APENDICE A

LISTADO DEL PROGRAMA

A continuación se muestra el programa de la nueva interfase del método de Remez.

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

% Last Modified by GUIDE v2.5 13-Dec-2003 01:52:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @REMVAR_OpeningFcn, ...
                   'gui_OutputFcn',  @REMVAR_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(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 REMVAR is made visible.
function REMVAR_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to REMVAR (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = REMVAR_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function unidadfrec_CreateFcn(hObject, eventdata, handles)
% hObject    handle to unidadfrec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in unidadfrec.
function unidadfrec_Callback(hObject, eventdata, handles)
% hObject    handle to unidadfrec (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 unidadfrec contents as cell array
%        contents{get(hObject,'Value')} returns selected item from unidadfrec

d = [handles.fmuestreo,handles.muestreo];
unidadfrec = get(hObject,'Value');

if unidadfrec == 2
    set(d,'Enable','on')
elseif unidadfrec == 1
    set(d,'Enable','off')
end

% --- Executes during object creation, after setting all properties.
function fmuestreo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fmuestreo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function fmuestreo_Callback(hObject, eventdata, handles)
% hObject    handle to fmuestreo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fmuestreo as text
%        str2double(get(hObject,'String')) returns contents of fmuestreo as a double


% --- Executes during object creation, after setting all properties.
function vectorfrec_CreateFcn(hObject, eventdata, handles)
% hObject    handle to vectorfrec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function vectorfrec_Callback(hObject, eventdata, handles)
% hObject    handle to vectorfrec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of vectorfrec as text
%        str2double(get(hObject,'String')) returns contents of vectorfrec as a double


% --- Executes on button press in polocero.
function polocero_Callback(hObject, eventdata, handles)
% hObject    handle to polocero (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

b = handles.coeficientes;
bandera_cuantizacion = handles.bandera_cuantizacion;
if bandera_cuantizacion == 1    % cuantizacion encendida
    cla
```

```
   Hq = handles.Hq;
   figure
   zplane(Hq)
elseif bandera_cuantizacion == 0
   zplane(b,1)
end
title('Pole & Zero Plot')


% --- Executes on button press in retardogrupo.
function retardogrupo_Callback(hObject, eventdata, handles)
% hObject    handle to retardogrupo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
b = handles.coeficientes;
  grpdelay(b,1);

title('Group Delay Plot')

% --- Executes on button press in impulso.
function impulso_Callback(hObject, eventdata, handles)
% hObject    handle to impulso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

b = handles.coeficientes;
picho = handles.bandera_cuantizacion;
 if picho == 1  %cuantizacion encendida
   cla
   Hq = handles.Hq;
   figure
   impz(Hq)
elseif picho == 0
   impz(b,1);
end

    %leyendas
   title('Impulse Response')
   ylabel('Amplitude')
   xlabel('Samples  ')

% --- Executes on button press in fase.
function fase_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to fase (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

unidadfrec = get(handles.unidadfrec,'Value');
bandera = handles.bandera;

if bandera == 0         %no compara
   bandera_cuantizacion = handles.bandera_cuantizacion;

      b = handles.coeficientes;
      if unidadfrec == 1          % frec normalizada
         fmuestreo = 2;      %frec normalizada
      elseif unidadfrec == 2      %frec no normalizada
         fmuestreo = str2num(get(handles.fmuestreo,'String'));
      end
      [h,f] = freqz(b,1,512,fmuestreo);
      %angulo = unwrap(angle(h)*180/pi);   %Fase en grados continua
      s.plot = 'phase';
      % %leyendas
      if unidadfrec == 1
         s.xunits= 'Normalized Frequency (x pi rad/Sample)';
      elseif unidadfrec == 2
         s.xunits = 'Frecuencia en Hz';
      end
      freqzplot(h,f,s);
      title('Phase Response')
      ylabel('Degrees')
      grid on;


elseif bandera == 1     %si compara
   b = handles.coeficientes;
   b1 = handles.coeficientes1;
   if unidadfrec == 1          % frec normalizada
       fmuestreo = 2;
   elseif unidadfrec == 2      %frec no normalizada
       fmuestreo = handles.fmuestra;
   end

   [h1,f] = freqz(b,1,512,fmuestreo);
   h2 = freqz(b1,1,512,fmuestreo);
```

```
     %leyendas
    s.plot = 'phase';

    if unidadfrec == 1
        s.xunits= 'Normalized Frequency (x pi rad/Sample)';
    elseif unidadfrec == 2
        s.xunits = 'Frequency Hz';
    end
    h = [h1 h2];
    freqzplot(h,f,s);
    legend('Initial filter','New filter')
    title('Phase Response')
    ylabel('Degrees')
    grid on;

elseif bandera == 2

    h1 = handles.filtroh;
    g1 = handles.filtrog;
    [Hh,w]=freqz(h1,1,512); Hg=freqz(g1,1,512);
    H = Hh.*Hg; % Compounded response
    cla
    freqzplot(H,w,'phase');
    legend('Overall Filter');
    figure
    freqzplot([Hh,Hg],w,'phase');
    legend('Periodic Filter','Image Suppressor Filter');


elseif bandera == 3
    d1 = handles.filtrod;
    h1 = handles.filtroh;
    g1 = handles.filtrog;
    [Hh,w]=freqz(h1,1,512); Hg=freqz(g1,1,512);
    H = Hh.*Hg; % Compounded response
    Hd = freqz(d1,1,512); % Branch 2 response
    Hoverall = H+Hd;
    cla
    freqzplot(Hoverall,w,'phase');
    legend('Overall Filter');
    figure
    freqzplot([Hh,Hg],w,'phase');
    legend('Periodic Filter','Image Suppressor Filter');
```

```
end


% --- Executes on button press in magnitud.
function magnitud_Callback(hObject, eventdata, handles)
% hObject    handle to magnitud (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

bandera = handles.bandera;
unidadfrec = get(handles.unidadfrec,'Value');
bandera_cuantizacion =handles.bandera_cuantizacion;

if bandera == 0     % NO mantener valores de filtro


    if bandera_cuantizacion == 1
      cla
      Hq = handles.Hq;
      figure
      freqz(Hq)

   elseif bandera_cuantizacion == 0

      b = handles.coeficientes;
      if unidadfrec == 1          % frec normalizada
         fmuestreo = 2;      %frec normalizada
      elseif unidadfrec == 2      %frec no normalizada
         fmuestreo = str2num(get(handles.fmuestreo,'String'));
      end
      [h,f] = freqz(b,1,512,fmuestreo);
      mag = abs(h);
      plot(f,mag); grid
      %leyendas
      title('Magnitude Response')
      ylabel('Magnitude')
      if unidadfrec == 1
        xlabel('Normalized Frequency (x pi rad/Sample)')
      elseif unidadfrec == 2
         xlabel('Frequency Hz')
      end
```

