APÉNDICE

CÓDIGOS DEL PROGRAMA EN MATLAB

7.1. CODIFICACIÓN ESPACIO-TEMPORALES DE TRELLIS

Se muestran a continuación las funciones más significativas de cada sistema.

7.1.1. SISTEMA 4PSK CON 4 ESTADOS

7.1.1.1. DOS ANTENAS EN TRANSMISIÓN Y DOS EN RECEPCIÓN

[salida,n1,n2,antrx1,antrx2,anttx1,anttx2,H,vdecod,SNR,rutas,vaux] = *sistema2antenas4PSK* (entrada,Es,No,SNR);

Función que implementa todo el sistema de transmisión, recepción y descodificación de la trama de bits de entrada.
entrada: secuencia de bits a transmitir.
Es: energía de símbolo.
No: potencia de ruido.
SNR: relación señal a ruido en recepción.

salida: secuencia de bits descodificados.
n1: componente de ruido del canal.
n2: componente de ruido del canal.
antrxj: señal recibida por la antena j.
anttxi: señal transmitida por la antena i.
H: matriz que contiene los coeficientes del canal.
vdecod: vector de datos descodificados en formato binario.
SNR: relación señal a ruido en recepción.
rutas: matriz que contiene los caminos supervivientes.
vaux: vector de datos descodificados en formato complejo.
Convierte la trama de bits de entrada a formato complejo, parte real, parte imaginaria.

[trama] = *conversión* (entrada);

Con esta función se tiene almacenado, en ceros y unos, el árbol generador del código de Trellis para diversidad de dos antenas receptoras y transmisoras.
[diagrama]=estados8PSK;
Con esta función se consigue tener en la matriz C4PSK el mismo árbol generador de código pero ahora, en formato complejo.
[C8PSK]=conversion(diagrama);
A partir de la trama que realmente se quiere transmitir en el emisor, se generan las secuencias de datos que deben salir por cada antena, siguiendo lo que marca el árbol de Trellis.

[anttx1,anttx2]=transmision(trama,Es,C8PSK);
Las tres funciones anteriores estan incluidas en diversidad, que es la función que implementa el canal. Utiliza las señales de salida de cada antena y genera las de llegada a cada antena de recepción. Implementa la ganancia del canal así como la suma de los multicaminos en cada antena receptora.

[antrx1,antrx2,h11,h12,h21,h22,C4PSK,anttx1,anttx2,n1,n2,signal1,signal2] = *diversidad* (trama,Es, No,SNR);

Se reciben dos señales a la entrada de las antenas, y a partir de ellas se decide a que datos corresponden implementando una descodificación blanda de Viterbi.
Devuelve la trama de datos descodificados en formato complejo.

[vdecod,rutas,salida,H,vaux] = *viterbisoft4psk4estados* (C4PSK,antrx1,antrx2,Es,h11,h12,h21,h22);

Convierte la salida descodificada a formato de bits.

[antrx1,antrx2,h11e,h12e,h21e,h22e,C4PSK,anttx1,anttx2,n1,n2,signal1,signal2] = *diversidad* (trama,Es,No,SNR);

Función que implementa la diversidad para dos antenas en recepción y dos en transmisión.
Se diseña tanto la transmisión como la recepción y posterior decisión y descodificación.
Se desea transmitir una secuencia de símbolos 4PSK en formato complejo. Dichos datos se introducen en el transmisor. Por cada símbolo, siguiendo el árbol de estados, se envía un dato por cada antena.
Se implementa el canal de modo que a las antenas del receptor les llegué el símbolo que transmitió su pareja en el transmisor más una parte correspondiente al enviado por la antena "cruzada".
El canal es una matriz de cuatro elementos, donde cada hij representa la ganancia entre la antena transmisora i y la receptora j.
Se supone que el ruido y el canal permanecen constantes durante cada trama.
trama: conjunto de datos a transmitir, símbolos 4PSK.
Es: energía de símbolo.
No: potencia de ruido.
SNR: relación señal a ruido en recepción.
antrx1: señal recibida en la antena 1 del receptor.
antrx2: señal recibida en la antena 2 del receptor.
hije: coeficientes escalados del canal.
C4PSK: matriz de las posibles transiciones entre estados.
anttx1: señal emitida por la antena 1 del transmisor.
anttx2: señal emitida por la antena 2 del transmisor.
n1: componente de ruido del canal.
n2: componente de ruido del canal.
signal1: señal emitida modificada por el canal pero sin el efecto del ruido.
signal2: señal emitida modificada por el canal pero sin el efecto del ruido.

Se parte del estado 0, estado inicial.
Para cada símbolo a transmitir se estudia con que estado corresponde y cuales son los valores que deben transmitirse por cada antena.

[diagrama] = *estados4PSK;*
[C4PSK] = *conversion* (diagrama);

```
En C4PSK se tiene la matriz de transiciones entre estados colocada en formato complejo,
lista para comparar con los datos entrantes.
Según trama, se generan dos vectores correspondientes a los datos que hay que transmitir
por cada antena, anttx1, anttx2.

[anttx1,anttx2] = transmision (trama,C4PSK);

Ahora basta transmitirlos por el canal.
En las antenas receptoras se recibe:
antrx1=h11*anttx1+h21*anttx2+ruido
antrx2=h12*anttx1+h22*anttx2+ruido
Una vez modelados ruido y canal, solo queda decidir cual es la trama que ha sido enviada,
siguiendo un criterio de métrica mínima.

long=length(anttx1);

long es la longitud de la trama recibida en cada antena.

sig=sqrt(.5);

CANAL REAL
h11a = ruido (sig,long);
h12a = ruido (sig,long);
h21a = ruido (sig,long);
h22a = ruido (sig,long);
h11=h11a(1);
h12=h12a(1);
h21=h21a(1);
h22=h22a(1);

CANAL IDEAL
h11=ones(1,long);
h12=zeros(1,long);
h22=ones(1,long);
h21=zeros(1,long);

H=[h11 h12;h21 h22];

Para las antenas receptoras el ruido esta incorrelado. Se supone que éste es constante a lo largo
de una trama, gaussiano, de media cero y su potencia es No/2 por dimensión.

desvruido=sqrt(No);

CANAL RUIDOSO
n1 = ruido (desvruido,long);
n2 = ruido (desvruido,long);

CANAL NO RUIDOSO
n1=zeros(1,long);
n2=zeros(1,long);
```

```
for i=1:long
  signal1(i)=sqrt(Es/2)*(h11*anttx1(i)+h21*anttx2(i));
  signal2(i)=sqrt(Es/2)*(h12*anttx1(i)+h22*anttx2(i));
end;

[h11e,h12e,h21e,h22e] = escalado (signal1,signal2,h11,h12,h21,h22,No,SNR);

for i=1:long
  antrx1(i)=sqrt(Es/2)*(h11e*anttx1(i)+h21e*anttx2(i))+n1(i);
  antrx2(i)=sqrt(Es/2)*(h12e*anttx1(i)+h22e*anttx2(i))+n2(i);
end;
```

Una vez se tienen las señales a la entrada de cada antena, solo queda aplicar la descodificación blanda de Viterbi.

[vdecod,rutas,tramadecod,H,vaux] = *viterbisoft4psk4estados* (C4PSK,antrx1,antrx2,Es,h11e, h12e,h21e,h22e);

Es la función que implementa la descodificación de Viterbi con decisión blanda.
Recibe una señal entrante en cada antena y a partir de ellas, por comparación con el árbol de estados, decide cual es la trama transmitida usando el criterio de métrica mínima.

Para cada símbolo, partiendo del estado cero se compara con toda la fila de posibles transiciones. Se decide en función del que tenga métrica mínima y se averigua el estado al que se llega. Se queda con cuatro estados supervivientes, uno correspondiente a cada uno de los caminos entrantes en cada estado de llegada. Se guarda la transición en la matriz donde se conservará la ruta seguida. En el nuevo estado se vuelve a hacer lo mismo, buscando las transiciones que den menor métrica.

C4PSK: matriz que contiene las transiciones entre estados.
antrxj: vector de símbolos recibidos en la antena j.
Es: energía de símbolo.
hije: coeficientes escalados del canal.

vdecod: vector de datos descodificados en formato binario.
rutas: matriz que contiene los caminos supervivientes.
tramadecod: vector de datos descodificados en formato complejo.
H: matriz de los coeficientes del canal.
vaux: vector de datos descodificados en formato complejo.

tam es la longitud de la trama entrante a cada una de las antenas

```
tam=length(antrx1);
rutas=zeros(4,4);
rutasaux=rutas;
estproc=[1:4];
mtotal=zeros(1,4);
maux=zeros(4,4);
m=zeros(4,4);
k=1;
for j=1:tam                                  Recorriendo la longitud de la trama recibida
   H=[h11e h12e;h21e h22e];
```

```
  for h=1:4;                                          Para todos los estados de los que se procede
    ind=1;                                  Recorre todas las transiciones del estado h a todos los demás
    for i=1:2:7;
      r=[antrx1(j) antrx2(j)];
      c=[C4PSK(h,i) C4PSK(h,i+1)];
      comp=sqrt(Es/2)*c*H;                          Se tiene conocimiento del canal al multiplicar *H
      metric=(r-comp)*(r-comp)';
      m(h,ind)=metric;
      ind=ind+1;
    end;
  end;
```

En este momento se tiene, para cada iteración, una matriz de métricas, debe elegirse a los supervivientes una vez analizado cada dato de entrada. Hay que buscar el mínimo por columnas.

```
  for i=1:4
    maux(i,:)=m(i,:)+mtotal(i);
  end;
  for i=1:4
    col=maux(:,i);
    aux=find(col==min(col));
```

Da los índices de métrica mínima , quedándose con el primero de ellos, que se corresponde con el estado desde el que se parte para llegar al estado i con menos coste.

```
        estproc(i)=aux(1);
```

En este punto se tiene la mínima transición de cualquier estado al estado i es decir, una transición superviviente.
Se actualiza la métrica total.

```
    est=estproc(i);
    mtotal(i)=maux(est,i);
    l=2*(i-1);
    rutas(i,:)=rutasaux(est,:);
    rutas(i,k:k+1)=C4PSK(est,l+1:l+2);          Camino superviviente que llega al estado i
  end;
  k=k+2;
  rutas=[rutas,zeros(4,2)];
  rutasaux=rutas;
end;
indsuper=find(mtotal==min(mtotal));                   Índice q da la métrica mínima
indice=indsuper(1);
long=length(rutas(indice,:));
vdecod=rutas(indice,1:long-2);                                      Camino mínimo
```

En vdecod se tiene la mezcla de lo descodificado en cada antena. Lo correspondiente a la segunda antena es la trama transmitida, así que para obtenerla se procede como sigue.

```
[vaux] = conversioninversa (vdecod);
lon=length(vaux);
ind=3;
```

```
for i=1:2:(lon/2)-1
  tramadecod(i:i+1)=vaux(ind:ind+1);
  ind=ind+4;
end;
```

7.1.1.2. DOS ANTENAS EN TRANSMISIÓN Y UNA EN RECEPCIÓN

[salida,n1,antrx1,anttx1,anttx2,H,vdecod,SNR,rutas,vaux] = *sistema214PSK* (entrada,Es,No,SNR);

Función que implementa todo el sistema de transmisión, recepción y descodificación de la trama de bits de entrada.

entrada: secuencia de bits a transmitir.
Es: energía de símbolo.
No: potencia de ruido.
SNR: relación señal a ruido en recepción.

salida: secuencia de bits descodificados.
n1: componente de ruido del canal.
antrx1: señal recibida.
anttxi: señal transmitida por la antena i.
H: matriz que contiene los coeficientes del canal.
vdecod: vector de datos descodificados en formato binario.
rutas: matriz que contiene los caminos supervivientes.
vaux: vector de datos descodificados en formato complejo.

