Práctica 2. BER en Turbocódigos y LDPC

1) Objetivo
    - Gráfica de error
    - Manejo de otros bloques funcionales en Matlab
        - Modulador BPSK
        - Canal AWGN
        - Demodulador BPSK
2) Diferencia con la práctica 1
    - Uso de modulador BPSK, canal AWGN y demodulador BPSK

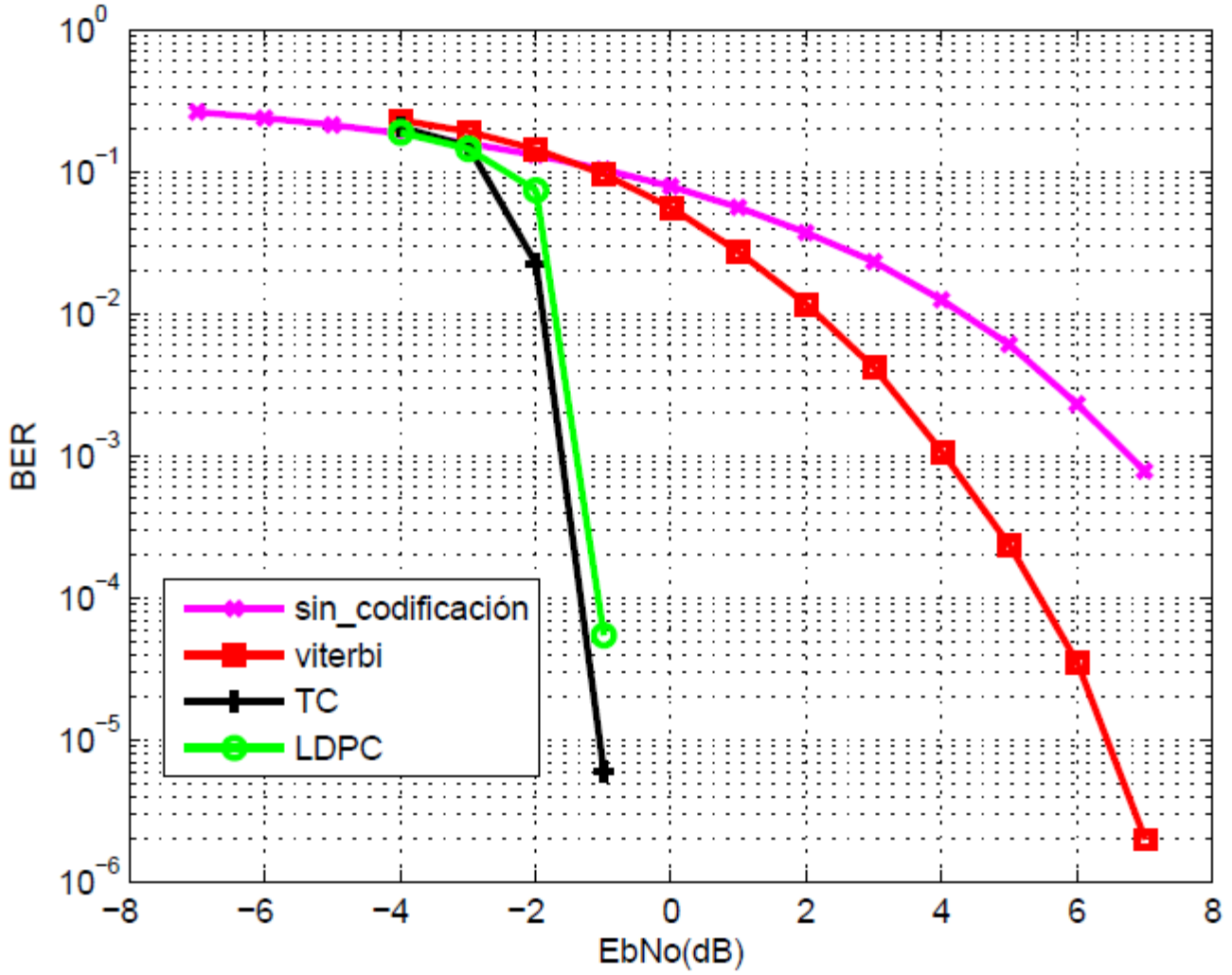
3) Modulador BPSK y Canal AWGN
4) Demodulador BPSK
5) Codificación y decodificación
        a) Sin codificación (de canal)
        b) Codificación convolucional
        c) Turbocódigos
        d) LDPC
6) BER
7) Otros

