Memòria Comdig AC1

Grup Nº2

Àlex Reina

David Torres

Albert Marquillas

Índex

Introducció

En aquesta pràctica es vol simular el comportament de diferents modulacions digitals en un sistema de comunicacions. Concretament, s'analitza la probabilitat d'error de bit (BER) per a les modulacions 4-QAM, 8-PSK i 16-QAM mitjançant simulacions de Monte Carlo en el programa Matlab. Per tal de poder veure els resultats obtinguts de manera grafica, també hem fet ús de l'eina BerTool.

L'objectiu principal ha estat comparar les corbes de BER obtingudes mitjançant la simulació dels nostres programes amb les corbes teòriques corresponents generades amb l'eina BerTool.

Les gràfiques ens han permès observar com la qualitat del senyal (Eb/No) afecta el rendiment de cada modulació en un canal AWGN. Aquest estudi ens ha permès entendre els efectes dels diferents tipus de modulació en la robustesa del sistema davant del soroll.

Descripció

Hem desenvolupat tres codis en MATLAB per simular sistemes de comunicació digital mitjançant les modulacions 4QAM (QPSK), 8QAM i 16QAM, integrats en l'entorn de BERTool. Cada codi implementa una simulació Monte Carlo d'una transmissió a través d'un canal AWGN i segueix una estructura comuna, amb algunes diferències fonamentals en la definició de la constel·lació i el mapeig de bits.

Els passos bàsics comuns en tots tres codis són:

- Inicialització i definició de paràmetres:

  S'inicialitzen variables com el nombre de símbols, els bits per símbol (calculat com k=log2(M)k=\log_2(M)k=log2(M)), i es defineix la constel·lació amb el seu corresponent mapeig de bits (normalment amb codificació Gray per minimitzar els errors). Per exemple, en la 4QAM es defineixen quatre símbols normalitzats (amb potència mitjana igual a 1) que porten 2 bits cadascun, mentre que en la 16QAM es treballa amb 16 símbols (també normalitzats) amb 4 bits per símbol.

- Generació de la senyal transmès:

  S'escullen aleatòriament índexs de símbols (equiprobables) que es mapegen a la constel·lació definida, simulant així la transmissió de dades.

- Addició de soroll AWGN:

  Es genera soroll complex gaussià amb una potència calculada a partir de la relació Eb/N0E_b/N_0Eb/N0 (convertida de dB a lineal), i s'afegeix a la senyal transmès. Aquesta etapa simula les condicions reals del canal, on el soroll afecta la qualitat de la transmissió.

- Demodulació i càlcul d'errors:

  S'aplica un detector de distància mínima (o de màxima versemblança) per a demodular la senyal rebut. Es comparen els bits obtinguts amb els bits transmesos, acumulant el nombre d'errors per calcular la taxa d'error de bits (BER).

- Acumulació de resultats:

  La simulació continua fins que es compleixi algun dels criteris de parada (nombre màxim d'errors o de bits processats). Finalment, es calcula el BER com la relació entre el nombre total d'errors i el nombre total de bits simulats.

Fucnions Importants

1. Càlcul del nombre de bits per símbol

```
bitsSimbolo = log2(cantidadSimbolos);
```

- log2(x): Calcula quants bits per símbol es poden transmetre.
- Per 4-QAM: log2(4) = 2 bits/símbol.

2. Càlcul de la potència del senyal

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

- abs(x): Retorna el mòdul dels símbols complexos.
- .^ 2: Eleva al quadrat per obtenir la potència de cada símbol.
- mean(x): Fa la mitjana de les potències dels símbols per obtenir la potència mitjana del senyal Ps.

3. Conversió d'Eb/N0 de dB a lineal

```
pRuidoEbNo = 10 ^ (EbNo / 10);
```

- 10^(x/10): Converteix Eb/N0 de decibels (dB) a valor lineal.

4. Càlcul de la potència del soroll

Pn = Ps / (pRuidoEbNo * bitsSimbolo);

- Pn = Ps / (Eb/N0 * k):
  - Ps: Potència del senyal transmès.
  - pRuidoEbNo: Relació energia per bit a densitat de soroll en escala lineal.
  - bitsSimbolo: Nombre de bits per símbol.
- Aquesta expressió sorgeix de la relació:

$$Pn = \frac{Eb}{N0/Eb} \cdot \frac{1}{k}$$

  on k és el nombre de bits per símbol.

5. Generació de símbols aleatoris

txSymb = randi([1 cantidadSimbolos], 1, numSymb);

- randi([a, b], m, n): Genera nombres enters aleatoris entre a i b en una matriu m × n.
- Això simula la transmissió de dades aleatòries.

6. Generació del soroll AWGN

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

- randn(1, numSymb): Genera nombres aleatoris distribuïts normalment (N(0,1)\mathcal{N}(0,1)N(0,1)) per la part real.
- 1i * randn(1, numSymb): Genera nombres aleatoris per la part imaginària.
- sqrt(Pn / 2): Ajusta la variància del soroll perquè tingui la potència adequada.
- Com el soroll AWGN és complex, es reparteix la potència igualment entre les parts real i imaginària.

7. Senyal rebut després del canal

rxSig = txSig.' + Soroll;

- La senyal rebut és la senyal transmesa (txSig) més el soroll (Soroll).

8. Demodulació per distància mínima

[detSym_idx, nerrors] = demodqam(rxSig, constelSymb, constelBits, txSymb);

- demodqam() és la funció que implementa demodulació per distància mínima:
    - Compara el senyal rebut rxSig amb tots els punts de la constel·lació constelSymb.
    - Assigna cada punt rebut al símbol més proper en l'espai complex.
    - Calcula els errors comparant els bits transmesos i els bits demodulats.

9. Càlcul de la BER (Bit Error Rate)

ber = totErr / numBits;

- BER = (Errors totals) / (Bits totals transmesos).
- Això mesura la qualitat de la modulació, on un BER més baix indica una transmissió més fiable.