```
        handles.mag = mag;
        handles.f = f;

    end

elseif bandera == 1   %Mantener valores de filtro
    b = handles.coeficientes;
    b1 = handles.coeficientes1;

    if unidadfrec == 1          % frec normalizada
        fmuestreo = 2;
        fmuestreo1 = 2;%frec normalizada
    elseif unidadfrec == 2      %frec no normalizada
         fmuestreo = handles.fmuestra;
         fmuestreo1 = handles.fmuestra1;
    end

    [h,f] = freqz(b,1,512,fmuestreo);
    mag = abs(h);
    [h1,f1] = freqz(b1,1,512,fmuestreo1);
    mag1 = abs(h1);
    plot(f,mag,f1,mag1); grid

      %leyendas
    title('Magnitude Response')
    ylabel('Magnitude')
    legend('Initial Filter','New Filter')
    if unidadfrec == 1
        xlabel('Normalized Frequecy (x pi rad/Sample')
    elseif unidadfrec == 2
        xlabel('Frequency Hz')
    end

    handles.mag =mag;
    handles.mag1 =mag1;
    handles.f = f;
    handles.f1 = f1;

elseif bandera == 2

    h1 = handles.filtroh;
    g1 = handles.filtrog;
```

```
   [Hh,w]=freqz(h1,1,512); Hg=freqz(g1,1,512);
   H = Hh.*Hg; % Compounded response
   cla
   s.plot   = 'mag';     % Plot magnitude only
   s.yunits = 'linear'; % Plot the magnitude squared
   freqzplot(H,w,s);
   legend('Overall Filter');
   figure
   freqzplot([Hh,Hg],w,'mag');
   legend('Periodic Filter','Image Suppressor Filter');
   handles.H = H;
   handles.w = w;


elseif bandera == 3
   d1 = handles.filtrod;
   h1 = handles.filtroh;
   g1 = handles.filtrog;
   [Hh,w]=freqz(h1,1,512); Hg=freqz(g1,1,512);
   H = Hh.*Hg; % Compounded response
   Hd = freqz(d1,1,512); % Branch 2 response
   Hoverall = H+Hd;
   cla
   s.plot   = 'mag';     % Plot magnitude only
   s.yunits = 'linear'; % Plot the magnitude squared
   freqzplot(Hoverall,w,s);
   legend('Overall Filter');
   figure
   freqzplot([Hh,Hg],w,'mag');
   legend('Periodic Filter','Image Suppressor Filter');
   handles.Hoverall = Hoverall;
   handles.w = w;

end



guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function tipofiltro_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to tipofiltro (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in tipofiltro.
function tipofiltro_Callback(hObject, eventdata, handles)
% hObject    handle to tipofiltro (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 tipofiltro contents as cell array
%        contents{get(hObject,'Value')} returns selected item from tipofiltro

tipofiltro = get(hObject,'Value');
ordenfiltro = get(handles.ordenfiltro,'Value');
encuentraorden = get(handles.encuentraorden,'Value');
ordenminimo = get(handles.ordenminimo, 'Value');
a = [handles.vectorpicos,handles.vectorpicosrizo];
b = [handles.vectorpeso,handles.vecpeso];
c = [handles.inicialn,handles.ni];
d = [handles.ordenfiltro,handles.ordenminimo,handles.encuentraorden,handles.n];
set(a,'Visible','on')
set(b,'Visible','on')
set(c,'Enable','on')
set(d,'Enable','on')
set(handles.vectoramp,'Visible','on')
set(handles.text7,'Visible','on')
set(handles.unidadfrec,'Enable','on')
set(handles.frectext,'Enable','on')
set(handles.interpfac,'Visible','off')
set(handles.factorinterpolacion,'Visible','off')
set(handles.mantenervalores,'Enable','on')

M = 'nada';
```

```
ftype = 'nada';
set(c,'visible','off')

switch tipofiltro
   case 1
      set(handles.encuentraorden,'Enable','on') %habilita encuentra orden
      set(handles.prueba,'String',' ')

      if ordenfiltro == 1
        set(b,'Visible','on')
        set(a,'Visible','off')
     elseif encuentraorden == 1
        set(a,'Visible','on')
        set(b,'Visible','off')
     elseif ordenminimo == 1
        set(a,'Visible','on')
        set(b,'Visible','off')
        M = 'minorder';
     end
   case 2
      set(handles.encuentraorden,'Enable','off')
      set(handles.encuentraorden,'Value',0)
      set(handles.ordenfiltro,'Value',1)
      set(handles.ordenminimo,'Value',0)
      set(handles.prueba,'String',' ')

      if ordenfiltro == 1
         set(a,'Visible','off')
         set(b,'Visible','off')
         ftype = 'differentiator';
      elseif ordenminimo == 1
         set(b,'Visible','off')
         set(a,'Visible','on')
         set(c,'Visible','on')
      elseif encuentraorden == 1
         set(b,'Visible','off')
      end
   case 3
      set(handles.encuentraorden,'Enable','off')
      set(handles.encuentraorden,'Value',0)
      set(handles.ordenfiltro,'Value',1)
      set(handles.ordenminimo,'Value',0)
      set(handles.prueba,'String',' ')
      if ordenfiltro == 1
         set(a,'Visible','off')
         set(b,'Visible','off')
         ftype = 'hilbert';
      elseif ordenminimo == 1
         set(b,'Visible','off')
         set(a,'Visible','on')
         set(c,'Visible','on')
      elseif encuentraorden == 1
         set(b,'Visible','off')
      end

   case 4

      set(d,'Enable','off')
      set(handles.vectoramp,'Visible','off')
      set(handles.vectorpeso,'Visible','off')
      set(handles.cuantizacion,'Enable','off')
      set(handles.unidadfrec,'Enable','off')
      set(handles.text7,'Visible','off')
      set(handles.vecpeso,'Visible','off')
      set(handles.frectext,'Enable','off')
      set(a,'Visible','on')
      set(handles.factorinterpolacion,'Visible','on')
      set(handles.interpfac,'Visible','on')
      set(handles.mantenervalores,'Enable','off')
      set(handles.prueba,'String',strvcat('Specify the filter band ','edge frequencies in vector f',' ','F and the Peak Ripple Vector',' must have the same length'))

   case 5
      set(d,'Enable','off')
      set(handles.vectoramp,'Visible','off')
      set(handles.vectorpeso,'Visible','off')
      set(handles.cuantizacion,'Enable','off')
      set(handles.unidadfrec,'Enable','off')
      set(handles.text7,'Visible','off')
      set(handles.vecpeso,'Visible','off')
      set(handles.frectext,'Enable','off')
      set(a,'Visible','on')
      set(handles.factorinterpolacion,'Visible','on')
      set(handles.interpfac,'Visible','on')
      set(handles.mantenervalores,'Enable','off')
```