Práctica 2.
1) Objetivo: BER en Turbocódigos y LDPC

- Simulación de
  - Modulador BPSK
  - Canal AWGN
  - Demodulador BPSK

(en la práctica anterior
se simuló un Canal Binario
Simétrico)

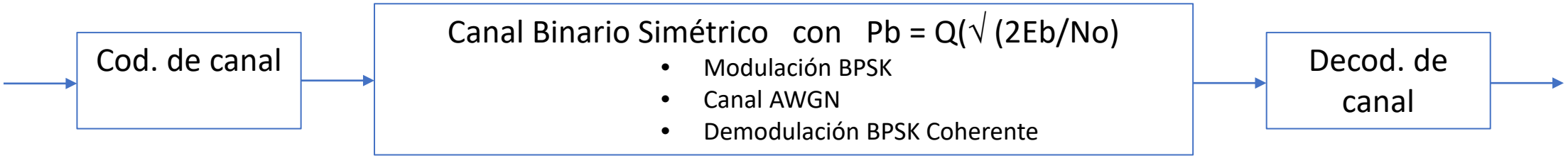
- Transmisión
  - Sin codificación
  - Codificación convolucional
  - Turbocódigos
  - LDPC



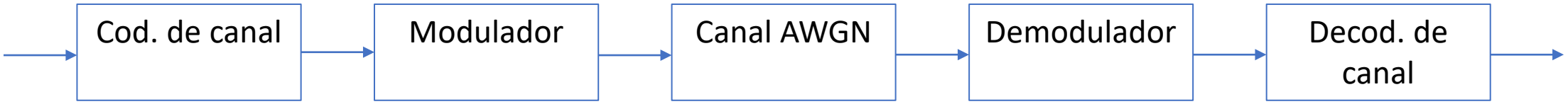
Práctica 2.
2) Diferencia con la práctica 1

Uso de modulador BPSK, canal AWGN y demodulador BPSK

- Práctica 1



- Práctica 2



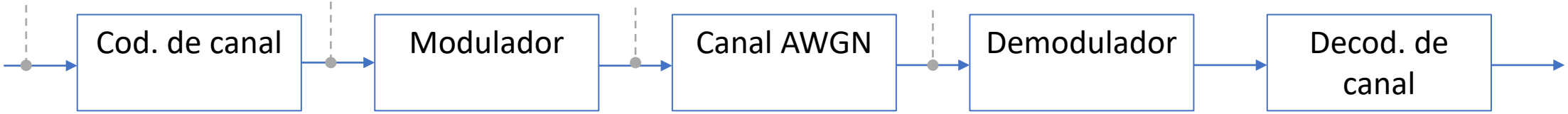
Práctica 2.
3) Modulador BPSK y Canal AWGN

mm: secuencia de mensajes
Son 0s y 1s

cc: secuencia de palabras código
Son 0s y 1s

coordtx: secuencia de coordenadas transmitidas
Son números complejos

coordrx: secuencia de coordenadas recibidas
Son números complejos



- Modulador

```
hMod=comm.PSKModulator(M,0,'BitInput',true,'SymbolMapping','Gray');
```

M=2 (BPSK)

- Canal

```
hcanal=comm.AWGNChannel('NoiseMethod','Signal to noise ratio (Eb/No)',
'EbNo',EbNo(i),'BitsPerSymbol',Ndbps);
```

Ndbps=1 (BPSK)

- Secuencia en Matlab

```
coordtx = step(hMod,cc);
hcanal=comm.AWGNChannel('NoiseMethod','Signal to noise ratio (Eb/No)',
'EbNo',EbNo(i),'BitsPerSymbol',1);
coordrx=step(hcanal,coordtx);
```

