Codificació de canal II: codis convolucionals

Margarita Cabrera
Francesc Tarrés

PID_00185015

UOC
Universitat Oberta de Catalunya

Índex

Introducció

Cal considerar aquest mòdul com una extensió i una continuació del mòdul anterior. Com hem vist, la codificació de canal consisteix a definir amb precisió el conjunt de símbols amb els quals s'enviaran els missatges entre l'emissor i el receptor. Aquesta definició tan genèrica admet que el conjunt de regles sistemàtiques per a determinar la seqüència de bits de sortida en funció de la seqüència d'entrada es pugui triar amb l'objectiu de solucionar problemes de diversa naturalesa, i tenir així una gran flexibilitat en funció de les aplicacions i de les possibilitats de procés de dades que es disposi.

En dissenyar estratègies per a protegir la informació d'errors eventuals, no tenim massa marge de maniobra. En efecte, l'única solució possible és que el receptor pugui distingir entre un missatge produït per l'emissor i un missatge en què s'han produït errors. Per a aconseguir-ho, cal que introduïm elements addicionals en la informació que ens proporcionin pistes sobre la integritat dels missatges. Aquesta introducció d'informació addicional rep el nom de *redundància estructurada* i suposa que el nombre de bits d'informació que finalment es transmeten al canal sigui superior al mínim necessari. Així doncs, la inserció de codis de protecció d'errors suposa la introducció de bits addicionals en el transmissor, fet que representa un augment de l'amplada de banda del senyal que transmetem. El preu que cal pagar per a protegir el missatge és, per tant, augmentar l'amplada de banda de la informació.

El mòdul comença amb una descripció de l'estructura general dels codis convolucionals. Aquest tipus de codis són molt eficients i representen una alternativa als codis de bloc utilitzada en moltes aplicacions. En la pràctica, l'interès dels codis convolucionals se centra en el fet que molts sistemes de comunicacions utilitzen de manera simultània codis de bloc i codis convolucionals. Veurem que, quan aquests dos tipus de codi es combinen adequadament (concatenació de codis), es poden aconseguir propietats de protecció que no és possible obtenir amb l'ús únic dels codis individuals. La concatenació d'aquests dos tipus de codi s'utilitza en diversos sistemes de comunicació com la televisió digital terrestre (DVB-T), els sistemes WiFi, els sistemes de telefonia mòbil, etc.

En la primera part del mòdul s'analitzen diverses maneres de representar i analitzar el funcionament dels codis convolucionals, i ens centrem principalment en els diagrames de Trellis. Un cop estudiat el procés de codificació, s'exposa amb detall el procés de descodificació d'una seqüència, utilitzant el conegut algorisme de Viterbi, sobre el qual se centra la major part del contingut d'aquest mòdul. L'algorisme de Viterbi ens permet determinar quins són els bits correctes i incorrectes de la seqüència transmesa. El mòdul continua amb

una presentació general dels conceptes de perforació de codis, concatenació de codis i exemples d'alguns sistemes de comunicació que utilitzen codis convolucionals. Finalment, es presenten les característiques generals dels denominats *turbocodis*. Aquests codis són bastant complexos des d'un punt de vista matemàtic i els seus detalls van més enllà dels objectius generals d'aquest text.

Es proporcionen alguns exemples i problemes resolts al llarg del text i al final del mòdul. També s'hi inclouen alguns exercicis perquè l'estudiant els resolgui de manera autònoma. Tots aquests exercicis i exemples constitueixen els problemes més típics que es poden realitzar sobre codis convolucionals.

Objectius

Els objectius que ha d'assolir l'estudiant amb aquest mòdul són els següents:

1. Comprendre la necessitat de protegir la informació d'errors del canal.

2. Diferenciar entre els codis de bloc i els codis convolucionals.

3. Comprendre els elements matemàtics bàsics per a realitzar codis convolucionals.

4. Dominar l'algorisme de descodificació de Viterbi.

5. Comprendre les necessitats i les característiques de la concatenació de codis.

6. Comprendre els mecanismes de protecció davant d'errors utilitzats en alguns sistemes de comunicació reals (prenent com a exemple la DVB).

1. Codis convolucionals

La característica principal dels codis convolucionals és que el procés de codificació té memòria, fet pel qual els bits en la sortida del codificador depenen no solament de la informació actual sinó també dels bits anteriors. La memòria d'un codi convolucional ens indica en quin grau els bits codificats depenen dels valors que ha pres la informació original en el passat. A la figura 1 es mostra un diagrama de blocs molt senzill corresponent a un codi convolucional que podem utilitzar com a exemple per a mostrar les característiques essencials d'aquests codis.

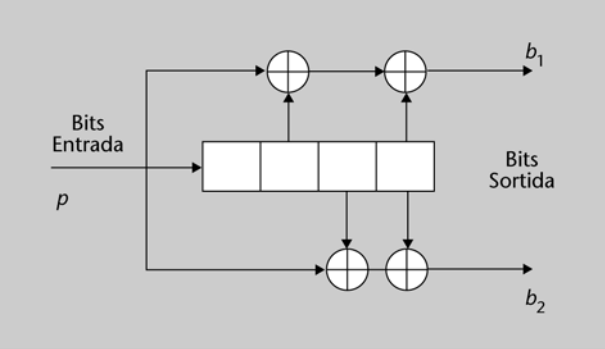

Figura 1. Diagrama de blocs d'un exemple de codi convolucional

En aquest exemple es mostra un conjunt de registres de desplaçament (en total 4 unitats de memòria) que emmagatzemen els valors anteriors de la seqüència d'entrada. Els bits del codi es determinen tenint en compte el bit actual i els bits retardats, realitzant les operacions de sumes exclusives indicades en el diagrama. Per cada bit de la seqüència d'entrada s'obtenen 2 bits en la seqüència de sortida que s'hauran d'alternar en aquesta sortida. Així doncs, es tracta d'un codi amb una taxa $R = k/n = 1/2$. En aquest exemple, els bits de sortida depenen del valor del bit actual d'entrada i del valor dels 4 bits anteriors. Aquesta situació es pot interpretar com si els 4 bits anteriors definissin un estat del codificador i que la sortida depèn d'aquest estat i del valor del bit actual. A més, cada cop que es rep un bit nou, l'estat del codificador varia. En el nostre exemple, com que el nombre de bits que defineix els possibles estats del codificador és 4, el nombre total d'estats del codificador és de 2^4. Finalment, hem de comentar que les característiques del codi són determinades per les operacions de suma exclusiva i els bits retardats que s'utilitzen. Aquestes connexions s'indiquen mitjançant un polinomi generador. Així, el polinomi generador associat a la branca inferior (b_2) es denota $G_2 = X^4 + X^3 + 1$, i indica que es realitza la suma exclusiva entre els bits de les posicions 4 i 3 (mostres retardades) del registre de desplaçament amb el bit de l'entrada. Els polinomis generadors es poden trobar en taules i es proporcionen amb notació octal. En el nostre exemple, el polinomi expressat en notació octal seria $31_{OCT} = (11\ 001)$*.

*** La notació octal utilitza els dígits del 0 al 7. Per a passar d'aquesta notació a binari convencional, n'hi ha prou de substituir cada dígit pel seu codi binari de 3 bits. Així, per exemple, el nombre octal 541 serà 101 100 001.**

Com a resum, podem identificar les característiques del codificador convolucional de la figura 1 de la manera següent:

- Nombre de línies d'entrada: $k = 1$
- Nombre de línies de sortida: $n = 2$
- Taxa del codi: $R = 1/2$
- Unitats de memòria: 4
- Nombre de possibles estats: $2^4 = 16$
- Longitud total de la convolució: 5
- Polinomi generador 1: $G_1 = X^4 + X^2 + 1 = 25_{OCT}$
- Polinomi generador 2: $G_2 = X^4 + X^3 + 1 = 31_{OCT}$

1.1. Diagrama de blocs genèric d'un codificador convolucional

A la figura 3 es mostra el diagrama de blocs genèric d'un codificador convolucional amb una taxa de codi $R = k/n$. En general, es pot suposar que hi ha un total de L etapes, cadascuna de les quals amb k bits. Per a obtenir el codi corresponent a una determinada seqüència de bits, hem de suposar que en l'entrada les dades s'estructuren en paquets de k bits, i que posteriorment es calculen els n bits corrssponents a la sortida. En l'exemple de la figura 3 el nombre total d'etapes de memòria és $L \cdot k$.

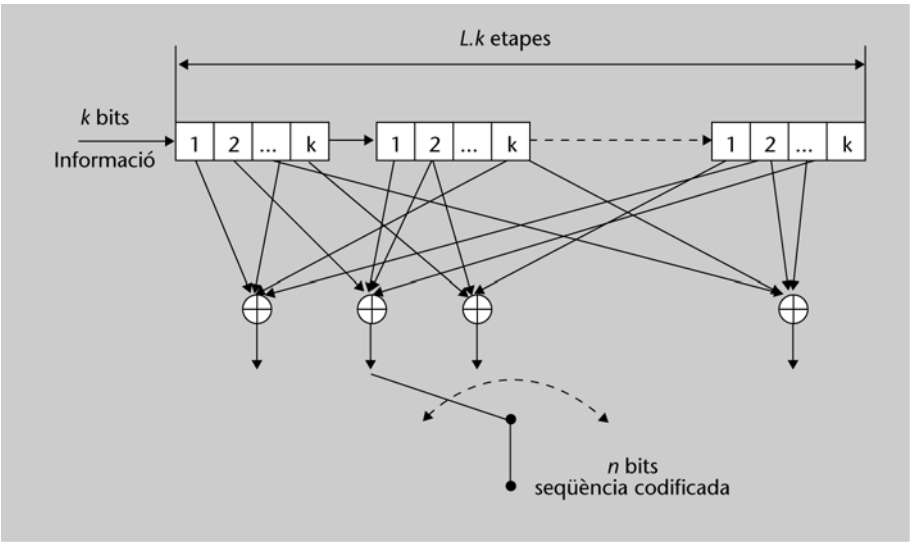

Figura 3. Diagrama genèric d'un codificador convolucional

1.2. Diagrames d'estat i diagrames de Trellis

Un codificador convolucional té una memòria finita, fet pel qual és possible representar tots els seus possibles estats i les seves transicions mitjançant un graf denominat *diagrama d'estat*. Per a veure com es representa un codificador mitjançant un diagrama d'estat, considerarem un exemple molt simple repre-

sentat a la figura 4. En aquesta figura, l'entrada s'ha representat a la dreta i la sortida, a l'esquerra perquè els estats concordin amb el contingut dels registres de desplaçament.

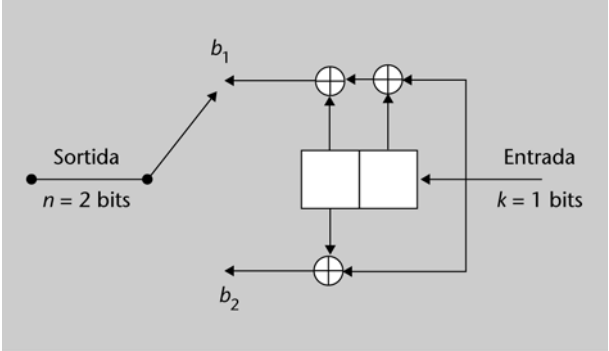

Figura 4. Codificador convolucional simple per a il·lustrar els diagrames d'estat i de Trellis

El codificador és caracteritzat pels paràmetres següents:

- Nombre de línies d'entrada: $k = 1$
- Nombre de línies de sortida: $n = 2$
- Taxa del codi: $R = 1/2$
- Unitats de memòria: 2
- Nombre de possibles estats: $2^2 = 4$
- Longitud total de la convolució: 3
- Polinomi generador 1: $G_1 = X^2 + X + 1 = 7_{OCT}$
- Polinomi generador 2: $G_2 = X^2 + 1 = 5_{OCT}$

Els possibles estats en què pot estar el sistema es corresponen amb els valors que poden prendre els dos registres interns, és a dir, {00, 01, 10 i 11}. En el diagrama d'estat, els estats s'indiquen amb un cercle a l'interior del qual apareixen els valors dels registres. Cada cop que hi entra un bit nou, l'estat intern dels registres es modifica, fet pel qual és possible que es produeixi una transició d'estats. Les transicions entre estats s'indiquen amb fletxes que es retolen amb 3 bits en la forma X/XX. El primer bit indica el valor de l'entrada que produeix la transició i els dos bits de després de la barra indiquen els dos valors que prendrà la sortida. En cas que el nombre de línies d'entrada i sortida sigui diferent, les fletxes es retolen amb el nombre de bits apropiat. De cada estat poden sortir dues fletxes dirigides a altres estats (o a aquest mateix) i a cadascun li arriben també dues fletxes.

