Proyecto estimador de canal OFDM en MATLAB

Miguel Nogales González-Regueral

Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, España

February 10, 2022

1 Introducción

Esta memoria servirá para detallar la funcionalidad de los códigos usados para la parte sobre programación en Matlab de la asignatura "Electrónica Digital para Comunicaciones" de 1º de MUIT y será presentada junto a otra memoria complementaria sobre códigos en VHDL y verificación.
El proyecto de la asignatura trata de realizar un estimador de canal enteramente en VHDL y opcionalmente complementarlo con la ecualización, además de realizar verificación del funcionamiento de los códigos aportados. El código de Matlab servirá como base y guía y para este fin, adicionalmente pudiendose usar para verificación.
Primero se comentarán temas generales y algunas de las gráficas obtenidas, con algunos detalles. Posteriomente estarán los códigos y sus comentarios.

2 Estimador de canal

Este trabajo instanciará un sistema OFDM en transmisión y recepción siguiendo el estándar de DVBT además de añadir los efectos del canal. Este sistema OFDM tiene ciertas características definidas como el ancho de banda de portadoras, el número de estas o como se organizan los pilotos para la estimación de canal.
Para este ejemplo se usará una 16-QAM, mostrandose las figuras más importantes. La constelación transmitida se puede ver en la Figura 2, siendo lo más importante como los símbolos están bastante cercanos entre ellos, lo cual hará que se necesite una mayor SNR para mantenter la BER, aunque permite enviar mayor cantidad de información por símbolo.

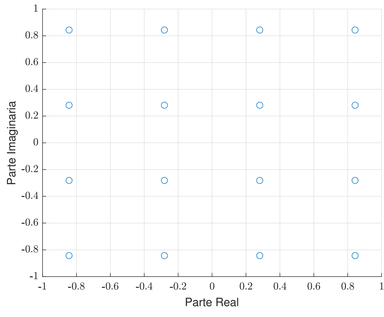

Figure 2.1: Constelación transmitida.

Algunas características de la señal OFDM interesantes y a tener en cuanta son sus propiedades espectrales y temporales. En frecuencia la señal tiene un ancho de banda determinado muy abrupto, esto la hace especialmente buen en cuanto a eficiencia espectral, mientras que en tiempo, a diferencia de otras técnicas de modulación, no se puede diferenciar nada a simple vista debido a la transformación por medio de la IFFT realizada en el transmisor.

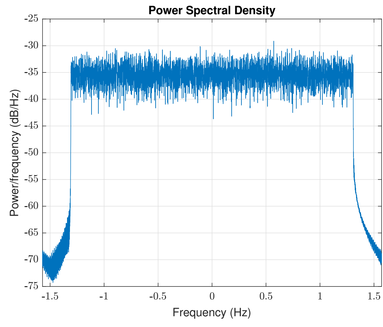

Figure 2.2: PSD de la señal OFDM.

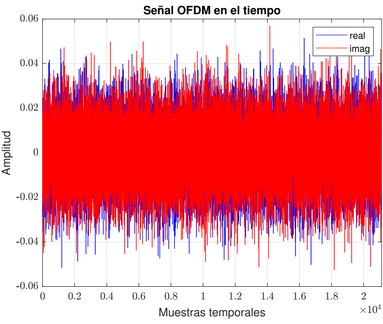

Figure 2.3: Muestras temporales señal OFDM.

Esta señal se pasará por el canal y mediante los pilotos, los cuales se introducen en el símbolo elegido cada 12 muestras, se estimará el canal despúes de deshacer el código con el bloque de PRBS definido en el estándar. El canal propuesto para probar el sistema junto con su estimación se muestra en la Figura 2.

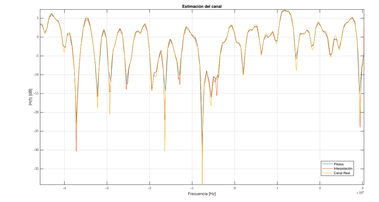

Figure 2.4: Canal real y canal estimado.

