ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

ESTUDIO TEORICO DE LA RELACION ENTRE LA BER, LOS CODIGOS BLOQUE LINEALES Y LA POTENCIA DE TRANSMISION EN SISTEMAS CON MODULACION PSK Y QAM MULTINIVEL

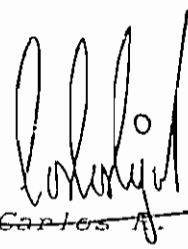
TESIS PREVIA A LA OBTENCION DEL TITULO DE INGENIERO EN LA ESPECIALIDAD DE ELECTRONICA Y TELECOMUNICACIONES

LUIS A. RIVERA SANCHEZ

QUITO, NOVIEMBRE DE 1994

CERTIFICACION

Certifico que esta Tesis ha sido desarrollada en su totalidad por el Sr. Luis A. Rivera Sánchez.



Ing. Carlos R. Egas A.

DIRECTOR DE TESIS

Quito, Noviembre de 1994

Mi agradecimiento muy especial a mis padres y hermanos por su apoyo constante, al Ing. Carlos Egas por su ayuda como Director de esta Tesis. A todas las personas que de alguna manera colaboraron para la culminación de este trabajo.

GRACIAS.....

"Para prosperar dentro de una profesión no hay que tomarla como medio sino como fin, y esto se realiza consagrando a la misma todos nuestros estudios, energías y desvelos".

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

CAPITULO I.

INTRODUCCION.

En este capítulo, se describe brevemente términos ya conocidos que son parte de Teoría de la Información como también de la materia de Propagación en Sistemas de Radioenlaces Digitales, con el propósito de obtener los suficientes elementos teóricos que servirán para dar un mejor enfoque de los objetivos planteados de este documento.

1.1 CODIGOS BLOQUE LINEALES.

Un código bloque de longitud n y 2^k palabras código, es llamado código bloque lineal (n,k), si y solo si

las 2^k palabras código forman un subespacio k-dimensional del espacio n-dimensional que puede formarse con todos los vectores de longitud n.

En otras palabras un código bloque es lineal, si y solo si la suma modulo 2 de una palabra código es otra palabra código.

Si un código bloque lineal C(n,k) es un subespacio k-dimensional del espacio vectorial v_n , entonces es posible encontrar k vectores código linealmente independientes:

$$g_0, g_1, \ldots\ldots\ldots\ldots, g_{k-1}$$

tal que cualquier palabra código v sea una combinación lineal de estos k vectores:

$$v = u_0\, g_0, u_1\, g_1, \ldots\ldots\ldots, u_{k-1}\, g_{k-1} \qquad (1.1)$$

Entonces, $u = (u_0, u_1, \ldots\ldots\ldots, u_{k-1})$ representa la palabra mensaje a ser codificado, y v su palabra código que es igual a:

$$v = u \cdot G \qquad (1.2)$$

Con los elementos de los k vectores código linealmente independientes se puede formar una matriz, que representa la matriz generadora, con la que es posible

encontrar todas las palabras código del código bloque lineal.

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \cdot \\ \cdot \\ \cdot \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & g_{0,2} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & g_{1,2} & \cdots & g_{1,n-1} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdots & g_{k-1,n-1} \end{bmatrix} \qquad (1.3)$$

$$g_i = (g_{i,0} \quad g_{i,1} \quad g_{i,2} \quad \cdots\cdots g_{i,n-1}) \qquad 0 \leq i \leq k-1$$

Esta Matriz Generadora G tiene las siguientes características:

- Genera el subespacio vectorial V_k.
- Esta formada por los elementos de los k vectores linealmente independientes.
- Se puede encontrar diversos conjuntos de vectores linealmente independientes que formen la matriz G, pero sólo algunos de ellos generan códigos buenos.

1.1.1 CODIGO BLOQUE LINEAL SISTEMATICO.

Bajo cierta estructura de la matriz G, es posible obtener que cada palabra código tenga cierta característica como se presenta en la figura 1.1.

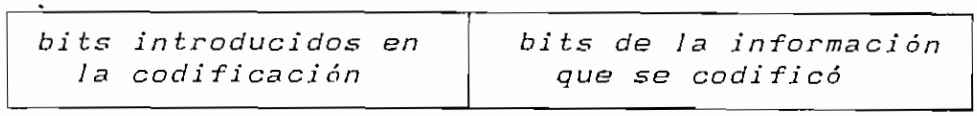


FIGURA 1.1 CODIGO SISTEMATICO

Cuando las palabras código poseen esta característica, al código se lo llama código lineal sistemático.

Esta estructura facilita el proceso de codificación ya que en lugar de calcular los n dígitos de la palabra código, el codificador solo debe calcular n−k dígitos denominados bits de control.

$$G = | \; P \quad I_k \; | \qquad (1.4)$$

donde:

I_k : Matriz identidad k x k.

P : Matriz de paridad k x n−k.

Para calcular la palabra código v utilizamos la ecuación (1.3).

Una manera de implementar un circuito codificador de un código bloque lineal sería el de almacenar todas las palabras código posibles, las cuales estarían relacionadas a solo una secuencia de mensaje a codificar, sin embargo este método para códigos cuyo

valor de n es demasiado grande, implicaría utilizar gran cantidad de memoria.

Para cualquier matriz G(k x n) con k filas linealmente independientes, existe una matriz que se la representará por H ((n−k) x n), denominada matriz verificadora de paridad, con n−k filas linealmente independientes tal que cualquier fila en G es ortogonal a cualquier fila de H, es decir:

$$g_i \cdot h_j = 0 \qquad (1.5)$$

La matriz H se la encuentra a partir de la matriz G.

$$H = \left| \; I_{n-k} \quad P^T \; \right| \qquad (1.6)$$

La matriz H genera un código lineal sistemático C_D denominado código dual del código C generado por la matriz G.

Esta propiedad implica que cualquier n-tupla v es una palabra código generado por G si y solo si se cumple la siguiente condición:

$$v \cdot H^T = 0 \qquad (1.7)$$

Utilizando esta propiedad se puede comprobar en el receptor si el bloque recibido esta libre de error.

En el receptor, el decodificador realiza el cálculo de la siguiente expresión cuyo resultado se lo conoce como Síndrome, es decir:

$$S = r \cdot H^T \qquad (1.8)$$

$$S = (s_0, s_1, \ldots\ldots\ldots\ldots, s_{n-k-1})$$

donde S representa el Síndrome.

Se debe tener en cuenta que S es igual a cero si y solo si r es una palabra código y por el contrario es diferente de cero si y solo si r no es una palabra código, por lo que existen errores en el bloque recibido r.

1.1.2 CODIGOS HAMMING.

Este tipo de códigos cíclicos son ampliamente usados debido a la facilidad de implementación de los circuitos codificadores y decodificadores para el control de errores y su característica principal es que pueden detectar tipos de errores con un máximo de dos errores y pueden corregir tipos de errores con un error.

Para cualquier valor entero $m \geq 3$ *existe un código lineal con los siguientes parámetros:*

longitud del código	$n = 2^m - 1$
número de símbolos de información	$k = 2^m - m - 1$
número de símbolos verif. paridad	$n - k = m$
capacidad de corrección de errores	$t = 1,\ d_{min} = 3$

Una característica especial de este código es que la matriz verificadora de paridad H *contiene como columnas a todos los vectores de* m *elementos, excepto el vector cero. En forma esquemática, las columnas de* H *están arregladas de la siguiente forma:*

$$H = | \ I_m \quad Q \ | \tag{1.9}$$

donde:

I_m : *Matriz identidad* $m \times m$.

Q : *Matriz transpuesta de la matriz de paridad* P, *consiste de* $2^m - m - 1$ *columnas que son las* m*-tuplas de ancho 2 o más.*

m : $n - k$

Un código corrector de t-errores es llamado código perfecto si y sólo si el código corrige no más de t errores en la secuencia de n dígitos. Así, los códigos Hamming forman una clase de códigos perfectos

correctores de simple error. Además de los códigos Hamming, otro código perfecto es el código Golay (23,12).

Podemos eliminar algunas l columnas de la matriz verificadora de paridad H de un código Hamming. Resultando una $m \times (2^m - l - 1)$ matriz H'.

Usando H' como una matriz chequeadora de paridad, nosotros obtenemos un código Hamming corto con los siguientes parámetros:

longitud del código	$n = 2^m - l - 1$
número de símbolos de inf.	$k = 2^m - m - l - 1$
número de símbolos verif. paridad	$n - k = m$
distancia mínima	$3 \leq d_{min}$

Si eliminamos apropiadamente columnas de H, podríamos obtener un código Hamming corto de distancia mínima 4.

Este código Hamming de distancia 4 puede ser usado para corregir todos los patrones de errores de simple error y simultáneamente detecta todos los patrones de errores de doble error.

1.1.3 CODIGOS CICLICOS.

De todos los códigos lineales, los más utilizados son los códigos cíclicos, los cuales son usados más en detección.

Un código lineal C(n,k) se denomina cíclico si con cada desplazamiento circular de un vector código en C se obtiene otro vector también en C. Así, para el vector:

$$v = (v_o, v_1, ..., v_{n-1})$$

con un desplazamiento hacia la derecha se obtiene un vector que pertenece al código C:

$$v^{(1)} = (v_{n-1}, v_o, v_1, ..., v_{n-2})$$

Los componentes de v pueden tratarse como coeficientes de un polinomio de grado menor o igual a n-1, que se lo conoce como polinomio de código.

$$v(X) = v_o + v_1X + v_2X^2 + ...v_{n-1}X^{n-1}$$

con $v_i = 0$ o 1.

Los códigos cíclicos son una clase muy importante de los códigos bloque lineales ya que la implementación de los circuitos codificadores y decodificadores resulta ser menos complicada, esto se debe en gran parte a la propiedad cíclica que presentan.

Polinomio Generador.

Todos los polinomios que se pueden formar con todas las palabras código que conforman el código C, forman también un subespacio vectorial y por lo tanto existirá un polinomio que genere dicho espacio vectorial.

Este polinomio representado por $g(X)$ se lo conoce como polinomio generador, de tal manera que:

$$v(X) = u(X) \cdot g(X) \qquad (1.10)$$

donde:

$v(x)$: Polinomio de código.

$u(x)$: Polinomio de mensaje.

$g(x)$: Polinomio generador de grado mínimo.

$$g(X) = g_0 + g_1X + g_2X^2 + ... + g_{r-1}X^{r-1}$$

Propiedades del Polinomio Generador.

- *El polinomio generador de grado mínimo de un código cíclico es único.*

- *En el polinomio generador $g_o = 1$.*

- *En un código cíclico (n,k) existe uno y solo un polinomio generador de grado n-k.*

- *El grado del polinomio generador es igual al número de bits de control introducidos en la codificación.*

De esta manera el polinomio generador es:

$$g(X) = 1 + g_1X + g_2X^2 + \ldots\ldots + g_{n-k}X^{n-k}$$

El número de polinomios binarios de grado menor o igual a n-1 que son múltiplos de g(X) es igual a 2^{n-k} y son llamados polinomios de código del código cíclico C(n,k).

El código cíclico (n,k) esta completamente definido por su polinomio generador de grado mínimo g(X).

De esta forma para que un polinomio de grado menor a n-1 sea un código polinomial entonces este polinomio debe ser factor de g(X).

1.1.4 CODIGOS BCH.

El código BCH (Bose - Chaudhuri - Hocquenghem) es quizá el código más importante dentro de la clase de los códigos cíclicos.

Este tipo de códigos son una generalización de los códigos Hamming para la corrección múltiple de errores.

Fueron descubiertos por Hocquenghem (1959) e independientemente por Bose y Chaudhuri (1960) y tienen propiedades especiales para la corrección y algoritmos de decodificación simples.

Para cualquier entero positivo m tal que $m \geq 3$ y cualquier t con $t < 2^{m-1}$ existe un código BCH que tiene los siguientes parámetros:

longitud del bloque	$n = 2^m - 1$
número de dígitos de control	$n - k \leq mt$
distancia mínima	$d_{min} \geq 2t + 1$

Código Reed – Solomon.

En este tipo de código, cada símbolo puede ser representado por m bits. Estos códigos tienen los siguientes parámetros:

bits por símbolo	m
longitud del bloque	$n = m(2^m - 1)$ *bits.*
Número de bits de paridad	$n - k = 2mt$ *bits.*
Distancia mínima	$d = m(2t + 1)$ *bits.*

Códigos Reed – Solomon son muy adecuados para la corrección de ráfagas de errores.

1.1.5 OTROS CODIGOS BLOQUE LINEALES.

Código Dual.

En los códigos cíclicos se puede encontrar un polinomio h(X) a partir del cual se puede generar un código dual, C_D, al código C generado por g(X).

$$h(X) = (X^n + 1)/g(X) \qquad (1.11)$$

de donde:

$$h(X) = h_0 + h_1 X + \ldots\ldots + h_k X^k$$

con $h_0 = h_k = 1$.

h(X) es el polinomio generador de otro código cíclico, denominado código dual.

Código de máxima longitud.

Estos códigos cíclicos poseen los siguientes parámetros:

longitud del bloque	$n = 2^m - 1$
número de bits de información	$k = m$
distancia mínima	$d = 2^m - 1$

Se lo llama también código simplex.

Código de residuo cuadrático.

Las distancias mínimas de estos códigos son comparables a los códigos BCH de distancia semejante.

Los códigos de residuo cuadrático son códigos cíclicos con los siguientes parámetros:

longitud del bloque $n = p_r$

donde p_r es un número primo de la forma $8m + 1$.

número de bits de información $k = (p + 1).2$

distancia mínima $d > \sqrt{n}$

Código de repetición.

Es un código muy simple de forma (n,1). En éste, cada uno de los n-1 bits de paridad son iguales a los bit de información.

La distancia mínima del código es n y es así que para valores grandes de n el código posee una gran capacidad de corrección de errores, pero su eficiencia es baja.

El número de errores que pueden ser corregidos es:

$$t = (n - 1)/2$$

Código de Golay.

Este código es importante desde el punto de vista de corrección múltiple de errores (t > 1), es además un código perfecto. El código de Golay es un código cíclico (23,12) que tiene una distancia mínima de 7 y corrige 3 o menos errores. Relacionado con éste código (23,12) esta el código de Golay (24,12) el cual se lo

obtiene del anterior con la adición de un bit de paridad, tiene una distancia mínima de 8 y también corrige 3 errores.

1.1.6 OBTENCION DE CODIGOS BLOQUE LINEALES EN FORMA SISTEMATICA.

Sea $u = (u_0, u_1, \ldots, u_{k-1})$ el mensaje a ser codificado, entonces su representación polinomial es la siguiente:

$$u(X) = u_0 + u_1 X + \ldots + u_{k-1} X^{k-1}$$

Para obtener el polinomio código se multiplica a este polinomio por X^{n-k}:

$$X^{n-k} + u_1 X^{n-k+1} + \ldots + u_{k-1} X^{n-1}$$

Se divide esta expresión para $g(X)$ de donde se tiene que:

$$X^{n-k}.u(X) = a(X).g(X) + b(X) \qquad (1.12)$$

donde:

$u(x)$: Polinomio de mensaje.