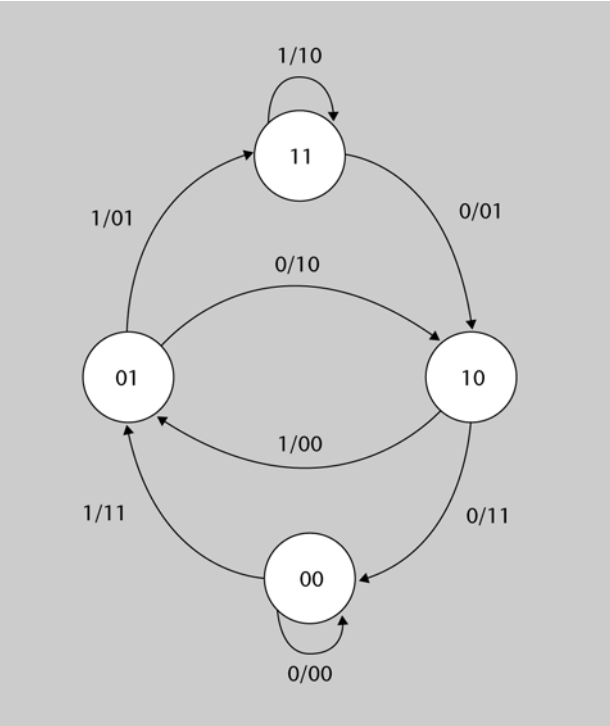

Figura 5. Diagrama d'estat corresponent al codificador convolucional de la figura 4

Activitat

Comproveu que el diagrama d'estat representat a la figura 5 concorda amb el codificador convolucional representat a la figura 4.

El diagrama de Trellis és una representació alternativa que també ens indica els diversos estats del codificador i les seves possibles transicions en funció de la informació d'entrada. La diferència més important és que en aquesta representació es mostra l'evolució de les transicions en el temps. De fet, el diagrama de Trellis és una seqüència en l'eix temporal de les possibles transicions que es puguin produir en el codificador a mesura que hi entra informació nova. En l'eix vertical es representen tots els possibles estats del codificador. Les transicions es representen mitjançant fletxes. En el cas concret en què solament hi hagi una línia d'entrada, se sol utilitzar una línia contínua per a representar l'entrada d'un zero i una línia discontínua per a un u.

A la figura 6 es representa el diagrama de Trellis associat al codificador de la figura 4. Observeu que suposem que l'estat inicial del codificador és el {00}, fet pel qual els possibles estats que pot prendre el sistema evoucionen al llarg del temps fins a arribar a una mateixa representació gràfica que es repeteix en cada transició (quan tots els estats han pogut ser assolits). En cada fletxa s'indiquen els valors que prenen les línies de sortida (en el nostre cas les dues línies). Si el nombre de línies d'entrada fos major, és convenient retolar les fletxes utilitzant la mateixa notació que en el diagrama d'estat.

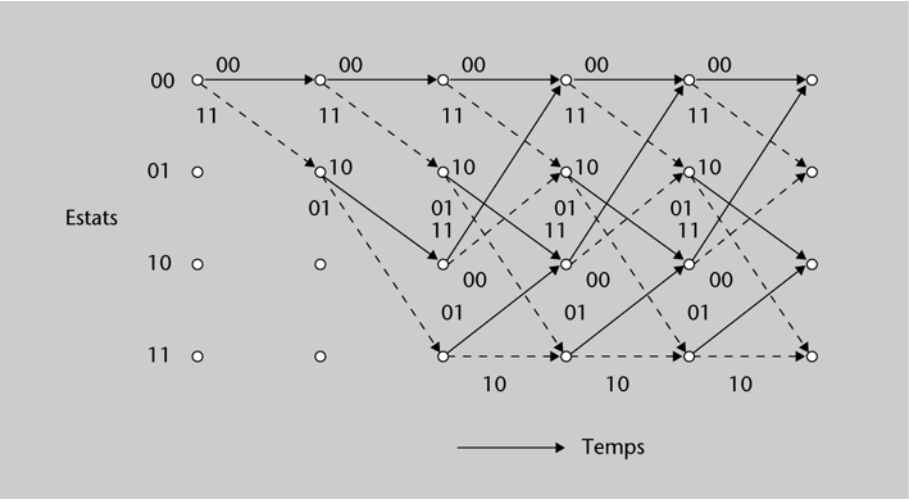

Figura 6. Diagrama de Trellis del codificador convolucional de la figura 4

Activitat

Comproveu que el diagrama de Trellis de la figura 6 es correspon amb el codificador de la figura 4.

1.3. Procés de codificació d'una seqüència

La sistemàtica per a codificar una seqüència d'informació mitjançant un codi convolucional és molt simple. Es pot realitzar directament a partir del diagrama de blocs del codificador, veient com evolucionen els registres a mesura que s'introdueix la informació i realitzant les operacions de sumes exclusives indicades. No obstant això, resulta molt més simple efectuar el càlcul d'una manera més sistemàtica utilitzant els diagrames d'estat o els diagrames de Trellis. El diagrama de Trellis proporciona una valuosa informació de com han anat evolucionant els estats del codificador en el temps.

En general, per a determinar la seqüència codificada hem d'introduir un bloc de k bits de la informació original (suposem que el codificador té k línies), i en funció d'aquests k bits determinar els n bits que obtindrem en la sortida i el nou estat al qual accedirem. Habitualment, se suposa que el codificador en el seu estat inicial té tots els registres a zero. A més, quan determinem la sortida a una determinada seqüència d'entrada, sempre afegim zeros suficients a la informació d'entrada perquè el codificador torni a l'estat inicial zero en acabar la codificació.

Considerem un exemple senzill amb el codificador que estem utilitzant en aquest apartat (figura 4). Suposem que la seqüència que volem codificar és: {0, 1, 1, 1, 0, 1, 1}. En el nostre cas, com que solament tenim una línia d'entrada, els bits entraran al codificador d'un en un. Atès que els dos últims bits són 1, l'estat final en què quedarà el codificador és el {11}. Per tant, haurem d'estendre la seqüència d'entrada amb zeros perquè l'estat final sigui el {00}. Així, la seqüència d'entrada definitiva serà: {0, 1, 1, 1, 0, 1, 1, 0, 0}. Vegem com es realitza la codificació de la seqüència amb l'ajuda de la taula següent, en què utilitzem la informació del diagrama d'estats de la figura 5.

Estat inicial	Entrada	Sortida	Estat final
00	0	00	00
00	1	11	01
01	1	01	11
11	1	10	11
11	0	01	10
10	1	00	01
01	1	01	11
11	0	01	10
10	0	11	00

Tabla 1. Successius estats del codificador per a la seqüència d'entrada de l'exemple

La manera de construir la taula és molt simple. Inicialment estem en l'estat 00 i rebem un zero, fet pel qual romandrem en el mateix estat i obtindrem una sortida de 00. Quan l'entrada següent val la unitat, passem a l'estat 01 i, en aquesta transició, la sortida del sistema és 11. Així anem completant la taula i obtenim la següent sortida del codificador: {00 11 01 10 01 00 01 01 11}.

El diagrama de Trellis ens proporciona una visió més gràfica de com es va realitzant la codificació de la seqüència. A la figura adjunta s'han representat en traç gruixut les transicions de la taula anterior sobre el diagrama de Trellis. Les transicions que es corresponen amb un zero en l'entrada s'indiquen amb traç continu. Anàlogament, els uns s'indiquen usant traç discontinu.

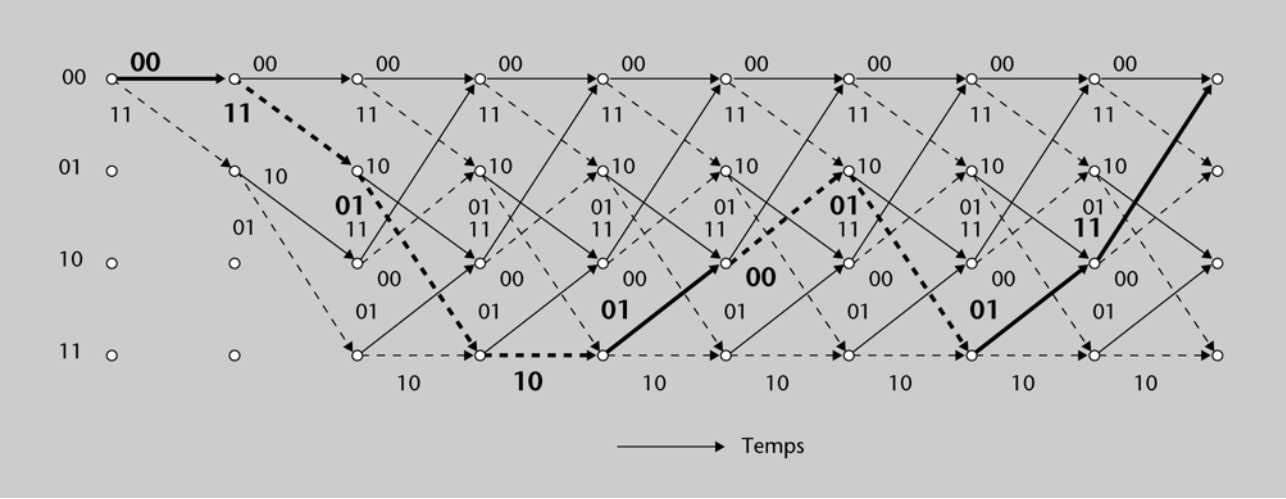

Figura 7. Evolució en el diagrama de Trellis en codificar un missatge

Amb el diagrama de Trellis és fàcil seguir com van evolucionant els estats en el codificador a mesura que apareixen dades noves. Amb una simple inspecció del diagrama, tenim informació de quina ha estat la seqüència d'entrada (0 si la línia és contínua o 1 si és discontínua), de quina és la seqüència d'estats interns del codificador (el nivell en la posició vertical) i dels bits codificats en la sortida del sistema (parells de bits en negreta).

1.4. Descodificació d'una seqüència. Algorisme de Viterbi

El procés de descodificació d'una seqüència és una mica més complex que el de codificació a causa que si es produeixen errors en la seqüència, no resulta trivial recuperar quina és la informació original. Per a realitzar la descodificació se sol utilitzar l'algorisme de Viterbi, un algorisme molt conegut que té aplicació en diverses àrees del processament de senyal com, per exemple, en els esquemes de desmodulació PCM, en l'estimació de màxima versembança (ML) de seqüències transmeses en canals amb interferència intersimbòlica, en alguns mètodes per al reconeixement de veu i en diversos esquemes per a la classificació de patrons. En tots aquests problemes es tracta de buscar un camí òptim (o el camí més probable) dins d'un diagrama de Trellis.

L'algorisme de Viterbi té un enunciat formal bastant complex i els detalls de les demostracions escapen als objectius d'aquest text. Per això presentarem l'algorisme mitjançant un exemple i posteriorment intentarem generalitzarne els resultats de manera que quedi clara l'essència de l'algorisme i el procediment que cal seguir per a descodificar una seqüència mitjançant aquest.

Per a començar amb l'exemple suposarem que s'ha transmès la mateixa seqüència que hem codificat en l'exemple de l'apartat anterior i que s'han produït 2 errors en les posicions que es mostren a la figura 8.

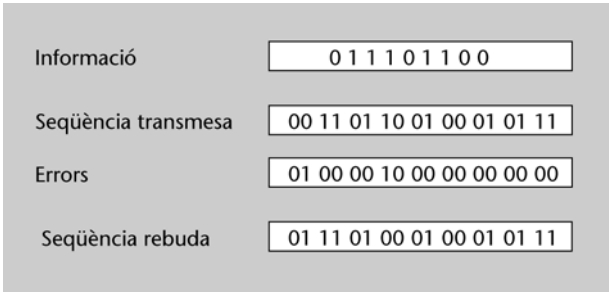

Figura 8. Seqüència de l'exemple rebuda després d'haver-se produït dos errors en el canal

Per a començar amb l'algorisme per a la descodificació de la seqüència, els bits rebuts s'agrupen en paquets de n bits (n = 2 en el nostre exemple). Per a començar la descodificació dibuixarem un diagrama de Trellis amb una profunditat (nombre d'etapes) igual al nombre de paquets de n bits de la seqüència rebuda (en el nostre cas tenim 9 paquets de 2 bits).

Una clau important per a la descodificació és que coneixem l'estat inicial del codificador (sempre comencem la codificació d'una seqüència amb tots els registres amb valor inicial zero) i també el seu estat final (ja que hem afegit zeros a la seqüència d'entrada fins a garantir que deixàvem el codificador en el seu mateix estat inicial). Per aquest motiu, sabem que el camí que ha seguit el codificador en el diagrama de Trellis comença i acaba en l'estat {00}, solament ens queda conèixer quines han estat les transicions intermèdies. De fet, tenim

alguna pista addicional. En efecte, si l'estat final és {00}, significa que els dos últims bits rebuts han estat zero.