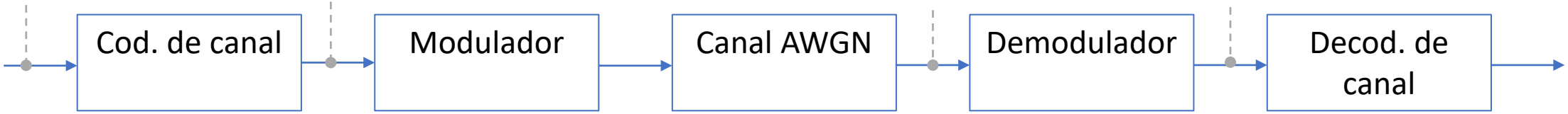
Práctica 2.
4) Demodulador BPSK

mm: secuencia de mensajes
Son 0s y 1s

cc: secuencia de palabras código
Son 0s y 1s

coordrx: secuencia de coordenadas recibidas
Son números complejos

rrdem: secuencia de palabras recibidas
0 y 1s ó LLRs!



- Demodulador
  - Para secuencias
    - Sin codificador de canal
    - Con codificación/decodificación convolucional

```
hDemod=comm.PSKDemodulator(M,0,'SymbolMapping','Gray','BitOutput',true, 'DecisionMethod','Hard decision');
```

  - Para secuencias
    - Con turbocódigos
    - Con LDPC

```
hDemod=comm.PSKDemodulator(M,0,'SymbolMapping','Gray','BitOutput',true, 'DecisionMethod','log-likelihood ratio');
```

- Secuencia en Matlab

```
rrdem=step(hDemod,coordrx);
```

Será un vector columna de 0s y 1s con la secuencia de palabras recibidas
Será un vector de LLRs de los 0s y 1s de la secuencia de palabras recibidas (si LLR < 0 → 1, si LLR > 0 → 0)

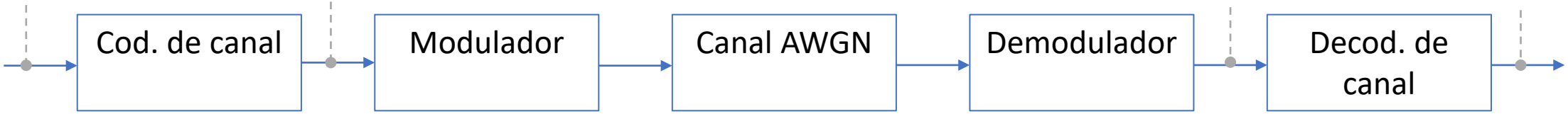
Práctica 2.
5) Codificación y decodificación

mm: secuencia de mensajes
Son 0s y 1s

cc: secuencia de palabras código
Son 0s y 1s

rrdem: secuencia de palabras recibidas
0 y 1s ó LLRs!