```
    set(handles.prueba,'String',strvcat('Specify the filter band ','edge frequencies
in vector f',' ','F and the Peak Ripple Vector',' must have the same length'))


end




% --- Executes on button press in ordenfiltro.
function ordenfiltro_Callback(hObject, eventdata, handles)
% hObject    handle to ordenfiltro (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ordenfiltro
set(handles.prueba,'String','')

off = [handles.ordenminimo,handles.encuentraorden];
set(off,'Value',0)

 ordenfiltro = get(hObject,'Value');
 tipofiltro = get(handles.tipofiltro,'Value');
a = [handles.vectorpicos,handles.vectorpicosrizo];
b = [handles.vectorpeso,handles.vecpeso];
c = [handles.inicialn,handles.ni];


switch tipofiltro
    case 1
        if ordenfiltro == 1
            set(a,'Visible','off')
            set(b,'Visible','on')
            set(c,'visible','off')
        end
    case 2
        if ordenfiltro == 1
            set(b,'Visible','off')
            set(c,'visible','off')
            ftype = 'differentiator';
        end
    otherwise
```

```
      if ordenfiltro == 1
         set(a,'Visible','off')
         set(b,'Visible','off')
         set(c,'visible','off')
         ftype = 'Hilbert';
      end
end



% --- Executes during object creation, after setting all properties.
function n_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function n_Callback(hObject, eventdata, handles)
% hObject    handle to n (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n as text
%        str2double(get(hObject,'String')) returns contents of n as a double


% --- Executes on button press in ordenminimo.
function ordenminimo_Callback(hObject, eventdata, handles)
% hObject    handle to ordenminimo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of ordenminimo
set(handles.prueba,'String','')
```

```
off = [handles.ordenfiltro,handles.encuentraorden];
set(off,'Value',0)


ordenminimo = get(hObject,'Value');
tipofiltro = get(handles.tipofiltro,'Value');
a = [handles.vectorpicos,handles.vectorpicosrizo];
b = [handles.vectorpeso,handles.vecpeso];
c = [handles.inicialn,handles.ni];
M = 'minorder';

switch tipofiltro
   case 1
      if ordenminimo == 1
         set(a,'Visible','on')
         set(b,'Visible','off')
         set(c,'visible','off')
      end
   case 2
      if ordenminimo == 1
         set(a,'Visible','on')
         set(b,'Visible','off')
         set(c,'Visible','on')
      end
   case 3
      if ordenminimo == 1
         set(a,'Visible','on')
         set(b,'Visible','off')
         set(c,'Visible','on')
      end
end


% --- Executes on button press in encuentraorden.
function encuentraorden_Callback(hObject, eventdata, handles)
% hObject    handle to encuentraorden (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of encuentraorden

off = [handles.ordenminimo,handles.ordenfiltro];
set(off,'Value',0)
```

```
encuentraorden = get(hObject,'Value');
a = [handles.vectorpicos,handles.vectorpicosrizo];
b = [handles.vectorpeso,handles.vecpeso];
c = [handles.inicialn,handles.ni];

if encuentraorden == 1
   set(a,'Visible','on')
   set(b,'Visible','off')
   set(c,'visible','off')
   set(handles.prueba,'String',strvcat('Length of Frequency Vector must be','2*(A)-
2, where A is the length of the ','Amplitude Vector.','Type "help remezord" for more
information.'))
end



% --- Executes during object creation, after setting all properties.
function vectoramp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to vectoramp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function vectoramp_Callback(hObject, eventdata, handles)
% hObject    handle to vectoramp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of vectoramp as text
%        str2double(get(hObject,'String')) returns contents of vectoramp as a double


% --- Executes during object creation, after setting all properties.
function vectorpeso_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to vectorpeso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function vectorpeso_Callback(hObject, eventdata, handles)
% hObject    handle to vectorpeso (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of vectorpeso as text
%        str2double(get(hObject,'String')) returns contents of vectorpeso as a double


% --- Executes during object creation, after setting all properties.
function vectorpicosrizo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to vectorpicosrizo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function vectorpicosrizo_Callback(hObject, eventdata, handles)
% hObject    handle to vectorpicosrizo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of vectorpicosrizo as text
%          str2double(get(hObject,'String')) returns contents of vectorpicosrizo as a
double


% --- Executes during object creation, after setting all properties.
function ni_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ni (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function ni_Callback(hObject, eventdata, handles)
% hObject    handle to ni (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ni as text
%        str2double(get(hObject,'String')) returns contents of ni as a double



% --- Executes on button press in design.
function design_Callback(hObject, eventdata, handles)
% hObject    handle to design (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

on                                                                          =
[handles.magnitud,handles.fase,handles.limpiar,handles.retardogrupo,handles.pol
ocero,handles.impulso,handles.polocerotexto,handles.tf,handles.cuantizacion];
set(on,'Enable','on')

handles.bandera_cuantizacion = 0;
```

```
bandera1 = 0;
%obtenemos estado de las interfaces
tipofiltro = get(handles.tipofiltro,'Value');
ordenfiltro = get(handles.ordenfiltro,'Value');
encuentraorden = get(handles.encuentraorden,'Value');
ordenminimo = get(handles.ordenminimo,'Value');

