Anexo 2: Código MATLAB

```
function acepta_G(orden,h_slider)
% "Callback del boton "aceptar" la interfaz "configurador_g"
% Actualiza la matriz G y guarda los resultados en "sd_matrizG.mat"
% PARAMETROS DE ENTRADA:
%   orden: Determina la configuracion de la matriz de coeficientes.
%         h_slider:   Vector con los manejadores de los deslizadores de la interfaz.
%               Son necesarios para tomar el valor de coeficiente seleccionado.
%
% Invoca la función actualiza_g
% Ver tambien: actualiza_g, cancela_G.m, reset_G.m, actualiza_G.m, g_edit.m, g_slider.m

G=actualiza_g(orden,h_slider); %actualizamos la matriz G
load sd_matrizG %cargamos los datos de control almacenados en sd_matrizG
save sd_matrizG G g_reset slider_value; %salvamos la matriz y las variable de control
close(gcf); %cerramos el configurador de coeficientes 'g'
```

```
function G=actualiza_g(orden,h_slider)
%DESCRIPCION:
%   Guarda en una matriz G los valores de los coeficientes tomados de la
%   interfaz "configurador_g"
%   Los coeficientes (g,g',g") de cada integrador son almacenados en una
%   matriz G de la forma:
%
```

```
%          Integrador1             Integrador2               ...
%
%          entrada 1 (+)           entrada 1 (+)             ...
%   G(3xN)= entrada 2 (-)           entrada 2 (-)             ...
%          entrada realimentada (-) entrada realimentada (-)   ...
%
%   Ejemplo1: Para un modulador 2-1-1, G tendra la forma
%       g1  g2  g3  g4
%    G=  0   0   g3' g4'
%       g1' g2' g3" g4"
%   Ejemplo2: Para un modulador 2-2, G tendra la forma
%       g1  g2  g3  g4
%    G=  0   0   g3' 0
%       g1' g2' g3" g4'
%
% PARAMETROS DE ENTRADA:
%   orden: Determina la configuracion de la matriz de coeficientes.
%         h_slider:   Vector con los manejadores de los deslizadores de la interfaz.
%              Son necesarios para tomar el valor de coeficiente seleccionado.
%
% PARAMETROS DE SALIDA
%   G: matriz de coeficientes
%
%Ver: acepta_G.m, cancela_G.m, reset_G.m, configurador_g.m, g_edit.m, g_slider.m

%capturamos el orden del primer modulador de la cascada
primer_mod=orden(1);
%calculamos el numero total de integradores de la cascada
num_integradores=sum(orden);
%Obtenemos la matriz G, reservamos memoria rellenandola de unos
G=ones(3,num_integradores);

%capturamos los coeficientes del primer modulador de la cascada
%este es un caso especial: los ingradores tendran siempre solo 2 entradas
for (i=1:1:primer_mod)
   G(1,i)=get(h_slider(2*i-1),'value'); %coeficiente g
   G(2,i)=0; %cero, no existe este coeficiente
   G(3,i)=get(h_slider(2*i),'value'); %coeficiente g', entrada realimentada
end

%A partir de la segunda etapa de la cascada, todos los casos son iguales
k=orden(1)+1; %segunda coordenada de G. Indica el numero de integrador
j=2*orden(1)+1; %indice para desplazarnos por el vector de handles (manejadores)
```

```
for(i=2:1:length(orden))
   G(1,k)=get(h_slider(j),'value');%coeficiente g
   G(2,k)=get(h_slider(j+1),'value');%coeficiente g'
   G(3,k)=get(h_slider(j+2),'value');%coeficiente g", entrada realimentada
   k=k+1;
   j=j+3;
   if(orden(i)==2)%si esl orden del modulador de la etapa 'i' es 2, añadimos...
      G(1,k)=get(h_slider(j),'value');%coeficiente g
      G(2,k)=0;%cero, no existe este coeficiente
      G(3,k)=get(h_slider(j+1),'value');%coeficiente g', entrada realimentada
      k=k+1;
      j=j+2;
   end
end

%guardamos en el vector "slider_value" los valores de todos los deslizadores
for(i=1:1:length(h_slider))
  slider_value(i)=get(h_slider(i),'value');
end

g_reset=0; %variable de control de reset a cero
save sd_matrizG g_reset slider_value; %salvamos la variable de control y los valores de todos los deslizadores
```

```
function [BW,Fmin,Fmax]=calcula_BW(signal,f,Fx,caida)
%Calcula el ancho de banda de caida en dB
%PARAMETROS DE ENTRADA:
%   signal:señal de salida total en dB
%   f: vector de frecuencias de signal
%   Fx: Frecuencia centrl de la señal a la que se quiere calcular el BW
%   caida: caida en dB bajo la frec central de la señal en limites del BW
%
%PARAMETROS DE SALIDA
%   BW: ancho de banda de señal
%   Fmin: frecuencia inferior del BW
%   Fmax: frecuencia superior del BW
%
% Ver tambien: calcula_SNR

%calculamos el indice en f de Fx

if(f(1)>=Fx)
```

```
   iFx=1 ;
else
   %cogemos la frecuencia mas cercana a Fx
   ind=find(f<Fx);
   if((Fx-f(ind(end)))<(f(ind(end)+1))-Fx)
      iFx=length(ind);
   else
      iFx=length(ind)+1;
   end
end


s_max=signal(iFx);%valor maximo de señal


%calculamos el indice en f de Fmax


iFmax=iFx;%partimos del incide de Fx
while((signal(iFmax)>(s_max-caida))&(iFmax<length(signal)))
   iFmax=iFmax+1; %aumentamos mientras tengamos frecuencias superiores y no superemos la caida en dB
end


%calculamos el indice en f para Fmin


iFmin=iFx;%partimos del incide de Fx
while((signal(iFmin)>(s_max-caida))&(iFmin>1)&(f(iFmin)>0))
   iFmin=iFmin-1;%disminuimos mientras tengamos frecuencias inferiores y no superemos la caida en dB
end
if(f(iFmin)<0)
   iFmin=iFmin+1;
end


%asignamos los valores a las salidas
Fmax=f(iFmax);
Fmin=f(iFmin);
BW=Fmax-Fmin;
```

```
function SNR=calcula_SNR(salida,Fx,Fs,B,Fmin,Fmax,f);
%Calcula la SNR (Signal to Noise Ratio) de una señal a la salida en dB de un modulador
%Puede operar vectorialmente si se le dan las salidas agrupadas por filas en
%una matriz
%PARAMETROS DE ENTRADA
%   salida: Señal de salida total en dB: Ruido + Señal. Salidas agrupadas
```

```
%   por filas en una matriz
%   Fx: frecuencia central de señal
%   Fs: frecuencia de muestreo de la señal
%   B: ancho de banda del modulador
%   Fmin: frecuencia minima de señal
%   Fmax: frecuencia maxima de señal
%   f: vector de frecuencias de la salida
%PARAMETROS DE SALIDA
%   SNR: vector con los SNR de las sucesivas etapas.
%
%OBSERVACION:
%   Incluye la funcion 'integra'
%
% Ver tambien calcula_SNR


%calculamos la SNR para cada fila de salida
for(i=1:1:size(salida,1))
   ydB=salida(i,:);%tomamos la salida i-esima
   y=10.^(ydB/20);%pasamos a naturales
   y=y.^2;%elevamos al cuadrado para calcular potencia
   %buscamos el indice de la frecuencia inferior de señal
   if(Fmin>=f(end))
      imin=length(f);
   else
      ind=find(f>Fmin);
      if(ind(1)>1)
         if((f(ind(1))-Fmin)>Fmin-f(ind(1)-1))
            imin=ind(1)-1;
         else
           imin=ind(1);
         end
      else
         imin=1;
      end
   end

   %calculamos el indice de la frecuencia superior de señal
   ind=find(f<Fmax);
   if((Fmax-f(ind(end)))>(f(ind(end)+1)-Fmax))
      imax=ind(end)+1;
   else
      imax=ind(end);
```

```
    end


    % el paso en frecuencia es:
    df=f(imin+1)-f(imin);
    %Potencia de señal
    S=integra(y(imin:imax),df);

    % el primer intervalo de ruido va desde 0 hasta fmin
    % N1: potencia del primer intervalo de ruido
    fminr=0;
    ind=find(f>fminr);
    iminr=ind(1);
    if(imin<=iminr)
       N1=0;
    else
       N1=integra(y(iminr:imin),df);
    end

    % el segundo intervalo de ruido va desde fmax hasta B
    % N2: Potencia del segundo intervalo de ruido
    ind=find(f<B);
    imaxr=ind(length(ind));
    N2=integra(y(imax:imaxr),df);


    %calculamos la SNR
    SNR(1,i)=10*log10(S/(N1+N2));

end

function I=integra(y,dx)
% Integracion con rectángulos y cuadrados.

n=length(y);
I = (sum(y(1:n-1))+sum(abs(y(2:n)-y(1:n-1))*1/2))*dx;

%I=0;
%for i=1:n-1
%   I=I+y(i)*dx+dx/2*abs(y(i)-y(i+1));
%end
```

```
function salida=cancela(entrada,orden,G);
% Esta función simula el bloque de cancelación digital de ruido de cuantización en un convertidor sigma-delta en configuración cascada.
% Invoca las siguientes funciones presentes en el mismo fichero "cancela.m":
%       H_par: Modela un filtro de cancelación de subíndice par.
%       H_impar: Modela un filtro de cancelación de subíndice impar.
%       d_par: Modela un coeficiente 'd' de cancelación de subíndice par.
%       d_impar: Modela un coeficiente 'd' de cancelación de subíndice impar.
%   (Ver la documentacion para comprobar nomenclatura)
%
%PARAMETROS DE ENTRADA
%       entrada
%       Matriz correspondiente a la salida del modulador sigma-delta cascada.
%       Cada fila 'i' se corresponde con la salida de la etapa í-esima de modulación .
%       orden: vector de orden del modulador. Indica el orden de cada etapa de la cascada.
%       G: Matriz de coeficientes 'g'. Ganancias a la entrada de los integradores.
%
%PARAMETROS DE SALIDA
%       Salida
%       Matriz de salida del modulador completo (con cancelación de ruido)
%       Cada fila 'i' se corresponde con la salida de la etapa í-esima de modulación  con cancelación de ruido.
%
% Ver tambien configurador_g, cascada, m1, m2, cuantiza



Y=entrada;
%recorremos cada salida del modulador y anulamos el reudo de cuantizacion
for(i=1:1:length(orden)-1)
   aux1=H_impar(Y(i,:),orden,i); %aux1 es la salida del primer filtro H_impar
   aux2=d_impar(orden,i,G)*Y(i+1,:)+d_par(orden,i,G)*aux1; %aux2 es la entrada al segundo filtro H_par
   Y(i+1,:)=H_par(aux2,orden,i)+aux1;%la salida del filtro H_par es la correspondiente a la etapa i+1 por lo que la guardamos en dicha fila en una matriz que contendra todas las salidas por filas Y
end
salida=Y; %la salida sera una matriz que almacene por filas las correspondientes salidas


function b=H_impar(a,orden,i) %filto de subindice impar
tam=length(a); %el tamaño del vector de entrada
b=zeros(1,tam);%Inicializamos a cero los vectores intermedios que vamos a utilizar
if(orden(i+1)==1)% si el orden del siguiente modulador en la cascada es uno...
   b(2:tam)=a(1:tam-1);% entonces el filtro es 1/z que equivale a un retraso
else if(orden(i+1)==2) % si el orden del siguiente modulador en la cascada es dos...
      b(3:tam)=a(1:tam-2);% entonces el filtro es (1/z)*(1/z) que equivale a un retraso doble
```

```
    else disp('Error: Modulador de orden superior a 2 en serie'); %si no ERROR
    end
end



function b=H_par(a,orden,i) %filto de subindice par
orden_f=sum(orden(1:i)); %el orden del filtro es la suma de los moduladores hasta la etapa i. Recordar que estamos en la i+1.
filtro=poly(ones(1,orden_f)); %calculamos el polinomio caracteristico del filtro pasando un vectro con tantos unos como orden sea el filtro. El filto sera de la forma H=(1-z^(-1))^orden_f
tam=length(a);
suma=zeros(1,tam);%Inicializamos a cero los vectores intermedios que vamos a utilizar
for(j=1:1:length(filtro)) %recorremos el vector con los coeficientes del filtro
    aux=zeros(1,tam); %vector auxiliar inicializado a ceros
    aux(j:tam)=a(1:tam-j+1); %retrasamos tantas posiciones como posicion en el vector de coeficientes del filtro
    aux=filtro(j)*aux; %multiplicamos por el correspondiente coeficiente para ese retraso
    suma=suma+aux; %sumamos lo obtenido a lo calculado anteriormente
end
b=suma; %la salida sera la suma total



function d=d_par(orden,i,G) %coeficiente 'd' de subindice par
if (i==1)%d0
    d=G(2,3)/(G(1,1)*G(1,2)*G(1,3))-1; %d=g3'/(g1*g2*g3) - 1
else%d2,d4,d6...
    d=0;
end



function d=d_impar(orden,i,G) %coeficiente 'd' de subindice par
k=sum(orden(1:i))+1;%k es el subindice max que hemos de considerar
prod=1;
for(j=1:1:k)
    prod=prod*G(1,j); %multiplicamos g1*g2*g3*...*gk
end
d=G(3,k)/prod; % d=gk''/(g1*g2*g3*...*gk)
```

```
function cancela_G
% "Callback del boton "cancelar" la interfaz "configurador_g"
% Cierra el configurador de coeficientes 'g' sin hacer nada mas.
%
% % Ver tambien: actualiza_g, acepta_G.m, reset_G.m, actualiza_G.m, g_edit.m, g_slider.m
```

```
close(gcf);
```

```
function salidasY=cascada(orden,x,G,cuant_levels,sat_level,perdidas)
% Esta función realiza una modulación en cascada sigma-delta, sin incluir la cancelación de ruido.
% Para un vector de entrada 'x', de longitud 'N', genera una matriz 'salidasY' 'MxN' ('M', número de etapas total de la cascada)
% Cada una de las filas 'i' se corresponde con la modulación de la entrada,
% hasta la etapa i-ésima, en un  modulado sigma-delta cascada
%
% En la cascada, solo de permiten moduladores de primer y segundo orden en serie, para garantizar la estabilidad del sistema (ver Apartado 3.3.4).
%
% La función no limita el número de etapas de la cascada.
%
% PARAMETROS DE ENTRADA
%         orden: Vector que determina orden de cada etapa de la cascada del modulador.
%         x: Entrada a modular.
%         G: Matriz de coeficientes 'g'
%         cuant_levels: Vector con el numero de escalones de cuantización para los cuantizadores de cada etapa
%         sat_level: Vector con el nivel de saturación para los cuantizadores de cada etapa.
%    perdidas: coeficiente de perdidas en integradores. Comprendido entre 0 y 1
%
% PARAMETROS DE SALIDA
%         salidasY: Matriz con las salidas de cada etapa de modulación de la cascada ordenadas por filas.
%
% Ver tambien configurador_g, m1, m2, cuantiza, cancela


%si solo se especifican 3 argumentos de entrada
if (nargin==3)
    cuant_levels=2*ones(1,length(orden)); % si no se especifica como entrada, todos los cuantizadores son de dos niveles
    sat_level=ones(1,length(orden)); % si no se especifica como entrada, todos los cuantizadores saturan en 1V
    perdidas=0;%si no se especifica como entrada, no habra perdidas
end


intermedio=x;%la entrada de los moduladores sera intermedio (valor intermedio del modulador anterior). Inicializamos a x para el primer modulador
Y(1,:)=zeros(1,length(x));%la matriz Y guarda por filas cada salida de cada modulador de la cascada. Inicializamos a cero porque al primer modulador no habra que restarle la salida de ninguno anterior
j=1;%inicializamos el indice de la matriz de coeficientes G. Una columna para cada integrador.(columnas de tres filas ya que cada integrador como max tendra 3 coeficientes)
```

```
for i=[1:1:length(orden)];%por cada modulador serie de la cascada...
   if (orden(i)==1)%si modulador de primer orden

[Y(i+1,:),intermedio]=m1(intermedio,Y(i,:),G(j),G(j+1),G(j+2),cuant_levels(i),sat_level(i),perdidas);%
modulamos con orden1 el valor intermedio del mod anterior, la salida anterior (se resta dentro del mod)
e intoducimos coeficientes de G. Devuelve en una fila de Y la salida y el valor intermedio para qu lo
emplee el siguiente mod de la cascada
      j=j+3;%actualizamos el indice de G
   else
      if (orden(i)==2)%si modulador de segundo orden

[Y(i+1,:),intermedio]=m2(intermedio,Y(i,:),G(j),G(j+1),G(j+2),G(j+3),G(j+5),cuant_levels(i),sat_level(
i),perdidas);%modulamos con orden 2 el valor intermedio del mod anterior, la salida enterior (se resta
dentro del mod) e intoducimos coeficientes de G. Devuelve en una fila de Y la salida y el valor
intermedio para qu lo emplee el siguiente mod de la cascada
         j=j+6;%actualizamos el indice de G
      else %si el modulador serie no es de orden 1 o 2 entonces...
         disp('Error: Modulador de orden superior a 2 en serie');
      end
   end
end
salidasY=Y(2:end,:);%extraemos la primera fila de ceros para que cada fila se corresponda con las
salida correspondiente (Y1 con fila1...);
```