$g(x)$: Polinomio generador.

$a(x)$: Cuociente de la división.

$b(x)$: Residuo de la división.

o lo que es lo mismo:

$$b(X) + X^{n-k}.u(X) = a(X).g(X) \qquad (1.13)$$

El polinomio representado por $b(X) + X^{n-k}.u(X)$ es un polinomio código, por ser múltiplo de $g(X)$.

Los coeficientes de este polinomio forman el vector de codigo v:

$$v = (b_0, b_1, \ldots, b_{n-k-1}, u_0, u_1, \ldots, u_{k-1})$$

Se concluye entonces que los bits de control de un código bloque lineal se obtienen de los coeficientes del residuo que resulta de dividir $X^{n-k}.u(X)$ para $g(X)$.

1.2 PROBABILIDAD DE DECODIFICACION ERRONEA DE LOS CODIGOS BLOQUE LINEALES.

La probabilidad de decodificación errónea es la probabilidad de que los datos decodificados en la recepción sean erróneos, los cuales se deben a que el decodificador no corrigió todos los errores que contenían las palabras código y también se incluye a la probabilidad de no detección de errores debido a

que los errores producidos en el canal ocacionan que la palabra código transmitida sea transformada en otra palabra código que el decodificador la interpretara como palabra recibida correcta y sin errores.

Si C es un código bloque lineal (n,k) y si además definimos a A_i con el número de palabras código de ancho i (palabras código que tienen i elementos diferentes de cero) en C, a los valores $A_0, A_1, \ldots, A_n$ toman el nombre de distribución de ancho de Hamming de cada palabra código, del código C.

Cuando un código lineal (n,k) es usado solo para detección de errores sobre un BSC, la probabilidad de no detección de error, $P_u(E)$, puede ser calculada con la siguiente relación[1]*:*

$$P_u(E) = \sum_{i=1}^{n} A_i \, p^i \, (1-p)^{n-i} \qquad (1.14)$$

donde:

$P_u(E)$: *Probabilidad de no detección de error.*

A_i : *Ancho de Hamming de la palabra código i.*

p : *Probabilidad de transición o de error del canal binario simétrico.*

---

1 ERROR CONTROL CODING, Lin Shu, pags. 65 y 66

Existe una relación entre la distribución de ancho Hamming de un código lineal y la distribución de ancho de Hamming de su código dual. Esta relación puede facilitar el cálculo de $P_u(E)$. Si la longitud de la palabra mensaje del código k de un código bloque lineal (n,k) es muy grande comparado con la longitud de los bits de paridad $n-k$, se puede realizar el cálculo de Pu(E) utilizando su código dual $(n,n-k)$.

Sean $(A_o, A_1, \ldots\ldots, A_n)$ y $(B_o, B_1, \ldots\ldots, B_n)$ las distribuciones de ancho de Hamming del código C y su dual respectivamente. A estas distribuciones de ancho de Hamming se las puede representar en forma polinomial como se indica:

$$A(z) = A_o + A_1 z + \ldots\ldots + A_n z^n$$

$$B(z) = B_o + B_1 z + \ldots\ldots + B_n z^n$$

Entonces $A(z)$ y $B(z)$ están relacionadas por la siguiente identidad:

$$A(z) = 2^{-(n-k)} \cdot (1+z)^n \cdot B\left(\frac{1-z}{1+z}\right) \qquad (1.15)$$

Identidad que es conocida como identidad de MacWilliams[2].

---

[2] ERROR CONTROL CODING, Lin Shu, pags. 76 y 77

Utilizando esta identidad, se puede calcular la probabilidad de no detección de error, Pu(E), de un código lineal (n,k) con la distribución de ancho de Hamming de su código dual.

A partir de la ecuación (1.14) se puede obtener:

$$P_u(E) = (1 - p)^n \cdot \left[A\left(\frac{p}{1-p}\right) - 1\right] \qquad (1.16)$$

donde:

$$A\left(\frac{p}{1-p}\right) - 1 = \sum_{i=1}^{n} A_i \cdot \left(\frac{p}{1-p}\right)^i \qquad (1.17)$$

considerando que $A_o = 1$.

Utilizando la ecuación (1.15), identidad de MacWilliams, se tiene:

$$P_u(E) = 2^{-(n-k)} \cdot B(1-2p) - (1 - p)^n \qquad (1.18)$$

donde:

$$B(1-2p) = \sum_{i=0}^{n} B_i (1 - 2p)^i \qquad (1.19)$$

Teóricamente, se puede calcular la distribución de ancho de Hamming de un código lineal (n,k) al examinar

cada una de las 2^k palabras código o cada una de las 2^{n-k} palabras código de su dual para aplicar la identidad de MacWilliams.

Para valores grandes de n, k, y n-k, el cálculo es prácticamente imposible. Excepto para algunos códigos lineales cortos. Consecuentemente, esta es una gran dificultad para calcular la probabilidad de no detección de error.

Experimentalmente se ha determinado la existencia de códigos bloque lineales cuya probabilidad de error decrece exponencialmente con el número de dígitos de paridad, es decir que existen códigos bloque lineales (n,k) tal que[3]*:*

$$P_u(E) \leq 2^{-(n-k)} \qquad (1.20)$$

En los códigos Hamming, la distribución de anchos de Hamming es fácilmente determinado. El número de vectores de ancho i, Ai, es simplemente el coeficiente de z^i en la expansión del siguiente polinomio[4]*:*

$$A(z) = \frac{1}{n+1} \cdot [(1+z)^n + n.(1-z).(1-z^2)^{(n-1)/2}] \qquad (1.21)$$

---

[3] ERROR CONTROL CODING, Lin Shu, pags. 78 y 79

[4] ERROR CONTROL CODING, Lin Shu, pag. 81

Este polinomio representa la distribución de anchos de Hamming de los códigos Hamming.

Para los códigos dual $(2^m - 1, m)$ de los códigos Hamming, es muy simple determinar la distribución de anchos de Hamming, este consiste de todos los ceros de la palabra código y $2^m - 1$ palabras código de ancho 2^{m-1}. Su polinomio de anchos de Hamming es:

$$B(z) = 1 + (2^m - 1) z^{2^{m-1}} \tag{1.22}$$

La probabilidad de no detección de error, $P_u(E)$, para los códigos Hamming puede ser calculada con:

$$P_u(E) = 2^{-m}[1 + (2^m - 1)(1 - 2p)^{2^{m-1}}] - (1 - p)^{2^m - 1} \tag{1.23}$$

La probabilidad $P_u(E)$ para estos códigos satisface el límite superior $2^{-(n-k)}$ y es igual a 2^{-m} para $p \leq 1/2$.

El código Hamming C_2 con distancia 4 cuya longitud es $2^m - 1$ consiste de los vectores con ancho par del código Hamming C_1 de distancia 3 y de longitud $2^m - 1$, como sus vectores código. El polinomio de anchos $A_2(z)$ para C_2 puede ser determinado del polinomio de anchos $A_1(z)$ para C_1. $A_2(z)$ consiste solamente de los términos de potencias pares de $A_1(z)$.

$$A_2(z) = \frac{[A_1(z) + A_1(-z)]}{2} \qquad (1.24)$$

entonces:

$$A_2(z) = \frac{1}{2.(n+1)} \cdot A' \qquad (1.25)$$

donde:

$$A' = [(1+z)^n + (1-z)^n + 2.n.(1-z^2)^{(n-1)/2}]$$

$n = 2^m - 1$.

El dual del código Hamming de distancia 4 es un código cíclico $(2^m - 1,\ m + 1)$ que tiene las siguiente distribución de anchos de Hamming:

$$B_0 = 1,\quad B_{2^{m-1}-1} = 2^m - 1,\quad B_{2^{m-1}} = 2^m - 1,\quad B_{2^m-1} = 1 \qquad (1.26)$$

Si un código Hamming con distancia 4 es usado para detección de errores en un BSC, su probabilidad de no detección de error, $P_u(E)$, puede ser calculada con la siguiente expresión:

$$P_u(E) = 2^{-(m+1)}[1 + 2(2^m - 1)(1-p)(1-2p)^{2^{m-1}-1} + (1-2p)^{2^m-1}] \qquad (1.27)$$

Los códigos cíclicos Hamming de distancia 4 satisfacen

las condiciones de borde superior $2^{-(n-k)} = 2^{-(m+1)}$.

Para un código primitivo BCH corrector de t errores usado para la detección de errores en un BSC con probabilidad de transición p y de longitud $2^m - 1$, *si el número de dígitos verificadores de paridad es igual a mt y m es mayor que cierta constante* $m_o(t)$, *el número de vectores código de ancho i satisface las siguientes ecuaciones*[5]:

$$A_i = 0 \qquad \text{para } 0 < i \leq 2t$$

$$A_i = (1 + \lambda_0 . n^{-1/10}) \left(\frac{n}{i} \right) 2^{-(n-k)} \qquad \text{para } i > 2t$$

(1.28)

donde $n = 2^m - 1$ *y* λ_0 *esta limitada por una constante.*

Sustituyendo la ecuación (1.28) en la ecuación (1.14), la probabilidad de no detectar error se obtiene con:

$$P_u(E) = (1 + \lambda_0 . n^{-1/10}) 2^{-(n-k)} \sum_{i=2t+1}^{n} \left(\frac{n}{i} \right) p^i (1 - p)^{n-i}$$

(1.29)

---

[5] ERROR CONTROL CODING, Lin Shu, pags. 179 y 180

y tiene por límite:

$$P_u(E) \leq (1 + \lambda_0 . n^{-1/10}) 2^{-(n-k)}$$

(1.30)

Para un código Reed-Solomon, corrector de t errores, de longitud $q - 1$, con símbolos de GF(q), el número de vectores código de ancho de Hamming j es:

$$A_j = \binom{q-1}{j} \sum_{i=0}^{j-2t-1} (-1)^i \binom{j}{i} (q^{j-2t-i} - 1)$$

(1.31)

1.3 TASA DE BITS ERRONEOS, BER.

En todo sistema de transmisión de datos por un canal, se tiene que los datos que llegan al receptor pueden contener errores. Un caso ideal sería que los datos que se han enviado se obtengan en el receptor sin necesidad de realizar ninguna corrección.

Debido a la existencia de errores en la secuencia recibida, se hace necesario utilizar diferentes métodos para corregirlos de tal manera que se obtenga alta confiabilidad de la transmisión.

La relación que existe entre el número de bits erroneos respecto el número de bits totales

transmitidos se conoce como tasa de bits erroneos (BER). En una transmisión de datos, este es uno de los parámetro que nos indica la calidad de la transmisión ya que mientras tenga menor valor, el número de bits erróneos es menor.

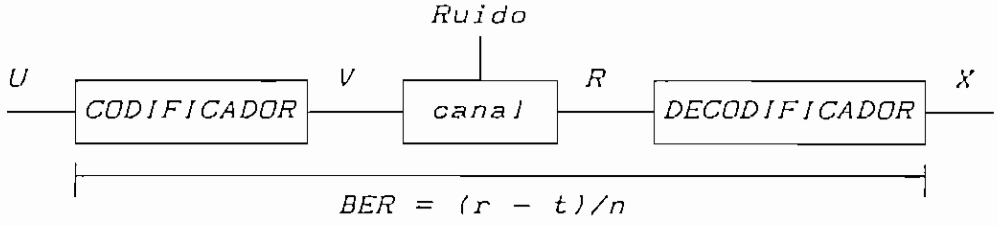


FIGURA 1.2 TRANSMISION DE DATOS A TRAVES DE UN CANAL.

Para el sistema en la figura 1.2, la tasa de bits erróneos en el canal de transmisión equivalente esta dada por $p = r/n$ (BER en la palabra código V) considerando una secuencia de n dígitos de una palabra código de un código (n,k) que corrige t errores, mientras que la probabilidad de error en la palabra mensaje U esta dada por:

$$BER = \frac{r - t}{n} \qquad (1.32)$$

donde:

r : Número de bits erróneos. $t < r \leq n$

t : Número de bits que corrige el código.

n : Número total de bits transmitidos.

1.3.1 RUIDO.

En sistemas de radioenlace, los objetivos de calidad de los sistemas digitales se miden en términos de BER. En el proceso de regeneración se produce la metamorfosis del ruido en errores, por ello en condiciones normales existe una relación matemática estadística entre la BER y la relación entre la potencia de la señal S y la potencia de ruido N, S/N.

En la figura 1.3 se observa la densidad espectral de potencia de ruido en función de la frecuencia y con la temperatura como parámetro (Ley de Planck). Se puede aproximar la curva por dos asíntotas: el ruido térmico que depende linealmente de T (temperatura absoluta en grados Kelvin) y el ruido cuántico que depende linealmente de la frecuencia.

Se puede observar que para $T = 290°K$ el ruido térmico es preponderante frente al cuántico hasta frecuencias del infrarrojo (10^{13} Hz.).

La transmisión por radioenlace esta afectada por el ruido térmico (que es producido por la agitación térmica molecular), así como también las interferencias a nivel de canales digitales.

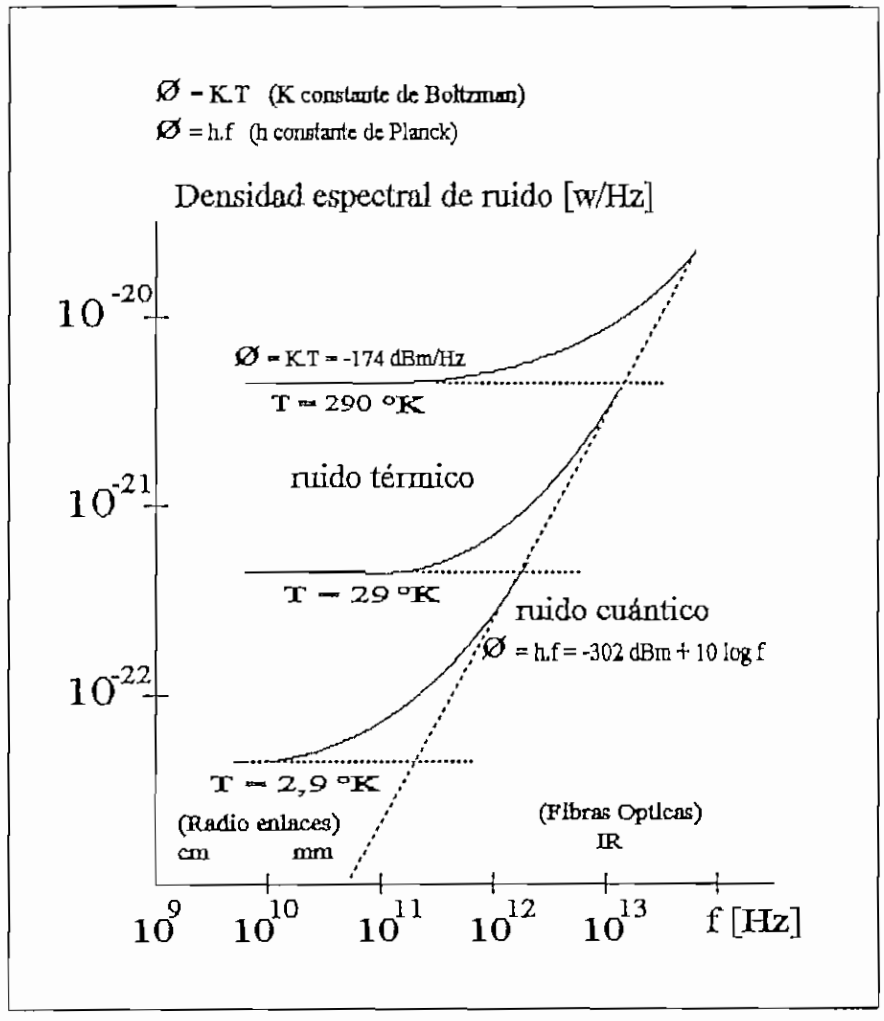


FIGURA 1.3 DENSIDAD DE RUIDO vs. FRECUENCIA.

