CAPÍTULO 2

CÓDIGOS CONVOLUCIONALES

2.1. FUNCIONAMIENTO DEL CODIFICADOR

Los códigos convolucionales fueron introducidos por primera vez por Elias, en el año 1955, como alternativa al uso de códigos de bloques [9].La técnica de codificación convolucional está diseñada para reducir la probabilidad de error de transmisión en canales altamente ruidosos [9]. Un código convolucional se genera pasando la secuencia de información a transmitir a través de un registro de desplazamiento con un número finito de estados. En general, dicho registro consiste en *K* estados, cada uno de ellos con *k* bits. En cada instante, los bits de información se desplazan *k* posiciones en la estructura del registro. Además, existen *n* sumadores de módulo-2; cada uno de ellos selecciona ciertas posiciones del registro, y realiza una operación OR exclusiva[1] sobre el contenido de las mismas [10]; tal y como se muestra en la figura 2.1.

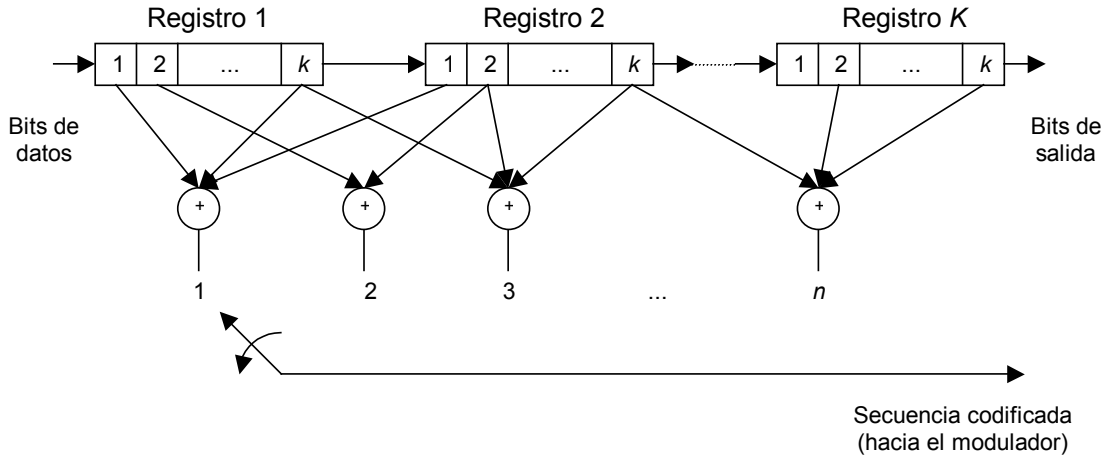


Figura 2.1.Codificador convolucional.

1 La operación OR exclusiva o XOR tiene la siguiente tabla de resultados. E_1 y E_2 son las entradas y S la salida.

E_1	E_2	0	0	0	1	1	0	1	1
S		0		1		1		0	

En cada instante, los bits de información se desplazan *k* posiciones en la estructura del registro. Se obtienen *n* bits de salida por cada *k* bits de entrada. En consecuencia, se define la **tasa del código** como

$$R_c = k/n \ . \tag{2.1}$$

El parámetro *K* se denomina **longitud reducida** del código convolucional y se define como el número de desplazamientos durante los cuales *k* bits de entrada tienen influencia sobre la salida codificada [9] .

Otra alternativa equivalente para su representación consiste en especificar un conjunto de *n* vectores, denominados **funciones generadoras**, (o simplemente generadores), uno por cada sumador de módulo-2. Cada vector tiene dimensión *Kk* y contiene las conexiones del codificador al sumador correspondiente. Un 1 en la *i*-ésima posición del vector indica que el correspondiente estado en el registro de desplazamiento está conectado al sumador, mientras que un 0 indica que no existe conexión.

Los códigos convolucionales pueden describirse, como los códigos de bloque, mediante su **matriz generadora**, **G**, que contiene por columnas, cada una de las funciones generadoras. Nótese que el conjunto de posibles palabras de código es el espacio expandido por las columnas de **G**. Dichas columnas forman una base que no es única para cualquier espacio lineal; es evidente que existen muchas matrices generadoras distintas que generan el mismo espacio de palabras de código [12].

A continuación se muestran algunos ejemplos que se irán desarrollando a lo largo del capítulo [10].

Considérese un codificador convolucional binario con *K* = 3, *k* = 1 y *n* = 3, tal y como se muestra en la figura 2.2.

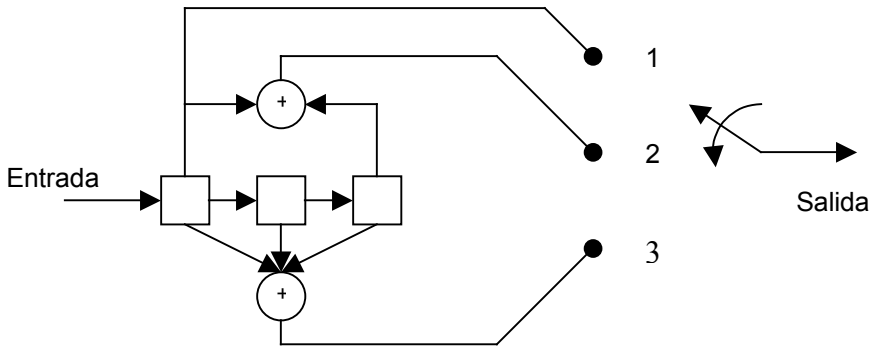


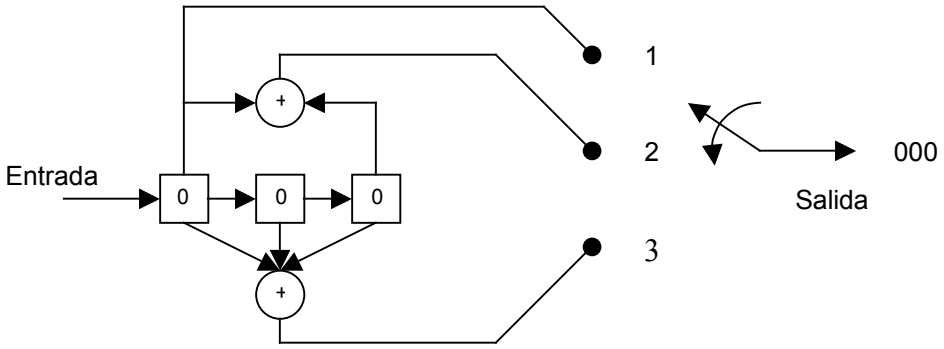
Figura 2.2.Codificador convolucional. *K* = 3, *k* = 1, *n* = 3.

Las funciones generadoras del código son en este caso, $\mathbf{g_1}$ = [1 0 0], $\mathbf{g_2}$ = [1 0 1], $\mathbf{g_3}$ = [1 1 1]. La matriz **G** contiene, por columnas, los generadores si el vector **r** es la secuencia de bits de entrada, la salida se obtiene como $\mathbf{s} = \mathbf{rG}^t$. Así, cada elemento del vector salida se consigue como $s_1 = \mathbf{r}\,\mathbf{g_1}$, $s_2 = \mathbf{r}\,\mathbf{g_2}$, $s_3 = \mathbf{r}\,\mathbf{g_3}$.

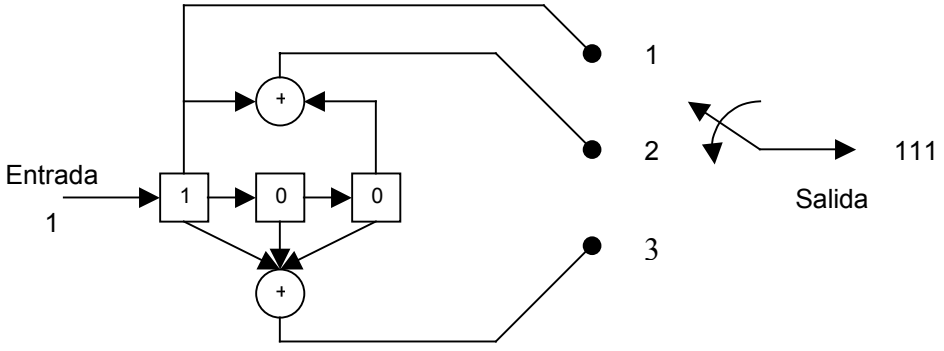
Inicialmente se supone que el registro de desplazamiento se encuentra en el estado todo a cero. Se desglosará el ejemplo viendo el estado en el que queda el registro y la salida cada vez que se introduce un bit de entrada.

Supongamos que la entrada al sistema es {0 1 0}.

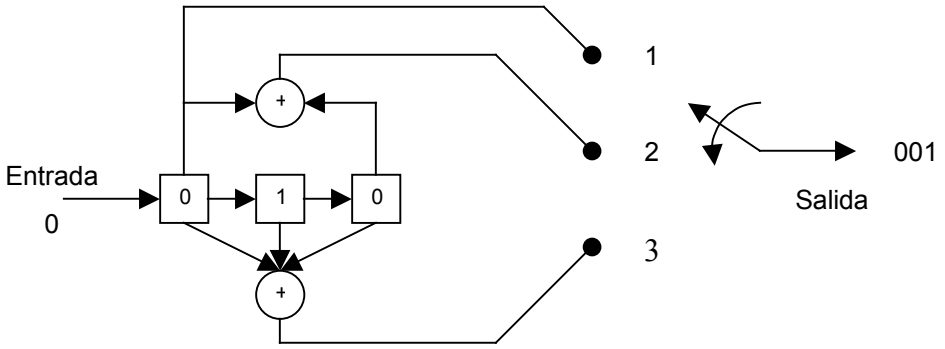
- Estado inicial



- Si el primer bit de entrada es un 1, entonces la salida es [1 1 1]



- Si el segundo bit de entrada es un 0, entonces la salida es [0 0 1]



- Si el bit de entrada es un 1, entonces la salida es [1 0 0]

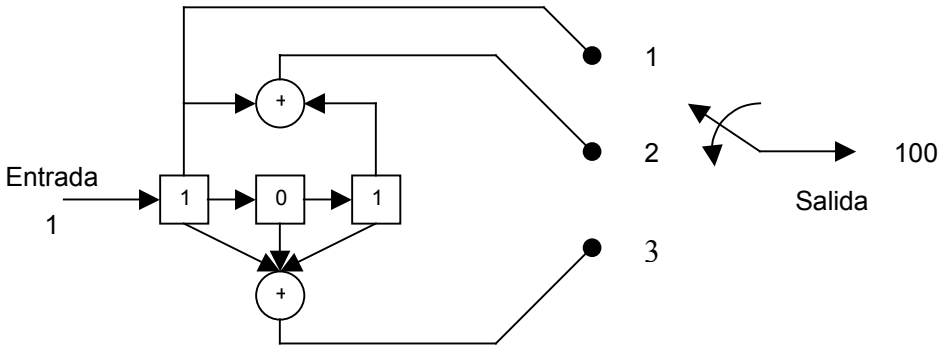


Figura 2.3. Ejemplo paso a paso del codificador *K* = 3, *k* = 1, *n* = 3.

Se puede concluir que se necesitan *n* generadores, cada uno de dimensión *Kk*, para especificar el codificador. En particular, para nuestro ejemplo son necesarios *n* funciones generadoras de dimensión *K*.

La figura 2.4 muestra otro ejemplo ilustrativo cuando *K* = 2, *k* = 2 y *n* = 3. La tasa es ahora $R_c = 2/3$. En cada instante se desplazan dos bits de información dentro del codificador y se generan 3 a la salida. Las funciones generadoras que lo definen son , $\mathbf{g_1}$ = [1 0 1 1], $\mathbf{g_2}$ = [1 1 0 1], $\mathbf{g_3}$ = [1 0 1 0].

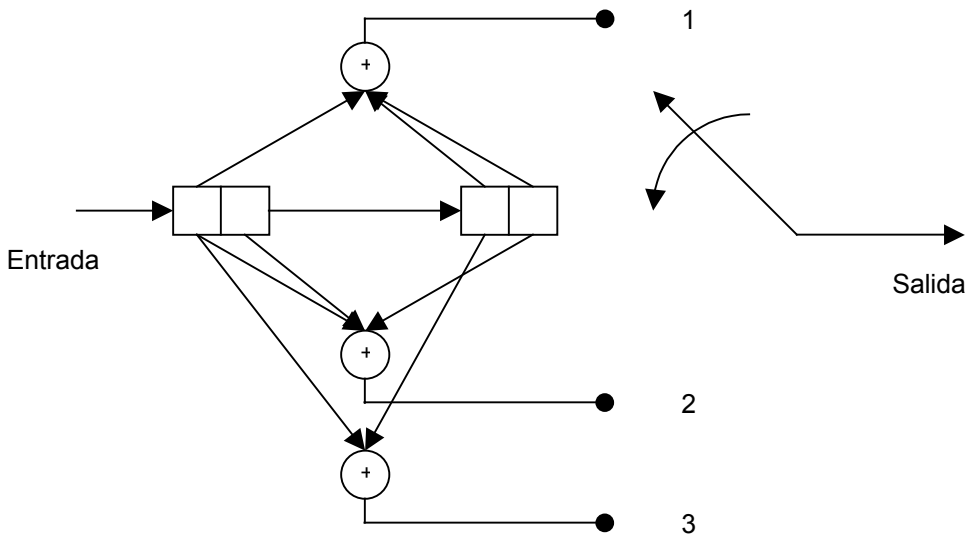


Figura 2.4.Codificador convolucional. $K = 2$, $k = 2$, $n = 3$.