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

% Last Modified by GUIDE v2.5 10-Nov-2003 10:35:09

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

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

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes configurador_arquitectura wait for user response (see UIRESUME)
% uiwait(handles.configurador_arquitectura);
```

```
%Creamos una clase en la estructura para almacenar el orden
handles.orden=0;
guidata(hObject,handles);

%Creamos una clase en la estructura para ver si ha cambiado la arquitectura
handles.cambio=0;
guidata(hObject,handles);

%Inicializamos a cero los marcadores de orden
for(i=1:1:10)
   clear_orden=['set(handles.mod',int2str(i),'_1, ''Value'', 0);'];
   eval(clear_orden); %Inicializamos a cero los marcadores de primer orden
   clear_orden=['set(handles.mod',int2str(i),'_2, ''Value'', 0);'];
   eval(clear_orden); %Inicializamos a cero los marcadores de segundo orden
end

%desactivamos todos los marcadores menos el correspondiente al primer modulador
for(i=2:1:10)
   desactiva_mod_1=['set(handles.mod',int2str(i),'_1',',''Enable'',''off'');'];
   eval (desactiva_mod_1); %desactiva marcador de primer orden
   desactiva_mod_2=['set(handles.mod',int2str(i),'_2',',''Enable'',''off'');'];
   eval (desactiva_mod_2); %desactiva marcador de segundo orden
end

%desactivamos todos las casillas de niveles de cuantizacion
for(i=1:1:10)
   desactiva_cuant_nivel=['set(handles.cuant',int2str(i),',''Enable'',''off'');'];
   eval (desactiva_cuant_nivel);
end

%desactivamos todos las casillas de nivel de saturacion del cuantizador
for(i=1:1:10)
   desactiva_cuant_sat=['set(handles.sat',int2str(i),',''Enable'',''off'');'];
   eval (desactiva_cuant_sat);
end

%Cargamos los posibles valores de una arquitectura previamente configurada
load sd_arquitectura;

if(m_reset==0) %si no es la primera vez que se abre la ventana
   %activamos el orden de los moduladores correspondientes y el siguiente al ultimo
   for(i=1:1:length(orden)+1)
      if(i<11)%si no estamos en el caso limite de 10 etapas
```

```
            activa_mod_1=['set(handles.mod',int2str(i),'_1',','Enable'','on'');'];
            eval (activa_mod_1);
            activa_mod_2=['set(handles.mod',int2str(i),'_2',','Enable'','on'');'];
            eval (activa_mod_2);
        end
    end

    for(i=1:1:length(orden))
        %activamos los niveles de los cuantizadores y su saturacion
        activa_cuant_nivel=['set(handles.cuant',int2str(i),','Enable'','on'');'];
        eval (activa_cuant_nivel);
        activa_cuant_sat=['set(handles.sat',int2str(i),','Enable'','on'');'];
        eval (activa_cuant_sat);
        %fijamos el orden de cada etapa en los marcadores
        if(orden(i)==1) %si primer orden
            %fijamos el marcador correspondiente
            set_orden=['set(handles.mod',int2str(i),'_1, ''Value'', 1);'];
            eval(set_orden);
            %guardamos el orden en la estructura
            orden_struct=['handles.orden(',int2str(i),')=1;'];
            eval(orden_struct);
            guidata(hObject,handles);
        elseif(orden(i)==2)%si segundo orden
            %fijamos el marcador correspondiente
            set_orden=['set(handles.mod',int2str(i),'_2, ''Value'', 1);'];
            eval(set_orden);
            %guardamos el orden en la estructura
            orden_struct=['handles.orden(',int2str(i),')=2;'];
            eval(orden_struct);
            guidata(hObject,handles);
        end
        %fijamos el numero de escalones y niveles de saturacion de cada etapa
        set_cuant_nivel=['set(handles.cuant',int2str(i),','string'',',int2str(cuant(i)),');'];
        eval (set_cuant_nivel);
        %fijamos el nivel de saturacion del cuantizador de cada etapa
        set_cuant_sat=['set(handles.sat',int2str(i),','string'',',num2str(sat(i)),');'];
        eval (set_cuant_sat);
    end
    %Mostramos el modulador mediante texto no editable
    set(handles.texto_orden,'String',int2str(handles.orden));

end
```

```
% --- Outputs from this function are returned to the command line.
function varargout = configurador_arquitectura_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 on button press in mod3_1.
function mod3_1_Callback(hObject, eventdata, handles)
% hObject    handle to mod3_1 (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 mod3_1

handles.orden(3)=1;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod3_1, 'Value', 1);
set(handles.mod3_2, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod4_1,'Enable','on');
set(handles.mod4_2,'Enable','on');
set(handles.cuant3,'Enable','on');
set(handles.sat3,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes on button press in mod3_2.
function mod3_2_Callback(hObject, eventdata, handles)
% hObject    handle to mod3_2 (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 mod3_2
```

```
handles.orden(3)=2;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod3_2, 'Value', 1);
set(handles.mod3_1, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod4_1,'Enable','on');
set(handles.mod4_2,'Enable','on');
set(handles.cuant3,'Enable','on');
set(handles.sat3,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function cuant3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cuant3 (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 cuant3_Callback(hObject, eventdata, handles)
% hObject    handle to cuant3 (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 cuant3 as text
%        str2double(get(hObject,'String')) returns contents of cuant3 as a double


% --- Executes during object creation, after setting all properties.
```

```
function sat3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sat3 (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 sat3_Callback(hObject, eventdata, handles)
% hObject    handle to sat3 (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 sat3 as text
%        str2double(get(hObject,'String')) returns contents of sat3 as a double


% --- Executes on button press in mod4_1.
function mod4_1_Callback(hObject, eventdata, handles)
% hObject    handle to mod4_1 (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 mod4_1

handles.orden(4)=1;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod4_1, 'Value', 1);
set(handles.mod4_2, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod5_1,'Enable','on');
set(handles.mod5_2,'Enable','on');
set(handles.cuant4,'Enable','on');
set(handles.sat4,'Enable','on');
```

```
%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes on button press in mod4_2.
function mod4_2_Callback(hObject, eventdata, handles)
% hObject    handle to mod4_2 (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 mod4_2

handles.orden(4)=2;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod4_2, 'Value', 1);
set(handles.mod4_1, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod5_1,'Enable','on');
set(handles.mod5_2,'Enable','on');
set(handles.cuant4,'Enable','on');
set(handles.sat4,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function cuant4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cuant4 (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 cuant4_Callback(hObject, eventdata, handles)
% hObject    handle to cuant4 (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 cuant4 as text
%        str2double(get(hObject,'String')) returns contents of cuant4 as a double


% --- Executes during object creation, after setting all properties.
function sat4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sat4 (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 sat4_Callback(hObject, eventdata, handles)
% hObject    handle to sat4 (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 sat4 as text
%        str2double(get(hObject,'String')) returns contents of sat4 as a double


% --- Executes on button press in mod5_1.
function mod5_1_Callback(hObject, eventdata, handles)
% hObject    handle to mod5_1 (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 mod5_1
```

```
handles.orden(5)=1;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod5_1, 'Value', 1);
set(handles.mod5_2, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod6_1,'Enable','on');
set(handles.mod6_2,'Enable','on');
set(handles.cuant5,'Enable','on');
set(handles.sat5,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes on button press in mod5_2.
function mod5_2_Callback(hObject, eventdata, handles)
% hObject    handle to mod5_2 (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 mod5_2

handles.orden(5)=2;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod5_2, 'Value', 1);
set(handles.mod5_1, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod6_1,'Enable','on');
set(handles.mod6_2,'Enable','on');
set(handles.cuant5,'Enable','on');
set(handles.sat5,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function cuant5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cuant5 (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 cuant5_Callback(hObject, eventdata, handles)
% hObject    handle to cuant5 (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 cuant5 as text
%        str2double(get(hObject,'String')) returns contents of cuant5 as a double


% --- Executes during object creation, after setting all properties.
function sat5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sat5 (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 sat5_Callback(hObject, eventdata, handles)
% hObject    handle to sat5 (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 sat5 as text
%        str2double(get(hObject,'String')) returns contents of sat5 as a double


% --- Executes on button press in mod6_1.
function mod6_1_Callback(hObject, eventdata, handles)
% hObject    handle to mod6_1 (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 mod6_1

handles.orden(6)=1;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod6_1, 'Value', 1);
set(handles.mod6_2, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod7_1,'Enable','on');
set(handles.mod7_2,'Enable','on');
set(handles.cuant6,'Enable','on');
set(handles.sat6,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function cuant6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cuant6 (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 cuant6_Callback(hObject, eventdata, handles)
% hObject    handle to cuant6 (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 cuant6 as text
%        str2double(get(hObject,'String')) returns contents of cuant6 as a double


% --- Executes during object creation, after setting all properties.
function sat6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sat6 (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 sat6_Callback(hObject, eventdata, handles)
% hObject    handle to sat6 (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 sat6 as text
%        str2double(get(hObject,'String')) returns contents of sat6 as a double


% --- Executes on button press in mod7_1.
function mod7_1_Callback(hObject, eventdata, handles)
% hObject    handle to mod7_1 (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 mod7_1
```

```
handles.orden(7)=1;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod7_1, 'Value', 1);
set(handles.mod7_2, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod8_1,'Enable','on');
set(handles.mod8_2,'Enable','on');
set(handles.cuant7,'Enable','on');
set(handles.sat7,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes on button press in mod7_2.
function mod7_2_Callback(hObject, eventdata, handles)
% hObject    handle to mod7_2 (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 mod7_2

handles.orden(7)=2;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod7_2, 'Value', 1);
set(handles.mod7_1, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod8_1,'Enable','on');
set(handles.mod8_2,'Enable','on');
set(handles.cuant7,'Enable','on');
set(handles.sat7,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function cuant7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cuant7 (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 cuant7_Callback(hObject, eventdata, handles)
% hObject    handle to cuant7 (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 cuant7 as text
%        str2double(get(hObject,'String')) returns contents of cuant7 as a double


% --- Executes during object creation, after setting all properties.
function sat7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sat7 (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 sat7_Callback(hObject, eventdata, handles)
% hObject    handle to sat7 (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 sat7 as text
%        str2double(get(hObject,'String')) returns contents of sat7 as a double


% --- Executes on button press in mod8_1.
function mod8_1_Callback(hObject, eventdata, handles)
% hObject    handle to mod8_1 (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 mod8_1

handles.orden(8)=1;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod8_1, 'Value', 1);
set(handles.mod8_2, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod9_1,'Enable','on');
set(handles.mod9_2,'Enable','on');
set(handles.cuant8,'Enable','on');
set(handles.sat8,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function cuant8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cuant8 (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 cuant8_Callback(hObject, eventdata, handles)
% hObject    handle to cuant8 (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 cuant8 as text
%        str2double(get(hObject,'String')) returns contents of cuant8 as a double


% --- Executes during object creation, after setting all properties.
function sat8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sat8 (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 sat8_Callback(hObject, eventdata, handles)
% hObject    handle to sat8 (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 sat8 as text
%        str2double(get(hObject,'String')) returns contents of sat8 as a double


% --- Executes on button press in mod9_1.
function mod9_1_Callback(hObject, eventdata, handles)
% hObject    handle to mod9_1 (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 mod9_1
```

```
handles.orden(9)=1;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod9_1, 'Value', 1);
set(handles.mod9_2, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod10_1,'Enable','on');
set(handles.mod10_2,'Enable','on');
set(handles.cuant9,'Enable','on');
set(handles.sat9,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes on button press in mod9_2.
function mod9_2_Callback(hObject, eventdata, handles)
% hObject    handle to mod9_2 (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 mod9_2

handles.orden(9)=2;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod9_2, 'Value', 1);
set(handles.mod9_1, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod10_1,'Enable','on');
set(handles.mod10_2,'Enable','on');
set(handles.cuant9,'Enable','on');
set(handles.sat9,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function cuant9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cuant9 (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 cuant9_Callback(hObject, eventdata, handles)
% hObject    handle to cuant9 (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 cuant9 as text
%        str2double(get(hObject,'String')) returns contents of cuant9 as a double