Para concluir, la constelacion habiendo deshecho los efectos del canal es la siguiente:

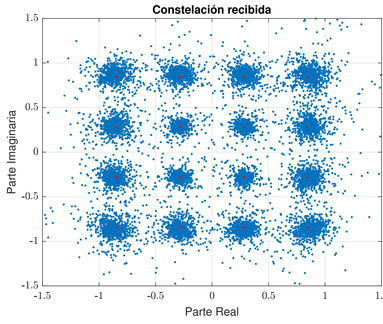

Figure 2.5: Constelación recibida.

Antes de comenzar con los códigos es interesante apreciar el efecto del canal en la BER. Por el mero hecho de introducir el canal, aunque lo ecualicemos totalmente, la probabilidad de error aumentará, y llegará una SNR donde por más que se aumente habrá una Pe constante.

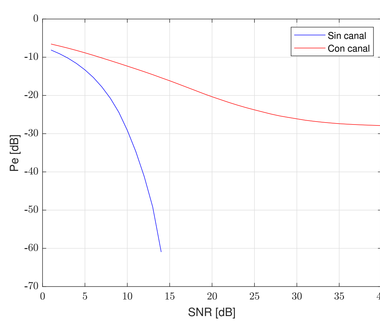

Figure 2.6: Efectos del canal sobre la ber.

3 Análisis de los códigos

3.1 Channel

Este primer código es la función que se encarga de modelar el canal y de convolucionarlo con la señal. Este canal se define en frecuencia discreta y mediante una sentencia vectorial se prepara el canal para cada uno de los *taps*, despúes se suman todos. Mediante la IFFT se pasa a frencuencia y después se convoluciona.
Se deja un canal de prueba con un solo tap:

```
%% dummy channel
% deltaF = 1/Tsymb;
% k = -NFFT/2:NFFT/2-1;
% canal = 0.4*exp(-1j*2*pi*deltaF*k*1e-6)+0.1*exp(-1j*2*pi*deltaF*k*5.4e-6);
% canalTiempo = ifft(ifftshift(canal));
```

Además, se ha eliminado todo tipo de bucle para recrear el canal, dejando la expresión:

```
deltaF = 1/Tsymb;
k = -NFFT/2:NFFT/2-1;
canal =
 ↪  rho.*exp(-1j*theta).*exp(-1j*2*pi*deltaF*repmat(k,20,1).*repmat(tau,1,NFFT));
canal = sum(canal,1);
```

3.2 Noise

La función que define el ruido es muy sencilla, se le pasa de parámetros la señal y la SNR en dB y así se le suma el ruido blanco con la potencia correcta para cumplir la relación señal a ruido.

3.3 OFDM TX DVT

El código del transmisor ya empieza a ser más complejo. Se definen unos parámetros por defecto y la colocación de los pilotos (cada 12 muestras). Posteriormente se añade el bloque PRBS calculándose tantos signos para los pilotos como sea necesario por la cantidad de símbolos. Luego se define la constelación dentro de las tres posibles y se genera una secuencia aleatoria de bits a transmitir.
El siguiente paso consta del cambio de bits a símbolos, primero pasándolo a decimales mediante una transformación con una matriz de potencias de dos y luego usándolo como índice a la matriz de la constelación, definida de manera que use codificación de Gray.

```
aux  = reshape(bits_tx, M, []).'; % numbits/M x M
symb = zeros(size(aux, 1),1);     % numbits/M x 1
pot2 = kron(ones(length(symb),1),(2.^(0:M-1)));
symb = sum(pot2.*aux,2); % primera columna = lsb
```

Luego solo resta la conversión de serie a paralelo e IFFT. Habiendo realizado estos pasos la señal de transmisión está creada.

