Activitat pràctica 1 (AC1)

Comunicacions Digitals

Curs 2025-2026

Joan Claudi Socoró

Ferran Castro

Índex

1. Introducció

Aquest primer exercici tractarà de la simulació de modulacions QAM amb l'entorn Matlab. Es tracta de simular la BER de tres modulacions (4-QAM, 8-PSK i 16-QAM) i comparar les corbes de BER obtingudes amb les corresponents corbes teòriques. Per fer la simulació usarem la interfície d'usuari BerTool, que permet generar corbes de probabilitat d'error i comparar simulacions. Aquesta eina realitza simulacions de Monte Carlo, és a dir, simula l'enviament i recepció d'informació digital per un canal de transmissió i compara la informació enviada i rebuda per tal de calcular la probabilitat d'error. La eina BerTool crida a una plantilla (BerTooltemplate.m) on haureu de posar el vostre codi.

2. Eina BerTool i bertooltemplate.m

Quan executeu la comanda BerTool dins del Matlab se us obre una interfície gràfica d'usuari, a on podeu simular els vostres sistemes de comunicacions. En la part inferior podem escollir paràmetres de la simulació com són els valors de relació Eb/No en dB (en forma de vector de valors), el tipus de canal o la modulació. De les pestanyes d'aquesta part inferior anirem a la que es diu Monte Carlo, la que ens permet adaptar la simulació a qualsevol context. En aquesta, només escollim els valors d'Eb/No, ja que la modulació i el canal els implementa la funció *.m que es crida en el camp "Function name". A part, també definim els criteris de parada alhora de calcular la corba de probabilitat d'error per aquest cas, per evitar que la simulació trigui massa temps i puguem tenir uns resultats estadísticament significatius, que són:

- Number of errors: número màxim d'errors a partir del qual es pararà la simulació.
- Number of bits: número màxim de bits transmesos a partir del qual es pararà la simulació.

Farem que s'aturi la simulació de cada valor de Eb/No per cada QAM quan s'arribi a 1000 bits erronis o bé 1e8 bits transmesos (el que succeeixi primer).

Per realitzar l'activitat pràctica partiu del fitxer bertooltemplate.m, que conté la estructura d'un exemple per poder simular amb la eina bertool.m. La funció que es generi ha de permetre calcular la probabilitat d'error d'un bloc de bits simulat, usant un determinat sistema de comunicacions (amb un model de transmissor, canal i receptor), en unes condicions de canal concretes. La funció tindrà la següent sintaxi:

```
[ber, numBits] = nom_funcio(EbNo, maxNumErrs, maxNumBits)
```

Per això, una variable d'entrada és la Eb/No en dB, és a dir, la qualitat de senyal en el receptor que volem simular. La interfície BerTool farà la simulació a partir d'uns certs valors de Eb/No mínims i màxims així com un increment que es defineix en el camp Eb/No range de la pestanya Monte Carlo. Per cada valor especificat es cridarà a la funció amb un valor d'Eb/No i la funció retornarà el valor de BER obtingut amb un cert bloc de dades. La funció haurà de realitzar les següents tasques:

1. Simular el sistema de comunicacions definit amb un valor d'Eb/No concret passat com a primer argument de la funció

2. Parar la simulació quan el nombre d'errors (bits erronis) superi el llindar especificat com a segon argument d'entrada (argument maxNumErrs), o quan el nombre de bits simulat superi el llindar especificat pel tercer argument d'entrada (argument maxNumBits)

Podeu començant fent la crida a l'editor de Matlab amb:

edit bertooltemplate

Us obrirà una plantilla preparada per a que la editeu (i la guardeu amb un nou nom!), a on podreu veure on heu d'inserir el vostre codi. Veureu que en el codi s'invoca a la variable BERTool.getSimulationStop, per permetre que la crida a la vostra funció s'aturi (invocant a la funció break) quan l'usuari pari la simulació amb el botó Stop.

Dins de la funció es defineixen les variables totErr (nº d'errors detectats) i numBits (nº de bits simulats). Dins del "while" és a on haurem d'anar incrementant aquestes dues variables fins que s'assoleixi algun dels dos criteris de parada esmentats.

3. Nocions bàsiques sobre simulació de sistemes de comunicacions

La tècnica de simulació de Monte-Carlo, es basa en el següent procediment:

- Generar aleatòriament bits d'informació a transmetre
- Codificar-los, modular i generar la forma d'ona del senyal transmès
- Passar el senyal transmès per un model de canal (AWGN o amb Fading – selectiu o no selectiu en freqüència).
- Desmodular el senyal rebut, detectar i descodificar els bits.
- Comparar els bits transmesos amb els bits rebuts per calcular una BER basada en una mitja estadística entre el nombre de bits erronis i el nombre total del bits transmesos.