%determinamos si f esta normalizada o no
f = eval(get(handles.vectorfrec,'String'));
unidadfrec = get(handles.unidadfrec,'Value');
fmuestreo = str2num(get(handles.fmuestreo,'String'));


if unidadfrec == 1

   fmuestreo = 2;      %frec normalizada
   fn = f;
elseif unidadfrec == 2

   f1 = f;             % almacenamos f
   fn = 2*f/fmuestreo;  %normalizamos f
end


% Determinamos que tipo de filtro se va a diseñar, obtenemos los datos
% definidos por el usuario y posteriormente se calculan los coeficientes
% del filtro

if (tipofiltro == 1) & (ordenfiltro == 1)
   %gremez sencillo
   %b = gremez(n,f,a,w)

   % Obtenemos parametros definidos por el usuario
   n = str2num(get(handles.n,'String'));
   f = fn;
   a = eval(get(handles.vectoramp,'String'));
   w = eval(get(handles.vectorpeso,'String'));


   %Verificamos que los datos introducidos tengan el formato correcto
   lengthf = length(f);
   lengtha = length(a);
   lengthf = lengthf/2;
   lengtha = lengtha/2;
   lenta = int8(lengtha);
   lentf = int8(lengthf);
   lenta = double(lenta);
   lentf = double(lentf);
   resf = lengthf - lentf;
   resa = lengtha - lenta;

   if unidadfrec == 1

   maximo = max(f);         %determinamos si los datos del vector de frecuencia se
encuentran dentro de rango
   minimo = min(f);
      if maximo > 1
         errordlg('Frequencies must lie between 0 and 1','Frequency error')
         return
         %elseif minimo > 0
         %errordlg('Frequencies must lie between 0 and f sampling/2','Frequency
error')
         %return
      end
   elseif unidadfrec == 2

      maximo = max(f);         %determinamos si los datos del vector de frecuencia
se encuentran dentro de rango
      minimo = min(f);
      if maximo > fmuestreo/2
         errordlg('Frequencies must lie between 0 and f sampling/2','Frequency
error')
         return
         %elseif minimo > 0
         %errordlg('Frequencies must lie between 0 and f sampling/2','Frequency
error')
         %return
      end
   end


   if n < 3          %verificamos orden mayor a 2
       errordlg('Filter order must be 3 or more','Order Error')
       return
    elseif length(f) ~= length(a)  %vectores de amplitud y fre. deben ser del mismo
tamaño
```

```
        errordlg('Frequency and Amplitude vectors must be the same size, with even
length','Input Data Error')
        return
    elseif resf ~= 0
        errordlg('The number of frequency points must be even','Frequency Error')
        return
    elseif resa ~= 0
        errordlg('The number of frequency points must be even','Frequency Error')
        return
    elseif length(w) ~= length(a)/2
        errordlg('There should be one weight per band','Weight Vector Error')
        return
    elseif min(w) <= 0
        errordlg('All weights must be positive greater than zero','Weight Vector Error')
        return
    end

    le = length(f);    %verificamos vector de frecuencias en orden ascendente
    for i = 2:le
        if f(i) < f(i-1)
            errordlg('Frequencies must be non-decreasing','Frequency error')
            return
        end
    end



   %calculamos los coeficientes del filtro
   b = gremez(n,f,a,w);
   %salvamos los coeficientes en la estructura handles

elseif (tipofiltro == 1) & (encuentraorden == 1)
   %remezord
   % [N,Fo,Ao,W] = remezord(f,a,DEV,Fs)
   %Obtenemos parametros definidos por el usuario
   if unidadfrec == 1
      maximo = max(f);          %determinamos si los datos del vector de frecuencia
se encuentran dentro de rango
      minimo = min(f);
      if maximo > 1
         errordlg('Frequencies must lie between 0 and 1','Frequency error')
         return
      end
      f = fn;
   else unidadfrec == 2
      maximo = max(f);        %determinamos si los datos del vector de frecuencia se encuentran dentro de rango
      minimo = min(f);
      if maximo > fmuestreo/2
         errordlg('Frequencies must lie between 0 and f sampling/2','Frequency error')
         return
      end
      f = fn*fmuestreo/2;
   end



   a = eval(get(handles.vectoramp,'String'));
   DEV = eval(get(handles.vectorpicosrizo,'String'));
   Fs = fmuestreo;
   % verificamos los datos del usuario
   if length(f) ~= 2*length(a)-2
      errordlg('Length of F must be 2*length(A)-2.','Frequency error')
      return
   elseif  length(a) ~= length(DEV)
      errordlg('A and Peak Ripple must be vectors of the same length.','Input data error')
      return
   end

   le = length(f);    %verificamos vector de frecuencias en orden ascendente
    for i = 2:le
       if f(i) < f(i-1)
          errordlg('Frequencies must be non-decreasing','Frequency error')
          return
       end
    end
   %calculamos los coeficientes del filtro
   [N,Fo,Ao,W] = remezord(f,a,DEV,Fs);
   b = gremez(N,Fo,Ao,W);
   bandera1 = 1;

elseif (tipofiltro == 1) & (ordenminimo == 1)
   %filtro de orden minimo
```

```
% b = gremez(M,f,a,R)

   f = fn;
   a = eval(get(handles.vectoramp,'String'));
   R = eval(get(handles.vectorpicosrizo,'String'));
   %determinamos si el guey metio bien el dato,
   minimo = min(f);
   maximo = max(f)*fmuestreo/2;
   lengthf = length(f);
   lengtha = length(a);
   lengthf = lengthf/2;
   lengtha = lengtha/2;
   lenta = int8(lengtha);
   lentf = int8(lengthf);
   lenta = double(lenta);
   lentf = double(lentf);
   resf = lengthf - lentf;
   resa = lengtha - lenta;

   if minimo ~= 0
      errordlg('Can only estimate order if F(1)=0 and F(end)=1','Input Data Error')
       return
   elseif (unidadfrec == 1) & (maximo > 1)
      errordlg('Can only estimate order if F(1)=0 and F(end)=1','Input Data Error')
       return
   elseif (unidadfrec == 2) & (maximo > fmuestreo/2)
      errordlg('Can only estimate order if F(1)=0 and F(end)=fmuestreo/2','Input
Data Error')
       return
   elseif length(f) ~= length(a)  %vectores de amplitud y fre. deben ser del mismo
tamaño
       errordlg('Frequency and Amplitude vectors must be the same size, with even
length','Input Data Error')
       return
    elseif resf ~= 0
       errordlg('The number of frequency points must be even','Frequency Error')
       return
    elseif resa ~= 0
       errordlg('The number of Amplitude points must be even','Frequency Error')
       return
    elseif length(R) ~= length(a)/2
       errordlg('There should be one peak ripple per band','Weight Vector Error')
       return
```

