude
ax_up = uiaxes(axis);
ax_up.OuterPosition = [(screen_width/5)+10 screen_height/2
2*(screen_width/3) screen_height/2];
ax_up.PositionConstraint = 'outerposition';
ax_up.Title.String = 'Impedance Magnitude';
ax_up.XLabel.String = 'Frequency (Hz)';
```

```
ax_up.YLabel.String = 'Magnitude (?)';
ax_up.XScale = 'log';

% Impedance Phase
ax_down = uiaxes(axis);
ax_down.Position = [(screen_width/5)+10 0 2*(screen_width/3)
screen_height/2];
ax_down.PositionConstraint = 'outerposition';
ax_down.Title.String = 'Impedance Phase';
ax_down.XLabel.String = 'Frequency (Hz)';
ax_down.YLabel.String = 'Phase (°)';
ax_down.XScale = 'log';

% Buttons
uilabel(axis,'Text','Resistance Value:', 'Position', [10
2+33*partition_height buttons_width_real partition_height+1],
'HorizontalAlignment', 'left', 'VerticalAlignment', 'center');
rcal = uitextarea(axis, 'Position',[((screen_width/5)+10)/2
2+33*partition_height buttons_width/2 partition_height+4]);
uilabel(axis,'Text','Reference file: ','Position',[10
2+31*partition_height buttons_width_real partition_height+1]);
arxiu_ref = uitextarea(axis, 'Position',[((screen_width/5)+10)/2
2+31*partition_height buttons_width/2 partition_height+4]);
import_ref = uibutton(axis, 'Text', 'Select File', 'Position', [70
2+29*partition_height buttons_width_real-100 partition_height+5],
'ButtonPushedFcn', @(import_ref,event)
click_import_data_ref(import_ref,axis));
uilabel(axis,'Text','File to calibrate: ','Position',[10
2+27*partition_height buttons_width_real partition_height+1]);
arxiu_cal = uitextarea(axis, 'Position',[((screen_width/5)+10)/2
2+27*partition_height buttons_width/2 partition_height+4]);
import_cal = uibutton(axis, 'Text', 'Select File', 'Position', [70
2+25*partition_height buttons_width_real-100 partition_height+5],
'ButtonPushedFcn', @(import_cal,event)
click_import_data_cal(import_cal,axis));
calibrate = uibutton(axis, 'Text', 'Calibrate', 'Position', [70
2+23*partition_height buttons_width_real-100 partition_height+5],
'ButtonPushedFcn', @(calibrate,event)
click_calibrate(calibrate,axis));

end

% Callback // Click Select File (calibrate)
function click_import_data_cal(import_cal, axis)
global path;
global filename;
global arxiu_cal;
global tiempo_cal;
global frecuencia_cal;
global magnitud_cal;
global fase_cal;

[filename, path] = uigetfile();
leer = [path, filename];
arxiu_cal.Value = leer;

try
    datos_cal = readtable(leer);
    tiempo_cal = datos_cal.Time;
    frecuencia_cal = datos_cal.Frequency;
```

```
    magnitud_cal = datos_cal.Magnitude;
    fase_cal = datos_cal.Phase;

catch excepcio
    uiconfirm(axis, 'Incorrect Format File', 'Error', 'Options',
{'OK'}, 'icon', 'error');
end
end

% Callback // Click Select File (reference)
function click_import_data_ref(import_ref, axis)
global path;
global filename;
global arxiu_ref;
global tiempo_ref;
global frecuencia_ref;
global magnitud_ref;
global fase_ref;

[filename, path] = uigetfile();
leer = [path, filename];
arxiu_ref.Value = leer;

try
    datos_ref = readtable(leer);
    tiempo_ref = datos_ref.Time;
    frecuencia_ref = datos_ref.Frequency;
    magnitud_ref = datos_ref.Magnitude;
    fase_ref = datos_ref.Phase;
catch excepcio
    uiconfirm(axis, 'Incorrect Format File', 'Error', 'Options',
{'OK'}, 'icon', 'error');
end
end

% Callback // Click Calibrate
function click_calibrate(calibrate,axis)
global axis;
global ax_up;
global ax_down;
global magnitud_ref;
global fase_ref;
global frecuencia_cal;
global magnitud_cal;
global fase_cal;
global rcal;

rcal_num = typecast((single(str2double(rcal.Value))), 'single');

% Magnitude graph
mag_final = magnitud_cal.*(rcal_num./magnitud_ref);
hold (ax_up, 'on');
semilogx(ax_up, frecuencia_cal, magnitud_cal);
semilogx(ax_up, frecuencia_cal, mag_final);
hold (ax_up, 'off');

% Phase graph
fase_final = fase_cal-fase_ref;
hold (ax_down, 'on');
semilogx(ax_down, frecuencia_cal, fase_cal);
```

```
semilogx(ax_down, frecuencia_cal, fase_final);
hold (ax_down, 'off');

% Save calibrated data
fitxer = table(frecuencia_cal, mag_final, fase_final);
writetable(fitxer,'C:\Users\34638\Documents\UPC\TFG\Results_Calibracio
.txt', 'WriteMode', 'Append');
uiconfirm(axis, 'File saved!', 'Success', 'Options', {'OK'}, 'icon',
'success');

end
```

Model adjusting

```
function GUI_Inicialization()
% Global variables
global arxiu_adjust;
global ax;

% Cleaning previous graphs
close all;