2.2. REPRESENTACIÓN DE LOS CODIFICADORES

Existen tres alternativas para describir y representar los códigos convolucionales: **diagrama de árbol**, **diagrama de Trellis** y **diagrama de estados** [10].

2.2.1. DIAGRAMAS DE ÁRBOL

En la figura 2.5 se muestra el diagrama de árbol correspondiente al codificador de la figura 2.2. La regla a seguir consiste en moverse por la rama superior cuando se recibe un 0 o la inferior cuando a la entrada se tiene un bit a 1. Suponiendo que el codificador se encuentra inicialmente en el estado todo cero, el diagrama muestra que, si el primer bit de entrada es un 0, la secuencia de salida es 000 y, si el primer bit de entrada es un 1, la secuencia de salida es 111. Si los bits de entrada son 1 y 0, siguiendo la rama de abajo y luego, al bifurcarse, la de arriba, la salida será 111 001. Continuando a lo largo del árbol, se puede observar que si el tercer bit de entrada es un 0, entonces la salida es 011, mientras que si es un 1 la salida sería 100. De ese modo, se traza un camino particular a lo largo del árbol determinado por la secuencia de bits de entrada.

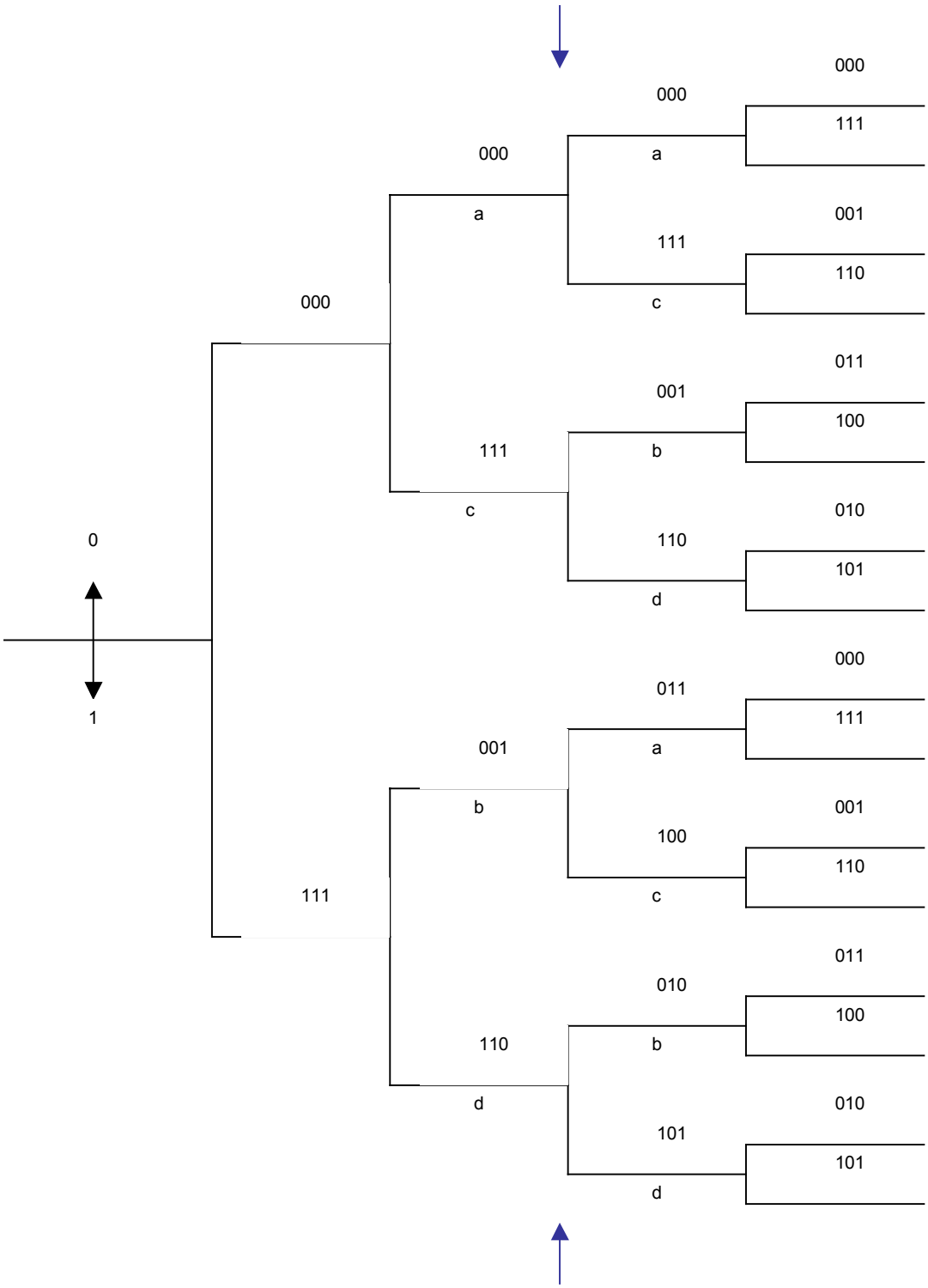


Figura 2.5.Diagrama de árbol. $K = 3$, $k = 1$, $n = 3$.

En la figura 2.6 se muestra el diagrama de árbol del codificador mostrado en la figura 2.4.

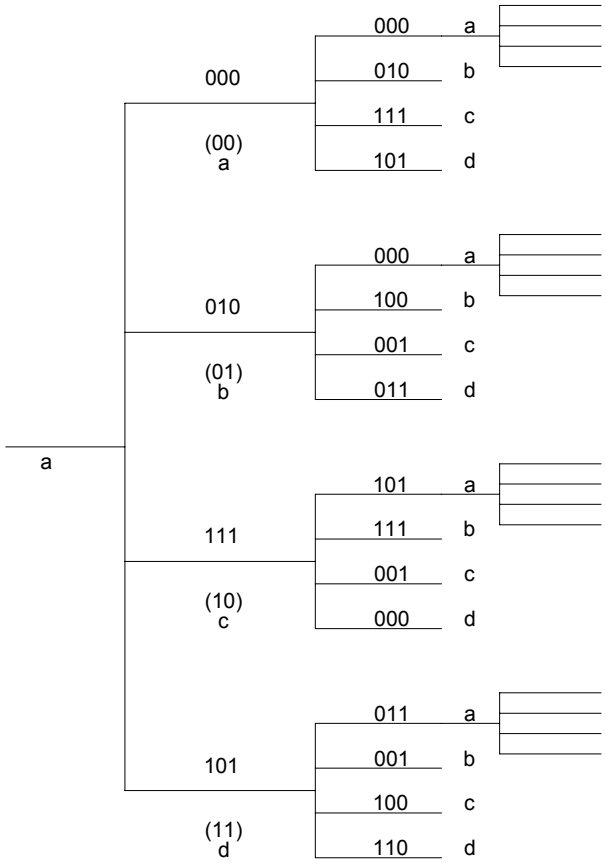


Figura 2.6. Diagrama de árbol. K =2, k = 2, n = 3.

Mirando las ramas de arriba abajo, los dos primeros bits de entrada pueden ser 00, 01, 10 o 11 y las salidas correspondientes 000, 010, 111, 101. Cuando la siguiente pareja de bits de entrada entre en el codificador, el primer par se desplaza a la segunda etapa del registro de desplazamiento. La salida depende del par de bits desplazados y el nuevo par de bits entrante. Así, el diagrama de árbol mostrado en la figura 2.6 contiene cuatro ramas por nodo, correspondientes a los cuatro posibles parejas de símbolos de entrada. Como la longitud reducida es K = 2, el árbol comienza a repetirse a partir de la segunda etapa.

2.2.2. DIAGRAMA DE TRELLIS

Observando la figura 2.5, puede notarse que la estructura se repite a partir del tercer paso (indicado en la figura mediante flechas azules). Es decir, tras un breve transitorio llegamos a un punto en el que dado un bit de entrada , 0 o 1, y dado un estado de partida, *a*, *b*, *c* o *d* siempre se producirá la misma transición y se obtendrá la misma salida independientemente de la etapa en la que nos encontremos. Por ejemplo, situémonos en el estado *a* de la etapa que marca la flecha. Si a la entrada se recibe un 0 a la salida se tendrá 000 y volveremos a estar en el estado *a*. Si por el contrario la entrada es un 1, la salida será 111 y pasaremos al estado *c*. Suponiendo que nos encontramos en el primer caso, llegamos de nuevo al estado *a*, vemos como se vuelve a repetir el diagrama. Si recibimos un 1 avanzamos hasta *a* con una salida 000 y si lo que entra es un 1 llegamos a *c* con salida 111. Todo lo explicado ocurre para cada estado, independientemente de la etapa en la que nos encontremos, una vez pasado el transitorio. Este comportamiento es consistente con el hecho de que la longitud reducida vale 3. Esto significa que la secuencia de salida de tres bits está determinada en cada estado por el bit de entrada en ese instante y los dos bits de entrada previos. Se puede decir que la terna de bits de salida para cada entrada está determinada por el bit de entrada y los cuatro posibles estados previos de los dos primeros bits del registro de desplazamiento, denotados como *a* = 00, *b* = 01, *c* = 10 y *d* = 11. Si se etiqueta cada nodo del árbol para hacerlo corresponder con cada uno de esos cuatro estados posibles, se puede observar que en la tercera etapa hay dos nodos marcados como *a*, dos *b*, dos *c* y dos *d*. También se aprecia que todas las ramas procedentes de nodos etiquetados con el mismo nombre son idénticas, en el sentido de que generan idénticas secuencias de salida. Esto implica que todos los nodos con la misma etiqueta se pueden fusionar en uno solo. Si se realiza este proceso de fusión en la figura 2.5, se obtiene otro diagrama, más compacto, denominado diagrama de Trellis. La figura 2.7 muestra dicho diagrama para el codificador convolucional de la figura 2.2. Por convenio, las líneas continuas denotan las salidas generadas por un bit de entrada a 0 y las discontinuas por un bit de entrada a 1.

De manera similar, para el ejemplo de la figura 2.5, todas las ramas procedentes de nodos etiquetados de igual manera conducen a salidas idénticas. Fusionando los nodos con la misma etiqueta se llega al diagrama de Trellis correspondiente, que se muestra en la figura 2.8.

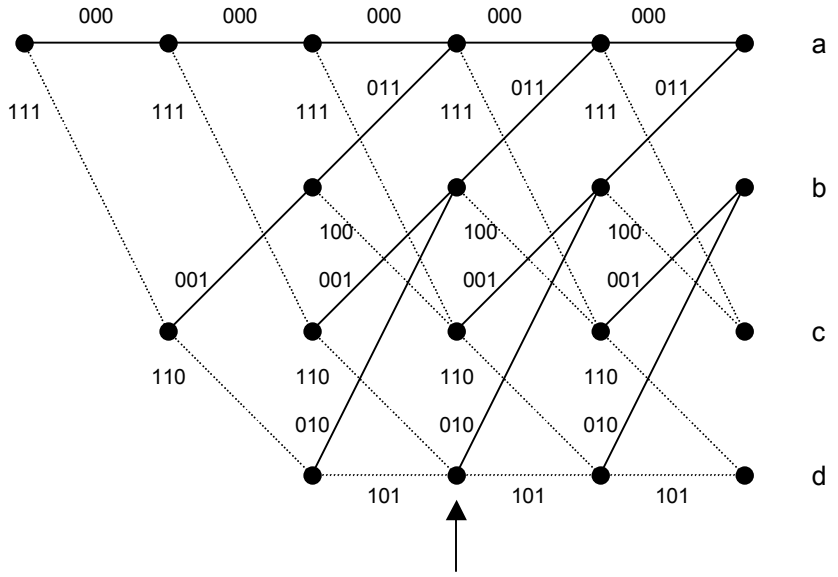


Figura 2.7. Diagrama de Trellis. $K = 3$, $k = 1$, $n = 3$.