Convierte la trama de bits de entrada a formato complejo, parte real,
parte imaginaria.

[trama] = *conversion* (entrada);

Es la función que implementa el canal. Utiliza las señales de salida de cada antena y genera la de llegada a la antena de recepción. Implementa la ganancia del canal así como la suma de los multicaminos en la antena receptora.

[antrx1,h11,h21,C4PSK,anttx1,anttx2,n1,signal1] = *diversidad*(trama,Es, No,SNR);

Se recibe una señal a la entrada de la antena, y a partir de ella se decide a que datos corresponde implementando una descodificación blanda de Viterbi.
Devuelve la trama de datos descodificados en formato complejo.

[vdecod,rutas,salida,H,vaux] = *viterbisoft4psk4estados* (C4PSK,antrx1,Es,
h11,h21);

[antrx1,h11e,h21e,C4PSK,anttx1,anttx2,n1,signal1] = *diversidad* (trama,Es, No,SNR);

Función que implementa la diversidad para una antena en recepción y dos en transmisión.
Se diseña tanto la transmisión como la recepción y posterior decisión y descodificación.
Se desea transmitir una secuencia de símbolos 4PSK en formato complejo. Dichos datos se introducen en el transmisor. Por cada símbolo, siguiendo el árbol de estados, se envía un dato por cada antena.

Se implementa el canal de modo que a la antena del receptor le llega el símbolo que transmitió su pareja en el transmisor más una parte correspondiente al enviado por la antena "cruzada".
El canal es una matriz de dos elementos, donde cada hi1 representa la ganancia entre la antena transmisora i y la receptora.
Se supone que el ruido y el canal permanecen constantes durante cada trama.

trama: conjunto de datos a transmitir, símbolos 4PSK.
Es: energía de símbolo.
No: potencia de ruido.
SNR: relación señal a ruido en recepción.

antrx1: señal recibida en el receptor.
h11e: coeficiente escalado del canal.
h21e: coeficiente escalado del canal.
C4PSK: matriz de las posibles transiciones entre estados.
anttx1: señal emitida por la antena 1 del transmisor.
anttx2: señal emitida por la antena 2 del transmisor.
n1: componente de ruido del canal.
signal1: señal emitida modificada por el canal pero sin el efecto del ruido.

Se parte del estado 0, estado inicial.
Para cada símbolo a transmitir se estudia con que estado corresponde y cuales son los valores que deben transmitirse por cada antena.

[diagrama] = *estados4PSK;*
[C4PSK] = *conversion* (diagrama);

Según trama, se generan dos vectores correspondientes a los datos que hay que transmitir por cada antena, anttx1, anttx2.

[anttx1,anttx2] = *transmision* (trama,C4PSK);

Ahora basta transmitirlos por el canal.
En la antena receptora se recibe: antrx1=h11*anttx1+h21*anttx2+ruido
Una vez modelados ruido y canal, solo queda decidir cual es la trama que ha sido enviada, siguiendo un criterio de métrica mínima.

long=length(anttx1);

long es la longitud de la trama recibida en cada antena.

sig=sqrt(.5);

CANAL REAL
h11a = *ruido* (sig,long);
h21a = *ruido* (sig,long);
h11=h11a(1);
h21=h21a(1);

CANAL IDEAL
h11=ones(1,long);
h21=zeros(1,long);

```
El ruido es gaussiano, de media cero y potencia es No/2 por dimensión.

desvruido=sqrt(No);

CANAL RUIDOSO
n1 = ruido (desvruido,long);

CANAL NO RUIDOSO
n1=zeros(1,long);

for i=1:long
   signal1(i)=sqrt(Es/2)*(h11*anttx1(i)+h21*anttx2(i));
end;
[h11e,h21e]=escalado(signal1,h11,h21,No,SNR);
for i=1:long
   antrx1(i)=sqrt(Es/2)*(h11e*anttx1(i)+h21e*anttx2(i))+n1(i);
end;

[vdecod,rutas,tramadecod,H,vaux] = viterbisoft4psk4estados (C4PSK,antrx1, Es,h11e,h21e);

Es la función que implementa la descodificación de Viterbi con decisión blanda.
Recibe señal entrante en la antena y a partir de ella, por comparación con el árbol de estados,
decide cual es la trama transmitida usando el criterio de métrica mínima.

Para cada símbolo, partiendo del estado cero se compara con toda la fila de posibles transiciones.
Se decide en función del que tenga métrica mínima y se averigua el estado al que se llega. Se que-
da con cuatro estados supervivientes, uno correspondiente a cada uno de los caminos entrantes
en cada estado de llegada. Se guarda la transición en la matriz donde se conservará la ruta segui-
da. En el nuevo estado se vuelve a hacer lo mismo, buscando las transiciones que den menor mé-
trica.

C4PSK: matriz que contiene las transiciones entre estados.
antrx1: vector de símbolos recibidos.
Es: energía de símbolo.
hije: coeficientes escalados del canal.

vdecod: vector de datos descodificados en formato binario.
rutas: matriz que contiene los caminos supervivientes.
tramadecod: vector de datos descodificados en formato complejo.
H: matriz de los coeficientes del canal.
vaux: vector de datos descodificados en formato complejo.

tam es la longitud de la trama entrante a cada una de las antenas

tam=length(antrx1);
rutas=zeros(4,4);
rutasaux=rutas;
estproc=[1:4];
mtotal=zeros(1,4);
maux=zeros(4,4);
m=zeros(4,4);
```

```
k=1;
for j=1:tam                                   Recorriendo la longitud de la trama recibida
   H=[h11e;h21e];
   for h=1:4;                                 Para todos los estados de los que se procede
     ind=1;                                   Recorre las transiciones del estado h a todos los demás
     for i=1:2:7;
       r=[antrx1(j)];
       c=[C4PSK(h,i) C4PSK(h,i+1)];
       comp=sqrt(Es/2)*c*H;                   Se tiene conocimiento del canal H
       metric=(r-comp)*(r-comp)';
       m(h,ind)=metric;
       ind=ind+1;
     end;
   end;
```

En este momento se tiene, para cada iteración, una matriz de métricas, debe elegirse a los supervivientes una vez analizado cada dato de entra-da. Hay que buscar el mínimo por columnas.

```
  for i=1:4
    maux(i,:)=m(i,:)+mtotal(i);
  end;
  for i=1:4
    col=maux(:,i);
    aux=find(col==min(col));
```

Da los índices de métrica mínima , quedándose con el primero de ellos, que se corresponde con el estado desde el que se parte para llegar al estado i con menos coste.

```
          estproc(i)=aux(1);
```

En este punto se tiene la mínima transición de cualquier estado al estado i es decir, una transición superviviente. Se actualiza la métrica total.

```
    est=estproc(i);
    mtotal(i)=maux(est,i);
    l=2*(i-1);
    rutas(i,:)=rutasaux(est,:);
    rutas(i,k:k+1)=C4PSK(est,l+1:l+2);        Camino superviviente que llega al estado i
  end;
  k=k+2;
  rutas=[rutas,zeros(4,2)];
  rutasaux=rutas;
end;
indsuper=find(mtotal==min(mtotal));           Índice que da la métrica mínima
indice=indsuper(1);
long=length(rutas(indice,:));
vdecod=rutas(indice,1:long-2);
[vaux] = conversioninversa (vdecod);
lon=length(vaux);
ind=3;
for i=1:2:(lon/2)-1
  tramadecod(i:i+1)=vaux(ind:ind+1);
```

```
  ind=ind+4;
end;
```

7.1.1.3. UNA ANTENA EN TRANSMISIÓN Y UNA EN RECEPCIÓN

[salida,n1,antrx1,anttx1,H,vdecod,SNR,rutas] = *sistema114PSK* (entrada,Es, No,SNR);

Función que implementa todo el sistema de transmisión, recepción y descodificación de la trama de bits de entrada.

entrada: secuencia de bits a transmitir.
Es: energía de símbolo.
No: potencia de ruido.
SNR: relación señal a ruido en recepción.

salida: secuencia de bits descodificados.
n1: componente de ruido.
antrx1: señal recibida en la antena.
anttx1: señal transmitida por la antena.
H: coeficiente del canal.
vdecod: vector de datos descodificados.
rutas: matriz de caminos supervivientes.

Convierte la trama de bits de entrada a formato complejo, parte real, parte imaginaria.

[trama] = *conversion* (entrada);

Con esta función se tiene almacenado, en ceros y unos, el árbol generador del código de Trellis para el sistema que sólo tiene una antena receptora y transmisora.

A partir de la trama q realmente se quiere transmitir en el emisor, se genera la secuencia de datos que debe salir por la antena transmisora, siguiendo lo que marca el árbol de Trellis.

[antrx1,h11,C4PSK,anttx1,n1,signal1] = *diversidad* (trama,Es,No,SNR);

Se recibe una señal a la entrada de las antena receptora, y a partir de ella se decide a que datos corresponde, implementando una descodificación blanda de Viterbi.
Devuelve la trama de datos descodificados en formato complejo.