Per començar amb el procés de descodificació, veiem que des de l'estat inicial {00} podem arribar als estats {00} o {01}. En el primer cas la sortida del sistema hauria estat 00 i en el segon hauria estat 11. Així doncs, tant en un cas com en l'altre, el que hem rebut no coincideix amb el que hauríem d'haver rebut. En tots dos casos tenim un bit correcte i un bit incorrecte. Per continuar amb el procediment de descodificació anotarem el valor 1 (nombre de bits coincidents) en els dos nodes d'arribada. En la segona etapa, si partim de l'estat {00} podem arribar al {00} o al {01} i si partim del {01}, podem anar al {10} o al {11}, amb la qual cosa ja podem haver assolit qualsevol dels 4 estats del codificador. Ara procedim a retolar cada estat amb el nombre de coincidències entre la seqüència que hauríem obtingut per a arribar a aquest estat. Totes aquestes etapes de la fase de descodificació es mostren a la figura 9. Cal observar, per exemple, que en l'etapa 2, l'estat {01} està retolat amb el nombre 3, ja que es produeixen tres coincidències entre la seqüència rebuda fins a aquest moment {01 11} i la seqüència que hauríem rebut per a arribar a aquest estat {00 11}. Anàlogament, es poden calcular els valors i les coincidències dels altres estats.

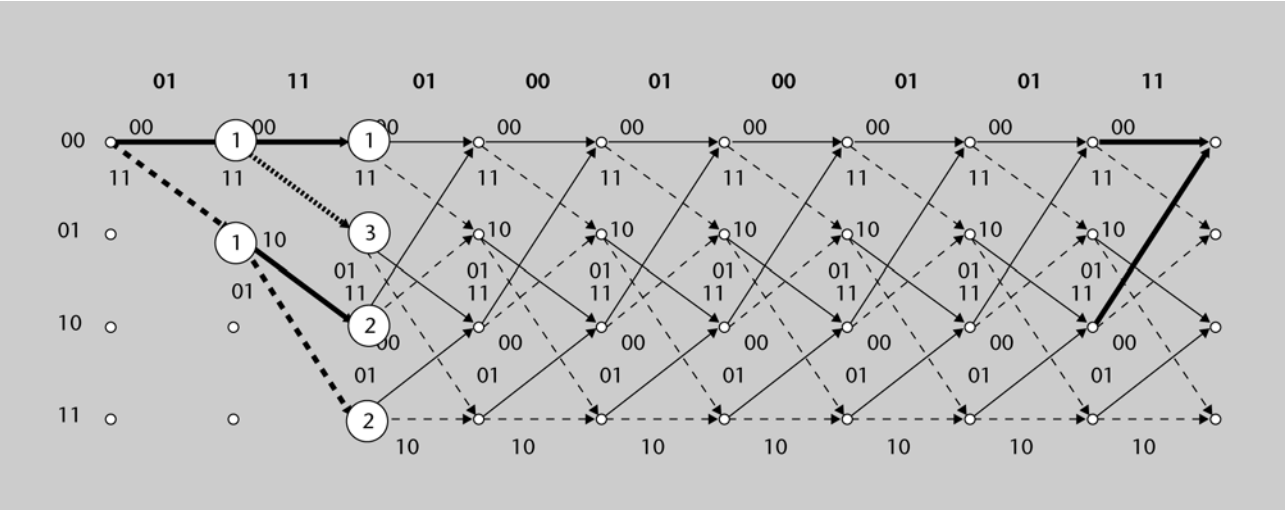

Figura 9. Primeres etapes del procés de descodificació d'una seqüència

En aquest punt, quan ja hem assolit els possibles estats, és on comença l'algorisme de Viterbi pròpiament dit. En l'etapa següent, cada estat pot ser assolit des de 2^k estats diferents (en el nostre cas 2). L'algorisme de Viterbi proposa que es calculin el nombre de coincidències que es produeixen des de cadascuna de les possibles transicions i que s'agafi el camí que produeix el màxim nombre de coincidències, tot descartant els altres.

Vegem què ocorre en el nostre exemple. Per a assolir l'estat {00}, en la tercera etapa podem procedir de l'estat {00} o de l'estat {10} en l'etapa prèvia. Si arribem des de l'estat {00}, la sortida actual serà 00, motiu pel qual solament té una coincidència amb la sortida de la tercera etapa 01. El nombre total de coincidències assignades a aquesta branca és, per tant, de dues (les acumulades en l'etapa anterior, que apareixen en el rètol, més l'actual). En canvi, si arri-

bem a l'estat {00} des del {10}, ja tenim 2 coincidències acumulades. A més, ara la sortida és 11, fet pel qual també es produeix una coincidència amb la seqüència rebuda, de manera que el nombre total de coincidències és tres. Com a resultat final ens quedem amb la branca procedent de l'estat {10} i retolem el nou estat amb el nombre de coincidències total obtingut des d'aquesta branca, és a dir, tres. Fem aquesta mateixa operació amb la resta d'estats de la tercera etapa amb la mateixa idea al pensament: dels dos possibles camins d'arribada a un estat, solament hem de considerar aquell que presenti un major nombre de coincidències i descartar l'altre. El resultat final després de realitzar totes aquestes operacions per a l'etapa tres es mostra a la figura 10.

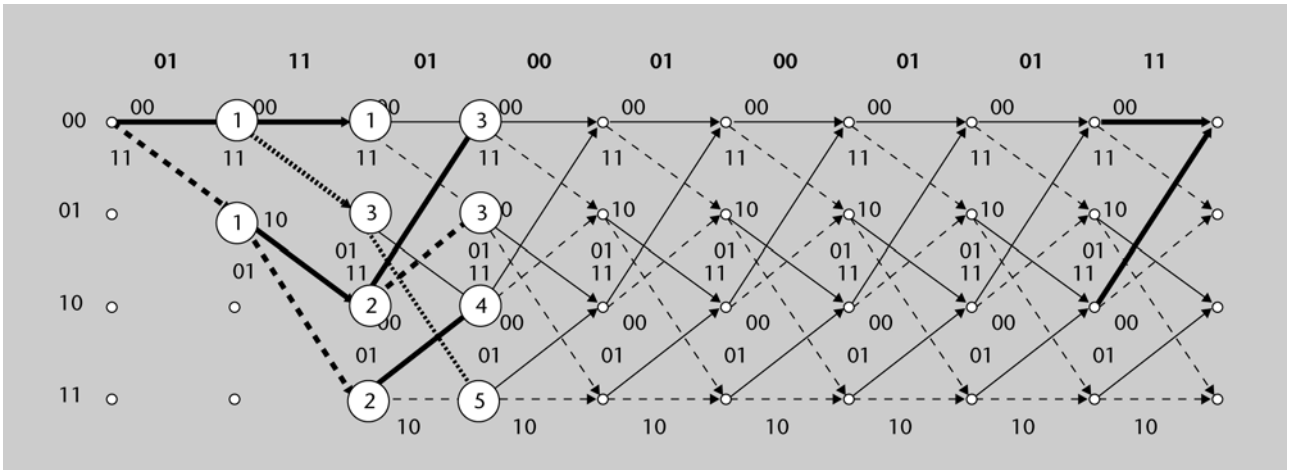

Figura 10. Procés de selecció dels camins òptims en la tercera etapa

Ara podem eliminar totes les branques (possibles camins) que han estat descartades i aquelles de les etapes anteriors que també queden en via morta. Després de realitzar aquest procés, obtenim el diagrama representat a la figura 11.

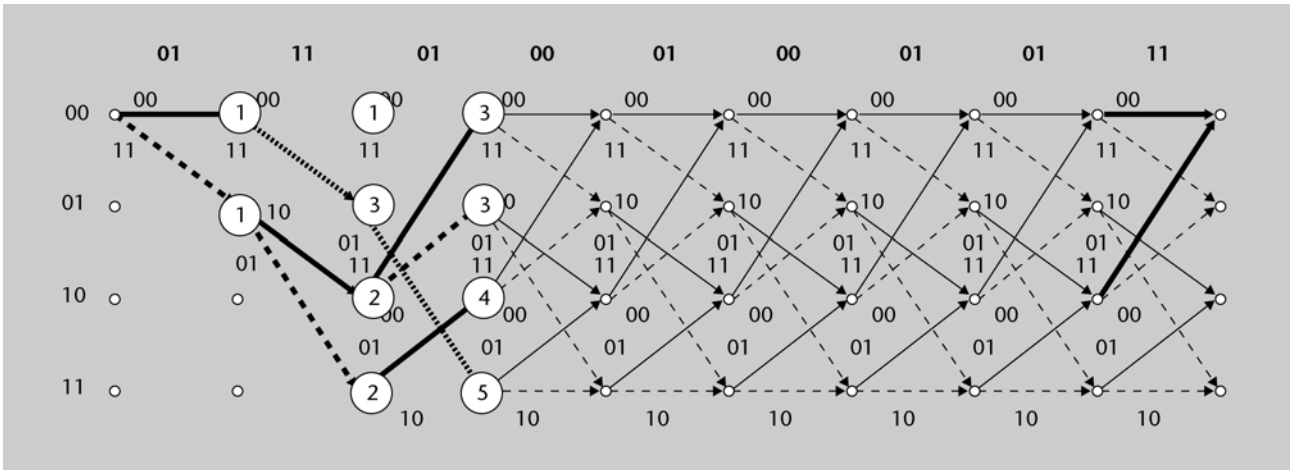

Figura 11. Eliminem les branques que no condueixen a cap node de destinació de la tercera etapa.

La idea bàsica de l'esquema de Viterbi és que, per a un estat en l'etapa n al qual es pot arribar des de dos (en un cas genèric serien 2^k) estats de l'etapa anterior (n – 1), hem de considerar únicament el camí que presenta la mètrica més alta, és a dir, el nombre més gran de coincidències. La demostració que aquest procediment és òptim no és evident, però es poden trobar els detalls en un article clàssic (Viterbi, 1967) i en alguns textos avançats de comunicacions.

És important observar que les mètriques o coincidències calculades en l'etapa $n-1$ es poden utilitzar per a calcular les coincidències de la branca que estem avaluant amb les coincidències que teníem en el node del qual partim. Així doncs, per a cada node de l'etapa n avaluarem el seu cost (nombre de coincidències) partint de cadascun dels nodes de l'etapa anterior que hi arriben, quedant-nos amb la branca que presenta més coincidències i eliminant la resta de branques. Podria ocórrer que el valor màxim s'assolís per a dues branques diferents, ja que el nombre de coincidències pels dos camins és el mateix. En aquest cas, l'algorisme de Viterbi permet triar arbitràriament qualsevol de les dues branques. En efecte, és possible demostrar que no hi ha manera de solucionar aquest tipus d'ambigüetats, fet pel qual les dues solucions són equivalents (igual de dolentes). En les figures 12, 13, 14 i 15 es mostren les etapes successives que es van obtenint en aplicar l'algorisme en les etapes 4, 5, 6 i 7 respectivament.

Activitat

Es proposa com a exercici que es verifiquin els detalls de com es van obtenint els estats en les diverses etapes.