mDec: secuencia de mensajes decodificados
Son 0s y 1s o booleanos



Opciones

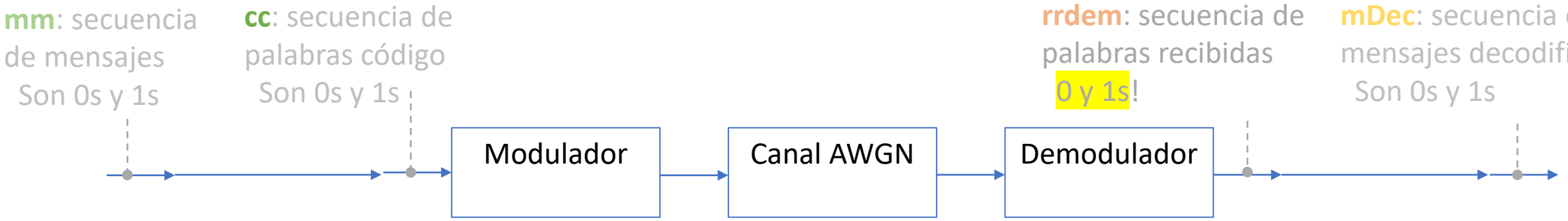
a) Sin codificación

b) Codificación convolucional

c) Turbocódigos

d) LDPC

Práctica 2.
5) Codificación y decodificación

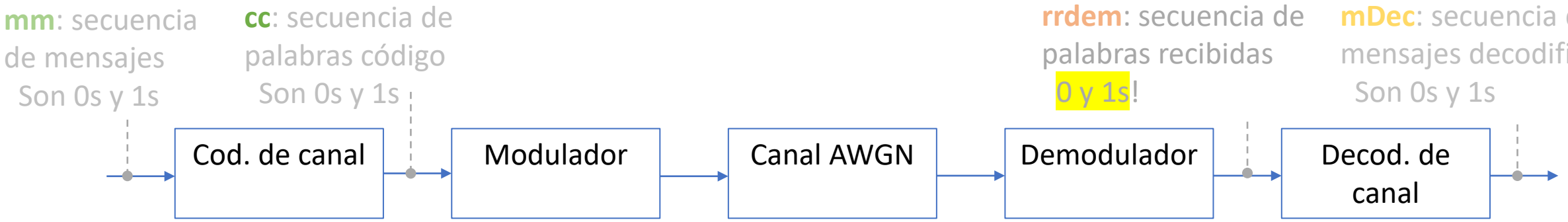


Opciones

a) Sin codificación (Demodulación Hard Decision)

```
k=1152;
...
mm              = randi ([0 1], k,1);
cc              = mm;
...
coordtx         = step(hMod,cc);
hcanal          = comm.AWGNChannel('NoiseMethod','Signal to noise ratio (Eb/No)',
'EbNo',EbNo(i),'BitsPerSymbol',1);
coordrx         = step(hcanal,coordtx);
Rrdem           = step(hDemod,coordrx);
mDec            = rrdem;
...
```

Práctica 2.
5) Codificación y decodificación



Opciones

b) Codificación convolucional (Demodulación Hard Decision)

```
tre  = poly2trellis(4,[13 15],13);
hEnc = comm.ConvolutionalEncoder(tre,'TerminationMethod','Terminated');
hDec = comm.ViterbiDecoder(tre,'InputFormat','hard','TracebackDepth',4,
           'TerminationMethod','Terminated');
```

NB: El codificador añade la cola, habrá que quitarla en la secuencia mDec

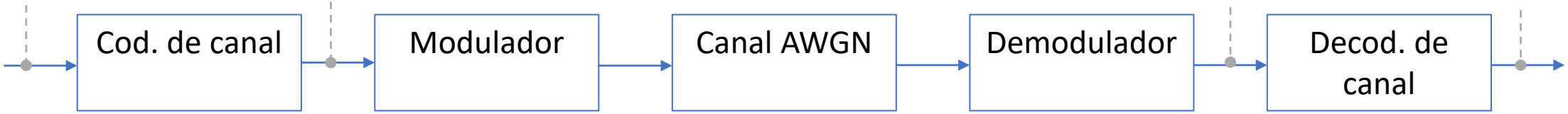
Práctica 2.
5) Codificación y decodificación

mm: secuencia de mensajes
Son 0s y 1s

cc: secuencia de palabras código
Son 0s y 1s

rrdem: secuencia de palabras recibidas
0 y 1s!

mDec: secuencia de mensajes decodificados
Son 0s y 1s



Opciones

b) Codificación convolucional (Demodulación Hard Decision)

```
k=1152;
...
mm              = randi ([0 1], k,1);
...
cc              = step(hEnc,mm);
...
coordtx         = step(hMod,cc);
hcanal          = comm.AWGNChannel('NoiseMethod','Signal to noise ratio (Eb/No)', 'EbNo',EbNo(i),'BitsPerSymbol',1);
coordrx         = step(hcanal,coordtx);
rrdem           = step(hDemod,coordrx);
mDec            = step(hDec,rrdem);
mDec            = mDec(1:end-3);

...
```

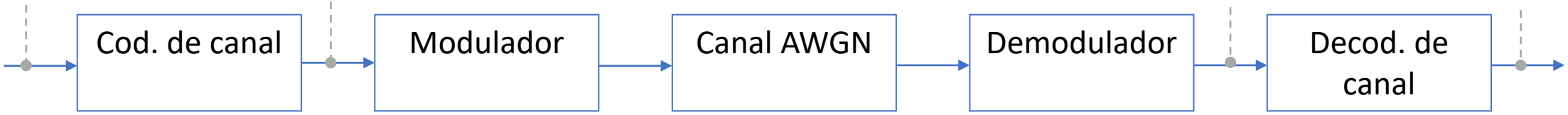
Práctica 2.
5) Codificación y decodificación

mm: secuencia de mensajes
Son 0s y 1s

cc: secuencia de palabras código
Son 0s y 1s

rrdem: secuencia de palabras recibidas
LLRs!