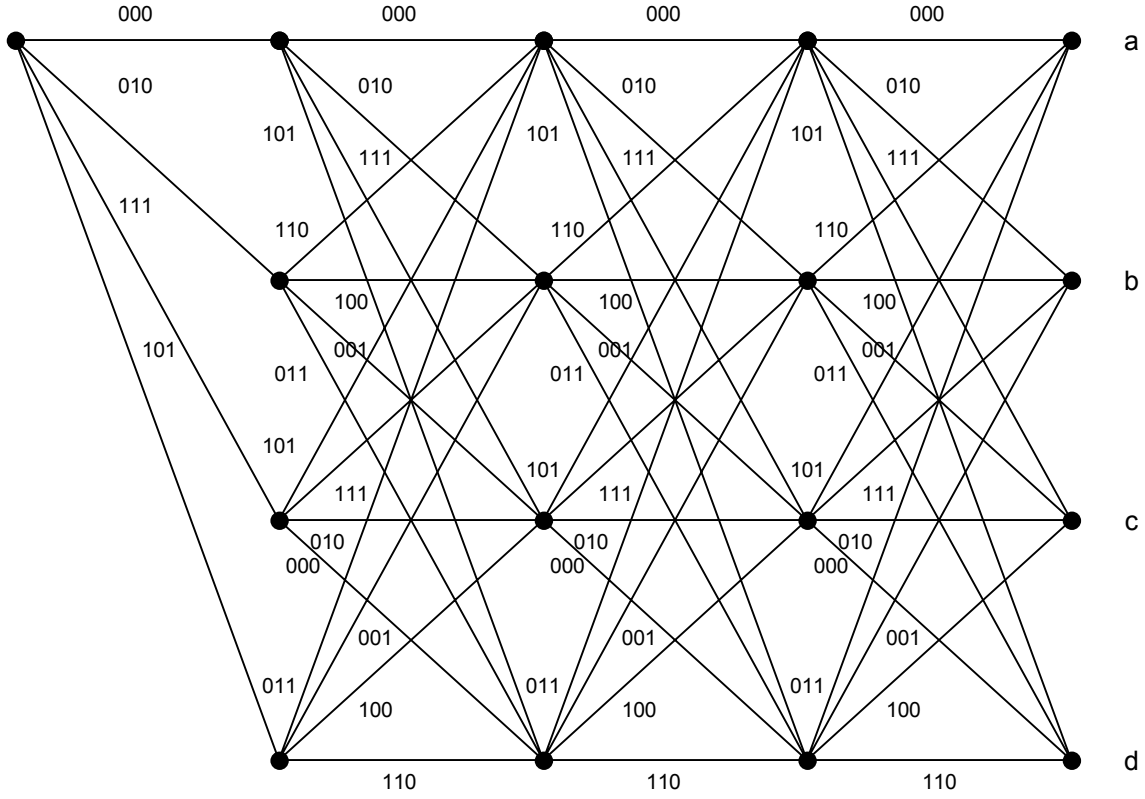


Figura 2.8. Diagrama de Trellis. $K = 2$, $k = 2$, $n = 3$.

En la figura 2.9 se muestra el estado estacionario del diagrama con las entradas correspondientes a cada rama para que resulte más clara su interpretación, ya que, en este caso, se pierde el criterio de líneas continuas y discontinuas.

Para cualquier estado en el que nos encontremos una entrada de valor 00 nos indica que nos movemos por la primera de las ramas que surgen de él, una entrada 01 por la segunda, una 10 por la tercera y una 11 por la cuarta. En todos los casos las primeras ramas llegan al estado *a*, las segundas al *b*, las terceras al *c* y las cuartas al *d*.

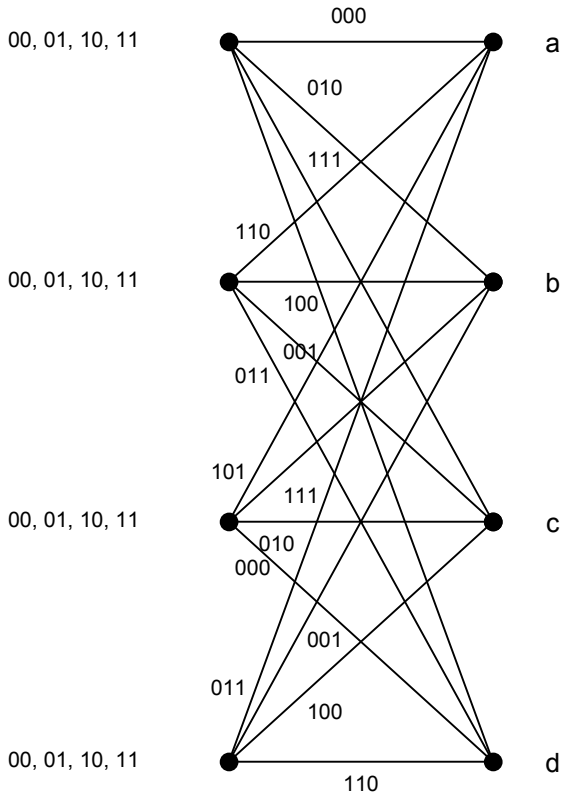


Figura 2.9. Estado estacionario del diagrama de Trellis $K = 2$, $k = 2$, $n = 3$.

2.2.3. DIAGRAMA DE ESTADOS

En el ejemplo del codificador de la figura 2.2, se aprecia que, tras el transitorio inicial, el Trellis de la figura 2.7 contiene cuatro nodos en cada etapa, correspondientes a los cuatro estados posibles del registro de desplazamiento. En ese momento, cada nodo tiene dos caminos de entrada y dos de salida. De estos dos últimos, uno corresponde a la entrada 0 y el otro a la entrada 1.

Usando este hecho y como la secuencia de salida está determinada por la entrada y el estado del codificador, puede obtenerse un diagrama incluso más compacto que el de Trellis, denominado diagrama de estado. Dicho diagrama es un simple gráfico de los estados posibles y las transiciones entre ellos.

La figura1.10 muestra el diagrama de estados del codificador representado en la figura 2.2. Las posibles transiciones son

$$a \overset{0}{\rightarrow} a,\ a \overset{1}{\rightarrow} c,\ b \overset{0}{\rightarrow} a,\ b \overset{1}{\rightarrow} c,\ c \overset{0}{\rightarrow} b,\ c \overset{1}{\rightarrow} d,\ d \overset{0}{\rightarrow} b,\ d \overset{1}{\rightarrow} d,$$

donde $\alpha \overset{1}{\rightarrow} \beta$ denota la transición desde el estado α al β cuando el bit de entrada es un 1. Los tres bits representados junto a cada rama se corresponden con los bits de salida. Las líneas discontinuas indican que la entrada es un 1 y las continuas que es un 0, cumpliendo así con el convenio utilizado en el apartado 1.2.2.

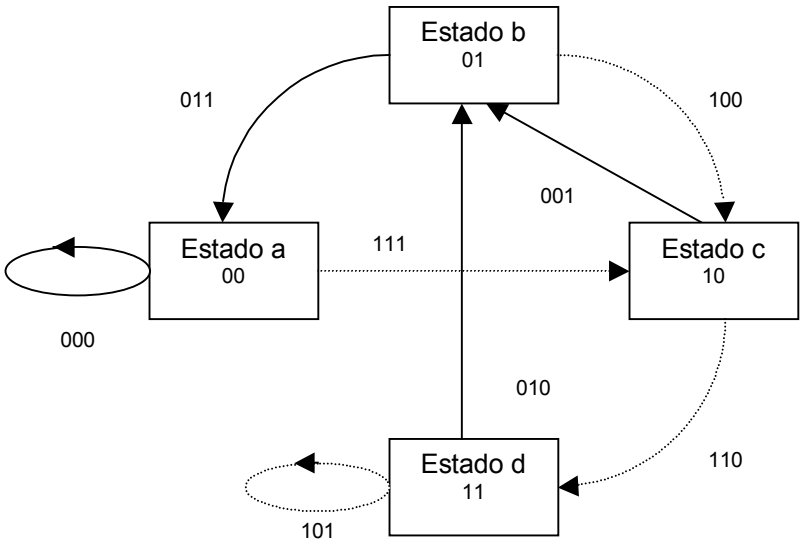


Figura 2.10. Diagrama de estado. *K* = 3, *k* = 1, *n* = 3.

Para el ejemplo descrito en la figura 2.4, se tiene el diagrama de estados, representado en la figura 2.11.

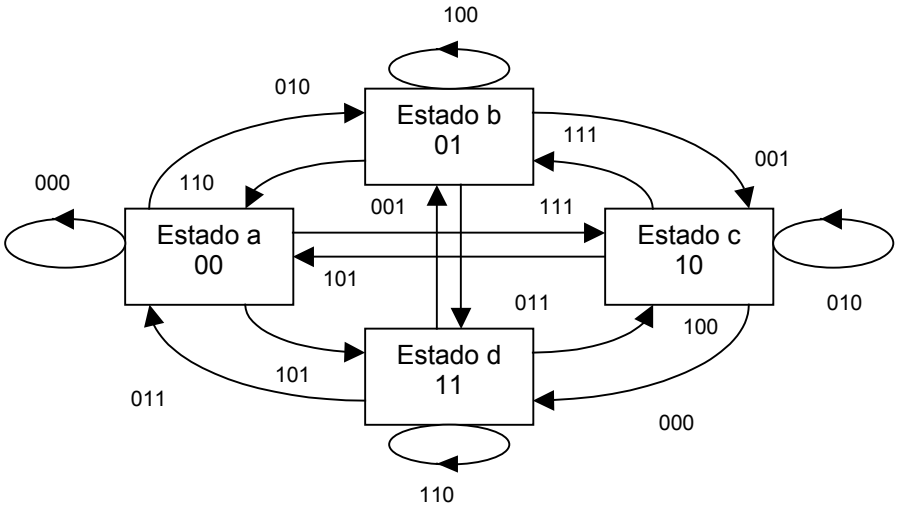


Figura 2.11. Diagrama de estado. K = 2, k = 2, n = 3.

2.2.4. CONCLUSIONES

Generalizando se comprueba que para una tasa dada de $R_c = k/n$ y una longitud reducida K, el código convolucional se caracteriza por 2^k ramas salientes de cada nodo del diagrama de árbol, el cual posee además $2^{k(K-1)}$ estados. En los diagramas de Trellis y de estado, a partir del transitorio inicial, existen 2^k ramas entrantes a cada estado y 2^k salientes.

Los tres tipos de diagramas descritos se utilizan también para representar **códigos convolucionales no binarios**. Cuando el número de símbolos en el alfabeto del código es $q = 2^k$, $k>1$, el código no binario resultante puede ser representado por un código binario equivalente. Por ejemplo, consideramos el código convolucional binario mostrado en la figura 2.12. Este codificador genera un código con los siguientes parámetros: K = 2, k = 2, n = 4, $R_c = 1/2$, a partir de las funciones generadoras $\mathbf{g_1}$=[1 0 1 0], $\mathbf{g_2}$=[0 1 0 1], $\mathbf{g_3}$=[1 1 1 0], $\mathbf{g_4}$=[1 0 0 1].

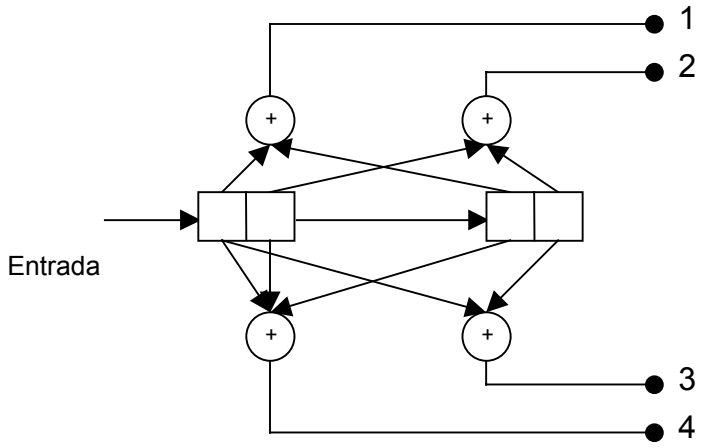


Figura 2.12. Codificador convolucional. $K = 2$, $k = 2$, $n = 4$.

Excepto por la diferencia en el valor de la tasa, este código es similar en su forma al de tasa 2/3 y $k = 2$ considerado en el ejemplo de la figura 2.4.

Sin embargo, también podemos considerar que el código generado por el codificador de la figura 2.12 puede describirse como un código convolucional no binario ($q = 4$) con un símbolo cuaternario como entrada y dos como salida. En cualquier caso, el árbol, el Trellis y el diagrama de estado son independientes de cómo veamos el código. En este caso particular, el código está caracterizado por un árbol del que salen cuatro ramas por cada nodo, por un Trellis con cuatro posibles estados y cuatro ramas entrando y saliendo de cada uno, o por un diagrama de estado con los mismos parámetros que el de Trellis [10].

2.3. FUNCIÓN DE TRANSFERENCIA DE LOS CÓDIGOS CONVOLUCIONALES

Se define la **tasa de error** de un código convolucional como el número de símbolos descodificados erróneamente partido por el número de símbolos transmitidos. Así mismo, se define la **distancia de Hamming** como el número de bits que tienen que cambiarse para transformar una palabra de código no válida en otra palabra de código válida, es decir, el número de bits en los que se diferencian la palabra de código transmitida y la recibida. Las propiedades de distancia y tasa de error pueden obtenerse, para un código convolucional, a partir de su diagrama de estados. Para verlo utilizaremos como ejemplo el diagrama de estados de la figura 2.10 [10].