El ruido térmico que introduce el receptor da lugar al denominado factor de ruido o figura de ruido del receptor, definición que se expresa por la degradación de la señal a ruido y que se relaciona mediante:

$$NF = 1 + (Tr/290) \qquad (1.33)$$

donde:

NF : Figura de ruido.

Tr : Temperatura de ruido equivalente del receptor.

El valor de NF se interpreta como los dB que se empeora la S/N en el receptor debido al ruido introducido por el preamplificador.

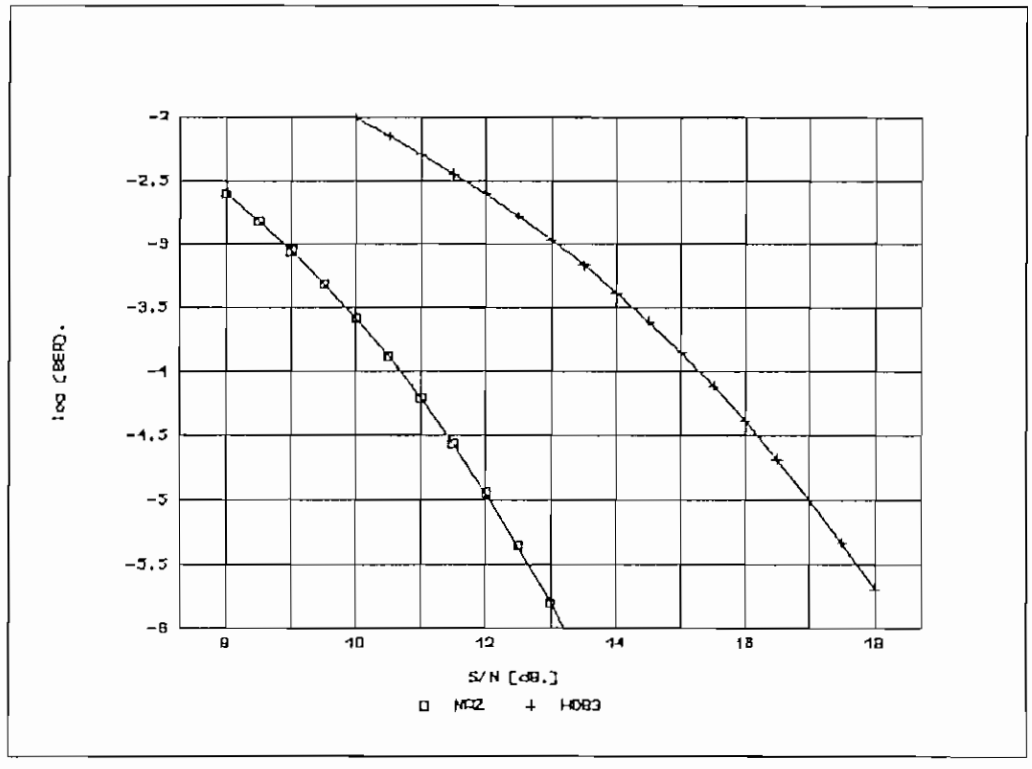


FIGURA 1.4 RELACION BER vs. S/N.

El ruido térmico que tiene una distribución de ruido gaussiano y de fase uniforme se adiciona a la señal

dando lugar a un valor de relación S/N (figura 1.4) y produciendo errores cuando se supera el umbral de decisión.

La probabilidad de error p representa el valor teórico de la BER. Los valores prácticos que vinculan la BER y S/N han sido medidos para transmisiones que utilizan códigos HDB3 y NRZ.

Existe una relación matemática precisa entre la probabilidad de error de los símbolos p_s, el número de niveles de la señal M (M = 2 para señal binaria y p_s = p) y la relación señal a ruido S/N. Se expresa como:

$$p_s = \left(1 - \frac{1}{M}\right) erfc \left[\frac{3.\log_2 M}{M^2 - 1} . W \right]^{1/2} \qquad (1.34)$$

donde W es la relación señal a ruido normalizada que se define mediante:

$$W = 10 \log (S/N).(Beq/R_b) \qquad (1.35)$$

Beq es el ancho de banda equivalente de ruido (ancho de banda plano con igual potencia que un ancho de banda gaussiano) y R_b la velocidad binaria de transmisión.

La calidad del canal digital puede medirse en términos de BER pero mas efectivo es utilizando la correlación BER vs. tiempo de duración. Esto se debe por ejemplo, a que una ráfaga de errores puede producir una elevada BER durante un corto período de tiempo y no resulta peligroso.

El ruido se superpone a la señal de impulsos en la banda base, y se producen errores de bits. En la radiocomunicación los ruidos que causan errores de bits se generan principalmente en la banda de frecuencia portadora, por eso hay que tomar en cuenta las relaciones entre los ruidos y la señal de esa banda.

1.4 POTENCIA DE TRANSMISION EN SISTEMAS DE RADIOENLACES DIGITALES CON MODULACION PSK Y QAM MULTINIVEL.

El funcionamiento de extremo a extremo de cualquier enlace de comunicación puede expresarse en términos de la razón de señal sobre ruido (S/N) evaluada en un ancho de banda apropiado, en algún punto del sistema de recepción.

En un sistema analógico en el cual el ancho de banda

del ruido suele ser mayor que el ancho de banda de la señal, a menudo se recurre a la razón promedio de potencia sobre ruido de la portadora (Pr/N), como la S/N de interés particular:

$$\frac{S}{N} = \frac{P_r}{N} = \frac{EIRP\ (G_r/T_s)}{L_p\ L_{os}\ L_{oi}\ K\ B} \qquad (1.36)$$

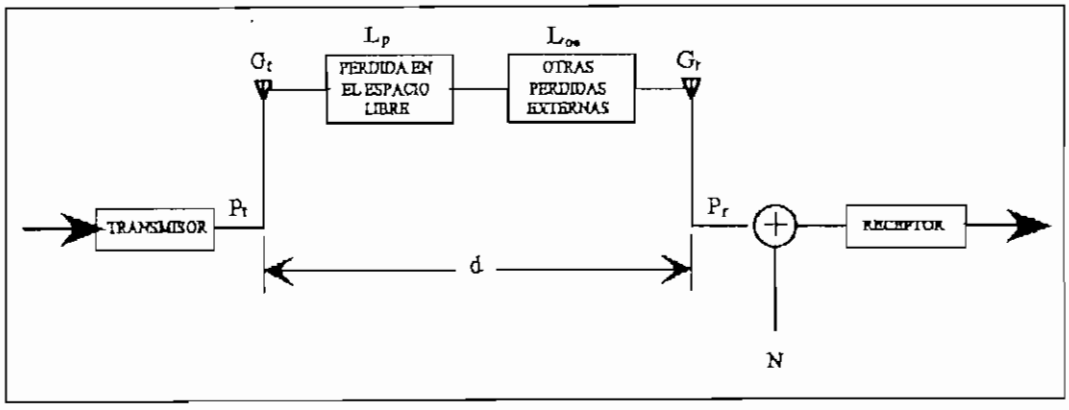


FIGURA 1.5 MODELO SIMPLIFICADO DE UN ENLACE DE COMUNICACIONES

Los diversos parámetros de la ecuación (1.36) pueden definirse con la ayuda del modelo de enlace que se muestra en la figura 1.5.

donde:

P_r : Potencia de la señal recibida en la entrada del detector, watts.

N : $K\ T_s\ B$ = Potencia del ruido térmico a la

entrada del detector, watts.

K : Constante de Boltzmann, 1.38×10^{-23} J/°K.

T_e : Temperatura de ruido equivalente del sistema, °K.

B : Ancho de banda a la entrada al detector, Hz.

$EIRP$: $P_t \cdot G_t$ = Potencia radiada isotrópica equivalente, watts.

P_t : Potencia transmitida, watts.

G_t : Ganancia de la antena de transmisión.

G_r : Ganancia de la antena de recepción.

G_r/T_e : Factor de mérito, razón de ganancia sobre temperatura de ruido equivalente del sistema, $^oK^{-1}$.

L_p : $(4\pi d/\lambda)^2$ = Pérdida de espacio libre, d y λ en las mismas unidades.

λ : Longitud de onda de la frecuencia del enlace.

d : Distancia del enlace.

L_{os} : Otras pérdidas externas = pérdida atmosférica + pérdida por polarización de antena + pérdida por direccionamiento de antena.

L_{oi} : Otras pérdidas internas = pérdida del circuito de transmisión + pérdida del circuito de recepción + pérdida por ruido de intermodulación.

En un sistema digital en el que el ancho de banda de la señal se considera igual al ancho de banda del ruido, el rendimiento del enlace se expresa en términos de la razón potencia de la señal recibida sobre densidad espectral de ruido (P_r/N_o):

$$\frac{P_r}{N_o B} = \frac{EIRP\ (G_r/T_s)}{L_p\ L_{os}\ L_{oi}\ K\ B} \qquad (1.37)$$

donde $N_o = N/B$

N_o : Densidad espectral de ruido en watts por hertz.

Si se supone que toda la potencia recibida proviene de la señal de modulación (portadora suprimida), entonces es posible evaluar el rendimiento del enlace en términos de la razón de la energía en los bits E_b sobre la densidad espectral de ruido N_o.

Puede expresarse:

$$\frac{P_r\, B}{N} = \frac{P_r}{N_o} = \left(\frac{E_b}{N_o}\right) R_b \qquad (1.38)$$

Entonces de las ecuaciones (1.37) y (1.38) se tiene:

$$\frac{E_b}{N_o} = \frac{EIRP\ (G_r/T_s)}{L_p\, L_{os}\, L_{oi}\, K\, R_b} \qquad (1.39)$$

donde R_b es la velocidad de transmisión de datos de información en bits por segundo. Si la potencia de la portadora no es despreciable, aún se puede utilizar la ecuación (1.39), suponiendo que la potencia de la portadora se considera una pérdida en el parámetro L_{oi}. En decibeles, la ecuación (1.39) puede expresarse como:

La cantidad de margen de enlace por utilizar depende de gran medida de la naturaleza estadística del enlace y de la habilidad que se tenga para predecir todas las fuentes de ganancia y pérdida y fenómenos de ruido. En la práctica se han utilizado márgenes de enlace que varían de 0 a 6 dB., lo que refleja la gran variación del nivel de la señal recibida con el cual es posible modelar los diversos enlaces de comunicación.

CAPITULO II.

CODIGOS BLOQUE LINEALES Y CODIGOS CONVOLUCIONALES.

2.1 COMPARACION ENTRE LOS CODIGOS BLOQUE LINEALES Y LOS CODIGOS CONVOLUCIONALES.

En los códigos bloque lineales, un bloque de k dígitos de datos se codifica mediante una palabra de código de n dígitos (n > k). Para cada sucesión de k dígitos de datos, existe una palabra de código distinta de n dígitos.

En los códigos de convolución, la sucesión de n dígitos codificada depende no sólo de los k dígitos de datos sino también de los anteriores M - 1 bloques de dígitos de datos (M > 1). Por lo tanto, la sucesión codificada para un cierto grupo de k dígitos de datos no es única sino que depende de los M-1 dígitos de datos anteriores. En los códigos de bloques, los k dígitos de datos se acumulan y luego se codifican en una palabra de código de n dígitos. En los códigos de convolución, la codificación se lleva a cabo sobre una base continua, en lugar de acumular k dígitos de datos.

Un código convolucional (n,k,M) puede ser implementado con un circuito secuencial lineal de k entradas, n salidas y con memoria de entrada M.

Típicamente n y k son enteros pequeños con k < n, pero el orden de la memoria M puede ser largo para tener baja probabilidad de error. En el caso especial cuando k = 1, la secuencia de información no es dividida en bloques y puede ser procesada continuamente.

El código convolucional debe su origen al desarrollo de la decodificación secuencial (probabilística) de Wozencraft y de la decodificación por umbral

(realimentación, algebraico) de Massey. El algoritmo de decodificación secuencial de Fano se utilizó en muchas versiones primitivas de decodificadores secuenciales.

Posteriormente aparece la técnica de decodificación con el algoritmo de Viterbi, el cual dio origen al aparecimiento de las decisiones por máxima probabilidad.

2.1.1 EFICIENCIA.

La eficiencia de un código bloque lineal (que se conoce también como índice de código) esta dado por la siguiente relación:

$$\eta = \frac{k}{n} \qquad (2.1)$$

donde:

η : *Eficiencia.*

k : *Longitud de la palabra mensaje.*

n : *Longitud de la palabra código.*

Este código se conoce como (n,k). *Los dígitos de datos* $(u_0, u_1, \ldots, u_{k-1})$ *son una k-tupla y en consecuencia,*

un vector **d** de **k** dimensiones. En forma similar, la palabra de código (v_0, v_1, ..., v_{n-1}) es un vector v de **n** dimensiones.

Para los códigos de convolución **k** y **n** son por lo general pequeños, se pueden diseñar para la corrección de errores aleatorios, ráfagas de errores, o ambos.

Un codificador de convolución con restricción de longitud **M** consta de un registro de corrimiento de **M** etapas y **v** sumadores módulo 2. En la práctica, **k >> M**, en consecuencia hay aproximadamente **kv** dígitos codificados de salida por cada **k** dígitos de datos, dando una eficiencia:

$$\eta = \frac{k}{(M + k)\,v} \qquad (2.2)$$

$$\eta \approx \frac{1}{v} \qquad (2.3)$$

donde:

η : Eficiencia.

k : Longitud de la palabra mensaje.

M : Memoria de entrada del código, longitud de la restricción.

v : Número de sumadores módulo 2.

2.1.2 REDUNDANCIA.

Para los códigos Hamming, si k dígitos de datos se transmiten mediante una palabra de código de n dígitos, el número de dígitos de comprobación o redundancia es $m = n - k$.

Para los códigos primitivos BCH, que corrigen t errores y tienen longitud $n = 2^m - 1$, *los dígitos de comprobación o redundancia esta dado por la siguiente relación:*

$$mt = n - k \tag{2.4}$$

donde:

t : Número de errores que corrige el código.

n : Longitud de la palabra código.

k : Longitud de la palabra mensaje.

Para los códigos convolucionales, la redundancia es igual a:

$$kv - k \tag{2.5}$$

donde:

k : Longitud de la palabra mensaje.

v : Número de sumadores módulo 2.