Existeixen també altres tècniques d'estimació de la BER, per exemple, les tècniques semi-analítiques, que estimen la BER a partir de l'estimació de la Eb/N0, el que implica obtenir només la qualitat de senyal rebuda, i en general requereix menys mostres de senyal per a la seva estimació. No obstant, el seu ús es limita a casos en que podem conèixer a priori la corba de probabilitat d'error de la modulació en l'entorn de canal que es vol simular (senzill per canals AWGN i amb modulacions senzilles, però força més complex per a modulacions més complexes, o per canals amb multicamí i variacions temporals).

La simulació del sistema de comunicacions la fem amb senyal passa-baixes equivalent, és a dir, suposant que no els pugem en freqüència a la banda de la portadora. Això permet treballar amb freqüències de mostratge baixes, que en el límit poden arribar a ser igual a la freqüència de símbol (és a dir, generem 1 mostra per símbol), el que permet accelerar el procés de simulació. En aquest context, cal tenir present que els canals de comunicació a simular també han de ser canals modelats en el context dels senyals passa-baixes equivalents. Per exemple, en el cas del soroll additiu de la comunicació, aquest serà un soroll complex que tindrà energia tant a la part real com a la part imaginària del senyal.

A continuació donem algunes pautes per poder començar a fer una simulació.

3.1 Variables

Definirem les següents variables de la simulació en la part a on diu "Set up parameters" de la funció .m:

- **constel_symb**: vector de longitud M (ordre de la modulació) amb els valors complexos (o reals) dels diversos punts de la constel·lació. Per exemple, per una 4-QAM, constel_symb = [1+1i;1-1i;-1-1i;-1+1i]. L'ordre de la constel·lació es pot calcular a partir de la longitud d'aquest vector, és a dir, M = length(constel_symb). Si volem calcular el número de bits per símbol, ho podem fer amb k = log2(M);
- **constel_bits**: matriu de M files i k columnes (k = log2(M) = nº de bits per símbol), amb les combinacions en bits de cada símbol correlatiu a la mateixa posició del vector anterior. Per exemple, pel cas de la 4-QAM, constel_bits = ['00';'11';'01';'10'].
- **nBitsBloc**: Número de bits que simularem en cada iteració de la funció. Amb aquesta variable, podem calcular el número de símbols per cada iteració (nSymbolsBloc) si abans hem calculat la variable k.

3.2 Modulació

El cos de la simulació es realitza dins del "while" de la funció .m. Primer caldrà generar una trama de nSymbolsBloc símbols amb la funció randi.m, que genera una seqüència de N números aleatoris enters (per exemple N=1000):

```
txSymb = randi([1 4],1,N);
```

Aquest exemple genera un vector txSymb de N números enters aleatoris entre 1 i 4, a on N seria el número de símbols aleatoris, i 4 l'ordre de la modulació. En realitat, txSymb són els índex dels símbols que volem transmetre, i no el valor dels símbols en si.

Per modular amb una 4-QAM usem el vector de constel·lació abans definit i mapem els enters cap als símbols que hem de transmetre amb la senzilla crida:

```
txSig = constel_symb(txSymb);
```

De forma que txSig és un vector de la mateixa longitud que txSymb però que ja conté els valors complexos de la modulació dissenyada en el vector constel_symb. Fixeu-vos que aquesta és una assignació vectorial que permet evitar fer un bucle assignant els valors un a un.

Si volem calcular la potència de senyal transmès (Ps) ho podem fer assumint que els símbols de la constel·lació son equiprobables (la funció randi.m ens garanteix aquesta propietat, en ser números aleatoris amb distribució uniforme). Per calcular aquesta potència amitjanarem l'energia de cada símbol, essent l'energia el mòdul al quadrat de cadascun dels símbols, usant la funció mean.m:

```
Ps = mean(abs(constel_symb).^2);
```

3.3 Canal

Per simular el canal de comunicacions començarem amb un model senzill: el canal AWGN (Additive White Gaussian Noise channel). Per cada valor de Eb/No s'haurà de calcular la potència de soroll a afegir al senyal transmès per calcular el senyal rebut pel canal. Optem per la solució de fixar la potència de senyal transmès, i anar variant la potència de soroll (també es podria fer al revés). Hem de tenir en compte que donat que la simulació és amb senyals passa-baixes equivalents (sense pujar a la freqüència de portadora), el soroll és un procés aleatori complex, i que les dues components (real i imaginària) es modelen com a processos independents i de la mateixa potència. Per tant, la potència de soroll es reparteix per igual en ambdues components. Aleshores, sigui Pn la potència de soroll a simular, podem generar N mostres de soroll complex amb la següent comanda:

```
Soroll = sqrt(Pn/2)*(randn(1,N)+1i*randn(1,N));
```

La potència de soroll la podem expressar tenint en compte que aquest és un procés blanc (White), és a dir, amb espectre pla i densitat espectral No [W/Hz]. Tenint en compte que treballem amb una freqüència de mostratge de fs = Rs [símbols/s] = 1/Ts, dins l'ample de banda de fs Hz tindrem la potència total de soroll igual a:

$$Pn = No \times fs$$

Per calcular la potència de soroll donada la restricció del valor de Eb/No, hem de tenir present que la energia de bit és igual a la potència de senyal transmès multiplicat pel temps de bit, és a dir:

$$Eb = Ps \times Tb = Ps \times (Ts/k)$$

Essent k el número de bits per símbol, i Ts el temps de símbol.