En primer lugar, se etiquetan las ramas del diagrama como $D^0 = 1$, D^1, D^2, o D^3, donde el exponente de D denota la distancia de Hamming de entre la secuencia de bits de salida de cada

rama y la secuencia de bits de salida correspondiente a la rama todo-cero. El auto-lazo del nodo *a* puede ser eliminado porque no aporta nada a las propiedades de distancia del código. Además, el nodo *a* lo dividiremos en dos nodos, uno de ellos representando la entrada y el otro la salida del diagrama de estados. La figura 2.13 muestra el diagrama resultante.

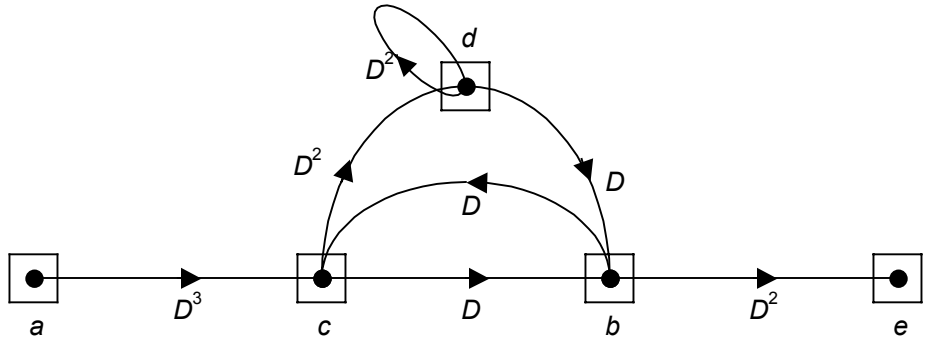


Figura 2.13. Diagrama de estado para el código convolucional de tasa 1/3, *K* = 3.

A partir de este momento se utilizará este diagrama, consistente en cinco estados, para escribir las ecuaciones de estado, las cuales se van a definir como

$$\begin{aligned} X_c &= D^3 X_a + D X_b \\ X_b &= D X_c + D X_d \\ X_d &= D^2 X_c + D^2 X_d \\ X_e &= D^2 X_b \end{aligned} \quad . \tag{2.2}$$

La **Función de Transferencia** para el código se define como

$$T(D) = X_e / X_a \ . \tag{2.3}$$

Resolviendo las ecuaciones de estado anteriores se llega a

$$T(D) = \frac{D^6}{1 - 2D^2} = D^6 + 2D^8 + 4D^{10} + 8D^{12} + ... = \sum_{d=6}^{\infty} a_d D^d \ , \tag{2.4}$$

donde, por definición,

$$a_d = 2^{(d-6)/2} \quad \text{para } d \text{ par}, \tag{2.5}$$

$$a_d = 0 \qquad \text{para } d \text{ impar}. \tag{2.6}$$

La función de transferencia indica que existe un único camino entre *a* y *e*, cuya distancia de Hamming respecto al camino todo-cero sea *d* = 6. A partir del diagrama de la figura 2.10 o del Trellis de la figura 2.7, se observa que dicho camino con *d* = 6 es el *acbe*. El segundo término de la ecuación (2.4) indica que existen dos caminos desde el nodo *a* al nodo *e* con distancia *d* = 8, *acdbe*

y *acbcbe*. El tercer término indica la existencia de 4 caminos con distancia *d* = 10 y así sucesivamente. De este modo, la función de transferencia proporciona las propiedades de distancia del código convolucional. La distancia mínima del código se denomina **distancia libre mínima**, d_{free}. En el caso que se está estudiando, $d_{free} = 6$.

La función de transferencia puede proporcionar una información más detallada que la de la distancia de varios caminos. Para ello vamos a introducir un factor *N* en todas las ramas en las que se produzca una transición debido a un bit de entrada a 1. Además, incluiremos un factor *J* en cada rama, y así su exponente indicará el número de ramas, para un camino dado, entre los nodos *a* y *e*. Para el ejemplo analizado el nuevo diagrama con los factores *J* y *N* se muestra en la figura 2.14.

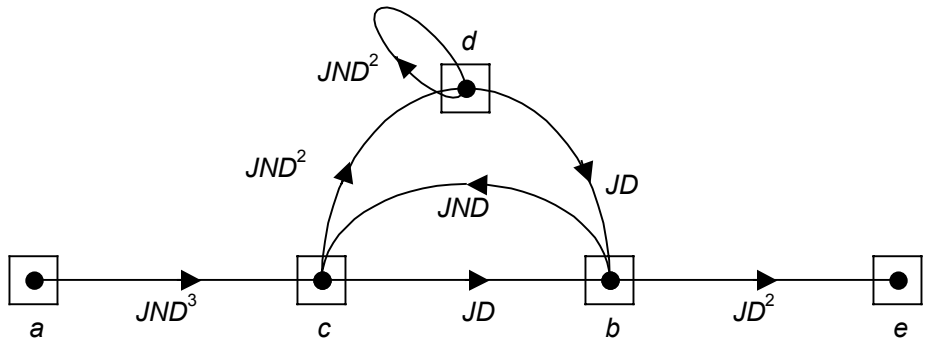


Figura 2.14. Diagrama de estado para el código de tasa 1/3, *K* = 3, con factores *J* y *N*.

Las ecuaciones de estado se formulan ahora como

$$\begin{aligned} X_c &= JND^3 X_a + JNDX_b \\ X_b &= JDX_c + JDX_d \\ X_d &= JND^2 X_c + JND^2 X_d \\ X_e &= JD^2 X_b \end{aligned} \quad . \tag{2.7}$$

La función de transferencia, definida como el cociente X_e / X_a , se obtiene resolviendo las ecuaciones de estado,

$$T(D,J,N) = \frac{J^3 ND^6}{1 - JND^2(1+J)} = \tag{2.8}$$
$$J^3 ND^6 + J^4 N^2 D^8 + J^5 N^2 D^8 + J^5 N^3 D^{10} + 2J^6 N^3 D^{10} + J^7 N^3 D^{10} + \ldots$$

Esta formulación de la función de transferencia proporciona las propiedades de todos los caminos del código. El primer término indica que el camino de distancia *d* = 6 tiene una longitud de tres ramas y que, de los tres bits de información, uno es un 1. El segundo y tercer término de la expansión de *T(D,J,N)* indican la existencia de dos caminos con distancia *d* = 8. El primero de ellos

tiene longitud 4 y el segundo 5. Dos de los cuatro bits de información del primer camino valen 1 y dos de los cinco del segundo también son 1.

En resumen, el exponente de *J* indica la longitud del camino entre los estados *a* y *e*, el exponente de *N* indica el número de bits de información a 1 para ese camino, y el exponente del factor *D* proporciona la distancia del camino al todo-cero.

El factor *J* es especialmente importante cuando se transmite una secuencia de duración finita, *m* bits. En ese caso, se trunca el código convolucional después de *m* nodos o *m* ramas. Esto implica que la función de transferencia del código truncado se consigue truncando el desarrollo de *T(D,J,N)* por el término J^m. Por el contrario, si se transmite una secuencia extremadamente larga, es decir, una secuencia que puede considerarse infinita, podría eliminarse la dependencia de la función de transferencia con *J.* Esto se consigue fácilmente haciendo *J* = 1. Para el ejemplo desarrollado la función de transferencia queda como

$$T(D,N,1) = T(D,N) = \frac{ND^6}{1-2ND^2} = ND^6 + 2N^2D^8 + 4N^3D^{10} + ... = \sum_{d=6}^{\infty} a_d N^{(d-4)/2} D^d \quad , \quad (2.9)$$

donde los coeficientes $\{a_d\}$ se definieron en (2.5) y (2.6).

El procedimiento detallado para determinar la función de transferencia de un código convolucional binario se extiende con facilidad a uno no binario. En el siguiente ejemplo se determinará la función de transferencia para el ejemplo de código convolucional no binario mostrado en la figura 2.12 [10].

En este caso se tienen distintas opciones para etiquetar los nodos y contar los errores en función de si decidimos tratar el código como binario o no binario. En nuestro caso vamos a considerar el código como no binario, de modo que los símbolos a la entrada y la salida del descodificador son cuaternarios. En particular, siendo 00, 01, 10 y 11 símbolos cuaternarios, la distancia entre las secuencias 0111 y 0000 medida en símbolos es 2. Lo que es más, si se transmite el símbolo 00 y se descodifica el 11, se considera que se ha cometido un sólo error de símbolo. Con estas consideraciones el diagrama de estados para el codificador queda como muestra la figura 2.15.

Las ecuaciones de estado obtenidas son

$$\begin{aligned} X_b &= NJD^2X_a + NJDX_b + NJDX_c + NJD^2X_d \\ X_c &= NJD^2X_a + NJD^2X_b + NJDX_c + NJDX_d \\ X_d &= NJD^2X_a + NJDX_b + NJD^2X_c + NJDX_d \\ X_e &= JD^2(X_b + X_c + X_d) \end{aligned} \quad . \quad (2.10)$$

La solución de estas ecuaciones conduce a la función de transferencia siguiente

$$T(D,J,N)=\frac{3NJ^2D^4}{1-2NJD-NJD^2}. \tag{2.11}$$

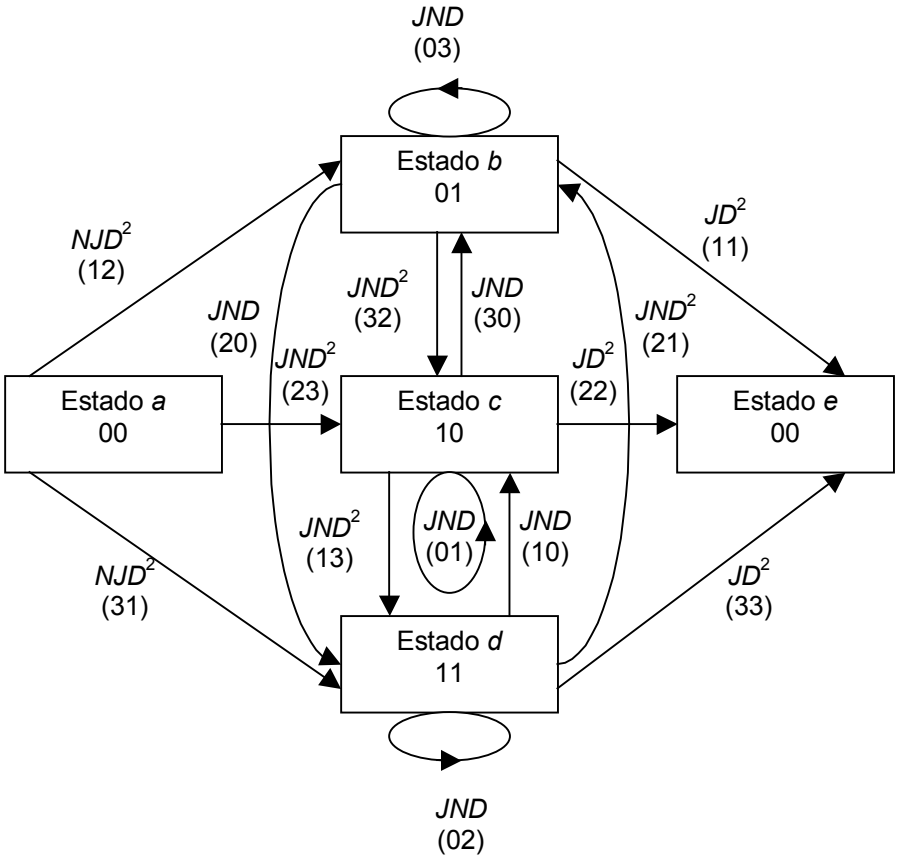


Figura 2.15. Diagrama de estado para el código convolucional no binario con tasa 1/2, *K* = 2, *k* = 2.

2.4. DESCODIFICACIÓN ÓPTIMA . ALGORITMO DE VITERBI

2.4.1. DETECTOR DE MÁXIMA VEROSIMILITUD

Se supone que, en cada ranura de tiempo de duración *T* segundos, se transmite una de las *M* posibles señales equiprobables $s_1(t), s_2(t), ..., s_M(t)$. Para un canal AWGN, una posible realización o muestra, $x(t)$, del proceso aleatorio $\mathbf{X}(t)$ recibido puede describirse como

$$x(t) = s_i(t) + w(t) \quad \begin{matrix} 0 \le t \le T \\ i = 1,2,...,M \end{matrix} , \tag{2.12}$$

donde $w(t)$ es una muestra del proceso gaussiano blanco $\mathbf{W}(t)$, de media cero y densidad espectral de potencia *No*/2. El receptor recibe la señal $x(t)$ y hace la mejor estimación posible de la señal transmitida $s_i(t)$ o, lo que es equivalente, del símbolo m_i correspondiente. Podemos representar $s_i(t)$ como el sumatorio de los **coeficientes de expansión** s_{ij} y las **funciones de una base ortonormal** $\varphi(t)$

$$\mathrm{s}_i(t) = \sum_{j=1}^{N} s_{ij}\varphi_j(t) \quad \begin{matrix} 0 \le t \le T \\ i = 1,2,...,M \end{matrix} . \tag{2.13}$$

Se define $\mathbf{x}$ como **vector recibido o vector observación**, $\mathbf{s_i}$ como **vector señal** o **vector transmitido** y $\mathbf{w}$ como **vector de ruido**. El vector $\mathbf{x}$ difiere de $\mathbf{s_i}$ en $\mathbf{w}$, cuya orientación es aleatoria, es decir,

$$\mathbf{x} = \mathbf{s_i} + \mathbf{w} . \tag{2.14}$$

Tomando en consideración todas las definiciones se puede dibujar el esquema de vectores y puntos en el espacio euclídeo de la figura 2.16.