2.1.3 CAPACIDAD DE CORRECCION DE ERRORES.

La información se traduce a secuencias binarias y se transmite vía técnicas de modulación digital. Estas técnicas, si bien son muy eficientes para la transmisión de alta velocidad, sufren del defecto de que si unos pocos bits son recibidos incorrectamente, el mensaje no puede ser traducido al texto original.

Los códigos correctores de errores son usados generalmente para corregir errores donde solamente la transmisión en un sólo sentido es posible.

La información que va a ser transmitida, primero se la transforma en secuencias binarias. Estas secuencias son luego codificadas agregando bits redundantes (control), y la secuencia total de bits de información y control es transmitida.

En el receptor, un decodificador aplica el proceso inverso del algoritmo de codificación para encontrar los errores en la totalidad de información y control. Estos errores son corregidos, y las secuencias de información corregidas son traducidas al texto original.

A) Para los Códigos Bloque Lineales.

Para llegar a una expresión que permita definir la capacidad de un código bloque lineal es necesario definir los siguientes conceptos.

Ancho de Hamming.

Si v es una n-tupla binaria entonces se define el ancho de Hamming de v representado por w(v) como el número de elementos diferentes de cero que tiene v.

Distancia de Hamming.

La distancia de Hamming entre dos palabras código v y w representado como d(v,w) se define como el número de lugares en los cuales ambas palabras código difieren. Este parámetro determina la capacidad de corrección y detección de errores de un código.

La distancia de Hamming es una función métrica que satisface la desigualdad triangular. Sea v, w y x tres vectores o palabras código, entonces:

$$d(v,w) + d(w,x) \geq d(v,x) \qquad (2.6)$$

Con la definición de distancia de Hamming y de la suma en módulo 2 la distancia de Hamming entre dos palabras código, v y w, es igual al ancho de Hamming de la suma

de v y w, es decir:

$$d(v,w) = w(v + w) \qquad (2.7)$$

Distancia mínima de Hamming.

La distancia mínima de Hamming de C, d_{min}, esta definida como:

$$d_{min} = \min[d(v,w) : v, w \in C, v \neq w] \qquad (2.8)$$

Si C es un código bloque lineal, la suma de dos vectores es también otro vector código, por lo tanto,

$$d_{min} = \min [w(v + w) : v, w \in C, v \neq w]$$

$$= \min [w(x) : x \in C, x \neq 0]$$

$$d_{min} \triangleq w_{min} \qquad (2.9)$$

donde el ancho mínimo del código lineal C es:

$$w_{min} \triangleq \min [w(x) : x \in C, x \neq 0] \qquad (2.10)$$

Cuando un vector v es transmitido a través de un canal con ruido, si en el vector r recibido existen l

errores, entonces r difiere del vector transmitido en l lugares, obteniéndose que $d(v,r) = l$.

Un código bloque lineal con distancia mínima d_{min} es capaz de detectar todos los errores con $d_{min} - 1$, o menores, sin embargo puede darse la posibilidad de que pueda detectar algunos con d_{min} o más errores.

En general un código bloque lineal (n,k) es capaz de detectar $2^n - 2^k$ secuencias erróneas de longitud n, denominados errores detectables.

El número de tipos de errores que puede corregir un código esta íntimamente relacionado con el número de síndromes diferentes que puede generar un decodificador, ya que cada valor de síndrome esta asociado con un tipo de error diferente. El tipo de error es una palabra de longitud n, la cual sumada a la palabra código recibida r (de longitud n) que tiene error da por resultado la correcta palabra código transmitida v. El tipo de error se lo puede obtener a partir de la ecuación (1.8) como se indica a continuación:

$$S = r \cdot H^T$$

$$S = (e + v) \cdot H^T$$

$$S = e.H^T + v.H^T$$

Aplicando la ecuación (1.7) se tiene:

$$S = e \cdot H^T \qquad (2.11)$$

donde:

S : Síndrome.

e : Tipo de error.

H^T : *Transpuesta de la matriz verificadora de paridad.*

Un código bloque con distancia mínima d_{min} *garantiza la corrección de todos los tipos de errores de*[1]*:*

$$t = (d_{min} - 1)/2, \text{ o menos errores.} \qquad (2.12)$$

Este parámetro es llamado capacidad de corrección de errores aleatorios del código.

Un código bloque con capacidad de corrección de errores aleatorios t es usualmente capaz de corregir algunos tipos de errores de t + 1 o más errores. Un código lineal (n,k) corrector de t errores, es capaz de corregir un total de 2^{n-k} *tipos de errores incluyendo aquellos con t o menos errores.*

Si un código bloque (n,k) corrector de t errores es

---

[1] ERROR CONTROL CODING, Lin Shu, pag. 67.

Folleto de TEORIA DE LA INFORMACION Y CODIFICACION, Ing. Carlos Egas, pag. 46.

estrictamente utilizado para corregir en un BSC con probabilidad de transición p, la probabilidad que el decodificador cometa una decodificación errónea (probabilidad de error en la palabra mensaje) tiene como borde superior:

$$P(E) \leq \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}$$

(2.13)

donde:

P(E) : Probabilidad que el decodificador cometa una decodificación errónea.

n : Longitud de la palabra código.

t : Número de errores que corrige el código.

p : Probabilidad de transición del BSC.

Otra forma de corregir errores es mediante el diseño de un código para detectar (no para corregir) hasta t errores. Cuando el receptor detecta un error, solicita retransmisión. Ya que la detección de errores requiere de menos dígitos de comprobación, estos códigos operan con mayor eficiencia.

Para detectar t errores, las palabras de código deben estar separadas una distancia de Hamming de no más de t+1. Supongamos que una palabra de código transmitida v_j tiene un número α de errores ($\alpha \leq t$); entonces la

En el código Gallager adaptivo la operación del codificador es similar a la descrita para la técnica Massey.

El decodificador de un código Gallager corrige ráfagas de hasta 2b bits (b pares de información y paridad). El código Gallager es capaz de corregir ya sea errores aleatorios como errores tipo ráfaga.

El peor de los errores, si una ráfaga ocurre en el lugar en el que el código esta buscando un espacio de guardia, no solamente que la ráfaga en el decodificador no será corregida, sino que la ráfaga que ingresa se proyectará hacia adelante, generándose así nuevos errores y la ráfaga no puede corregirse.

Por lo tanto, el código Gallager puede emplearse con más éxito en canales donde los errores ocurran en ráfagas densas con espacios de guardia libres de errores.

El algoritmo de Viterbi es utilizado en numerosas aplicaciones en sistemas de comunicaciones, es particularmente útil en transmisiones de datos de alta velocidad, donde existen muchos errores.

procedimientos de codificación, el control de errores híbrido exhibe las ventajas de ambos y las desventajas de ninguno.

En un ambiente de error aleatorio, el código corrector de errores brindará corrección de errores dentro del código y la retransmisión eliminará cualquier error residual.

En un ambiente de ráfagas de error de alta densidad la retransmisión se utilizará para la corrección de errores, y el control de errores con códigos correctores elimina los errores en los intervalos entre ráfagas.

Si la densidad de errores en ráfagas aumenta hacia un nivel extremadamente alto y las ráfagas se hacen excesivamente largas, o si errores aleatorios ocurren con tasa de error más alta que un umbral (porcentaje de bits de error corregible) derivados del código, el sistema entra en retransmisión continua y fallará.

En falla, el código corrige un insuficiente número de errores y la longitud de la ráfaga excede la longitud del bloque, causando retransmisión continua.

El desempeño de este sistema híbrido esta descrito por la siguiente ecuación de eficiencia de transmisión.

$$E = \frac{\textit{Información a ser transmitida (segundos)}}{\textit{Tiempo total de operación (segundos)}}$$

$$E = \frac{I}{I(1 + R_d + R_c) + I_r(1 + R_d + R_c) + D_T} \qquad (2.15)$$

donde:

E : *Eficiencia de transmisión*

I : *Información a ser transmitida (segundos), igual a datos (bits) por tasa de datos (bits/seg).*

R_d : *Redundancia de detección de errores, igual a la relación de paridad de detección con respecto a los bits de información.*

R_c : *Redundancia de corrección de errores, igual a la relación de paridad de corrección con respecto a la paridad de detección y a los bits de información.*

I_r : *Información rechazada (segundos).*

D_T : *Tiempo de retardo (segundos), igual al producto del retardo del control de errores (1 + mensajes transmitidos) (segundos) por sistemas interrumpidos.*

Si el sistema opera como un sistema continuo:

D_T = *Retardo de control de error.*

Si no hay retransmisiones:

$$E = \frac{I}{I(1 + R_d + R_c) + D_T}$$

(2.16)

ya que $I_r = 0$.

En el caso en que D_T *es despreciable comparable con I y no hay retransmisiones:*

$$E = \frac{I}{I(1 + R_d + R_c)} = \textbf{eficiencia del código híbrido}$$

(2.17)

Si $D_T = 0$, *o* $D_T << I$, *la eficiencia es independiente de I y de la tasa de datos. De este modo, es posible considerar la eficiencia como la tasa efectiva* $1/(1+R)$ *del sistema híbrido.*

La tasa efectiva es siempre menor que la tasa del código.

La cercanía de E hacia la tasa del código actúa como un criterio de desempeño del sistema.

2.2 CALCULO DE Pu(E) PARA LOS CODIGOS BLOQUE LINEALES, ALGORITMO Y PROGRAMA.

La probabilidad de no detección de error Pu(E) para un canal binario simétrico BSC utilizando códigos bloque lineales, en forma general, se la calcula con la ecuación (1.14) y que se la indica a continuación:

$$P_u(E) = \sum_{i=1}^{n} A_i \, p^i \, (1 - p)^{n-i}$$

donde:

A_i : Número de palabras código de ancho i, distribución de ancho del código.

p : Probabilidad de transición del canal binario simétrico.

n : Longitud de la palabra código.

Para el cálculo de Pu(E) con el código dual, se utiliza las ecuaciones (1.18) y (1.19) obteniéndose:

$$P_u(E) = 2^{-(n-k)} \left[\sum_{i=0}^{n} B_i \, (1 - 2p)^i \right] - (1 - p)^n \qquad (2.18)$$

donde:

k : Longitud de la palabra mensaje.

B_i : *Número de palabras código de ancho i, distribución de anchos del código dual.*

2.2.1 CALCULO DE LA DISTRIBUCION DE ANCHO DEL CODIGO (n,k) O SU DUAL (n,n-k).

Uno de los parámetros importantes para el cálculo de la probabilidad de no detección de error, Pu(E), es la Distribución del Ancho del Código, Ai (Bi si es dual).

El tipo de codificación a utilizarse es de la forma sistemática, ya que de esta manera se puede distinguir fácilmente cual es la parte de los bits de paridad y cual la de los bits de mensaje.

El cálculo de la Distribución de Ancho de los Códigos Bloque, se realiza utilizando los polinomios generadores de los códigos primitivos binarios BCH[2].

Los coeficientes de estos polinomios generadores se encuentra agrupados en forma octal, por lo que para su mejor manejo se ha realizado un reagrupamiento en forma hexadecimal.

---

[2] ERROR CONTROL CODING, Lin Shu, Appendix C. Polinomios Generadores de Códigos Primitivos Binarios BCH de longitud de hasta $n = 2^{10} - 1$, presentados en forma octal.

Por ejemplo, para el código (63,45), el polinomio generador en forma hexadecimal (Anexo II.1) es:

7 8 2 C F

expandiendo esto en código binario, se tiene

0111 1000 0010 1100 1111

El polinomio generador es:

$$g(X) = X^{18} + X^{17} + X^{16} + X^{15} + X^{9} + X^{7} + X^{6} + X^{3} + X^{2} + X + 1$$

Utilizando un programa auxiliar, COE_HEXA.BAS, se puede realizar el proceso indicado anteriormente, es decir únicamente reagrupar los coeficientes de los polinomios generadores. Este programa una vez ejecutado almacena los resultados en el archivo TIPO_COD.TBL (Anexo II.1), y por lo tanto ya no es necesario repetirlo. Como información, el listado de este programa en QBasic se encuentra en el Anexo I.1.

Para un código (n,k) o su dual (n,n-k) se puede calcular la distribución de ancho creando todas las palabras binarias de longitud k, n-k para el dual, para luego codificarlas y determinar el ancho de cada una de estas palabras código, es decir contabilizar el

número de 1's que posee dicha palabra código. Se obtiene luego el número de palabras que tienen el mismo ancho i, A_i o B_i.

*Esto no es complicado pero si muy extenso si se trata de realizarlo manualmente, por lo que para facilitar el cálculo con los diferentes códigos, se ha desarrollado un programa en computador, **ANCH_BLQ.BAS**, que permite este cálculo. El diagrama de flujo se lo presenta en la figura 2.1 y su listado en QBasic en el Anexo I.2.*

*Es necesario indicar que se utiliza el cálculo a partir del código dual cuando para un código (n,k), el valor de **n-k** es menor que **k**. De darse este caso, el programa calcula el polinomio generador del código dual.*

La secuencia que sigue el programa de la figura 2.1 es la siguiente:

*Previamente son almacenados en los vectores **N()**, **K()**, **T()** y **G1$()** cada uno los valores de n, **k**, t y los coeficientes del polinomio generador representados como una sola palabra, estos elementos se los toma del archivo TIPO_COD.TBL.*