Codi:

1. 4-QAM

```
function [ber, numBits] = simula_qam1(EbNo, maxNumErrs, maxNumBits)


%Funció i Sortida
%   La línia de dalt defineix una funció que es diu simula_qam1 que simula
%   una transmissió de 4-QAM (o QPSK) a un canal AWGN
%   Entrades:
%       EbNo: relació d'energia per bit a densitat de soroll (en dB)
%       maxNumErrs: nombre màxim d'errors a acumular abans de parar la
%       simulació
%       maxNumBits: nombre màxim de bits a simular.
%   Sortides:
%       ber: taxa d'error de bits (Bit Error Rate)
%       numBits: nombre total de bits processats a la simulació
```

```
%Verificació del nombre d'arguments:
%   Aquesta instrucció comprova que la funció es truqui amb exactament 3
%   arguments. Si no és així, es llença un error.
narginchk(3,3)



%Ús de funcions extrínseques:
%   Això indica a MATLAB Coder que la funció isBERToolSimulationStopped no
%   es compilarà (es tractarà com una funció externa).
%   En simulacions amb BERTool es permet que l'usuari interrompi la
%   simulació, i aquesta funció comprova aquesta condició.
coder.extrinsic('isBERToolSimulationStopped')

%Inicialització de variables de control
%   S'inicialitzen a zero les variables:
%       totErr: acumulador del nombre total d'errors detectats
%       numBits: acumulador del nombre total de bits simulats
%   A simulacions Monte Carlo es necessiten comptadors per determinar quan
%   s'han aconseguit els llindars preestablerts
totErr  = 0; % Number of errors observed
numBits = 0; % Number of bits processed



%Mapeig de símbols a bits
%   Es defineix l'assignació de cada símbol de la constel·lació a una
%   seqüència de bits
```

```
    %   Es fa servir codi Gray, de manera que els símbols veïns difereixin en
    %   un sol bit, minimitzant l'error en cas de desviament
    %   Cada fila correspon a un símbol al mateix ordre que en 'constel_symb'

    constelBits = ['00';   % 1+1i
                   '01';   % -1+1i
                   '11';   % -1-1i
                   '10'];  % 1-1i


    %Definició de la constel·lació 4-QAM (QPSK) normalitzada:
    %   Es defineixen els 4 símbols complexos que representen la constel·lació
de
    %   QPSK
    %   Es multiplica per 1/2 per normalitzar la potència mitjana a 1
    %   La normalització és important perquè la potència del senyal sigui
    %   consistent i es pugui relacionar correctament amb la potència del
    %   soroll (Pn). A QPSK, els símbols sense normalitzar tenen magnitud.
    %   2^(1/2) normalitzar-los assegura que E[|s|^2]=1
    constelSymb = (1/sqrt(2)) * [ 1+1i; -1+1i; -1-1i; 1-1i ];


    %Número de símbols
    %   Es calcula cantidadSimbolos, que és el nombre de punts de la
constel·lació
    %   En aquest cas cantidadSimbolos=4
    cantidadSimbolos = length(constelSymb);
```

```
    %Bits per símbol:
    %   Es calcula el nombre de bits que es poden representar amb cada símbol,
    %   fent servir la fórmula bitsSimbolo=log2(cantidadSimbolos)
    %   Per cantidadSimbolos=4, bitsSimbolo=2
    %   Aquest paràmetre és clau per determinar l'eficiència de la modulació
    %   (més bits per símbol implica major eficiència espectral, però també
major
    %   susceptibilitat al soroll)
    bitsSimbolo = log2(cantidadSimbolos);


    %Número de bits per bloc
    %   Defineix quants bits se simularan a cada iteració del bucle (bloc de
    %   simulació)
    %   Aquí se simulen 10000 símbols, i per això el nombre de bits en cada bloc
    %   serà de 10000*k (on k = 2)
    nbitsBloc = 10000 * bitsSimbolo;



    %Potència del senyal
    %   Es calcula la potència mitjana del senyal transmès Ps
    %   S'utilitza la mitja del quadrat del valor absolut de cada símbol
    %   La potència del senyal és fonamental per determinar el nivell de soroll
    %   necessari en funció de la relació Eb/N0
    Ps = mean( abs(constelSymb) .^ 2);
```

```
%Conversió de dB a lineal
%   Converteix el valor de Eb/N0 de decibels (dB) al seu valor lineal
%   mitjançant la fórmula 10^(EbNo/10)
pRuidoEbNo = 10 ^ (EbNo / 10);


%Càlcul de la potència del soroll
%   Es calcula Pn, la potència del soroll, fent servir la relació:
%   Pn=(Ps)/(k*(Eb/E0))
%   Donat que l'energia per bit Eb es pot obtenir com (considerant el Ts=1)
%   Eb=Ps/k
%   i sabent que Eb/N0 no és la raó de senyal a soroll, es pot deduir que:
%   N0=Eb/(Eb/N0)
%   En aquesta simulació, es fa servir Pn com una aproximació del soroll
%   que s'afegeix al senyal
Pn = Ps / (pRuidoEbNo * bitsSimbolo);


