Apéndice I

MANUAL DE PROCEDIMIENTOS PARA DESARROLLAR LOS ENSAYOS DE LABORATORIO

A.1. Inicialización del Simulador ISDB-T

Para iniciar el simulador se debe ejecutar el archivo:
medidas.m

Cuando éste archivo es abierto por primera vez se corre la simulación con los parámetros establecidos para la última simulación del modelo ISDB-T, desarrollado en Simulink.

Una vez corrido el modelo ISDB-T se toman las muestras necesarias para realizar el cálculo de los diferentes parámetros que se visualizan en la pantalla principal del simulador.

El Simulador de medición de parámetros de ISDB-T está compuesto de tres etapas. A continuación se describen cada una de ellas.

A.1.1. Etapas del simulador de medición de parámetros de ISDB-T

Transmisión: a partir del flujo de información binaria, se generan las tramas que serán transmitidas, se codifica y modula la información y se envía como una señal analógica en el tiempo.

Canal: la señal recorre diferentes caminos y puede verse degradada por las diferentes interferencias y el ruido que puede existir. Para esta simulación se utilizó un canal RICE para simular los efectos del multitrayecto y un canal AWGN para simular un canal afectado únicamente por ruido gaussiano.

Recepción: se recupera la señal en la medida de lo posible, se deshacen las etapas de transmisión, con el objetivo de volver a obtener las tramas iniciales.

A.2. Configuración de las Medidas

Para obtener las medidas de algunos tipos de señales, va a ser necesario que el usuario ingrese algunos parámetros relativos a las características particulares de las mismas, cuando estas difieran de la configuración por defecto (ultima configuración guardada).

A.2.1. Configuración de un Canal Digital ISDB-T/TB (COFDM)

Pulsar la tecla de CONF. (Configuración de medidas) para acceder al menú de CONFIGURACIÓN (Figura A.1). Los parámetros relativos a la modulación COFDM se describen a continuación:

Modo: Identificación del modo de transmisión basado en la separación de las frecuencias de las portadoras OFDM. Para este caso, la separación de frecuencia debe obligatoriamente ser de aproximadamente 4kHz, 2kHz ó 1kHz, respectivamente para los modos 1, 2 y 3. El número de portadoras varía dependiendo del modo, pero la tasa útil de cada modo debe necesariamente ser la misma. Para modificar su valor seleccionar con el mouse hasta posicionar el cursor sobre el campo Modo y entonces pulsarlo. Seleccionar de la lista desplegable el valor deseado y finalmente pulsarlo de nuevo para validar.

Modulación: Define el tipo de modulación. Al seleccionar esta función se des-

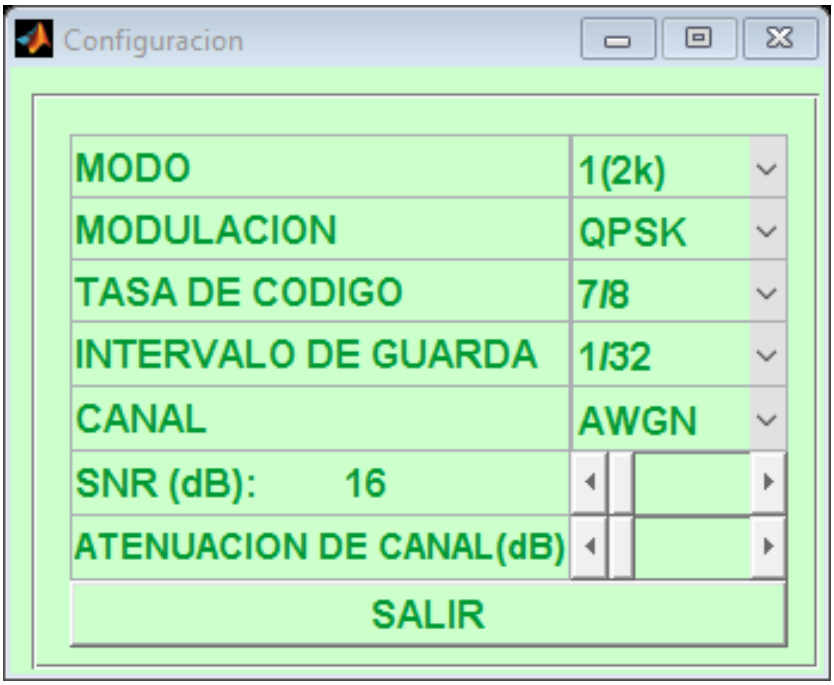

Figura A.1: Menú de Configuración

pliega un menú mediante el cual es posible seleccionar una de las siguientes modulaciones: QPSK, 16QAM o 64QAM.

Tasa de Código: También conocido como relación de Viterbi. Define la relación entre el número de bits de datos y el número de bits totales transmitidos (la diferencia corresponde al número de bits de control para la detección y recuperación de errores). Al seleccionar esta función se debe escoger el valor deseado de la lista desplegable dentro de las siguientes tasas $\frac{1}{2}$,$\frac{2}{3}$, $\frac{3}{4}$, $\frac{5}{6}$, $\frac{7}{8}$.

Guarda: El parámetro Intervalo de Guarda corresponde al tiempo muerto entre símbolos, su finalidad es permitir una detección correcta en situaciones de ecos por multitrayecto. Este parámetro se expresa en función de la duración del símbolo: $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{32}$.

Posicionar el cursor sobre el campo Guarda. Para definir su valor, se selecciona entre estos valores de la lista desplegable y finalmente pulsarlo para validar.

Canal: Se debe escoger de la lista desplegada la opción CANAL RICIAN para ver los efectos multitrayecto o AWGN para ver los efectos del ruido gaussiano. Al terminar la configuración se debe presionar obligatoriamente el botón SALIR para guardar los cambios hechos y actualizar la simulación, en caso contrario se guardan los cambios pero no se actualiza la simulación.

SNR (dB): Este parámetro establece el nivel de relación señal a ruido del canal AWGN ó el nivel del Factor K para el canal Rice. Este valor puede variar desde cero hasta 80 dB, según lo establecido en el slider de la derecha.

Atenuación de canal (dB): Este parámetro agrega atenuación al canal, el mínimo valor ajustable es 0 dB.

A.3. Selección de las Medidas

Potencia del Canal: La potencia del canal se mide asumiendo que la densidad espectral de potencia es uniforme en todo el ancho de banda del canal. Para que la lectura sea correcta es indispensable definir el parámetro Ancho de Banda. Este se encuentra configurado por defecto como 6MHz.

El nivel de ruido se mide en ausencia de la señal, por lo tanto es necesario configurar la frecuencia central y el ancho de banda correctamente para asi determinar la frecuencia fuera de banda y medir el valor de ruido a esa frecuencia.

C/N: Se calcula como la diferencia entre la potencia de la señal y la potencia de ruido. MER Relación de error de la modulación con indicación del margen de ruido para una señal ISDB-T/TB.

CBER: Medida del BER (tasa de error) para la señal digital antes de la corrección de errores (BER antes del FEC) para una señal ISDB-T/TB.

VBER: Medida del BER (tasa de error) para la señal digital después de la corrección de errores (BER después de Viterbi) para una señal ISDB-T/TB.

Medida de Potencia de un Canal

Al seleccionar el modo de medida POTENCIA CANAL en el monitor aparece la siguiente información:

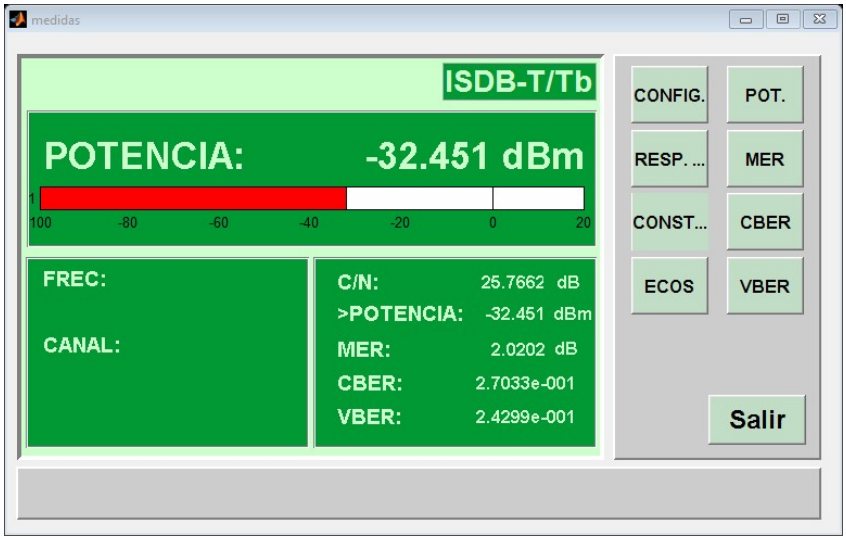

Figura A.2: Interfaz gráfica de la potencia

Además de la potencia del canal digital (-32.45 dBm en el ejemplo de la figura anterior) se muestra la frecuencia de sintonía o el canal, de acuerdo con el modo de sintonía seleccionado. Para que la medida de potencia de un canal digital sea correcta es imprescindible haber definido previamente el ancho de banda del canal mediante la función Ancho de Banda establecida en 6 MHz.