[vdecod,rutas,salida,H] = *viterbisoft4psk4estados* (C4PSK,antrx1,Es,h11);

[antrx1,h11e,C4PSK,anttx1,n1,signal1] = *diversidad* (trama,Es,No,SNR);

Función que implementa la diversidad para una antena en recepción y otra en transmisión.
Se diseña tanto la transmisión como la recepción y posterior decisión y descodificación.
Se desea transmitir una secuencia de símbolos 4PSK en formato complejo. Dichos datos se introducen en el transmisor. Por cada símbolo, siguiendo el árbol de estados, se envía un dato por la antena.
En este caso, la información llega a la antena receptora modificada por el coeficiente que representa al canal, y con una componente de ruido aditivo gausiano y blanco.
En principio, supondremos que el ruido y el canal permanecen constantes durante cada trama.

trama, es el conjunto de datos a transmitir, símbolos 4PSK.
Es es la energía de símbolo.
No es la potencia del ruido del canal.
SNR relación señal a ruido en recepción.

anttrx1: señal que llega a la antena del receptor.
h11e: coeficiente del canal escalado.
C4PSK: matriz que contiene el árbol de transiciones posibles del sistema.
anttx1: señal que sale de la antena transmisora.
n1: componente de ruido gaussiano, blanco y aditivo.
signal1: señal transmitida por el canal que no incluye la componente de ruido.

Se parte del estado 0, estado inicial.
Para cada símbolo a transmitir se ve con que estado corresponde y cuales son los valores que deben transmitirse por cada antena.

[diagrama] = *estados4PSK*;
[C4PSK] = *conversion* (diagrama);

En C4PSK tenemos la matriz de transiciones de estados en formato complejo, lista para comparar con los datos entrantes.
Según trama, se genera un vector anttx1, que corresponde con la secuencia de datos a transmitir.

Una vez modelados ruido y canal, solo queda decidir cual es la trama que ha sido enviada, siguiendo un criterio de métrica mínima.

```
anttx1=trama;
long=length(anttx1);
```

long es la longitud de la trama recibida en la antena.

```
sig=sqrt(.5);
```

CANAL REAL

```
h11a = ruido (sig,long);
h11=h11a(1);
```

CANAL IDEAL

```
h11=ones(1,long);
```

El ruido es constante a lo largo de una trama, gaussiano, de media cero y su potencia es No/2 por dimensión.

```
desvruido=sqrt(No);
```

CANAL RUIDOSO

```
n1 = ruido (desvruido,long);
```

CANAL NO RUIDOSO

```
n1=zeros(1,long);

for i=1:long
  signal1(i)=sqrt(Es/2)*(h11*anttx1(i));
end;

[h11e] = escalado (signal1,h11,No,SNR);
for i=1:long
  antrx1(i)=sqrt(Es/2)*(h11e*anttx1(i))+n1(i);
end;
```

[vdecod,rutas,tramadecod,H] = *viterbisoft4psk4estados* (C4PSK,antrx1,Es, h11e);

Es la función que implementa la descodificación de Viterbi con decisión blanda.
Recibe la señale entrante en la antena receptora y a partir de ella, por comparación con el árbol de estados, decide cual es la trama transmitida usando el criterio de métrica mínima.

C4PSK: matriz que contiene el diagrama de estados.
antrx1: señal recibida en la antena.
Es: energía de símbolo.
h11e: coeficiente escalado del canal.

vdecod: vector que contiene la señal descodificada en formato complejo.
rutas: matriz que contiene los caminos supervivientes.
tramadecod: secuencia de datos que realmente se transmitió en el emisor.
Formato de bits.
H: coeficiente del canal.

Se toma un símbolo, partiendo del estado cero se compara con todas las posibles transiciones entre estados. Se decide en función del que tenga métrica mínima y se averigua el estado al que se llega. Se guardan los caminos supervivientes, uno correspondiente a cada uno de los caminos entrantes en cada estado de llegada. Se almacena la transición en la matriz donde se conservará la ruta seguida. En el nuevo estado se vuelve a hacer lo mismo, buscando las transiciones que den menor métrica.

tam es la longitud de la trama entrante a la antena.

```
tam=length(antrx1);
rutas=zeros(4,4);
rutasaux=rutas;
estproc=[1:4];
mtotal=zeros(1,4);
maux=zeros(4,4);
m=zeros(4,4);
k=1;
for j=1:tam                                   Recorriendo la longitud de la trama recibida
    H=[h11e];
    for h=1:4;                                Para todos los estados de los que se procede
      ind=1;                                  Recorre todas las transiciones del estado h a todos los demás
      for i=1:4;
        r=[antrx1(j)];
        c=C4PSK(h,i);
```

```
      comp=sqrt(Es/2)*c*H;                                  Se tiene conocimiento del canal H
      metric=(r-comp)*(r-comp)';
      m(h,ind)=metric;
      ind=ind+1;
    end;
  end;
```

En este momento se obtiene, para cada iteración, una matriz de métricas, debiendo elegir a los supervivientes una vez analizado cada dato de entrada. Hay que buscar el mínimo por columnas.

```
  for i=1:4
    maux(i,:)=m(i,:)+mtotal(i);
  end;
  for i=1:4
    col=maux(:,i);
    aux=find(col==min(col));
```

Da los índices de métrica mínima, quedándose con el primero de ellos, que se corresponde con el estado desde el que se parte para llegar al estado i con menos coste.

```
          estproc(i)=aux(1);
```

En este punto se tiene la mínima transición de cualquier estado al estado i es decir, una transición superviviente.
Se actualiza la métrica total, que es un vector de los caminos supervivientes.

```
    est=estproc(i);
    mtotal(i)=maux(est,i);
    rutas(i,:)=rutasaux(est,:);
    rutas(i,k)=C4PSK(est,i);                          Camino superviviente q llega al estado i
  end;
  k=k+1;
  rutas=[rutas,zeros(4,1)];
  rutasaux=rutas;
end;
indsuper=find(mtotal==min(mtotal));                   Índice q da la métrica mínima
indice=indsuper(1);
long=length(rutas(indice,:));
vdecod=rutas(indice,1:long-1);
[tramadecod] = conversioninversa (vdecod);
```

7.1.2. SISTEMA 8PSK CON 8 ESTADOS

7.1.2.1. DOS ANTENAS EN TRANSMISIÓN Y DOS EN RECEPCIÓN

[salida,n1,n2,antrx1,antrx2,anttx1,anttx2,H,vdecod,SNR,rutas] = *sistema2antenas8PSK* (entrada,Es,No,SNR);

Función que implementa todo el sistema de transmisión, recepción y descodificación de la trama de bits de entrada.

entrada: secuencia de bits a transmitir.
Es: energía de símbolo.
No: potencia de ruido.
SNR: relación señal a ruido en recepción.

salida: secuencia de bits descodificados.
n1: componente de ruido del canal.
n2: componente de ruido del canal.
antrxj: señal recibida por la antena j.
anttxi: señal transmitida por la antena i.
H: matriz que contiene los coeficientes del canal.
vdecod: vector de datos descodificados en formato binario.
SNR: relación señal a ruido en recepción.
rutas: matriz que contiene los caminos supervivientes.

Convierte la trama de bits de entrada a formato complejo, parte real, parte imaginaria.

[trama] = *conversion* (entrada);

Con esta función se tiene almacenado, en ceros y unos, el árbol generador del código de Trellis para diversidad de dos antenas receptoras y transmisoras.

[diagrama]=estados8PSK;

Con esta función se consigue tener en la matriz C4PSK el mismo árbol generador de código pero ahora, en formato complejo.

[C8PSK]=conversion(diagrama);

A partir de la trama que realmente se quiere transmitir en el emisor, se generan las secuencias de datos que deben salir por cada antena, siguiendo lo que marca el árbol de Trellis.

[anttx1,anttx2]=transmision(trama,Es,C8PSK);

Las tres funciones anteriores estan incluidas en diversidad, que es la función que implementa el canal. Utiliza las señales de salida de cada antena y genera las de llegada a cada antena de recepción. Implementa la ganancia del canal así como la suma de los multicaminos en cada antena receptora.

[antrx1,antrx2,h11,h12,h21,h22,C8PSK,anttx1,anttx2,n1,n2,signal1,signal2] = *diversidad* (trama,Es,No,SNR);

Se reciben dos señales a la entrada de las antenas, y a partir de ellas se decide a que datos corresponden implementando una descodificación blanda de Viterbi.
Devuelve la trama de datos descodificados en formato complejo.

[vdecod,rutas,salida,H] = *viterbisoft8psk8estados* (C8PSK,antrx1,antrx2,Es,h11,h12,h21,h22);

Convierte la salida descodificada a formato de bits.

[antrx1,antrx2,h11e,h12e,h21e,h22e,C8PSK,anttx1,anttx2,n1,n2,signal1,signal2]=diversidad(trama, Es,No,SNR);

Función que implementa la diversidad para dos antenas en recepción y dos en transmisión.
Se diseña tanto la transmisión como la recepción y posterior decisión y descodificación.
Se desea transmitir una secuencia de símbolos 4PSK en formato complejo. Dichos datos se introducen en el transmisor. Por cada símbolo, siguiendo el árbol de estados, se envía un dato por cada antena.
Se implementa el canal de modo que a las antenas del receptor les llegue el símbolo que transmitió su pareja en el transmisor más una parte correspondiente al enviado por la antena "cruzada".
El canal es una matriz de cuatro elementos, donde cada hij representa la ganancia entre la antena transmisora i y la receptora j.
Se supone que el ruido y el canal permanecen constantes durante cada trama.

trama: conjunto de datos a transmitir, símbolos 4PSK.
Es: energía de símbolo.
No: potencia de ruido.
SNR: relación señal a ruido en recepción.

antrx1: señal recibida en la antena 1 del receptor.
antrx2: señal recibida en la antena 2 del receptor.
hije: coeficientes escalados del canal.
C8PSK: matriz de las posibles transiciones entre estados.
anttx1: señal emitida por la antena 1 del transmisor.
anttx2: señal emitida por la antena 2 del transmisor.
n1: componente de ruido del canal.
n2: componente de ruido del canal.
signal1: señal emitida modificada por el canal pero sin el efecto del ruido.
signal2: señal emitida modificada por el canal pero sin el efecto del ruido.