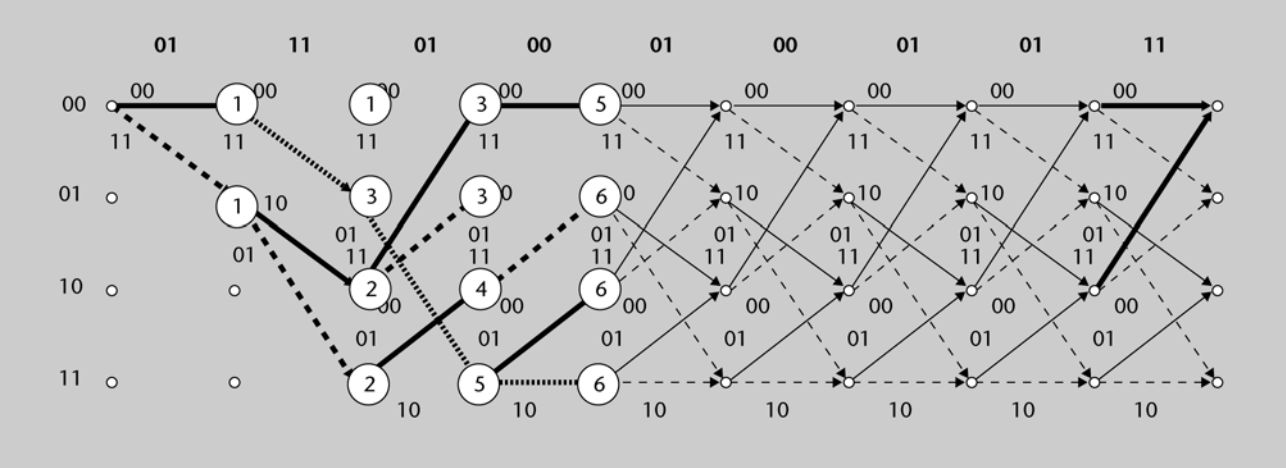

Figura 12. Estat de la descodificació en assolir la quarta etapa

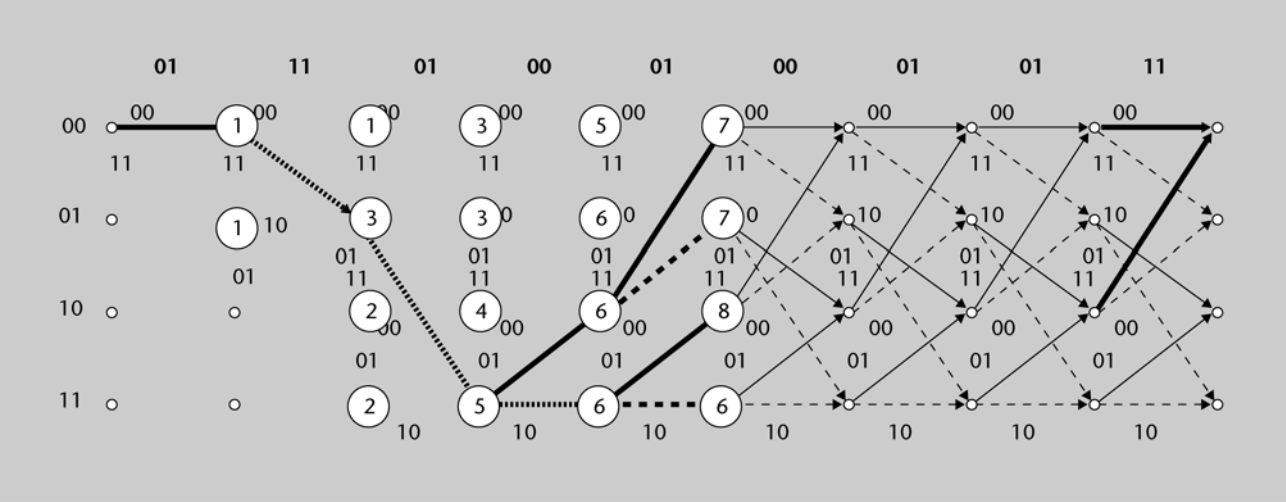

Figura 13. Estat de la descodificació en assolir la cinquena etapa

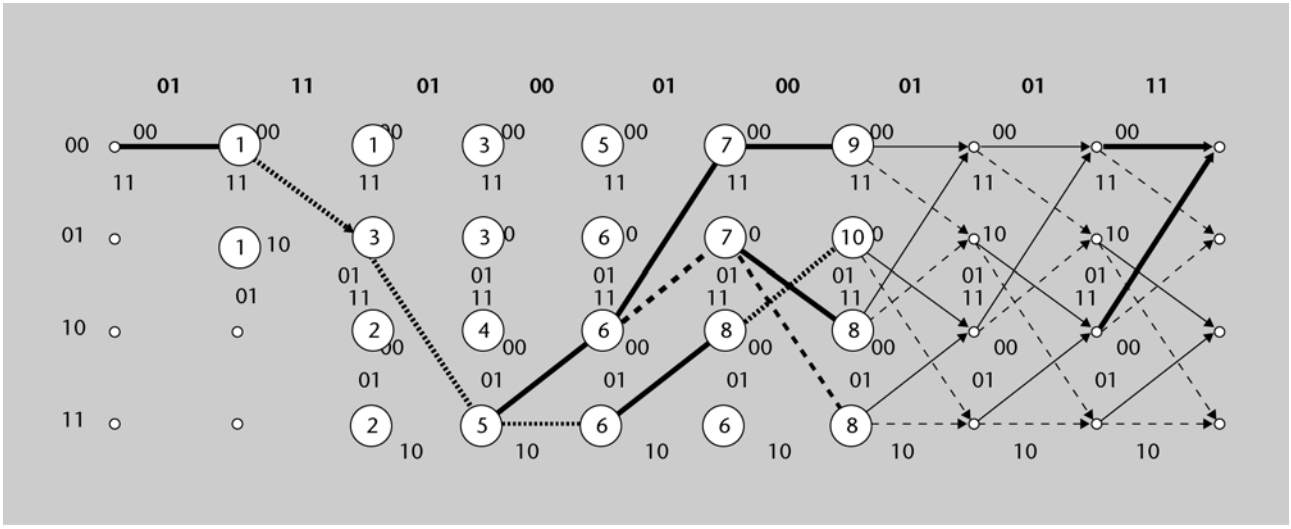

Figura 14. Estat de la descodificació en assolir la sisena etapa

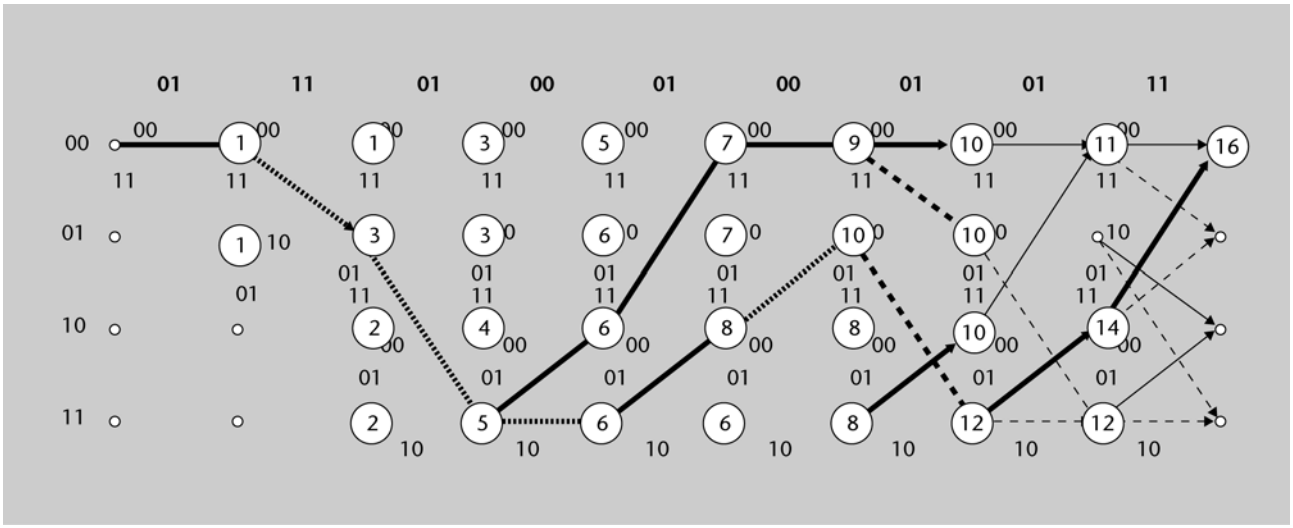

Figura 15. Estat de la descodificació en assolir la setena etapa

A la figura 15 es mostra la resolució completa de l'algorisme de Viterbi sobre el diagrama de Trellis. El resultat final obtingut per l'algorisme consisteix a extreure el camí que originalment havia utilitzat el codificador. Per a recuperar la seqüència de bits d'informació original, hem de començar a recórrer el diagrama de Trellis seguint el conveni utilitzat per a la codificació (un zero es representa amb una transició comtínua, mentre que els uns es representen amb línies discontínues). Així, la seqüència de bits originals que obtenim és {0 1 1 1 0 1 1 0 0} que, com podem comprovar, coincideix amb la de la figura 8. Anàlogament, podem obtenir la correcció de la seqüència rebuda només seguint en el diagrama de Trellis les sortides que s'obtenen en cada transició. Finalment, observem que el node final sempre queda retolat amb el nombre de bits de la seqüència rebuda que han coincidit amb el nombre de bits que hem descodificat. En el nostre exemple el rètol que apareix en el node final és 16, i açò significa que s'han produït i corregit 2 errors en la seqüència original de 18 bits i que els 16 bits restants eren correctes.

L'algorisme de Viterbi obté l'estimació de màxima versemblança (ML, *maximum likelihood*) de la seqüència transmesa. La seva complexitat és proporcional al nombre d'estats que existeixen en el diagrama de Trellis. Això significa que la complexitat de l'algorisme augmenta de manera exponencial amb el nombre de mostres d'entrada (unitats de memòria) que utilitza el codi convolucional per a determinar les sortides (el producte $L \cdot k$ de la figura 3). En la pràctica, la complexitat de l'algorisme de Viterbi pot ser assumida solament quan el nombre d'unitats de memòria del codificador no és excessiu. Quan la memòria del codificador és considerable, caldrà valorar alternatives a l'algorisme de Viterbi que produeixen descodificacions no optimitzades. Existeixen moltes variants d'aquests descodificadors alternatius, entre les quals hem de destacar els algorismes de Fano (1963), Jelinek (1969) i Heller (1975).

En l'algorisme de Viterbi que hem vist en aquest apartat s'ha utilitzat una descodificació dura (*hard decision*), en la qual el receptor decideix el valor lògic de cadascun dels bits rebuts abans d'iniciar el procés de descodificació de la seqüència rebuda. No obstant això, l'algorisme es pot generalitzar de manera bastant directa per a realitzar també una descodificació tova (*soft)*. Hi ha diverses estratègies per a generalitzar aquest algorisme per a la descodificació utilitzant decisions toves (*soft decision*).

L'alternativa que es fa servir més consisteix a modificar la mètrica de decisió i utilitzar el valor absolut de les diferències entre el senyal rebut i el que teòricament hauríem d'haver rebut en absència de soroll en el canal. En aquest cas, la millor branca és la que tingui una mètrica menor, motiu pel qual haurem de descartar sempre aquells camins que presenten el rètol amb valors més grans. També es pot utilitzar l'algorisme amb les diferències elevades al quadrat en lloc del valor absolut, o bé definir una nova mètrica que permeti mesurar el grau de coincidència entre la seqüència que idealment hauríem d'haver rebut i la que realment s'ha rebut.

Per a il·lustrar el mecanisme mitjançant el qual es por utilitzar l'algorisme de Viterbi amb una descodificació tova, se n'inclou un exemple en l'apartat d'activitats, anàleg al realitzat en aquest apartat, en el qual es fan servir les diferències en valor absolut entre els valors ideals i els rebuts.

1.5. Elecció dels codis convolucionals

Els codis convolucionals combinen mitjançant operacions de sumes exclusives els últims bits de la seqüència d'informació, és a dir, els que estan en la memòria del codificador. El nom d'aquests codis es deu a l'operació de convolució que s'utilitza per a expressar la sortida d'un sistema linial i invariant en funció de l'entrada i la resposta impulsional. En el nostre cas, la convolució es realitza entre la seqüència de bits d'informació i un polinomi que explicita com s'han de realitzar les operacions de sumes exclusives entre els bits actuals i els bits emmagatzemats en la memòria del codificador. El polinomi que defineix el codificador és, per tant, anàleg a la resposta impulsional que defineix el filtre lineal.

Tot i que sembla que es pot establir un cert paral·lelisme entre els sistemes lineals i els codis convolucionals, en la pràctica les tècniques d'anàlisi i disseny són molt diferents. De fet, hi ha moltes tècniques matemàtiques per al disseny de filtres que ens permeten determinar els coeficients dels filtres en funció de les característiques que volem que tingui el nostre sistema lineal. En canvi, per al disseny de codis convolucionals no existeixen aquestes tècniques sistemàtiques en el disseny dels coeficients del polinomi. L'elecció d'un polinomi o d'un altre es redueix generalment a la realització de simulacions i proves amb diversos polinomis i a la confecció de taules on s'inclouen aquells polinomis que s'han comprovat que tenen "bones propietats". A més, el concepte de bones propietats tampoc no està clarament definit, ja que depèn de l'aplicació i de les característiques que volem que tingui el codi.

En general, amb un codi convolucional busquem que una seqüència de bits d'entrada (generalment llarga) es transformi en una paraula codi per a transmetre-la a través del canal. El propòsit és que aquesta paraula codi estigui més protegida enfront del soroll del canal i pugui ser identificada de manera més eficient pel receptor. Segons aquest principi, esperem que dues seqüències d'informació que siguin molt diferents produeixin també paraules codi que siguin molt diferents. Si això no fos així, el codi tindria poca utilitat pràctica.

1.5.1. Codis catastròfics

Posem un exemple de codis convolucionals inadequats: els codis catastròfics, que presenten les característiques indesitjables descrites en els paràgrafs anteriors. Formalment, un codi es denomina *catastròfic* quan per a dues seqüències d'entrada que difereixen en un nombre infinit de posicions, es produeixen dues seqüències de sortida que solament difereixen en un nombre finit de posicions. Segons aquesta definició, és possible que, si es produeix un nombre finit d'errors en la recepció d'una seqüència, el descodificador produeixi un nombre infinit d'errors en la seqüència descodificada. Sens dubte, aquesta definició pot semblar una mica abstracta, per la qual cosa és millor utilitzar un exemple de codi catastròfic per a entendre més clarament quins són els seus efectes.

Considerem com a exemple el codi següent, la realització i el diagrama d'estats del qual es representa a la figura 16.