A.3.1. Medida del BER

Para seleccionar la de medida del BER:

1. Seleccionar la Configuración de Medidas de señales digitales pulsando la tecla CONF.

2. Seleccionar mediante la opción MODULACION del menú de CONFIGURACIÓN: QPSK, 16QAM o 64QAM

3. Introducir los parámetros relativos a la señal digital que aparecen en el menú de CONFIGURACIÓN de la medida, según se ha descrito anteriormente.

4. Seleccionar la opción salir del menú de CONFIGURACIÓN de las medidas.

Una vez establecidos los parámetros de la señal ISDB-T/TB, será posible medir el BER.

Se presentan dos medidas:
A continuación se presenta la *medida del BER antes de la corrección de errores*: **BER antes del FEC: CBER.**

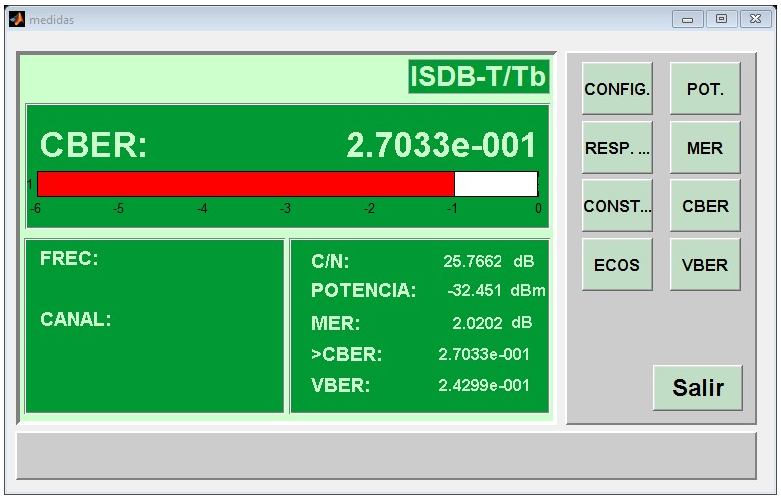

Figura A.3: Medida del BER

En un sistema de recepción de señal digital terrestre, tras el decodificador de señal COFDM se aplican dos métodos de corrección de errores. Obviamente,

cada vez que se aplica un corrector de errores sobre la señal digital, la tasa de error cambia, por lo que si se mide la tasa de error a la salida del demodulador de COFDM, después de Viterbi y a la salida del decodificador de Reed-Solomon se obtienen tasas de error distintas. El SIMULADOR ISDB-T/TB proporciona la medida del BER después de Viterbi (VBER) y el CBER a la salida del demodulador de COFDM.

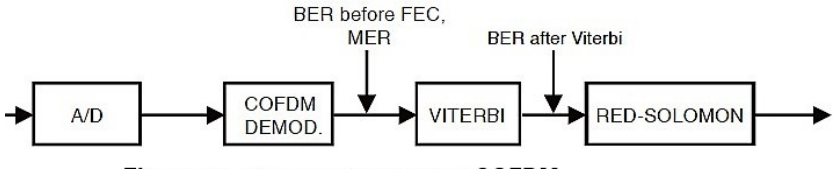

Figura A.4: Sistema de Recepción COFDM

La medida del BER se presenta en valor absoluto en notación científica (1,0 E-7 significa 1,0x10-7, es decir en valor medio un bit erróneo cada 10.000.000) y mediante una barra analógica (cuanto menor sea su longitud mejor será la calidad de la señal). La representación analógica se presenta sobre una escala logarítmica (no lineal), es decir, las marcas de la barra se corresponden con el exponente de la medida.

A.3.2. Medida del MER

Una vez establecidos los parámetros de recepción apropiados para la señal ISDB-T/TB, será posible medir el MER, pulsando la tecla medida del MER.

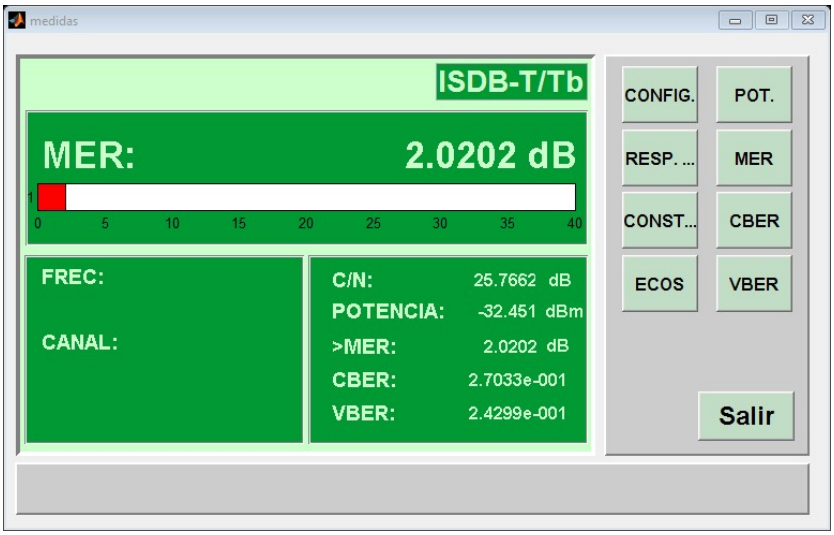

Figura A.5: Pantalla de medida del MER para señales ISDB-T/TB.

En primer lugar se presenta la *medida de la relación de error de modulación:* **MER**. Al medir la señal ISDB-T/Tb.

La relación de error de modulación (MER), utilizada en los sistemas digitales es análoga a la medida de Señal-Ruido (S/N) en los analógicos. El MER permite valorar que tanto se asemeja el diagrama de constelación de la señal recibida, con el diagrama de constelación ideal de la modulación digital.

A modo de ejemplo los demoduladores 64 QAM requieren un MER superior a 23 dB para operar. Si bien, es preferible contar con un margen de al menos 3 ó 4 dB para posibles degradaciones del sistema. Mientras los demoduladores 256 QAM requieren un MER superior a 28 dB con márgenes de al menos 3 dB. Habitualmente el valor máximo de MER visualizable en analizadores portátiles es de aproximadamente 34 dB.

A.3.3. Diagrama de Constelación

El diagrama de la constelación es una representación gráfica de los símbolos digitales recibidos en un periodo de tiempo. Existen distintos tipos de diagramas de constelación según el tipo de modulación. El SIMULADOR ISDB-T/TB representa las constelaciones de señales en ISDB-T/TB.

En el caso de un canal de transmisión ideal, sin ruido ni interferencias, todos los símbolos son reconocidos por el demodulador sin errores. En este caso, son representados en el diagrama de constelación como puntos bien definidos que impactan en la misma zona formando un punto muy concentrado.

El ruido y las interferencias provocan que el demodulador no siempre lea los símbolos de forma correcta. En este caso los impactos se dispersan y crean diferentes formas que permiten determinar visualmente el tipo de problema en la señal.

Cada tipo de modulación se representa de forma diferente. Una señal 16-QAM se representa en pantalla por un total de 16 zonas diferentes y una 64-QAM, se representa mediante un diagrama de 64 zonas diferentes y así sucesivamente.

Al acceder a la opción CONST. (Constelación), en la pantalla se irán registrando los impactos que producen los símbolos recibidos durante la trasmisión de la señal digital.

NOTA: Una mayor dispersión de los símbolos indica mayor nivel de ruido o peor calidad de la señal. Si aparece concentración de símbolos, es indicativo de una buena relación señal/ruido o ausencia de problemas como ruido de fase, etc.