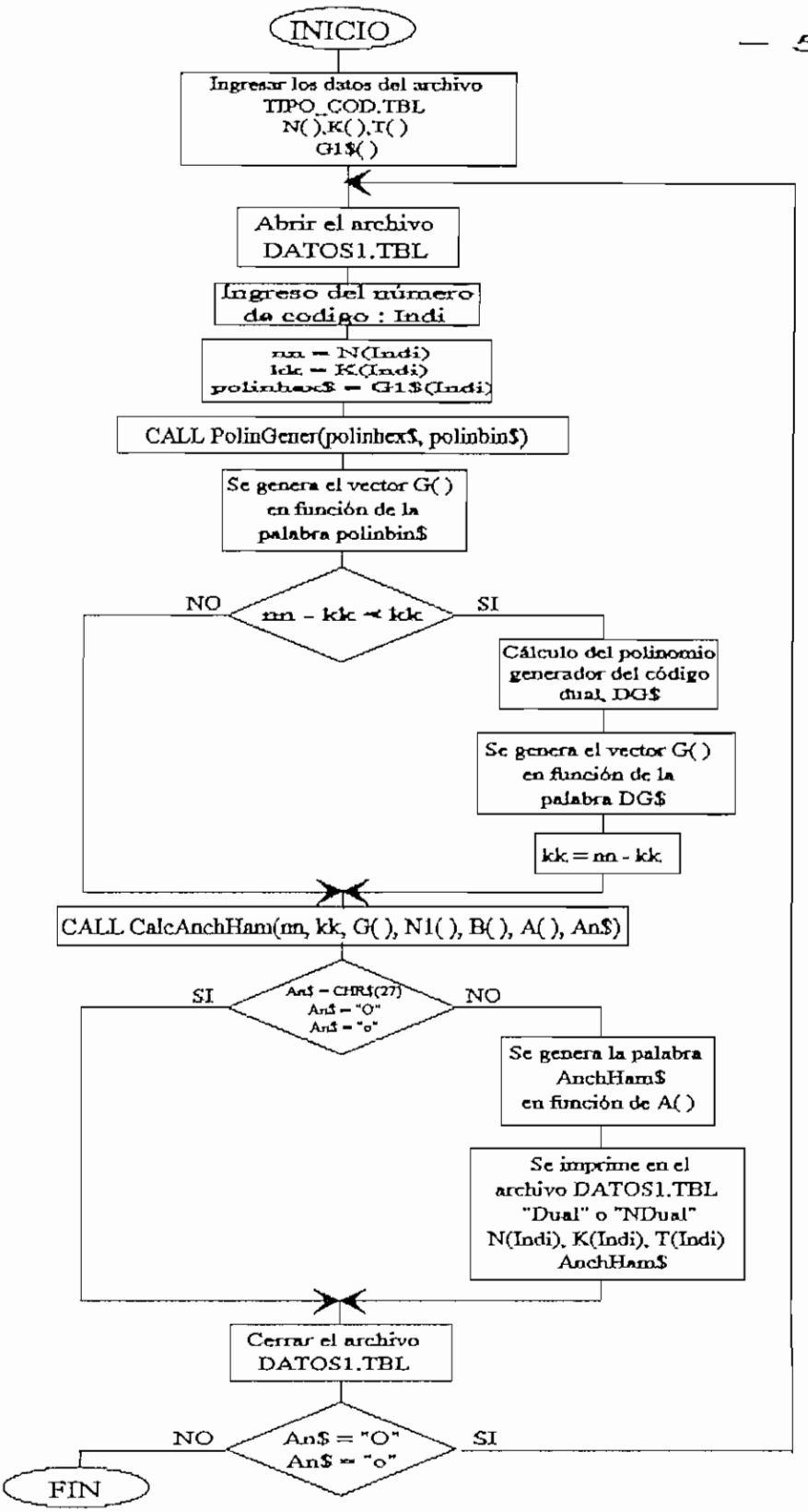
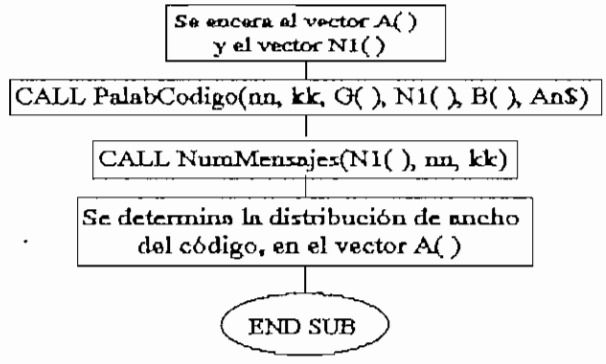
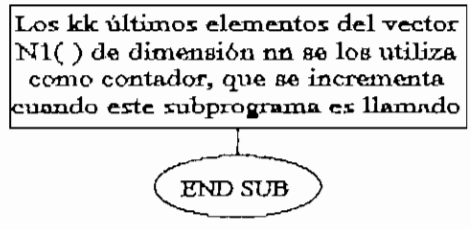


FIGURA 2.1 DIAGRAMA DE FLUJO DEL PROGRAMA QUE PERMITE CALCULAR LA DISTRIBUCION DE ANCHO DE CODIGO.

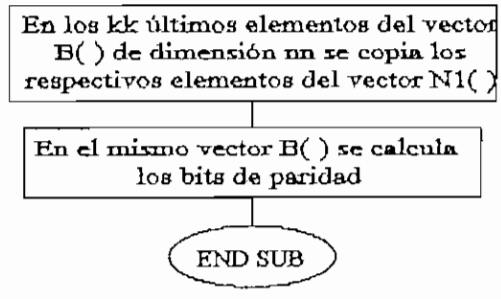
SUB CalcAnchHam(nn, kk, G( ), N1( ),B( ), A( ), An$)



SUB NumMensajes(N1( ), nn, kk)



SUB PalabCodigo(nn, kk, G( ), N1( ), B( ), An$)



SUB PolinGener(polinhex$, polinbin$)

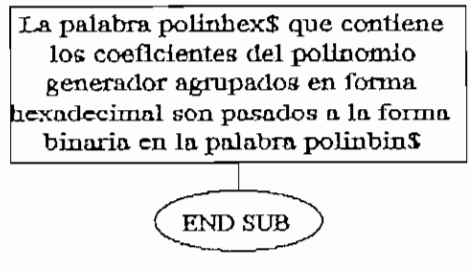


FIGURA 2.1 CONTINUACION...

Se tiene en cuenta la ubicación de los parámetros de cada código en los respectivos vectores, por ejemplo ***N(i), K(i), T(i), G1$(i)*** *donde i representa la ubicación de los parámetros* ***n, k, t*** *y los coeficientes del polinomio generador para un determinado código en el archivo* ***TIPO_COD****. Esto permite que la utilización del programa pueda interrumpirse en determinado código y luego se pueda continuar desde el siguiente código o cualquier otro.*

Se requiere de tres subprogramas principales que son:

- *SUB CalcAnchHam*
- *SUB PalabCodigo*
- *SUB NumMensajes*

cuyas características se describen a continuación.

Subprograma ***CalcAnchHam.-***

Este subprograma determina el ancho de Hamming de las palabras código y a la vez el número de palabras que tienen el mismo ancho de Hamming. Se inicia con la palabra mensaje donde sus bits son todos 0.

Utiliza los otros dos subprogramas en forma simultánea, primero el subprograma ***PalabCodigo*** *y en forma consecutiva* ***NumMensajes.***

Los resultados son almacenados en el vector A(), para luego todo el vector ser agrupado en una sola palabra, ***AnchHam$.***

Subprograma PalabCodigo.-

En este subprograma se realiza la codificación en forma sistemática de las palabras de mensaje que son generadas en el subprograma ***NumMensajes****. Es decir, utilizando el polinomio generador se calcula los bits de paridad de longitud* $n-k$*, que se juntan a los bits de mensaje.*

Subprograma NumMensajes.-

Aquí se generan todas las posibles palabras mensaje de longitud k *o* $n-k$ *para el código dual, que se van a codificar. El número total de palabras de mensaje generados es* 2^k *(*2^{n-k} *para el código dual).*

La característica principal de esta subrutina es su estructura, ya que simula un contador de bits, de esta manera se genera en forma sencilla y ordenada todas las posibles palabras de longitud k *o* $n-k$*.*

La determinación de la distribución de los anchos de Hamming se realiza en forma continua de todos los códigos que contiene el archivo TIPO_COD.TBL, pero

debido a la utilización prolongada de tiempo en computador por códigos cuya longitud de las palabras mensaje son muy largas, se ha probado para algunos códigos con valores pequeños de ***k*** *o de* ***n-k****, hasta un máximo 16 bits, con los que se obtienen resultados en un tiempo razonable. Por ejemplo para el código (63,45) el tiempo de utilización del computador 386 de 40 MHz y coprocesador matemático, supera las 24 horas.*

Cabe notarse que el programa ha sido desarrollado en una forma general para todos los Códigos Bloque Lineales, por lo que es posible su ejecución en un computador de alta velocidad de resolución.

Para los códigos BCH que corrigen 2 ó 3 errores, donde $n = 2^m - 1$*, y* ***m*** *toma los siguientes valores*[3]*:*

- *para los códigos que corrigen 2 errores:*
  - *m impar mayor o igual a 3.*
  - *m par mayor o igual a 4.*
- *para los códigos que corrigen 3 errores:*
  - *m impar mayor o igual a 5.*
  - *m par mayor o igual a 6.*

se ha desarrollado un programa, ***ANCH_BCH.BAS****, en el que se utiliza relaciones matemáticas que han sido determinadas y tabuladas para los casos anteriores que*

---

[3] ERROR CONTROL CODING, Lin Shu, pags. 177 y 178

permiten calcular la distribución de ancho de su respectivo código dual. En el Anexo I.3 se presenta el listado de este programa en QBasic.

Los resultados de los programas ***ANCH_BLQ.BAS*** *y* ***ANCH_BCH.BAS*** *son almacenados en los archivos* ***DATOS1.TBL*** *(Anexo II.2) y* ***DATOS2.TBL*** *(Anexo II.3). Utilizando el programa* ***UNIFICA.BAS*** *estos archivos son agrupados y ordenados en forma ascendente del valor de las longitudes de las palabras código* ***n*** *formándose grupos que contienen éste mismo valor y en los que también se ha realizado un ordenamiento en forma ascendente de acuerdo al numero de errores que corrige* ***t***. *El listado de este programa en QBasic se presenta en el Anexo I.4, mientras que los resultados son almacenados en el archivo* ***TIP_COD1.TBL***. *La tabla* ***2.1*** *presenta algunos resultados que están en el archivo* ***TIP_COD1.TBL***.

Se puede apreciar que para cada código se tiene un orden de almacenamiento como se indica a continuación:

- *Indicativo si se utilizó el dual "Dual" o no "NDual" para el cálculo de la distribución de anchos del código.*
- *Código bloque lineal (n, k, t) utilizado.*
- *Vector con la distribución de anchos de Hamming.*