%Longitud del bloc de símbols
%   Es defineix numSymb com el nombre de símbols que se simularan a cada
iteració
%   del bucle
numSymb = 10000;


%Bucle de simulació
%   S'entra en un bucle que continuarà simulant blocs de transmissió fins
%   que:
```

```
%       - S'hagin acumulat almenys maxNumErrs errors
%       - S'hagin transmès almenys maxNumBits
%   Aquest mètode de parada (criteri d'error o de bits) és típic a
%   simulacions Monte Carlo per garantir resultats estadísticament
%   significatius sense excedir temps de processament excessius
while((totErr < maxNumErrs) && (numBits < maxNumBits))

    % Check if the user clicked the Stop button of BERTool.
    % ==== DO NOT MODIFY ====

    %Detecció de parada manual
    %   Es comprova si l'usuari ha sol·licitat detenir la simulació
    %   Si és així surt del bucle
    if isBERToolSimulationStopped()
        break
    end
    % ==== END of DO NOT MODIFY ====

    % --- Proceed with simulation.
    % --- Be sure to update totErr and numBits.
    % --- INSERT YOUR CODE HERE.

    %Generació de símbols a transmetre
    %   Es genera un vector de numSymb nombres sencers aleatoris entre 1 i
cantidadSimbolos
    %   Cada nombre representa l'índex d'un símbol de la constel·lació
```

```
        %   La generació aleatòria assegura que cada símbol es transmeti amb
        %   una igual probabilitat, simulant una font d'informació equiprobable
        txSymb = randi([1 cantidadSimbolos], 1, numSymb);      %vector de los
indices de los simbolos que queremos enviar


        %Mapeig d'índex a símbols
        %   S'utilitza el vector 'txSymb' per seleccionar els símbols
        %   corresponents de 'contel_symb'
        %   Això genera el vector del senyal transmès, on cada posició conté
        %   un valor complex de la constel·lació
        txSig = constelSymb(txSymb);



        %Generació del soroll AWGN
        %   Es genera un vector de soroll complex
        %   randn(1, numSymb) produeix numSymb mostres de soroll gaussià real
amb mitja 0 i
        %   variància 1
        %   Es genera soroll per les parts real i imaginària, i es multiplica
        %   per (PN/2)^(1/2) per ajustar la variància de cada component
        %   En un canal AWGN, el soroll es complex, i la potència total es
        %   reparteix equitativament entre la part real i la part imaginària
        Soroll = sqrt(Pn / 2) * (randn( 1, numSymb) + 1i * randn( 1, numSymb));


        %Senyal rebut
        %   El senyal rebut rxSig és la suma del senyal transmès (convertida en
        %   vector columna amb la transposició ') i el soroll generat
```

```
        %   Això simula el pas del senyal per un canal AWGN, on s'afegeix
        %   soroll blanc gaussià al senyal transmès
        rxSig =  txSig.' + Soroll;

        %Demodulació i càlcul d'errors
        %   Es truca a la funció demodqam per demodular el senyal rebut
        %   La funció compara el senal rebut rxSig amb la constel·lació definida
        %   (constel_symb) i el mapeig de bits (constel_bits)
        %   Es calcula el nombre d'errors (nerrors) comparant els bits del
        %   símbol transmés (fent servir txSymb) amb els bits detectats
        %   La demodulació per mínima distància (o detector de màxima
        %   versemblança) és el mètode que es fa servir per decidir quin va ser
        %   el símbol transmès a partir del senyal amb soroll

        [detSym_idx, nerrors] = demodqam(rxSig, constelSymb, constelBits,
txSymb);

        %Acumulació de bits transmesos
        %   S'incrementa el comptador total de bits simulats a la quantitat
        %   corresponent al bloc actual 'nbitsBloc'
        %   Això permet calcular la taxa d'error com la relació entre errors i
        %   bits totals processats
        %   Això permet calcular la taxa d'error com la relació entre errors i
        %   bits totals processats
        numBits = numBits + nbitsBloc;
```

```
        %Acumulació d'errors
        %   S'actualitza el comptador total d'errors sumant els errors
        %   detectats en aquest bloc
        totErr = totErr + nerrors;

    end

    %Càlcul final del BER
    %   Es calcula la taxa d'error de bits (BER) dividint el número total
    %   d'errors acumulats entre el nombre total de bits simulats
    %   El BER és una mesura fonamental a comunicacions digitals, indicant la
    %   fracció de bits erronis rebuts. Un BER menor indica un sistema més.
    %   robust enfront del soroll
    ber = totErr/numBits;
```