% --- Executes during object creation, after setting all properties.
function sat9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sat9 (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 sat9_Callback(hObject, eventdata, handles)
% hObject    handle to sat9 (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 sat9 as text
%        str2double(get(hObject,'String')) returns contents of sat9 as a double


% --- Executes on button press in mod10_1.
function mod10_1_Callback(hObject, eventdata, handles)
% hObject    handle to mod10_1 (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 mod10_1

handles.orden(10)=1;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod10_1, 'Value', 1);
set(handles.mod10_2, 'Value', 0);

set(handles.cuant10,'Enable','on');
set(handles.sat10,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes on button press in mod10_2.
function mod10_2_Callback(hObject, eventdata, handles)
% hObject    handle to mod10_2 (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 mod10_2

handles.orden(10)=2;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod10_2, 'Value', 1);
set(handles.mod10_1, 'Value', 0);

set(handles.cuant10,'Enable','on');
set(handles.sat10,'Enable','on');

%ha habido un cambio en la arquitectura
```

```
handles.cambio=1;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function cuant10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cuant10 (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 cuant10_Callback(hObject, eventdata, handles)
% hObject    handle to cuant10 (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 cuant10 as text
%        str2double(get(hObject,'String')) returns contents of cuant10 as a double


% --- Executes during object creation, after setting all properties.
function sat10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sat10 (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 sat10_Callback(hObject, eventdata, handles)
% hObject    handle to sat10 (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 sat10 as text
%        str2double(get(hObject,'String')) returns contents of sat10 as a double


% --- Executes on button press in mod1_1.
function mod1_1_Callback(hObject, eventdata, handles)
% hObject    handle to mod1_1 (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 mod1_1

handles.orden(1)=1;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod1_1, 'Value', 1);
set(handles.mod1_2, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod2_1,'Enable','on');
set(handles.mod2_2,'Enable','on');
set(handles.cuant1,'Enable','on');
set(handles.sat1,'Enable','on');

m_reset=0;
save sd_arquitectura m_reset;

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes on button press in mod1_2.
function mod1_2_Callback(hObject, eventdata, handles)
% hObject    handle to mod1_2 (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 mod1_2

handles.orden(1)=2;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod1_1, 'Value', 0);
set(handles.mod1_2, 'Value', 1);

%activamos los controles del siguiente modulador
set(handles.mod2_1,'Enable','on');
set(handles.mod2_2,'Enable','on');
set(handles.cuant1,'Enable','on');
set(handles.sat1,'Enable','on');

m_reset=0;
save sd_arquitectura m_reset;

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function cuant1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cuant1 (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 cuant1_Callback(hObject, eventdata, handles)
% hObject    handle to cuant1 (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 cuant1 as text
```

```
%        str2double(get(hObject,'String')) returns contents of cuant1 as a double


% --- Executes during object creation, after setting all properties.
function sat1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sat1 (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 sat1_Callback(hObject, eventdata, handles)
% hObject    handle to sat1 (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 sat1 as text
%        str2double(get(hObject,'String')) returns contents of sat1 as a double


% --- Executes on button press in mod2_1.
function mod2_1_Callback(hObject, eventdata, handles)
% hObject    handle to mod2_1 (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 mod2_1

handles.orden(2)=1;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod2_1, 'Value', 1);
set(handles.mod2_2, 'Value', 0);

%activamos los controles del siguiente modulador
```

```
set(handles.mod3_1,'Enable','on');
set(handles.mod3_2,'Enable','on');
set(handles.cuant2,'Enable','on');
set(handles.sat2,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes on button press in mod2_2.
function mod2_2_Callback(hObject, eventdata, handles)
% hObject    handle to mod2_2 (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 mod2_2

handles.orden(2)=2;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod2_2, 'Value', 1);
set(handles.mod2_1, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod3_1,'Enable','on');
set(handles.mod3_2,'Enable','on');
set(handles.cuant2,'Enable','on');
set(handles.sat2,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function cuant2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to cuant2 (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 cuant2_Callback(hObject, eventdata, handles)
% hObject    handle to cuant2 (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 cuant2 as text
%        str2double(get(hObject,'String')) returns contents of cuant2 as a double


% --- Executes during object creation, after setting all properties.
function sat2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sat2 (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 sat2_Callback(hObject, eventdata, handles)
% hObject    handle to sat2 (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 sat2 as text
%        str2double(get(hObject,'String')) returns contents of sat2 as a double


% --- Executes on button press in m_cancelar_pushbutton.
function m_cancelar_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to m_cancelar_pushbutton (see GCBO)
```

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


load sd_arquitectura;
m_reset=0;
save sd_arquitectura orden cuant sat m_reset;

close(handles.configurador_arquitectura);

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

load sd_arquitectura
%recuperamos el orden
orden=handles.orden;
%recuperamos el numero escalones de cuantizacion y niveles de saturacion de los cuantizadores
for(i=1:1:length(orden))
   guarda_cuant=['cuant(',int2str(i),')=str2double(get(handles.cuant',int2str(i),',''String''));'];
   eval(guarda_cuant);
   guarda_sat=['sat(',int2str(i),')=str2double(get(handles.sat',int2str(i),',''String''));'];
   eval(guarda_sat);
end

%guardamos el orden del modulador, y los niveles y rangos de los cuantizadores
save sd_arquitectura orden cuant sat m_reset;
%cerramos la ventana del configurador de arquitectura
close(handles.configurador_arquitectura);
%volvemos a la interfaz de configuracion del modulador
GUI_modulador

if(handles.cambio==1)
   %reseteamos el configurador_G ya que cambiara al cambiar el modulador
   g_reset=1;
   save sd_matrizG g_reset;
end

%reseteamos el configurador_salidas ya que cambiara al cambiar el modulador
salidas_plot=1;
s_reset=1;
```

```
save sd_salida_aux salidas_plot s_reset;


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

m_reset=1;
save sd_arquitectura m_reset;
close(handles.configurador_arquitectura);
configurador_arquitectura

% --- Executes on button press in mod6_2.
function mod6_2_Callback(hObject, eventdata, handles)
% hObject    handle to mod6_2 (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 mod6_2

handles.orden(6)=2;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod6_2, 'Value', 1);
set(handles.mod6_1, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod7_1,'Enable','on');
set(handles.mod7_2,'Enable','on');
set(handles.cuant6,'Enable','on');
set(handles.sat6,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);

% --- Executes on button press in mod8_2.
function mod8_2_Callback(hObject, eventdata, handles)
% hObject    handle to mod8_2 (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 mod8_2

handles.orden(8)=2;
guidata(hObject,handles);
set(handles.texto_orden,'String',int2str(handles.orden));
set(handles.mod8_2, 'Value', 1);
set(handles.mod8_1, 'Value', 0);

%activamos los controles del siguiente modulador
set(handles.mod9_1,'Enable','on');
set(handles.mod9_2,'Enable','on');
set(handles.cuant8,'Enable','on');
set(handles.sat8,'Enable','on');

%ha habido un cambio en la arquitectura
handles.cambio=1;
guidata(hObject,handles);
```

```
%FUNCION: "configurador_g"
%DESCRIPCION:
%   Genera una interfaz de usuario para tomar los coeficientes 'g'
%   correspondientes a la arquitectura previamente seleccionada mediante
%   "configurador_arquitectura" para el programa "sd_cascada".
%   Los coeficientes (g,g',g") de cada integrador son almacenados en una
%   matriz G de la forma:
%
%          Integrador1            Integrador2              ...
%
%          entrada 1 (+)          entrada 1 (+)            ...
%   G(3xN)= entrada 2 (-)          entrada 2 (-)            ...
%          entrada realimentada (-) entrada realimentada (-)   ...
%
%   Ejemplo1: Para un modulador 2-1-1, G tendra la forma
%       g1  g2  g3  g4
%    G=  0   0   g3' g4'
%       g1' g2' g3" g4"
%   Ejemplo2: Para un modulador 2-2, G tendra la forma
%       g1  g2  g3  g4
%    G=  0   0   g3' 0
%       g1' g2' g3" g4'
%
%PARAMETROS:
```

```
%   No tiene parametros propiamente dichos.
%   Toma la arquitectura del modulador cargado el archivo "sd_arquitectura"
%
%Ver: acepta_G.m, cancela_G.m, reset_G.m, actualiza_G.m, g_edit.m, g_slider.m

clear all;
load sd_arquitectura %cargamos la arquitectura previamente seleccionada

%Ajustamos el tamaño y distribucion de la ventana dependiendo del numero de
%moduladores de la cascada
primer_mod=orden(1);
num_moduladores=sum(orden);
if(num_moduladores<11)
   tam_x=400;
   tam_y=(num_moduladores+1)*50;
else if (num_moduladores<21)
       tam_x=800;
       tam_y=11*50;
   else
       disp ('Demasiados moduladores. Introducir G Matricialmente');
   end
end
%Ajustamos la posicion de la ventana
ind_x=800-tam_x;
ind_y=580-tam_y;

%creamos la ventana y fijamos sus propiedades
fig=figure(1);
set(1,'units','pixels','Menubar','none','Name','Configurador de coeficientes "g"','Resize','off','position',[ind_x ind_y tam_x tam_y]);
set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

%Implementamos los botones de control "aceptar", "cancelar" y un marco
h_frame=uicontrol(fig,'Style','frame','units','pixels','position',[ 255 tam_y-20-50*1-5  102+10 71 ]);
h_reset=uicontrol(fig,'Style','pushbutton','units','pixels','position',[ 260 tam_y-26 (102) (17) ], 'String','Reset','Callback','reset_G(h_slider,h_edit)');
h_aceptar=uicontrol(fig,'Style','pushbutton','units','pixels','position',[ 260 tam_y-48 (102) (17) ], 'String','Aceptar','Callback','acepta_G(orden,h_slider)');
h_cancelar=uicontrol(fig,'Style','pushbutton','units','pixels','position',[ 260 tam_y-20-50*1 (102) (17) ], 'String','Cancelar','Callback','cancela_G');

%Implementamos los controles para el primer modulador que no contendra g"
for(i=1:1:primer_mod)
   nombre_var=['g',int2str(i)];
```

```
    h_text(2*i-1)=uicontrol(fig,'Style','text','units','pixels','position',[ 20 tam_y-50*i 20 20 ], 'String',nombre_var);

    h_edit(2*i-1)=uicontrol(fig,'Style','edit','position',[ 70 tam_y-i*50 50 20 ], 'String','0','BackgroundColor','white');

    h_slider(2*i-1)=uicontrol(fig,'Style','slider','position',[ 20 (tam_y-20-i*50) (102) (15) ], 'Min',0,'Max',4);

    nombre_var=['g',int2str(i),''''];

    h_text(2*i)=uicontrol(fig,'Style','text','position',[ 140  tam_y-50*i (20) (20) ], 'String',nombre_var);

    h_edit(2*i)=uicontrol(fig,'Style','edit','position',[ 190 tam_y-i*50 (50) (20) ], 'String','0','BackgroundColor','white');

    h_slider(2*i)=uicontrol(fig,'Style','slider','position',[ 140 (tam_y-20-i*50) (102) (15) ], 'Min',0,'Max',4);

end


%Implementamos los controles para el resto de moduladores de la cascada. Contendra g,g',g"

desplaza_x=0;

desplaza_y=0;

index=0;

for(j=2:1:length(orden))

    i=sum(orden(1:j));

    if(orden(j)==2)

        i=i-1;

    end

    if (i==11) %si tenemos mas de 10 moduladores distribuiremos los coeficientes en 2 columnas

        desplaza_x=400;

        desplaza_y=10*50;

    end

    for(k=1:1:orden(j))

        nombre_var=['g',int2str(i)];

        h_text(3*i-2-primer_mod+index)=uicontrol(fig,'Style','text','position',[ 20+desplaza_x tam_y-50*i+desplaza_y (25) (20) ], 'String',nombre_var);

        h_edit(3*i-2-primer_mod+index)=uicontrol(fig,'Style','edit','position',[ 70+desplaza_x tam_y-50*i+desplaza_y (50) (20) ], 'String','0','BackgroundColor','white');

        h_slider(3*i-2-primer_mod+index)=uicontrol(fig,'Style','slider','position',[ 20+desplaza_x (tam_y-20-i*50)+desplaza_y (102) (15) ], 'Min',0,'Max',4);

        nombre_var=['g',int2str(i),''''];

        h_text(3*i-1-primer_mod+index)=uicontrol(fig,'Style','text','position',[ 140+desplaza_x tam_y-50*i+desplaza_y (25) (20) ], 'String',nombre_var);

        h_edit(3*i-1-primer_mod+index)=uicontrol(fig,'Style','edit','position',[ 190+desplaza_x tam_y-50*i+desplaza_y (50) (20) ], 'String','0','BackgroundColor','white');

        h_slider(3*i-1-primer_mod+index)=uicontrol(fig,'Style','slider','position',[ 140+desplaza_x (tam_y-20-i*50)+desplaza_y (102) (15) ], 'Min',0,'Max',4);

        if(k==1) %si es de primer orden habra g"

            nombre_var=['g',int2str(i),''''];

            h_text(3*i-primer_mod+index)=uicontrol(fig,'Style','text','position',[ 260+desplaza_x tam_y-50*i+desplaza_y (25) (20) ], 'String',nombre_var);

            h_edit(3*i-primer_mod+index)=uicontrol(fig,'Style','edit','position',[ 310+desplaza_x tam_y-50*i+desplaza_y (50) (20) ], 'String','0','BackgroundColor','white');
```

```
        h_slider(3*i-primer_mod+index)=uicontrol(fig,'Style','slider','position',[ 260+desplaza_x
(tam_y-20-i*50)+desplaza_y (102) (15) ], 'Min',0,'Max',4);
        i=i+1;
      else
      index=index-1; %para ajestar el indice en el caso de que solo hubiera g y g'
      end
   end
end

%Ajustamos los callback de los 'slider' y 'edit'
for(i=1:1:length(h_text))
   set(h_slider(i),'Callback',['g_slider(h_slider,h_edit,',num2str(i),')'])%,',orden)'])
   set(h_edit(i),'Callback',['g_edit(h_slider,h_edit,',num2str(i),')'])%,',orden)'])
end

%cargamos la matriz G peviamente guardada
load sd_matrizG
%si no ha sido reseteada, representamos los valores almacenados
if(g_reset==0)
   for(i=1:1:length(h_slider))
      set(h_slider(i),'value',slider_value(i));%para cada delizador, marcamos su valor almacenado
      g_slider (h_slider,h_edit,i); %sincronizamos el deslizador y la caja de texto para que marque el
mismo valor
   end
end
```