mDec: secuencia de mensajes decodificados
Son 0s y 1s



Opciones

c) Turbocódigos (Demodulación Log-likelihood ratio)

```
tre   = poly2trellis(4,[13 15],13);
hEnc1 = comm.ConvolutionalEncoder('TrellisStructure',tre,'TerminationMethod',
        'Terminated');
hEnc2 = comm.ConvolutionalEncoder('TrellisStructure',tre,'TerminationMethod',
        'Terminated');
hDec1 = comm.APPDecoder('TrellisStructure',tre,'TerminationMethod',
        'Terminated','Algorithm','True APP');
hDec2 = comm.APPDecoder('TrellisStructure',tre,'TerminationMethod',
        'Terminated','Algorithm','True APP');
```

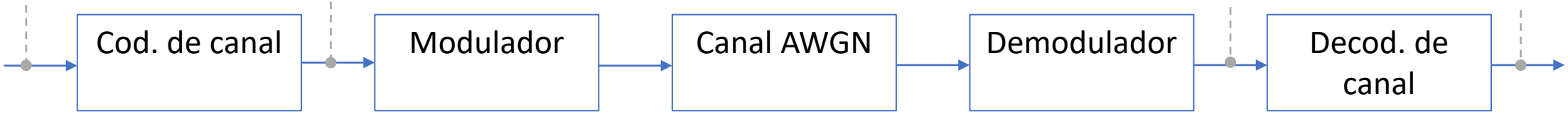
Práctica 2.
5) Codificación y decodificación

mm: secuencia de mensajes
Son 0s y 1s

cc: secuencia de palabras código
Son 0s y 1s

rrdem: secuencia de palabras recibidas
LLRs!

mDec: secuencia de mensajes decodificados
Son 0s y 1s



Opciones

c) Turbocódigos (Demodulación Log-likelihood ratio)

```
k=1152;
...
mm              = randi ([0 1], k,1);
...
cc              = turbo_codif(hEnc1,hEnc2,mm);
...
coordtx         = step(hMod,cc);
hcanal          = comm.AWGNChannel('NoiseMethod','Signal to noise ratio (Eb/No)',
'EbNo',EbNo(i),'BitsPerSymbol',1);
coordrx         = step(hcanal,coordtx);
rrdem           = step(hDemod,coordrx);
mDec            = turbo_dec(rrdem*(-1), k,6,hDec1,hDec2);

...
```

El Demod de Matlab y turbo_dec trabajan con $LLR^{(0)} = \ln(P(X=0)/P(X=1))$
Ya no es necesario multiplicar por -1 el LLR

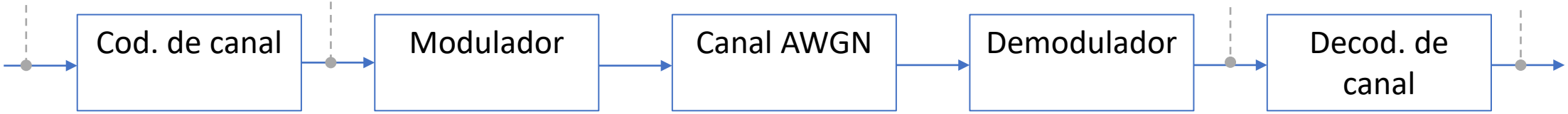
Práctica 2.
5) Codificación y decodificación

mm: secuencia de mensajes
Son 0s y 1s

cc: secuencia de palabras código
Son 0s y 1s

rrdem: secuencia de palabras recibidas
Son LLRs!

mDec: secuencia de mensajes decodificados
Es un vector de booleanos



Opciones

d) LDPC (Demodulación Log-likelihood ratio)

```
load('matrixH.mat')
hEnc = comm.LDPCEncoder(H);
hDec = comm.LDPCDecoder(H);
```