```
     elseif min(R) <= 0
        errordlg('All weights must be positive greater than zero','Weight Vector Error')
        return
     end

     le = length(f);    %verificamos vector de frecuencias en orden ascendente
     for i = 2:le
        if f(i) < f(i-1)
           errordlg('Frequencies must be non-decreasing','Frequency error')
           return
        end
     end
    % obtenemos los coeficientes del filtro

    b = gremez('minorder',f,a,R);


elseif (tipofiltro == 2) & (ordenfiltro == 1)
    %gremez diferenciador
    %b = gremez(n,f,a,'differentiador')

    %Obtenemos parametros definidos por el usuario
    n = str2num(get(handles.n,'String'));
    f = fn;
    a = eval(get(handles.vectoramp,'String'));

    %determinamos si el guey metio bien el dato,
    minimo = min(f);
    maximo = max(f)*fmuestreo/2;
    lengthf = length(f);
    lengtha = length(a);
    lengthf = lengthf/2;
    lengtha = lengtha/2;
    lenta = int8(lengtha);
    lentf = int8(lengthf);
    lenta = double(lenta);
    lentf = double(lentf);
    resf = lengthf - lentf;
    resa = lengtha - lenta;


    if (unidadfrec == 1) & (maximo > 1)
       errordlg('Can only estimate order if  F(end)=1','Input Data Error')
```

```
        return
    elseif (unidadfrec == 2) & (maximo > fmuestreo/2)
        errordlg('Can only estimate order if F(end)=fmuestreo/2','Input Data Error')
        return
    elseif length(f) ~= length(a)  %vectores de amplitud y fre. deben ser del mismo
tamaño
        errordlg('Frequency and Amplitude vectors must be the same size, with even
length','Input Data Error')
        return
     elseif resf ~= 0
        errordlg('The number of frequency points must be even','Frequency Error')
        return
     elseif resa ~= 0
        errordlg('The number of Amplitude points must be even','Frequency Error')
        return
     elseif n < 3
        errordlg('Filter order must be 3 or more.','Frequency Error')
        return

     end

     le = length(f);    %verificamos vector de frecuencias en orden ascendente
     for i = 2:le
        if f(i) < f(i-1)
           errordlg('Frequencies must be non-decreasing','Frequency error')
           return
        end
     end


    %calculamos los coeficientes del filtro
    b = gremez(n,f,a,'differentiator')
    %salvamos los coeficientes en la estructura handles

elseif (tipofiltro == 2) & (ordenminimo == 1)
    %gremez menor orden diferenciador
    %b = gremez({M,NI},f,a,R)

    f = fn;
    a = eval(get(handles.vectoramp,'String'));
    R = eval(get(handles.vectorpicosrizo,'String'));
    NI = str2num(get(handles.ni,'String'));
    %determinamos si el guey metio bien el dato,
    minimo = min(f);
    maximo = max(f)*fmuestreo/2;
    lengthf = length(f);
    lengtha = length(a);
    lengthf = lengthf/2;
    lengtha = lengtha/2;
    lenta = int8(lengtha);
    lentf = int8(lengthf);
    lenta = double(lenta);
    lentf = double(lentf);
    resf = lengthf - lentf;
    resa = lengtha - lenta;

    if (unidadfrec == 1) & (maximo > 1)
       errordlg('Can only estimate order if  F(end)=1','Input Data Error')
        return
    elseif (unidadfrec == 2) & (maximo > fmuestreo/2)
       errordlg('Can only estimate order if  F(end)=fmuestreo/2','Input Data Error')
        return
    elseif length(f) ~= length(a)  %vectores de amplitud y fre. deben ser del mismo
tamaño
        errordlg('Frequency and Amplitude vectors must be the same size, with even
length','Input Data Error')
        return
     elseif resf ~= 0
        errordlg('The number of frequency points must be even','Frequency Error')
        return
     elseif resa ~= 0
        errordlg('The number of Amplitude points must be even','Frequency Error')
        return
     elseif length(R) ~= length(a)/2
        errordlg('There should be one peak ripple per band','Weight Vector Error')
        return
     elseif min(R) <= 0
        errordlg('All weights must be positive greater than zero','Weight Vector Error')
        return
     elseif NI < 3
        errordlg('Filter order must be 3 or more','Order Error')
        return
     end

     le = length(f);    %verificamos vector de frecuencias en orden ascendente
     for i = 2:le
        if f(i) < f(i-1)
           errordlg('Frequencies must be non-decreasing','Frequency error')
           return
        end
     end

     % Obtenemos los coeficientes del filtro
    b = gremez({'minorder',NI},f,a,R);


elseif (tipofiltro == 3) & (ordenfiltro == 1)
    %gremez hilbert
    %b = gremez(n,f,a,'Hilbert')


    %Obtenemos parametros definidos por el usuario
    n = str2num(get(handles.n,'String'));
    f = fn;
    a = eval(get(handles.vectoramp,'String'));

    %determinamos si el guey metio bien el dato,
    minimo = min(f);
    maximo = max(f)*fmuestreo/2;
    lengthf = length(f);
    lengtha = length(a);
    lengthf = lengthf/2;
    lengtha = lengtha/2;
    lenta = int8(lengtha);
    lentf = int8(lengthf);
    lenta = double(lenta);
    lentf = double(lentf);
    resf = lengthf - lentf;
    resa = lengtha - lenta;


    if (unidadfrec == 1) & (maximo > 1)
       errordlg('Can only estimate order if  F(end)=1','Input Data Error')
        return
    elseif (unidadfrec == 2) & (maximo > fmuestreo/2)
       errordlg('Can only estimate order if F(end)=fmuestreo/2','Input Data Error')
        return
    elseif length(f) ~= length(a)  %vectores de amplitud y fre. deben ser del mismo
tamaño
         errordlg('Frequency and Amplitude vectors must be the same size, with even
length','Input Data Error')
         return
     elseif resf ~= 0
         errordlg('The number of frequency points must be even','Frequency Error')
         return
     elseif resa ~= 0
         errordlg('The number of Amplitude points must be even','Frequency Error')
         return
     elseif n < 3
         errordlg('Filter order must be 3 or more.','Frequency Error')
         return

     end

     le = length(f);    %verificamos vector de frecuencias en orden ascendente
     for i = 2:le
         if f(i) < f(i-1)
             errordlg('Frequencies must be non-decreasing','Frequency error')
             return
         end
     end