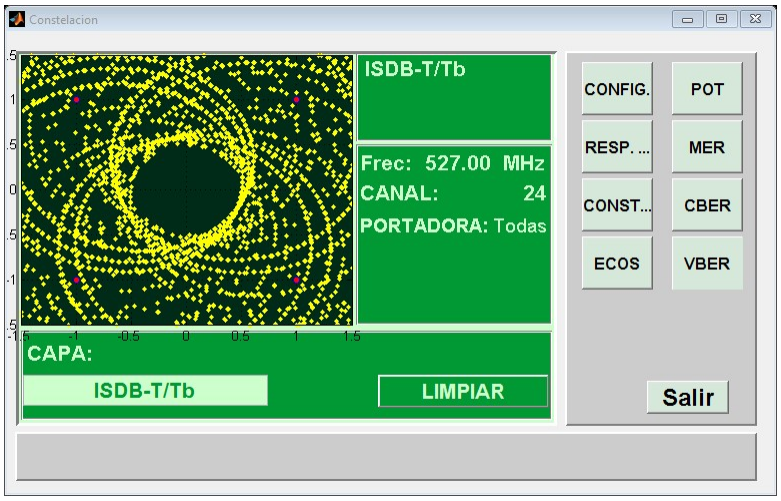

Figura A.6: Pantalla de representacion del Diagrama de Constelación para señales ISDB-T/TB

A.3.4. Analizador de Espectros

El modo Analizador de Espectros permite comprobar rápidamente las señales presentes en la banda de frecuencias y realizar medidas al mismo tiempo. Para seleccionarlo, basta pulsar la tecla **RESP**. En el monitor aparecerá una pantalla tal como se describe en la figura A.7.

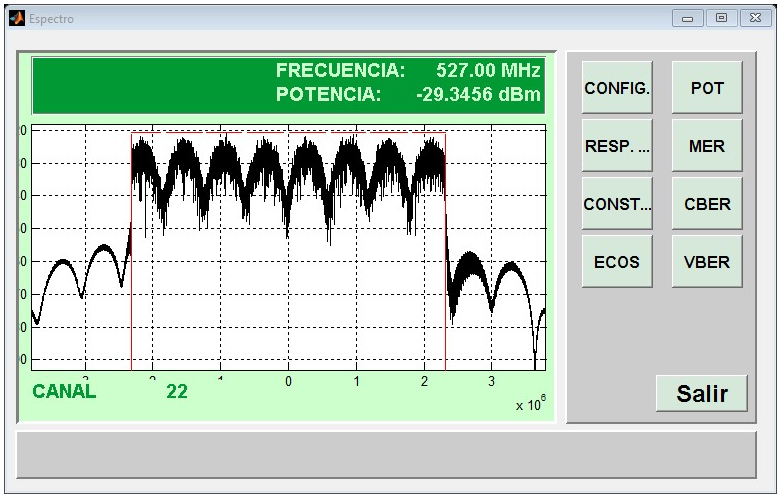

Figura A.7: Pantalla de medida del Espectro para señales ISDB-T/TB

A.3.5. Análisis de ECOS (ISDB-T/TB)

La utilidad de la función ECOS es la detección de los ecos que pueden aparecer debido a la recepción simultánea de la misma señal procedente de varios transmisores. Otra causa que puede provocar ecos es la reflexión de la señal sobre grandes objetos, como edificios o montañas.

Esta función es sólo aplicable a señales ISDB-T/TB con canal RICIAN. Por lo tanto, previamente se tendrá que configurar el aparato para la recepción de este tipo de señales. Si no se hace así, la medición de ECOS no es correcta.

Pulsar el botón **ECOS** para entrar. Aparecerá entonces la pantalla ECOS e iniciará la detección de los ecos.

La pantalla de la figura A.8, muestra una representación gráfica de los ecos. El eje horizontal de la representación gráfica se corresponde con el retraso en la recepción del eco respecto al camino principal (la señal con más potencia). En el eje vertical se representa la atenuación en dB del eco respecto al camino principal.

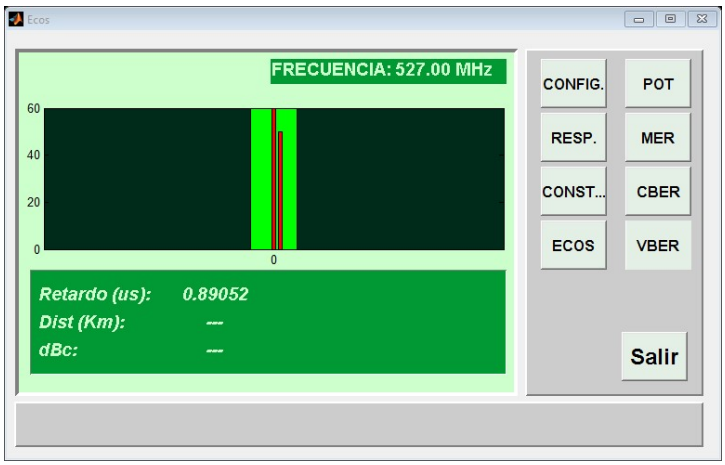

Figura A.8: Pantalla de medida de Ecos para señales ISDB-T/TB

El área junto a la señal principal es de un color diferente. Esta área representa el intervalo de guarda.