- Nombre de línies d'entrada: $k = 1$
- Nombre de línies de sortida: $n = 2$
- Taxa del codi: $R = 1/2$
- Unitats de memòria: 2

- Nombre de possibles estats: $2^2 = 4$
- Longitud total de la convolució: 3
- Polinomi generador 1: $G_1 = X^2 + X = 6_{OCT}$
- Polinomi generador 2: $G_2 = X + 1 = 3_{OCT}$

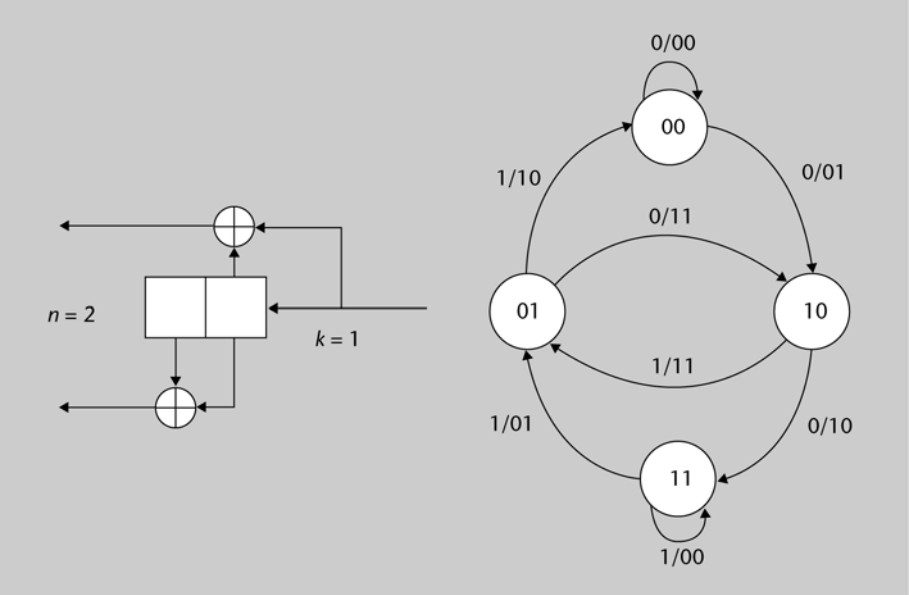

Figura 16. Exemple d'un codi convolucional catastròfic.

Per a comprovar que el codi és catastròfic, suposem que la seqüència d'informació és una seqüència amb tots els símbols igual a la unitat {1 1 1 1 1 .... 1 1 1}. La seqüència que obtenim en la sortida del codificador es pot calcular molt fàcilment a partir del diagrama d'estats:

{10 01 00 00 00 00 00 00 … 00 00 10 01}

D'altra banda, si la seqüència d'entrada són tot zeros { 0 0 0 0 0 0 ..... 0 0 }, la sortida que obtenim és:

{ 00 00 00 00 00 00 .... 00 00 00}

Aquest resultat ens indica que tot i que les seqüències d'entrada difereixen en tots els bits (infinits, si les seqüències tenen una longitud infinita), les seqüències de sortida solament difereixen en un nombre finit de bits, en concret, 4 bits per al nostre exemple.

Els codis catastròfics es poden detectar de manera més o menys immediata a partir del seu diagrama d'estats. La manera més senzilla de detectar-los és observar que un codi és catastròfic quan té algun estat en què les seves sortides són sempre zero quan la seqüència d'entrada té bits diferents de zero. En el nostre exemple, el problema està en l'estat {11}, ja que tot i que la seqüència d'entrada sigui diferent de zero {1 1 1 1 1….}, pot produir una sortida en la qual tot són zeros.

1.5.2. Distància lliure d'un codi

Una característica que se sol utilitzar per a indicar que un codi té bones prestacions és la distància lliure del codi (d_{free}). La distància lliure es defineix com el nombre d'uns que té la seqüència de sortida més curta tal que partint de l'estat {00} torna a l'estat {00}. En l'exemple del codi catastròfic anterior, la distància lliure del codi és 4. En efecte, el camí més curt que surt i retorna a l'estat {00} passa pels estats {00} => {01} => {10} => {00}, i les sortides que obtenim són {10 11 01}. La distància lliure es pot interpretar com un equivalent a la paraula codi de pes mínim que s'utilitza en els codis de bloc. En aquest cas, la sortida que hem obtingut representa la seqüència de bits d'entrada més pròxima a la paraula tot zeros. És important notar que una distància lliure gran no ens garanteix que el codi tingui bones prestacions. És fàcil imaginar un diagrama d'estats en el qual la distància lliure és molt gran i, en canvi, hi ha estats que provoquen que el codi resultant sigui catastròfic.

En la pràctica hi ha taules en les quals es proporcionen polinomis generadors que se saben a partir de simulacions que tenen bones propietats de distància. Aquestes taules depenen de les taxes del codi. En la taula següent es proporcionen els polinomis generadors (en notació octal) per a codis de taxa 1/2.

Taxa codi = 1/2; Codis amb d_{free} màxima			
L Long. conv.	**Polinomis generadors octal**		**d_{free}**
3	5	7	5
4	15	17	6
5	23	35	7
6	53	75	8
7	133	171	10
8	247	371	10
9	561	753	12
10	1167	1545	12
11	2335	3661	14
12	4335	5723	15
13	10533	17661	16
14	21675	27123	16

Taula 2. Taula de codis convolucionals de taxa 1/2 amb distàncies lliures màximes.

1.5.3. Longitud de convolució

Un paràmetre molt important és la longitud de la convolució que afecta les capacitats correctores del codi. Així, per exemple, per a un sistema de comunicacions que treballi amb una relció E_0/N_0 de 5 dB, amb modulacions QPSK,

podem obtenir una probabilitat d'error típica (BER, *bit rate error*) de $5 \cdot 10^{-3}$, quan no utilitzem cap codi convolucional. Si mantenim la relació $E_0/N_{0,}$ però s'utilitza un codi convolucional amb una taxa de codi $R = 1/2$ i una longitud de convolució de 2, obtenim una reducció en la BER (després del codi) d'aproximadament $3 \cdot 10^{-5}$, cosa que ens dóna una idea de la millora introduïda pel codi convolucional. Si la longitud de convolució augmenta fins a 7, la taxa d'error es redueix fins a $5 \cdot 10^{-7}$.

Per posar-ne un exemple concret, els sistemes de difusió de televisió digital (DVB, *digital video broadcasting*) utilitzen codis convolucionals de taxa $R = 1/2$ i longitud de convolució de 7. Així, treballant amb una relació E_0/N_0 típica de 3,2 dB s'aconsegueix obtenir una taxa d'error de $2 \cdot 10^{-4}$. Aquesta taxa d'error pot semblar excessiva per a un sistema de comunicacions real, però hem de tenir en compte que en DVB s'utilitzen dos codis concatenats, que complementen la correcció d'errors. En aquest exemple, a més, juntament amb el codi convolucional s'utilitza un codi de Reed-Solomon que reduirà la taxa d'error fins a aproximadament 10^{-11}.

1.6. *Puncturing*

Un dels desavantatges dels codis convolucionals és que les seves taxes de codi són molt baixes. En efecte, el codificador que hem estat considerant en el nostre exemple té una taxa $R = 1/2$, que significa que un 50% dels bits en la seqüència codificada corresponen a redundància.

Una tècnica directa per a millorar la taxa del codi consisteix a realitzar la perforació (*puncturing*) de la seqüència codificada resultant. Aquesta tècnica consisteix a deixar de transmetre sistemàticament alguns bits de la seqüència resultant. Així, per exemple, podríem deixar de transmetre un de cada tres bits, amb la qual cosa un codi que tingués una taxa $R = 1/2$ passaria a tenir una taxa $R = 1/2 \times 3/2 = 3/4$. Malgrat que les capacitats correctores del codi es veuen alterades en realitzar la perforació, amb alguns petits canvis en el procediment, podem seguir utilitzant l'algorisme de descodificació de Viterbi.

En efecte, a la figura 17 es mostra un exemple d'una seqüència d'informació que es codifica amb un codi de taxa $R = 1/2$. Cada bit de la informació original dóna lloc a dos bits de la seqüència codificada. Abans de realitzar la transmissió, el codi es perfora (*puncture*) amb una taxa de 3/2, és a dir, de cada 3 bits de la seqüència de dades solament transmetem els dos primers. Si suposem que la seqüència es rep sense errors, el descodificador pot inserir bits desconeguts (indicats amb X) en aquelles posicions que sap que el codificador no ha transmès. Per a realitzar el procés de descodificació amb l'algorisme de Viterbi, les X són ara valors desconeguts en el senyal rebut que no s'utilitzen per a in-

crementar o decrementar les coincidències entre la seqüència descodificada i la seqüència rebuda.

Dades originals	1	0	1	1	0	0	0
Cod. conv. (R = 1/2)	11	10	00	01	01	11	00
Puncturing (3/2)	11	$_1$0	0$_0$	01	$_0$1	1$_1$	00
Seqüència transmesa R=3/4		11	00	01	11	00	
Dades descodificador (algorisme de Viterbi)	11	X0	0X	01	X1	1X	00

Figura 17. Procés de codificació, *puncturing* i obtenció de la seqüència d'entrada a l'algorisme de Viterbi

2. Concatenació de codis

La concatenació de codis consisteix a aplicar un codi de protecció d'errors a una seqüència de bits a la qual ja s'ha aplicat un altre codi de protecció. El concepte pot semblar molt trivial i suggereix que possiblement es tracta d'una simplificació per a no utilitzar codis de protecció més complexos. No obstant això, veurem que amb la concatenació de codis es poden obtenir característiques molt específiques, que no es poden obtenir amb un únic codi.

A la figura 18 es mostra un exemple en el qual es concatenen dos codis de protecció d'errors. En l'emissor, la seqüència de bits originals es passa a través d'un primer codi anomenat *codi extern*. El resultat d'aquesta primera codificació es passa a través d'un altre descodificador que denominem *codi intern*. Les dades obtingudes després de la concatenació dels dos codis s'envien al canal. En el receptor, la seqüència es descodifica en el sentit invers, és a dir, primer s'aplica el descodificador intern i posteriorment l'extern. La taxa del codi resultant total és el producte de les taxes dels dos codis. És a dir, si la taxa de bit en l'entrada del primer codi és R_0, la taxa de bit en la sortida serà R_0/R_1, on R_1 és la taxa del codi extern. Aquesta és la taxa de bits que entra al segon codificador i s'obté una taxa de bit final R_0/R_1R_2, per la qual cosa es pot considerar que la taxa del codi concatenat és el producte de les taxes individuals de cada codi.

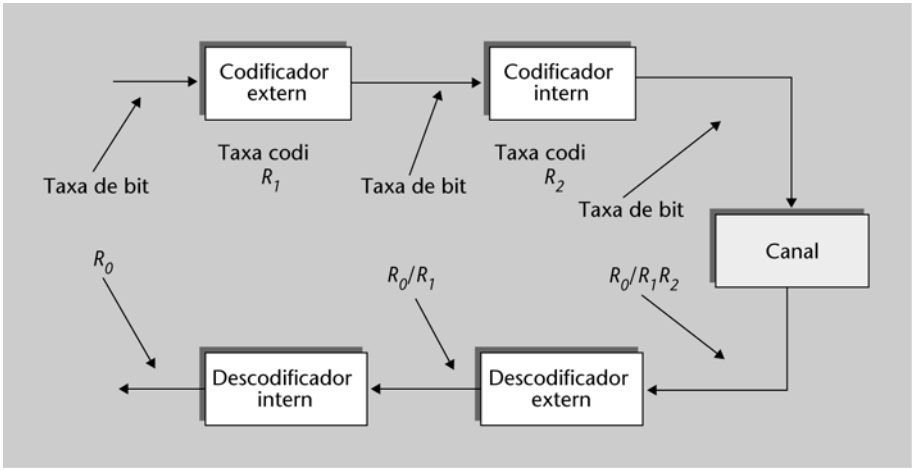

Figura 18. Diagrama de blocs de la concatenació de dos codis

2.1. Avantatges de la concatenació de codis

Per a veure clarament l'avantatge de la concatenació de dos codis, en posarem un exemple concret en què s'utilitza un codi de Reed-Solomon (codi RS)

com a codi extern i un codi de Hamming com a codi intern. Com es mostra a la figura 19, el codi RS pren un conjunt de *K* símbols i hi afegeix *R* símbols de redundància. Suposem que pot arribar a corregir fins a un total de C símbols, on cada símbol està format per 8 bits. El codi intern, de Hamming, pren un paquet de 4 bits (la meitat d'un símbol) i hi afegeix 3 bits de redundància, i així pot arribar a corregir 1 bit per cada paquet.

La concatenació dels dos codis presenta els avantatges de cadascun dels codis individuals. En efecte, si en el canal es produeixen molts errors aïllats (d'un bit per cada paquet), aquests poden ser corregits pel codi intern (de Hamming), motiu pel qual la informació es rep sense errors en el codi extern. En canvi, si es produeixen ràfegues d'errors, que afecten diversos bits consecutius, no podran ser corregits pel codi intern però, en aquest cas, entrarà en acció el codi RS. En conseqüència, gràcies a la concatenació dels dos codis podem arribar a obtenir un istema amb capacitat per a corregir errors aïllats i ràfegues d'errors.

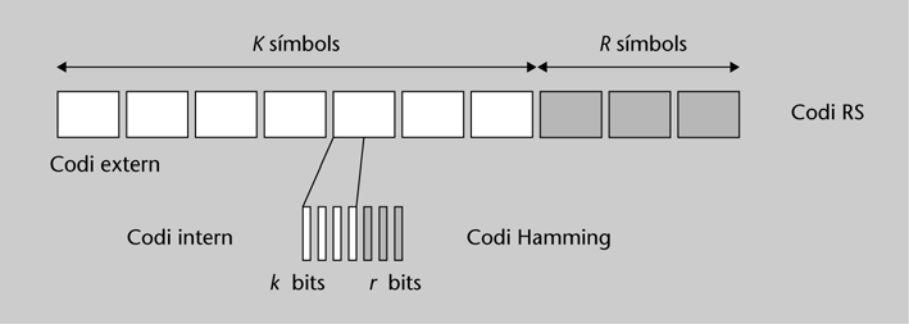

Figura 19. Concatenació d'un codi de Reed-Solomon amb un codi de Hamming.