3.4 OFDM RX DVT

Por otro lado está el receptor el cual realiza los procesos inversos al transmisor, pasando de parelaleo a serie, quitando el prefijo cíclico, haciendo la FFT y demás. En la línea 69 se realiza la estimación del canal, calculando la H a partir de los piltoos, los cuales ya se les ha quitado el código. Después se encuentra un bucle for el cual itera sobre los varios símbolos OFDM recibidos, interpolando mediante ciertas operaciones matriciales y diviendiendo los datos recibidos entre ese canal estimado, realizando una equalización *zero forcing*, mostrado en el siguiente código.

```
% Interpolar h
for i = 1:NUM_SYMB
        y = zeros(length(pilotsLoc)-1,2);
        y(:,2) = hest(1:end-1,i);
        y(:,1) = (hest(2:end,i)-hest(1:end-1,i))/12;
        v = repmat((0:11),length(y),1);
        v = v.*kron(y(:,1),ones(1,12));
        v = v+kron(y(:,2),ones(1,12));
        v = reshape(v.',[],1);
        % Es una sola concatenación, el warning sale por que la variable
        % donde asigno el valor es una de las que concateno. Si usara en
        % vez de esa línea algo como "nueva_var = [v; hest(i,end)];" no
        % daría error. (comprobado)
        v = [v; hest(i,end)];
        ofdm_freq_rx_eq(:,i) =
 ↪  ofdm_freq_rx_o(ceil((NFFT-useCarrier)/2)+(1:useCarrier),i)./v;
end
```

Posteriormente se demodula la constelación equalizada usando propiedades para no tener que calcular la distancia mínima a los puntos.

```
switch CONSTEL
case 'BPSK'
        bits_rx = rx_constel<0;
case 'QPSK'
        bits_rx = zeros(1,length(rx_constel)*2);
        bits_rx(2:2:end) = real(rx_constel)<0;
        bits_rx(1:2:end) = imag(rx_constel)<0;
case '16QAM'
        Xcx = mean(max(abs(C))-min(abs(C)));
        Xcy = Xcx;
        bits_rx = zeros(1,length(rx_constel)*M);
        bits_rx(3:4:end) = real(rx_constel)<0;
        bits_rx(1:4:end) = imag(rx_constel)<0;
        bits_rx(4:4:end) = abs(real(rx_constel))-Xcx<0;
        bits_rx(2:4:end) = abs(imag(rx_constel))-Xcy<0;
end
```

3.5 Integración

Este código (llamado extremelySimplified.m) sirve a modo de *toplevel*, para unificar las funciones y otorgar cierta parametrización al conjunto de manera que se puedan simular muchas posibilidades. Esto se puede hacer asignando valores simplemente a estas variables:

```
%% Configuración del sistema OFDM
mode = 8;
CONSTEL = '16QAM';    % Constelación utilizada BPSK, QPSK, 16QAM
SNR = 60;
verbose = 0;
noiseON = 0;
canalON = 0;
```

4 OCTAVE IT

En esta sección se comentarán brevemente los códigos modificados más importantes que se han usado para verificación con CoCoTb, siendo llamados por la librería Oct2py.

4.1 PRBS

El código del PRBS es directamente extraído de los códigos de transmisor y receptor, y encapsulado en una función, nada extra que comentar.

4.2 Golden Channel Estimator

De la misma manera que el código anterior, se encapsula gran parte del sistema, pero sin llegar a demodular ya que no es necesario. Se devuelven los valores de las muestras transmitidas, la h estimada, el canal interpolado v, los pilotos y la señal ecualizada.

4.3 Golden Channel Estimator 2

Esta es una pequeña modificación del código para que los pilotos sean tratados correctamente como dicta el estándar de DVBT, y al cambiar el VHDL poder contrastar.

5 RESUMEN HITOS REALIZADOS

Recogiendo todo lo mencionado anteriormente, se han realizado las siguientes tareas:

- Realización de los códigos en Matlab para el estimador de canal, receptor y transmisor.
- Optimización de estos minimizando los bucles mediante programación vectorial.
- Modos 2k y 8k, gran parametrización del código.
- Añadido de 16-QAM.
- Programación en funciones para aumentar la interoperabilidad.
- Preparación de códigos para ser movidos en Octave vía Oct2Py.

5.1 Archivos usados

- channel.m
- noise.m
- OFDM_TX_DVT.m
- OFDM_RX_DVT.m
- extremelySimplified.m
- de OctaveIt
  - GoldenChannelEstim
  - GoldenChannelEstim2
  - PRBS.m