Apéndice II

CÓDIGOS REALIZADOS

B.1. medidas.m

```
1  function varargout = medidas(varargin)
2  gui_Singleton = 1;
3  gui_State = struct('gui_Name',       mfilename, ...
4                     'gui_Singleton',  gui_Singleton, ...
5                     'gui_OpeningFcn', @medidas_OpeningFcn, ...
6                     'gui_OutputFcn',  @medidas_OutputFcn, ...
7                     'gui_LayoutFcn',  [] , ...
8                     'gui_Callback',   []);
9  if nargin && ischar(varargin{1})
10     gui_State.gui_Callback = str2func(varargin{1});
11 end
12 if nargout
13     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
14 else
15     gui_mainfcn(gui_State, varargin{:});
16 end
17 function medidas_OpeningFcn(hObject, eventdata, handles, varargin)
18 set(handles.cn,'string','---'); set(handles.pot,'string','---');
19 set(handles.mer,'string','---'); set(handles.cber,'string','---');
20 set(handles.vber,'string','---'); set(handles.V1,'string','---');
21 set(handles.V4,'string','---'); axes(handles.axes1)
22 barh(1,60,3.3,'w'); axis image; set(handles.canal,'string','22');
23 set(handles.f,'string','521.00 MHz');
24 handles.output = hObject;
25 guidata(hObject, handles);
26 function varargout = medidas_OutputFcn(hObject, eventdata, handles)
27 varargout{1} = handles.output;
28 load status i model channel; load statuspc pc1; load freq f
29 switch channel
30     case 22
31         set(handles.f,'string','521.00 MHz');
32         df=f(4097)-521e+6;
33         set(handles.df,'string',[num2str(df) ' Hz']);
34     case 23
35         set(handles.f,'string','527.00 MHz');
36         df=f(4097)-527e+6;
37         set(handles.df,'string',[num2str(df) ' Hz']);
```

```
38      case 24
39          set(handles.f,'string','533.00 MHz');
40          df=f(4097)-533e+6;
41          set(handles.df,'string',[num2str(df) ' Hz']);
42      case 25
43          set(handles.f,'string','539.00 MHz');
44          df=f(4097)-539e+6;
45          set(handles.df,'string',[num2str(df) ' Hz']);
46  end
47  if i==0; Cargando; end
48  load status i model channel
49  switch i
50      case 1          potenc_Callback(hObject, eventdata, handles)
51      case 2          merpush_Callback(hObject, eventdata, handles)
52      case 3          cberpush_Callback(hObject, eventdata, handles)
53      case 4          vberpush_Callback(hObject, eventdata, handles)
54      otherwise
55          load variables
56          switch pc1
57      case 1    p=1/32; case 2    p=1/16
58      case 3    p=1/8;  case 4    p=1/4
59      case 5    p=1/2
60          end
61  load status i model channel
62        N=str2double(get_param([model '/IFFT1'], 'FFTLength'));
63        M=str2double(get_param([model '/Modulador QAM'],'M'));
64        Ts=str2double(get_param([model '/Random Integer Generator'], '
              Ts'));
65        Fs=1/((log2(M)*(Ts/8)*N)+(p*N*(Ts/8)*log2(M)))
66        save status i Fs model channel
67  [pot,potruido,CNR]=potencia(postcanal_periodograma,Fs);
68  set(handles.cn,'string',[num2str(CNR) ' dB']);
69  set(handles.pot,'string',[num2str(pot+30) ' dBm']);
70  MER=mer(tx,rx);
71  set(handles.mer,'string',[num2str(MER) ' dB']);
72  CBER=sum(xor(tx1,rx1))/length(tx1);
73  set(handles.cber,'string',num2str(CBER,'%8.3e'));
74  VBER=evalin('base','VBER(1)');
75  set(handles.vber,'string',num2str(VBER,'%8.3e'));
76  set(handles.V1,'string','C/N:');
77  set(handles.V4,'string',[num2str(CNR) ' dB']);
78  axes(handles.axes1)
79  barh(1,60,3.3,'w'); hold on
80  barh(1,CNR,3.3,'r'); axis image; hold off
81  end
82  function atras_Callback(hObject, eventdata, handles)
83  load status model Fs channel
84  i=0; save status i model Fs channel
85  close all
86  function config_Callback(hObject, eventdata, handles)
87  Configuracion;
88  function espectro_Callback(hObject, eventdata, handles)
```

```
esp=1; save espectro esp;
Espectro; close medidas;
function CONSTELACION_Callback(hObject, eventdata, handles)
Constelacion; close medidas;
function ecos_Callback(hObject, eventdata, handles)
Ecos; close medidas;
function potenc_Callback(hObject, eventdata, handles)
load variables; load status Fs
[pot,potruido,CNR]=potencia(postcanal_periodogram,Fs);
set(handles.V1,'string','POTENCIA:');
set(handles.text10,'string','>POTENCIA:');
set(handles.V4,'string',[num2str(pot+30) '␣dBm']);
axes(handles.axes1)
barh(1,-100,5,'r'); hold on; barh(1,20,5,'w'); hold on
barh(1,pot,5,'w'); axis image; hold off
set(handles.cn,'string',[num2str(CNR) '␣dB']);
set(handles.pot,'string',[num2str(pot+30) '␣dBm']);
MER=mer(tx,rx);
set(handles.mer,'string',[num2str(MER) '␣dB']);
CBER=sum(xor(tx1,rx1))/length(tx1);
set(handles.cber,'string',num2str(CBER,'%8.3e'));
VBER=evalin('base','VBER(1)');
set(handles.vber,'string',num2str(VBER,'%8.3e'));
set(handles.text5,'string','C/N:');
set(handles.text10,'string','>POTENCIA:');
set(handles.text11,'string','MER:');
set(handles.text12,'string','CBER:');
set(handles.text13,'string','VBER:');
function merpush_Callback(hObject, eventdata, handles)
load variables; load status Fs
MER=mer(tx,rx);
set(handles.V1,'string','MER:');
set(handles.V4,'string',[num2str(MER) '␣dB']);
axes(handles.axes1)
barh(1,40,2,'W'); hold on; barh(1,MER,2,'R'); axis image; hold off
[pot,potruido,CNR]=potencia(postcanal_periodogram,Fs);
set(handles.cn,'string',[num2str(CNR) '␣dB']);
set(handles.pot,'string',[num2str(pot+30) '␣dBm']);
set(handles.mer,'string',[num2str(MER) '␣dB']);
CBER=sum(xor(tx1,rx1))/length(tx1);
set(handles.cber,'string',num2str(CBER,'%8.3e'));
VBER=evalin('base','VBER(1)');
set(handles.vber,'string',num2str(VBER,'%8.3e'));
set(handles.text5,'string','C/N:');
set(handles.text10,'string','POTENCIA:');
set(handles.text11,'string','>MER:');
set(handles.text12,'string','CBER:');
set(handles.text13,'string','VBER:');
function cberpush_Callback(hObject, eventdata, handles)
load status Fs; load variables
CBER=sum(xor(tx1,rx1))/length(tx1);
set(handles.V1,'string','CBER:');
```

```
141 set(handles.V4,'string',num2str(CBER,'%8.3e'));
142       if CBER>1e-6 && CBER<=9.9e-6   C=-6;  end
143       if CBER>1e-5 & CBER<=9.9e-5    C=-5   end
144       if CBER>1e-4 & CBER<=9.9e-4    C=-4   end
145       if CBER>1e-3 & CBER<=9.9e-3    C=-3   end
146       if CBER>1e-2 & CBER<=9.9e-2    C=-2   end
147       if CBER>1e-1 & CBER<=9.9e-1    C=-1   end
148       if CBER==0e-0                  C=0    end
149 axes(handles.axes1)
150 barh(1,-6,0.3,'R'); hold on
151 barh(1,C,0.3,'W'); axis image; hold off
152 [pot,potruido,CNR]=potencia(postcanal_periodogram,Fs);
153 set(handles.cn,'string',[num2str(CNR) '␣dB']);
154 set(handles.pot,'string',[num2str(pot+30) '␣dBm']);
155 MER=mer(tx,rx);
156 set(handles.mer,'string',[num2str(MER) '␣dB']);
157 set(handles.cber,'string',num2str(CBER,'%8.3e'));
158 VBER=evalin('base','VBER(1)');
159 set(handles.vber,'string',num2str(VBER,'%8.3e'));
160 set(handles.text5,'string','C/N:');
161 set(handles.text10,'string','POTENCIA:');
162 set(handles.text11,'string','MER:');
163 set(handles.text12,'string','>CBER:');
164 set(handles.text13,'string','VBER:');
165 function vberpush_Callback(hObject, eventdata, handles)
166 load variables; load status
167 VBER=evalin('base','VBER(1)');
168 set(handles.V1,'string','VBER:');
169 set(handles.V4,'string',num2str(VBER,'%8.3e'));
170       if VBER>1e-6 & VBER<=9.9e-6   C=-6; end
171       if VBER>1e-5 & VBER<=9.9e-5   C=-5  end
172       if VBER>1e-4 & VBER<=9.9e-4   C=-4  end
173       if VBER>1e-3 & VBER<=9.9e-3   C=-3  end
174       if VBER>1e-2 & VBER<=9.9e-2   C=-2  end
175       if VBER>1e-1 & VBER<=9.9e-1   C=-1  end
176       if VBER==0e-0                 C=0   end
177 axes(handles.axes1)
178 barh(1,-6,0.3,'R'); hold on; barh(1,C,0.3,'W'); axis image; hold off
179 [pot,potruido,CNR]=potencia(postcanal_periodogram,Fs);
180 set(handles.cn,'string',[num2str(CNR) '␣dB']);
181 set(handles.pot,'string',[num2str(pot+30) '␣dBm']);
182 MER=mer(tx,rx);
183 set(handles.mer,'string',[num2str(MER) '␣dB']);
184 CBER=sum(xor(tx1,rx1))/length(tx1);
185 set(handles.cber,'string',num2str(CBER,'%8.3e'));
186 set(handles.vber,'string',num2str(VBER,'%8.3e'));
187 set(handles.text5,'string','C/N:');
188 set(handles.text10,'string','POTENCIA:');
189 set(handles.text11,'string','MER:');
190 set(handles.text12,'string','CBER:');
191 set(handles.text13,'string','>VBER:');
192 function slidercanal_Callback(hObject, eventdata, handles)
```

```
193 load status Fs model
194 canal=num2str(get(hObject,'Value')+22);
195 channel=round(str2double(canal));
196 set(handles.canal,'string',channel);
197 save status Fs channel model
198 switch channel
199     case 22
200         set(handles.f,'string','521.00 MHz');
201         f=(linspace(-1,1,8192)*2500*Fs)+521e+6 +(rand(1))*1e+1;
202         df=f(4097)-521e+6;
203         set(handles.df,'string',[num2str(df) ' Hz']);
204         save freq f
205     case 23
206         set(handles.f,'string','527.00 MHz');
207         f=(linspace(-1,1,8192)*2500*Fs)+527e+6 +(rand(1))*1e+1;
208         df=f(4097)-527e+6;
209         set(handles.df,'string',[num2str(df) ' Hz']);
210         save freq f
211     case 24
212         set(handles.f,'string','533.00 MHz');
213         f=(linspace(-1,1,8192)*2500*Fs)+533e+6 +(rand(1))*1e+1;
214         df=f(4097)-533e+6;
215         set(handles.df,'string',[num2str(df) ' Hz']);
216         save freq f
217          case 25
218         set(handles.f,'string','539.00 MHz');
219         f=(linspace(-1,1,8192)*2500*Fs)+539e+6 +(rand(1))*1e+1;
220         df=f(4097)-539e+6;
221         set(handles.df,'string',[num2str(df) ' Hz']);
222         save freq f
223 end
224 function mask_Callback(hObject, eventdata, handles)
225 esp=2; save espectro esp; Espectro; close medidas;
```