Se parte del estado 0, estado inicial.
Para cada símbolo a transmitir se estudia con que estado corresponde y cuales son los valores que deben transmitirse por cada antena.

```
[diagrama]=estados8PSK;
[C8PSK] = conversion (diagrama);
```

En C8PSK se tiene la matriz de transiciones entre estados colocada en formato complejo, lista para comparar con los datos entrantes.
Según trama, se generan dos vectores correspondientes a los datos que hay que transmitir por cada antena, anttx1, anttx2.

```
[anttx1,anttx2] = transmision (trama,C8PSK);
```

Ahora basta transmitirlos por el canal.
En las antenas receptoras se recibe:
antrx1=h11*anttx1+h21*anttx2+ruido
antrx2=h12*anttx1+h22*anttx2+ruido
Una vez modelados ruido y canal, solo queda decidir cual es la trama que ha sido enviada, siguiendo un criterio de métrica mínima.

```
long=length(anttx1);
```

long, es la longitud de la trama recibida en cada antena.

```
sig=sqrt(.5);

CANAL REAL
h11a= ruido (sig,long);
h12a= ruido (sig,long);
h21a= ruido (sig,long);
h22a= ruido (sig,long);
h11=h11a(1);
h12=h12a(1);
h21=h21a(1);
h22=h22a(1);

CANAL IDEAL
h11=ones(1,long);
h12=zeros(1,long);
h22=ones(1,long);
h21=zeros(1,long);

H=[h11 h12;h21 h22];
```

Para las antenas receptoras el ruido esta incorrelado. Se supone que éste es constante a lo largo de una trama, gaussiano, de media cero y su potencia es No/2 por dimensión.

```
desvruido=sqrt(No);

CANAL RUIDOSO
n1= ruido (desvruido,long);
n2= ruido (desvruido,long);

 CANAL NO RUIDOSO
n1=zeros(1,long);
n2=zeros(1,long);

for i=1:long
   signal1(i)=sqrt(Es/2)*(h11*anttx1(i)+h21*anttx2(i));
   signal2(i)=sqrt(Es/2)*(h12*anttx1(i)+h22*anttx2(i));
end;
[h11e,h12e,h21e,h22e] = escalado (signal1,signal2,h11,h12,h21,h22,No,SNR);
for i=1:long
   antrx1(i)=sqrt(Es/2)*(h11e*anttx1(i)+h21e*anttx2(i))+n1(i);
   antrx2(i)=sqrt(Es/2)*(h12e*anttx1(i)+h22e*anttx2(i))+n2(i);
end;
```

Una vez se tienen las señales a la entrada de cada antena, solo queda aplicar la descodificación blanda de Viterbi.

[vdecod,rutas,tramadecod,H] = *viterbisoft8psk8estados* (C8PSK,antrx1,antrx2,Es,h11e, h12e,h21e,h22e);

Es la función que implementa la descodificación de Viterbi con decisión blanda.

Recibe una señal entrante en cada antena y a partir de ellas, por comparación con el árbol de estados, decide cual es la trama transmitida usando el criterio de métrica mínima.
Para cada símbolo, partiendo del estado cero se compara con toda la fila de posibles transiciones. Se decide en función del que tenga métrica mínima y se averigua el estado al que se llega. Se queda con cuatro estados supervivientes, uno correspondiente a cada uno de los caminos entrantes en cada estado de llegada. Se guarda la transición en la matriz donde se conservará la ruta seguida. En el nuevo estado se vuelve a hacer lo mismo, buscando las transiciones que den menor métrica.

C8PSK: matriz que contiene las transiciones entre estados.
antrxj: vector de símbolos recibidos en la antena j.
Es: energía de símbolo.
hije: coeficientes escalados del canal.

vdecod: vector de datos descodificados en formato binario.
rutas: matriz que contiene los caminos supervivientes.
tramadecod: vector de datos descodificados en formato complejo.
H: matriz de los coeficientes del canal.

tam es la longitud de la trama entrante a cada una de las antenas

```
tam=length(antrx1);
rutas=zeros(8,4);
rutasaux=rutas;
estproc=[1:8];
mtotal=zeros(1,8);
maux=zeros(8,8);
m=zeros(8,8);
k=1;
for j=1:tam                                          Recorriendo la longitud de la trama recibida
   H=[h11e h12e;h21e h22e];
   for h=1:8;                                        Para todos los estados de los que se procede
     ind=1;                                 Recorre todas las transiciones del estado h a todos los demás
     for i=1:2:15;
        r=[antrx1(j) antrx2(j)];
        c=[C8PSK(h,i) C8PSK(h,i+1)];
        comp=sqrt(Es/2)*c*H;                        Se tiene conocimiento del canal al multiplicar *H
        metric=(r-comp)*(r-comp)';
        m(h,ind)=metric;
        ind=ind+1;
     end;
   end;
```

En este momento se tiene, para cada iteración, una matriz de métricas, debe elegirse a los supervivientes una vez analizado cada dato de entrada. Hay que buscar el mínimo por columnas.

```
  for i=1:8
     maux(i,:)=m(i,:)+mtotal(i);
  end;
  for i=1:8
     col=maux(:,i);
     aux=find(col==min(col));
```

Da los índices de métrica mínima , quedándose con el primero de ellos, que se corresponde con el estado desde el que se parte para llegar al estado i con menos coste.

```
        estproc(i)=aux(1);
```

En este punto se tiene la mínima transición de cualquier estado al estado i es decir, una transición superviviente. Se actualiza la métrica total.

```
    est=estproc(i);
    mtotal(i)=maux(est,i);
    l=2*(i-1);
    rutas(i,:)=rutasaux(est,:);
    rutas(i,k:k+1)=C8PSK(est,l+1:l+2);          Camino superviviente que llega al estado i
  end;
  k=k+2;
  rutas=[rutas,zeros(8,2)];
  rutasaux=rutas;
end;
indsuper=find(mtotal==min(mtotal));              Índice que da la métrica mínima
long=length(rutas(indsuper,:));
vdecod=rutas(indsuper,1:long-2);                                  Camino mínimo
```

En vdecod se tiene la mezcla de lo descodificado en cada antena. Lo correspondiente a la segunda antena es la trama transmitida, asi que para obtenerla se procede como sigue.

```
[vaux]=conversioninversa(vdecod);
lon=length(vaux);
ind=4;
for i=1:3:(lon/2)-1
  tramadecod(i:i+2)=vaux(ind:ind+2);
  ind=ind+6;
end;
```

7.1.2.2. DOS ANTENAS EN TRANSMISIÓN Y UNA EN RECEPCIÓN

[salida] = *sistema2antenas8PSK* (entrada,Es,No,SNR);

Función que implementa todo el sistema de transmisión, recepción y descodificación de la trama de bits de entrada.

entrada: secuencia de bits a transmitir.
Es: energía de símbolo.
No: potencia de ruido.
SNR: relación señal a ruido en recepción.

salida: secuencia de bits descodificados.

Convierte la trama de bits de entrada a formato complejo, parte real, parte imaginaria.

```
[trama]=conversion(entrada);
```

Es la función que implementa el canal. Utiliza las señales de salida de cada antena y genera la de llegada a la antena de recepción. Implementa la ganancia del canal así como la suma de los multi-caminos en la antena receptora.

[antrx1,h11,h21,C8PSK,anttx1,anttx2,n1,signal1] = *diversidad* (trama,Es,No,SNR);

Se recibe una señal a la entrada de la antena, y a partir de ella se decide a que datos corresponde implementando una descodificación blanda de Viterbi.
Devuelve la trama de datos descodificados en formato complejo.

[vdecod,rutas,salida,H] = *viterbisoft8psk8estados* (C8PSK,antrx1,Es,h11,h21);

[antrx1,h11e,h21e,C8PSK,anttx1,anttx2,n1,signal1] = *diversidad* (trama,Es,No,SNR);

Función que implementa la diversidad para una antena en recepción y dos en transmisión.
Se diseña tanto la transmisión como la recepción y posterior decisión y descodificación.
Se desea transmitir una secuencia de símbolos 8PSK en formato complejo.
Dichos datos se introducen en el transmisor. Por cada símbolo, siguiendo el árbol de estados, se envía un dato por cada antena.
Se implementa el canal de modo que a la antena de recepción le llega el símbolo que transmitió su pareja en el transmisor más una parte correspondiente al enviado por la antena "cruzada".
El canal es una matriz de dos elementos, donde cada hi1 representa la ganancia entre la antena transmisora i y la receptora.
Se suponen ruido y canal constantes durante cada trama.

trama: conjunto de datos a transmitir, símbolos 8PSK.
Es: energía de símbolo.
No: potencia de ruido.
SNR: relación señal a ruido en recepción.

antrx1: señal recibida en el receptor.
h11e: coeficiente escalado del canal.
h21e: coeficiente escalado del canal.
C8PSK: matriz de las posibles transiciones entre estados.
anttx1: señal emitida por la antena 1 del transmisor.
anttx2: señal emitida por la antena 2 del transmisor.
n1: componente de ruido del canal.
signal1: señal emitida modificada por el canal pero sin el efecto del ruido.

Se parte del estado 0, estado inicial.
Para cada símbolo a transmitir se estudia con que estado corresponde y cuales son los valores que deben transmitirse por cada antena.

[diagrama] = *estados8PSK;*
[C8PSK] = *conversion* (diagrama);
[anttx1,anttx2] = *transmision* (trama,C8PSK);

En la antena receptora se recibe:
antrx1=h11*anttx1+h21*anttx2+ruido
Una vez modelados ruido y canal, solo los queda decidir cual es la trama que ha sido enviada, siguiendo un criterio de métrica mínima.

```
long=length(anttx1);
```

long es la longitud de la trama recibida en cada antena.

```
sig=sqrt(.5);

CANAL REAL
h11a= ruido (sig,long);
h21a= ruido (sig,long);
h11=h11a(1);
h21=h21a(1);

CANAL IDEAL
h11=ones(1,long);
h21=zeros(1,long);
```

El ruido es gaussiano, de media cero y potencia es No/2 por dimensión.

```
desvruido=sqrt(No);

CANAL RUIDOSO
n1= ruido (desvruido,long);

CANAL NO RUIDOSO
n1=zeros(1,long);

for i=1:long
  signal1(i)=sqrt(Es/2)*(h11*anttx1(i)+h21*anttx2(i));
end;
[h11e,h21e] = escalado (signal1,h11,h21,No,SNR);
for i=1:long
  antrx1(i)=sqrt(Es/2)*(h11e*anttx1(i)+h21e*anttx2(i))+n1(i);
end;
```

[vdecod,rutas,tramadecod,H] = *viterbisoft8psk8estados* (C8PSK,antrx1,Es,h11e,h21e);

Es la función que implementa la descodificación de Viterbi con decisión blanda.
Recibe señal entrante en la antena y a partir de ella, por comparación con el árbol de estados, decide cual es la trama transmitida usando el criterio de métrica mínima.