2.2. Intercalador

La concatenació de codis estudiada en l'apartat anterior ens permet corregir gran quantitat d'errors de bits aïllats, gràcies al codi de Hamming, i ràfegues d'errors que involucren fins a un total de *C* símbols en la trama RS. El nombre de símbols que es poden corregir depèn de les característiques del codi RS. Si una ràfega d'errors afecta més de *C* símbols, no es podrà realitzar la correcció de manera adequada.

Una possible estratègia per a millorar el comportament dels codis concatenats és afegir un intercalador entre els dos codis. Així, aconseguim protecció de ràfegues d'errors que afecten més de *C* símbols. A la figura 20 es representa un esquema bàsic d'un codi concatenat que conté un intercalador. L'intercalador consisteix a alterar les posicions, l'ordre en què es transmeten els bits, de manera que aquells bits que en la trama original apareixen consecutius quedin desordenats en la trama transmesa. En el receptor el desintercalador s'encarrega de restaurar l'ordre original dels bits. L'avantatge de desordenar els bits

(o símbols) abans de transmetre'ls permet dispersar els errors de ràfega, com s'il·lustra a la figura 21.

Figura 20. Ús d'un intercalador en codis concatenats

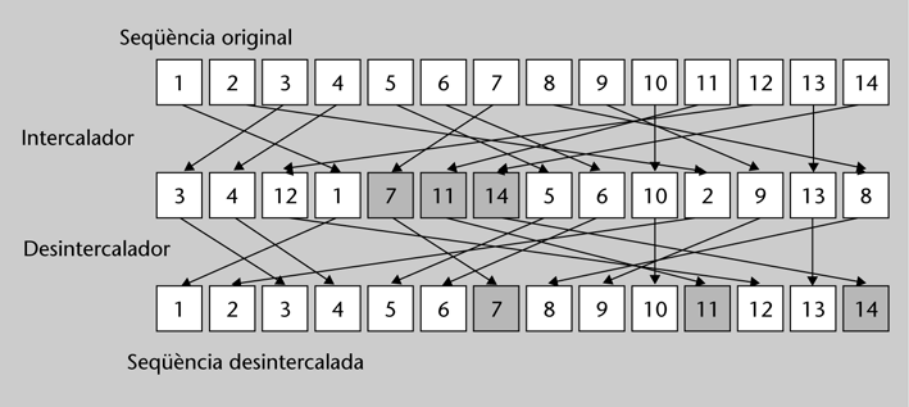

Figura 21. Dispersió dels errors de ràfega mitjançant l'ús d'un intercalador i un desintercalador

Hi ha dues alternatives per a realitzar un intercalador:

- l'intercalador de blocs (*block interleaver*),
- l'intercalador convolucional (*convolutional interleaver*).

El principi de funcionament de l'intercalador de blocs és molt simple. Es tracta d'una memòria organitzada en forma de matriu (*array*) en què la seqüència de símbols d'entrada s'escriu fila a fila, com s'indica a la figura 22. Un cop tot el bloc de memòria ha estat escrit es llegeix i es transmeten els símbols en el sentit de les columnes. En el receptor, els símbols rebuts s'escriuen en la memòria columna a columna. Per tant, si s'ha produït una ràfega d'errors durant la transmissió de les dades, tots els símbols erronis estaran en la mateixa columna (o en columnes adjacents). La lectura de la memòria es realitza ara per files, de manera que la ràfega d'errors que afectava diversos símbols ara solament afecta un nombre reduït de símbols, fet pel qual podran ser corregits per codis no excessivament complexos.

Les tècniques d'IL de bloc generen, inevitablement, un retard en la desmodulació dels bits que depèn de les dimensions de la matriu d'entrellaçament. Aproximadament, el retard és del temps de N bits en el modulador més el temps de N bits en el desmodulador, de manera que s'acumula un retard total de $2N$ bits. A fi de disminuir el retard per entrellaçament, s'utilitzen les tècniques d'IL convolucionals, mitjançant les quals el retard

màxim a causa de l'entrellaçament queda dividit per dos i és sempre més petit que un retard de N bits.

Els intercaladors convolucionals es basen a passar els símbols d'entrada a través d'un sistema format per L unitats de retard, cadascuna de les quals amb retards diferents, i anar commutant l'entrada i la sortida dels símbols, de manera sincronitzada, com s'indica a la figura 23. L'avantatge principal d'aquesta tecnologia d'intercalació és que la sincronització dels blocs en el receptor és més simple.

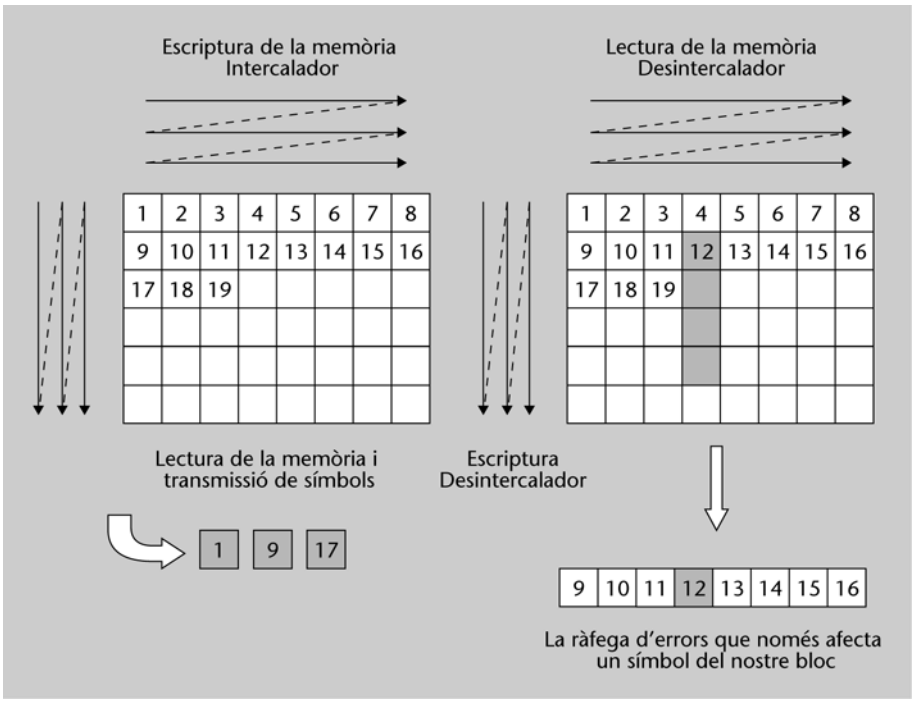

Figura 22. Esquema de funcionament d'un intercalador i un desintercalador de bloc

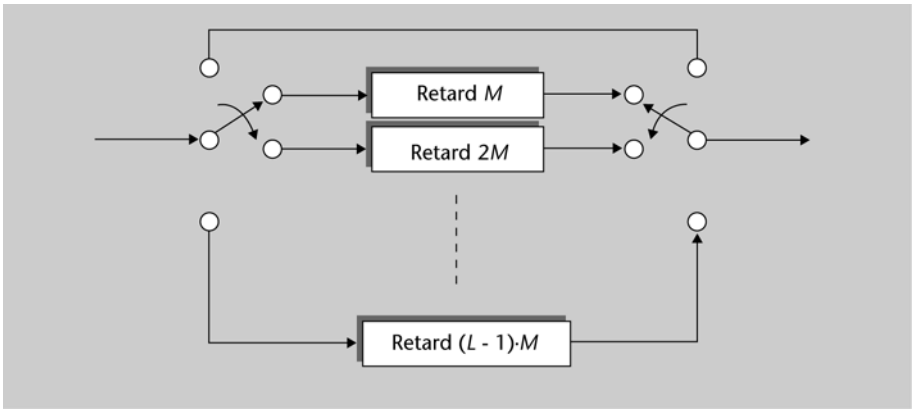

Figura 23. Diagrama d'un intercalador convolucional

A continuació, s'explica la manera de processar els bits en un IL convolucional.

1) En l'operació d'IL. A partir de la seqüència de bits, es processa seqüencialment una trama de K bits, de manera que K és un múltiple enter d'un paràmetre B, també enter.

- El primer bit de la trama no es retarda.
- El segon bit de la trama pateix un retard igual a K períodes de bit.
- El tercer bit de la trama pateix un retard igual a $2K$ períodes de bit.
- ...
- El B-èsim bit de la trama pateix un retard de $(B-1)K$ períodes de bit.
- El $B+1$-èsim bit de la trama no es retarda.
- El $B+2$-èsim bit de la trama pateix un retard igual a K períodes de bit.
- Etc.

2) En l'operació de desentrellaçament (DeIL) del desmodulador es complementen els retards. A partir de la seqüència de bits, es processa seqüencialment una trama de K bits.

- El primer bit de la trama pateix un retard de $(B-1)K$ períodes de bit.
- El segon bit de la trama pateix un retard igual a $(B-2)K$ períodes de bit.
- El tercer bit de la trama pateix un retard igual a $(B-3)K$ períodes de bit.
- ...
- El B-èsim bit de la trama no es retarda.
- El $B+1$-èsim bit de la trama pateix un retard de $(B-1)K$ períodes de bit.
- El $B+2$-èsim bit de la trama pateix un retard igual a $(B-2)K$ períodes de bit.
- Etc.

Cada bit en particular, sumant el retard que ha patit en el modulador amb el retard que ha patit en el desmodulador, ha acumulat un retard de $(B-1)K$ bits.

2.3. Concatenació de codis en DVB

Un exemple típic de concatenació de codis són els sistemes de radiodifusió de senyals de televisió digitals, tant en la seva modalitat de televisió digital per satèl·lit com la de televisió digital terrestre. Aquests dos sistemes estan regulats pels estàndards ETS 421 (satèl·lit) i ETS 744 (terrestre), i l'arquitectura dels components que intervenen en la codificació de canal és idèntica en els dos casos. La normativa de codificació de canal i transmissió d'aquests sistemes s'estableix en el fòrum internacional DVB (*digital video broadcasting*). La codificació de canal en els sistemes terrestre i satèl·lit consisteix en la concatenació d'un codi de Reed-Solomon (extern) amb un codi convolucional (intern) i un intercalador també convolucional.

El codi de Reed-Solomon utilitza símbols de 8 bits. Està dissenyat per a corregir un total de 8 símbols en cada paquet de dades utilitzant un codi amb 239 símbols d'informació i 16 símbols de redundància. Així doncs, el codi RS de partida és un codi (255,239).

No obstant això, la trama de transport de l'MPEG-2 utilitza paquets de dades de 188 bytes. Això condiciona que els blocs d'informació del codi RS s'hagin

de prendre en mòduls de 188 bytes. La solució consisteix a utilitzar un codi (255,239) en el qual, dels 239 bytes d'in formació, els 51 primers es posen a zero, i tot i que no cal enviar-los, s'utilitzen per a calcular els 16 símbols de redundància que s'afegeixen. El codi RS resultant és una versió retallada del codi (255,239) que es denota com a (204,188).

L'intercalador utilizat en el DVB té una arquitectura convolucional i utilitza un total de 12 unitats de retard (L) amb un retard de cadascuna que és un múltiple enter de 17 símbols (M) (vegeu la figura 23).

El codi convolucional té una taxa $R = 1/2$ amb una longitud de la convolució de 7 bits. Els polinomis generadors vénen donats per $G_1 = 171_{OCT}$ i $G_2 = 133_{OCT}$. Opcionalment, podem aplicar *puncturing* al codi convolucional resultant triant unes taxes de codi de 2/3, 3/4, 5/6 o 7/8.

La codificació de televisió digital per cable és més simple ja que la transmissió per cable està més protegida d'errors i interferències. En aquest cas, no s'utilitza el codi convolucional intern.

3. Turbocodis

Els turbocodis es poden considerar un cas especial de codis concatenats, amb certes peculiaritats que els fan molt efectius quant a la seva capacitat de correcció d'errors i la simplicitat en la implementació del codificador. Tot i això, en contra seu està la complexitat computacional dels algorismes de descodificació. Per aquestes característiques, són especialment utilitzats en aquelles aplicacions en què el transmissor té una complexitat computacional limitada i un cost reduït, mentre que el receptor pot ser complex. Un escenari d'aplicació típic són els equips o terminals de telefonia mòbil, generalment amb una capacitat de processament numèric reduïda però que es comuniquen amb una estació central en la qual es poden realitzar processos de càlcul complexos.