B.2. Configuracion.m

```
1  function varargout = Configuracion(varargin)
2  gui_Singleton = 1;
3  gui_State = struct('gui_Name',       mfilename, ...
4                     'gui_Singleton',  gui_Singleton, ...
5                     'gui_OpeningFcn', @Configuracion_OpeningFcn, ...
6                     'gui_OutputFcn',  @Configuracion_OutputFcn, ...
7                     'gui_LayoutFcn',  [] , ...
8                     'gui_Callback',   []);
9  if nargin && ischar(varargin{1})
10     gui_State.gui_Callback = str2func(varargin{1});
11 end
12 if nargout
13     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
14 else
15     gui_mainfcn(gui_State, varargin{:});
```

```
16 end
17 function Configuracion_OpeningFcn(hObject, eventdata, handles,
       varargin)
18 global model Fs
19 load status model i; load statuspc pc1
20 modo=str2double(get_param([model '/IFFT1'], 'FFTLength'));
21 switch modo
22     case 2048;  set(handles.modo, 'value',1);
23     case 4096;  set(handles.modo, 'value',2);
24     case 8192;  set(handles.modo, 'value',3);
25 end
26 mod=str2double(get_param([model '/Modulador_QAM'], 'M'));
27 switch mod
28     case 4;  set(handles.QAM, 'value',1);
29     case 16; set(handles.QAM, 'value',2);
30     case 64; set(handles.QAM, 'value',3);
31 end
32 fec=get_param([model '/Convolutional_Encoder'], 'punctureVector');
33 switch fec
34     case ''
35     case '[1;_1;_0;_1]'
36         set(handles.fec, 'value',2);
37     case '[1;_1;_0;_1;_1;_0]'
38         set(handles.fec, 'value',3);
39     case  ''
40     case '[1;_1;_0;_1;_0;_1;_0;_1;_1;_0;_0;_1;_1;_0]'
41         set(handles.fec, 'value',5);
42 end
43 switch model
44     case 'ISDBT_AWGN'
45         channel=1;
46         set(handles.canal, 'value',1);
47         K=str2double(get_param('ISDBT_AWGN/channel','SNRdB'));
48         set(handles.K,'string',['SNR_(dB):_________' num2str(K)]);
49     case 'QPSK'
50         channel=2;
51         set(handles.canal, 'value',2);
52         K=str2double(get_param('QPSK/channel','K'));
53         set(handles.K,'string',['FACTOR_K_(dB):_________' num2str(K)
               ]);
54 end
55         set(handles.pc,'value',pc1);
56 switch pc1
57     case 1;  p=1/32;     case 2;  p=1/16;
58     case 3;  p=1/8;      case 4;  p=1/4;
59     case 5;  p=1/2;
60 end
61       N=str2double(get_param([model '/IFFT1'], 'FFTLength'));
62       M=str2double(get_param([model '/Modulador_QAM'], 'M'));
63       Ts=str2double(get_param([model '/Random_Integer_Generator'], '
             Ts'));
64       Fs=1/((log2(M)*(Ts/8)*N)+(p*N*(Ts/8)*log2(M)))
```

```
65 save status modo mod fec channel model K pc1 Fs i
66 handles.output = hObject;
67 guidata(hObject, handles);
68 function varargout = Configuracion_OutputFcn(hObject, eventdata,
       handles)
69 varargout{1} = handles.output;
70 function modo_Callback(hObject, eventdata, handles)
71 global model; modo=get(handles.modo, 'value');
72 switch modo
73     case 1
74         N=num2str(2048);
75 set_param([model '/IFFT1'], 'FFTLength',N)
76 set_param([model '/FFT1'], 'FFTLength',N)
77 set_param([model '/OFDM␣frame␣structure␣Tx/Selector1'], 'Indices','
       [1:624␣1249:2048␣625:1248]')
78 set_param([model '/OFDM␣frame␣structure␣Rx/Selector'], 'Indices','
       [1:624␣1425:2048]')
79 set_param([model '/OFDM␣frame␣structure␣Tx/Pad'], 'numOutRows',N)
80 set_param([model '/OFDM␣frame␣structure␣Tx/Reshape'],'
       OutputDimensions','[1248,␣204]')
81 set_param([model '/OFDM␣frame␣structure␣Rx/Reshape1'], '
       OutputDimensions','[254592,1]')
82     case 2
83         N=num2str(4096);
84 set_param([model '/IFFT1'], 'FFTLength',N)
85 set_param([model '/FFT1'], 'FFTLength',N)
86 set_param([model '/OFDM␣frame␣structure␣Tx/Selector1'], 'Indices','
       [1:1248␣2497:4096␣1249:2496]')
87 set_param([model '/OFDM␣frame␣structure␣Rx/Selector'], 'Indices','
       [1:1248␣2849:4096]')
88 set_param([model '/OFDM␣frame␣structure␣Tx/Pad'], 'numOutRows',N)
89 set_param([model '/OFDM␣frame␣structure␣Tx/Reshape'],'
       OutputDimensions','[2496,␣204]')
90 set_param([model '/OFDM␣frame␣structure␣Rx/Reshape1'], '
       OutputDimensions','[509184,1]')
91     case 3
92         N=num2str(8192);
93 set_param([model '/IFFT1'], 'FFTLength',N)
94 set_param([model '/FFT1'], 'FFTLength',N)
95 set_param([model '/OFDM␣frame␣structure␣Tx/Selector1'], 'Indices','
       [1:2496␣4993:8192␣2497:4992]')
96 set_param([model '/OFDM␣frame␣structure␣Rx/Selector'], 'Indices','
       [1:2496␣5697:8192]')
97 set_param([model '/OFDM␣frame␣structure␣Tx/Pad'], 'numOutRows',N)
98 set_param([model '/OFDM␣frame␣structure␣Tx/Reshape'],'
       OutputDimensions','[4992,␣204]')
99 set_param([model '/OFDM␣frame␣structure␣Rx/Reshape1'], '
       OutputDimensions','[1018368,1]')
100 end
101 function modo_CreateFcn(hObject, eventdata, handles)
102 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
       defaultUicontrolBackgroundColor'))
```

```
103     set(hObject,'BackgroundColor','white');
104 end
105 function QAM_Callback(hObject, eventdata, handles)
106 global model; QAM = get(handles.QAM, 'value');
107 switch QAM
108     case 1
109 set_param([model '/Modulador_QAM'],'M','4');
110 set_param([model '/Demodulador_QAM'],'M','4');
111 set_param([model '/Random_Integer_Generator'],'mul','256');
112 set_param([model '/Random_Integer_Generator'], 'Ts','1396.16e-9')
113 set_param([model '/Integer_to_Bit_Converter'],'nbits','8');
114 set_param([model '/Bit_to_Integer_Converter'],'nbits','8');
115     case 2
116 set_param([model '/Modulador_QAM'],'M','16');
117 set_param([model '/Demodulador_QAM'],'M','16');
118 set_param([model '/Random_Integer_Generator'],'mul','256');
119 set_param([model '/Random_Integer_Generator'], 'Ts','3.44382264E-07')
120 set_param([model '/Integer_to_Bit_Converter'],'nbits','8');
121 set_param([model '/Bit_to_Integer_Converter'],'nbits','8');
122     case 3
123 set_param([model '/Random_Integer_Generator'], 'Ts','3.44382264E-07')
124 set_param([model '/Modulador_QAM'],'M','64')
125 set_param([model '/Demodulador_QAM'],'M','64')
126 set_param([model '/Random_Integer_Generator'],'mul','256')
127 set_param([model '/Integer_to_Bit_Converter'],'nbits','8');
128 set_param([model '/Bit_to_Integer_Converter'],'nbits','8');
129 end
130 function fec_Callback(hObject, eventdata, handles)
131 global model; fec=get(handles.fec,'value');
132 QAM=str2num(get_param([model '/Modulador_QAM'],'M'));
133 modo=str2num(get_param([model '/IFFT1'], 'FFTLength'));
134 switch fec
135     case 1
136         switch QAM
137             case 4
138                 if modo==2048
139 set_param([model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','254592')
140 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','254592')
141 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
    156*188')
142                 elseif modo==4096
143 set_param([model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','509184')
144 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','509184')
145 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
    312*188')
146                 else
147 set_param([ model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','1018368'
    )
148 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','1018368')
149 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
    624*188')
150                 end
```

```
151             case 16
152                 if modo==2048
153 set_param([model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','509184')
154 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','509184')
155 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
    312*188')
156                 elseif modo==4096
157 set_param([model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','1018368')
158 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','1018368')
159 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
    624*188')
160                 else
161 set_param([model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','2036736')
162 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','2036736')
163 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
    1248*188')
164                 end
165             case 64
166                 if modo==2048
167 set_param([model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','763776')
168 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','763776')
169 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
    468*188')
170                 elseif modo==4096
171 set_param([model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','1527552')
172 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','1527552')
173 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
    936*188')
174                 else
175 set_param([model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','3055104')
176 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','3055104')
177 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
    1872*188')
178                 end
179         end
180      case 2
181         switch QAM
182             case 4
183                 if modo==2048
184 set_param([model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','339456')
185 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','339456')
186 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
    208*188')
187                 elseif modo==4096
188 set_param([model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','678912')
189 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','678912')
190 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
    2*208*188')
191                 else
192 set_param([model '/Energy_Dispersal_Tx/Frame_Buffer'], 'N','1357824')
193 set_param([model '/Energy_Dispersal_Rx/Frame_Buffer'], 'N','1357824')
194 set_param([model '/Random_Integer_Generator'], 'sampPerFrame','
```

```
    4*208*188')
195                 end
196             case 16
197                 if modo==2048
198 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','678912')
199 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','678912')
200 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    416*188')
201                 elseif modo==4096
202 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','1357824')
203 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','1357824')
204 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    832*188')
205                 else
206 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','2715648')
207 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','2715648')
208 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    1664*188')
209                 end
210             case 64
211                 if modo==2048
212 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','1018368')
213 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','1018368')
214 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    624*188')
215                 elseif modo==4096
216  set_param([ model '/Energy Dispersal Tx/Frame Buffer'], 'N','2036736
    ')
217 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','2036736')
218 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    1248*188')
219                 else
220 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','4073472')
221 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','4073472')
222 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    2496*188')
223                 end
224         end
225 set_param([model '/Convolutional Encoder'], 'punctureVector','[1; 1;
    0; 1]')
226 set_param([model '/Viterbi Decoder'],'punctureVector','[1; 1; 0; 1]')
227      case 3
228           switch QAM
229               case 4
230                   if modo==2048
231 set_param([ model '/Energy Dispersal Tx/Frame Buffer'], 'N','381888')
232 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','381888')
233 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    234*188')
234                   elseif modo==4096
235 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','763776')
236 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','763776')
```

```
237 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    468*188')
238                 else
239 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','1527552')
240 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','1527552')
241 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    936*188')
242                 end
243             case 16
244                 if modo==2048
245 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','763776')
246 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','763776')
247 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    468*188')
248                 elseif modo==4096
249 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','1527552')
250 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','1527552')
251 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    2*468*188')
252                 else
253 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','3055104')
254 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','3055104')
255 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    4*468*188')
256                 end
257             case 64
258                 if modo==2048
259 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','1145664')
260 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','1145664')
261 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    702*188')
262                 elseif modo==4096
263 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','2291328')
264 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','2291328')
265 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    1404*188')
266                 else
267 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','4582656')
268 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','4582656')
269 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    2808*188')
270                 end
271         end
272 set_param([model '/Viterbi Decoder'],'punctureVector','[1; 1; 0; 1;
    1; 0]')
273 set_param([model '/Convolutional Encoder'], 'punctureVector','[1; 1;
    0; 1; 1; 0]')
274     case 4
275         switch QAM
276             case 4
277                 if modo==2048
278 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','424320')
```

```
279 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','424320')
280 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    260*188')
281                 elseif modo==4096
282 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','848640')
283 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','848640')
284 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    520*188')
285                 else
286 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','1697280')
287 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','1697280')
288 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    1040*188')
289                 end
290             case 16
291                 if modo==2048
292 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','848640')
293 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','848640')
294 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    520*188')
295                 elseif modo==4096
296 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','1697280')
297 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','1697280')
298 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    1040*188')
299                 else
300 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','3394560')
301 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','3394560')
302 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    2080*188')
303                 end
304             case 64
305                 if modo==2048
306 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','1272960')
307 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','1272960')
308 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    780*188')
309                 elseif modo==4096
310 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','2545920')
311 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','2545920')
312 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    1560*188')
313                 else
314 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','5091840')
315 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','5091840')
316 set_param([model '/Random Integer Generator'], 'sampPerFrame','
    3120*188')
317                 end
318         end
319      case 5
320         switch QAM
321             case 4
```

```
322                 if modo==2048
323 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','445536')
324 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','445536')
325 set_param([model '/Random Integer Generator'], 'sampPerFrame','
        273*188')
326                 elseif modo==4096
327 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','891072')
328 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','891072')
329 set_param([model '/Random Integer Generator'], 'sampPerFrame','
        546*188')
330                 else
331 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','1782144')
332 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','1782144')
333 set_param([model '/Random Integer Generator'], 'sampPerFrame','
        1092*188')
334                 end
335             case 16
336                 if modo==2048
337 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','891072')
338 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','891072')
339 set_param([model '/Random Integer Generator'], 'sampPerFrame','
        546*188')
340                 elseif modo==4096
341 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','1782144')
342 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','1782144')
343 set_param([model '/Random Integer Generator'], 'sampPerFrame','
        1092*188')
344                 else
345 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','3564288')
346 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','3564288')
347 set_param([model '/Random Integer Generator'], 'sampPerFrame','
        2184*188')
348                 end
349             case 64
350                 if modo==2048
351 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','1336608')
352 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','1336608')
353 set_param([model '/Random Integer Generator'], 'sampPerFrame','
        819*188')
354                 elseif modo==4096
355 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','2673216')
356 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','2673216')
357 set_param([model '/Random Integer Generator'], 'sampPerFrame','
        2*819*188')
358                 else
359 set_param([model '/Energy Dispersal Tx/Frame Buffer'], 'N','5346432')
360 set_param([model '/Energy Dispersal Rx/Frame Buffer'], 'N','5346432')
361 set_param([model '/Random Integer Generator'], 'sampPerFrame','
        4*819*188')
362                 end
363         end
364 set_param([model '/Viterbi Decoder'],'punctureVector','[1; 1; 0; 1;
```

```
    0;␣1;␣0;␣1;␣1;␣0;␣0;␣1;␣1;␣0]')
365 set_param([model '/Convolutional␣Encoder'], 'punctureVector','[1;␣1;␣
    0;␣1;␣0;␣1;␣0;␣1;␣1;␣0;␣0;␣1;␣1;␣0]')
366 end
367 function pc_Callback(hObject, eventdata, handles)
368 global model pc; pc=get(handles.pc,'value');
369 modo=str2num(get_param([model '/IFFT1'], 'FFTLength'));
370 switch pc
371     case 1
372         if modo==2048
373 set_param([model '/adicion␣PC'], 'Indices','[1985:2048␣1:2048],')
374 set_param([model '/sustraccion␣PC'], 'Indices','[65:2112],')
375 set_param([model '/Reshape2'], 'OutputDimensions','[2112*204,1]')
376 set_param([model '/Reshape3'], 'OutputDimensions','[2112,204]')
377         elseif modo==4096
378 set_param([model '/adicion␣PC'], 'Indices','[3969:4096␣1:4096],')
379 set_param([model '/sustraccion␣PC'], 'Indices','[129:4224],')
380 set_param([model '/Reshape2'], 'OutputDimensions','[4224*204,1]')
381 set_param([model '/Reshape3'], 'OutputDimensions','[4224,204]')
382         else
383 set_param([model '/adicion␣PC'], 'Indices','[7937:8192␣1:8192],')
384 set_param([model '/sustraccion␣PC'], 'Indices','[257:8448],')
385 set_param([model '/Reshape2'], 'OutputDimensions','[8448*204,1]')
386 set_param([model '/Reshape3'], 'OutputDimensions','[8448,204]')
387         end
388     case 2
389         if modo==2048
390 set_param([model '/adicion␣PC'], 'Indices','[1921:2048␣1:2048],')
391 set_param([model '/sustraccion␣PC'], 'Indices','[129:2176],')
392 set_param([model '/Reshape2'], 'OutputDimensions','[2176*204,1]')
393 set_param([model '/Reshape3'], 'OutputDimensions','[2176,204]')
394         elseif modo==4096
395 set_param([model '/adicion␣PC'], 'Indices','[3841:4096␣1:4096],')
396 set_param([model '/sustraccion␣PC'], 'Indices','[257:4352],')
397 set_param([model '/Reshape2'], 'OutputDimensions','[4352*204,1]')
398 set_param([model '/Reshape3'], 'OutputDimensions','[4352,204]')
399         else
400 set_param([model '/adicion␣PC'], 'Indices','[7681:8192␣1:8192],')
401 set_param([model '/sustraccion␣PC'], 'Indices','[513:8704],')
402 set_param([model '/Reshape2'], 'OutputDimensions','[8704*204,1]')
403 set_param([model '/Reshape3'], 'OutputDimensions','[8704,204]')
404         end
405     case 3
406         if modo==2048
407 set_param([model '/adicion␣PC'], 'Indices','[1793:2048␣1:2048],')
408 set_param([model '/sustraccion␣PC'], 'Indices','[257:2304],')
409 set_param([model '/Reshape2'], 'OutputDimensions','[2304*204,1]')
410 set_param([model '/Reshape3'], 'OutputDimensions','[2304,204]')
411         elseif modo==4096
412 set_param([model '/adicion␣PC'], 'Indices','[3585:4096␣1:4096],')
413 set_param([model '/sustraccion␣PC'], 'Indices','[513:4608],')
414 set_param([model '/Reshape2'], 'OutputDimensions','[4608*204,1]')
```

```
415 set_param([model '/Reshape3'], 'OutputDimensions','[4608,204]')
416         else
417 set_param([model '/adicion␣PC'], 'Indices','[7169:8192␣1:8192],')
418 set_param([model '/sustraccion␣PC'], 'Indices','[1025:9216],')
419 set_param([model '/Reshape2'], 'OutputDimensions','[9216*204,1]')
420 set_param([model '/Reshape3'], 'OutputDimensions','[9216,204]')
421         end
422     case 4
423         if modo==2048
424 set_param([model '/adicion␣PC'], 'Indices','[1537:2048␣1:2048],')
425 set_param([model '/sustraccion␣PC'], 'Indices','[513:2560],')
426 set_param([model '/Reshape2'], 'OutputDimensions','[2560*204,1]')
427 set_param([model '/Reshape3'], 'OutputDimensions','[2560,204]')
428         elseif modo==4096
429 set_param([model '/adicion␣PC'], 'Indices','[3073:4096␣1:4096],')
430 set_param([model '/sustraccion␣PC'], 'Indices','[1025:5120],')
431 set_param([model '/Reshape2'], 'OutputDimensions','[5120*204,1]')
432 set_param([model '/Reshape3'], 'OutputDimensions','[5120,204]')
433         else
434 set_param([model '/adicion␣PC'], 'Indices','[6145:8192␣1:8192],')
435 set_param([model '/sustraccion␣PC'], 'Indices','[2049:10240],')
436 set_param([model '/Reshape2'], 'OutputDimensions','[10240*204,1]')
437 set_param([model '/Reshape3'], 'OutputDimensions','[10240,204]')
438         end
439     case 5
440        if modo==2048
441 set_param([model '/adicion␣PC'], 'Indices','[1025:2048␣1:2048],')
442 set_param([model '/sustraccion␣PC'], 'Indices','[1025:3072],')
443 set_param([model '/Reshape2'], 'OutputDimensions','[3072*204,1]')
444 set_param([model '/Reshape3'], 'OutputDimensions','[3072,204]')
445         elseif modo==4096
446 set_param([model '/adicion␣PC'], 'Indices','[2049:4096␣1:4096],')
447 set_param([model '/sustraccion␣PC'], 'Indices','[2049:6144],')
448 set_param([model '/Reshape2'], 'OutputDimensions','[6144*204,1]')
449 set_param([model '/Reshape3'], 'OutputDimensions','[6144,204]')
450         else
451 set_param([model '/adicion␣PC'], 'Indices','[4097:8192␣1:8192],')
452 set_param([model '/sustraccion␣PC'], 'Indices','[4097:12288],')
453 set_param([model '/Reshape2'], 'OutputDimensions','[12288*204,1]')
454 set_param([model '/Reshape3'], 'OutputDimensions','[12288,204]')
455         end
456 end
457 save statuspc pc
458 function pc_CreateFcn(hObject, eventdata, handles)
459 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
460     set(hObject,'BackgroundColor','white');
461 end
462 function salir_Callback(hObject, eventdata, handles)
463 global model pc
464 modo2=str2double(get_param([model '/IFFT1'], 'FFTLength'));
465 mod2=str2double(get_param([model '/Modulador␣QAM'],'M'));
```

```
466 fec2=get_param([model '/Convolutional_Encoder'], 'punctureVector');
467 switch model
468     case 'QPSK'
469         K1=str2double(get_param('QPSK/channel','K'));
470     case 'ISDBT_AWGN'
471         K1=str2double(get_param('ISDBT_AWGN/channel','SNRdB'));
472 end
473 canal=get(handles.canal, 'value');
474 load status modo mod fec channel K pc1 i
475 load statuspc pc
476 if modo ~= modo2 | mod ~= mod2 | length(fec) ~= length(fec2) |
        channel ~= canal | K ~= K1 | pc ~= pc1
477     pc1=get(handles.pc,'value');
478     save statuspc pc1
479     close Configuracion;
480     Cargando; medidas;
481 else
482     pc1=get(handles.pc,'value');
483     save statuspc pc1;   close Configuracion
484 end
485 function salir_CreateFcn(hObject, eventdata, handles)
486 function canal_Callback(hObject, eventdata, handles)
487 canal=get(handles.canal,'value');
488 global model
489 switch canal
490     case 1
491 load status modo mod fec channel K pc1 i Fs
492 model='ISDBT_AWGN';
493 load_system(model);
494 save status modo mod fec channel model K pc1 i Fs
495 K2=str2double(get_param('ISDBT_AWGN/channel','SNRdB'));
496 set(handles.K,'string',['SNR_(dB):_________' num2str(K2)]);
497     case 2
498 load status modo mod fec channel K pc1 i Fs
499 model='QPSK';
500 load_system(model);
501 save status modo mod fec channel model K pc1 i Fs
502 K2=str2double(get_param('QPSK/channel','K'));
503 set(handles.K,'string',['FACTOR_K_(dB):_________' num2str(K2)]);
504 end
505 function canal_CreateFcn(hObject, eventdata, handles)
506 global model
507 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
508     set(hObject,'BackgroundColor','white');
509 end
510 function sliderK_Callback(hObject, eventdata, handles)
511 load status model
512 switch model
513     case 'QPSK'
514 K=num2str(get(hObject,'Value'));
515 set(handles.K,'string',['FACTOR_K_(dB):_________' num2str(K)]);
```

```
516 set_param('QPSK/channel','K',K);
517      case 'ISDBT_AWGN'
518 SNR=num2str(get(hObject,'Value'));
519 set(handles.K,'string',['SNR (dB): ' num2str(SNR)]);
520 set_param('ISDBT_AWGN/channel','SNRdB',SNR);
521 end
522 function sliderK_CreateFcn(hObject, eventdata, handles)
523 if isequal(get(hObject,'BackgroundColor'), get(0,'
       defaultUicontrolBackgroundColor'))
524      set(hObject,'BackgroundColor',[.9 .9 .9]);
525 end
526 function K_Callback(hObject, eventdata, handles)
527 function K_CreateFcn(hObject, eventdata, handles)
528 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
       defaultUicontrolBackgroundColor'))
529      set(hObject,'BackgroundColor','white');
530 end
531 function edit18_Callback(hObject, eventdata, handles)
532 function edit18_CreateFcn(hObject, eventdata, handles)
533 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
       defaultUicontrolBackgroundColor'))
534      set(hObject,'BackgroundColor','white');
535 end
536 function slider2_Callback(hObject, eventdata, handles)
537 load status model
538 switch model
539      case 'QPSK'
540 K=get(hObject,'Value');
541 %set(handles.K,'string',['FACTOR K (dB):          ' num2str(K)]);
542 set_param('QPSK/Amplifier', 'linGaindB',num2str(-K));
543      case 'ISDBT_AWGN'
544 K=get(hObject,'Value');
545 %set(handles.K,'string',['SNR (dB):          ' num2str(SNR)]);
546 set_param('ISDBT_AWGN/Amplifier','linGaindB',num2str(-K));
547 end
548 function slider2_CreateFcn(hObject, eventdata, handles)
549 if isequal(get(hObject,'BackgroundColor'), get(0,'
       defaultUicontrolBackgroundColor'))
550      set(hObject,'BackgroundColor',[.9 .9 .9]);
551 end
```