Para cada símbolo, partiendo del estado cero se compara con toda la fila de posibles transiciones. Se decide en función del que tenga métrica mínima y se averigua el estado al que se llega. Se queda con cuatro estados supervivientes, uno correspondiente a cada uno de los caminos entrantes en cada estado de llegada. Se guarda la transición en la matriz donde se conservará la ruta seguida. En el nuevo estado se vuelve a hacer lo mismo, buscando las transiciones que den menor métrica.

C8PSK: matriz que contiene las transiciones entre estados.
antrx1: vector de símbolos recibidos.
Es: energía de símbolo.
hije: coeficientes escalados del canal.

vdecod: vector de datos descodificados en formato binario.
rutas: matriz que contiene los caminos supervivientes.
tramadecod:vector de datos descodificados en formato complejo.
H: matriz de los coeficientes del canal.

tam es la longitud de la trama entrante a cada una de las antenas

```
tam=length(antrx1);
rutas=zeros(8,4);
rutasaux=rutas;
estproc=[1:8];
mtotal=zeros(1,8);
maux=zeros(8,8);
m=zeros(8,8);
k=1;
for j=1:tam                                          Recorriendo la longitud de la trama recibida
   H=[h11e;h21e];
   for h=1:8;                                        Para todos los estados de los que se procede
     ind=1;                                 Recorre todas las transiciones del estado h a todos los demás
     for i=1:2:15;
        r=[antrx1(j)];
        c=[C8PSK(h,i) C8PSK(h,i+1)];
        comp=sqrt(Es/2)*c*H;                        Se tiene conocimiento del canal al multiplicar *H
        metric=(r-comp)*(r-comp)';
        m(h,ind)=metric;
        ind=ind+1;
     end;
   end;
```

En este momento se tiene, para cada iteración, una matriz de métricas, debe elegirse a los supervivientes una vez analizado cada dato de entrada. Hay que buscar el mínimo por columnas.

```
  for i=1:8
     maux(i,:)=m(i,:)+mtotal(i);
  end;
  for i=1:8
     col=maux(:,i);
     aux=find(col==min(col));
```

Da los índices de métrica mínima , quedandose con el primero de ellos, que se corresponde con el estado desde el que se parte para llegar al estado i con menos coste.

```
          estproc(i)=aux(1);
```

En este punto se tiene la mínima transición de cualquier estado al estado i es decir, una transición superviviente. Se actualiza la métrica total.

```
     est=estproc(i);
     mtotal(i)=maux(est,i);
     l=2*(i-1);
     rutas(i,:)=rutasaux(est,:);
```

```
    rutas(i,k:k+1)=C8PSK(est,l+1:l+2);          Camino superviviente que llega al estado i
  end;
  k=k+2;
  rutas=[rutas,zeros(8,2)];
  rutasaux=rutas;
end;
indsuper=find(mtotal==min(mtotal));                    Índice que da la métrica mínima
long=length(rutas(indsuper,:));
vdecod=rutas(indsuper,1:long-2);                                  Camino mínimo
[vaux]=conversioninversa(vdecod);
lon=length(vaux);
ind=4;
for i=1:3:(lon/2)-1
  tramadecod(i:i+2)=vaux(ind:ind+2);
  ind=ind+6;
end;
```

7.1.2.3. UNA ANTENA EN TRANSMISIÓN Y UNA EN RECEPCIÓN

[salida] = *sistema1antena8PSK* (entrada,Es,No,SNR);

Función que implementa todo el sistema de transmisión, recepción y descodificacion de la trama de bits de entrada.

entrada: secuencia de bits a transmitir.
Es: energia de símbolo.
No: potencia de ruido.
SNR: relación señal a ruido en recepción.

salida: secuencia de bits descodificados.

Convierte la trama de bits de entrada a formato complejo, parte real, parte imaginaria.

[trama]=conversion(entrada);

Con esta función se almacena, en formato binario, el árbol generador del código de Trellis para una antena receptora y otra transmisora.

[antrx,C8PSK,anttx,n,signal,he] = *diversidad* (trama,Es,No,SNR);

[vdecod,rutas,salida] = *viterbisoft8psk8estados* (C8PSK,antrx,Es,he);

[antrx,C8PSK,anttx,n,signal,he] = *diversidad* (trama,Es,No,SNR);

Se diseña tanto la transmisión como la recepción y posterior decisión y descodificación.
Se desea transmitir una secuencia de símbolos 8PSK en formato complejo. Dichos datos se introducen en el transmisor. Por cada símbolo, siguiendo el árbol de estados, se envía un dato por la antena. Se supone que el ruido y el canal permanecen constantes durante cada trama.

trama: conjunto de datos a transmitir, símbolos 8PSK.
Es: energía de símbolo.
No: potencia de ruido.

```
SNR: relación señal a ruido en recepción.

antrx: vector de datos recibidos en la antena del receptor.
C8PSK: matriz que contiene todas las transiciones posibles del sistema.
anttx: vector de datos emitido por la antena transmisora.
n: componente de ruido blanco, gaussiano y aditivo.
signal: señal recibida sin la componente de ruido.
he: coeficiente del canal escalado.

Se parte del estado 0, estado inicial.
Para cada símbolo a transmitir se ve con que estado corresponde y cuales son los valores que se
deben transmitir por la antena.

[diagrama] = estados118PSK;
[C8PSK] = conversion (diagrama);

En la antena receptora se recibe:
antrx=h*anttx+ruido
Una vez modelados ruido y canal, sólo queda decidir cual es la trama que ha sido enviada,
siguiendo un criterio de métrica mínima.

anttx=trama;
long=length(anttx);

long es la longitud de la trama recibida en la antena.

sig=sqrt(.5);

CANAL REAL
ha= ruido (sig,long);
h=ha(1);

CANAL IDEAL
h=ones(1,long);

El ruido es constante a lo largo de una trama y sabiendo, que es gaussiano, de media cero y que
su potencia es No/2 por dimensión.

desvruido=sqrt(No);

CANAL RUIDOSO
n= ruido (desvruido,long);

CANAL NO RUIDOSO
n=zeros(1,long);

for i=1:long
   signal(i)=sqrt(Es/2)*(h*anttx(i));
end;
[he]= escalado (signal,h,No,SNR);
for i=1:long
   antrx(i)=sqrt(Es/2)*(he*anttx(i))+n(i);
```

```
end;
```

Una vez tenemos la señale a la entrada de la antena, solo queda aplicar la descodificación blanda de Viterbi.

[vdecod,rutas,tramadecod] = *viterbisoft8psk8estados* (C8PSK,antrx,Es,he);

Es la función que implementa la decodificación de Viterbi con decisión blanda.
Recibe la señale entrante en la antena y a partir de ella, por comparación con el árbol de estados, decide cual es la trama transmitida usando el criterio de métrica mínima.

C8PSK: matriz que contiene el diagrama de estados.
antrx: vector de símbolos recibidos.
Es: energía de símbolo.
he: coeficiente escalado del canal.

vdecod: vector de datos descodificados en formato complejo.
rutas: matriz que contiene los caminos supervivientes.
tramadecod: vector de datos descodificados en formato binario.

Se toman los símbolos recibidos de uno en uno. Partiendo del estado cero, comparo con toda la fila de posibles transiciones de estados. Se decide en función del que tenga métrica mínima y se almacena el estado al que se llega. Se obtienen ocho caminos supervivientes, correspondientes a uno de los caminos entrantes en cada estado de llegada. Se guarda la transición en la matriz donde se conservará la ruta seguida. En el nuevo estado se vuelve a hacer lo mismo, buscando las transiciones que den menor métrica.

tam es la longitud de la trama entrante a la antena.

```
tam=length(antrx);
rutas=zeros(8,1);
rutasaux=rutas;
estproc=[1:8];
mtotal=zeros(1,8);
maux=zeros(8,8);
m=zeros(8,8);
k=1;

for j=1:tam                                   Recorriendo la longitud de la trama recibida
   for h=1:8;                                 Para todos los estados de los que se procede
     ind=1;                                   Recorre todas las transiciones del estado h a todos los demás
     for i=1:8;
        r=antrx(j);
        c=C8PSK(h,i);
        comp=sqrt(Es/2)*c*he;
        metric=(r-comp)*(r-comp)';
        m(h,ind)=metric;
        ind=ind+1;
     end;
   end;
```

En este momento se tiene, para cada iteración, una matriz de métricas. Hay que elegir a los supervivientes una vez analizado cada dato de entrada. Hay que buscar el mínimo por columnas.

```
  for i=1:8
    maux(i,:)=m(i,:)+mtotal(i);
  end;
  for i=1:8
    col=maux(:,i);
    aux=find(col==min(col));
```

Da los índices de métrica mínima, quedándose con el primero de ellos, que se corresponde con el estado desde el que se parte para llegar al estado i con menos coste.

```
        estproc(i)=aux(1);
```

En este punto se tiene la mínima transición de cualquier estado al estado i, es decir, una transición superviviente.
Se actualiza la metrica total.

```
    est=estproc(i);
    mtotal(i)=maux(est,i);
    rutas(i,:)=rutasaux(est,:);
    rutas(i,k)=C8PSK(est,i);                    Camino superviviente que llega al estado i
  end;
  k=k+1;
  rutas=[rutas,zeros(8,1)];
  rutasaux=rutas;
end;
indsuper=find(mtotal==min(mtotal));             Índice que da la métrica mínima
long=length(rutas(indsuper,:));
vdecod=rutas(indsuper,1:long-1);
[tramadecod]=conversioninversa(vdecod);
```

7.1.3. SISTEMA 4PSK CON 8 ESTADOS

7.1.3.1. DOS ANTENAS EN TRANSMISIÓN Y DOS EN RECEPCIÓN

[salida] = *sistema2antenas4psk8estados* (entrada,Es,No,SNR);

Función que implementa el sistema 4PSK con 8 estados en el caso de que se tengan dos antenas de transmisión y dos de recepción.