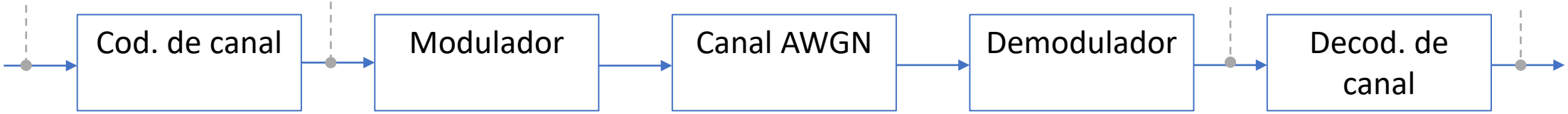
Práctica 2.
5) Codificación y decodificación

mm: secuencia de mensajes
Son 0s y 1s

cc: secuencia de palabras código
Son 0s y 1s

rrdem: secuencia de palabras recibidas
Son LLRs!

mDec: secuencia de mensajes decodificados
Es un vector de booleanos



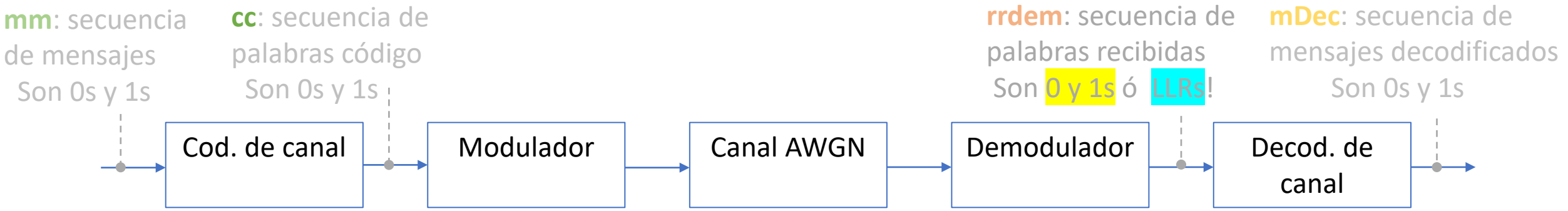
Opciones

d) LDPC (Demodulación Log-likelihood ratio)

```
nmenosk=size(H,1);
n=size(H,2);
k=n-nmenosk;
...
mm              = randi ([0 1], k,1);
...
cc              = step(hEnc,mm);
...
coordtx         = step(hMod,cc);
hcanal          = comm.AWGNChannel('NoiseMethod','Signal to noise ratio (Eb/No)', 'EbNo',EbNo(i),'BitsPerSymbol',1);
coordrx         = step(hcanal,coordtx);
rrdem           = step(hDemod,coordrx);
mDec            = step(hDec,rrdem);

...
```

Práctica 2.
6) BER



Si se utiliza: Sin codificación, codificación convolucional o turbocódigos

```
NumTrans=1000;
for i=1:length(SNR)
...
hCalculoError=comm.ErrorRate;
...
                for j=1:NumTrans
                ...
                hcanal                          = comm.AWGNChannel('NoiseMethod','Signal to noise ratio (Eb/No)',
                                                                   'EbNo',EbNo(i),'BitsPerSymbol',1);
                ...
                VectorErrores    = step(hCalculoError, mm, mDec);
                end
BER(i)=VectorErrores(1);
end
```

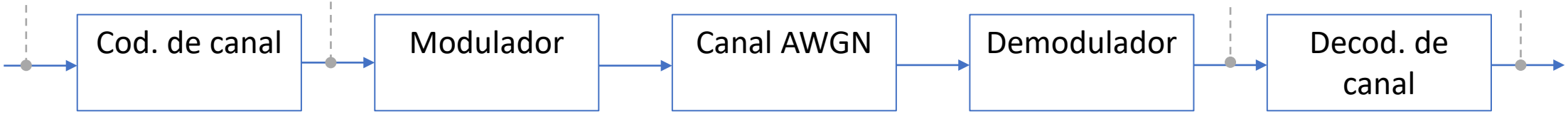
Práctica 2.
6) BER

mm: secuencia de mensajes
Son 0s y 1s

cc: secuencia de palabras código
Son 0s y 1s

rrdem: secuencia de palabras recibidas
Son LLRs!