TABLA 2.1 DISTRIBUCION DE ANCHOS DE LOS CODIGOS BLOQUE LINEALES.

```
Dual
 7            4            1
1 0 0 0 7 0 0 0

Dual
 15           11           1
1 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0

NDual
 15           7            2
1 0 0 0 0 18 30 15 15 30 18 0 0 0 0 1

NDual
 15           5            3
1 0 0 0 0 0 0 15 15 0 0 0 0 0 0 1

Dual
 31           26           1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Dual
 31           21           2
1 0 0 0 0 0 0 0 0 0 0 0 0 310 0 0 0 527 0 0 0 186 0 0 0 0 0 0 0 0 0 0 0

Dual
 31           16           3
1 0 0 0 0 0 0 0 465 0 0 0 8680 0 0 0 18259 0 0 0 5208 0 0 0 155 0 0 0 0 0 0 0

NDual
 31           11           5
1 0 0 0 0 0 0 0 0 0 0 186 310 0 0 527 527 0 0 310 186 0 0 0 0 0 0 0 0 0 0 1

NDual
 31           6            7
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 31 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

Los resultados de otros códigos que ha sido posible obtener su distribución de anchos, con cualquiera de los programas (ANCH_BLQ.BAS o ANCH_BCH.BAS), se encuentran tabulados en el Anexo II.4. Al igual que en los resultados presentados en la Tabla 2.1, se

mantienen el mismo orden de almacenamiento.

2.2.2 PROGRAMA PRINCIPAL.

Cuando ya se tiene determinado los valores de la distribución de ancho del código, se puede proceder al cálculo de la probabilidad de no detección de error Pu(E) respecto la probabilidad de transición de un canal BSC p, utilizando las relaciones generales anteriormente indicadas y considerando el canal binario simétrico que se presenta en la figura 2.2.

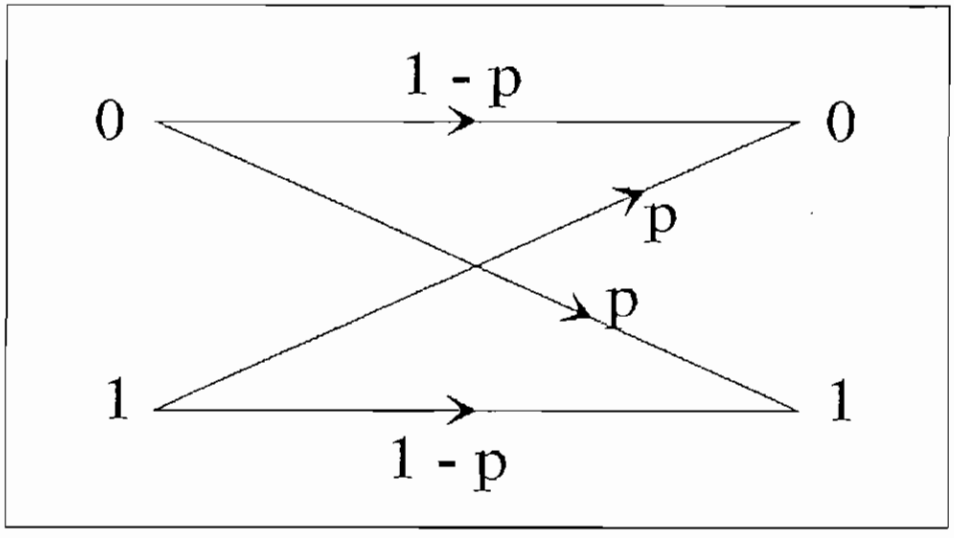


FIGURA 2.2 CANAL BINARIO SIMETRICO, BSC

Para mejor comprensión del comportamiento de Pu(E), los resultados son obtenidos en forma gráfica.

Se ha desarrollado un programa general, denominado ***PROGRAMA PRINCIPAL.*** *En este programa, se puede realizar la codificación en forma sistemática de un mensaje, para cualquier código bloque lineal, la obtención de la probabilidad de no detección de error Pu(E) respecto de* ***p*** *y la obtención de la probabilidad de error en el mensaje* ***P(E)*** *respecto de la probabilidad de error en los bits* ***p*** *(que representa el valor teórico de la BER), para el BSC de la figura 2.2. En el Anexo I.5, se detalla el listado en QBasic de éste programa.*

La figura 2.3 presenta el diagrama de flujo del programa principal, en el que se puede apreciar la utilización de los datos de los anteriores programas.

En el diagrama de flujo de la figura 2.3, el programa presenta cuatro opciones, las mismas que se encuentran distribuidas en un Menú Principal que permiten una mejor utilización.

MENU PRINCIPAL

1 --> Tipos de Códigos (n,k)

2 --> Codificación Sistemática

3 --> Gráficos

4 --> Salir del programa

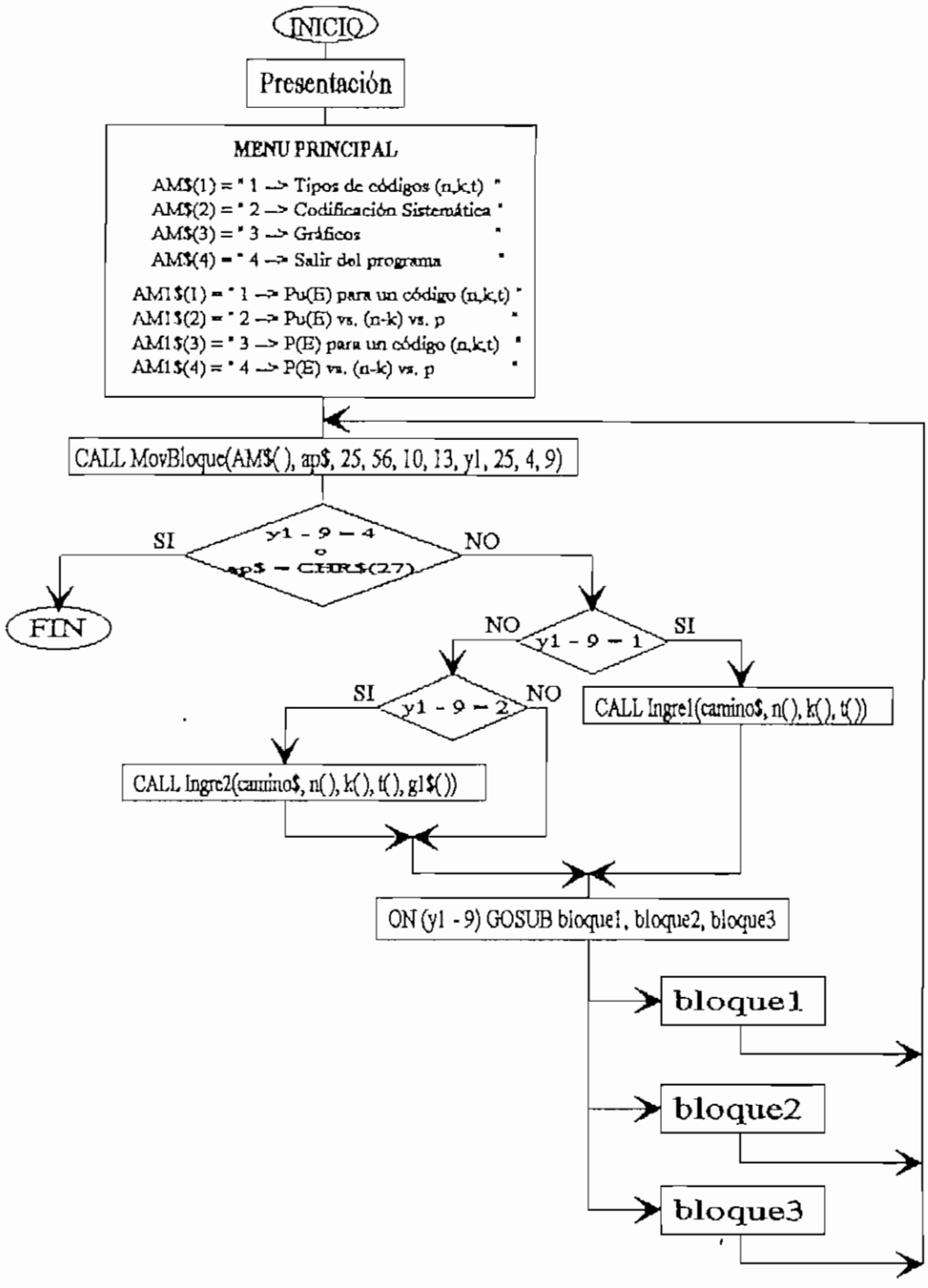
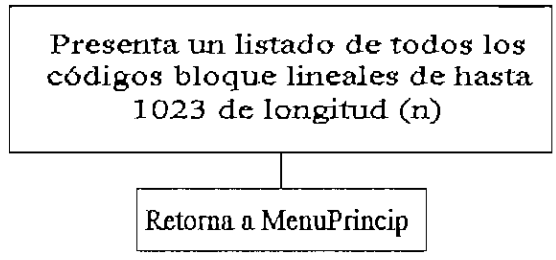
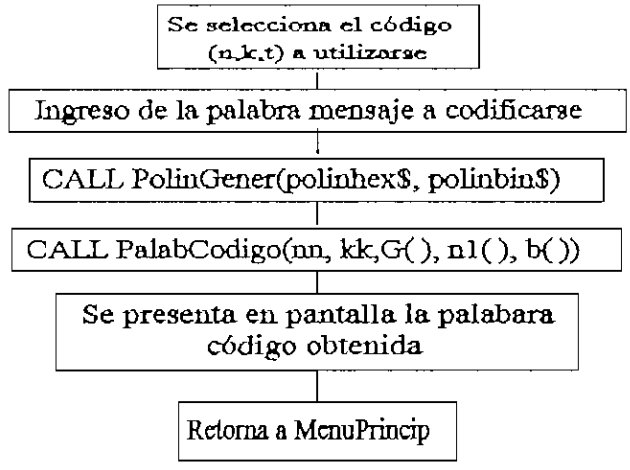


FIGURA 2.3 DIAGRAMA DE FLUJO DEL PROGRAMA PRINCIPAL.

SUBRUTINA bloque1



SUBRUTINA bloque2



SUBRUTINA bloque3

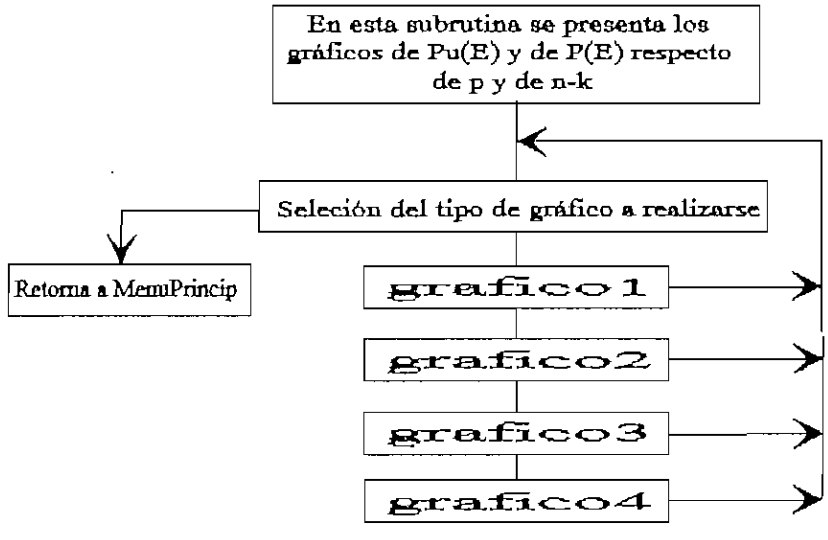
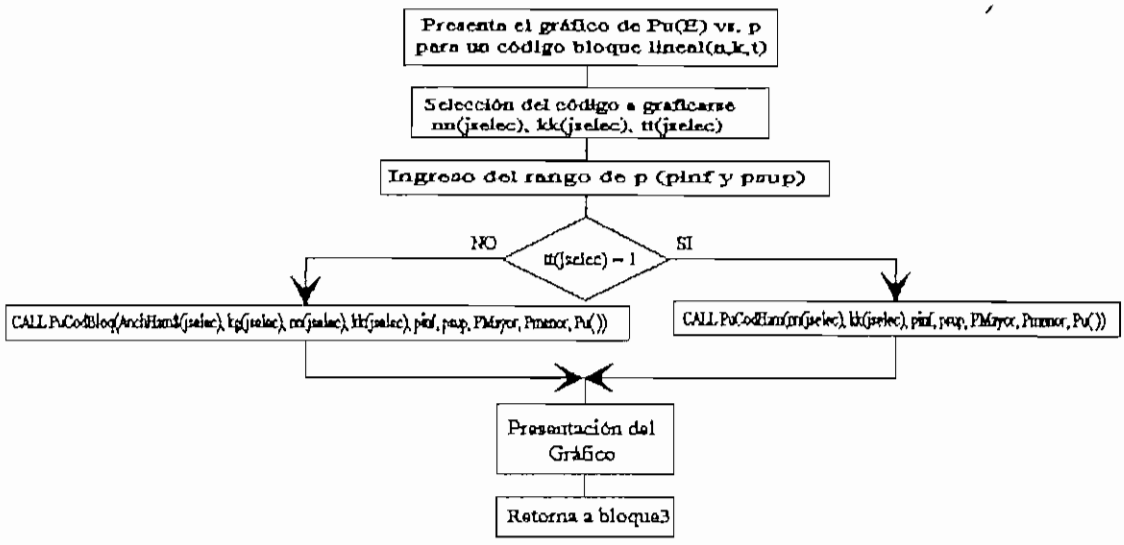
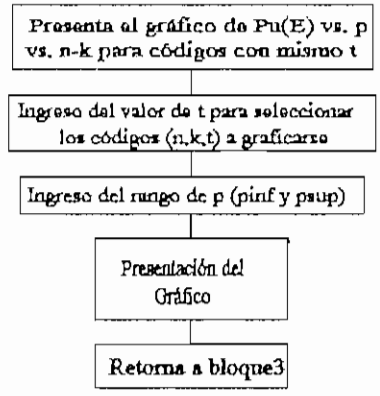


FIGURA 2.3 CONTINUACION...

SUBRUTINA grafico1



SUBRUTINA grafico2



SUBRUTINA grafico3

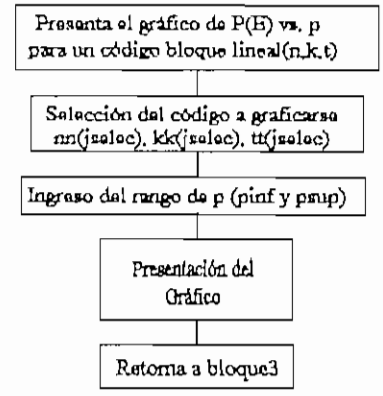
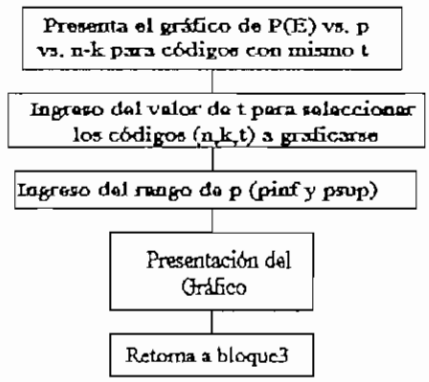
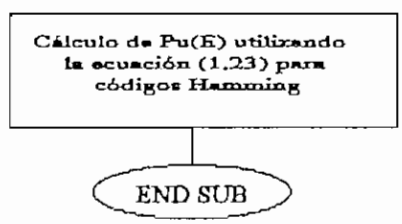


FIGURA 2.3 CONTINUACION...

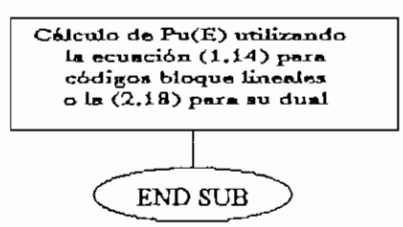
SUBRUTINA grafico4



SUB PuCodHam(nn, kk, pinf, psup, PMayor, PMenor, Pu( ))



SUB PuCodBloq(DistAnch$, dualid$, nn, kk, pinf, psup, PMayor, PMenor, Pu( ))



Los subprogramas: PolinGener( ) y PalabCodigo( ), ya se indican en la figura 2.1

FIGURA 2.3 CONTINUACION...

Las tres primeras opciones del Menú Principal representan tres subrutinas principales del programa, dejando la cuarta opción para finalizar y salir.

1 --> Tipos de Códigos (n,k)

En esta subrutina se realiza la presentación del listado de los diferentes códigos existentes, el mismo que se encuentra dividido en tres páginas y ordenados en forma ascendente de n. Cada página cuenta con cuatro columnas de códigos (n,k,t). Se debe recordar que son de longitud de hasta $n = 2^{10} - 1$

2 --> Codificación Sistemática

Realiza la codificación en forma sistemática utilizando cualquiera de los códigos enunciados en la opción anterior.

Los pasos a seguir en esta parte del programa son:

- *Se escoge el tipo de código (n, k, t) a ser utilizado.*

- *El programa requiere la palabra a codificar, indicando en pantalla el número máximo de bits (1's o 0's) que debe tener la misma de acuerdo al código.*

- *Se utiliza un subprograma denominado PalabCodigo de la que se obtiene la palabra código final y se la presenta en la pantalla.*

3 --> Gráficos

Debido a que se tiene varias opciones de gráficos, se presenta un submenú el cual ayuda a clasificarlas.

- Pu(E) para un código (n,k,t).

Se obtiene los gráficos de Pu(E) vs. la probabilidad de transición del BSC, p. Se utiliza datos previamente calculados de los anchos de Hamming (ver 2.2.1) para algunos códigos que son posibles de calcular y que se encuentran almacenados en el archivo TIP_COD1.TBL.

Estos valores de anchos de Hamming son recuperados a un solo vector denominado AnchHam$() cuya selección esta a la par con el código escogido para graficar su Pu(E) vs. p.

Una vez seleccionado el código a graficarse se presenta un submenú en el que se detalla el código escogido y requiere el ingreso del rango de probabilidad de transición del BSC, p, en el que se va a graficar es decir:

$$0 \leq pinf < psup \leq 1$$

donde:

pinf : *representa el valor de la probabilidad inicial para realizar el gráfico.*

psup : *representa el valor de la probabilidad final para realizar el gráfico.*

Ingresados estos datos se procede a calcular, para cincuenta valores de p, comprendidos en el rango, Pu(E) y que son almacenados en un vector, con el que se realiza el gráfico.

– Pu(E) vs. (n–k) vs. p.

En esta parte del programa se realiza gráficos en los que se agrupa las curvas que se obtienen de Pu(E) vs. p para los códigos que corrigen el mismo número de errores t en función del valor de n–k. Se utiliza el mismo procedimiento indicado anteriormente, pero se incluye el agrupamiento, estos gráficos se lo presenta en un sistema tridimensional.

– P(E) para un código (n,k,t).

El programa incluye también los gráficos de la probabilidad de error en la palabra mensaje P(E) como función de la probabilidad de error en los bits (probabilidad de transición del BSC) p, para un BSC.