2. 8-PSK

```
function [ber, numBits] = simula_qam2(EbNo, maxNumErrs, maxNumBits)
  %Funció i Sortida
  %   La línia de dalt defineix una funció que es diu simula_qam2 que simula
  %   una transmissió de 8-PSK rectangular a un canal AWGN
  %   Entrades:
```

```
%       EbNo: relació d'energia per bit a densitat de soroll (en dB)
%       maxNumErrs: nombre màxim d'errors a acumular abans de parar la
%       simulació
%       maxNumBits: nombre màxim de bits a simular.
%   Sortides:
%       ber: taxa d'error de bits (Bit Error Rate)
%       numBits: nombre total de bits processats a la simulació
 %Verificació del nombre d'arguments:
%   Aquesta instrucció comprova que la funció es truqui amb exactament 3
%   arguments. Si no és així, es llença un error.
narginchk(3,3)
 %Ús de funcions extrínseques:
%   Això indica a MATLAB Coder que la funció isBERToolSimulationStopped no
%   es compilarà (es tractarà com una funció externa).
%   En simulacions amb BERTool es permet que l'usuari interrompi la
%   simulació, i aquesta funció comprova aquesta condició.

coder.extrinsic('isBERToolSimulationStopped')
%Inicialització de variables de control
%   S'inicialitzen a zero les variables:
%       totErr: acumulador del nombre total d'errors detectats
%       numBits: acumulador del nombre total de bits simulats
%   A simulacions Monte Carlo es necessiten comptadors per determinar quan
%   s'han aconseguit els llindars preestablerts
totErr  = 0;
numBits = 0;
```

```
  %Mapeig de símbols a bits
  %   Es defineix l'assignació de cada símbol de la constel·lació a una
  %   seqüència de bits
  %   Es fa servir codi Gray, de manera que els símbols veïns difereixin en
  %   un sol bit, minimitzant l'error en cas de desviament
  %   Cada fila correspon a un símbol al mateix ordre que en 'constel_symb'
  constelBits = ['000';'100';'110';'010';'011';'111';'101';'001'];
  %Definició de la constel·lació 8-PSK rectangular:
  %   Es defineixen els 8 símbols complexos que representen la constel·lació de
  %   8-PSK rectangular.
  %   A diferència d'una 8-PSK clàssica (circular), en aquest cas la
constel·lació
  %   té punts distribuïts rectangularment

  constelSymb = [1i; (1 + 1i) * 1/sqrt(2); 1; (1 - 1i) * 1/sqrt(2); -1i; (-1 -
1i)* 1/sqrt(2); -1; (-1 + 1i)*1/sqrt(2)];

  %Número de símbols
  %   Es calcula cantidadSimbolos, que és el nombre de punts de la
constel·lació
  %   En aquest cas cantidadSimbolos=8
  cantidadSimbolos = length(constelSymb);
   %Bits per símbol:
   %   Es calcula el nombre de bits que es poden representar amb cada símbol,
   %   fent servir la fórmula bitsSimbolo=log2(cantidadSimbolos)
   %   Per cantidadSimbolos=8, bitsSimbolo=3
```

```
    %   Aquest paràmetre és clau per determinar l'eficiència de la modulació
    %   (més bits per símbol implica major eficiència espectral, però també
major
    %   susceptibilitat al soroll)
  bitsSimbolo = log2(cantidadSimbolos);


  %Número de bits per bloc
  %   Defineix quants bits se simularan a cada iteració del bucle (bloc de
  %   simulació)
  %   Aquí se simulen 10000 símbols, i per això el nombre de bits en cada bloc
  %   serà de 10000 * bitsSimbolo (on bitsSimbolo = 3)
  nbitsBloc = 10000 * bitsSimbolo;
  %Potència del senyal
  %   Es calcula la potència mitjana del senyal transmès Ps
  %   S'utilitza la mitja del quadrat del valor absolut de cada símbol
  %   La potència del senyal és fonamental per determinar el nivell de soroll
  %   necessari en funció de la relació Eb/N0
  Ps = mean( abs( constelSymb) .^ 2);
  %Conversió de dB a lineal
  %   Converteix el valor de Eb/N0 de decibels (dB) al seu valor lineal
  %   mitjançant la fórmula 10^(EbNo/10)
  pRuidoEbNo = 10 ^ (EbNo / 10);
  %Càlcul de la potència del soroll
  %   Es calcula Pn, la potència del soroll, fent servir la relació:
  %   Pn=(Ps)/(bitsSimbolo*(Eb/E0))
  %   Donat que l'energia per bit Eb es pot obtenir com (considerant el Ts=1)
```

```
  %   Eb=Ps/bitsSimbolo
  %   i sabent que Eb/N0 no és la raó de senyal a soroll, es pot deduir que:
  %   N0=Eb/(Eb/N0)
  %   En aquesta simulació, es fa servir Pn com una aproximació del soroll
  %   que s'afegeix al senyal
  Pn = Ps/(pRuidoEbNo * bitsSimbolo);
  %Longitud del bloc de símbols
  %   Es defineix numSymb com el nombre de símbols que se simularan a cada
iteració
  %   del bucle
  numSymb = 10000;
   %Bucle de simulació
  %   S'entra en un bucle que continuarà simulant blocs de transmissió fins
  %   que:
  %       - S'hagin acumulat almenys maxNumErrs errors
  %       - S'hagin transmès almenys maxNumBits
  %   Aquest mètode de parada (criteri d'error o de bits) és típic a
  %   simulacions Monte Carlo per garantir resultats estadísticament
  %   significatius sense excedir temps de processament excessius
  while((totErr < maxNumErrs) && (numBits < maxNumBits))
       %Detecció de parada manual
      %   Es comprova si l'usuari ha sol·licitat detenir la simulació
      %   Si és així surt del bucle
      if isBERToolSimulationStopped()
          break
      end
```

```
        %Generació de símbols a transmetre
      %   Es genera un vector de numSymb nombres sencers aleatoris entre 1 i
cantidadSimbolos
      %   Cada nombre representa l'índex d'un símbol de la constel·lació
      %   La generació aleatòria assegura que cada símbol es transmeti amb
      %   una igual probabilitat, simulant una font d'informació equiprobable
      txSymb = randi([1 cantidadSimbolos],1,numSymb);
        %Mapeig d'índex a símbols
      %   S'utilitza el vector 'txSymb' per seleccionar els símbols
      %   corresponents de 'contel_symb'
      %   Això genera el vector del senyal transmès, on cada posició conté
      %   un valor complex de la constel·lació
      txSig = constelSymb(txSymb);
        %Generació del soroll AWGN
      %   Es genera un vector de soroll complex
      %   randn(1, numSymb) produeix numSymb mostres de soroll gaussià real amb
mitja 0 i
      %   variància 1
      %   Es genera soroll per les parts real i imaginària, i es multiplica
      %   per (PN/2)^(1/2) per ajustar la variància de cada component
      %   En un canal AWGN, el soroll es complex, i la potència total es
      %   reparteix equitativament entre la part real i la part imaginària
      Soroll = sqrt(Pn / 2) * (randn( 1, numSymb) + 1i * randn( 1, numSymb));
        %Senyal rebut
      %   El senyal rebut rxSig és la suma del senyal transmès (convertida en
      %   vector columna amb la transposició ') i el soroll generat
      %   Això simula el pas del senyal per un canal AWGN, on s'afegeix
```

```
    %   soroll blanc gaussià al senyal transmès
    rxSig =  txSig.' + Soroll;
     %Demodulació i càlcul d'errors
    %   Es truca a la funció demodqam per demodular el senyal rebut
    %   La funció compara el senal rebut rxSig amb la constel·lació definida
    %   (constel_symb) i el mapeig de bits (constel_bits)
    %   Es calcula el nombre d'errors (nerrors) comparant els bits del
    %   símbol transmés (fent servir txSymb) amb els bits detectats
    %   La demodulació per mínima distància (o detector de màxima
    %   versemblança) és el mètode que es fa servir per decidir quin va ser
    %   el símbol transmès a partir del senyal amb soroll
    [~, nerrors] = demodqam(rxSig, constelSymb, constelBits, txSymb);
    %Acumulació de bits transmesos
    %   S'incrementa el comptador total de bits simulats a la quantitat
    %   corresponent al bloc actual 'nbitsBloc'
    %   Això permet calcular la taxa d'error com la relació entre errors i
    %   bits totals processats
    %   Això permet calcular la taxa d'error com la relació entre errors i
    %   bits totals processatsdd
    numBits = numBits + nbitsBloc;
     %Acumulació d'errors
    %   S'actualitza el comptador total d'errors sumant els errors
    %   detectats en aquest bloc
    totErr = totErr + nerrors;
end
%Càlcul final del BER
```

```
  %   Es calcula la taxa d'error de bits (BER) dividint el número total
  %   d'errors acumulats entre el nombre total de bits simulats
  %   El BER és una mesura fonamental a comunicacions digitals, indicant la
  %   fracció de bits erronis rebuts. Un BER menor indica un sistema més.
  %   robust enfront del soroll
  ber = totErr/numBits;
```