entrada: secuencia de bits de entrada al sistema.
Es: energía de símbolo.
No: densidad de potencia de ruido.
SNR: relación señal a ruido por antena en recepción.

salida: secuencia de bits descodificados correspondientes al procesado de la señal de entrada

[anttx1,anttx2,TRANSICIONES,estadoprocedente] = *transmision* (entrada);

[anttx1]= *conversion* (anttx1);
[anttx2]= *conversion* (anttx2);

[antrx1,antrx2,h11e,h12e,h21e,h22e]= *canal* (anttx1,anttx2,Es,No,SNR);

[estados,secuenciaestados]= *Viterbi* (h11e,h12e,h21e,h22e,antrx1,antrx2,TRANSICIONES,Es);
[salida]= *Descodificacion* (estados);

[anttx1,anttx2,TRANSICIONES,estadoprocedente] = *transmision* (entrada);

Función que implementa la codificación de los bits de entrada. Se recibe una secuencia de unos y ceros y se decide en función de ellos y de la matriz de transiciones del código, cuales son las señales que se envían por cada antena.

entrada: secuencia de bits de entrada al sistema.

anttxi: señales binarias que se transmiten por cada antena.
TRANSICIONES: matriz que contiene las transiciones del diagrama de Trellis para un sistema 4PSK con 8 estados.

```
TRANSICIONES=[0 0 0 1 0 2 0 3;
              1 0 1 1 1 2 1 3;
              2 0 2 1 2 2 2 3;
              3 0 3 1 3 2 3 3;
              2 2 2 3 2 0 2 1;
              3 2 3 3 3 0 3 1;
              0 2 0 3 0 0 0 1;
              1 2 1 3 1 0 1 1];

    estadoprocedente=1;
    l=1;
    indice=1;
    tam=length(entrada);
    for k=1:2:tam;
      dato=[entrada(k) entrada(k+1)];
      if and(dato(1)==0,dato(2)==0)
        indice=1;
      end;
      if and(dato(1)==0,dato(2)==1)
        indice=3;
      end;
      if and(dato(1)==1,dato(2)==1)
        indice=5;
      end;
      if and(dato(1)==1,dato(2)==0)
        indice=7;
      end;
      anttx1(l)=TRANSICIONES(estadoprocedente,indice);
      anttx2(l)=TRANSICIONES(estadoprocedente,indice+1);
      l=l+1;
```

El proceso se realiza en dos pasos, dependiendo del estado del que partamos. Con este código no todas las transiciones entre estado son posibles, sólo algunas están permitidas.
En particular, si partimos de un estado impar podemos llegar a los 4 primeros estados, si partimos desde un estado par llegamos a los 4 últimos.
Como la matriz Transiciones es 8x4, hay que saber dependiendo de la fila y columna donde nos encontremos a que estado o transición entre estados nos referimos.

```
If or(or((estadoprocedente==1), (estadoprocedente==3)), or((estadoprocedente==5),
 (estadoprocedente==7)))
   if indice==1
      estadoprocedente=1;
   end;
   if indice==3
      estadoprocedente=2;
   end;
   if indice==5
      estadoprocedente=3;
   end;
   if indice==7
      estadoprocedente=4;
   end;

 else
   if indice==1
      estadoprocedente=5;
   end;
   if indice==3
      estadoprocedente=6;
   end;
   if indice==5
      estadoprocedente=7;
   end;
   if indice==7
      estadoprocedente=8;
   end;
 end;
end;
```

[estados,secuenciaestados]=*Viterbi* (h11e,h12e,h21e,h22e,antrx1,antrx2,TRANSICIONES,Es);

hije: coeficientes escalados del canal.
antrxi: señal recibida en la antena i del receptor, formato complejo.
TRANSICIONES: matriz que contiene todas las transiciones posibles entre estados.
Es: energía de símbolo.

estados: camino óptimo elegido entre todos los supervivientes. Secuencia de estados de métrica mínima.
secuenciaestados: matriz que contiene todos los posibles caminos supervivientes en función de la secuencia de estados seguida.

```
tam=length(antrx1);
k=1;
```

```
H=[h11e h12e;h21e h22e];
mtotal=zeros(1,8);
maux=zeros(8,4);
m=zeros(8,4);
estproc=ones(1,8);
secuenciaestados=zeros(8,2);
secestaux=zeros(8,2);
[TRANSICIONES]= conversionmatrices (TRANSICIONES);
```

Tengamos en cuenta que nuestro descodificador de Viterbi parte del estado 1 y vuelve a el.
Así se ha obligado que sea al definir la señal entrada.

PASO INICIAL

Como partimos del estado 1 solo podemos pasar a los estados 1, 2, 3 y 4. El análisis de la primera transición requiere un estudio distinto del de una transición intermedia general.

```
dato=[antrx1(1) antrx2(1)];
for i=1:2:7
  r=dato;
  c=[TRANSICIONES(1,i) TRANSICIONES(1,i+1)];
  comp=sqrt(Es/2)*c*H;
  metric=(r-comp)*(r-comp)';
  mtotal(k)=metric;
  k=k+1;
end;
secuenciaestados=[1 1;1 2; 1 3; 1 4 ; 0 0 ; 0 0 ;0 0 ;0 0];
secestaux=secuenciaestados;
```

El proceso se realiza en dos pasos, dependiendo del estado del que partamos. Con este código no todas las transiciones entre estado son posibles, sólo algunas están permitidas.
En particular, si partimos de un estado impar podemos llegar a los 4 primeros estados, si partimos desde un estado par llegamos a los 4 últimos.
Como la matriz Transiciones es 8x4, hay que saber dependiendo de la fila y columna donde nos encontremos a que estado o transición entre estados nos referimos.

PASO 2

Del mismo modo, el segundo paso tampoco es exactamente igual que el general puesto que la matriz secuenciaestados no solo tiene datos adecuados en su mitad superior.

```
dato=[antrx1(2) antrx2(2)];
for h=1:4
  ind=1;
  for i=1:2:7
    r=dato;
    c=[TRANSICIONES(h,i) TRANSICIONES(h,i+1)];
    comp=sqrt(Es/2)*c*H;
    metric=(r-comp)*(r-comp)';
    m(h,ind)=metric;
    ind=ind+1;
  end;
```

```
end;
```

ASIGNACIÓN DE CAMINOS Y MÉTRICAS

```
maux(1,:)=m(1,:)+mtotal(1,1:4);
for i=2:4
   maux(i,:)=m(i,:);
end;
for i=1:4
   col=maux(1:2:3,i);
   aux=find(col==min(col));
   estproc(i)=aux(1);
   est=2*estproc(i)-1;
   mtotal(i)=maux(est,i);
   secuenciaestados(i,1:2)=secestaux(est,:);
   secuenciaestados(i,3)=i;
  end;
for i=1:4
   ind=i+4;
   col=maux(2:2:4,i);
   aux=find(col==min(col));
   estproc(ind)=aux(1);
   est=2*estproc(ind);
   mtotal(ind)=maux(est,i);
   secuenciaestados(ind,1:2)=secestaux(est,:);
   secuenciaestados(ind,3)=ind;
end;
secestaux=secuenciaestados;
```

PASO GENERAL

Para cualquier dato recibido que hay que analizar se compara, según el estado del que procede y por tanto, los estados a los que puede llegar, con las transiciones posibles. Se calculan las métricas de todos los caminos posibles y nos quedamos, como siempre, con uno por cada estado de llegada, el que de métrica mínima.

```
for j=3:tam
  dato=[antrx1(j) antrx2(j)];
  for h=1:8
     ind=1;
     for i=1:2:7
        r=dato;
        c=[TRANSICIONES(h,i) TRANSICIONES(h,i+1)];
        comp=sqrt(Es/2)*c*H;
        metric=(r-comp)*(r-comp)';
        m(h,ind)=metric;
        ind=ind+1;
     end;
  end;
```

ASIGNACIÓN DE CAMINOS Y MÉTRICAS

```
 for i=1:8
    maux(i,:)=m(i,:)+mtotal(i);
 end;
 for i=1:4
    col=maux(1:2:7,i);
    aux=find(col==min(col));
    estproc(i)=aux(1);
    est=2*estproc(i)-1;
    mtotal(i)=maux(est,i);
    secuenciaestados(i,1:j)=secestaux(est,:);
    secuenciaestados(i,j+1)=i;
 end;
 for i=1:4
    ind=i+4;
    col=maux(2:2:8,i);
    aux=find(col==min(col));
    estproc(ind)=aux(1);
    est=2*estproc(ind);
    mtotal(ind)=maux(est,i);
    secuenciaestados(ind,1:j)=secestaux(est,:);
    secuenciaestados(ind,j+1)=ind;
 end;
 secestaux=secuenciaestados;
end;
```

El camino óptimo es aquel cuya métrica sea la menor de todos los supervivientes.

```
indsuper=find(mtotal==min(mtotal));
ind=indsuper(1);
estados=secuenciaestados(ind,:);
```

7.1.3.2. DOS ANTENAS EN TRANSMISIÓN Y UNA EN RECEPCIÓN

[salida] = *sistema1antena4psk8estados* (entrada,Es,No,SNR);

Función que implementa el sistema 4PSK con 8 estados en el caso de que se tengan dos antenas de transmisión y una de recepción.

entrada: secuencia de bits de entrada al sistema.
Es: energía de símbolo.
No: densidad de potencia de ruido.
SNR: relación señal a ruido por antena en recepción.

salida: secuencia de bits descodificados correspondientes al procesado de la señal de entrada

```
[anttx1,anttx2,TRANSICIONES]= transmision (entrada);

[anttx1]= conversion (anttx1);
[anttx2]= conversion (anttx2);

[antrx1,h11e,h21e]= canal (anttx1,anttx2,Es,No,SNR);
```

[estados,secuenciaestados]= *Viterbi* (h11e,h21e,antrx1,TRANSICIONES,Es);
[salida]= *Descodificacion* (estados);

[anttx1,anttx2,TRANSICIONES]= *transmision* (entrada);

Función que implementa la codificación de los bits de entrada. Se recibe una secuencia de unos y ceros y se decide en función de ellos y de la matriz de transiciones del código, cuales son las señales que se envían por cada antena.

entrada: secuencia de bits de entrada al sistema.

anttxi: señales binarias que se transmiten por cada antena.
TRANSICIONES: matriz que contiene las transiciones del diagrama de Trellis para un sistema 4PSK con 8 estados.

```
TRANSICIONES=[0 0 0 1 0 2 0 3;
                1 0 1 1 1 2 1 3;
                2 0 2 1 2 2 2 3;
                3 0 3 1 3 2 3 3;
                2 2 2 3 2 0 2 1;
                3 2 3 3 3 0 3 1;
              0 2 0 3 0 0 0 1;
              1 2 1 3 1 0 1 1];

      estadoprocedente=1;
      l=1;
      indice=1;
      tam=length(entrada);
      for k=1:2:tam;
        dato=[entrada(k) entrada(k+1)];
        if and(dato(1)==0,dato(2)==0)
          indice=1;
        end;
        if and(dato(1)==0,dato(2)==1)
          indice=3;
        end;
        if and(dato(1)==1,dato(2)==1)
          indice=5;
        end;
        if and(dato(1)==1,dato(2)==0)
          indice=7;
        end;

        anttx1(l)=TRANSICIONES(estadoprocedente,indice);
        anttx2(l)=TRANSICIONES(estadoprocedente,indice+1);
        l=l+1;
```

El proceso se realiza en dos pasos, dependiendo del estado del que partamos. Con este código no todas las transiciones entre estado son posibles, sólo algunas están permitidas.
En particular, si partimos de un estado impar podemos llegar a los 4 primeros estados, si partimos desde un estado par llegamos a los 4 últimos.