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

```
%      stop.  All inputs are passed to configurador_nolineal_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 configurador_nolineal

% Last Modified by GUIDE v2.5 09-Dec-2003 19:29:33

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

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

% Update handles structure
```

```
guidata(hObject, handles);

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

load sd_nolineal

if (jitter_act)%si esta activado el Jitter
    set (handles.jitter, 'value',1);%Marcador
    set (handles.dt_jitter, 'string',num2str(dt_jitter));%cargamos la desviacion tipica
    set (handles.m_jitter, 'string',num2str(m_jitter));%cargamos la media
else%si no esta activado
    set (handles.dt_jitter, 'enable', 'off')%inhabilitamos controles de desviacion tipica
    set (handles.m_jitter, 'enable', 'off')%inhabilitamos controles de media
end

if (var_g_act)%si estan activadas las variaciones de ganancia
    set (handles.variacion_g, 'value',1);%Marcador
    set (handles.dt_var_g, 'string',num2str(dt_var_g));%cargamos la desviacion tipica
    set (handles.m_var_g, 'string',num2str(m_var_g));%cargamos la media
    set (handles.corregir_canc,'value',corregir_canc);
else
    set (handles.dt_var_g, 'enable', 'off')%inhabilitamos controles de desviacion tipica
    set (handles.m_var_g, 'enable', 'off')%inhabilitamos controles de media
    set (handles.corregir_canc,'enable','off');
end

if (perdidas_act)%si estan activadas las perdidas
    set (handles.perdidas, 'value',1);%Marcador
    set (handles.coef_perdidas, 'string',num2str(coef_perdidas));%cargamos el coeficiente de perdidas
    set (handles.slider_perdidas, 'value',coef_perdidas);%tambien en el deslizador
else%si no esta activado
    set (handles.coef_perdidas, 'enable', 'off')%inhabilitamos control de coeficiente
    set (handles.slider_perdidas, 'enable', 'off')%inhabilitamos deslizador
end


% --- Outputs from this function are returned to the command line.
function varargout = configurador_nolineal_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 dt_jitter_CreateFcn(hObject, eventdata, handles)
% hObject    handle to dt_jitter (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 dt_jitter_Callback(hObject, eventdata, handles)
% hObject    handle to dt_jitter (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 dt_jitter as text
%        str2double(get(hObject,'String')) returns contents of dt_jitter as a double


% --- Executes during object creation, after setting all properties.
function m_jitter_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m_jitter (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 m_jitter_Callback(hObject, eventdata, handles)
% hObject    handle to m_jitter (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 m_jitter as text
%        str2double(get(hObject,'String')) returns contents of m_jitter as a double



% --- Executes during object creation, after setting all properties.
function dt_var_g_CreateFcn(hObject, eventdata, handles)
% hObject    handle to dt_var_g (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 dt_var_g_Callback(hObject, eventdata, handles)
% hObject    handle to dt_var_g (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 dt_var_g as text
%        str2double(get(hObject,'String')) returns contents of dt_var_g as a double


% --- Executes during object creation, after setting all properties.
function m_var_g_CreateFcn(hObject, eventdata, handles)
% hObject    handle to m_var_g (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 m_var_g_Callback(hObject, eventdata, handles)
% hObject    handle to m_var_g (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 m_var_g as text
%        str2double(get(hObject,'String')) returns contents of m_var_g as a double


% --- Executes during object creation, after setting all properties.
function coef_perdidas_CreateFcn(hObject, eventdata, handles)
% hObject    handle to coef_perdidas (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 coef_perdidas_Callback(hObject, eventdata, handles)
% hObject    handle to coef_perdidas (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 coef_perdidas as text
%        str2double(get(hObject,'String')) returns contents of coef_perdidas as a double
```

```
NewStrVal = get(handles.coef_perdidas, 'String'); %capturamos el nuevo valor de la caja de texto editable
NewVal = str2double(NewStrVal); %convertimos la cadena de caracteres en un double
% Comprobamos que el valor introducido cae dentro del rango permitido
if  isempty(NewVal) | (NewVal< 0) | (NewVal>1),
   % Regresar al valor anterior
   OldVal = get(handles.slider_perdidas,'Value'); %capturamos el valor anterior del deslizador
   set(handles.coef_perdidas, 'String',OldVal) %fijamos en la caja de texto editable el valor anterior
else, % Usar el nuevo valor
   % Fija en el deslizador el nuevo valor
   set(handles.slider_perdidas,'Value',NewVal)
end


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

load sd_nolineal

if(get(handles.jitter,'value')==1)
   jitter_act=1;
   dt_jitter=str2double(get(handles.dt_jitter, 'string'));
   m_jitter=str2double(get(handles.m_jitter, 'string'));
   save sd_nolineal jitter_act var_g_act perdidas_act dt_jitter m_jitter dt_var_g m_var_g coef_perdidas;
   GUI_entrada;
else
   jitter_act=0;
   dt_jitter=0;
   m_jitter=0;
   save sd_nolineal jitter_act var_g_act perdidas_act dt_jitter m_jitter dt_var_g m_var_g coef_perdidas;
   GUI_entrada;
end

if(get(handles.variacion_g,'value')==1)
   var_g_act=1;
   dt_var_g=str2double(get(handles.dt_var_g, 'string'));
   m_var_g=str2double(get(handles.m_var_g, 'string'));
   corregir_canc=get(handles.corregir_canc,'value');
else
```

```
    var_g_act=0;
    dt_var_g=0;
    m_var_g=0;
    corregir_canc=0;
end

if(get(handles.perdidas,'value')==1)
    perdidas_act=1;
    coef_perdidas=str2double(get(handles.coef_perdidas, 'string'));
else
    perdidas_act=0;
    coef_perdidas=0;
end



save sd_nolineal jitter_act var_g_act perdidas_act dt_jitter m_jitter dt_var_g m_var_g corregir_canc
coef_perdidas;
close (configurador_nolineal)

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

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

% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
% --- Executes on slider movement.
function slider_perdidas_Callback(hObject, eventdata, handles)
% hObject    handle to slider_perdidas (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,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

NewVal = get(handles.slider_perdidas, 'Value'); %tomamos el nuevo valor del deslizador
set(handles.coef_perdidas,'String',NewVal); %fijamos el nuevo valor en la caja de texto editable



% --- Executes on button press in jitter.
function jitter_Callback(hObject, eventdata, handles)
% hObject    handle to jitter (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 jitter

if (get (handles.jitter, 'value')==1)
   set (handles.jitter, 'value',1);
   set (handles.dt_jitter, 'enable', 'on')
   set (handles.m_jitter, 'enable', 'on')
else
   set (handles.jitter, 'value',0);
   set (handles.dt_jitter, 'enable', 'off')
   set (handles.m_jitter, 'enable', 'off')
end

% --- Executes on button press in variacion_g.
function variacion_g_Callback(hObject, eventdata, handles)
% hObject    handle to variacion_g (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 variacion_g

if (get (handles.variacion_g, 'value')==1)
   set (handles.variacion_g, 'value',1);
   set (handles.dt_var_g, 'enable', 'on')
   set (handles.m_var_g, 'enable', 'on')
```

```
    set (handles.corregir_canc,'enable','on')
else
    set (handles.variacion_g, 'value',0);
    set (handles.dt_var_g, 'enable', 'off')
    set (handles.m_var_g, 'enable', 'off')
    set (handles.corregir_canc,'enable','off')
end

% --- Executes on button press in perdidas.
function perdidas_Callback(hObject, eventdata, handles)
% hObject    handle to perdidas (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 perdidas

if (get (handles.perdidas, 'value')==1)
    set (handles.perdidas, 'value',1);
    set (handles.coef_perdidas, 'enable', 'on')
    set (handles.slider_perdidas, 'enable', 'on')
else
    set (handles.perdidas, 'value',0);
    set (handles.coef_perdidas, 'enable', 'off')
    set (handles.slider_perdidas, 'enable', 'off')
end



% --- Executes on button press in corregir_canc.
function corregir_canc_Callback(hObject, eventdata, handles)
% hObject    handle to corregir_canc (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 corregir_canc
```

```
function varargout = configurador_salidas(varargin)
% CONFIGURADOR_SALIDAS M-file for configurador_salidas.fig
%      CONFIGURADOR_SALIDAS, by itself, creates a new CONFIGURADOR_SALIDAS or raises the existing
%      singleton*.
%
%      H = CONFIGURADOR_SALIDAS returns the handle to a new CONFIGURADOR_SALIDAS or the handle to
```

```
%      the existing singleton*.
%
%      CONFIGURADOR_SALIDAS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CONFIGURADOR_SALIDAS.M with the given input arguments.
%
%          CONFIGURADOR_SALIDAS('Property','Value',...) creates a new CONFIGURADOR_SALIDAS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before configurador_salidas_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to configurador_salidas_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 configurador_salidas

% Last Modified by GUIDE v2.5 15-Sep-2003 10:59:39

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

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

% Update handles structure
guidata(hObject, handles);

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

%cargamos datos
load sd_arquitectura;
load sd_salida_aux; %este archivo nos servira para comunicarnos con la interfaz GUI_salida

tam=length(orden); %numero de etapas del modulador

%Mostramos las posibles salidas correspondientes a cada etapa de la cascada
for(i=1:1:10)
   if(i<tam)
      escribe_orden=['set(handles.s_m',int2str(i),'_t,''string'',''',int2str(orden(1:i)),''');'];
      eval(escribe_orden);
      fija_cero=['set (handles.s_m',int2str(i),','value'',0);'];
      eval(fija_cero);
   elseif(i==tam)%la ultima salida, modulador con todas las etapas de la cascada
      escribe_orden=['set(handles.s_m',int2str(i),'_t,''string'',''',int2str(orden(tam+1-i:tam)),''');'];
      eval(escribe_orden);
      if(s_reset==1)%la ultima salida la activamos por defecto ya que es el modulador total
         fija_uno=['set (handles.s_m',int2str(i),','value'',1);'];
         eval(fija_uno);
      end
   else %desactivamos los controles que no necesitamos
       desactiva=['set(handles.s_m',int2str(i),','enable'',''off'');'];
       eval(desactiva);
   end
end

if(s_reset==0)% si no esta activado el reset
   for(i=1:1:length(salidas_plot)) %marcamos las salidas previamente seleccionadas
```

```
        act_sal=['set (handles.s_m',int2str(salidas_plot(i)),',''value'',1);'];
        eval(act_sal);
    end
end

s_reset=0; %desactivamos el reset
save sd_salida_aux salidas_plot s_reset;% guardamos datos

% --- Outputs from this function are returned to the command line.
function varargout = configurador_salidas_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 on button press in s_m1.
function s_m1_Callback(hObject, eventdata, handles)
% hObject    handle to s_m1 (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 s_m1


% --- Executes on button press in s_m2.
function s_m2_Callback(hObject, eventdata, handles)
% hObject    handle to s_m2 (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 s_m2


% --- Executes on button press in s_m3.
function s_m3_Callback(hObject, eventdata, handles)
% hObject    handle to s_m3 (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 s_m3


% --- Executes on button press in s_m4.
function s_m4_Callback(hObject, eventdata, handles)
% hObject    handle to s_m4 (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 s_m4


% --- Executes on button press in s_m7.
function s_m7_Callback(hObject, eventdata, handles)
% hObject    handle to s_m7 (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 s_m7


% --- Executes on button press in s_m9.
function s_m9_Callback(hObject, eventdata, handles)
% hObject    handle to s_m9 (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 s_m9


% --- Executes on button press in s_m5.
function s_m5_Callback(hObject, eventdata, handles)
% hObject    handle to s_m5 (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 s_m5


% --- Executes on button press in s_m10.
function s_m10_Callback(hObject, eventdata, handles)
% hObject    handle to s_m10 (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 s_m10



% --- Executes on button press in s_m8.
function s_m8_Callback(hObject, eventdata, handles)
% hObject    handle to s_m8 (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 s_m8



% --- Executes on button press in s_m6.
function s_m6_Callback(hObject, eventdata, handles)
% hObject    handle to s_m6 (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 s_m6


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

%cargamos datos
load sd_arquitectura
load sd_salida_aux
tam=length(orden);

%capturamos si la salida de cada etapa ha sido seleccionada para
%representarse
i=1;%inicializamos el indice del vector salidas_plot
if(get (handles.s_m1,'value'))%si la primera etapa esta seleccionada
   salidas_plot(i)=1; %añadimos al vector salidas_plot un 1
   i=i+1;%incrementamos el indice del vector
```

```
end
if(get (handles.s_m2,'value'))%si la segunda etapa esta seleccionada
   salidas_plot(i)=2; %añadimos al vector salidas_plot un 2
   i=i+1;%incrementamos el indice del vector
end
if(get (handles.s_m3,'value'))%si la tercera etapa esta seleccionada
   salidas_plot(i)=3; %añadimos al vector salidas_plot un 3
   i=i+1;%incrementamos el indice del vector
end
if(get (handles.s_m4,'value'))%si la cuarta etapa esta seleccionada
   salidas_plot(i)=4; %añadimos al vector salidas_plot un 4
   i=i+1;%incrementamos el indice del vector
end
if(get (handles.s_m5,'value'))%si la quinta etapa esta seleccionada
   salidas_plot(i)=5; %añadimos al vector salidas_plot un 5
   i=i+1;%incrementamos el indice del vector
end
if(get (handles.s_m6,'value'))%si la sexta etapa esta seleccionada
   salidas_plot(i)=6; %añadimos al vector salidas_plot un 6
   i=i+1;%incrementamos el indice del vector
end
if(get (handles.s_m7,'value'))%si la septima etapa esta seleccionada
   salidas_plot(i)=7; %añadimos al vector salidas_plot un 7
   i=i+1;%incrementamos el indice del vector
end
if(get (handles.s_m8,'value'))%si la octava etapa esta seleccionada
   salidas_plot(i)=8; %añadimos al vector salidas_plot un 8
   i=i+1;%incrementamos el indice del vector
end
if(get (handles.s_m9,'value'))%si la novena etapa esta seleccionada
   salidas_plot(i)=9; %añadimos al vector salidas_plot un 9
   i=i+1;%incrementamos el indice del vector
end
if(get (handles.s_m10,'value'))%si la decima etapa esta seleccionada
   salidas_plot(i)=10; %añadimos al vector salidas_plot un 10
   i=i+1;%incrementamos el indice del vector
end

%salvamos los datos
salidas_plot=salidas_plot(1:(i-1));
save sd_salida_aux salidas_plot s_reset;

close(configurador_salidas) %cerramos el configurador de salidas
```

```
GUI_salida;%volvemos a la interfaz de salida

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

close(configurador_salidas)%cierra la ventana activa sin hacer nada
```

```
function cuantizado=cuantiza(entrada,n_levels,rg_cuant)
% Cuantiza un vector de entrada con un numero determinado de escalones de cuantización y un rango de cuantización fijado.
% OBSERVACION:
%   Esta funcion opera vectorialmente a excepcion de la comprobacion de
%   sobrecarga del cuantizador.
%   Para obtener un comportamiento vectorial comentar las lineas 23 a 35
%   (comprobacion de saturacion)
%
% PARAMETROS DE ENTRADA:
%   entrada: vector a cuantizar.
%   n_levels: niveles digitales posibles a la salida.
%   rg_cuant: rango de cuantizacion, Vmax-Vmin a la entrada.
%
% PARAMETROS DE SALIDA
%         cuantizado vector de valores cuantizados
%
% Ver tambien configurador_g, cascada, m1, m2, cancela

delta=rg_cuant/n_levels; %escalon de cuantizacion
dig=[(-rg_cuant+delta)/2:delta:(rg_cuant-delta)/2];%vector de salidas digitales posibles

%comprobamos si la entrada sobrecarga el cuantizador
for(i=1:1:length(entrada))
   if(entrada(i)>=(rg_cuant/2))%comprobamos si hay sobrecarga superior en el cuantizador
      disp ('Warning: Sobrecarga superior en el cuantizador');
      disp ('x>(rg_cuant/2)')
      entrada(i)=dig(n_levels);%no obstante damos como salida el valor digital mas alto
   else%si no, seguimos...
   if(entrada(i)<=(-rg_cuant/2))%comprobamos si hay sobrecarga inferior en el cuantizador
      disp ('Warning: Sobrecarga inferior en el cuantizador');