mDec: secuencia de mensajes decodificados
Es un vector de booleanos



Si se utiliza: LDPC **mDec** es un vector de « booleanos ». Hay que convertirlo a números

```
NumTrans=1000;
for i=1:length(SNR)
...
hCalculoError=comm.ErrorRate;
...
                for j=1:NumTrans
                ...
                hcanal                          = comm.AWGNChannel('NoiseMethod','Signal to noise ratio (Eb/No)',
                                                                   'EbNo',EbNo(i),'BitsPerSymbol',1);
                ...
                VectorErrores    = step(hCalculoError, mm, cast(mDec,'double'));
                end
BER(i)=VectorErrores(1);
end
```

Práctica 2.
7) Otros

- Representación de códigos convolucionales

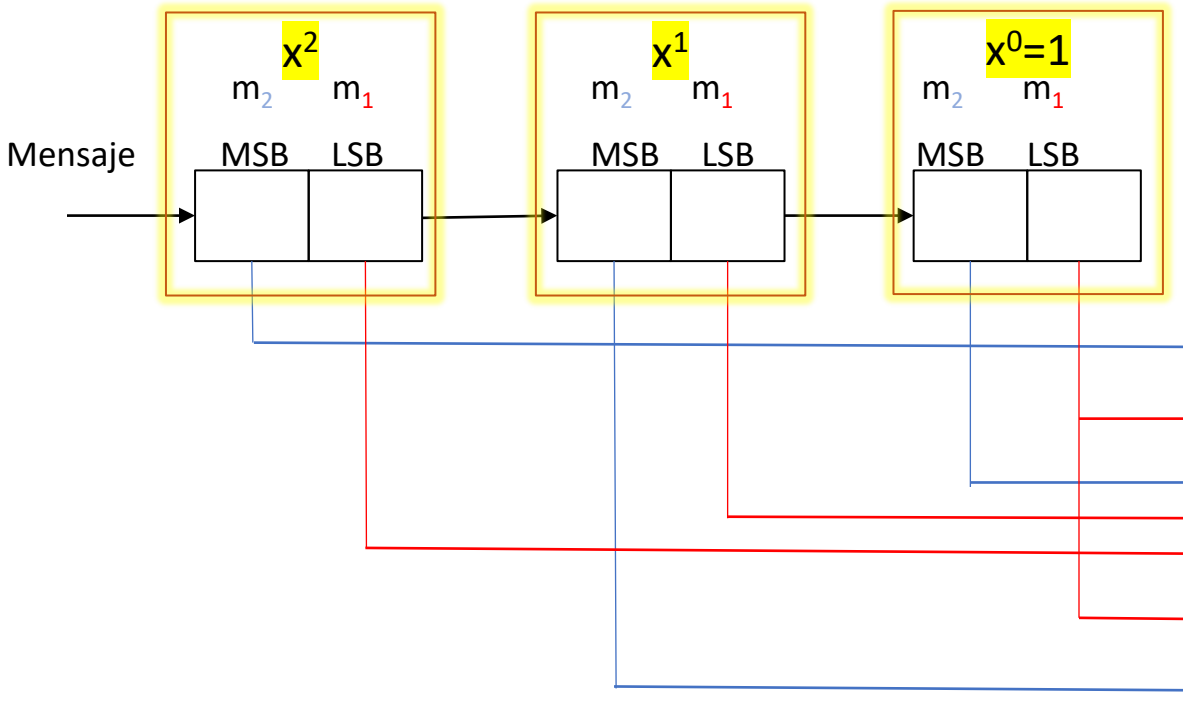
Representación polinómica de códigos convolucionales en Matlab

Código convolucional sin realimentación

C(3,2,3), n=3, k=2, K=3

poly2trellis([3 3],[4 1 2; 1 6 1]);

La segunda entrada de poly2trellis está en Octal



Rama del código

MSB	r_3	$g_{23}(x) = x^2$	[1 0 0]	4
		$g_{13}(x) = 1$	[0 0 1]	1
	r_2	$g_{22}(x) = 1$	[0 0 1]	1
		$g_{12}(x) = x^2 + x$	[1 1 0]	6
LSB	r_1	$g_{21}(x) = x$	[0 1 0]	2
		$g_{11}(x) = 1$	[0 0 1]	1

