Práctica 4: Modulación digital.

1. Objetivo y contenido

En esta práctiva veremos cómo simular sistemas de modulación digital con matlab. En primer lugar se verán ejemplos sobre simulación de procesos de cuantización, tanto uniforme como no uniforme. Después se presentarán varios ejemplos sobre esquemas de modulación digital, incluyendo los básicos, ASK, PSK y FSK y los M-arios. Por último, veremos algunos modelos Simulink de sistemas de modulación digital.

2. Cuantización

En el ejemplo 1 se cuantiza un fragmento de señal sinusoide, definiéndose 11 valores (vector `particion`, que define una partición del conjunto de números reales, o dicho de otro modo, dónde se colocan los "escalones" para el proceso de cuantización). Estos 11 escalares definen 12 intervalos de valores. Para cuantizar de manera uniforme una señal, basta con pasar como parámetros la señal analógica original y la partición a la función `quantiz`.

Ejemplo 1 ficom_pr4_0001_cuantizacion_simple.m

```
% Ejemplo simple de cuantización
% Instantes de muestreo
t=[0:0.1:2*pi];
% Señal
m=sin(t)
% Partición
particion = [-1:.2:1];
% Cuantización
intervalos = quantiz(m,particion);
% Representación
plot(t,m)
hold on
plot(t,m,'x')
stem(t,particion(intervalos),'.r');
hold off
```

En el ejemplo 2 se utiliza, además de una partición, un vector para los valores de salida del cuantizador. Cada elemento del vector `particion` lleva asociado un elemento del vector `codebook`, que especifica el valor de salida del cuantizador que se debe asignar a cada rango de la partición. En este caso, se modifican ligeramente los valores con objeto de reducir la distorsión introducida por el proceso de cuantización, de modo que la señal cuantizada sea más "parecida" a la original[1].

Ejemplo 2 ficom_pr4_0002_cuantizacion_codebook.m

```
% Cuantización con partición y codebook
% Instantes de muestreo
t=[0:0.1:2*pi];
% Señal
m=sin(t)
% Partición
particion = [-1:.2:1];
codebook = [0 -1 -0.7 -0.5 -0.3 -0.1 0.1 0.3 0.5 0.7 1]
% Cuantización
[intervalos,valores] = quantiz(m,particion,codebook);
% Representación
plot(t,m)
hold on
plot(t,m,'x')
stem(t,valores,'.r');
hold off
```

Otra manera de evitar que el proceso de cuantización lleve consigo la pérdida de información de interés es emplear compresores y expansores. Para simular estos sistemas, en matlab existe la función predefinida `compand`, que permite simular compresores y expansores de ley μ y de ley A. En el ejemplo 3 se muestra el efecto de un compresor de ley $\mu = 255$:

3. Modulación digital en paso de banda

Los siguientes ejemplos sobre sistemas de modulación digital empezarán definiendo señales ya cuantizadas, es decir, supondremos que el proceso de cuantización se ha realizado ya siguiendo algún procedimiento similar al visto en los ejemplos anteriores. El proceso de muestreo vendrá definido del siguiente modo: el vector que representa la señal digital tendrá tantos elementos como muestras se hayan tomado en el intervalo de tiempo que representa el vector.

Para simular moduladores y demoduladores, se dispone de la función `dmod`[2]. En el

[1] Este es un caso muy simple de mejora del proceso de cuantización, véase la nota sobre cuantización para más detalles sobre otros procedimientos más formales y precisos

[2] Ojo, esta función empieza a quedar obsoleta en las últimas versiones de Matlab. Ver la tercera nota, al final.

Ejemplo 3 ficom_pr4_0003_compansor.m

```
% Ejemplo de compresores/expansores de ley mu y A
t=0:0.1:2*pi;
% señal y=x
m=t;
% señal sinusoide
m2=sin(t);
% Expandir señal m
m_comp_mu=compand(m,255,max(m),'mu/compressor');
m_comp_a=compand(m,255,max(m),'a/compressor');
% Expandir señal a
m2_comp_mu=compand(m2,255,max(m2),'mu/compressor');
m2_comp_a=compand(m2,255,max(m2),'a/compressor');
% Representar:
subplot(211)
plot(t,m)
hold on
plot(t,m_comp_mu,'Color','red')
plot(t,m_comp_a,'Color','black')
subplot(212)
plot(t,m2)
hold on
plot(t,m2_comp_mu,'Color','red')
plot(t,m2_comp_a,'Color','black')
```

ejemplo 4 se utiliza para simular sistemas 4-ASK. En este primer ejemplo utilizamos también la función `randint` para generar un vector de números enteros aleatorios:

En el ejemplo, la representación gráfica se hace mediante líneas; para ver mejor los símbolos alterados puede ser conveniente utilizar representación gráfica por puntos (usando `plot(...,...,'.')`).

Utilizar la función `isequal` para comparar `m` y `m_reconst`.

Modifcar el ejemplo anterior para que se simulen los esquemas de modulación y demodulación PSK y FSK.