Como la matriz Transiciones es 8x4, hay que saber dependiendo de la fila y columna donde nos encontremos a que estado o transición entre estados nos referimos.

```
        if or(or((estadoprocedente==1), (estadoprocedente==3)), or((estadoprocedente==5),
          (estadoprocedente==7)))
          if indice==1
            estadoprocedente=1;
          end;
          if indice==3
            estadoprocedente=2;
          end;
          if indice==5
            estadoprocedente=3;
          end;
          if indice==7
            estadoprocedente=4;
            end;
        else
          if indice==1
            estadoprocedente=5;
          end;

          if indice==3
            estadoprocedente=6;
          end;
          if indice==5
            estadoprocedente=7;
          end;
          if indice==7
            estadoprocedente=8;
          end;
        end;
      end;
```

[estados,secuenciaestados] = *Viterbi* (h11e,h21e,antrx1,TRANSICIONES,Es);

hi1e: coeficientes escalados del canal.
antrx1: señal recibida en la antena del receptor, formato complejo.
TRANSICIONES: matriz que contiene todas las transiciones posibles entre estados.
Es: energía de símbolo.

estados: camino óptimo elegido entre todos los supervivientes. Secuencia de estados de métrica mínima.
secuenciaestados: matriz que contiene todos los posibles caminos supervivientes en función de la secuencia de estados seguida.

```
tam=length(antrx1);
k=1;
H=[h11e;h21e];
mtotal=zeros(1,8);
maux=zeros(8,4);
m=zeros(8,4);
```

```
estproc=ones(1,8);
secuenciaestados=zeros(8,2);
secestaux=zeros(8,2);
[TRANSICIONES]=conversionmatrices(TRANSICIONES);
```

Tengamos en cuenta que nuestro descodificador de Viterbi parte del estado 1 y vuelve a el.
Así se ha obligado que sea al definir la señal entrada.

PASO INICIAL

Como partimos del estado 1 solo podemos pasar a los estados 1, 2, 3 y 4. El análisis de la primera transición requiere un estudio distinto del de una transición intermedia general.

```
dato=[antrx1(1)];
for i=1:2:7
  r=dato;
  c=[TRANSICIONES(1,i) TRANSICIONES(1,i+1)];
  comp=sqrt(Es/2)*c*H;
  metric=(r-comp)*(r-comp)';
  mtotal(k)=metric;
  k=k+1;
end;
secuenciaestados=[1 1;1 2; 1 3; 1 4 ; 0 0 ; 0 0 ;0 0 ;0 0];
secestaux=secuenciaestados;
```

El proceso se realiza en dos pasos, dependiendo del estado del que partamos. Con este código no todas las transiciones entre estado son posibles, sólo algunas están permitidas.
En particular, si partimos de un estado impar podemos llegar a los 4 primeros estados, si partimos desde un estado par llegamos a los 4 últimos.
Como la matriz Transiciones es 8x4, hay que saber dependiendo de la fila y columna donde nos encontremos a que estado o transición entre estados nos referimos.

PASO 2

Del mismo modo, el segundo paso tampoco es exactamente igual que el general puesto que la matriz secuenciaestados no solo tiene datos adecuados en su mitad superior.

```
dato=[antrx1(2)];
for h=1:4
  ind=1;
  for i=1:2:7
    r=dato;
    c=[TRANSICIONES(h,i) TRANSICIONES(h,i+1)];
    comp=sqrt(Es/2)*c*H;
    metric=(r-comp)*(r-comp)';
    m(h,ind)=metric;
    ind=ind+1;
  end;
end;
```

ASIGNACION DE CAMINOS Y MÉTRICAS

```
maux(1,:)=m(1,:)+mtotal(1,1:4);
for i=2:4
   maux(i,:)=m(i,:);
end;
for i=1:4
   col=maux(1:2:3,i);
   aux=find(col==min(col));
   estproc(i)=aux(1);
   est=2*estproc(i)-1;
   mtotal(i)=maux(est,i);
   secuenciaestados(i,1:2)=secestaux(est,:);
   secuenciaestados(i,3)=i;
  end;
for i=1:4
   ind=i+4;
   col=maux(2:2:4,i);
   aux=find(col==min(col));
   estproc(ind)=aux(1);
   est=2*estproc(ind);
   mtotal(ind)=maux(est,i);
   secuenciaestados(ind,1:2)=secestaux(est,:);
   secuenciaestados(ind,3)=ind;
  end;
  secestaux=secuenciaestados;

 PASO GENERAL

Para cualquier dato recibido que hay que analizar se compara, según el estado del que procede y
por tanto, los estados a los que puede llegar, con las transiciones posibles.
Se calculan las métricas de todos los ca,inos posibles y nos quedamos, como siempre, con uno por
cada estado de llegada, el que de métrica mínima.

for j=3:tam
  dato=[antrx1(j)];
  for h=1:8
     ind=1;
     for i=1:2:7
        r=dato;
        c=[TRANSICIONES(h,i) TRANSICIONES(h,i+1)];
        comp=sqrt(Es/2)*c*H;
        metric=(r-comp)*(r-comp)';
        m(h,ind)=metric;
        ind=ind+1;
     end;
  end;

  ASIGNACION DE CAMINOS Y MÉTRICAS

  for i=1:8
     maux(i,:)=m(i,:)+mtotal(i);
  end;
  for i=1:4
```

```
    col=maux(1:2:7,i);
    aux=find(col==min(col));
    estproc(i)=aux(1);
    est=2*estproc(i)-1;
    mtotal(i)=maux(est,i);
    secuenciaestados(i,1:j)=secestaux(est,:);
    secuenciaestados(i,j+1)=i;
  end;
  for i=1:4
    ind=i+4;
    col=maux(2:2:8,i);
    aux=find(col==min(col));
    estproc(ind)=aux(1);
    est=2*estproc(ind);
    mtotal(ind)=maux(est,i);
    secuenciaestados(ind,1:j)=secestaux(est,:);
    secuenciaestados(ind,j+1)=ind;
  end;
  secestaux=secuenciaestados;
end;
```

El camino óptimo es aquel cuya métrica sea la menor de todos los supervivientes.

```
indsuper=find(mtotal==min(mtotal));
ind=indsuper(1);
estados=secuenciaestados(ind,:);
```

7.2. CODIFICACIÓN ESPACIO-TEMPORAL DE BLOQUES

En los apartados siguientes se muestran las funciones más significativas de cada sistema.

7.2.1. DOS ANTENAS EN TRANSMISIÓN Y UNA EN RECEPCIÓN

[salida,h1e,h2e] = *stbc* (entrada,Es,No,SNR);

Función que implementa un sistema que utiliza códigos de bloque espacio-temporales.
Se tienen dos antenas en transmisión y una en recepción.
Los símbolos codificados según la constelación 8PSK se toman por parejas y se envían en dos intervalos consecutivos de tiempo segun la siguiente la ley:
r1=h1*c1+h2*c2+n1
r2=-h1*c2'+h2*c1'+n2
c=[c1;c2] vector que contiene la pareja de símbolos a analizar
H=[h1 h2;h2' -h1'] matriz que implementa el canal.
Modelaremos el canal como una gausiana de media cero y varianza 0.5
n=[n1;n2']es ruido, modelado por gausianas independientes de media cero y varianza No.
r=[r1;r2'] vector que contiene la señal que llega al receptor en dos instantes consecutivos
En el receptor, debido a que el canal es ortogonal, se pueden desacoplar las dos ramas y decidir por separado a que símbolo corresponde cada dato recibido.

entrada: vector de símbolos binarios 8PSK de entrada al sistema.
Es: energía de símbolo.

```
No: densidad de potencia de ruido.
SNR: relación señal a ruido en la antena de recepción.

salida: vector binario obtenido tras la descodificación.

Convierto mis datos de entrada a formato complejo según la codificación 8PSK.

[tram]= conversion (entrada);                    tram es la secuencia con energía unidad
trama=tram*sqrt(Es);                  trama es la secuencia con la energía que indica el parámetro
long=length(trama);
[n1]= ruido (sqrt(No),long/2);
[n2]= ruido (sqrt(No),long/2);

se cumple que h1 y h2 son constantes durante dos periodos de símbolo consecutivos

[h1]= ruido (sqrt(.5),long/2);
[h2]= ruido (sqrt(.5),long/2);

C es el vector que contiene en formato complejo los 8 símbolos de la constelación con energía Es.

bin=[ 0 0 0 0 0 1 0 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 0 0];
[aux]=conversion(bin);
C=aux*sqrt(Es);
j=1;
for i=1:2:long-1
   c1=trama(i);
   c2=trama(i+1);
   p=h1(j)*h1(j)'.'+h2(j)*h2(j)'.';
   signal1(j)=h1(j)*c1+h2(j)*c2;
   signal2(j)=-h1(j)*c2'.'+h2(j)*c1'.';
   j=j+1;
end;

Obtenemos la señal que llega a la antena sin ruido.

j=1;
for i=1:2:long-1
   signal(i)=signal1(j);
   signal(i+1)=signal(j);
   n(i)=n1(j);
   n(i+1)=n2(j);
   j=j+1;
end;

Sabiendo que el ruido tendrá varianza No, escalamos dicha señal de modo que tenga media cero y
varianza que cumpla la SNR requerida.

signal=signal-mean(signal);
vars=var(signal);
signal=sqrt(No/vars)*signal;
[h1e,h2e]= escalado (signal,h1,h2,No,SNR);
```