      disp ('x<(-rg_cuant/2)')
```

```
        entrada(i)=dig(1);%no obstante damos como salida el valor digital mas bajo
    end
    end
end


%ahora cuantizamos propiamente:
x=entrada+rg_cuant/2;%desplazamos el eje x para que los valores esten comprendidos entre 0 y rg_din
index=floor(x/delta)+1;%tomamos la parte entera de la division por el escalon de cuantizacion y le sumamos uno. Esto nos da el numero de la salida digital correspondiente (ordenadas de menor a mayor en dig)
cuantizado=dig(index);%tomamos los valores cuantizados marcados por index
```

```
function g_edit (h_slider,h_edit,i)
% "Callback" de las cajas de texto editables de la interfaz "configurador_g".
% Actualiza el valor del deslizador del parámetro correspondiente, al de la caja de texto editable que acaba de ser modificada.
% Comprueba que el valor introducido cae dentro del rango permitido. Si no
% es asi, fija el valor anterior.
% PARAMETROS DE ENTRADA
%       h_slider: vector con los manejadores de los deslizadores.
%       h_edit: vector con los manejadores de las cajas editables de texto.
%       i: indice que nos indica a que parámetro concreto nos estamos refiriendo.
%
% Ver tambien: actualiza_g, acepta_G.m, cancela_G.m, actualiza_G.m, g_slider.m, reset_g.m

NewStrVal = get(h_edit(i), 'String'); %capturamos el nuevo valor de la caja de texto editable
NewVal = str2double(NewStrVal); %convertimos la cadena de caracteres en un double
% Comprobamos que el valor introducido cae dentro del rango permitido
if  isempty(NewVal) | (NewVal< 0) | (NewVal>4),
    % Regresar al valor anterior
    OldVal = get(h_slider(i),'Value'); %capturamos el valor anterior del deslizador
    set(h_edit(i), 'String',OldVal) %fijamos en la caja de texto editable el valor anterior
else, % Usar el nuevo valor
    % Fija en el deslizador el nuevo valor
    set(h_slider(i),'Value',NewVal)
end
```

```
function g_slider (h_slider,h_edit,i);
% "Callback" de los deslizadores de la interfaz "configurador_g".
% Actualiza el valor de la caja de texto editable del parámetro correspondiente, al del deslizador que acaba de ser modificado.
% PARAMETROS DE ENTRADA
```

```
%        h_slider: vector con los manejadores de los deslizadores.
%        h_edit: vector con los manejadores de las cajas editables de texto.
%        i: indice que nos indica a que parámetro concreto nos estamos refiriendo.
%
% Ver tambien: actualiza_g, acepta_G.m, cancela_G.m, actualiza_G.m, g_edit.m, reset_g.m

NewVal = get(h_slider(i), 'Value'); %tomamos el nuevo valor del deslizador
set(h_edit(i),'String',NewVal); %fijamos el nuevo valor en la caja de texto editable
```

```
function varargout = GUI_entrada(varargin)
% GUI_ENTRADA M-file for GUI_entrada.fig
%      GUI_ENTRADA, by itself, creates a new GUI_ENTRADA or raises the existing
%      singleton*.
%
%      H = GUI_ENTRADA returns the handle to a new GUI_ENTRADA or the handle to
%      the existing singleton*.
%
%      GUI_ENTRADA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI_ENTRADA.M with the given input arguments.
%
%      GUI_ENTRADA('Property','Value',...) creates a new GUI_ENTRADA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_entrada_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUI_entrada_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_entrada

% Last Modified by GUIDE v2.5 13-Nov-2003 20:49:52

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

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

% Update handles structure
guidata(hObject, handles);

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

%Inicializamos los radio buttons
set(handles.fft, 'Value', 0);
set(handles.psd, 'Value', 1);%activamos la PSD por defecto

handles.flag=0; %indica que no hay barra de herramientas activa
guidata(hObject,handles);

load sd_entrada;%cargamos los valores de las variables de entrada
%fijamos en la ventana los valores correspondientes
set(handles.frecuencia,'String',Fx);
set(handles.sobremuestreo,'String',M);
set(handles.ciclos,'String',n_ciclos);
set(handles.ancho_de_banda,'String',B);
```

```
set(handles.amplitud,'String',A);
set(handles.e_nfft,'String',256);
%generamos la entrada propiamente dicha
entrada_Callback(hObject, eventdata, handles);

% --- Outputs from this function are returned to the command line.
function varargout = GUI_entrada_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 frecuencia_CreateFcn(hObject, eventdata, handles)
% hObject    handle to frecuencia (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 frecuencia_Callback(hObject, eventdata, handles)
% hObject    handle to frecuencia (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 frecuencia as text
%        str2double(get(hObject,'String')) returns contents of frecuencia as a double


% --- Executes during object creation, after setting all properties.
function ancho_de_banda_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ancho_de_banda (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 ancho_de_banda_Callback(hObject, eventdata, handles)
% hObject    handle to ancho_de_banda (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 ancho_de_banda as text
%        str2double(get(hObject,'String')) returns contents of ancho_de_banda as a double



% --- Executes during object creation, after setting all properties.



function sobremuestreo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sobremuestreo (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 sobremuestreo_Callback(hObject, eventdata, handles)
% hObject    handle to sobremuestreo (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 sobremuestreo as text
%        str2double(get(hObject,'String')) returns contents of sobremuestreo as a double


% --- Executes during object creation, after setting all properties.
function ciclos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ciclos (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 ciclos_Callback(hObject, eventdata, handles)
% hObject    handle to ciclos (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 ciclos as text
%        str2double(get(hObject,'String')) returns contents of ciclos as a double


% --- Executes during object creation, after setting all properties.
function amplitud_CreateFcn(hObject, eventdata, handles)
% hObject    handle to amplitud (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 amplitud_Callback(hObject, eventdata, handles)
% hObject    handle to amplitud (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 amplitud as text
%        str2double(get(hObject,'String')) returns contents of amplitud as a double


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


% Tomamos datos de entrada del usuario a traves de la interfaz GUI
Fx = str2double(get(handles.frecuencia,'String')); %frecuencia de la señal sinusoidal de entrada
M = str2double(get(handles.sobremuestreo,'String')); %tasa de sobremuestreo
n_ciclos = str2double(get(handles.ciclos,'String')); %numero de ciclos de la señal sinusoidal de entrada
B = str2double(get(handles.ancho_de_banda,'String')); %ancho de banda
A = str2double(get(handles.amplitud,'String')); %amplitud de la señal sinusoidal de entrada

load sd_nolineal%cargamosposibles no linealidades
%Muestreamos la señal
[x Fs kTs]=muestrea(Fx,M,n_ciclos,B,A,m_jitter,(dt_jitter)^2);

%Mostramos la frecuencia de muestreo
set(handles.Fs, 'String', Fs);

%Calculamos la fft
[Xw_fft,eje_f_fft]=sd_fft(x,Fs);

%Calculamos la psd
NFFT= str2double(get(handles.e_nfft,'string'));%tomamos nfft de la GUI
%Comprobamos primero si las partes en que vamos a dividir la señal son mayores que la propia señal
if (NFFT>length(x))%si se señala una NFFT mayor que la longitud de las secuencias
   disp('Warning: max(NFFT)=length(x)')
   NFFT=length(x)%en ese caso fijamos la particion al tamaño de la señal
   set(handles.e_nfft,'string',NFFT)%fijamos el valor maximo permitido
end
[Xw_psd,eje_f_psd]=sd_psd(x,NFFT,Fs,eval(['boxcar(',int2str(NFFT),')']));
```

```
%Representamos en frecuencia
if (get(handles.fft, 'Value'))% si esta marcada la fft en la GUI
    %dibujamos en frecuencia la fft
    axes(handles.ejes_frecuencia) % seleccionamos los ejes adecuados
    %al ser real la señal en el tiempo, en frecuencia su espectro sera par
    %dibujamos solo el espectro correspondiente a "frecuencias positivas"
    plot(eje_f_fft,20*log10(abs(Xw_fft)));
    %fijamos los ejes para mostrar solo las frecuencias positivas
    axis([0,eje_f_fft(end),min(20*log10(abs(Xw_fft))), max(20*log10(abs(Xw_fft)))]);
    xlabel('Frecuencia');
    ylabel('20*log()');
    set(handles.ejes_frecuencia,'XMinorTick','on')
    grid on
else % Si esta marcado en la GUI la PSD
    %dibujamos en frecuencia la psd
    axes(handles.ejes_frecuencia) % seleccionamos en frecuencia los ejes adecuados
    plot(eje_f_psd,10*log10(abs(Xw_psd)));
    xlabel('Frecuencia');
    ylabel('Magnitud [dB]');
    set(handles.ejes_frecuencia,'XMinorTick','on')
    grid on
end

% Dibujamos la señal en el tiempo
axes(handles.ejes_tiempo) % seleccionamos los ejes apropiados
%stem(kTs,x);
plot(kTs,x,'X');
xlabel('Tiempo [seg.]');
ylabel('Amplitud');
set(handles.ejes_tiempo,'XMinorTick','on')
grid on

%guardamos los datos obtenidos en un archivo ".mat"
save sd_entrada  x A n_ciclos M Fx B Fs kTs Xw_fft eje_f_fft Xw_psd eje_f_psd  NFFT;


% --- Executes on button press in fft.
function fft_Callback(hObject, eventdata, handles)
% hObject    handle to fft (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 fft
```

```
%Fijamos el selector de la fft
set(handles.fft, 'Value', 1);
set(handles.psd, 'Value', 0);

% Create frequency "fft" plot
load sd_entrada; %Cargamos los datos de la entrada
%dibujamos en frecuencia la fft
axes(handles.ejes_frecuencia) % seleccionamos los ejes adecuados
%al ser real la señal en el tiempo, en frecuencia su espectro sera par
%dibujamos solo el espectro correspondiente a "frecuencias positivas"
plot(eje_f_fft,20*log10(abs(Xw_fft)));
%fijamos los ejes para mostrar solo las frecuencias positivas
axis([0,eje_f_fft(end),min(20*log10(abs(Xw_fft))), max(20*log10(abs(Xw_fft)))]);
xlabel('Frecuencia');
ylabel('Magnitud [dB]');
set(handles.ejes_frecuencia,'XMinorTick','on')
grid on


% --- Executes on button press in psd.
function psd_Callback(hObject, eventdata, handles)
% hObject    handle to psd (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 psd

%Fijamos el selector de la psd
set(handles.psd, 'Value', 1);
set(handles.fft, 'Value', 0);

% Create frequency "psd" plot

load sd_entrada; %cargamos los datos de la entrada
%por si se ha modificado la NFFT recalculamos la psd
NFFT= str2double(get(handles.e_nfft,'string'));%tomamos nfft de la GUI
%Comprobamos primero si las partes en que vamos a dividir la señal son mayores que la propia señal
if (NFFT>length(x))%si se señala una NFFT mayor que la longitud de las secuencias
   disp('Warning: max(NFFT)=length(x)')
   NFFT=length(x)%en ese caso fijamos la particion al tamaño de la señal
   set(handles.e_nfft,'string',NFFT)%fijamos el valor maximo permitido
end
```

```
[Xw_psd,eje_f_psd]=sd_psd(x,NFFT,Fs,eval(['boxcar(',int2str(NFFT),')']));

axes(handles.ejes_frecuencia) % Select the proper axes
plot(eje_f_psd,10*log10(abs(Xw_psd)));
xlabel('Frecuencia');
ylabel('Magnitud [dB]');
set(handles.ejes_frecuencia,'XMinorTick','on')
grid on


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


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

%abrimos la ventana de impresion
printdlg(gcbf);

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

%abrimos la ventana de previsualizar impresion
printpreview(gcbf);

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

file = uigetfile('*.mat','Cargar entrada');%recogemos es archivo a cargar
if ~isequal(file, 0)%si se ha seleccionado un archivo
   datos=open (file);%cargamos los valores de las variables
```

```
    %fijamos en la ventana los valores correspondientes
    set(handles.frecuencia,'String',datos.Fx);
    set(handles.sobremuestreo,'String',datos.M);
    set(handles.ciclos,'String',datos.n_ciclos);
    set(handles.ancho_de_banda,'String',datos.B);
    set(handles.amplitud,'String',datos.A);
    set(handles.e_nfft,'String',256);
    %generamos la entrada propiamente dicha
    entrada_Callback(hObject, eventdata, handles);
end

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

file=uiputfile('*.mat','Guardar entrada');%tomamos el nombre del fichero
% Get user input from GUI
Fx = str2double(get(handles.frecuencia,'String'));
M = str2double(get(handles.sobremuestreo,'String'));
n_ciclos = str2double(get(handles.ciclos,'String'));
B = str2double(get(handles.ancho_de_banda,'String'));
A = str2double(get(handles.amplitud,'String'));
%Muestreamos la señal
[x Fs kTs]=muestrea(Fx,M,n_ciclos,B,A);
%Grabamos las variables de interes
save (file, 'x', 'Fs', 'kTs', 'Fx', 'M', 'n_ciclos', 'B', 'A');

% --------------------------------------------------------------------
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)

%Cerramos todo abriendo una question box
selection = questdlg(['¿Quiere cerrar ' get(handles.GUI_entrada,'Name') '?'],...
                     ['Close ' get(handles.GUI_entrada,'Name') '...'],...
                     'Si','No','Si');
if strcmp(selection,'No')
    return;
end
```

```
delete(handles.GUI_entrada)


% --------------------------------------------------------------------
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)



% --------------------------------------------------------------------
function documentacion_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)

!ayuda.htm

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


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

GUI_modulador;%activamos el modulador

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

GUI_salida;%activamos la salida
```

```
% --- Executes during object creation, after setting all properties.
function e_nfft_CreateFcn(hObject, eventdata, handles)
% hObject    handle to e_nfft (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 e_nfft_Callback(hObject, eventdata, handles)
% hObject    handle to e_nfft (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 e_nfft as text
%        str2double(get(hObject,'String')) returns contents of e_nfft as a double


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


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

if(handles.flag==0)%si no hay barra activada
    fig=gcf;
    htoolbar.htoolbar = uitoolbar('Parent',fig);%hObject);
    % Render Print buttons (Print, Print Preview)
    htoolbar.hprintbtns = render_sptprintbtns(htoolbar.htoolbar);
```

```
    % Render the annotation buttons (Edit Plot, Insert Arrow, etc)
    htoolbar.hscribebtns = render_sptscribebtns(htoolbar.htoolbar);
    % Render the zoom buttons
    htoolbar.hzoombtns = render_zoombtns(htoolbar.htoolbar);
    %Creamos una clase en la estructura para almacenar el handles de herramientas
    handles.herramientas=htoolbar.htoolbar;
    handles.flag=1;%indica que hemos activado la barra de herramientas
    guidata(hObject,handles);
end

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

set (handles.herramientas,'visible','off');%desactivamos
handles.flag=0;%indica que no hay barra de herramientas activa
guidata(hObject,handles);



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

zoom on %activamos el zoom

% --------------------------------------------------------------------
function zoomout_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

zoom out %anulamos el zoom
zoom off %desactivamos el zoom



% --------------------------------------------------------------------
function editar_Callback(hObject, eventdata, handles)
% hObject    handle to editar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
sigsetappdata(gcbf,'siggui','ZoomState', 'none');setzoomstate(gcbf);putdowntext('select',gcbo)

% --------------------------------------------------------------------
function insertar_texto_Callback(hObject, eventdata, handles)
% hObject    handle to insertar_texto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%sigsetappdata(gcbf,'siggui','ZoomState', 'none');setzoomstate(gcbf);
plotedit
putdowntext(gcbo);%,'textstart'

% --------------------------------------------------------------------
function insertar_flecha_Callback(hObject, eventdata, handles)
% hObject    handle to insertar_flecha (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
sigsetappdata(gcbf,'siggui','ZoomState', 'none');setzoomstate(gcbf);putdowntext('arrowstart',gcbo);

% --------------------------------------------------------------------
function insertar_linea_Callback(hObject, eventdata, handles)
% hObject    handle to insertar_linea (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
sigsetappdata(gcbf,'siggui','ZoomState', 'none');setzoomstate(gcbf);putdowntext('linestart',gcbo);
```