En este punto se tiene todo lo necesario para abordar el problema de la detección. Hay que obtener una estimación $\hat{m}$ del símbolo transmitido m_i, a partir del vector de observación $\mathbf{x}$, de modo que se minimice la **probabilidad media de error de símbolo** en la decisión. El **detector máximo-verosímil** es el que proporciona dicha estimación.

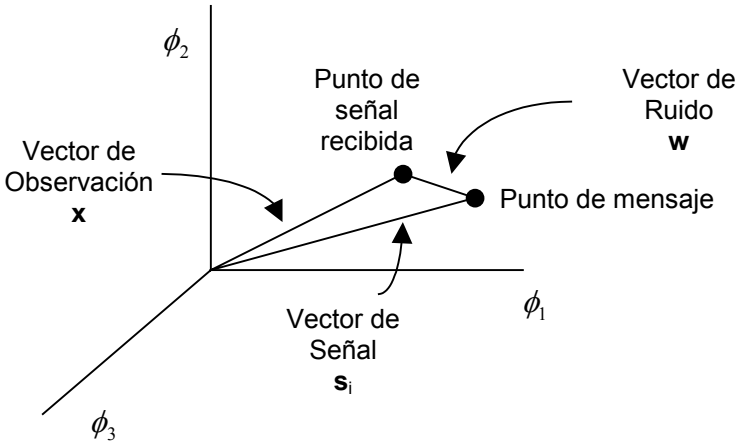


Figura 2.16. Espacio euclídeo.

Supóngase que, recibido el vector de observación $\mathbf{x}$, se decide que el símbolo estimado es $\hat{m} = m_i$. La probabilidad media de error de símbolo en esta decisión, denotada como $P_e(m_i, x)$, es simplemente

$$P_e(m_i, \mathbf{x}) = P(m_i \text{ no enviado} \mid \mathbf{x}) = 1 - P(m_i \text{ enviado} \mid \mathbf{x}). \tag{2.15}$$

Por tanto, la regla de decisión es como sigue:

$$\text{fijar } \hat{m} = m_i \text{ si } \quad P(m_i \text{ enviado} \mid \mathbf{x}) \geq P(m_k \text{ enviado} \mid \mathbf{x}), \forall k \neq i. \tag{2.16}$$

Esta regla de decisión se conoce como **máxima probabilidad a priori**. Aplicando el teorema de Bayes, se puede rescribir como

$$\text{fijar } \hat{m} = m_i \text{ si } \frac{p_k f_{\mathbf{x}}(\mathbf{x} \mid m_k)}{f_{\mathbf{x}}(\mathbf{x})} \text{ es máxima para } k = i, \tag{2.17}$$

donde p_k es la probabilidad a priori de ocurrencia del símbolo m_k, $f_{\mathbf{x}}(\mathbf{x} \mid m_k)$ es la función de probabilidad que resulta cuando se ha enviado el símbolo m_k y $f_{\mathbf{x}}(\mathbf{x})$ es la función de densidad de probabilidad conjunta del vector $\mathbf{X}$. Sabiendo que el denominador de la expresión es independiente del símbolo transmitido y que éstos son equiprobables, la regla de decisión puede simplificarse de modo que

$$\text{fijar } \hat{m} = m_i \text{ si } f_{\mathbf{x}}(\mathbf{x} \mid m_k) \text{ es máxima probabilidad a posteriori para } k = i. \tag{2.18}$$

Esta regla de decisión se conoce como **máxima probabilidad a posteriori**.

Resulta más conveniente trabajar con el logaritmo que con la función de verosimilitud. De este modo, la regla de decisión puede expresarse como

$$\text{fijar } \hat{m} = m_i \text{ si } \ln\left[f_{\mathbf{x}}(\mathbf{x} \mid m_k)\right] \text{ es máximo para } k = i. \tag{2.19}$$

Como ya se ha comentado, la regla se conoce como decisión de máxima-verosimilitud y el decisor que la implementa calcula las métricas para cada mensaje transmitido, las compara y decide a favor del que tenga métrica máxima.

Resulta de utilidad tener una interpretación gráfica de la regla de decisión de máxima-verosimilitud. Sea Z el espacio N-dimensional de todos los posibles vectores de observación $\mathbf{x}$. Nos referiremos a este espacio como **espacio de observación**, estando este dividido en M regiones de decisión $Z_1, Z_2, \ldots, Z_M$, tantas como m_i existen. De acuerdo con esto, se modifica la regla de decisión como sigue:

$$\text{el vector } \mathbf{x} \text{ pertenece a la región } Z_i \text{ si } \ln\left[f_{\mathbf{x}}(\mathbf{x} \mid m_k)\right] \text{ es máximo para } k = i. \tag{2.20}$$

Aparte de los límites entre las distintas regiones, es evidente que el conjunto de regiones cubre por completo el espacio de posibles vectores de observación $\mathbf{x}$. Todos los valores que caigan en una frontera se resuelven al azar. El resultado de estas acciones no condiciona el valor de la probabilidad de error, ya que en la frontera de dos regiones se cumple la regla con el signo de igualdad.

Para ilustrar lo explicado, considérese un canal AWGN, para el cual se puede demostrar que la función de verosimilitud es:

$$f_{\mathbf{x}}(\mathbf{x} \mid m_k) = (\pi N_o)^{-N/2} \exp\left[-\frac{1}{N_o} \sum_{j=1}^{N} (x_j - s_{kj})^2 \right] \quad k = 1, 2, ..., M. \tag{2.21}$$

El correspondiente valor de la métrica es

$$\ln\left[f_{\mathbf{x}}(\mathbf{x} \mid m_k)\right] = -\frac{N}{2}(\pi N_o) - \frac{1}{N_o} \sum_{j=1}^{N} (x_j - s_{kj})^2 \quad k = 1, 2, ..., M. \tag{2.22}$$

Ignorando el término constante y sustituyendo en la regla, se puede reformular la regla de decisión máximo-verosímil para canales AWGN como:

el vector $\mathbf{x}$ pertenece a la región Z_i si $-\frac{1}{N_o} \sum_{j=1}^{N} (x_j - s_{kj})^2$ es máximo para $k = i$. (2.23)

De forma equivalente se puede establecer que

el vector $\mathbf{x}$ pertenece a la región Z_i si $\sum_{j=1}^{N} (x_j - s_{kj})^2 = \left\| \mathbf{x} - \mathbf{s}_k \right\|^2$ es mínimo para $k = i$ (2.24)

siendo $\left\| \mathbf{x} - \mathbf{s}_k \right\|$ la distancia entre el punto de señal recibida y el mensaje, representados por los vectores $\mathbf{x}$ y s_k respectivamente, es decir, la **distancia euclídea**. De acuerdo con todo esto se puede rescribir la regla de este modo:

el vector $\mathbf{x}$ pertenece ala región Z_i si $\left\| \mathbf{x} - \mathbf{s}_k \right\|$ es mínimo para $k = i$, (2.25)

lo que significa que la regla de decisión consiste simplemente en elegir el punto de mensaje más cercano al punto de señal recibida [9].

2.4.2. DESCODIFICACIÓN ÓPTIMA

En la descodificación de un código de bloques para un canal sin memoria, se calculan las distancias (distancia de Hamming para **decisión dura** y distancia euclídea para **decisión blanda**) entre la palabra de código recibida y las 2^k posibles palabras de código transmitidas. El concepto de distancia de Hamming se definió en el apartado 2.3. y el de distancia euclídea en el 2.4.1.

Finalmente se selecciona la palabra de código más cercana a la palabra recibida. Esta regla de decisión, que requiere el cálculo de 2^k **métricas**, es óptima en el sentido de que minimiza la probabilidad de error para canales binarios simétricos con $p < 1/2$ y ruido aditivo gaussiano en el canal.

A diferencia de los códigos de bloques, que tienen una longitud fija *n*, un codificador convolucional es básicamente una máquina con un número finito de estados. De ese modo, el descodificador óptimo es el estimador de máxima verosimilitud (**MLSE**). La descodificación óptima de un código convolucional requiere recorrer el diagrama de Trellis en busca de la secuencia más probable. Dependiendo de si el detector que sigue al desmodulador implementa una decisión dura o blanda, la métrica utilizada en la búsqueda a lo largo del árbol será la distancia de Hamming o la euclídea, respectivamente.

Para que el análisis sea más claro se trabajará a modo de ejemplo sobre el diagrama de Trellis mostrado en la figura 2.7, correspondiente al codificador de la figura 2.2. Supongamos que introducimos 3 bits a la entrada del codificador, con lo cual serán 9 los bits transmitidos. Estos se van a denotar como $\{c_{jm}$, j = 1,2,3; m =1,2,3}, donde el índice j indica la j-ésima rama y m el m-ésimo bit en esa rama. Así mismo, se definen las salidas del desmodulador como $\{r_{jm}$, j = 1,2,3; m =1,2,3}. Cuando el detector implementa la decisión dura, sus salidas para cada bit transmitido son o bien 1 o bien 0. En el caso de que utilice decisión blanda y la secuencia transmitida sea BPSK, la entrada al descodificador resulta ser

$$r_{jm} = \sqrt{\varepsilon_c}\,(2c_{jm} - 1) + n_{jm} \quad , \tag{2.26}$$

donde c_{jm} puede ser 1 o 0, η_{jm} representa el ruido aditivo y ε_c la energía transmitida por cada bit del código [10].

2.4.2.1. DECISIÓN DURA

La métrica se define, para la rama j-ésima del camino i-ésimo a lo largo del diagrama de Trellis, como el logaritmo de la probabilidad conjunta de la secuencia $\{r_{jm}$, m = 1,2,3} condicionada a la secuencia transmitida $\{c_{jm}^{(i)}$, j,m= 1,2,3} para el camino i-ésimo. Esto es

$$\mu_j^{(i)} = \log P(R_j \mid C_j^{(i)}) \text{ , } j = 1,2,3,... \tag{2.27}$$

La métrica para el camino completo se calcula a partir de las B ramas a lo largo del diagrama y se define como

$$PM^{(i)} = \sum_{j=1}^{B} \mu_j^{(i)} \tag{2.28}$$

El criterio para decidir entre dos caminos es seleccionar aquel cuya métrica sea mayor. Esta regla maximiza la probabilidad de decidir correctamente o, equivalentemente, minimiza la probabilidad de error para la secuencia de bits de información.

Por ejemplo, se supone la secuencia recibida {101 000 100}. Se consideran los dos caminos en el Trellis que comienzan en el estado *a* y regresan a él después de tres transiciones, tres ramas, que se corresponden con las secuencias de información 000 y 100 y las secuencias transmitidas (caminos) 000 000 000 y 111 001 011, respectivamente. Sea *i* = 0 el índice del primer camino e *i* = 1 el del segundo.

Camino (0):	000000000
Palabra recibida:	101000100

Camino (1):	111001011
Palabra recibida:	101000100

Los bits coloreados en rojo son aquellos que se diferencian del camino, es decir, los bits erróneos. Por el contrario, los bits coloreados en azul son bits correctos. Comparando la palabra recibida con el camino (0) observamos que hay 3 bits incorrectos y 6 correctos mientras que, comparando con el camino (1), solo hay 4 bits válidos y el resto son erróneos. Se define *p* como la probabilidad de que transmitido un bit se reciba el mismo un bit distinto, es decir probabilidad de bit erróneo y (1-*p*) la probabilidad complementaria como probabilidad de bit correcto. Sabiendo que la métrica buscada es el sumatorio, para todos los bits de la palabra de código, del logaritmo de la función de probabilidad, ecuación (2.28), se llega a que las métricas para los dos caminos son,

$$PM^{(0)} = 6\log(1-p) + 3\log p\,, \tag{2.29}$$

$$PM^{(1)} = 4\log(1-p) + 5\log p\,. \tag{2.30}$$

donde *p* es la probabilidad de error de bit. Asumiendo que *p* <1/2, se observa que $PM^{(0)}$ es mayor que $PM^{(1)}$. Este resultado es consistente con el hecho de que la distancia de Hamming para el camino *i* = 0 es *d* = 3, mientras que para el otro camino, *i* = 1, la distancia a la secuencia recibida es *d* = 5. De este modo, queda comprobado que la distancia de Hamming es equivalente a la métrica propuesta para la descodificación con decisión dura. Así pues, **el criterio consiste en minimizar la distancia de Hamming** [10].