B.3. Ecos.m

```
1 function varargout = Ecos(varargin)
2 gui_Singleton = 1;
3 gui_State = struct('gui_Name',       mfilename, ...
4                    'gui_Singleton',  gui_Singleton, ...
5                    'gui_OpeningFcn', @Ecos_OpeningFcn, ...
6                    'gui_OutputFcn',  @Ecos_OutputFcn, ...
7                    'gui_LayoutFcn',  [] , ...
8                    'gui_Callback',   []);
```

```
 9 if nargin && ischar(varargin{1})
10     gui_State.gui_Callback = str2func(varargin{1});
11 end
12 if nargout
13     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
14 else
15     gui_mainfcn(gui_State, varargin{:});
16 end
17 function Ecos_OpeningFcn(hObject, eventdata, handles, varargin)
18 load variables postcanal_periodogram; load status
19 axes(handles.axes1)
20 [DL,retardo1]=retardo(postcanal_periodogram,Fs)
21 bar(0,60,6,'g'); xlim([-30 30]); hold on
22 bar(0,60,0.5,'r'); hold on
23 bar(retardo1*1e+6,50,0.5,'r');
24 set(gca,'Color',[0 0.17 0.1]); hold off
25 set(handles.retardo,'string',num2str(retardo1*1e+6));
26 handles.output = hObject;
27 guidata(hObject, handles);
28 function varargout = Ecos_OutputFcn(hObject, eventdata, handles)
29 varargout{1} = handles.output;
30 function config_CreateFcn(hObject, eventdata, handles)
31 function config_Callback(hObject, eventdata, handles)
32 Configuracion;
33 function espectro_Callback(hObject, eventdata, handles)
34 esp=1; save espectro esp;
35 Espectro; close Ecos
36 function constelacion_Callback(hObject, eventdata, handles)
37 Constelacion; close Ecos
38 function ecos_Callback(hObject, eventdata, handles)
39 Ecos;
40 function atras_Callback(hObject, eventdata, handles)
41 load status model Fs channel
42 i=0; save status i model Fs channel; close all
43 function potenc_Callback(hObject, eventdata, handles)
44 load status model Fs channel
45 i=1; save status i model Fs channel
46 medidas; close Ecos
47 function merpush_Callback(hObject, eventdata, handles)
48 load status model Fs channel
49 i=2; save status i model Fs channel
50 medidas; close Ecos
51 function cberpush_Callback(hObject, eventdata, handles)
52 load status model Fs channel
53 i=3; save status i model Fs channel
54 medidas; close Ecos
55 function vberpush_Callback(hObject, eventdata, handles)
56 load status model Fs channel
57 i=4; save status i model Fs channel
58 medidas; close Ecos
59 function mask_Callback(hObject, eventdata, handles)
60 esp=2; save espectro esp;
```

```
61 Espectro; close Ecos
```