Hi ha diversos tipus de configuracions l'estudi detallat de les quals escapa als propòsits d'aquest text. La configuració més utilitzada és la representada a la figura 24, que suposarem que treballa amb paquets d'informació de N bits. Hi ha dos codificadors convolucionals amb taxes de codi $R = 1/2$ que transmeten en paral·lel i estan connectats a través d'un intercalador. Els codis RSCC que s'utilitzen són codis convolucionals sistemàtics i recursius (RSCC, *recursive systematic convolutional code*). La propietat que siguin sistemàtics és equivalent a la definició que utilitzàvem per als codis de bloc, és a dir, els primers bits de la seqüència codificada coincideixen amb els bits d'informació. La *recursivitat* fa referència a l'estructura del generador del codi. Fins ara, tots els codis convolucionals que hem estudiat són no recursius, de manera que el bit de sortida solament depèn dels bits d'entrada retardats. En els codificadors recursius apareixen realimentacions entre els registres de desplaçament. Es pot veure la realització d'un codi convolucional recursiu a la figura 25. Cal observar que és bastant directe establir una relació entre els codis convolucionals recursius i els filtres de resposta impulsional infinita (IIR, *infinite impulse response*), en què la sortida del filtre es realimenta per a obtenir les successives sortides, de la mateixa manera que els codis no recursius es relacionen amb els filtres de resposta impulsional finita (FIR, *finite impulse response*), en els quals la sortida es calcula a partir de les mostres del senyal d'entrada. Per tant, la recursivitat en la implementació del codi suposa que la longitud de la convolució es pot considerar infinita. Així doncs, amb un processament relativament reduït podem implementar filtres amb una longitud de convolució equivalent molt gran i amb bones prestacions per a la seva protecció d'errors.

En la implementació del turbocodi representat a la figura 24, els N bits d'informació es transmeten directament al canal, de manera que cadascun dels codis convolucionals recursius, amb $R = 1/2$, solament transmet els N bits de

redundància. El nombre total de bits que s'envien al canal és, per tant, de 3*N*, fet pel qual la taxa del codi final és $R = 1/3$. Per a fer aquests turbocodis més eficients, és habitual utilitzar tècniques de *puncturing*.

Una característica dels turbocodis, que afecta directament les seves prestacions, és la longitud equivalent sobre la qual es reaitza l'entrellaçat. En les implementacions més habituals, aquest bloc sol afectar uns 1.000 bits o més.

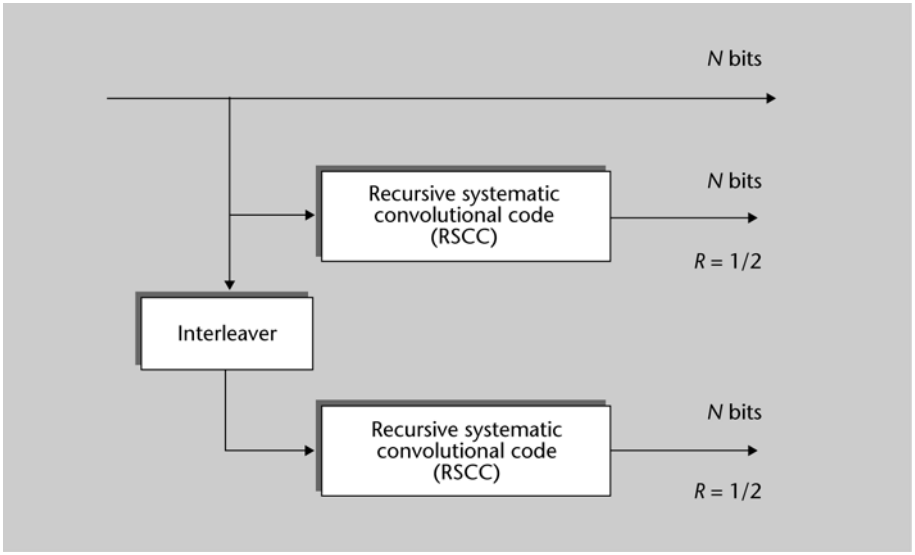

Figura 24. Diagrama de blocs d'una implementació d'un turbocodi

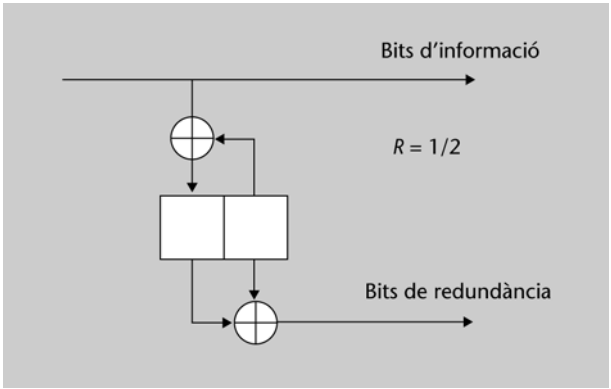

Figura 25. Exemple d'un codi convolucional sistemàtic (transmet els bits d'informació) i recursiu.

Activitats

Descodificació de Viterbi amb decisions *soft*

Considerem el codificador de la figura 4, el diagrama d'estat i diagrama de Trellis del qual estan representats a les figures 5 i 6. Suposem que transmetem la seqüència d'informació que es mostra a la figura 27 a través d'un canal gaussià i sabem que el detector produirà una tensió de valor mitjà 1 volt per al bit 1 i un valor de tensió mitjà de –1 volt per al valor lògic 0. La seqüència rebuda es mostra també a la figura 26. Determinarem la seqüència corregida que obtindrem utilitzant un descodificador de Viterbi que utilitzi una descodificació tova (*soft*).

Informació	0 1 1 1 0 1 1 0 0
Seqüència transmesa	00 11 01 10 01 00 01 01 11
Seqüència detectada	{-0.8,1.2; 0.9, 0.8; -1.1,0.9; -0.6,-1.2; -1, 1.1; -0.8,-0.6 -0.9, 0.9; -1.3,0.7; 1.1,0.9}

Figura 26. Exemple de descodificació *software* mitjançant l'algorisme de Viterbi

Solucionari

El procés de descodificació de la seqüència és semblant al que s'ha realitzat per a la descodificació *hardware*, encara que ara la manera com es comptabilitza la mètrica és diferent. A la figura 27 es mostren els resultats obtinguts fins a assolir la segona etapa del procés de descodificació. Vegem amb detall com s'han retolat els dos nodes de la primera etapa.

El node corresponent a l'estat {00} de la primera etapa s'ha retolat amb un 2,4. Per a arribar a aquest node hauríem d'haver tingut una sortida {00}, que es correspondria amb unes tensions de –1 V, –1 V. Les tensions que hem rebut són (–0,8, 1,2), fet pel qual la diferència en valors absoluts entre el senyal rebut i el que hauríem d'haver rebut és de: abs(–1 – (–0,8)) + abs(1,2 – (–1)) = 2,4. Anàlogament, si realitzem un càlcul semblant per a l'estat {01} de la primera etapa, obtenim una diferència en valor absolut de 2. Per a la segona etapa, en cada node hem de sumar l'error absolut acumulat fins al node anterior amb l'error comès en aquesta transició.

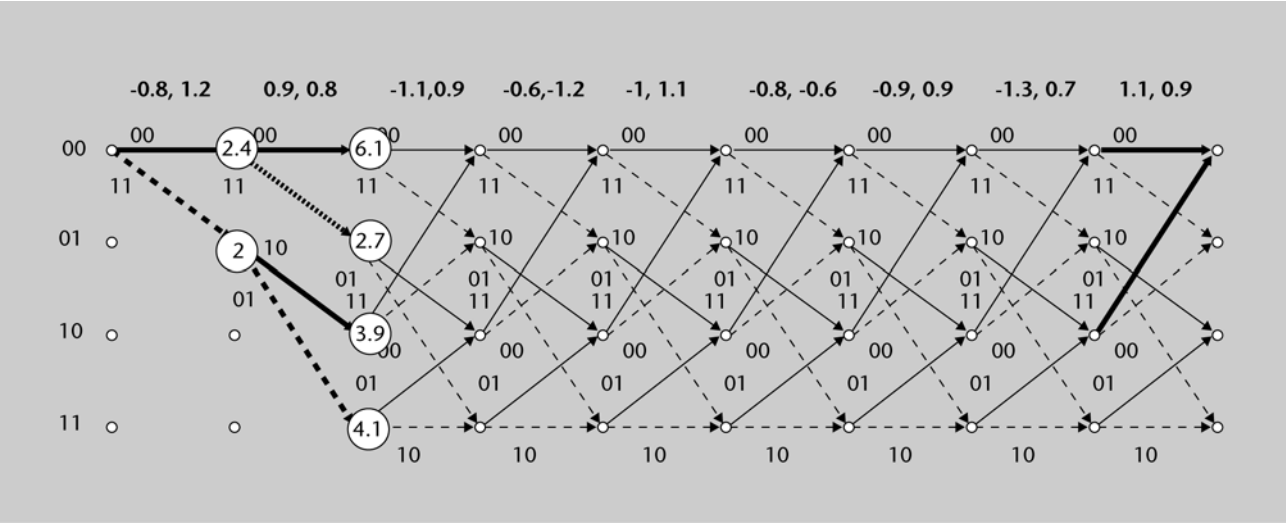

Figura 27. Primeres dues etapes de la descodificació *software* d'una seqüència mitjançant l'algorisme de Viterbi

Per a continuar amb la tercera etapa hem de valorar els dos camins que arriben a cadascun dels node, i ara ens quedarem amb el que tingui el cost menor. Igual que abans, eliminarem tots aquells camins que quedin tallats en un node de l'etapa anterior. Els resultats obtinguts després de la tercera etapa es mostren a la figura 28. En les il·lustracions següents es mostren els resultats obtinguts en les etapes 4, 5, 6, 7 i, finalment, les etapes 8 i 9. Observeu com el camí òptim final coincideix amb el de l'exemple que s'ha analitzat en el text, per la qual cosa els errors s'han corregit correctament.

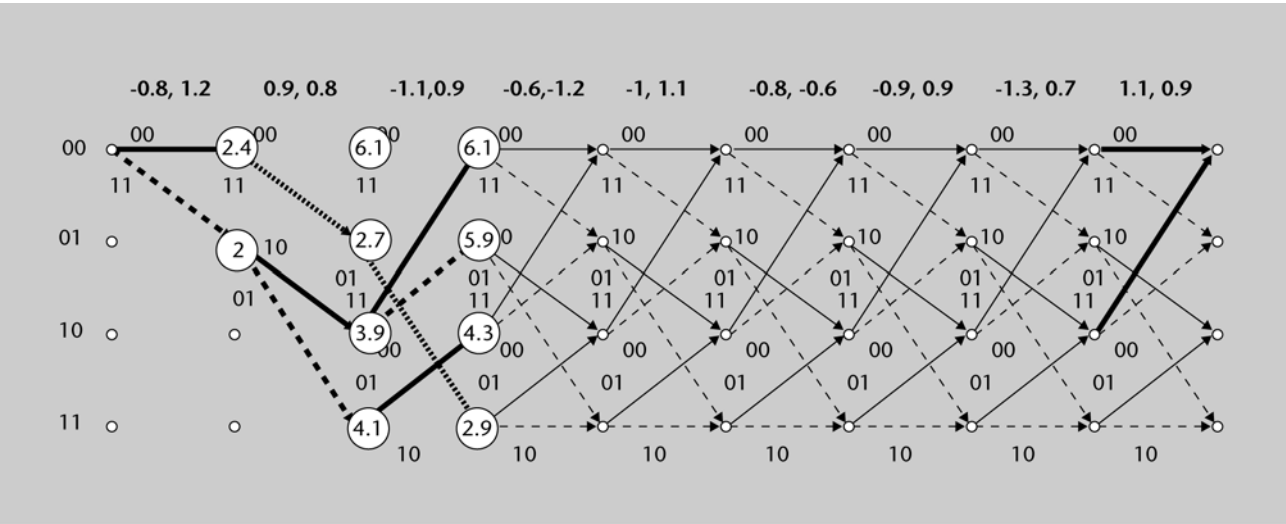

Figura 28. Descodificació *software* en la tercera etapa amb l'algorisme de Viterbi

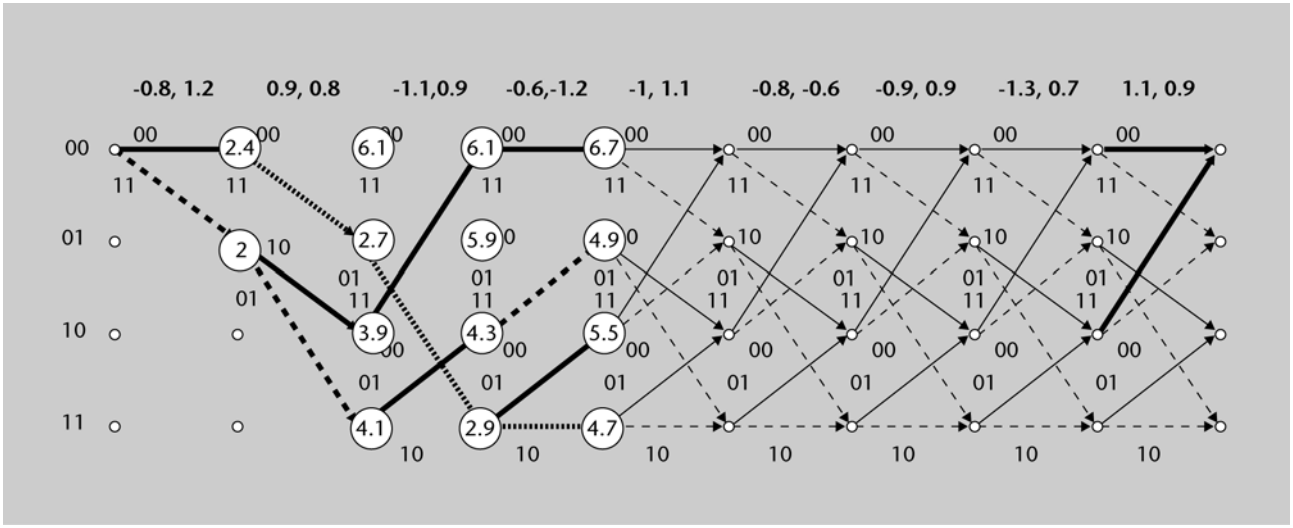

Figura 29. Descodificació *software* en la quarta etapa amb l'algorisme de Viterbi