Se observa que la señal modulada, `g`, puede tener cuatro valores de amplitud. La forma de la constelación correspondiente al modulador anterior se puede obtener mediante la función `modmap`, como en el ejemplo 5.

Modificar el ejemplo 5 para visualizar distintas constelaciones, por ejemplo, 8ASK, 32FSK, 64PSK

Considerando cualquiera de los casos de modulación/demodulación ASK, PSK o FSK:

- **Modificar los valores de `f_s` y/o `f_d`, de modo que el segundo deje de ser significativamente mayor que el primero. Comprobar el efecto sobre el proceso de demodulación.**
- **Comprobar el efecto de modificar los valores de `f_c` y/o `f_s`, de modo**

Ejemplo 4 ficom_pr4_0004_mod_demod_ask.m

```
% Ejemplo de modulación ASK
% Representación de señales digitales:
% Alfabeto de 4 símbolos
M = 4;
% Señal mensaje digital aleatoria de enteros en [0,M-1]
m = [ zeros(1,10) ones(1,10) 2*ones(1,10) 3* ones(1,10) randint(1,60,M) ]
% Modulación QASK M-aria
f_c=1;
f_d=1;      % 1 puntos (símbolo) por segundo en m
f_s=100;    % 100 puntos por segundo en g
% Modulador ASK: m -> g
[g,t_g] = dmod(m,f_c,f_d,f_s,'ask',M);
% Demodulador ASK: g -> m_reconst
m_reconst= ddemod(g,f_c,f_d,f_s,'ask',M);

% Representar
t=(1:length(m))/f_d;
subplot(311)
plot(t,m)
subplot(312)
plot(t_g,g)
subplot(313)
plot(t,m_reconst)
```

Ejemplo 5 ficom_pr4_0005_constelacion.m

```
% Ejemplo de constelación
M=8;
figure
modmap('fsk',M)
```

que el primero no sea demasiado mayor que el segundo.

El ejemplo 6 simula un sistema de modulación digital QAM M-ario con M=32:

Ejemplo 6 ficom_pr4_0006_mod_demod_qask.m

```
% Ejemplo de modulación 32-QASK (QAM):
% Alfabeto de 32 símbolos
M = 32;
% Señal mensaje digital aleatoria de enteros en [0,M-1]
m=[zeros(1,10) (M-10)*ones(1,10) (M-9)*ones(1,10) (M-8)*ones(1,10) (M-7)*ones(1,10)...
% Modulación QAM M-aria
f_c=1;
f_d=1;      % 1 puntos (símbolo) por segundo en m
f_s=100;    % 100 puntos por segundo en g
% Modulador QASK: m -> g
[g,t_g] = dmod(m,f_c,f_d,f_s,'qask',M);
% Demodulador QASK: g -> m_reconst
m_reconst= ddemod(g,f_c,f_d,f_s,'qask',M);

% Constelación
figure
modmap('qask',M);

% Representar
figure
t=(1:length(m))/f_d;
subplot(311)
plot(t,m)
subplot(312)
plot(t_g,g)
subplot(313)
plot(t,m_reconst)
```

El ejemplo 7 es similar a los anteriores, si bien introduce un elemento adicional: entre el modulador y el demodulador: se simula la aparición de ruido blanco gaussiano.

En las últimas líneas se utiliza la función `biterr`, que, junto con la función `symerr`, permite realizar un análisis básico de errores. Estas funciones calculan en número total y la tasa de errores de bits y de símbolos respectivamente. Las dos últimas líneas representan sendos diagramas oculares para la señal modulada sin ruido y con ruido.

Se recomienda **ejecutar varias veces el ejemplo** (para ver el efecto de los ruidos aleatorios); puede ser que en algunos casos no se produzcan errores. **Probar a aumentar la amplitud máxima del error**.

Probar a cambiar el esquema de modulación por otro cualquiera de los predefinidos en matlab y comparar de manera visual cómo afecta el ruido a cada tipo de modulación

¿Cómo se podría visualizar el espectro de las señales `g` y `g_ruido`.

Ejemplo 7 ficom_pr4_0007_mod_ruido.m

```
% Ejemplo de Modulación/Demodulación con ruido
% Alfabeto de 2 símbolos
M = 2;
% Señal mensaje digital aleatoria de enteros en [0,M-1]
m=[zeros(1,10) ones(1,10) zeros(1,10) ones(1,10) zeros(1,10) ones(1,10) zeros(1,10)];
% Modulación QASK M-aria
f_c=1;
f_d=1;      % 1 puntos (símbolo) por segundo en m
f_s=100;    % 100 puntos por segundo en g
% Modulador ASK: m -> g
[g,t_g] = dmod(m,f_c,f_d,f_s,'fsk',M);
% Demodulador ASK: g -> m_reconst
g_ruido = g+3*randn(1,length(g));
m_reconst= ddemod(g_ruido,f_c,f_d,f_s,'fsk',M);

% Representar
t=(1:length(m))/f_d;
subplot(411)
plot(t,m,'.')
subplot(412)
plot(t_g,g)
subplot(413)
plot(t_g,g_ruido)
subplot(414)
plot(t,m_reconst,'.')

% mostrar número y tasa de errores de bit
[n_errores, tasa]=biterr(m,m_reconst)


% Diagramas de ojo
eyediagram(g,20)
eyediagram(g_ruido,20)
```