Obtenemos la señal de ruido total que llega a la antena de recepción.

```
j=1;
for i=1:2:long-1
   n(i)=n1(j);
   n(i+1)=n2(j);
   j=j+1;
end;
```

Escalamos el ruido para que tenga varianza No.

```
n=n-mean(n);
varn=var(n);
alfa=sqrt(No/varn);
n=alfa*n;
n1=alfa*n1;
n2=alfa*n2;
j=1;
for i=1:2:long-1
      c1=trama(i);
      c2=trama(i+1);
      r1=h1e(j)*c1+h2e(j)*c2+n1(j);
     r2=-h1e(j)*c2'.'+h2e(j)*c1'.'+n2(j);
     H=[h1e(j) h2e(j); h2e(j)'.' -h1e(j)'.'];
     c=[c1;c2];
     r=[r1;r2'.'];
     p=h1e(j)*h1e(j)'.'+h2e(j)*h2e(j)'.';
  n=[n1(j);n2(j)'.'];
     rm=H'*r;      '.' es conjugado
```

La regla de descodificación es: cm=argmin(norma^2(rm-p*cm)), donde cm es el vector de las parejas de símbolos posibles
Como las ramas se desacoplan, decidiremos cada símbolo independientemente.

```
     metrica1=zeros(1,8);
     metrica2=zeros(1,8);
     for k=1:8;
        cm=[C(k);C(k)];
        metrica=rm-p*cm;
        metrica1(k)=metrica(1)*metrica(1)';
        metrica2(k)=metrica(2)*metrica(2)';
     end;
```

En este punto tenemos en los vectores metricai, los valores que toma la regla de decisión en cada caso. Ahora basta quedarse con la métrica menor en cada caso.

```
   aux1=find(metrica1==min(metrica1));
   aux2=find(metrica2==min(metrica2));
```

En auxi tenemos los índices de métrica mínima, que se corresponden con unos símbolos determinados en formato complejo(vector C), o directamente con su equivalente en binario (vector bin).

```
  sal(i)=C(aux1(1));
  sal(i+1)=C(aux2(1));
  j=j+1;
end;
```

Como nuestra función ofrece los resultados en binario, los convertimos.
La función conversión inversa trata con símbolos de energía 1, hay que normalizar

```
sal=sal/sqrt(Es);
[salida]=conversioninversa(sal);
```

7.2.2. SISTEMA DE CANCELACIÓN DE INTERFERENCIAS

[salida,g11e,g22e,h22e,h11e,sig11,int11,sig11e,int11e] = *stbcinterferencia* (entrada1, entrada2,Es,No,SNR,SIR);

Función que implementa un sistema que utiliza códigos de bloque espacio-temporales.
Se tienen dos antenas en transmisión y dos en recepción para cada uno de los usuarios.
Tendremos un usuario que transmite la información que queremos recibir y otro interferente.
Los símbolos codificados según la constelación 8PSK se toman por parejas y se envían en dos intervalos consecutivos de tiempo según la siguiente la ley:
r1=h1*c1+h2*c2+n1
r2=-h1*c2'+h2*c1'+n2
Para el usuario interferente la ley es la misma:
r1=g1*s1+g2*s2+n1
r2=-g1*s2'+g2*s1'+n2
c=[c1;c2] vector que contiene la pareja de símbolos a analizar
s=[s1;s2] vector de símbolos interferentes
H=[h1 h2;h2' -h1'] matriz que implementa el canal para el usuario principal
G=[g1 g2;g2' -g1'] matriz que implementa el canal para el usuario interferente
Modelaremos el canal como una gaussiana de media cero y varianza 0.5
n=[n1;n2']es ruido, modelado por gaussianas independientes de media cero y varianza No.
r=[r1;r2'] vector que contiene la señal que llega al receptor en dos instantes consecutivos, incluye señal, interferencia y ruido.
En el receptor se implementa la solución MMSE para cancelación de interferencias.

entradai: vector de símbolos binarios 8PSK de entrada al sistema del usuario i.
Es: energía de símbolo.
No: densidad de potencia de ruido.
SNR: relación señal a ruido en la antena de recepción.
SIR: relación señal a interferencia.

salidaij: vector binario obtenido tras la descodificación, del usuario i por la antena de recepción j.

Convierto mis datos de entrada a formato complejo según la codificación 8PSK.

```
[tram1]=conversion(entrada1);                tram1 es la secuencia con energía unidad
trama1=tram1*sqrt(Es);          trama1 es la secuencia con la energía que indica el parámetro
long=length(trama1);
[tram2]=conversion(entrada2);                tram2 es la secuencia con energía unidad
trama2=tram2*sqrt(Es);          trama2 es la secuencia con la energía que indica el parámetro
```

```
[n11]= ruido (sqrt(No),long/2);
[n12]= ruido (sqrt(No),long/2);
[n21]= ruido (sqrt(No),long/2);
[n22]= ruido (sqrt(No),long/2);

n1=[n11;n12];  vector de ruido del sistema 1
n2=[n21;n22];  vector de ruido del sistema 2

Se cumple que hij y gij son constantes durante dos periodos de símbolo consecutivos

[h11]= ruido (sqrt(.5),long/2);
[h21]= ruido (sqrt(.5),long/2);
[h12]= ruido (sqrt(.5),long/2);
[h22]= ruido (sqrt(.5),long/2);

[g11]= ruido (sqrt(.5),long/2);
[g21]= ruido (sqrt(.5),long/2);
[g12]= ruido (sqrt(.5),long/2);
[g22]= ruido (sqrt(.5),long/2);

I=[1 0 0 0;0 1 0 0;0 0 1 0; 0 0 0 1];

C es el vector que contiene en formato complejo los 8 símbolos de la constelación con energía Es.

bin=[ 0 0 0 0 0 1 0 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 0 0];
[aux]=conversion(bin);
C=aux*sqrt(Es);

snr=10^((1/.5)/10);                                    snr valor decimal de la SNR en decibelios
j=1;
for i=1:2:long
  H1=[h11(j) h21(j);h21(j)'.' -h11(j)'.'];
  H2=[h12(j) h22(j);h22(j)'.' -h12(j)'.'];
  G1=[g11(j) g21(j);g21(j)'.' -g11(j)'.'];
  G2=[g12(j) g22(j);g22(j)'.' -g12(j)'.'];
  c1=trama1(i);
  c2=trama1(i+1);
  s1=trama2(i);
  s2=trama2(i+1);
  sig11(j)=h11(j)*c1+h21(j)*c2;
  sig12(j)=h21(j)'.'*c1-h11(j)'.'*c2;
  sig21(j)=h12(j)*c1+h22(j)*c2;
  sig22(j)=h22(j)'.'*c1-h12(j)'.'*c2;
  int11(j)=g11(j)*c1+g21(j)*c2;
  int12(j)=g21(j)'.'*c1-g11(j)'.'*c2;
  int21(j)=g12(j)*c1+g22(j)*c2;
  int22(j)=g22(j)'.'*c1-g12(j)'.'*c2;
  j=j+1;
end;

sig11p=sig11;
sig11=sig11-mean(sig11);
```

```
vars=var(sig11);
[h11e,h21e]= escalado (sig11,h11,h21,No,SNR);

sig21=sig21-mean(sig21);
vars=var(sig21);
[h12e,h22e]= escalado (sig21,h12,h22,No,SNR);


INR=SNR-SIR;                                    relación interferencia a ruido en dB. 10log(I/No)
int11=int11-mean(int11);
vars=var(int11);
[g11e,g21e]= interferencia2 (int11,int12,g11,g21,sig11,sig12,SIR);

int21=int21-mean(int21);
vars=var(int21);
[g12e,g22e]= interferencia2 (int21,int22,g12,g22,sig21,sig22,SIR);

j=1;
for i=1:2:long
   c1=trama1(i);
   c2=trama1(i+1);
   s1=trama2(i);
   s2=trama2(i+1);
   sig11e(j)=h11e(j)*c1+h21e(j)*c2;
   sig12e(j)=h21e(j)'.'*c1-h11e(j)'.'*c2;
   sig21e(j)=h12e(j)*c1+h22e(j)*c2;
   sig22e(j)=h22e(j)'.'*c1-h12e(j)'.'*c2;
   int11e(j)=g11e(j)*c1+g21e(j)*c2;
   int12e(j)=g21e(j)'.'*c1-g11e(j)'.'*c2;
   int21e(j)=g12e(j)*c1+g22e(j)*c2;
   int22e(j)=g22e(j)'.'*c1-g12e(j)'.'*c2;
   j=j+1;
end;
n1=[n11;n12];
n2=[n21;n22];

j=1;
for i=1:2:long
   H1=[h11e(j) h21e(j);h21e(j)'.' -h11e(j)'.'];
   H2=[h12e(j) h22e(j);h22e(j)'.' -h12e(j)'.'];
   G1=[g11e(j) g21e(j);g21e(j)'.' -g11e(j)'.'];
   G2=[g12e(j) g22e(j);g22e(j)'.' -g12e(j)'.'];
   c1=trama1(i);
   c2=trama1(i+1);
   s1=trama2(i);
   s2=trama2(i+1);
   c=[c1;c2];
   s=[s1;s2];
   H=[H1;H2];
   M=H*H'+1/snr*I;
   w1=(inv(M))*H(:,1);
   w2=(inv(M))*H(:,2);
```

```
  r1=H1*c+G1*s+n1(:,j);
  r2=H2*c+G2*s+n2(:,j);
  r=[r1;r2];
  aux1=w1'*r;
  aux2=w2'*r;

CALCULO DE LA METRICA
  metrica=zeros(8,8);
  for h=1:8
    for k=1:8
      caux1=C(h);
      caux2=C(k);
      metrica(k,h)=((aux1-caux1)*(aux1-caux1)'.')+((aux2-caux2)*(aux2-caux2)'.');
    end;
  end;
  [ind1,ind2]=find(metrica==min(min(metrica)));
  sal(i)=C(ind2(1));
  sal(i+1)=C(ind1(1));
  j=j+1;
end;
sal=sal/sqrt(Es);
[salida]= conversioninversa (sal);
```