2.4.2.2. DECISIÓN BLANDA

En descodificación blanda se actúa de forma similar. Se supone ruido aditivo gaussiano. La

salida del desmodulador se describe estadísticamente a partir de la función de densidad de probabilidad

$$p(r_{jm} \mid c_{jm}^{(i)}) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{ - \frac{\left[r_{jm} - \sqrt{\varepsilon_c (2c_{jm}^{(i)} - 1)} \right]^2}{2\sigma^2} \right\}, \tag{2.31}$$

donde σ^2 =1/2 N_o es la varianza del ruido gaussiano aditivo.

Para el cálculo de la métrica debemos tener en cuenta que cuando $r_{jm} = c_{jm}^{(i)}$ la métrica debe aumentar en 1, mientras que si $r_{jm} \neq c_{jm}^{(i)}$ la métrica permanece como estaba. Siguiendo este criterio podemos llegar a la siguiente expresión:

$$\mu_j^{(i)} = \sum_{m=1}^{n} r_{jm} \cdot c_{jm}^{(i)} + (1 - r_{jm}) \cdot (1 - c_{jm}^{(i)}) . \tag{2.32}$$

El primer término del sumatorio contabiliza el caso en el que $r_{jm} = c_{jm}^{(i)} = 1$, mientras que el segundo se encarga de sumar los casos en los que $r_{jm} = c_{jm}^{(i)} = 0$. Desarrollando la ecuación llegamos a

$$\mu_j^{(i)} = \sum_{m=1}^{n} r_{jm} \cdot (2c_{jm}^{(i)} - 1) + (1 - c_{jm}^{(i)}) . \tag{2.33}$$

Obviando los términos que hay en común para las métricas de todas las ramas, es decir, el segundo sumando, la métrica para la rama *j*-ésima del camino *i*-ésimo puede expresarse como

$$\mu_j^{(i)} = \sum_{m=1}^{n} r_{jm} (2c_{jm}^{(i)} - 1) , \tag{2.34}$$

donde, en nuestro ejemplo, *n* = 3. Las métricas totales de los dos caminos bajo consideración son

$$CM^{(0)} = \sum_{j=1}^{3} \sum_{m=1}^{3} r_{jm} (2c_{jm}^{(0)} - 1) \tag{2.35}$$

$$CM^{(1)} = \sum_{j=1}^{3} \sum_{m=1}^{3} r_{jm} (2c_{jm}^{(1)} - 1) \tag{2.36}$$

El criterio para decidir entre dos caminos es seleccionar aquel cuya métrica sea mayor. Ocurre lo mismo que en el caso de decisión dura, la diferencia se encuentra en la definición de la métrica [10].

2.4.3. ALGORITMO DE VITERBI

El **Algoritmo de Viterbi** busca el camino óptimo en el diagrama de Trellis de la forma más eficiente posible. La idea fundamental en la que se basa el algoritmo es el llamado **Principio de optimalidad**, publicado por Viterbi en 1967 [11]. Años más tarde, J. Omura propuso un esquema dinámico para la búsqueda de la distancia mínima de descodificación, demostrando que el algoritmo de Viterbi era de máxima verosimilitud [11]. El concepto de Trellis llegó de forma separada. Unido al diagrama de Trellis, el algoritmo ha conseguido ser una herramienta ampliamente utilizada hoy en día, a pesar de realizar una búsqueda exhaustiva. Ello es debido a que los pasos a seguir se repiten en cada nodo y cada etapa del diagrama de Trellis, reduciendo así la carga computacional. El algoritmo lleva a cabo una descodificación máximo verosímil e implica el cálculo, en cada instante, de una medida de similitud o distancia entre la palabra de código y los caminos posibles. La ventaja de éste algoritmo sobre otros descodificadores empleados es que, la complejidad de la descodificación de Viterbi no depende del número de símbolos de la secuencia codificada de información.

La equivalencia entre descodificación máximo-verosímil y descodificación de distancia mínima implica que se puede descodificar un código convolucional eligiendo un camino en el árbol de código cuya secuencia codificada difiera de la secuencia recibida en el menor número de posiciones. Pero además, ya que, un árbol de código es equivalente a un diagrama Trellis, se puede igualmente limitar la elección a un camino posible en la representación de Trellis. La razón por la que se prefiere al Trellis en lugar del árbol es que el número de nodos a cualquier nivel del Trellis no continua creciendo mientras lo hace el número de bits de entrada sino que permanece constante a 2^{K-1} [11].

Habiendo definido las métricas de las ramas y las de los caminos, calculadas por el descodificador, se explicará ahora el uso del Algoritmo de Viterbi para la descodificación óptima de la secuencia de información codificada. Se trabajará a partir de los dos caminos definidos como ejemplo en el apartado 2.4.2.1, que surgían del estado *a* y regresaban a él tras tres transiciones.

Si $CM^{(0)} > CM^{(1)}$ en el nodo *a*, donde confluyen los dos caminos tras las tres transiciones, $CM^{(0)}$ seguirá siendo mayor que $CM^{(1)}$ para cualquier camino que salga de dicho nodo. Esto significa que el camino $i = 1$ puede ser descartado para consideraciones futuras, quedando el camino $i = 0$ como **superviviente** con su correspondiente métrica $CM^{(0)}$. De forma análoga, uno de los dos caminos que llegan a *b* podría ser descartado basándose en las dos métricas correspondientes. El proceso se repite para los nodos *c* y *d*. Como resultado, después de las tres primeras transiciones, se tienen cuatro caminos supervivientes, cada uno de los cuales termina en uno de los estados *a*, *b*, *c*, *d*, y cuatro métricas correspondientes. Este proceso se repite en cada

etapa del diagrama de Trellis según van llegando las nuevas señales en los siguientes intervalos de tiempo [10].

Una dificultad que puede surgir al aplicar el algoritmo es la posibilidad de que aparezcan dos caminos entrantes en un estado con igual métrica. En una situación así, se elige entre ellos al azar. En la implementación que se ha realizado en este proyecto, se opta por el primero de los caminos analizados [11].

En resumen, el algoritmo de Viterbi es un descodificador máximo verosímil que es óptimo para canales con ruido blanco gaussiano y aditivo. A continuación se muestra el algoritmo paso a paso :

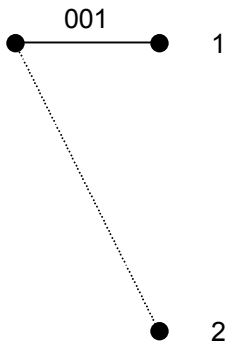
Inicio (*j* = 0). Etiquetar los estados más a la izquierda del Trellis como 0, es decir, nos encontramos en el estado inicial todo a cero.
Para *j* = 1,2,3,... y para todos los estados del diagrama.
En el nivel *j* .

- Calcular $\mu_{(j)k}^{(i)} = \sum_{m=1}^{n} r_{(j)m}(2c_{jm}^{(i)} - 1)$ para la rama *j*+1 de todo camino *i* entrante al estado *k*.
- Acumular las métricas para cada camino *i*, $CM_k^{(i)} = CM_{k\ min}^{(i)} + \sum_{m=1}^{n} r_{(j-1)m}(2c_{jm}^{(i)} - 1)$.
- Identificar los caminos supervivientes; para cada estado *k* se elige el camino que llega con menor métrica. Se busca el $min(CM_k^{(i)})$
- Almacenar el camino superviviente y su métrica correspondiente para cada estado del diagrama de Trellis $CM_{k\ min}^{(i)}$.

Paso final. Se continua iterando hasta que se complete la búsqueda a lo largo del diagrama de Trellis y por lo tanto, se llegue al nodo final, es decir, al estado cero. En ese momento, se toma la decisión de cual es el camino máximo-verosímil. La secuencia de símbolos asociada a ese camino se saca del descodificador como secuencia de salida descodificada.

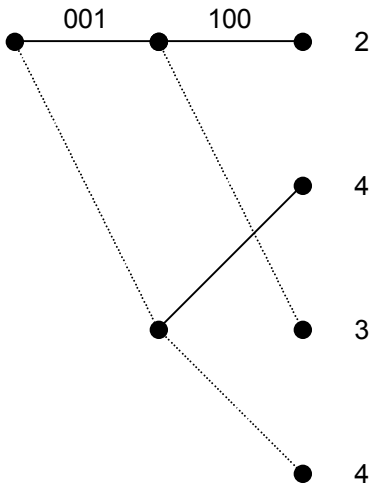
Veamos un ejemplo gráfico para ilustrar el algoritmo. Consideramos El diagrama de Trellis mostrado en la figura 2.7. Supongamos que se envía la secuencia {000 000 000 000 000 000} y se recibe {001 100 110 000 010 000}. En cada paso se calculan las distancias de Hamming entre la señal recibida y los bits de la transición del diagrama para cada rama.

PASO 1



Se recibe 001 y hay que calcular las distancias de Hamming para la rama superior, (000) y para la inferior, (111). Denotemos como d_1 a la primera distancia y d_2 a la segunda. En general, d_i es el número de bits en que difieren las secuencias comparadas, de modo que se tiene: $d_1 = 1$ y $d_2 = 2$.

PASO 2

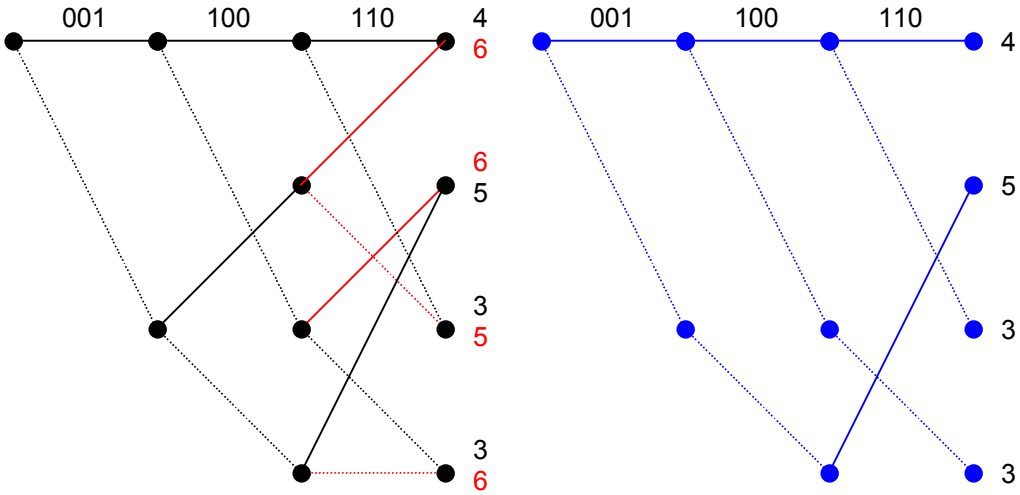


Recibimos 100 y actuando de modo similar al paso anterior calculamos las métricas. Con este paso terminamos la fase transitoria, ya tenemos un camino por cada estado del diagrama. Las métricas se van acumulando según vamos avanzando por las ramas. Cada nodo se etiqueta con el valor de la métrica necesaria para llegar a él desde el estado inicial.

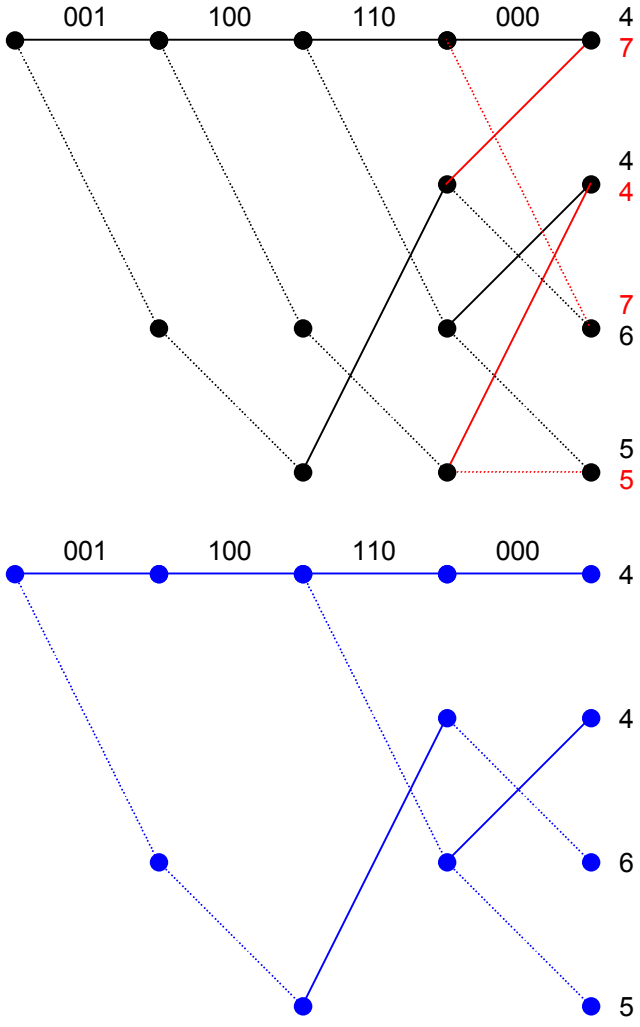
PASO 3

En este paso, no sólo se calculan las distancias y se actualizan las métricas, también se decide cuáles son los caminos supervivientes. Recordemos que nuestro objetivo es elegir el camino cuya distancia a la señal recibida sea menor, es decir, buscamos distancia la de Hamming mínima. En cada paso nos quedamos con uno de los dos caminos entrantes a cada estado y almacenamos el valor de su métrica. En este caso, cada nodo se etiqueta con dos números, uno correspondiente al camino que entra por la rama superior y el otro correspondiente al que lo hace desde la inferior. La métrica coloreada de rojo, la mayor de las dos, corresponde al camino que se descarta porque incumple el criterio de minimizar la distancia. Así mismo, también se colorean de rojo las ramas que forman parte de ese camino. Los caminos supervivientes, uno por cada estado, son de métrica mínima en cada paso y se muestran coloreados de azul.