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

```
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_modulador

% Last Modified by GUIDE v2.5 03-Dec-2003 23:49:46

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

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

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

```
% UIWAIT makes GUI_modulador wait for user response (see UIRESUME)
% uiwait(handles.GUI_modulador);

%Cargamos y mostramos el orden configurado
load sd_arquitectura;
if(m_reset==1)%si no se habia configurado ninguna arquitectura previamente
    %mostramos cedenas vacias
    set (handles.muestra_orden,'String','');
    set (handles.muestra_cuant,'String','');
    set (handles.muestra_sat,'String','');
    set (handles.G_pushbutton,'enable','off');
    set (handles.no_lineal_pushbutton,'enable','off');
else%si se habia configurado ninguna arquitectura previamente; la cargamos
    set (handles.muestra_orden,'String',int2str(orden));
    %Cargamos y mostramos el cuantizador configurado
    set (handles.muestra_cuant,'String',int2str(cuant));
    %Cargamos y mostramos el cuantizador configurado
    set (handles.muestra_sat,'String',num2str(sat));
    set (handles.G_pushbutton,'enable','on');
    set (handles.no_lineal_pushbutton,'enable','on');
end

% --- Outputs from this function are returned to the command line.
function varargout = GUI_modulador_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 orden_CreateFcn(hObject, eventdata, handles)
% hObject    handle to orden (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

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

%abrimos el configurador de coeficientes 'g'
configurador_g

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

%Abrimos el configurador de no-linealidades
configurador_nolineal

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

%abrimos el configurador de orden
configurador_arquitectura


% --------------------------------------------------------------------
function m_cargar_Callback(hObject, eventdata, handles)
% hObject    handle to m_cargar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file = uigetfile('*.mat','Cargar modulador');%recogemos es archivo a cargar
if ~isequal(file, 0)%si se ha seleccionado un archivo
    datos=open (file);%cargamos los valores de las variables
    %fijamos en la ventana los valores correspondientes
    cuant=datos.cuant;
    orden=datos.orden;
```

```
    sat=datos.sat;
    G=datos.G;
    slider_value=datos.slider_value;
    m_reset=0;
    save sd_arquitectura orden cuant sat m_reset;
    g_reset=0;
    save sd_matrizG G g_reset slider_value;
    s_reset=1;
    salidas_plot=length(orden);
    save sd_salida_aux s_reset salidas_plot;
    jitter_act=datos.jitter_act;
    var_g_act=datos.var_g_act;
    perdidas_act=datos.perdidas_act;
    dt_jitter=datos.dt_jitter;
    m_jitter=datos.m_jitter;
    dt_var_g=datos.dt_var_g;
    m_var_g=datos.m_var_g;
    corregir_canc=datos.corregir_canc;
    coef_perdidas=datos.coef_perdidas;
    save sd_nolineal jitter_act var_g_act perdidas_act dt_jitter m_jitter dt_var_g m_var_g corregir_canc
coef_perdidas;
    GUI_modulador('varargin')
    if(jitter_act==1)
        GUI_entrada%si esta activado el jitter la entrada ha de cambiar
    end
end

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

%Regogemos el nombre que el usuario desee para el archivo
file=uiputfile('*.mat','Guardar modulador');
%Cargamos los valores del modulador configurados
load sd_arquitectura;
load sd_matrizG;
load sd_nolineal;
%los salvamos
save(file,'orden','cuant','sat','G','slider_value','jitter_act','var_g_act','perdidas_act','dt_jitter','m_jitter','dt
_var_g','m_var_g','corregir_canc','coef_perdidas');

% --------------------------------------------------------------------
```

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

printdlg(handles.GUI_modulador)

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

%Cerramos todo abriendo una question box
selection = questdlg(['¿Quiere cerrar ' get(handles.GUI_modulador,'Name') '?'],...
                  ['Close ' get(handles.GUI_modulador,'Name') '...'],...
                  'Si','No','Si');
if strcmp(selection,'No')
   return;
end

delete(handles.GUI_modulador)

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


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

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

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

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


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

!ayuda.htm
```

```
function varargout = GUI_salida(varargin)
% GUI_SALIDA M-file for GUI_salida.fig
%      GUI_SALIDA, by itself, creates a new GUI_SALIDA or raises the existing
%      singleton*.
%
%      H = GUI_SALIDA returns the handle to a new GUI_SALIDA or the handle to
%      the existing singleton*.
%
%      GUI_SALIDA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI_SALIDA.M with the given input arguments.
%
%      GUI_SALIDA('Property','Value',...) creates a new GUI_SALIDA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUI_salida_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUI_salida_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
```

```
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_salida

% Last Modified by GUIDE v2.5 03-Dec-2003 23:49:01

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

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

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GUI_salida wait for user response (see UIRESUME)
% uiwait(handles.GUI_salida);
```

```
%cargamos sd_salida_aux para comprobar el estado de s_reset
load sd_salida_aux
if(s_reset==0); % si no ha sido reseteada
   if(get(handles.s_fft,'value')) %si esta marcada la FFT
      s_fft_Callback(hObject, eventdata, handles); %realiza la FFT de la salida
   elseif (get(handles.s_psd,'value'))%si esta marcada la PSD
      s_psd_Callback(hObject, eventdata, handles);%realiza la PSD de la salida
   elseif (get(handles.s_t,'value'))%si esta marcada la representacion temporal
      s_t_Callback(hObject, eventdata, handles);%realiza la representacion en tiempo
   else
      %Fijamos por defecto la psd
      set(handles.s_t, 'Value', 0);
      set(handles.s_fft, 'Value', 0);
      set(handles.s_psd, 'Value', 1);
      s_psd_Callback(hObject, eventdata, handles);%realiza la PSD de la salida
   end

   load sd_salida
   load sd_entrada
   load sd_arquitectura

   %calculamos SNR

   if(get(handles.BW_fijo,'value')==1)
      BW_fijo_Callback(hObject, eventdata, handles);
   else
      BW_caida_Callback(hObject, eventdata, handles);
   end
end

handles.flag=0;%indica que no hay barra de herramientas activa
guidata(hObject,handles);


% --- Outputs from this function are returned to the command line.
function varargout = GUI_salida_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 on button press in simular_pushbutton.
function simular_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to simular_pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%cargamos los valores configurados en la entrada
load sd_entrada;
%cargamos el modulador
load sd_arquitectura;%arquitectura del modulador
load sd_matrizG;%valor de los coeficientes
%cargamos el archivo de variables auxiliares
load sd_salida_aux;
%cargamos posibles no linealidades
load sd_nolineal;

%Simulamos variaciones en los coeficientes de los integradores
G_var=G+dt_var_g*randn(size(G))+m_var_g;
if(corregir_canc==1)%si queremos que los coeficientes de cancelacion se modifiquen con las variaciones
   G=G_var;%los coeficientes de cancelacion tendran en cuenta las variaciones ya que a cancela le pasamos G
end



%Realizamos la modulacion sigma-delta en cascada
salida_precanc=cascada(orden,x,G_var,cuant,sat,coef_perdidas);%salida antes de la cancelacion de ruido
%Realizamos la cancelacion analogica del ruido
salida_canc=cancela(salida_precanc,orden,G);%salida despues de la cancelacion de ruido


%salvamos ambas salidas
save sd_salida salida_canc salida_precanc;


%calculamos SNR
%procede a enventanar
n_vent=get(handles.s_vent,'value'); %captura la ventana seleccionada
salida=ventana(salida_canc,n_vent);
%realizamos ahora la FFT
[salida,f]=sd_fft(salida,Fs);
S_dB=20*log10(abs(salida));
```

```
%al simular por defecto solo mostramos la salida del modulador completo
salidas_plot=length(orden);%salidas_plot indica que salidas se representaran
%salvamos las variables auxiliares
save sd_salida_aux salidas_plot s_reset;

if(get(handles.BW_fijo,'value')==1)
   BW_fijo_Callback(hObject, eventdata, handles)
else
   BW_caida_Callback(hObject, eventdata, handles)
end

%Representamos graficamente la simulacion
if(get(handles.s_fft,'value')) %si esta marcada la FFT
   s_fft_Callback(hObject, eventdata, handles); %realizamos la FFT
elseif (get(handles.s_psd,'value'))%si esta marcada la PSD
   s_psd_Callback(hObject, eventdata, handles);%realizamos la PSD
elseif (get(handles.s_t,'value'))%si esta marcada la representacion temporal
   s_t_Callback(hObject, eventdata, handles);%representamos en el dominio del tiempo
else
   %Fijamos por defecto la psd
   set(handles.s_t, 'Value', 0);
   set(handles.s_fft, 'Value', 0);
   set(handles.s_psd, 'Value', 1);
   s_psd_Callback(hObject, eventdata, handles);%realizamos la PSD
end


% --- Executes during object creation, after setting all properties.
function s_vent_CreateFcn(hObject, eventdata, handles)
% hObject    handle to s_vent (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 s_vent.
function s_vent_Callback(hObject, eventdata, handles)
% hObject    handle to s_vent (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 s_vent contents as cell array
%        contents{get(hObject,'Value')} returns selected item from s_vent

%Cuando se da algun cambio en el enventanado, representamos la salida
%automaticamente
if(get(handles.s_fft,'value')) %si esta marcada la FFT
   s_fft_Callback(hObject, eventdata, handles); %realiza la FFT
elseif (get(handles.s_psd,'value'))%si esta marcada la PSD
   s_psd_Callback(hObject, eventdata, handles); %realiza la PSD
elseif (get(handles.s_t,'value')) %si esta marcado el dominio temporal
   s_t_Callback(hObject, eventdata, handles); %representa en el tiempo
end

load sd_salida
load sd_salida_aux
load sd_entrada
load sd_arquitectura
%calculamos SNR
if(get(handles.BW_fijo,'value')==1)
   BW_fijo_Callback(hObject, eventdata, handles);
else
   BW_caida_Callback(hObject, eventdata, handles);
end

% --- Executes on button press in s_fft.
function s_fft_Callback(hObject, eventdata, handles)
% hObject    handle to s_fft (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 s_fft

%Ajustamos los marcadores mutuamente excluyentes
set(handles.s_t, 'Value', 0);
set(handles.s_fft, 'Value', 1); %marcamos la FFT
set(handles.s_psd, 'Value', 0);
```

```
%Calculamos la fft

%cargamos datos
load sd_entrada
load sd_salida
load sd_salida_aux
%realizamos el enventanado
n_vent=get(handles.s_vent,'value'); %captura la ventana seleccionada
salida_vent=ventana(salida_canc,n_vent); %procede a enventanar
%realizamos ahora la FFT
[salida_fft,F]=sd_fft(salida_vent,Fs);

%representamos la FFT

total_plots=length(salidas_plot); %numero total de salidas a representar
%generamos la cadena de caracteres con instruccion para representar las salidas
s_plot=['plot(F,20*log10(abs(salida_fft(',int2str(salidas_plot(1)),',:)))'];
for(i=2:1:total_plots)
   s_plot=[s_plot,',F,20*log10(abs(salida_fft(',int2str(salidas_plot(i)),',:)))'];
end
s_plot=[s_plot,');'];
%seleccionamos los ejes adecuados
axes(handles.ejes_salida);
%calculamos los limites verticales de representacion
y_max=0;
y_min=0;
for(i=1:1:total_plots)
   temp1=max(20*log10(abs(salida_fft(salidas_plot(i),:))));
   if(temp1>y_max)
      y_max=temp1;
   end
   temp2=min(20*log10(abs(salida_fft(salidas_plot(i),:))));
   if(temp2<y_min)
      y_min=temp2;
   end
end
%evaluamos la cadena generada anteriormente
eval(s_plot);%dibuja las salidas
%fijamos los ejes para mostrar solo las frecuencias positivas
axis([0,F(end),y_min, y_max]);
xlabel('Frecuencia');
ylabel('Magnitud [dB]');
set(handles.ejes_salida,'XMinorTick','on')
```

```
grid on

%Calculamos la SNR
if(get(handles.BW_fijo,'value')==1)
   BW_fijo_Callback(hObject, eventdata, handles);
else
   BW_caida_Callback(hObject, eventdata, handles);
end

% --- Executes on button press in s_psd.
function s_psd_Callback(hObject, eventdata, handles)
% hObject    handle to s_psd (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 s_psd

set(handles.s_t, 'Value', 0);
set(handles.s_fft, 'Value', 0);
set(handles.s_psd, 'Value', 1); %marcamos la PSD

%Calculamos la psd

%cargamos datos
load sd_entrada; %es necesario conocer la frecuencia de muestreo Fs
load sd_salida
load sd_salida_aux %para ver las que tenemos que representar "salidas_plot"

%capturamos la NFFT, temaño de los intervalos de la PSD
NFFT=str2double(get(handles.s_nfft,'string'));
%Comprobamos primero si las partes en que vamos a dividir la señal son mayores que la propia señal
if (NFFT>length(salida_canc))%si se señala una NFFT mayor que la longitud de las secuencias
   disp('Warning: max(NFFT)=length(salida_canc)')
   NFFT=length(salida_canc)%en ese caso fijamos la particion al tamaño de la señal
   set(handles.s_nfft,'string',NFFT)%fijamos el valor maximo permitido
end

%comprobamos que ventana es la seleccionada
n_vent=get(handles.s_vent,'value');
if (n_vent==1), %calculamos la ventana correspondiente
   ventana = ['boxcar(',int2str(NFFT),')'];
elseif (n_vent==2),
   ventana = ['blackman(',int2str(NFFT),')'];
```

```
elseif (n_vent==3),
   ventana = ['hanning(',int2str(NFFT),')'];
elseif (n_vent==4),
   ventana = ['hamming(',int2str(NFFT),')'];
end;

%realizamos la PSD con los datos obtenidos
[salida_psd,F]=sd_psd(salida_canc,NFFT,Fs,eval(ventana));

%Representamos la PSD graficamente

%calculamos el numero total de salidas a representar
total_plots=length(salidas_plot);
%generamos la cadena de caracteres con la instruccion para representar
s_plot=['plot(F,10*log10(salida_psd(',int2str(salidas_plot(1)),',:))'];
for(i=2:1:total_plots)
   s_plot=[s_plot,',F,10*log10(salida_psd(',int2str(salidas_plot(i)),',:))'];
end
s_plot=[s_plot,');'];
%seleccionamos los ejes apropiados
axes(handles.ejes_salida);
%evaluamos la instruccion de la cadena de caracteres
eval(s_plot);
xlabel('Frecuencia');
ylabel('Magnitud [dB]');
set(handles.ejes_salida,'XMinorTick','on')
grid on

%Calculamos la SNR
if(get(handles.BW_fijo,'value')==1)
   BW_fijo_Callback(hObject, eventdata, handles);
else
   BW_caida_Callback(hObject, eventdata, handles);
end

% --- Executes on button press in s_t.
function s_t_Callback(hObject, eventdata, handles)
% hObject    handle to s_t (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 s_t
```

```
set(handles.s_t, 'Value', 1); %marcamos la representacion temporal
set(handles.s_fft, 'Value', 0);
set(handles.s_psd, 'Value', 0);

%cargamos datos
load sd_salida
load sd_entrada
load sd_salida_aux

%calculamos el numero de salidas a representar
total_plots=length(salidas_plot);
%generamos la cedena de caracteres con la instruccion para representar
s_plot=['plot(kTs,salida_canc(',int2str(salidas_plot(1)),',:),''X'''];
for(i=2:1:total_plots)
    s_plot=[s_plot,',kTs,salida_canc(',int2str(salidas_plot(i)),',:),''X'''];
end
s_plot=[s_plot,');'];
%seleccionamos los ejes adecuados
axes(handles.ejes_salida);
%evaluamos la intruccion de la cadena de caracteres
eval(s_plot);
xlabel('Tiempo [seg.]');
ylabel('Amplitud');
set(handles.ejes_salida,'XMinorTick','on')
grid on

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

%abrimos el configurador que permite seleccionar las salidas a representar
configurador_salidas

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


% --------------------------------------------------------------------
```