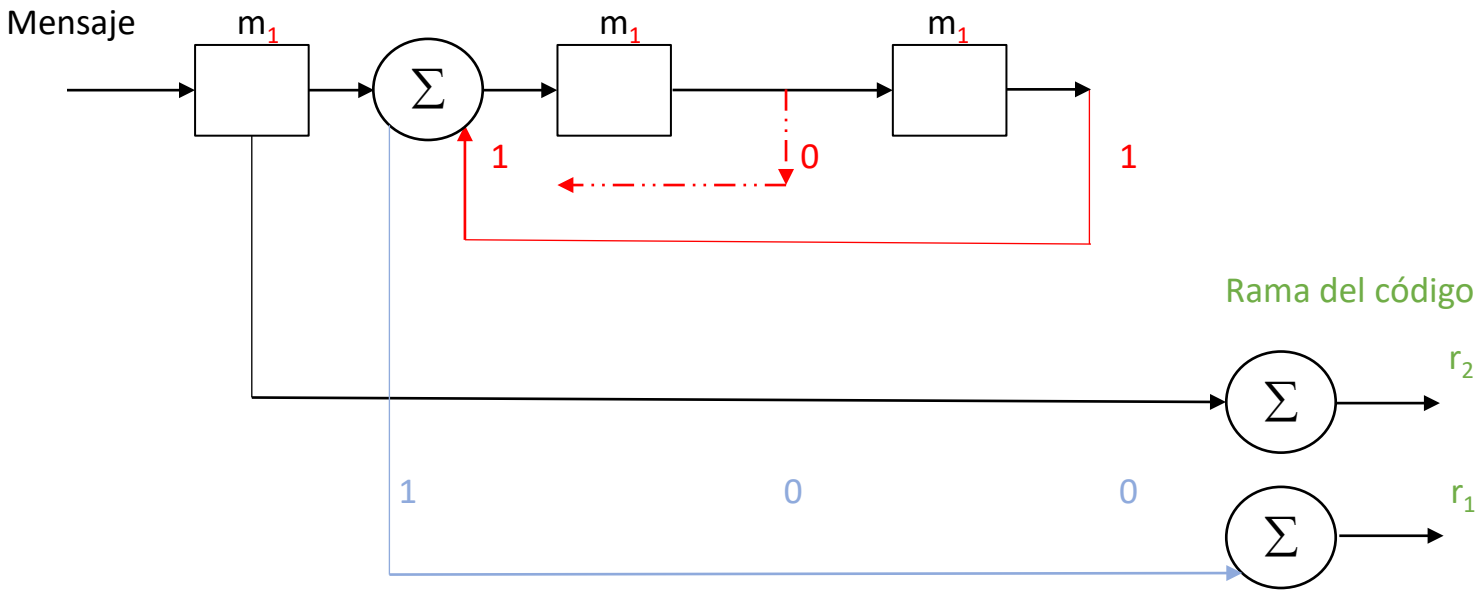
Representación polinómica de códigos convolucionales en Matlab

Código convolucional sistemático con realimentación

C(2,1,3), n=2, k=1, **K=3**

poly2trellis(3,[5 4], 5);

La segunda y tercera entrada de poly2trellis está en Octal



h1(x) = $x^2 + 1$ [1 0 1] 5

$g_1(x) = x^2$ [1 0 0] 4

Representación binaria y octal

2^3	2^2	2^1	2^0	Representación binaria	Cálculo	Representación decimal
0	1	1	1	0111	$0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$	7 = 4 + 2 + 1
1	0	0	0	1000	$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$	8 = 8
1	0	1	1	1011	$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$	11 = 8 + 2 + 1

8^3	8^2	8^1	8^0	Representación octal	Cálculo	Representación decimal
0	0	0	7	0007	$0 \times 8^3 + 0 \times 8^2 + 0 \times 8^1 + 7 \times 8^0$	7 = 7
0	0	1	0	0010	$0 \times 8^3 + 0 \times 8^2 + 1 \times 8^1 + 0 \times 8^0$	8 = 8
0	0	1	3	0013	$0 \times 8^3 + 0 \times 8^2 + 1 \times 8^1 + 3 \times 8^0$	11 = 8 + 3
0	1	5	6	0156	$0 \times 8^3 + 1 \times 8^2 + 5 \times 8^1 + 6 \times 8^0$	110 = 64 + 40 + 6