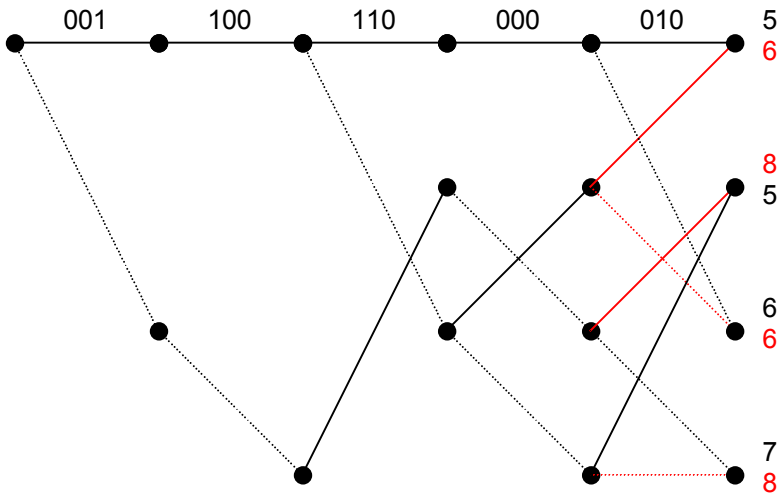
El proceso seguido en el paso 3 se repite mientras dure la secuencia de entrada.

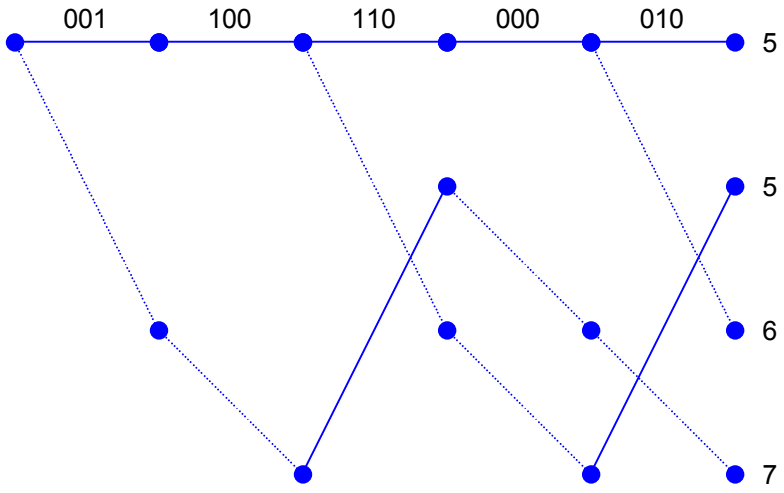

001
100
110
4
6
6
5
3
5
3
6
001
100
110
4
5
3
3

PASO 4

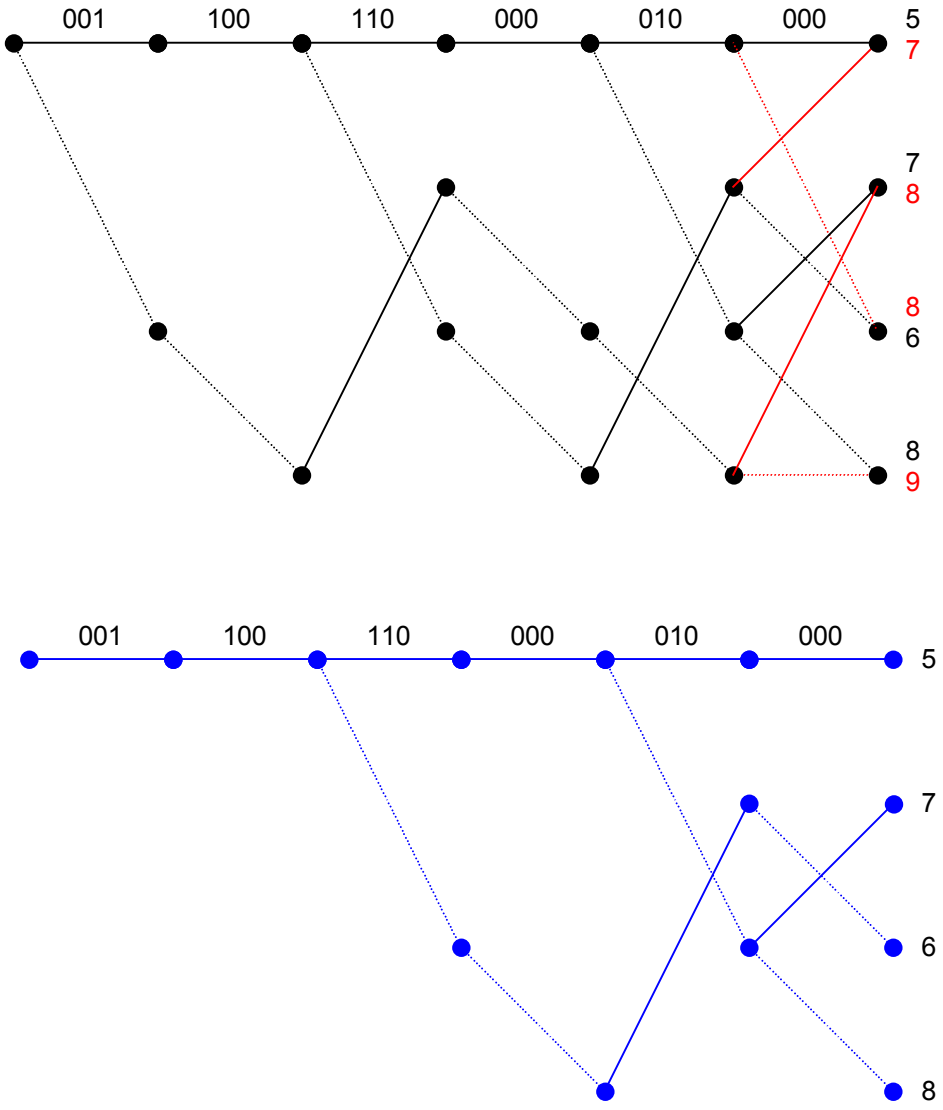

001
100
110
000
4
7
4
4
7
6
5
5
001
100
110
000
4
4
6
5

PASO 5





PASO 6



PASO FINAL

Por último, una vez analizadas todas las métricas para la secuencia recibida completa, nos quedan cuatro caminos supervivientes. De entre todos ellos, nos quedamos con aquel cuya métrica sea menor. En este ejemplo, dicho camino es el {000 000 000 000 000 000}, que coincide con el mensaje que se transmitió.

En general, un código convolucional binario en el que se desplazan *k* bits dentro del descodificador en cada instante, y con longitud reducida *K*, consta de un registro de desplazamiento de *Kk* posiciones que generan un diagrama de Trellis de $2^{k(K-1)}$ estados. En consecuencia, la descodificación de un código por medio del algoritmo de Viterbi requiere del cálculo y almacenamiento de $2^{k(K-1)}$ caminos supervivientes y $2^{k(K-1)}$ métricas. En cada etapa del Trellis existen 2^k caminos entrantes en cada nodo. Como cada camino que llega al mismo nodo requiere que se calcule su métrica, es necesario calcular 2^k métricas en cada estado. De los 2^k caminos que llegan a cada nodo, sólo uno sobrevive, es el más probable, el **camino de mínima distancia**. Por tanto**,** el número de operaciones en la implementación del algoritmo crece exponencialmente con *k* y *K*. Este crecimiento exponencial en la carga computacional limita el uso del algoritmo de Viterbi a valores relativamente pequeños de los parámetros *k* y *K* [10].

2.4.3.1. ALGORITMO DE VITERBI MODIFICADO

Normalmente, cuando se tienen tramas de información codificadas de gran longitud, el **retraso de descodificación** es grande y suele ser inadmisible para la mayoría de las aplicaciones prácticas. Además, la memoria requerida para almacenar los caminos supervivientes es costosa y de gran tamaño. Una posible solución a este problema consiste en modificar el algoritmo de Viterbi, de modo que el retraso de descodificación quede fijado a un valor sin afectar, de forma significativa, a la implementación óptima del algoritmo. La modificación consiste en quedarse en cada momento solamente con los δ últimos símbolos o bits descodificados de cada camino superviviente. Cuando se recibe un nuevo bit o símbolo la decisión final se hace sobre los bits recibidos δ ramas atrás en el diagrama de Trellis, por comparación de las métricas de las secuencias supervivientes y decidiendo a favor del bit de la secuencia que tenga mayor métrica. Si se elige δ suficientemente grande, todas las secuencias supervivientes contendrán el mismo bit o símbolo decodificado δ ramas atrás en el tiempo. Es decir, con una alta probabilidad, todos los caminos supervivientes, en un instante dado, salen del mismo nodo que en el instante t-δ . Simulaciones mediante ordenador han demostrado que, un retraso $\delta > 5K$ produce una degradación despreciable en la implementación del algoritmo óptimo de descodificación de Viterbi [10],[11].

2.4.4. PROBABILIDAD DE ERROR

2.4.4.1. PROBABILIDAD DE ERROR PARA DECISIÓN DURA

Se considera la implementación del algoritmo de Viterbi en un canal simétrico y binario. Para simplificar el desarrollo al calcular la probabilidad de error para códigos convolucionales, se hace uso

de la propiedad de linealidad, es decir, se supone que se transmite una secuencia de ceros y se calcula la probabilidad de decidir en favor de otra secuencia distinta. Se supone que se ha utilizado modulación BPSK para transmitir los bits de la rama *j*-ésima del código, denotados por $\{c_{jm}$, *m* = 1,2,...,n;}, y definidos en el apartado 1.4.2, y que la detección es coherente en el desmodulador. La salida del desmodulador, que es la entrada del descodificador de Viterbi, es la secuencia $\{r_{jm}$, *j* = 1,2,...; *m* =1,2,...,n} donde r_{jm} se definió en (2.26).

Para la descodificación de códigos convolucionales con decisión dura, las métricas del algoritmo de Viterbi son las distancias de Hamming entre la secuencia recibida y los $2^{k(K-1)}$ caminos supervivientes en cada nodo del diagrama de Trellis.

Hay que comenzar por determinar la **probabilidad de error del primer suceso** *(first-event error probability)*, que se define como la probabilidad de que otro camino que coincide con el camino todo-cero en el nodo *B*, por primera vez, tenga una métrica que supere a la métrica del camino todo-cero. Se supone pues que se transmite la secuencia todo-cero.

El camino que va a ser comparado con el todo-cero en un nodo *B* determinado está separado una distancia *d* de éste. Si *d* es impar, el camino todo-cero será seleccionado si el número de errores en la secuencia recibida es menor que (*d*+1)/2; en otro caso el camino incorrecto será el elegido. En consecuencia, la probabilidad de elegir el camino incorrecto es

$$P_2(d) = \sum_{k=(d+1)/2}^{d} \binom{d}{k} p^k (1-p)^{d-k} , \tag{2.37}$$

donde *p* es la probabilidad de error de bit para un canal simétrico y binario. A $P_2(d)$ se le denomina **probabilidad de error de caminos paralelos**. Se consideran caminos paralelos a los caminos que coinciden en un mismo nodo tras un número determinado de transiciones a lo largo del árbol de Trellis. Si *d* es par, el camino incorrecto será seleccionado cuando el número de errores supere *d*/2. Si el número de errores es igual a *d*/2, hay un empate entre las métricas de los dos caminos que se resuelve eligiendo uno de ellos al azar; habrá un error la mitad de las veces. Así pues, la probabilidad de elegir el camino incorrecto es

$$P_2(d) = \sum_{k=\frac{d}{2}+1}^{d} \binom{d}{k} p^k (1-p)^{d-k} + \frac{1}{2}\binom{d}{d/2} p^{d/2}(1-p)^{d/2} . \tag{2.38}$$

Como ya ha sido comentado con anterioridad, existen varios caminos posibles que convergen con el todo-cero en algún nodo dado. No existe una expresión simple y exacta para la probabilidad de error del primer suceso. Sin embargo, podemos acotar su valor mediante la suma de las

$P_2(d)$ de todos los caminos que coinciden con el camino todo-cero en un nodo dado. Dicha probabilidad se calcula a partir de la comparación entre las métricas de los dos caminos, el todo-cero y el erróneo. Así se obtiene la cota conjunta

$$P_e < \sum_{d=d_{free}}^{\infty} a_d P_2(d) \ , \tag{2.39}$$

donde los coeficientes $\{a_d\}$ representan el número de caminos correspondientes al conjunto de distancias $\{d\}$. Estos coeficientes son los **coeficientes de expansión** de la función de transferencia *T(D)* o *T(D,N)*.