3. 16-QAM

```
function [ber, numBits] = simula_qam3(EbNo, maxNumErrs, maxNumBits)
  %Funció i Sortida
  %   La línia de dalt defineix una funció que es diu simula_qam3 que simula
  %   una transmissió de 16-QAM a un canal AWGN
  %   Entrades:
  %       EbNo: relació d'energia per bit a densitat de soroll (en dB)
  %       maxNumErrs: nombre màxim d'errors a acumular abans de parar la
  %       simulació
  %       maxNumBits: nombre màxim de bits a simular.
  %   Sortides:
  %       ber: taxa d'error de bits (Bit Error Rate)
  %       numBits: nombre total de bits processats a la simulació
   %Verificació del nombre d'arguments:
  %   Aquesta instrucció comprova que la funció es truqui amb exactament 3
  %   arguments. Si no és així, es llença un error.
```

```
  narginchk(3,3)
   %Ús de funcions extrínseques:
  %   Això indica a MATLAB Coder que la funció isBERToolSimulationStopped no
  %   es compilarà (es tractarà com una funció externa).
  %   En simulacions amb BERTool es permet que l'usuari interrompi la
  %   simulació, i aquesta funció comprova aquesta condició.

  coder.extrinsic('isBERToolSimulationStopped')
  %Inicialització de variables de control
  %   S'inicialitzen a zero les variables:
  %       totErr: acumulador del nombre total d'errors detectats
  %       numBits: acumulador del nombre total de bits simulats
  %   A simulacions Monte Carlo es necessiten comptadors per determinar quan
  %   s'han aconseguit els llindars preestablerts
  totErr  = 0;
  numBits = 0;
  %Definició de la constel·lació 16-QAM normalitzada:
  %   Es defineixen els 16 símbols complexos que representen la constel·lació
de
  %   16-QAM amb amplitud de +/-1 i +/- 3.
  %   Es multiplica per 1/((10)^(1/2)) per normalitzar la potència mitjana
  %   a 1, ja que la potència mitjana sería 10
  %   La normalització és important perquè la potència del senyal sigui
  %   consistent i es pugui relacionar correctament amb la potència del
  %   soroll (Pn).
  %   En 16.QAM la eficiència espectral es major ja que k=4, pero
```

```
  %   requereix mes Eb/N0 per aconsseguir la mateixa BER que modulacions
  %   d'ordre inferior
  constelBits = ['0000';
                 '0100';
                 '1000';
                 '1100';
                 '0001';
                 '0101';
                 '1001';
                 '1101';
                 '0011';
                 '0111';
                 '1011';
                 '1111';
                 '0010';
                 '0110';
                 '1010';
                 '1110'];
  %Definició de la constel·lació 16-QAM normalitzada:
  %   Es defineixen els 16 símbols complexos que representen la constel·lació
de
  %   16-QAM amb amplitud de +/-1 i +/- 3.
  %   Es multiplica per 1/((10)^(1/2)) per normalitzar la potència mitjana
  %   a 1, ja que la potència mitjana sería 10
  %   La normalització és important perquè la potència del senyal sigui
  %   consistent i es pugui relacionar correctament amb la potència del
```

```
  %   soroll (Pn).
  %   En 16.QAM la eficiència espectral es major ja que k=4, pero
  %   requereix mes Eb/N0 per aconsseguir la mateixa BER que modulacions
  %   d'ordre inferior

  constelSymb = (1/sqrt(10)) * [ -3+3i; -1+3i; 3+3i; 1+3i; ...
                                 -3+1i; -1+1i; 3+1i; 1+1i; ...
                                 -3-1i; -1-1i; 3-1i; 1-1i; ...
                                 -3-3i; -1-3i; 3-3i; 1-3i ];


  %Número de símbols
  %   Es calcula cantidadSimbolos, que és el nombre de punts de la
constel·lació
  %   En aquest cas cantidadSimbolos=8
  cantidadSimbolos = length(constelSymb);
%Bits per símbol:
%   Es calcula el nombre de bits que es poden representar amb cada símbol,
%   fent servir la fórmula bitsSimbolo=log2(cantidadSimbolos)
%   Per cantidadSimbolos=8, bitsSimbolo=3
%   Aquest paràmetre és clau per determinar l'eficiència de la modulació
%   (més bits per símbol implica major eficiència espectral, però també major
%   susceptibilitat al soroll)
  bitsSimbolo = log2(cantidadSimbolos);


  %Número de bits per bloc
  %   Defineix quants bits se simularan a cada iteració del bucle (bloc de
```