    %calculamos los coeficientes del filtro
    b = gremez(n,f,a,'Hilbert');


elseif (tipofiltro == 3) & (ordenminimo == 1)
    %gremez menor orden hilbert
    %b = gremez({M,NI},f,a,R)

    f = fn;
    a = eval(get(handles.vectoramp,'String'));
    R = eval(get(handles.vectorpicosrizo,'String'));
    NI = str2num(get(handles.ni,'String'));
    %determinamos si el guey metio bien el dato,
    minimo = min(f);
    maximo = max(f)*fmuestreo/2;
    lengthf = length(f);
    lengtha = length(a);
    lengthf = lengthf/2;
    lengtha = lengtha/2;
    lenta = int8(lengtha);
    lentf = int8(lengthf);
    lenta = double(lenta);
    lentf = double(lentf);
    resf = lengthf - lentf;
    resa = lengtha - lenta;

    if (unidadfrec == 1) & (maximo > 1)
        errordlg('Can only estimate order if  F(end)=1','Input Data Error')
         return
    elseif (unidadfrec == 2) & (maximo > fmuestreo/2)
        errordlg('Can only estimate order if  F(end)=fmuestreo/2','Input Data Error')
         return
    elseif length(f) ~= length(a)  %vectores de amplitud y fre. deben ser del mismo
tamaño
         errordlg('Frequency and Amplitude vectors must be the same size, with even
length','Input Data Error')
         return
     elseif resf ~= 0
         errordlg('The number of frequency points must be even','Frequency Error')
         return
     elseif resa ~= 0
         errordlg('The number of Amplitude points must be even','Frequency Error')
         return
     elseif length(R) ~= length(a)/2
         errordlg('There should be one peak ripple per band','Weight Vector Error')
         return
     elseif min(R) <= 0
         errordlg('All weights must be positive greater than zero','Weight Vector Error')
         return
     elseif NI < 3
         errordlg('Filter order must be 3 or more','Order Error')
         return
     end
    le = length(f);    %verificamos vector de frecuencias en orden ascendente
     for i = 2:le
         if f(i) < f(i-1)
             errordlg('Frequencies must be non-decreasing','Frequency error')
             return
         end
     end
```

```
   b = gremez({'minorder',NI},f,a,R);

elseif tipofiltro == 4
   % IFIR tipo pasabajas
   %[h,g]= ifir(l,'low',f,DEV)
   f = fn;
   R = eval(get(handles.vectorpicosrizo,'String'));
   factorinterpolacion = str2num(get(handles.factorinterpolacion,'String'));

   %Verificar Datos de usuario
   le = length(f);    %verificamos vector de frecuencias en orden ascendente
    for i = 2:le
       if f(i) < f(i-1)
          errordlg('Frequencies must be non-decreasing','Frequency error')
          return
       end
    end


   % obtenemos filtro IFIR
   [h1,g1]= ifir(factorinterpolacion,'low',f,R);

   %desplegamos la funcion de transferencia
   tf = poly2sym(h1,'z');
   clc
   disp('Transfer Function of the Periodic Filter')
   pretty(tf)

   disp('Transfer Function of the Suppressor Filter')
   tf = poly2sym(g1,'z');
   pretty(tf)


elseif  tipofiltro == 5
    % IFIR tipo  pasaaltas
   %[h,g]= ifir(l,'high',f,DEV)
   f = fn;
   R = eval(get(handles.vectorpicosrizo,'String'));
   factorinterpolacion = str2num(get(handles.factorinterpolacion,'String'));


   %Verificar Datos de usuario
   le = length(f);    %verificamos vector de frecuencias en orden ascendente
```

```
     for i = 2:le
         if f(i) < f(i-1)
             errordlg('Frequencies must be non-decreasing','Frequency error')
             return
         end
     end

    % obtenemos filtro IFIR
    [h1,g1,d1]= ifir(factorinterpolacion,'high',f,R);

    %desplegamos la funcion de transferencia
    tf = poly2sym(h1,'z');
    clc
    disp('Transfer Function of the Periodic Filter')
    pretty(tf)

    disp('Transfer Function of the Suppressor Filter')
    tf = poly2sym(g1,'z');
    pretty(tf)

    disp('Delay')
    d1


end

%Determinamos si se mantienen los valores para compara dos filtros

mantenervalores = get(handles.mantenervalores,'Value');
d
=[handles.polocero,handles.retardogrupo,handles.polocerotexto,handles.tf,handles
.impulso];
handles.bandera = 0;

if tipofiltro == 4              %filtro IFIR
    handles.filtroh = h1;
    handles.filtrog = g1;
    handles.bandera = 2;
    set(handles.cuantizacion,'Enable','off')
    set(d,'Enable','off')
elseif tipofiltro == 5          %filtro iFIR
    handles.filtroh = h1;
    handles.filtrog = g1;
```

```
   handles.filtrod = d1;
   handles.bandera = 3;
   set(handles.cuantizacion,'Enable','off')
   set(d,'Enable','off')
elseif  mantenervalores == 0    %si tipo filtro = 3,
   set(handles.mantenervalores,'Enable','on')

   handles.coeficientes = b;
   handles.fmuestra = fmuestreo;
   set(d,'Enable','on')
elseif mantenervalores == 1
   set(handles.mantenervalores,'Enable','on')

   handles.coeficientes1 = b;
   handles.fmuestra1 = fmuestreo;
   set(d,'Enable','off')
   set(handles.mantenervalores,'Value',0)
   set(handles.fmuestreo,'Enable','on')
   set(handles.muestreo,'Enable','on')
   handles.bandera = 1;
end

%guardamos los cambios a la estructura handles
guidata(hObject, handles);


if bandera1 == 1

   valores = strvcat('Order = ',num2str(N),' ','Frequency Vector = ',num2str(Fo),'
','Amplitude Vector = ',num2str(Ao),' ','Weight Vector =',num2str(W))
   set(handles.prueba,'String',valores)
elseif handles.bandera == 2
   mensaje = strvcat('The transfer functions','appear at the Matlab ','Command
Window',' ','Use the button from the',' "Type of Graphics"','to plot the response of
the filter.');
   set(handles.prueba,'String',mensaje)
elseif handles.bandera == 3
   mensaje = strvcat('The transfer functions','appear at the Matlab ','Command
Window',' ','Use the button from the',' "Type of Graphics"','to plot the response of
the filter.');
   set(handles.prueba,'String',mensaje)
else
```