La probabilidad de error en la palabra mensaje, P(E), en un canal binario simétrico BSC (figura 2.2), utilizando códigos bloque lineales, en forma general se calcula utilizado la siguiente expresión:

$$P(E) = \sum_{i=t+1}^{n} \binom{n}{i} p^{i} (1 - p)^{n-i}$$

(2.19)

donde:

P(E) : Probabilidad de error en la palabra mensaje.

p : Probabilidad de error en los bits (probabilidad de transición del canal binario simétrico)

n : Longitud de la palabra código.

Para efectuar estos gráficos se utiliza los valores de las combinaciones $\binom{n}{i}$ para valores de $t+1 \leq i \leq n$. En vista de que se trata de una combinación binomial, se calcula para valores de i que van desde 2 hasta el valor de la parte entera de n/2 con el programa auxiliar COMBINA.BAS. La tabla 2.2 presenta los resultados para los códigos con valores de n igual a 7, 15, 31, para los otros valores de n (63, 127, 255, 511 y 1023) incluyendo estos, son almacenados en el

archivo DATCOMBI.TBL. Este cálculo se lo ha realizado previamente para dar mayor agilidad a los resultados finales que interesa.

TABLA 2.2 VALORES DE LAS COMBINACIONES $\binom{n}{i}$

$i = 2,3,\ldots,n/2$, PARA VALORES DE n 7, 15 Y 31.

n	Valores de $\binom{n}{i}$ donde: $i = 2,3,\ldots,n/2$
7	21 35
15	105 455 1365 3003 5005 6435
31	465 4495 31465 169911 736281 2629575 7888725 20160075 44352165 84672315 141120525 206253075 265182525 300540195

Una vez seleccionado el código a graficarse se presenta un submenú en el que se detalla el código escogido y requiere el ingreso del rango de probabilidad p, en el que se va a graficar es decir:

$$0 \leq pinf < psup \leq 1$$

donde:

pinf : representa el valor de la probabilidad inicial para realizar el gráfico.

psup : representa el valor de la probabilidad final para realizar el gráfico.

Ingresados estos datos se procede a calcular, para cincuenta valores de p comprendidos en el rango, P(E) y que son almacenados en un vector, con el que se realiza el gráfico.

– P(E) vs. (n–k) vs. p.

En igual forma que la opción 2 de gráficos, esta parte del programa agrupa las curvas de P(E) vs. p para los códigos que corrigen el mismo número de errores t, en función de n–k. El procedimiento a seguirse es el mismo indicado anteriormente, pero se incluye el agrupamiento de las curvas, estos gráficos se lo presenta en un sistema tridimensional.

2.3 RELACION ENTRE Pu(E), n, k, BER.

Considerando el sistema de la figura 2.2, se pueden obtener las siguientes relaciones tanto de la probabilidad de no detección de error Pu(E), la probabilidad de error en la palabra mensaje P(E) y los parámetros de los códigos.

Para determinar la relación que existe entre la probabilidad de no detección de error Pu(E) con la longitud de la palabra código n, con la longitud de la palabra mensaje k y con la tasa de bits erróneos BER

para un canal binario simétrico BSC, utilizando códigos bloque lineales, figura 2.2. Se buscará primeramente la relación que hay entre Pu(E) con n y k, luego la relación entre BER con n y k y como influyen la codificación en los sistemas de comunicación.

Relación de Pu(E), n, k.

La probabilidad de no detección de errores para un canal binario simétrico BSC (figura 2.2), utilizando códigos bloque lineales, en forma general se calcula utilizado la ecuación (1.14):

$$P_u(E) = \sum_{i=1}^{n} A_i . p^i . (1 - p)^{n-i}$$

donde:

Pu(E): Probabilidad de no detección de errores.

A_i : Número de palabras código con ancho de Hamming i.

p : Probabilidad de transición del canal binario simétrico.

n : Longitud de la palabra código.

Mientras que para el cálculo de Pu(E) con el código dual, se utiliza la ecuación (2.18):

Como se indicó anteriormente, se puede relacionar la BER como la probabilidad de error en los bits p para el caso de que un símbolo es igual a un bit.

Para sistemas M-arios, el cálculo de la probabilidad de error en los bits se explica más detenidamente en el Capitulo 3 (punto 3.1.1).

Es necesario también conocer, en un flujo de bits codificados con algún código bloque lineal, cual es la probabilidad de error en la palabra mensaje $P(E)$.

La p esta relacionada en forma directa con la relación señal a ruido, que para el caso de transmisión digital está representado por la razón de la energía en los bits E_b respecto de la densidad espectral de ruido N_o. Al utilizar un código para corregir errores, lo que se obtiene es una variación de este valor, ya que E_b/N_o se multiplica por un factor k/n.

La relación señal a ruido para la entrada del receptor se representa por E_b/N_o que es la razón de

energía del símbolo (o bit) E_b, sobre densidad espectral del ruido N_o.

Consideremos un código (n,k) de corrección de t errores. En este caso, **k** bits de información se codifican en **n** bits. Se supondrá que se transmiten **k** bits de información en el mismo intervalo de tiempo mediante un sistema codificado y otro no codificado y que la potencia recibida P_r se mantiene igual para ambos sistemas. Ya que sólo se requieren **k** bits para ser transmitidos a través del sistema no codificado (contra **n** a través del sistema codificado), la velocidad de los bits R_b es menor para los sistemas no codificados por un factor de **k/n** en comparación con los codificados. De la ecuación (1.38) se tiene:

$$\frac{E_b}{N_0} = \frac{P_r}{R_b \cdot N_0} = \left(\frac{P_r}{N}\right)\left(\frac{B}{R_b}\right)$$

donde:

P_r : Potencia de la señal recibida.

R_b : Velocidad de transferencia de los bits.

N_0 : Densidad de ruido ($N_0 = N/B$).

Este valor es mayor para el caso no codificado en comparación con el codificado por un factor **n/k**. Esto tiende a reducir la probabilidad de error en los bits para el caso no codificado. Sean p_c y p_u las probabilidades de error en los bits en los casos codificado y no codificado, respectivamente.

Para el caso no codificado una palabra de k bits se recibirá con error si uno cualquiera de los k bits se encuentra con error. Si $P(E)_c$ y $P(E)_u$ representan las probabilidades de error en el mensaje de los sistemas codificado y no codificado respectivamente, entonces:

$$P(E)_u = 1 - P(k \text{ dígitos se reciben correctos})$$
$$= 1 - (1 - p_u)^k$$
$$\approx k.p_u \qquad p_u \ll 1$$

(2.20)

Para un código (n,k) de corrección de t errores, el mensaje recibido tendrá error si más de t errores ocurren en n bits. Si p(j,n) es la probabilidad de j errores en n bits, entonces:

$$P(E)_c = \sum_{j=t+1}^{n} p(j,n)$$

(2.21)

en virtud de que existen $\binom{n}{j}$ *formas en las cuales pueden ocurrir j errores en n dígitos.*

$$p(j,n) = \binom{n}{j} p_c^{\,j} (1 - p_c)^{n-j}$$

(2.22)

$$P(E)_c = \sum_{j=t+1}^{n} \binom{n}{j} p_c^{\,j} (1 - p_c)^{n-j}$$