Exercici: Usant les dues equacions anteriors, trobeu l'expressió que permet expressar la potència de soroll Pn en funció de la Eb/No, la potència de senyal Ps i el número de bits per símbol.

3.4 Receptor

Per simular el procés de recepció, aplicarem el detector de màxima versemblança (Maximum Likelihood) que es basa en maximitzar la probabilitat del símbol detectat atès que es disposa de la observació del senyal rebut pel canal. Aquest criteri coincideix amb el criteri de la distància mínima, és a dir, escollim com a símbol detectat aquell símbol de la constel·lació amb mínima distància amb el senyal rebut. Per demodular amb la constel·lació personalitzada que hagueu creat ho podeu fer amb la crida a la funció següent:

```
[detSym_idx,nerrors] = demodqam(rxSig,constel_symb,constel_bits,txSymb)
```

A on:

- rxSig és la seqüència txSig passada pel canal de transmissió
- constel_symb i constel_bits son les variables que defineixen la constel·lació explicats en l'apartat 3.1
- txSymb són els índex dels símbols que hem transmès

- detSym és un vector de la mateixa longitud que rxSig i txSymb que conté els índex dels símbols detectats
- nerrors és una variable entera amb el número de bits erronis acumulats de tota la trama, tenint en compte el mapeig definit per la variable constel_bits

Ajuda: Observeu que nerrors és la variable que heu d'anar acumulant a dins de la variable totErr per poder obtenir el resultat de la simulació per un valor d'Eb/No concret.

4. Requisits Activitat 1

Genereu tres versions a partir de la funció bertooltemplate.m, a on canvieu el tipus de modulació QAM. Heu de simular un 4-QAM, un 8-PSK i un 16-QAM.

Pel lliurament de l'activitat haureu de dipositar al pou corresponent un fitxer .ZIP amb nom comdig_AC1_X.zip, essent X el número de grup publicat. El fitxer contindrà les tres versions de la funció, anomenades simula_qam1.m, simula_qam2.m i simula_qam3.m. El requisit és que totes tres funcionin sense donar error en la interfície bertool.m de Matlab, i permetin dibuixar les corbes de probabilitat d'error de les tres modulacions simulades en un canal AWGN. Dins del fitxer ZIP incloureu també un PDF amb la gràfica de les tres corbes d'error simulades amb la eina bertool.m. Anomeneu la llegenda de forma adequada.

5. Estructura de la memòria.

Pel lliurament de l'activitat haureu de dipositar al pou corresponent un fitxer .ZIP amb nom comdig_AC1_X.zip, essent X el número de grup publicat. El fitxer contindrà;

- Les tres versions de la funció, anomenades simula_qam1.m, simula_qam2.m i simula_qam3.m.

- Una memòria en format pdf amb els següents apartats;
  - Portada: indicant el títol de la pràctica, el nº de grup i els noms dels participants
  - Introducció; què es vol simular?
  - Descripció: Llistats dels vostres codis (els tres codis) i explicació de com funciona el codi
  - Captura de pantalla de les tres gràfiques de BER obtingudes juntament amb les tres gràfiques teòriques (totes elles al mateix dibuix)
  - Resultats: explicació dels resultats obtinguts i la seva interpretació, amb captura de pantalla de les gràfiques de BER obtingudes i una interpretació dels resultats
  - Conclusions

6. Data límit

El lliurament d'aquesta activitat es realitzarà amb data límit el **diumenge 1 de Març de 2026 a les 23:55h**. Qualsevol pràctica lliurada més tard d'aquesta data podrà optar com a màxim a una nota de 7 (retard fins a dos setmanes) o de 5 (retard superior a dos setmanes).

7. Avaluació

Encara que la pràctica es faci en grups de tres persones la nota final serà individual I es basarà en la documentació lliurada I una entrevista personal a la que haureu de contestar preguntes relacionades amb bels fonaments teòrics de l'exercici, preguntes sobre el vostre codi i haureu de fer modificacions al vostre codi software. Cada alumne haurà de portar el seu portàtil per poder fer preguntes/demanar canvis al codi simultàniament a tots els alumnes.