B.4. Espectro.m

```
1  function varargout = Espectro(varargin)
2  gui_Singleton = 1;
3  gui_State = struct('gui_Name',       mfilename, ...
4                     'gui_Singleton',  gui_Singleton, ...
5                     'gui_OpeningFcn', @Espectro_OpeningFcn, ...
6                     'gui_OutputFcn',  @Espectro_OutputFcn, ...
7                     'gui_LayoutFcn',  [] , ...
8                     'gui_Callback',   []);
9  if nargin && ischar(varargin{1})
10     gui_State.gui_Callback = str2func(varargin{1});
11 end
12 if nargout
13     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
14 else
15     gui_mainfcn(gui_State, varargin{:});
16 end
17 function Espectro_OpeningFcn(hObject, eventdata, handles, varargin)
18 load variables postcanal_periodogram
19 load status Fs channel i
20 set(handles.canal, 'string', num2str(channel));
21 switch channel
22     case 22;  set(handles.frec, 'string','521 MHz');
23     case 23;  set(handles.frec, 'string','527 MHz');
24     case 24;  set(handles.frec, 'string','533 MHz');
25     case 25;  set(handles.frec, 'string','539 MHz');
26 end
27 [pot,potr,CNR]=potencia(postcanal_periodogram,Fs)
28 set(handles.pot, 'string',[num2str(pot+30) ' dBm']);
29 load espectro esp
30 switch esp
31     case 1
32         axes(handles.axes1)
33         cla reset
34         AB(postcanal_periodogram,Fs,channel)
35     case 2
36         axes(handles.axes1)
37         cla reset
38         respuesta(postcanal_periodogram,Fs,channel)
39 end
40 handles.output = hObject;
41 guidata(hObject, handles);
42 function varargout = Espectro_OutputFcn(hObject, eventdata, handles)
43 varargout{1} = handles.output;
44 function atras_Callback(hObject, eventdata, handles)
45 load status model Fs channel
46 i=0; save status i model Fs channel
47 close all
```

```
48 function config_Callback(hObject, eventdata, handles)
49 Configuracion;
50 function espectro_Callback(hObject, eventdata, handles)
51 esp=1; save espectro esp;
52 Espectro;
53 function constelacion_Callback(hObject, eventdata, handles)
54 Constelacion; close Espectro;
55 function ecos_Callback(hObject, eventdata, handles)
56 Ecos; close Espectro;
57 function potenc_Callback(hObject, eventdata, handles)
58 load status model Fs channel channel
59 i=1; save status i model Fs channel channel
60 medidas; close Espectro;
61 function merpush_Callback(hObject, eventdata, handles)
62 load status model Fs channel
63 i=2; save status i model Fs channel
64 medidas; close Espectro
65 function cberpush_Callback(hObject, eventdata, handles)
66 load status model Fs channel
67 i=3; save status i model Fs channel
68 medidas; close Espectro
69 function vberpush_Callback(hObject, eventdata, handles)
70 load status model Fs channel
71 i=4; save status i model Fs channel
72 medidas; close Espectro
73 function mask_Callback(hObject, eventdata, handles)
74 esp=2; save espectro esp;
75 Espectro;
```