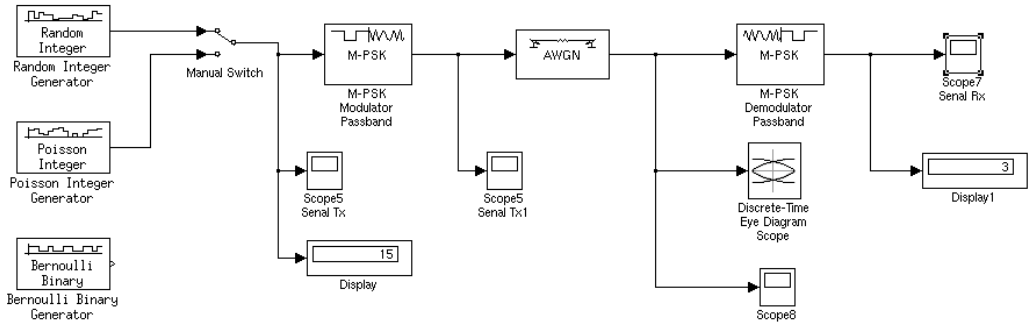

Figura 1: Ejemplo de modulación digital con Simulink

4. Modulación digital con Simulink

El bloque "Communications Blockset" de Simulink proporciona componentes predefinidos para gran cantidad de moduladores y demoduladores digitales. Asimismo, entre las demos de este bloque hay varios que nos pueden resultar interesantes, por ejemplo, la demo `qpskvmsk`, que compara los esquemas de modulación PSK y MSK.

En el ejemplo de la figura 1, tenemos los componentes necesarios para analizar el comportamiento de los esquemas de modulación digital ante un canal que introduce ruido blanco gaussiano. Uno de los componentes genera el diagrama ocular de la señal modulada con ruido durante la simulación. La mayoría de los componentes de este ejemplo proceden del "Communications Blockset" de Simulink.

Algunas de las demos de especial interés incluidas en el bloque de comunicaciones son "256 Channel ADSL" (orden `adsl_sim`), "Bluetooth Voice Transmission" (orden `bluetooth_voice`) y el bloque "CDMA reference" en general.

5. Notas

- Optimizar el proceso de cuantización de modo que se introduzca la menor distorsión posible es complicado. Existen muchos métodos; en matlab existe, por ejemplo, la función `lloyds`, que, a partir de una señal y un vector `codebook` inicial, calcula, siguiendo el algoritmo de Lloyds, unos vectores partición y `codebook` que minimizan la distorsión. Por otra parte, la función `quantiz` puede devolver un tercer parámetro en el que se indica una estimación de la distorsión.

  Asimismo, es muy frecuente el uso de esquemas predictivos de cuantización, que se basan en la idea de presuponer un rango limitado de variación del valor de la señal en torno a valores anteriores. Para simular el esquema PCM diferencial (DPCM), matlab predefine las funciones `dpcmenco` y `dpcmdeco` .

- En matlab existen funciones predefinidas para muchos esquemas de modulación digital, como los siguientes: MSK, DPSK, OQPSK, FSK, PSK, PAM y QAM entre

otros.

- En las últimas versiones de matlab (7.x), la función `dmod` se considera obsoleta, aunque sigue funcionando, y se recomienda empezar a usar las nuevas funciones que ya están disponibles y la sustituirán en el futuro: `pammod`, `fskmod`, `pskmod`, `qammod`, `mskmod`, etc.

- Existe la función predefinida `dmodce`, análoga a `dmod`. La única diferencia entre ambas es que `dmod` simula moduladores en paso de banda, mientras que `dmodce` simula moduladores en banda base. De la misma manera, la función `ddemodce` simula demoduladores en banda base.

- De entre las demos relacionadas con los sistemas de modulación digital incluidas en matlab, son destacables `scattereyedemo` (un ejemplo de uso de las funciones para representar constelaciones y diagramas oculares) y `basicsimdemo` (que compara, para el esquema de modulación QPSK, las prestaciones en cuanto a tasas de errores de símbolos y bits respecto a la relación E_b/N_0 (energía transmitida por bit/densidad espectral de ruido).

6. Referencias

[1] Comunicación Digital Avanzada (http://web.usc.es/~elusive/cd_e.html), asignatura impartida por el Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela.

[2] *Documentation for MathWorks Products, Release 14.*
http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml

[3] *GNU Octave Repository Categorical Index.*
http://octave.sourceforge.net/index/index.html

[4] V.K. Ingle y J.G. Proakis. *Digital Signal Processing Using MATLAB V.4.* PWS Publishing Company. 1997. ISBN: 0-534–93805-1.