En lugar de emplear la expresión para $P_2(d)$ dada por (2.37) y (2.38) podemos usar la cota superior

$$P_2(d) < \left[4p(1-p)\right]^{d/2} . \tag{2.40}$$

El uso de la ecuación (2.40), conduce a una cota superior de la probabilidad de error del primer suceso, dada por

$$P_e < \sum_{d=d_{free}}^{\infty} a_d \left[4p(1-p)\right]^{d/2} < T(D)\,|_{D=\sqrt{4p(1-p)}} \ . \tag{2.41}$$

Aunque la probabilidad de error del primer suceso da una medida del funcionamiento de un código convolucional, una medida más útil es la probabilidad de error de bit. Esta probabilidad puede ser acotada superiormente siguiendo el mismo proceso empleado para la probabilidad de error del primer suceso. Cuando se selecciona un camino incorrecto, los bits de información en los que se diferencia del correcto se decodifican de forma inadecuada. El exponente del factor *N* contenido en la función de transferencia *T(D,N)* indica el número de bits de información erróneos (número de bits a 1) al seleccionar un camino erróneo que confluye con el todo-cero en un nodo *B*. Si se multiplica la probabilidad de error de caminos paralelos $P_2(d)$ por el número de bits erróneos descodificados en el nodo en el que ambos caminos convergen, el incorrecto y el todo-cero, se obtiene la **tasa de error de bit** para ese camino. Los factores multiplicativos adecuados correspondientes al número de bits de información erróneos para cada camino incorrecto seleccionado puede obtenerse derivando la función de transferencia *T(D,N)* con respecto a *N*. En general, *T(D,N)* puede expresarse como

$$T(D,N) = \sum_{d=d_{free}}^{\infty} a_d D^d N^{f(d)} \ , \tag{2.42}$$

donde $f(d)$ indica que el exponente de *N* es una función de *d*. Tomando derivadas con respecto a *N* y particularizando para *N* = 1, se llega a

$$\left.\frac{dT(N,D)}{dN}\right|_{N=1} = \sum_{d=d_{free}}^{\infty} a_d f(d) D^d = \sum_{d=d_{free}}^{\infty} \beta_d D^d \ , \tag{2.43}$$

donde $\beta_d = a_d f(d)$.

A continuación se procederá al estudio de la **probabilidad de error de bit**. Para ello, se hará uso del hecho de que los exponentes de los factores *N* que aparecen en la función de transferencia *T(D,N)* indican el número de bits distintos de cero, y por tanto erróneos, que aparecen cuando se elige un camino incorrecto en lugar del camino todo-cero. Trabajaremos sobre el ejemplo de la figura 2.13. Derivando *T(D,N)* con respecto a *N* y particularizando para *N* = 1, los exponentes de *N* se convierten en factores multiplicativos de las correspondientes probabilidades de error $P_2(d)$.

$$\frac{dT(D,N)}{dN} = D^6 + 4ND^8 + 12N^2D^{10} + ... = \beta_d \cdot D^d N^{\left(\frac{d-4}{2}-1\right)}. \tag{2.44}$$

Se obtiene así la cota superior para la probabilidad de error de bit

$$P_b < \sum_{d=d_{free}}^{\infty} \beta_d P_2(d), \tag{2.45}$$

donde $\{\beta_d\}$ son los coeficientes del desarrollo de la derivada de la función de transferencia particularizados para *N* =1.

$$\beta_d = 2^{\frac{d-6}{2}} \cdot \frac{d-4}{2}. \tag{2.46}$$

Para $P_2(d)$ se pueden usar cualquiera de las expresiones dadas por (2.37) y (2.38) o la cota superior dada por (2.40). Si ésta última es la elegida, el límite superior para P_b puede expresarse como

$$P_b < \left.\frac{dT(D,N)}{dN}\right|_{N=1, D=\sqrt{4p(1-p)}} . \tag{2.47}$$

Cuando *k* > 1, los resultados dados por (2.45) y (2.47) deben ser divididos por *k* [10]. Se ob-tienen *n* bits de salida por cada *k* bits de entrada. Para una trama de entrada de longitud *L* se genera una trama de salida de longitud $\frac{L}{k}.n$. La probabilidad de bit en (2.45) y (2.47) ha sido calcula-da para *k* = 1. Cuando *k* > 1, el número de bits se reduce *k* veces, lo que implica la misma reducción en la probabilidad de error.

2.4.4.2. PROBABILIDAD DE ERROR PARA DECISIÓN BLANDA

En este apartado se va a abordar el estudio de la tasa de error cuando se utiliza un descodificador de Viterbi con decisión blanda en un canal con ruido gaussiano, blanco y aditivo.

El descodificador de Viterbi con detección blanda forma las métricas de las ramas tal y como se muestra en (2.35) y a partir de ellas calcula las métricas asociadas a los caminos

$$CM^{(i)} = \sum_{j=1}^{B} \mu_j^{(i)} = \sum_{j=1}^{B} \sum_{m=1}^{n} r_{jm}(2c_{jm}^{(i)} - 1) \ , \tag{2.48}$$

donde *i* denota cada uno de los caminos entrantes a un nodo y *B* es el número de ramas (símbolos de información) en un camino. Por ejemplo, el camino todo-cero, denotado como *i* = 0, tiene la siguiente métrica

$$CM^{(0)} = \sum_{j=1}^{B} \sum_{m=1}^{n} (-\sqrt{\varepsilon_c} + n_{jm})(-1) = \sqrt{\varepsilon_c} Bn + \sum_{j=1}^{B} \sum_{m=1}^{n} n_{jm} \quad . \tag{2.49}$$

Como el código convolucional no tiene una longitud fija necesariamente, derivamos su desarrollo de la probabilidad de error de las secuencias que confluyen, por primera vez, con la secuencia todo-cero en un nodo determinado del Trellis. Supongamos que el camino incorrecto, al que llamamos *i* =1, que coincide con el camino todo-cero en un nodo, se diferencia de éste en *d* bits. Es decir, en el camino *i* = 1 hay *d* bits a 1 y el resto a 0. La probabilidad de error de los caminos paralelos *(pairwise error probability)* se calcula a partir de la comparación de las métricas $CM^{(0)}$ y $CM^{(1)}$

$$P_2(d) = P(CM^{(1)} \geq CM^{(0)}) = P(CM^{(1)} - CM^{(0)} \geq 0) = P\left[2\sum_{j=1}^{B} \sum_{m=1}^{n} r_{jm}(c_{jm}^{(1)} - c_{jm}^{(0)}) \geq 0 \right]. \tag{2.50}$$

Como los bits codificados en los dos caminos son idénticos salvo por *d* posiciones, la ecuación (2.50) puede ser rescrita de una forma más simple

$$P_2(d) = P\left(\sum_{l=1}^{d} r_l^{'} \geq 0 \right) \ , \tag{2.51}$$

donde el índice *l* recorre el conjunto de los *d* bits en los que se diferencian ambos caminos y el conjunto $\{r_l^{'}\}$ representa las entradas al descodificador para esos *d* bits.

Los elementos de $\{r_l^{'}\}$ son variables independientes y están idénticamente distribuidos según una variable aleatoria gaussiana de media $-\sqrt{\varepsilon_c}$ y varianza $1/2N_0$. En consecuencia, la probabilidad de error en la comparación de estos dos caminos que difieren en *d* bits es

$$P_2(d) = Q\left(\sqrt{\frac{2\varepsilon_c}{N_0}d}\right) = Q\left(\sqrt{2\gamma_b R_c d}\right), \tag{2.52}$$

donde $\gamma_b = \varepsilon_b / N_0$ es la SNR por bit recibido y R_c es la tasa de código.

Aunque se ha calculado la probabilidad de error del primer suceso para un camino cuya distancia al todo-cero es *d* bits, existen otros muchos caminos posibles que confluyen en un nodo cualquiera *B* con el camino todo-cero. De hecho, la **función de transferencia** *T(D)* da una descripción completa de todos los caminos posibles que confluyen con el camino todo-cero en un nodo *B* determinado así como de sus distancias. De este modo, se puede sumar la probabilidad de error dada por (2.52) a lo largo de todas las distancias entre caminos posibles, obteniendo una cota superior para la probabilidad de error del primer suceso,

$$P_e \le \sum_{d=d_{free}}^{\infty} a_d P_2(d) \le \sum_{d=d_{free}}^{\infty} a_d Q\left(\sqrt{2\gamma_b R_c d}\right), \tag{2.53}$$

donde a_d denota el número de caminos cuya distancia al camino todo-cero es *d* y que coinciden con él por primera vez.

Hay dos razones por las cuales (2.53) es una cota superior de la probabilidad de error del primer suceso. La primera es que los sucesos o eventos resultantes en las probabilidades de error $\{P_2(d)\}$ son no disjuntos, conclusión a la que se puede llegar a partir de la observación del diagrama de Trellis. En segundo lugar, sumando para $d \ge d_{free}$, se ha supuesto implícitamente que el código convolucional tiene longitud infinita. Si el código se trunca periódicamente tras *B* nodos, la cota superior dada en (2.53) se puede mejorar sumando sólo para aquellos sucesos tales que $d_{free} \le d \le B$. La nueva cota calculada mejora a la anterior en el caso de códigos convolucionales cortos; sin embargo, el efecto es despreciable cuando *B* es grande.

La cota definida en (2.53) puede expresarse de forma ligeramente distinta si utilizamos la acotación de la función Q por la exponencial, es decir,

$$Q\left(\sqrt{2\gamma_b R_c d}\right) \le e^{-\gamma_b R_c d} = D^d \,|_{D=e^{-\gamma_b R_c}}. \tag{2.54}$$

Si se introduce (2.53) en (2.54) se llega a la nueva expresión para la probabilidad de error del primer suceso,

$$P_e < T(D)\,|_{D=e^{-\gamma_b R_c}}. \tag{2.55}$$

Por tanto, la probabilidad de error de bit para *k* = 1 está limitada superiormente por

$$P_b < \sum_{d=d_{free}}^{\infty} \beta_d P_2(d) < \sum_{d=d_{free}}^{\infty} \beta_d Q\left(\sqrt{2\gamma_b R_c d}\right). \tag{2.56}$$

Si la función Q está limitada por la exponencial tal y como se mostró en la ecuación (2.54) entonces (2.56) puede expresarse de una forma más simple

$$P_b < \sum_{d=d_{free}}^{\infty} \beta_d D^d \mid_{D=e^{-\gamma_b R_c}} < \frac{dT(D,N)}{dN} \Bigg|_{N=1, D=e^{-\gamma_b R_c}} . \tag{2.57}$$

Si *k* > 1, la probabilidad de error de bit equivalente se consigue dividiendo (2.56) o (2.57) por *k*.

La expresión de la probabilidad dada arriba está basada en la suposición de que los bits de información han sido transmitidos mediante una BPSK coherente. Los resultados se mantienen si la modulación empleada es una 4-PSK, ya que la desmodulación se desacopla en dos BPSK, una en fase y la otra en cuadratura. Para otro tipo de modulaciones, como la FSK binaria coherente o no coherente, la expresión puede ser utilizada simplemente recalculando la probabilidad de error $P_2(d)$. Es decir, un cambio en la técnica de modulación y desmodulación empleada para transmitir la secuencia de bits de información sólo afecta al cálculo de $P_2(d)$. Por lo demás, el desarrollo hasta obtener P_b permanece igual.

Aunque el análisis realizado para el cálculo de la probabilidad de error para la descodificación de Viterbi de un código convolucional se ha hecho para códigos binarios, resulta relativamente sencillo generalizarlo para códigos convolucionales no binarios. En particular, los coeficientes $\{\beta_d\}$ que aparecen en la expresión de la derivada de la función de transferencia (2.43), representan el número de símbolos erróneos en dos caminos separados una distancia (medida en términos de símbolos) de *d* símbolos. De nuevo, $P_2(d)$ denota la probabilidad de error de caminos paralelos de dos caminos separados una distancia *d*. De modo que la probabilidad de error de símbolo, para símbolos de *k* bits, está limitada superiormente por

$$P_M = \sum_{d=d_{free}}^{\infty} \beta_d P_2(d) . \tag{2.58}$$

La probabilidad de error de símbolo puede convertirse en una probabilidad de error de bit equivalente. Por ejemplo, si 2^k formas de onda ortogonales son usadas para transmitir los símbolos de *k* bits, la probabilidad de error de bit equivalente es P_M multiplicada por un factor $\frac{2^{k-1}}{2^k - 1}$ [10],

$$\frac{2^{k-1}}{2^k - 1} \cdot P_M = \frac{2^{k-1}}{2^k - 1} \cdot \sum_{d=d_{free}}^{\infty} \beta_d P_2(d) . \tag{2.59}$$