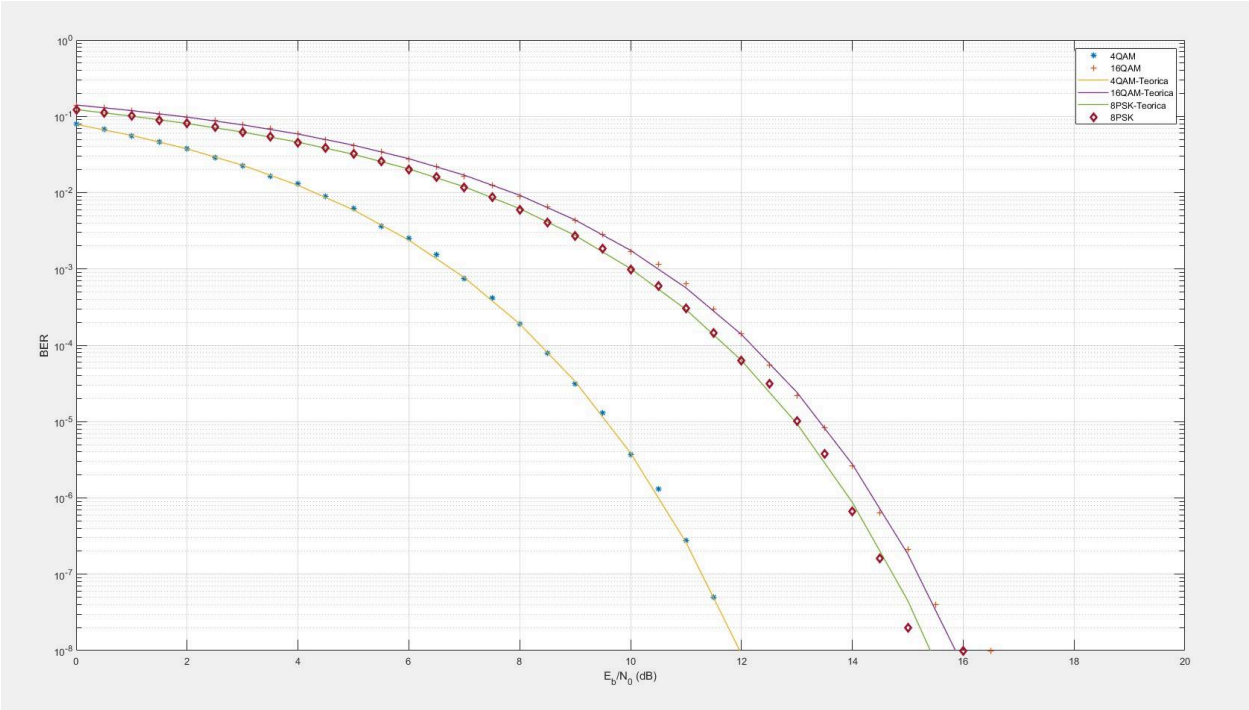
```
%   simulació)
%   Aquí se simulen 10000 símbols, i per això el nombre de bits en cada bloc
%   serà de 10000 * bitsSimbolo (on bitsSimbolo = 3)
nbitsBloc = 10000 * bitsSimbolo;
%Potència del senyal
%   Es calcula la potència mitjana del senyal transmès Ps
%   S'utilitza la mitja del quadrat del valor absolut de cada símbol
%   La potència del senyal és fonamental per determinar el nivell de soroll
%   necessari en funció de la relació Eb/N0
Ps = mean( abs( constelSymb) .^ 2);
%Conversió de dB a lineal
%   Converteix el valor de Eb/N0 de decibels (dB) al seu valor lineal
%   mitjançant la fórmula 10^(EbNo/10)
pRuidoEbNo = 10 ^ (EbNo / 10);
%Càlcul de la potència del soroll
%   Es calcula Pn, la potència del soroll, fent servir la relació:
%   Pn=(Ps)/(bitsSimbolo*(Eb/E0))
%   Donat que l'energia per bit Eb es pot obtenir com (considerant el Ts=1)
%   Eb=Ps/bitsSimbolo
%   i sabent que Eb/N0 no és la raó de senyal a soroll, es pot deduir que:
%   N0=Eb/(Eb/N0)
%   En aquesta simulació, es fa servir Pn com una aproximació del soroll
%   que s'afegeix al senyal
Pn = Ps/(pRuidoEbNo * bitsSimbolo);
%Longitud del bloc de símbols
```

```
  %   Es defineix numSymb com el nombre de símbols que se simularan a cada
iteració
  %   del bucle
  numSymb = 10000;
   %Bucle de simulació
  %   S'entra en un bucle que continuarà simulant blocs de transmissió fins
  %   que:
  %       - S'hagin acumulat almenys maxNumErrs errors
  %       - S'hagin transmès almenys maxNumBits
  %   Aquest mètode de parada (criteri d'error o de bits) és típic a
  %   simulacions Monte Carlo per garantir resultats estadísticament
  %   significatius sense excedir temps de processament excessius
  while((totErr < maxNumErrs) && (numBits < maxNumBits))
       %Detecció de parada manual
      %   Es comprova si l'usuari ha sol·licitat detenir la simulació
      %   Si és així surt del bucle
      if isBERToolSimulationStopped()
          break
      end
       %Generació de símbols a transmetre
      %   Es genera un vector de numSymb nombres sencers aleatoris entre 1 i
cantidadSimbolos
      %   Cada nombre representa l'índex d'un símbol de la constel·lació
      %   La generació aleatòria assegura que cada símbol es transmeti amb
      %   una igual probabilitat, simulant una font d'informació equiprobable
      txSymb = randi([1 cantidadSimbolos],1,numSymb);
       %Mapeig d'índex a símbols
```

```
%   S'utilitza el vector 'txSymb' per seleccionar els símbols
%   corresponents de 'contel_symb'
%   Això genera el vector del senyal transmès, on cada posició conté
%   un valor complex de la constel·lació
txSig = constelSymb(txSymb);
 %Generació del soroll AWGN
%   Es genera un vector de soroll complex
%   randn(1, numSymb) produeix numSymb mostres de soroll gaussià real amb
mitja 0 i
%   variància 1
%   Es genera soroll per les parts real i imaginària, i es multiplica
%   per (PN/2)^(1/2) per ajustar la variància de cada component
%   En un canal AWGN, el soroll es complex, i la potència total es
%   reparteix equitativament entre la part real i la part imaginària
Soroll = sqrt(Pn / 2) * (randn( 1, numSymb) + 1i * randn( 1, numSymb));
 %Senyal rebut
%   El senyal rebut rxSig és la suma del senyal transmès (convertida en
%   vector columna amb la transposició ') i el soroll generat
%   Això simula el pas del senyal per un canal AWGN, on s'afegeix
%   soroll blanc gaussià al senyal transmès
rxSig =  txSig.' + Soroll;
 %Demodulació i càlcul d'errors
%   Es truca a la funció demodqam per demodular el senyal rebut
%   La funció compara el senal rebut rxSig amb la constel·lació definida
%   (constel_symb) i el mapeig de bits (constel_bits)
%   Es calcula el nombre d'errors (nerrors) comparant els bits del
```

```
    %   símbol transmés (fent servir txSymb) amb els bits detectats
    %   La demodulació per mínima distància (o detector de màxima
    %   versemblança) és el mètode que es fa servir per decidir quin va ser
    %   el símbol transmès a partir del senyal amb soroll
    [~, nerrors] = demodqam(rxSig, constelSymb, constelBits, txSymb);
    %Acumulació de bits transmesos
    %   S'incrementa el comptador total de bits simulats a la quantitat
    %   corresponent al bloc actual 'nbitsBloc'
    %   Això permet calcular la taxa d'error com la relació entre errors i
    %   bits totals processats
    %   Això permet calcular la taxa d'error com la relació entre errors i
    %   bits totals processatsdd
    numBits = numBits + nbitsBloc;
     %Acumulació d'errors
    %   S'actualitza el comptador total d'errors sumant els errors
    %   detectats en aquest bloc
    totErr = totErr + nerrors;
 end
 %Càlcul final del BER
%   Es calcula la taxa d'error de bits (BER) dividint el número total
%   d'errors acumulats entre el nombre total de bits simulats
%   El BER és una mesura fonamental a comunicacions digitals, indicant la
%   fracció de bits erronis rebuts. Un BER menor indica un sistema més.
%   robust enfront del soroll
ber = totErr/numBits;
```

