UNIVERSIDADE DA CORUÑA
FACULTADE DE INFORMÁTICA

Práctica 1: Modulacións dixitais

Nombre estudiante: *Miguel Blanco Godón*

Curso: *Software de comunicacións (Enxeñaría de computadores)* – Docente: *Óscar Fresnedo Arias*
Fecha de entrega: *26 de marzo do 2021*

Introdución

O obxectivo desta práctica é a avaliación das modulacións do estándar IEEE 802.11n cando son transmitidas por canles AWGN. O método de avaliación será a simulación do sistema en Matlab. Concretamente, as modulacións avaliadas son 2-PSK, 4-PSK, 16-QAM, 64-QAM.

Arquitectura do sistema

O sistema está composto por tres partes, un modulador dixital, unha canle AWGN e un demodulador dixital. Un sistema de comunicación está formado por, como mínimo, un emisor, unha canle de comunicación en un receptor. O modulador forma parte do emisor, a canle AWGN é o medio para a comunicación e o demodulador forma parte do receptor.

Modulador dixital. É a primeira parte do sistema, a cal que encarga de convirtir o fluxo de bits un conxunto de símbolos dependentes da modulación. Para iso basámonos na representación discreta equivalente, a través dun método denominado *representación vectorial de sinais*. Este método, que forma parte do análise espacial de sinais, foi desenvolvido primeiramente por V.A.Kotel'nikov en 1947. A idea consiste en representar cada elemento dun conxunto de sinais transmitidas mediante un vector M-dimensional, sendo N o número de funcións básicas ortonormais necesarias para unha representación xeométrica única dos sinais transmitidos. Un parámetro importante na modulación, aparte do tipo, é o número de niveis. O número de niveis dunha modulación dánolo o valor de M, o número de sinais necesarias, e implica que cada símbolo conterá a información de $log_2 M$ bits. As bases a usar dependen de cada modulación:

- **Phase Shift Keying (PSK)**: modifica a fase nunha portadora de frecuencia constante.
- **Quadrature Amplitude Modulation (QAM)**: modifica a fase e a amplitude da sinal portadora.

Canle AWGN. En comunicacións, unha canle é un medio de transmisión a través dos cales viaxa a información. Neste caso, o estándar utilizado é un estándar de comunicación inalámbrico, polo que o medio de transmisión é o aire. Nesta práctica simúlase un canle AWGN (Additive White Gaussian Noise) debido a que é un tipo de canle que se asemella a moitos fenómenos de interferencia na vida real, como pode ser o ruído térmico. A canle modélase como: $\vec{r} = \vec{s} + \vec{n}$, onde o sinal recibido é igual ao sinal emitido máis a interferencia da canle.

Demodulador. O demodulador forma parte do receptor, e o seu traballo é recuperar o fluxo de bits transmitido a partir da información que porta a portadora. Para isto o demodulador debe estimar cal é o símbolo transmitido, e despois convertir os símbolos a bits. Para que o demodulador funcione, debe ser compatible co modulador.

Implementación en Matlab a través do modelo discreto equivalente

Para a implementación separo código en sete módulos. Ilústrase o funcionamento a partir dun caso de proba de 12 bits:

- **modulate.m**: implementa o modulador. En función dos parámetros de entrada, establece a dimensión das bases da modulación e modula o fluxo de bits a partir dun vector de símbolos adecuado a cada modulación.

```
% modulation type checking and modulation computation
if (strcmp(modulation_type, "PAM"))
        modulation = pam(modulation_levels, ordering);
        dimension = 1;
        complex = false;
elseif (strcmp(modulation_type, "PSK"))
        modulation = psk(modulation_levels, ordering);
        dimension = 2;
        complex = true;
elseif (strcmp(modulation_type, "QAM"))
        modulation = qam(modulation_levels, ordering);
        dimension = 2;
        complex = true;
else
        error('Unsupported modulation type. Supported modulations: PAM, PSK, QAM');
end
% reshaping matrix so that it fits the number of bits/symbol
input_matrix = reshape(input_bitstream, log2(modulation_levels), []);
% modulation_computation
modulated_stream = modulation(bi2de(input_matrix', 'left-msb')+1);
```

Como se pode observar no código, ao primeiro obtense o vector de símbolos para a modulación. Despois, reorganízase o vector como unha matriz de dúas dimensións, de tal xeito que en cada columna quede o equivalente a un símbolo, e por último, indexamos o vector de símbolos da modulación a partir dos da matriz;que pasa a ser outro vector debido a que se fai unha conversión por columnas a decimal, o que nos dá a posición (comezando a contar dende 0) do símbolo no vector de símbolos da modulación. Súmaselle 1 xa que matlab indexa dende 1 ata N. Así obtense o vector de símbolos modulados. Todo este proceso pódese facer chamando á función do módulo *modulate*:

```
bits = randn(1,12)>0.5

bits =

  1x12 logical array

   0   0   0   0   1   0   0   1   0   0   1   1

modulate(bits, uint8(4), "PAM", 'bin')

ans =

    -3    -3     1    -1    -3     3
```