```
set(handles.prueba,'String',strvcat('Use the button from the',' "Type of Graphics"','to
plot the response of the filter.'))
end



% --- Executes on button press in polocerotexto.
function polocerotexto_Callback(hObject, eventdata, handles)
% hObject    handle to polocerotexto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

b = handles.coeficientes;
r = roots(b);
set(handles.prueba,'String',num2str(r,3));


% --- Executes on button press in tf.
function tf_Callback(hObject, eventdata, handles)
% hObject    handle to tf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

b = handles.coeficientes;
tf = poly2sym(b,'z');
clc
pretty(tf)
%set(handles.prueba,'String',char(tf));

mensaje = strvcat('The transfer function','appears at the Matlab ','Command
Window');
set(handles.prueba,'String',mensaje)




% --- Executes during object creation, after setting all properties.
function prueba_CreateFcn(hObject, eventdata, handles)
% hObject    handle to prueba (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in prueba.
function prueba_Callback(hObject, eventdata, handles)
% hObject    handle to prueba (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 prueba contents as cell array
%        contents{get(hObject,'Value')} returns selected item from prueba


% --- Executes on button press in ayuda.
function ayuda_Callback(hObject, eventdata, handles)
% hObject    handle to ayuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.prueba,'String',strvcat('TYPE HELP GREMEZ AT THE COMMAND
WINDOW',' ',' ','DEVELOPED BY',' ','    DR. DAVID BAEZ LOPEZ',' ','    JOSE DE
JESUS JAUREGUI VENTURA'))



% --- Executes on button press in salir.
function salir_Callback(hObject, eventdata, handles)
% hObject    handle to salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


delete(handles.figure1)


% --- Executes on button press in mantenervalores.
function mantenervalores_Callback(hObject, eventdata, handles)
% hObject    handle to mantenervalores (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of mantenervalores

unidadfrec = get(handles.unidadfrec,'Value');
mantenervalores = get(hObject,'Value');
if mantenervalores == 1
   a = [handles.fmuestreo,handles.muestreo];
   set(a,'Enable','off')
elseif (mantenervalores == 0) & (unidadfrec == 2)
   a = [handles.fmuestreo,handles.muestreo];
   set(a,'Enable','on')
end

% --------------------------------------------------------------------
function ejes_contextmenu_Callback(hObject, eventdata, handles)
% hObject    handle to ejes_contextmenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --------------------------------------------------------------------
function magdb_Callback(hObject, eventdata, handles)
% hObject    handle to magdb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDAT
unidadfrec = get(handles.unidadfrec,'Value');
bandera = handles.bandera;

if bandera == 0     % NO mantener valores de filtro

   f = handles.f;
   mag = handles.mag;
   mag = 20*log10(mag);
   plot(f,mag); grid

   title('Magnitude Response')
   ylabel('Magnitude (dB)')
   if unidadfrec == 1
      xlabel('Normalized Frequency (x pi rad/Sample)')
   elseif unidadfrec == 2
```

```
        xlabel('Frequency Hz')
    end

elseif bandera == 1   %Mantener valores de filtro
    f = handles.f;
    mag = handles.mag;
    f1 = handles.f1;
    mag1 = handles.mag1;
    mag = 20*log10(mag);
    mag1 = 20*log10(mag1);
    plot(f,mag,f1,mag1); grid

        title('Magnitude Response')
    ylabel('Magnitude (dB)')
    if unidadfrec == 1
        xlabel('Normalized Frequency (x pi rad/Sample')
    elseif unidadfrec == 2
        xlabel('Frequency Hz')
    end

elseif bandera == 2
    H = handles.H;
    w = handles.w;
    s.plot   = 'mag';     % Plot magnitude only
    s.yunits = 'db'; % Plot the magnitude squared
    freqzplot(H,w,s);
    legend('Overall Filter');

elseif bandera == 3
    Hoverall = handles.Hoverall;
    w = handles.w;
    s.plot   = 'mag';     % Plot magnitude only
    s.yunits = 'db'; % Plot the magnitude squared
    freqzplot(Hoverall,w,s);
    legend('Overall Filter');

end


% --- Executes on button press in limpiar.
function limpiar_Callback(hObject, eventdata, handles)
% hObject    handle to limpiar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Reestablecemos los valores iniciales de la ventana
cla
legend off
clc
set(handles.vectorfrec,'String','[0 0.4 0.5 1]')
set(handles.vectoramp,'String','[1 1 0 0]')
set(handles.vectorpeso,'String','[1 1]')
set(handles.vectorpicosrizo,'String','[0.1 0.02]')
set(handles.n,'String','12')
set(handles.ni,'String','12')
set(handles.fmuestreo,'String','1000')
set(handles.magnitud,'String','Magnitude')
set(handles.limitcicle,'Enable','off')       %no permite limite de ciclos
set(handles.overflowmode,'Value',1)
set(handles.roundmode,'Value',5)
set(handles.mode,'Value',1)
set(handles.format,'String','[16 15]')

set(handles.prueba,'String','')

set(handles.unidadfrec,'Value',1)
set(handles.tipofiltro,'Value',1)
set(handles.ordenfiltro,'Value',1)
set(handles.vectorpicosrizo,'Visible','off')
set(handles.vectorpicos,'Visible','off')
off = [handles.ordenminimo,handles.encuentraorden];
set(off,'Value',0)
set(handles.ni,'Visible','off')
set(handles.inicialn,'Visible','off')
set(handles.mantenervalores,'Enable','off')

%apagar graficacion
set(handles.muestreo,'Enable','off')
set(handles.fmuestreo,'Enable','off')
set(handles.encuentraorden,'Enable','on')
set(handles.magnitud,'Enable','off')
set(handles.fase,'Enable','off')
set(handles.impulso,'Enable','off')
set(handles.polocero,'Enable','off')
set(handles.retardogrupo,'Enable','off')
set(handles.polocerotexto,'Enable','off')
set(handles.tf,'Enable','off')
%apagar cuantizacion
set(handles.cuantizacion,'Enable','off')
set(handles.cuantizacion,'Value',0)
f =
[handles.modetext,handles.roundmodetext,handles.overflowtext,handles.formattext
];
g =
[handles.mode,handles.roundmode,handles.overflowmode,handles.format,handles
.aplicarcuantizacion];
set(f,'Enable','off')
set(g,'Enable','off')

set(handles.vectoramp,'Visible','on')
set(handles.text7,'Visible','on')
set(handles.unidadfrec,'Enable','on')
set(handles.frectext,'Enable','on')
set(handles.interpfac,'Visible','off')
set(handles.factorinterpolacion,'Visible','off')
d = [handles.ordenfiltro,handles.ordenminimo,handles.encuentraorden,handles.n];
b = [handles.vectorpeso,handles.vecpeso];
set(b,'Visible','on')
set(d,'Enable','on')