Gràfiques

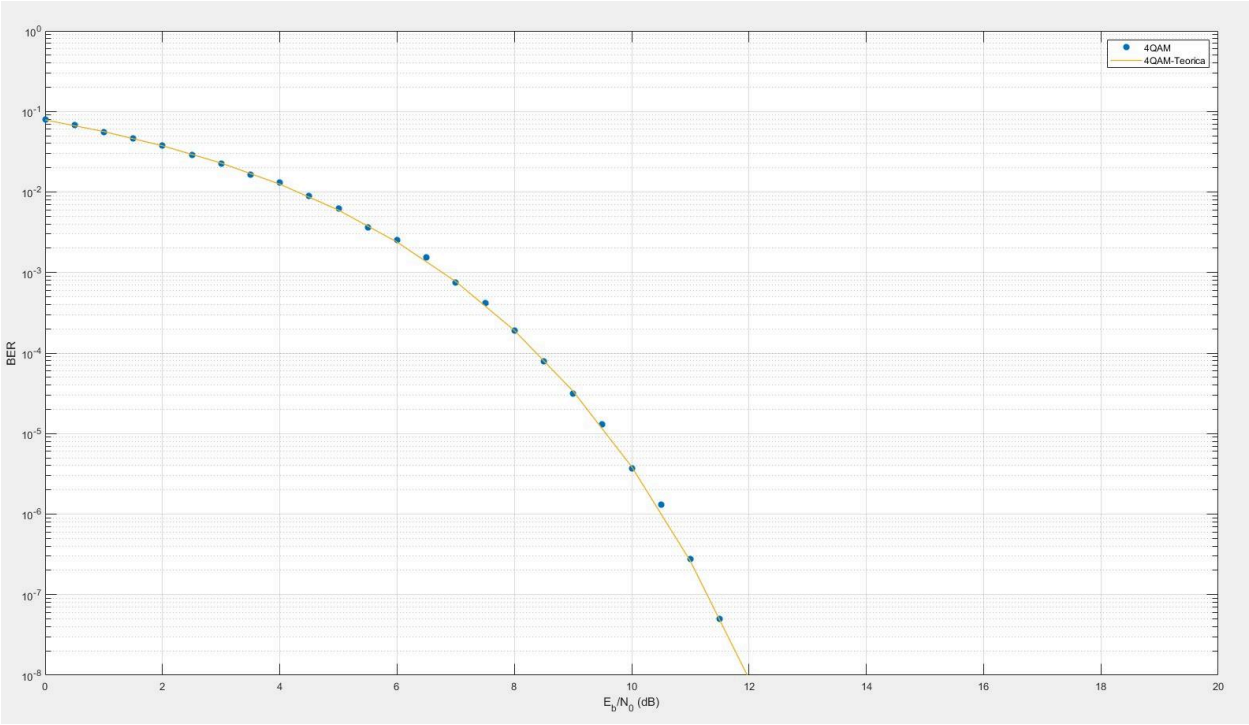

4QAM
16QAM
4QAM-Teorica
16QAM-Teorica
8PSK-Teorica
8PSK
BER
E_b/N_0 (dB)

Resultats

1. 4QAM

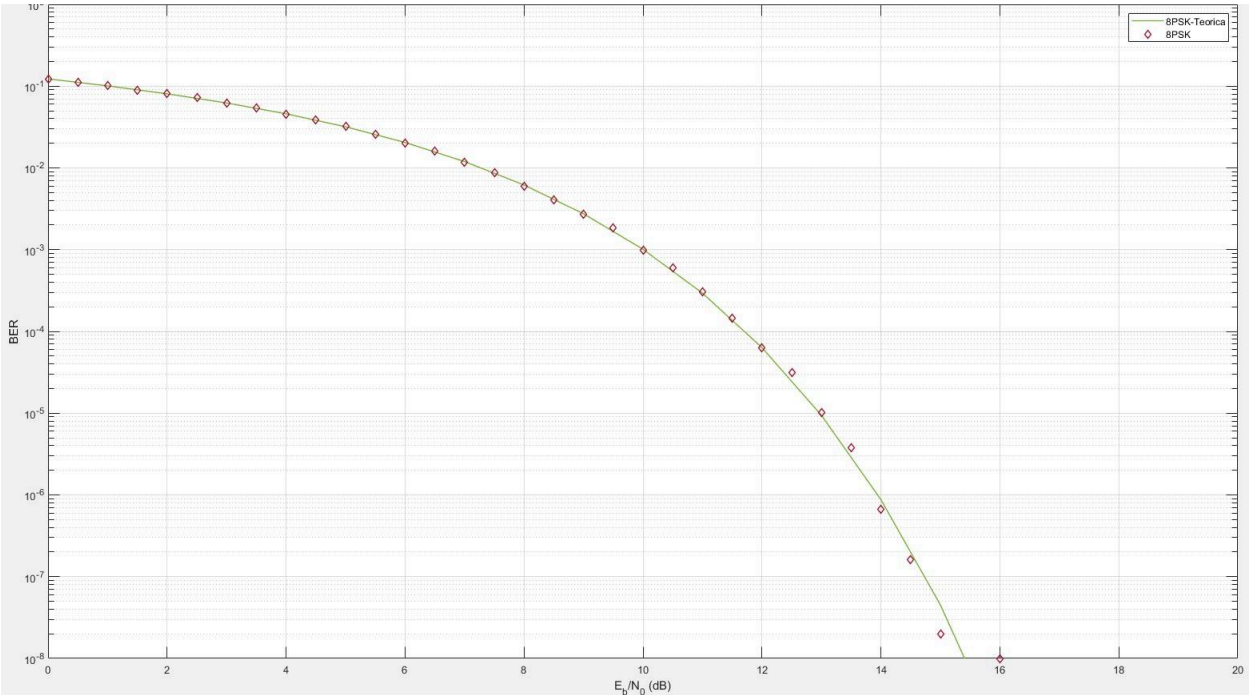
La simulació de Monte Carlo per a 4-QAM ha demostrat un ajustament molt proper a la corba teòrica. Això és degut al fet que 4-QAM (equivalent a QPSK) té una constel·lació amb símbols clarament diferenciats, cosa que redueix significativament la interferència entre ells en un canal AWGN.

S'ha observat que per valors baixos d'Eb/N0 (menys de 4 dB), la BER és relativament elevada, però decreix ràpidament a mesura que augmenta Eb/N0, ja que el soroll afegit té un impacte menor en la decisió del símbol quan la relació senyal/soroll millora. A valors baixos de Eb/N0, la simulació es deté ràpidament en assolir el límit d'errors establert, mentre que per valors més alts, s'arriba abans al límit de bits transmesos.