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



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


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

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

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


% --- Executes during object creation, after setting all properties.
function s_nfft_CreateFcn(hObject, eventdata, handles)
% hObject    handle to s_nfft (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 s_nfft_Callback(hObject, eventdata, handles)
% hObject    handle to s_nfft (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 s_nfft as text
%        str2double(get(hObject,'String')) returns contents of s_nfft as a double



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

file = uigetfile('*.mat','Cargar simulacion');%recogemos es archivo a cargar
if ~isequal(file, 0)%si se ha seleccionado un archivo
    datos=open (file);%cargamos los valores de las variables
    %fijamos en la ventana los valores correspondientes
    salida_canc=datos.salida_canc;
    salida_precanc=datos.salida_precanc;
    Fs=datos.Fs;
    kTs=datos.kTs;
    save sd_salida salida_canc salida_precanc Fs kTs; %las dos ultimas las necesitamos para
representar;


    %modulador
    cuant=datos.cuant;
    orden=datos.orden;
    sat=datos.sat;
    G=datos.G;
    slider_value=datos.slider_value;
    m_reset=0;
    save sd_arquitectura orden cuant sat m_reset;
    g_reset=0;
```

```
    save sd_matrizG G g_reset slider_value;
    s_reset=1;
    salidas_plot=length(orden);
    save sd_salida_aux s_reset salidas_plot;
    %no linealidades
    jitter_act=datos.jitter_act;
    var_g_act=datos.var_g_act;
    perdidas_act=datos.perdidas_act;
    dt_jitter=datos.dt_jitter;
    m_jitter=datos.m_jitter;
    dt_var_g=datos.dt_var_g;
    m_var_g=datos.m_var_g;
    corregir_canc=datos.corregir_canc;
    coef_perdidas=datos.coef_perdidas;
    save sd_nolineal jitter_act var_g_act perdidas_act dt_jitter m_jitter dt_var_g m_var_g corregir_canc
coef_perdidas;
    GUI_modulador('varargin')
    %entrada
    Fx=datos.Fx;
    M=datos.M;
    n_ciclos=datos.n_ciclos;
    B=datos.B;
    A=datos.A;
    save sd_entrada  A n_ciclos M Fx B;
    %generamos la entrada propiamente dicha

    GUI_entrada
    %GUI_entrada('entrada_Callback',hObject, eventdata, handles);



    %representamos la salida
    if(get(handles.s_fft,'value'))
       s_fft_Callback(hObject, eventdata, handles);
    elseif (get(handles.s_psd,'value'))
       s_psd_Callback(hObject, eventdata, handles);
    elseif (get(handles.s_t,'value'))
       s_t_Callback(hObject, eventdata, handles);
    else
       %Fijamos por defecto la psd
       set(handles.s_t, 'Value', 0);
       set(handles.s_fft, 'Value', 0);
       set(handles.s_psd, 'Value', 1);
       s_psd_Callback(hObject, eventdata, handles);
```

```
    end
end

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

file=uiputfile('*.mat','Guardar simulacion');
% Get user input from GUI
load sd_entrada;
load sd_arquitectura;
load sd_matrizG;
load sd_salida ;
load sd_nolineal;

%Grabamos las variables de interes
save (file, 'salida_canc', 'salida_precanc','x', 'Fs', 'kTs', 'Fx', 'M', 'n_ciclos', 'B', 'A','orden','cuant','sat','G','slider_value','jitter_act','var_g_act','perdidas_act','dt_jitter','m_jitter','dt_var_g','m_var_g','corregir_canc','coef_perdidas'); %, 'NFFT'

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

printdlg(gcbf)

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

printpreview(gcbf)

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

%Cerramos todo abriendo una question box
```

```
selection = questdlg(['¿Quiere cerrar ' get(handles.GUI_salida,'Name') '?'],...
                     ['Close ' get(handles.GUI_salida,'Name') '...'],...
                     'Si','No','Si');
if strcmp(selection,'No')
   return;
end

delete(handles.GUI_salida)


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


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

if(handles.flag==0)%si no hay barra activada
   fig=gcf;
   htoolbar.htoolbar = uitoolbar('Parent',fig);%hObject);
   % Render Print buttons (Print, Print Preview)
   htoolbar.hprintbtns = render_sptprintbtns(htoolbar.htoolbar);
   % Render the annotation buttons (Edit Plot, Insert Arrow, etc)
   htoolbar.hscribebtns = render_sptscribebtns(htoolbar.htoolbar);
   % Render the zoom buttons
   htoolbar.hzoombtns = render_zoombtns(htoolbar.htoolbar);
   %Creamos una clase en la estructura para almacenar el handles de herramientas
   handles.herramientas=htoolbar.htoolbar;
   handles.flag=1;%indica que hemos activado la barra de herramientas
   guidata(hObject,handles);
   %herramientas=1;
   %save aux_herramientas htoolbar herramientas;
end