% --- Executes on button press in cuantizacion.
function cuantizacion_Callback(hObject, eventdata, handles)
% hObject    handle to cuantizacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of cuantizacion
handles.bandera_cuantizacion = 0;
cuantizacion = get(hObject,'Value');
f =
[handles.modetext,handles.roundmodetext,handles.overflowtext,handles.formattext
];
g =
[handles.mode,handles.roundmode,handles.overflowmode,handles.format,handles
.aplicarcuantizacion];
```

```
if cuantizacion == 1    %presionado
   set(f,'Enable','on')
   set(g,'Enable','on')
   set(handles.mantenervalores,'Enable','off')  %  no  permite  comparar  valores
durante cuantizacion
elseif cuantizacion == 0    %liberado
   set(f,'Enable','off')
   set(g,'Enable','off')
   set(handles.mantenervalores,'Enable','on')  % permite comparar valores
   set(handles.limitcicle,'Enable','off')       %no permite limite de ciclos
   set(handles.magnitud,'String','Magnitude')
   set(handles.fase,'Enable','on')
   set(handles.retardogrupo,'Enable','on')
   set(handles.polocerotexto,'Enable','on')
   set(handles.tf,'Enable','on')


end

guidata(hObject, handles);




% --- Executes during object creation, after setting all properties.
function mode_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in mode.
function mode_Callback(hObject, eventdata, handles)
% hObject    handle to mode (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 mode contents as cell array
%        contents{get(hObject,'Value')} returns selected item from mode

mode = get(hObject,'Value');
f = [handles.roundmodetext,handles.overflowtext,handles.formattext];
g = [handles.roundmode,handles.overflowmode,handles.format];


switch mode
   case 1
      set(handles.format,'String','[16 15]')
      set(f,'Enable','on')
      set(g,'Enable','on')
   case 2
      set(handles.format,'String','[16 15]')
      set(f,'Enable','on')
      set(g,'Enable','on')
   case 3
      set(handles.overflowtext,'Enable','off')
      set(handles.overflowmode,'Enable','off')
      set(handles.roundmodetext,'Enable','on')
      set(handles.roundmode,'Enable','on')
      set(handles.formattext,'Enable','on')
      set(handles.format,'Enable','on')
      set(handles.format,'String','[32 8]')
   case 4
      set(handles.format,'String','[64 11]')
      set(f,'Enable','off')
      set(g,'Enable','off')
   case 5
      set(handles.format,'String','[32 8]')
      set(f,'Enable','off')
      set(g,'Enable','off')
end



% --- Executes during object creation, after setting all properties.
function roundmode_CreateFcn(hObject, eventdata, handles)
% hObject    handle to roundmode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in roundmode.
function roundmode_Callback(hObject, eventdata, handles)
% hObject    handle to roundmode (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 roundmode contents as cell array
%        contents{get(hObject,'Value')} returns selected item from roundmode

% --- Executes during object creation, after setting all properties.
function overflowmode_CreateFcn(hObject, eventdata, handles)
% hObject    handle to overflowmode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in overflowmode.
function overflowmode_Callback(hObject, eventdata, handles)
% hObject    handle to overflowmode (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 overflowmode contents as cell array
%       contents{get(hObject,'Value')} returns selected item from overflowmode


% --- Executes during object creation, after setting all properties.
function format_CreateFcn(hObject, eventdata, handles)
% hObject    handle to format (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function format_Callback(hObject, eventdata, handles)
% hObject    handle to format (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of format as text
%       str2double(get(hObject,'String')) returns contents of format as a double


% --- Executes on button press in limitcicle.
function limitcicle_Callback(hObject, eventdata, handles)
% hObject    handle to limitcicle (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Hq = handles.Hq;
pichote = limitcycle(Hq);
set(handles.prueba,'String',pichote)

% --- Executes on button press in aplicarcuantizacion.
function aplicarcuantizacion_Callback(hObject, eventdata, handles)
% hObject    handle to aplicarcuantizacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
b = handles.coeficientes;
```

```
mode = get(handles.mode,'Value');
contents = get(handles.roundmode,'String');
roundmode = contents{get(handles.roundmode,'Value')};
contents = get(handles.overflowmode,'String');
overflowmode = contents{get(handles.overflowmode,'Value')};
format = eval(get(handles.format,'String'));

clc
switch mode
   case 1  %fixed
      Hq                                                                    =
qfilt('fir',{b},'mode','fixed','roundmode',roundmode,'overflowmode',overflowmode,'for
mat',format)
   case 2  %ufixed
      Hq                                                                    =
qfilt('fir',{b},'mode','ufixed','roundmode',roundmode,'overflowmode',overflowmode,'f
ormat',format)
   case 3  %float
      Hq = qfilt('fir',{b},'mode','float','roundmode',roundmode,'format',format)
   case 4  %double
      Hq = qfilt('fir',{b},'mode','double')
   case 5  %single
      Hq = qfilt('fir',{b},'mode','single')
end
handles.bandera_cuantizacion = 1;
handles.Hq = Hq;
guidata(hObject, handles);
set(handles.limitcicle,'Enable','on')
set(handles.magnitud,'String','Magnitude & Phase')
set(handles.fase,'Enable','off')
set(handles.retardogrupo,'Enable','off')
set(handles.polocerotexto,'Enable','off')
set(handles.tf,'Enable','off')
set(handles.prueba,'String',strvcat('The quantized Coefficients,','the Reference
Coefficients and the','display information about the filter','and its property values
appear at','the comand window.'))


% --- Executes during object creation, after setting all properties.
function factorinterpolacion_CreateFcn(hObject, eventdata, handles)
% hObject    handle to factorinterpolacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function factorinterpolacion_Callback(hObject, eventdata, handles)
% hObject    handle to factorinterpolacion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of factorinterpolacion as text
%        str2double(get(hObject,'String')) returns contents of factorinterpolacion as a
double


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```