- **awgn.m**: neste módulo emúlase a canle AWGN en base a dous parámetros, a dimensión da modulación (se 1, só ruido real, se 2, real e complexo). Como a canle é de ruido aditivo, pódese calcular simplemente mediante a adición de vectores. O ruido créase en relación a un parámetro de simulación, N_0, e segue unha distribución gaussiana de $\mu = 0$ e $\sigma = \frac{N_0}{2}$. O parámetro N_0 calcúlase neste módulo a partires do valor de $\frac{E_b}{N_0}$.

```
% compute symbol energy
symbol_energy = mean(abs(modulated_stream).^2);
% compute bit energy
bit_energy = symbol_energy / double(bits_per_symbol);
% compute N0
N0 = bit_energy / (10^(dbEbN0/10));
```

```
if (dimension == 1)
        % creates gaussian noise with mean 0 and tipic deviation N0/2
        noise = sqrt(N0/2) * randn(size(modulated_stream));
        % adds noise to the modulated stream
        noisy_modulated_stream = modulated_stream + noise;
elseif ((dimension == 2) & complex)
        % creates gaussian noise with mean 0 and tipic deviation N0/2
        real_noise = sqrt(N0/2) * randn(size(modulated_stream));
        imag_noise = sqrt(N0/2) * randn(size(modulated_stream))*1j;
        noisy_modulated_stream = modulated_stream + real_noise + imag_noise;
end
```

Aquí vese cun exemplo a partir do vector de símbolos modulados anterior:

```
m =

    -3    -3     1    -1    -3     3

[recv, e_bit, e_simbolo, n0] = awgn(m, uint8(log2(4)), uint8(1), false, 5)

recv =

   -4.9284   -2.2008    1.8532   -1.6854   -2.9138    1.3956


e_bit =

    3.1667


e_simbolo =

    6.3333


n0 =

    1.0014
```

- **demodulate.m**: o demodulador realiza exactamente a operación inversa ao modulador, partindo dende o mesmo vector de símbolos da modulación.

```
% modulation type checking and modulation computation
if (strcmp(modulation_type, "PAM"))
        modulation = pam(modulation_levels, ordering);
elseif (strcmp(modulation_type, "PSK"))
        modulation = psk(modulation_levels, ordering);
elseif (strcmp(modulation_type, "QAM"))
        modulation = qam(modulation_levels, ordering);
else
        error('Unsupported modulation type. Supported modulations: PAM, PSK, QAM');
end
% vector replication to parallel contrast of input stream
input_matrix = repmat(modulated_stream, length(modulation), 1);
% correlate modulated input with modulation
input_matrix = input_matrix - modulation.';
% get the position to which symbol demodulates
[~, pos] = min(abs(input_matrix));
% returning bitstream
demodulated_stream = logical(reshape(de2bi(pos-1, 'left-msb')', 1, []));
```

Do mesmo xeito ca no modulador, obtense o vector de símbolos da modulación. Replícanse as filas do vector de símbolos para poder calcular a distancia euclídea a todos os símbolos cunha soa instrución. Despois extráese a posición do símbolo na constelación a través do mínimo, que matlab faino por columnas, e por último simplemente se desfai a tradución, pasando a binario as posicións correspondentes. Pódese observar o seguinte exemplo:

```
demodulate(recv, uint8(4), "PAM", 'bin')

ans =

  1x12 logical array

   0   0   0   0   1   0   0   1   0   0   1   0
```

Onde se pode observar que houbo un erro da demodulación.

- **transmit.m**: é unha abstracción do sistema, que xa fai todo o proceso de modulación, transmisión pola canle e demodulación.
- **pam.m**: devolve un vector cos símbolos da modulación M-PAM. Concretamente, calcula os vectores a partires da expresión $s_i = 2k + 1 - M \quad \forall k = 0, ..., M - 1$.
- **psk.m**: devolve un vector cos símbolos da modulación M-PSK. Concretamente, calcula os vectores a partires da expresión $s_i = \sqrt{\frac{1}{2}}cos(\theta_k) + \sqrt{\frac{1}{2}}sen(\theta_k)j \quad \forall k = 0, ..., M - 1$ onde $\theta_k = \frac{2\pi k}{M}$.
- **qam.m**: devolve un vector cos símbolos da modulación M-QAM. Bótase man da función *qammod* do *Communications toolbox* de Matlab.

Isto todo xúntase no script *p1.m* no cal se executan as transmisións e xéranse as figuras.

Resultados

Conclusión