% --------------------------------------------------------------------
function desactiva_herramientas_Callback(hObject, eventdata, handles)
% hObject    handle to desactiva_herramientas (see GCBO)
```

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

set (handles.herramientas,'visible','off');
handles.flag=0;%indica que no hay barra de herramientas activa
guidata(hObject,handles);


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

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

zoom out
zoom off


% --- Executes during object creation, after setting all properties.
function BW_signal_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BW_signal (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 BW_signal_Callback(hObject, eventdata, handles)
% hObject    handle to BW_signal (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 BW_signal as text
%        str2double(get(hObject,'String')) returns contents of BW_signal as a double

BW_fijo_Callback(hObject, eventdata, handles)


% --- Executes during object creation, after setting all properties.
function dB_caida_CreateFcn(hObject, eventdata, handles)
% hObject    handle to dB_caida (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 dB_caida_Callback(hObject, eventdata, handles)
% hObject    handle to dB_caida (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 dB_caida as text
%        str2double(get(hObject,'String')) returns contents of dB_caida as a double

BW_caida_Callback(hObject, eventdata, handles)

% --- Executes on button press in BW_fijo.
function BW_fijo_Callback(hObject, eventdata, handles)
% hObject    handle to BW_fijo (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 BW_fijo
set(handles.BW_fijo,'value',1);
set(handles.BW_caida,'value',0);
```

```
set(handles.BW_signal,'enable','on');
set(handles.dB_caida,'enable','off');

load sd_salida_aux
load sd_salida
load sd_entrada
load sd_arquitectura
%calculamos SNR

%procede a enventanar
n_vent=get(handles.s_vent,'value'); %captura la ventana seleccionada
if(get(handles.s_fft,'value')==1)
   salida=ventana(salida_canc,n_vent);
   [salida,f]=sd_fft(salida,Fs);
   S_dB=20*log10(abs(salida));
else
   %capturamos la NFFT, temaño de los intervalos de la PSD
   NFFT=str2double(get(handles.s_nfft,'string'));
   %Comprobamos primero si las partes en que vamos a dividir la señal son mayores que la propia señal
   if (NFFT>length(salida_canc))%si se señala una NFFT mayor que la longitud de las secuencias
      disp('Warning: max(NFFT)=length(salida_canc)')
      NFFT=length(salida_canc)%en ese caso fijamos la particion al tamaño de la señal
      set(handles.s_nfft,'string',NFFT)%fijamos el valor maximo permitido
   end

   %comprobamos que ventana es la seleccionada
   if (n_vent==1), %calculamos la ventana correspondiente
      ventana_selecc = ['boxcar(',int2str(NFFT),')'];
   elseif (n_vent==2),
      ventana_selecc = ['blackman(',int2str(NFFT),')'];
   elseif (n_vent==3),
      ventana_selecc = ['hanning(',int2str(NFFT),')'];
   elseif (n_vent==4),
      ventana_selecc = ['hamming(',int2str(NFFT),')'];
   end;

  [salida,f]=sd_psd(salida_canc,NFFT,Fs,eval(ventana_selecc));
  S_dB=10*log10(abs(salida));
end

BW=str2num(get(handles.BW_signal,'string'));%capturamos el ancho de banda introducido
BW_signal=BW*ones(1,length(orden));%el ancho de banda sera el mismo para todos los moduladores
Fmin=Fx-BW/2;%limite inferior del BW
```

```
Fmax=Fx+BW/2;%limite superior del BW
SNR=calcula_SNR(S_dB,Fx,Fs,B,Fmin,Fmax,f); %Calculamos la SNR

%representa, modulador, SNR y ancho de banda de señal
for(i=1:1:length(salidas_plot))
    cadena=['set(handles.m_',int2str(i),',''string'',''',int2str(orden(1:salidas_plot(i))),''');'];
    eval(cadena);
    cadena=['set(handles.SNR_',int2str(i),',''string'',''',num2str(SNR(salidas_plot(i))),' dB'');'];
    eval(cadena);
    cadena=['set(handles.BW_',int2str(i),',''string'',''',num2str(BW_signal(salidas_plot(i))),' Hz'');'];
    eval(cadena);
end
for(i=length(salidas_plot)+1:1:10)
   cadena=['set(handles.m_',int2str(i),',''string'','''');'];
   eval(cadena);
   cadena=['set(handles.SNR_',int2str(i),',''string'','''');'];
   eval(cadena);
   cadena=['set(handles.BW_',int2str(i),',''string'','''');'];
   eval(cadena);
end

% --- Executes on button press in BW_caida.
function BW_caida_Callback(hObject, eventdata, handles)
% hObject    handle to BW_caida (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 BW_caida

set(handles.BW_fijo,'value',0);
set(handles.BW_caida,'value',1);
set(handles.BW_signal,'enable','off');
set(handles.dB_caida,'enable','on');

load sd_salida
load sd_salida_aux
load sd_entrada
load sd_arquitectura
%calculamos SNR

%procede a enventanar
n_vent=get(handles.s_vent,'value'); %captura la ventana seleccionada
if(get(handles.s_fft,'value')==1)
```

```
    salida=ventana(salida_canc,n_vent);
    [salida,f]=sd_fft(salida,Fs);
    S_dB=20*log10(abs(salida));
else
    %capturamos la NFFT, temaño de los intervalos de la PSD
    NFFT=str2double(get(handles.s_nfft,'string'));
    %Comprobamos primero si las partes en que vamos a dividir la señal son mayores que la propia señal
    if (NFFT>length(salida_canc))%si se señala una NFFT mayor que la longitud de las secuencias
        disp('Warning: max(NFFT)=length(salida_canc)')
        NFFT=length(salida_canc)%en ese caso fijamos la particion al tamaño de la señal
        set(handles.s_nfft,'string',NFFT)%fijamos el valor maximo permitido
    end

    %comprobamos que ventana es la seleccionada
    if (n_vent==1), %calculamos la ventana correspondiente
        ventana_selecc = ['boxcar(',int2str(NFFT),')'];
    elseif (n_vent==2),
        ventana_selecc = ['blackman(',int2str(NFFT),')'];
    elseif (n_vent==3),
        ventana_selecc = ['hanning(',int2str(NFFT),')'];
    elseif (n_vent==4),
        ventana_selecc = ['hamming(',int2str(NFFT),')'];
    end;

   [salida,f]=sd_psd(salida_canc,NFFT,Fs,eval(ventana_selecc));
   S_dB=10*log10(abs(salida));
end

caida=str2num(get(handles.dB_caida,'string'));%capturamos la caida en dB
for(i=1:1:length(orden))
     [BW_signal(i),Fmin,Fmax]=calcula_BW(S_dB(i,:),f, Fx,caida);%calculamos el BW
     SNR(i)=calcula_SNR(S_dB(i,:),Fx,Fs,B,Fmin,Fmax,f);%calculamos el SNR
end

%representa, modulador, SNR y ancho de banda de señal
for(i=1:1:length(salidas_plot))
    cadena=['set(handles.m_',int2str(i),',''string'',''',int2str(orden(1:salidas_plot(i))),''');'];
    eval(cadena);
    cadena=['set(handles.SNR_',int2str(i),',''string'',''',num2str(SNR(salidas_plot(i))),' dB'');'];
    eval(cadena);
    cadena=['set(handles.BW_',int2str(i),',''string'',''',num2str(BW_signal(salidas_plot(i))),' Hz'');'];
    eval(cadena);
end
```

```
for(i=length(salidas_plot)+1:1:10)
   cadena=['set(handles.m_',int2str(i),',''string'','''');'];
   eval(cadena);
   cadena=['set(handles.SNR_',int2str(i),',''string'','''');'];
   eval(cadena);
   cadena=['set(handles.BW_',int2str(i),',''string'','''');'];
   eval(cadena);
end


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

!ayuda.htm
```

```
function [modulado,intermedio]=m1(x,z,g11,g12,g1r,levels,sat,coef_perdidas,ult);
% Realiza una modulación sigma-delta de primer orden sobre un vector de entrada.
% Permite fijar las características del cuantizador.
% Devuelve la señal modulada y la entrada del cuantizador (permite calcular el error de cuantización)
%
%PARAMETROS DE ENTRADA
%         x: vector de entrada 1 a modular
%         z: vector de entrada 2 a modular (se resta a la anterior)
%         g11: coeficiente 'g' que multiplica a la entrada 1
%         g12: coeficiente 'g' que multiplica a la entrada 2
%    g1r: coeficiente 'g' que multiplica a la realimentación
%         levels: número de escalones de cuantización del cuantizador
%         sat: nivel de saturación del cuantizador
%    coef_perdidas:coeficiente de perdidas en integradores. Comprendido entre 0 y 1
%
%PARAMETROS DE SALIDA
%         modulado: vector salida del modulador. Señal modulada.
%         intermedio: vector intermedio con las entradas al cuantizador.
%
% Ver tambien configurador_g, cascada, m2, cuantiza, cancela
%

%si solo nos dan la entrada a modular (un unico parametro)
if nargin==1
```

```
    z=zeros(1,length(x));
    g12=0
    g11=1;
    g1r=g11;
    coef_perdidas=0;
    ult=0;
end

%Modulacion sigma-delta de primer orden

%inicializamos los parametros
a(1)=g11*x(1)-g12*z(1);
b(1)=0;
y(1)=0;
%modulamos
for(n=2:1:length(x))
   b(n)=a(n-1)+(1-coef_perdidas)*b(n-1);%salida del filtro
   y(n)=cuantiza(b(n),levels,2*sat);%cuantizamos b(n), con "levels" niveles, y rango de cuantizacion igual al doble del nivel de saturacion
   a(n)=g11*x(n)-g12*z(n)-g1r*y(n);%realimentacion
end
modulado=y;
intermedio=b;%este valor es necesario para el siguiente modulo en cascada. Entrada al cuantizador.
```

```
function [modulado,intermedio]=m2(x,z,g11,g12,g1r,g21,g22,levels,sat,coef_perdidas,ult);
% Realiza una modulación sigma-delta de segundo orden sobre un vector de entrada.
% Permite fijar las características del cuantizador.
% Incluye modelado de perdidas en los integradores
% Devuelve la señal modulada y la entrada del cuantizador (permite calcular el error de cuantización)
%
%PARAMETROS DE ENTRADA
%         x: vector de entrada 1 a modular
%         z: vector de entrada 2 a modular (se resta a la anterior)
%         g11: coeficiente 'g' que multiplica a la entrada 1 del primer integrador
%         g12: coeficiente 'g' que multiplica a la entrada 2 del primer integrador
%    g1r: coeficiente 'g' que multiplica a la realimentación
%    g21: coeficiente 'g' que multiplica a la entrada 1 del segundo integrador
%         g22: coeficiente 'g' que multiplica a la entrada 2 del segundo
%              integrador (realimentada)
%         levels: número de escalones de cuantización del cuantizador
%         sat: nivel de saturación del cuantizador
%    coef_perdidas:coeficiente de perdidas en integradores. Comprendido entre 0 y 1
```

```
%
%PARAMETROS DE SALIDA
%       modulado: vector salida del modulador. Señal modulada.
%       intermedio: vector intermedio con las entradas al cuantizador.
%
% Ver tambien configurador_g, cascada, m1, cuantiza, cancela
%

%si solo nos dan la entrada a modular (un unico parametro)
if nargin==1
   z=zeros(1,length(x));
   g12=0
   g11=1;
   g1r=g11;
   g21=1;
   g22=2*g1r*g21;
   coef_perdidas=0;
end



%Modulacion sigma-delta de segundo orden

%inicializamos los parametros
a1(1)=g11*x(1)-g12*z(1);
a2(1)=0;
b1(1)=0;
b2(1)=0;
y(1)=0;
%modulamos
for(n=2:1:length(x))
  b2(n)=a2(n-1)+(1-coef_perdidas)*b2(n-1);%salida del filtro 2
  y(n)=cuantiza(b2(n),levels,2*sat);%cuantizamos b(n), con "levels" niveles, y rango de cuantizacion igual al doble del nivel de saturacion
  b1(n)=a1(n-1)+(1-coef_perdidas)*b1(n-1);%salida del filtro 1
  a2(n)=g21*b1(n)-g22*y(n);%realimentacion 2
  a1(n)=g11*x(n)-g12*z(n)-g1r*y(n);%realimentacion 1
end
modulado=y;
intermedio=b2;%este valor hace falta para el siguiente modulo en cascada
```

```
function [muestreada,Fs,kTs] =muestrea (Fx,M,n_ciclos,B,A,m,v)
%DESCRIPCION: muestrea una señal sinusoidal para los parametros de entrada
```

```
%PARAMETROS ENTRADA:
%       Fx: Frecuencia del tono.
%       M: Tasa de sobremuestreo.
%       n_ciclos: Número de periodos del tono.
%       B: Ancho de banda.
%       A: Amplitud.
%   m: media de efecto jitter
%   v: varianza de efecto jitter
%PARAMETROS SALIDA:
%       Muestreada: secuencia resultante.
%       Fs: Frecuencia de muestreo.
%       kTs: Eje para representación temporal.
%
%OBSERVACION: si no se pasa la media y la varianza no hay efecto jitter

if (nargin==5)
   m=0;
   v=0;
end

Fn=2*B; %Frecuencia de Niquist respecto a B
Fs=M*Fn; %Frecuencia de sobremuestreo
Ts=(1/Fs);%Periodo de sobremuestreo
kTs=[0:Ts:(n_ciclos*(1/Fx))];%generamos el vector correspondiente al eje x en representacion temporal
kTs=kTs+sqrt(v)*randn(size(kTs))+m;%efecto jitter
muestreada=A*sin(2*pi*Fx*kTs);%generamos la frecuencia para el tono deseado
```

```
function reset_G(h_slider,h_edit)
% "Callback" del botón "reset" la interfaz "configurador_g".
% Resetea el configurador g y fija a cero todos los valores
% PARAMETROS DE ENTRADA:
%       h_slider: vector con los manejadores de los deslizadores
%       h_edit: vector con los manejadores de las cajas de texto editables
%
% Ver tambien: actualiza_g, acepta_G.m, cancela_G.m, actualiza_G.m, g_edit.m, g_slider.m

g_reset=1 %activamos la variable de control de reset
save sd_matrizG g_reset; %guardamos la variable de control en sd_matrizG
for(i=1:1:length(h_slider))
   set(h_slider(i),'value',0); %fijamos a 0 todos los deslizadores
   set(h_edit(i),'string','0'); %fijamos a 0 todas las cajas de texto editables
```

```
end
```

```
disp('Simulador de convertidores Sigma-Delta cascada tiempo discreto')
disp('Para obtener ayuda escribir: "!ayuda.htm" en la linea de comandos de MATLAB')

warning off MATLAB:divideByZero;

%Inicializamos variables de entrada
Fx = 0;%frecuencia de la señal sinusoidal de entrada
M = 0;%tasa de sobremuestreo
n_ciclos =0;%numero de ciclos de la señal sinusoidal de entrada
B = 0;%ancho de banda
A = 0;%amplitud de la señal sinusoidal de entrada
save sd_entrada A n_ciclos M Fx B;

%reseteamos el configurador de ganancias de los amplificadores
g_reset=1;
save sd_matrizG g_reset;

%reseteamos la salida
s_reset=1;
save sd_salida_aux s_reset;

%reseteamos el configurador de arquitectura
m_reset=1;
save sd_arquitectura m_reset;

%inicializamos las no-linealidades
jitter_act=0;
var_g_act=0;
perdidas_act=0;
dt_jitter=0;
m_jitter=0;
dt_var_g=0;
m_var_g=0;
coef_perdidas=0;
save sd_nolineal jitter_act var_g_act perdidas_act dt_jitter m_jitter dt_var_g m_var_g coef_perdidas;

%Abrimos las ventanas de interfaz de usuario
GUI_entrada %Interfaz de entrada
GUI_modulador %interfaz de modulador
GUI_salida %interfaz de salida
```

```
function [Yw,F]=sd_fft(Y,Fs)
%Aplica la FFT a las filas de una matriz de entrada
%PARAMETROS DE ENTRADA:
%       Y: matriz de vectores por filas.
%       Fs: frecuencia de muestreo.
%PARAMETROS DE SALIDA:
%       Yw: matriz de transformadas PSD por filas.
%       F: eje de frecuencias.
%
%Ver tambien: transforma.m

aux=size(Y);
n_etapas=aux(1); %comprobamos el numero de etapas de la cascada
%realizamos la FFT por filas
for (i=1:1:n_etapas)
Yw(i,:)=transforma(Y(i,:)); %transformamos cada salida y las almacenamos por filas
end
%calculamos el eje x de frecuencias para la reresentacion
expo=ceil(log2(length(Y)));%Por defecto tomamos N como la potencia de 2 >= que la longitud de la secuencia para agilizar la fft
N=2^expo;
F=[-Fs/2:(Fs/N):(Fs/2)*((N-1)/N)];%de -pi a pi
```

```
function [Yw,F]=sd_psd(Y,NFFT,Fs,ventana)
%Aplica la PSD a las filas de una matriz de entrada
%PARAMETROS DE ENTRADA:
%       Y: matriz de vectores por filas.
%       NFFT
%       Fs: frecuencia de muestreo.
%       Ventana: enventanado a realizar.
%PARAMETROS DE SALIDA:
%       Yw: matriz de transformadas PSD por filas.
%       F: eje de frecuencias.

aux=size(Y);
n_etapas=aux(1); %comprobamos el numero de etapas o vectores

%Comprobamos primero si las partes en que vamos a dividir la señal son mayores que la propia señal
if (NFFT>aux(2))%si se señala una NFFT mayor que la longitud de las secuencias
   NFFT=aux(2);%en ese caso fijamos la particion al tamaño de la señal
```

```
end

%Aplicamos la PSD por filas
for (i=1:1:n_etapas)
[Yw(:,i),F]=psd(Y(i,:),NFFT,Fs,ventana); %transformamos cada fila de entrada y las almacenamos por filas en la salida
end
Yw=Yw';% trasponemos para organizar las salidas por filas
F=F';% trasponemos para devolver un vector fila
```

```
function transformada=transforma(secuencia,N);
%Realiza la transformada de Fourier discreta FFT de N puntos de un vector.
%  Si sólo se pasa un argumento, toma N como la potencia de 2 mayor o igual que la longitud de la secuencia para agilizar la FFT.
%  Si N es menor que la longitud de la secuencia, se trunca la secuencia.
%  Si N es mayor que la longitud de la secuencia; se rellena con ceros.
%  Se devuelve la FFT centrada en cero.
%PARAMERTOS DE ENTRADA
%       secuencia: vectores al que realizar la FFT.
%       N: Número de puntos de los que se desea realizar la transformada.
%PARAMETROS DE SALIDA
%       transformada: Transformada de Fourier discreta FFT de N puntos de secuencia.
%
% Ver tambien: sd_fft.m

if nargin==1
   expo=ceil(log2(length(secuencia)));%Por defecto tomamos N como la potencia de 2 >= que la longitud de la secuencia para agilizar la fft
   N=2^expo;
end
if(N<length(secuencia));
   disp('Warning: Se esta introduciendo efecto aliasing');
   disp('N<length(secuencia)');
else if (N>length(secuencia));
      disp('Warning: Rellenando con ceros. Zero padding');
      disp('N>length(secuencia)');
   end
end
transformada=(1/length(secuencia))*fftshift(fft(secuencia,N));%modulo de la fft de N puntos, centrando en w=0
%hemos escalado en magnitud para que no sea funcion de la longitud de 'secuencia'
```

```
function salida=ventana(entrada,v);
%Aplica enventanado por filas a una matriz
%
%PARAMETROS DE ENTRADA_
%   entrada: matriz de vectores ordenados por filas
%   v: seleccion de ventana
%       v=1: rectangular
%       v=2: blackman
%       v=3: hanning
%       v=4: hamming
%
%PARAMETROS DE SALIDA
%   salida: matriz de vectores enventanados ordenados por filas
%

aux=size(entrada);
n_etapas=aux(1); %comprobamos el numero de etapas de la cascada

tam=length(entrada);
if (v==1), %calculamos la ventana correspondiente
   window = boxcar(tam);
elseif (v==2),
   window = blackman(tam);
elseif (v==3),
   window = hanning(tam);
elseif (v==4),
   window = hamming(tam);
end;

for (i=1:1:n_etapas)
salida(i,:)=entrada(i,:).*window'; %aplicamos la ventana a cada fila y lo almacenamos tb por filas
end
```