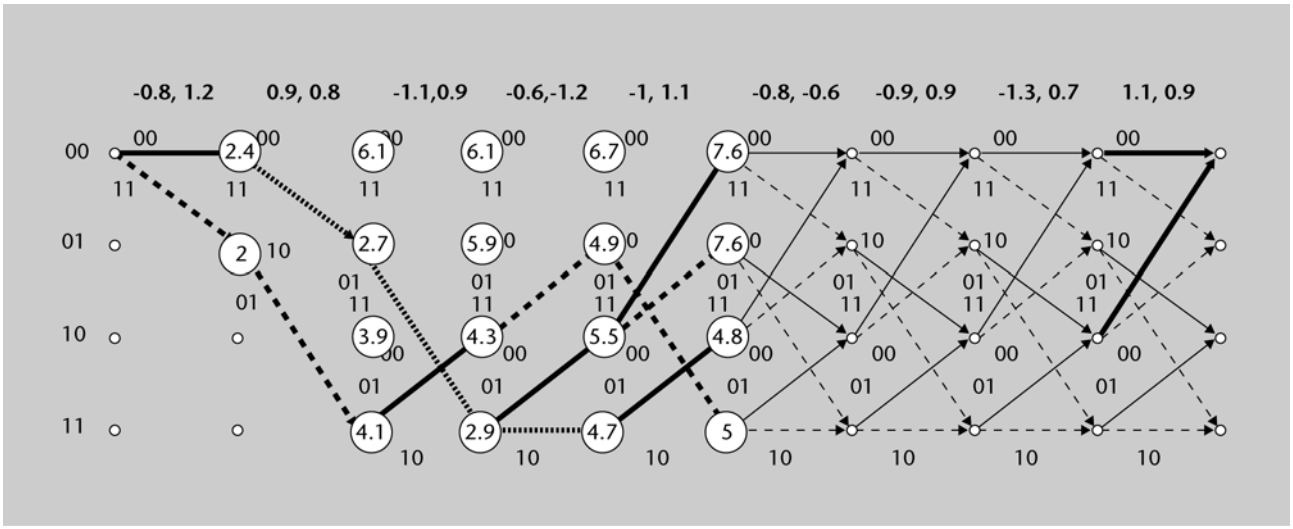

Figura 30. Descodificació *software* en la cinquena etapa amb l'algorisme de Viterbi

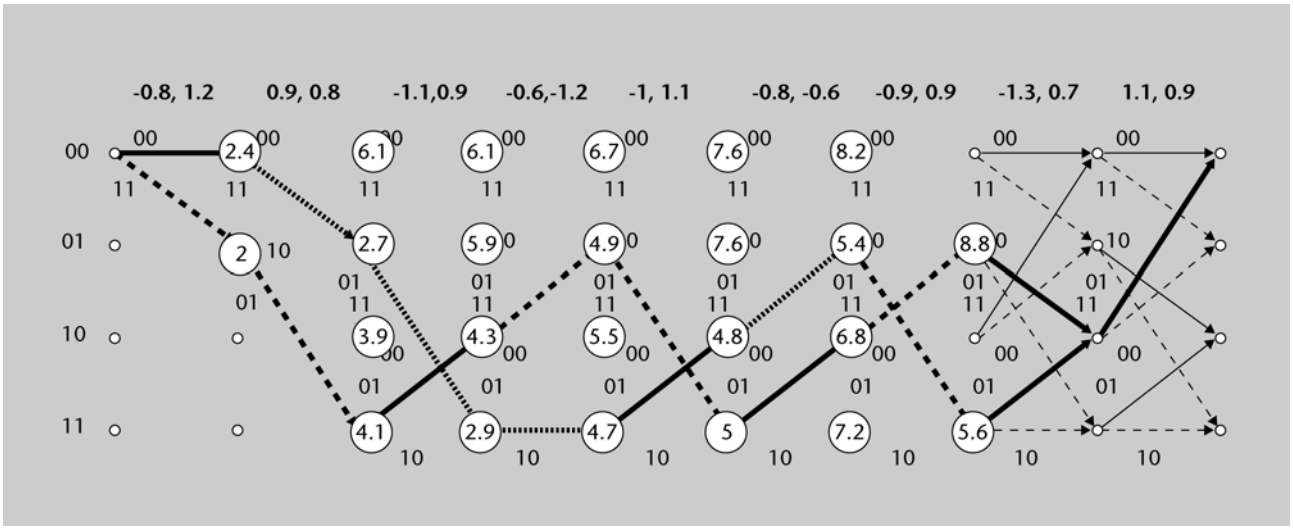

Figura 31. Descodificació *software* en la setena etapa amb l'algorisme de Viterbi

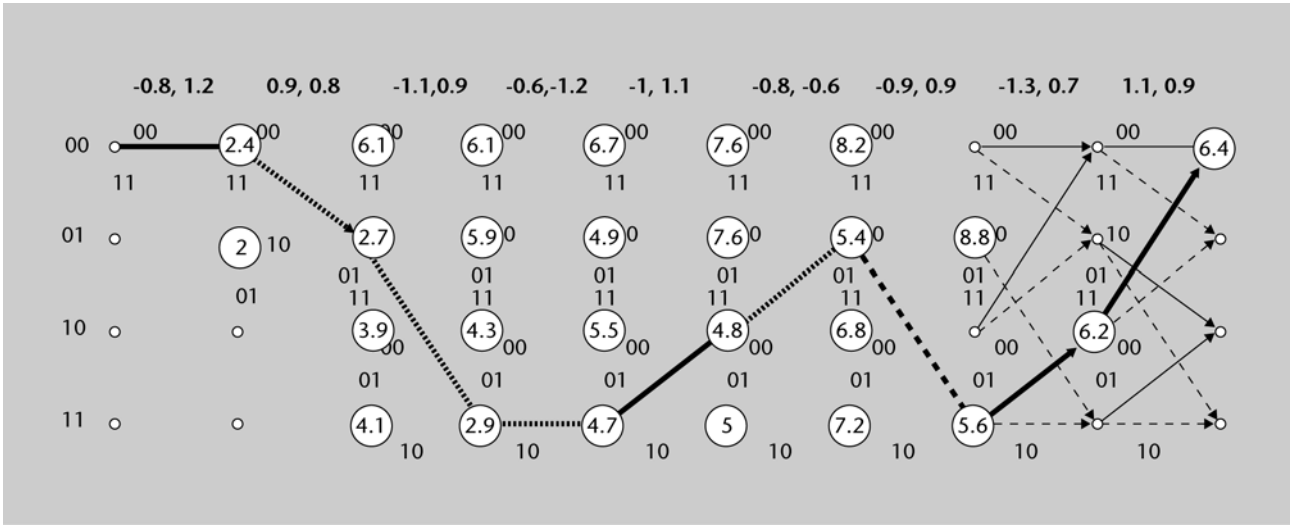

Figura 32. Descodificació *software* amb l'algorisme de Viterbi. Resultat final

Exercicis d'autoavaluació

1. Un codi convolucional queda descrit pels paràmetres següents:

- Nombre de línies d'entrada: $k = 1$
- Nombre de línies de sortida: $n = 3$
- Taxa del codi: $R = 1/2$
- Unitats de memòria: 2
- Nombre de possibles estats: $2^2 = 4$
- Longitud total de la convolució: 3
- Polinomi generador 1: $G_1 = 1 = 1_{OCT}$
- Polinomi generador 2: $G_2 = X^2 + 1 = 5_{OCT}$
- Polinomi generador 3: $G_3 = X^2 + X + 1 = 7_{OCT}$

A partir d'aquestes dades:

a) Determineu el diagrama de blocs del codificador.
b) Determineu i representeu el diagrama d'estat.
c) Representeu el diagrama de Trellis.
d) Verifiqueu si es tracta d'un codi catastròfic.
e) Calculeu-ne la distància lliure.

2. Per a les dades de l'exercici anterior, determineu:

a) La seqüència codificada que obtenim quan l'entrada és {1 0 1 1 1}.
b) Prenent el resultat de l'apartat anterior, suposem que s'ha produït el patró d'error següent {001 000 100 000 000 000 000}. Quina és la paraula que rebrem?
c) Utilitzant el resultat de l'apartat anterior determineu, utilitzant l'algorisme de Viterbi, l'estimació ML de la seqüència que s'ha transmès originalment.

3. Considereu un codi convolucional el diagrama de blocs del qual es representa a la figura 33.

a) Determineu la taxa del codi, els polinomis generadors i el nombre d'estats necessaris per a representar-lo.
b) Representeu el diagrama d'estats del codi.
c) Representeu el diagrama de Trellis.
d) Determineu la sortida del sistema per a la seqüència d'entrada següent {1 1 0 1 1}.
e) Suposem que es transmet una seqüència en un canal amb soroll i que es rep el resultat següent: {101 011 101 100 001 101 011}. Utilitzant l'algorisme de Viterbi realitzeu una estimació ML de la seqüència original que s'ha enviat. Així mateix, calculeu el nombre d'errors que s'estima que s'han produït i determineu la seqüència de dades original.

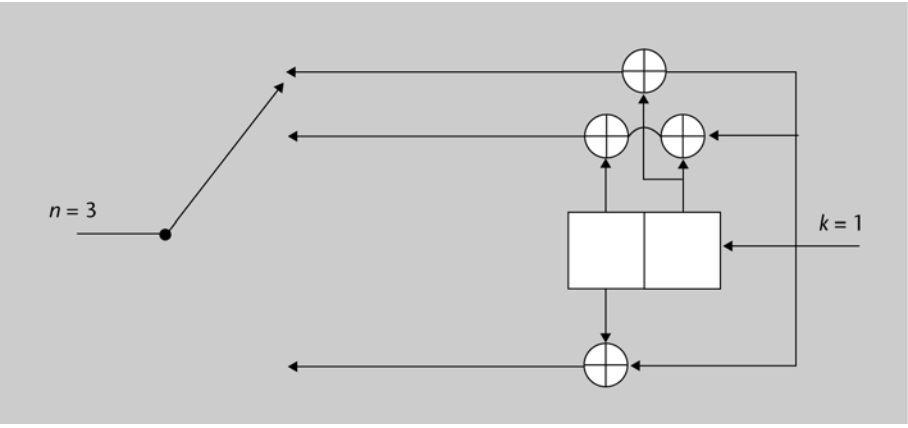

Figura 33. Diagrama d'un codi convolucional amb taxa 1/3

Bibliografia

Bibliografia bàsica

Proakis, J. G.; Salehi, M. (2002). *Communication Systems Engineering* (2a. ed.). Prentice Hall.

Benedetto, S.; Biglieri, E. (1999). *Principles of Digital Transmission*. Kluwer Academic Press / Plenum Publishers.

Proakis, J. G. (2003). *Digital Communications* (4a. ed.). McGraw Hill.

Bibliografia complementària

Carlson, A. B. (2001). *Communication Systems: An Introduction to Signals and Noise in Electrical Communication* (4a. ed.). McGraw Hill.

Viterbi, A. J. (1967). "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm". *IEEE Trans. on Information Theory* (núm. 2, pág. 260-269).

Fano, R. M. (1963, abril). "A Heuristic Discussion of Probabilistic Decoding". *IEEE Trans. on Information Theory* (vol. 9, pàg. 64-74).

Jelinek, F. (1969, nov.). "Fast Sequential Decoding Algorithm Using a Stack". *IBM J. Res. Dev.* (vol. 13, pàg. 675-685).

Heller., J. A. (1975). "Feedback Decoding of Convolutional Codes". *Advances in Communications Systems* (vol. 4.). J. Viterbi (ed.). Nova York: Academic Press.

Stix, G. (sept., 1991). "Encoding the Neatness of Ones and Zeroes". *Scientific American* (pàg. 54-58).

Segal, J. "El geómetra de la información". *Temas Investigación y Ciencia: La Información* (núm. 36, pàg. 12-15).

Bekenstein, Jacob D. "La información en el Universo Holográfico". *Temas Investigación y Ciencia: La Información* (núm. 36, pàg. 16-23).

Gibson, Jerry D. i altres (1998). *Digital Compression for Multimedia: Principles & Standards*. Morgan Kauffman.

Tarrés, F. (2001). *Sistemas Audiovisuales I: Televisión Analógica y Digital*. Ediciones UPC.