B.5. Constelacion.m

```
1  function varargout = Constelacion(varargin)
2  gui_Singleton = 1;
3  gui_State = struct('gui_Name',       mfilename, ...
4                     'gui_Singleton',  gui_Singleton, ...
5                     'gui_OpeningFcn', @Constelacion_OpeningFcn, ...
6                     'gui_OutputFcn',  @Constelacion_OutputFcn, ...
7                     'gui_LayoutFcn',  [] , ...
8                     'gui_Callback',   []);
9  if nargin && ischar(varargin{1})
10     gui_State.gui_Callback = str2func(varargin{1});
11 end
12 if nargout
13     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
14 else
15     gui_mainfcn(gui_State, varargin{:});
16 end
17 function Constelacion_OpeningFcn(hObject, eventdata, handles,
       varargin)
18 load variables tx rx
19 axes(handles.axes1)
```

```
20 plot(real(rx(1:8192)), imag(rx(1:8192)),'.y',real(tx(1:8192)), imag(
       tx(1:8192)),'o','MarkerFaceColor','r','MarkerSize',5)
21 xlim([min(real(tx(1:8192)))-0.5 max(real(tx(1:8192)))+0.5])
22 ylim([min(imag(tx(1:8192)))-0.5 max(imag(tx(1:8192)))+0.5])
23 set(gca,'Color',[0 0.17 0.1]);
24 grid on
25 handles.output = hObject;
26 guidata(hObject, handles);
27 function varargout = Constelacion_OutputFcn(hObject, eventdata,
       handles)
28 varargout{1} = handles.output;
29 function edit2_Callback(hObject, eventdata, handles)
30 function edit2_CreateFcn(hObject, eventdata, handles)
31 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
       defaultUicontrolBackgroundColor'))
32     set(hObject,'BackgroundColor','white');
33 end
34 function limpiar_Callback(hObject, eventdata, handles)
35 load variables tx rx
36 axes(handles.axes1)
37 for i=1:8192
38 plot(real(rx(1:i)), imag(rx(1:i)),'.y',real(tx(1:i)), imag(tx(1:i)),'
       o','MarkerFaceColor','r','MarkerSize',5)
39 xlim([min(real(tx(1:8192)))-0.5 max(real(tx(1:8192)))+0.5])
40 ylim([min(imag(tx(1:8192)))-0.5 max(imag(tx(1:8192)))+0.5])
41 set(gca,'Color',[0 0.17 0.1]); grid on
42 end
43 function atras_Callback(hObject, eventdata, handles)
44 load status model Fs channel
45 i=0; save status i model Fs channel
46 close all;
47 function config_Callback(hObject, eventdata, handles)
48 Configuracion;
49 function espectro_Callback(hObject, eventdata, handles)
50 esp=1; save espectro esp;
51 Espectro; close Constelacion;
52 function constelacion_Callback(hObject, eventdata, handles)
53 Constelacion;
54 function ecos_Callback(hObject, eventdata, handles)
55 Ecos; close Constelacion;
56 function potenc_Callback(hObject, eventdata, handles)
57 load status model Fs channel
58 i=1; save status i model Fs channel
59 medidas; close Constelacion;
60 function merpush_Callback(hObject, eventdata, handles)
61 load status model Fs channel
62 i=2; save status i model Fs channel
63 medidas; close Constelacion;
64 function cberpush_Callback(hObject, eventdata, handles)
65 load status model Fs channel
66 i=3; save status i model Fs channel
67 medidas; close Constelacion;
```

```
68 function vberpush_Callback(hObject, eventdata, handles)
69 load status model Fs channel
70 i=4; save status i model Fs channel
71 medidas; close Constelacion;
72 function mask_Callback(hObject, eventdata, handles)
73 esp=2; save espectro esp;
74 Espectro; close Constelacion
```

B.6. Cargando.m

```
 1 function varargout = Cargando(varargin)
 2 gui_Singleton = 1;
 3 gui_State = struct('gui_Name',       mfilename, ...
 4                    'gui_Singleton',  gui_Singleton, ...
 5                    'gui_OpeningFcn', @Cargando_OpeningFcn, ...
 6                    'gui_OutputFcn',  @Cargando_OutputFcn, ...
 7                    'gui_LayoutFcn',  [] , ...
 8                    'gui_Callback',   []);
 9 if nargin && ischar(varargin{1})
10     gui_State.gui_Callback = str2func(varargin{1});
11 end
12 if nargout
13     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
14 else
15     gui_mainfcn(gui_State, varargin{:});
16 end
17 function Cargando_OpeningFcn(hObject, eventdata, handles, varargin)
18 handles.output = hObject;
19 guidata(hObject, handles);
20 function varargout = Cargando_OutputFcn(hObject, eventdata, handles)
21 varargout{1} = handles.output;
22 try
23 load status model i
24 load_system(model);
25 switch model
26         case 'QPSK'
27             if i==0
28 r=(rand(1)+rand(1))*2e-6
29 set_param('QPSK/channel', 'pathDelays',['[0 ' num2str(r) ']'])
30             end
31     otherwise
32 end
33 sim(model)
34 catch
35 end
36 save variables tx rx tx1 rx1 postcanal_periodogram
37 close Cargando
```