% Graphs
axis = uifigure;
axis.Name = 'Model Adjusting Interface';

% Axis
ax = uiaxes(axis);
ax.Title.String = 'Arc de cole';
ax.XLabel.String = 'Part real';
ax.YLabel.String = '-Part imaginaria';

% Buttons
uilabel(axis,'Text','File to adjust: ','Position',[50 310 150 20]);
arxiu_adjust = uitextarea(axis, 'Editable', 'off', 'Position',[130 310
100 20]);
import_file = uibutton(axis, 'Text', 'Select File', 'Position', [250
310 100 20], 'ButtonPushedFcn', @(import_file,event)
click_import_file(import_file,axis));

end

% Callback // Click Select File
function click_import_file(import_file, axis)
global arxiu_adjust;
global ax;

[filename, path] = uigetfile();
leer = [path, filename];
arxiu_adjust.Value = leer;

try
    datos_adjust = readtable(leer);
    tiempo_adjust = datos_adjust.Time;
    frecuencia_adjust = datos_adjust.Frequency;
    magnitud_adjust = datos_adjust.Magnitude;
    fase_adjust = datos_adjust.Phase;

catch excepcio
```

```
    uiconfirm(axis, 'Incorrect Format File', 'Error', 'Options',
{'OK'}, 'icon', 'error');
end

% Convert polar to binomial
fase_radians = deg2rad(fase_adjust);
num_complex = magnitud_adjust.*exp(j.*fase_radians);
real_part = real(num_complex);
imag_part = imag(num_complex);

% Cole Cole function
[ro, ri, alf, fc, ecm2] = colez(real_part, imag_part,
frecuencia_adjust, ax);

% Cole Cole with the results
z_cole = [];
for i=1:length(frecuencia_adjust)
    valor = ri+((ro-ri)/(1+(j*(frecuencia_adjust(i)/fc))^alf));
    z_cole = [z_cole, valor];
end

end
```

Cole-Cole Arc

```
function [ro,ri,alf,fc,ecm2]=colez(r,i,f,ax);
%
% Función de Cole para impedancias.
% Outputs:
% ro=resistencia cero
% ri=resistencia infinita
% alf=ángulo alfa
% fc=frecuencia central
% ecm=error cuadrático medio
% Inputs:
% r= parte real del vector de impedancias
% i= parte imaginaria del vector de impedancias
% f= vector de frecuencias
%
% Cálculo del centro y el radio de la circunferencia aprox.
%
n=length(f);
for ii=1:4
    for j=1:4
    m(ii,j)=mean(r.^(ii-1).* i.^(j-1));
    end
end
sigmx2=m(3,1)-m(2,1)^2;
sigmxy=m(2,2)-m(1,2)*m(2,1);
sigmy2=m(1,3)-m(1,2)^2;
t1=(-m(4,1)+m(2,1)*m(3,1)-m(2,3)+m(2,1)*m(1,3));
t2=(-m(1,4)+m(1,2)*m(1,3)-m(3,2)+m(1,2)*m(3,1));
t3=sigmx2*sigmy2-sigmxy^2;
a=-0.5*(sigmy2*t1-sigmxy*t2)/t3;
b=-0.5*(sigmx2*t2-sigmxy*t1)/t3;
radius=mean(sqrt((r-a).^2+(i-b).^2));
%
% Cálculo de ro,ri,alf
%
```

```
ro=a+(radius^2-b^2)^0.5;
ri=a-(radius^2-b^2)^0.5;
alf=1-(2/pi)*asin((ro-ri)/(2*radius));
%
% Cálculo de fc
%
u=((r.^2-2*ri.*r+i.^2)+(ri^2)).^0.5;
v=((r.^2-2*ro.*r+i.^2)+(ro^2)).^0.5;
ww=log10(abs(u./v));
p=log10(2*pi*f);
h=polyfit(p,ww,1);
fc=((1/(2*pi))*10^(-h(2)/h(1)))/1000;
%
% Cálculo del error cuadrático medio
%
ye=(-b+(radius^2.-(r-a).^2).^0.5);
er=(abs(ye)-abs(i))./radius;
ecm=(((sum(er.^2))^0.5)/n)*100;
er2=(abs(ye)-abs(i))./abs(i);
ecm2=(((sum(er2.^2))^0.5)/n)*100;
ecm3=(sum(abs(er)))*100/n;
%
% Dibujo del arco de cole y los valores medidos
%
xx=[abs(ri):abs(ro)];
yy=(-b+(radius^2.-(xx-a).^2).^0.5);
plot(ax,xx,yy)
hold (ax,'on')
plot(ax,r,-i,'*')
ax.XLim = [ri-0.10*ro ro+0.10*ro]
ax.YLim = [0 max(yy)+0.10*max(yy)]
daspect(ax,[1 1 1])
rro=num2str(ro);
rri=num2str(ri);
alff=num2str(alf);
fcc=num2str(fc);
ecmm=num2str(ecm2);
v1=text(ax,ri-100,-max(yy)*0.12,['Ro : ',rro,' ohms      Rinf :
',rri,' ohms     alfa :' ,alff, ], 'Units', 'normalized', 'Position',
[0 -0.5]);
set(v1,'Clipping','off');
v4=text(ax,ri+450,-max(yy)*0.12,['fc : ',fcc,' kHz      ecm :', ecmm,
' %'], 'Units', 'normalized', 'Position', [0 -0.65]);
set(v4,'Clipping','off');
hold (ax,'off')
```