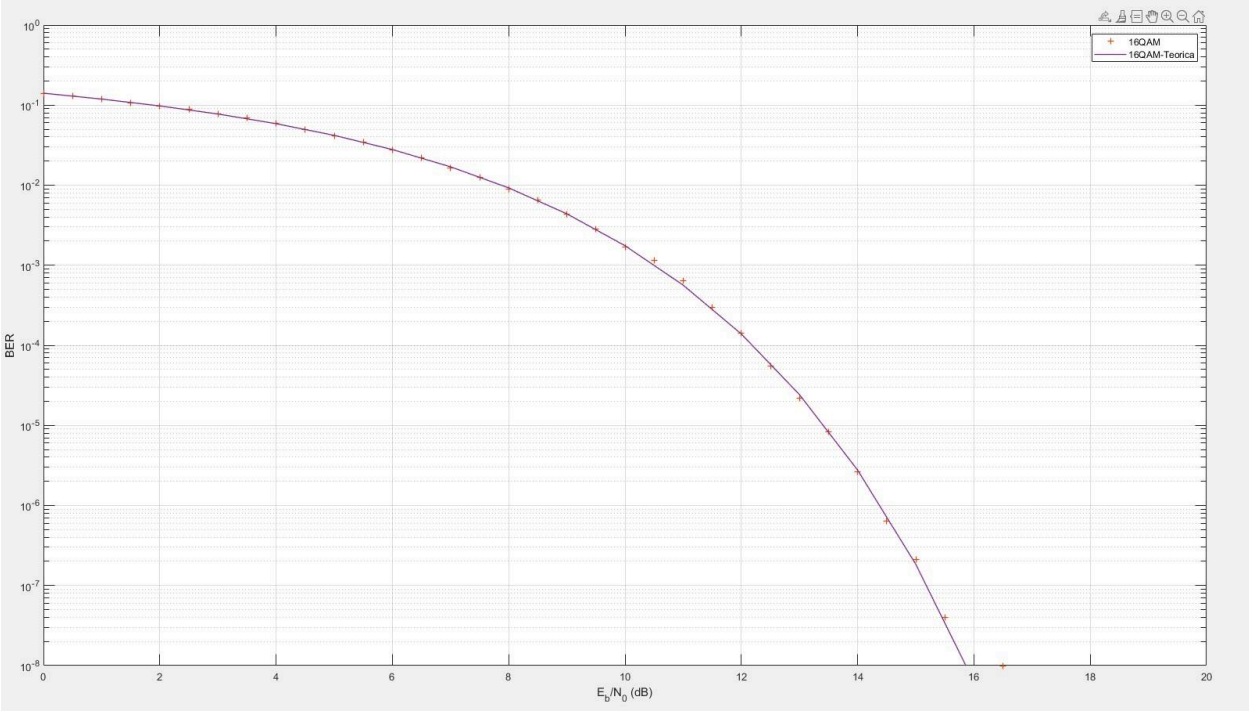
2. 8PSK

En el cas de la modulació 8-PSK, la BER resultant és superior a la de 4-QAM per un mateix Eb/N0. Això es deu a la menor separació angular entre símbols en la constel·lació, fent-los més vulnerables al soroll. Per valors baixos d'Eb/N0, la simulació s'atura ràpidament en assolir els 1000 errors, mentre que per valors alts de Eb/N0, es poden transmetre fins a 10^8 bits abans d'arribar a aquest límit d'errors.



3. 16QAM

La modulació 16-QAM es mostra més sensible al soroll en comparació amb 4-QAM i 8-PSK. Amb una constel·lació més densa, la distància entre símbols és menor, augmentant la probabilitat d'errors en presència de soroll. Els resultats indiquen que 16-QAM requereix valors més elevats de Eb/N0 per aconseguir una BER acceptable. A valors baixos d'Eb/N0, la simulació es deté ràpidament en assolir el límit d'errors, ja que els símbols es tornen més difícils de distingir.



Conclusions

L'anàlisi de les simulacions realitzades per a les modulacions 4-QAM, 8-PSK i 16-QAM ha permès comprendre millor el compromís entre eficiència espectral i robustesa enfront del soroll. Les dades obtingudes mostren clarament com l'augment del nombre de símbols en la constel·lació influeix en la taxa d'error de bits (BER), així com en els requisits d'energia per assegurar una comunicació fiable. A continuació, detallem les principals conclusions extretes de l'estudi:

- 4-QAM
  Aquesta modulació s'ha demostrat com la més robusta enfront del soroll, ja que presenta una BER més baixa amb valors relativament petits d'Eb/N0. Això es deu a la seva constel·lació simple, amb símbols clarament separats, fet que redueix la probabilitat d'error en la detecció.
  Avantatges:

    - Alta resistència al soroll, especialment en canals amb baixa relació senyal-soroll (SNR).
    - Requereix menys potència per assolir un rendiment acceptable.
    - Apropiada per aplicacions de comunicació sense fils, com Wi-Fi i LTE, on la fiabilitat és essencial.

      Desavantatges:
    - Baixa eficiència espectral, ja que només transmet 2 bits per símbol, fet que limita la velocitat de transmissió.
- 8-PSK

  La modulació 8-PSK millora l'eficiència espectral respecte a 4-QAM, ja que transmet 3 bits per símbol, però això comporta una BER més elevada per un mateix nivell de Eb/N0. A causa de la seva constel·lació circular, la separació angular entre símbols és menor, cosa que fa que siguin més vulnerables al soroll i a les distorsions del canal.

  Avantatges:

    - Major eficiència espectral en comparació amb 4-QAM.
    - Pot ser útil en aplicacions on cal augmentar el volum d'informació transmesa sense incrementar significativament l'ample de banda.

      Desavantatges:
    - Major probabilitat d'error en presència de soroll, ja que els símbols estan més junts en la constel·lació.
    - Requereix un processament més sofisticat per a una demodulació precisa en entorns sorollosos.
- 16-QAM

  La modulació 16-QAM ofereix una eficiència espectral molt més gran que les altres dues, ja que pot transmetre 4 bits per símbol. No obstant això, aquest avantatge té un cost: la seva vulnerabilitat al soroll és molt superior, ja que la constel·lació conté símbols molt més propers entre si, fent que sigui més fàcil cometre errors de detecció.

  Avantatges:

    - Excel·lent eficiència espectral, permetent transmetre més informació en el mateix ample de banda.
    - Útil en sistemes on la capacitat de dades és prioritària, com les xarxes 4G i 5G.

      Desavantatges:
    - Requereix una relació Eb/N0 molt més elevada per assolir una BER acceptable.
    - No és adequada per canals amb molt soroll o amb grans variacions de senyal, com en transmissions satel·litàries o en escenaris de comunicació mòbil en moviment.

L'elecció de la modulació òptima depèn del tipus de sistema de comunicació i de les condicions del canal. Si la prioritat és la robustesa i la fiabilitat, 4-QAM és la millor opció. Si es busca un compromís entre eficiència espectral i BER, 8-PSK pot ser una alternativa viable en entorns moderadament sorollosos. Finalment, si l'objectiu és maximitzar la velocitat de transmissió i es disposa d'un canal amb bona qualitat de senyal, 16-QAM ofereix el millor rendiment en termes d'eficiència espectral.