Para $p_c \ll 1$, el primer término de la suma de la ecuación anterior domina a todos los demás términos, y hay justificación para ignorar a todos, excepto el primer término. Por lo tanto,

$$P(E)_c = \binom{n}{t+1} p_c^{t+1} (1 - p_c)^{n-(t+1)}$$

$$P(E)_c \approx \binom{n}{t+1} p_c^{t+1} \qquad p_c \ll 1$$

(2.23)

Para una trasmisión con modulación B-PSK, ruido aditivo blanco gaussiano y que tiene valor típico de $R_b/B \approx 1$ bit/seg/Hz, se tiene[4]*:*

$$p_u = Q\left[\sqrt{2\left(\frac{S}{N}\right)}\right] = Q\left[\sqrt{2\left(\frac{E_b}{N_o}\right)\left(\frac{R_b}{B}\right)}\right] = Q\left[\sqrt{2\left(\frac{E_b}{N_o}\right)}\right]$$

(2.24)

ya que $\frac{E_b}{N_o}$ para el caso codificado es k/n veces el caso no codificado,

$$p_c = Q\left(\sqrt{2\left(\frac{k}{n}\right)\frac{E_b}{N_o}}\right)$$

(2.25)

---

[4] Enciclopedia de la ELECTRONICA INGENIERIA Y TECNICA, tomo 6, pags 1581 – 1584.

Reemplazando (2.24) y (2.25) en las ecuaciones (2.20) y (2.23) respectivamente, se tiene:

$$P(E)_u = k\, Q\left(\sqrt{2\,\frac{E_b}{N_o}}\right)$$

$$P(E)_c = \binom{n}{t+1}\left[Q\left(\sqrt{2\left(\frac{k}{n}\right)\frac{E_b}{N_o}}\right)\right]^{t+1} \qquad p_{ec} < 1$$

Se debe tener presente que:

$$Q(x) \approx \left(\frac{1}{x\sqrt{2.\pi}}\right) e^{-x^2/2} \qquad \text{para } x > 3$$

(2.26)

representa la función de error complementaria de argumento x.

2.4 INTERPRETACION DE RESULTADOS.

Los resultados que se analizan a continuación, se los obtiene del programa descrito en el punto 2.2.

Codificación en forma sistemática.

Utilizando el código (7,4) se puede obtener todas las palabras código de las palabras mensaje que se pueden formar con cuatro bits.

TABLA 2.3 CODIFICACION SISTEMATICA DEL CODIGO (7,4,1)

Palabra mensaje	Palabra código	Ancho
0 0 0 0	0 0 0 0 0 0 0	0
0 0 0 1	1 0 1 0 0 0 1	3
0 0 1 0	1 1 1 0 0 1 0	4
0 0 1 1	0 1 0 0 0 1 1	3
0 1 0 0	0 1 1 0 1 0 0	3
0 1 0 1	1 1 0 0 1 0 1	4
0 1 1 0	1 0 0 0 1 1 0	3
0 1 1 1	0 0 1 0 1 1 1	4
1 0 0 0	1 1 0 1 0 0 0	3
1 0 0 1	0 1 1 1 0 0 1	4
1 0 1 0	0 0 1 1 0 1 0	3
1 0 1 1	1 0 0 1 0 1 1	4
1 1 0 0	1 0 1 1 1 0 0	4
1 1 0 1	0 0 0 1 1 0 1	3
1 1 1 0	0 1 0 1 1 1 0	4
1 1 1 1	1 1 1 1 1 1 1	7

De igual manera se puede utilizar los otros códigos.

Para los códigos que no tienen una relación matemática directa para obtener la distribución de anchos del código, se utiliza estos resultados. Para el código (7,4,1) presentado en la tabla 2.3, se determina el ancho de cada palabra código contabilizando el número de 1's que tiene cada una y luego se determina el número de palabra código que tienen el mismo ancho, obteniéndose:

$A_0 = 1$

$A_1 = A_2 = A_5 = A_6 = 0$

$A_3 = A_4 = 7$

$A_7 = 1$

Se puede apreciar que en la palabra código, en forma sistemática, se mantiene la palabra mensaje y que los otros bits restantes (n − k) son los de paridad.

Gráficos.

Los gráficos obtenidos corresponden a un canal BSC, que se indica en la figura 2.2. Teniendo en cuenta la capacidad de éste canal BSC que se determina con la ecuación:

$$C = 1 + p \log_2 p + (1 - p) \log_2 (1 - p) \qquad (2.27)$$

donde:

C : Capacidad del canal BSC, bits por símbolo.

p : Probabilidad de transición del BSC.

Si analizamos esta ecuación, podemos notar que cuando p = 1/2, un símbolo de entrada genera cualquiera de las salidas con igual probabilidad, y la capacidad es cero. Mientras que si p = 0 o p = 1, la salida del canal está completamente determinada por la entrada, y la capacidad es 1 bit por símbolo.

La figura 2.4a presenta el gráfico de la probabilidad de no detección de errores Pu(E) en función de la probabilidad de transición del BSC, p, para los valores de 0 ≤ p ≤ 1 del código (7,4) que corrige un

solo error, t = 1. Como se puede apreciar en el gráfico, los valores Pu(E) aumentan en una forma lenta hasta un valor aproximado de p = 0.65, a partir del cual la variación es más pronunciada produciéndose que la probabilidad de que el código no detecte errores aumente cuando el valor de la probabilidad de transición del BSC, p, esta próximo a 1. Ya que como se indicó en la capacidad del canal, cuando el valor de p es 0 o 1 la salida del canal esta determinada. Lo que nos indica que se tomará la salida como verdadera aunque sea una respuesta errónea. Se debe tener presente que una palabra código, por causa de los errores, puede transformarse en otra palabra código. Si el valor de p es 1 lo suficiente para cambiar una palabra código.

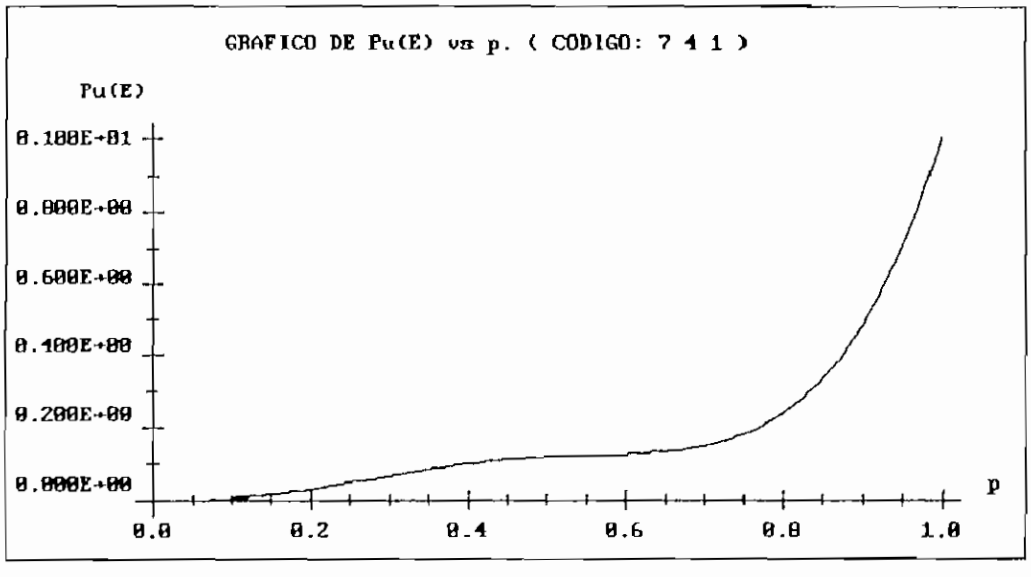
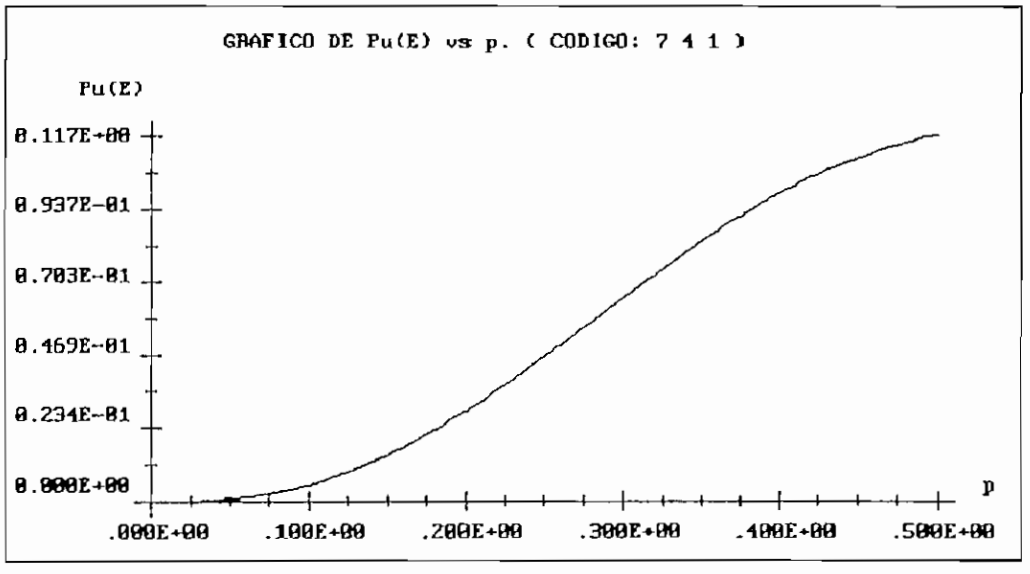
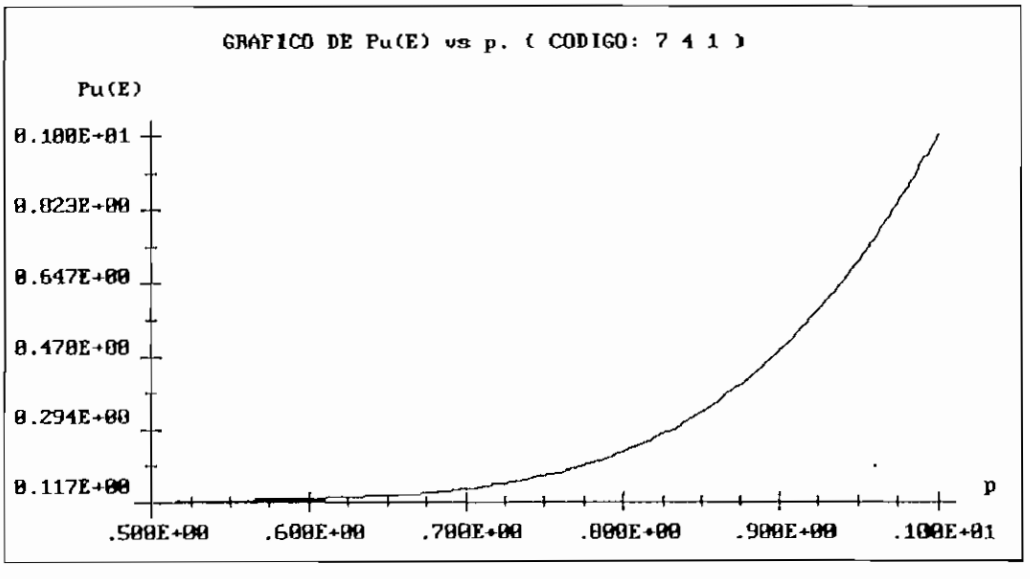


FIGURA 2.4a



(b)



(c)

FIGURA 2.4 Pu(E) vs. p PARA EL CODIGO (7,4)

(a) $0 \leq p \leq 1$

(b) $0 \leq p \leq 0.5$

(c) $0.5 \leq p \leq 1$

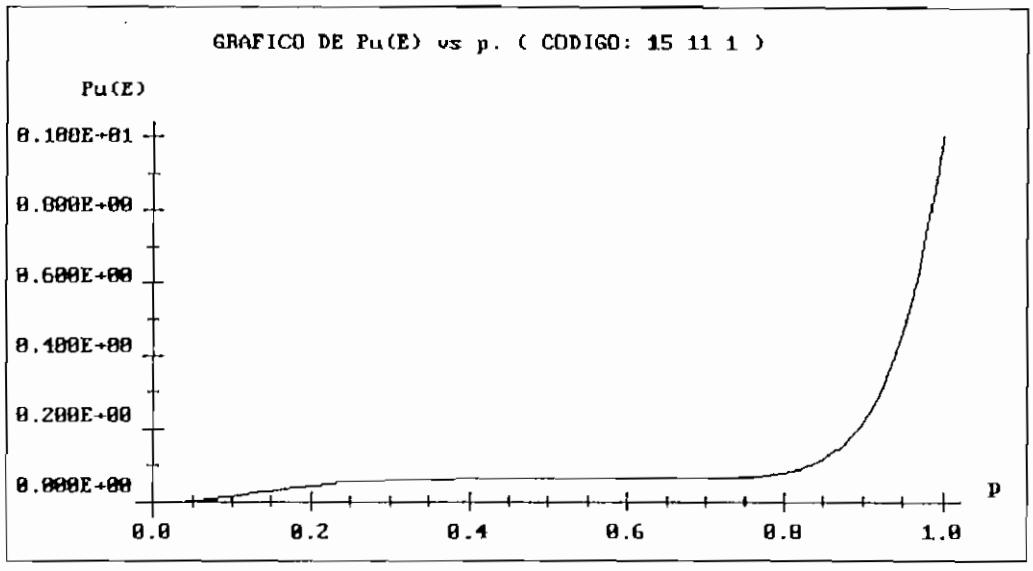
Las figuras 2.4b y 2.4c, se refieren al mismo código pero los rangos de graficación son diferentes. Para la figura 2.4b se tiene $0 \leq p \leq 0.5$ y para la figura 2.4c se tiene $0.5 \leq p \leq 1$. Permitiéndonos tener una visión más amplia de la variación de Pu(E) respecto de p.

Incrementando el número de bits de paridad n-k y el número de bits del mensaje k, por ejemplo el código (15,11) de la figura 2.5, n a 15 bits y k a 11 bits con t = 1, se tiene que la Pu(E) disminuye y es de un valor mucho menor y casi estable en un amplio rango de p. La figura 2.5a presenta la variación de Pu(E) respecto de p para un rango comprendido entre 0 y 1 del código (15,11).

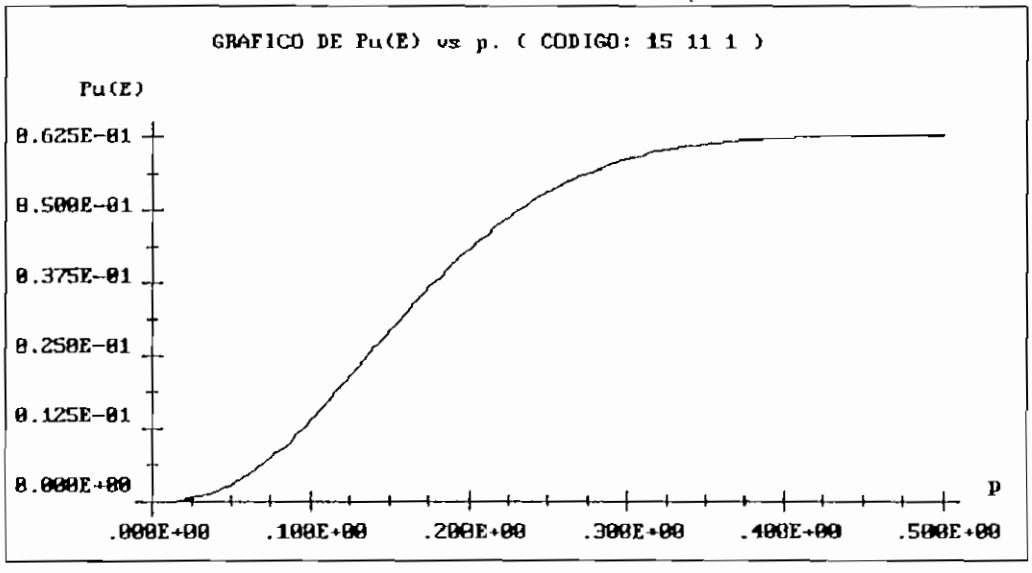
Se puede decir que este código es más confiable mientras el valor de p no este cercano a 1.

Las figuras 2.5b, 2.5c y 2.5d presentan, para el mismo código (15,11), diferentes rangos de p.

Si utilizamos el código (15,7), figura 2.6, que corrige dos errores t = 2, el valor de Pu(E) disminuye en forma considerable si analizamos los mismos rangos de graficación del la figura 2.5. En este caso se disminuye la longitud de la palabra mensaje k a 7 bits



(a)



(b)

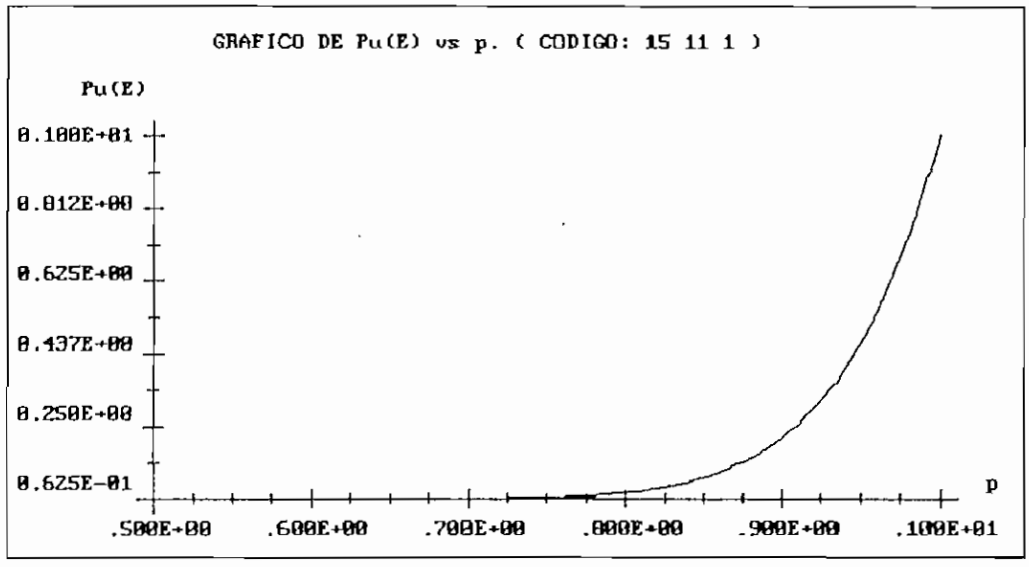
FIGURA 2.5 Pu(E) vs. p PARA EL CODIGO (15,11)

(a) $0 \leq p \leq 1$

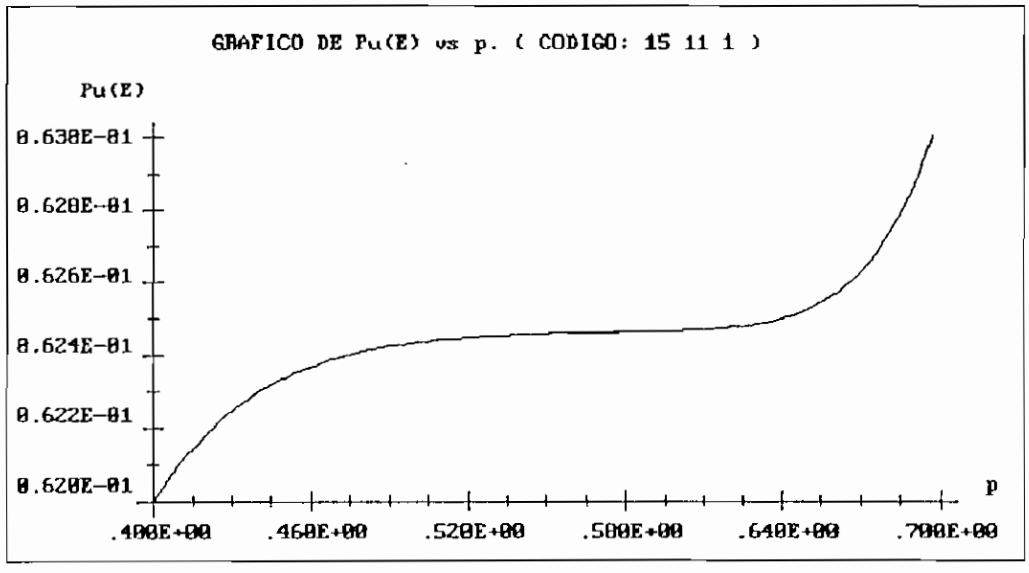
(b) $0 \leq p \leq 0.5$

(c) $0.5 \leq p \leq 1$

(d) $0.4 \leq p \leq 0.7$



(c)



(d)

FIGURA 2.5 CONTINUACION...

y manteniendo la longitud de la palabra código n en 15 se tiene que n−k aumenta a 8. Por lo tanto en canales

binario simétricos BSC, si se tiene un grupo de códigos que tienen la misma longitud de la palabra código n y se desea disminuir la Pu(E) se debe disminuir la longitud de la palabra mensaje k y por consiguiente aumenta el número de bits de paridad n−k.

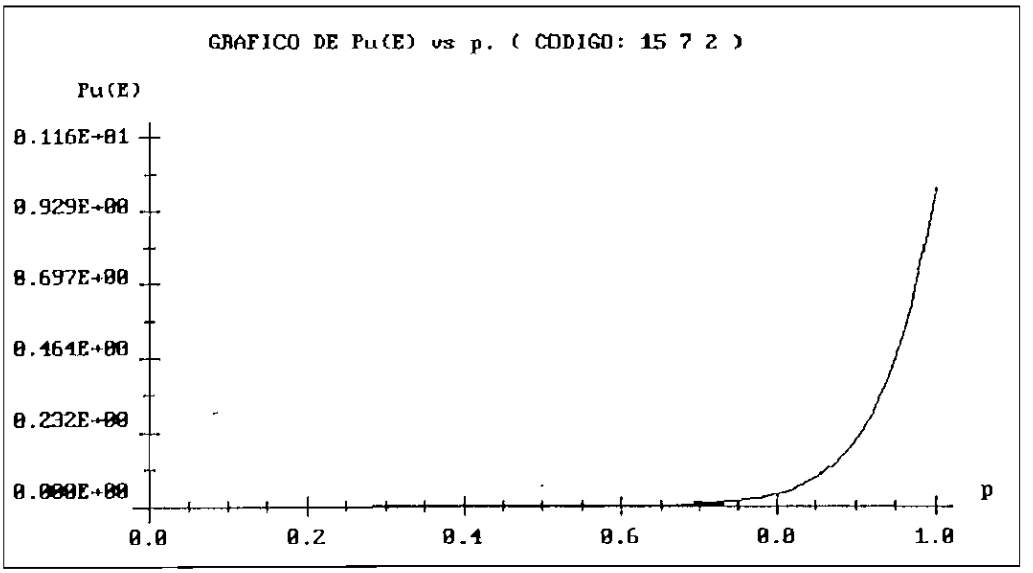


FIGURA 2.6a

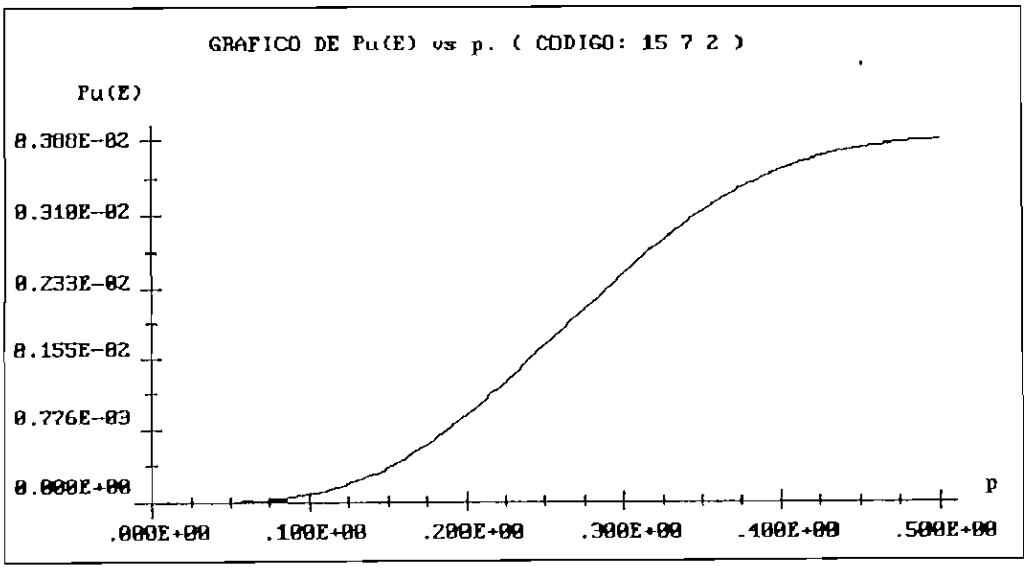
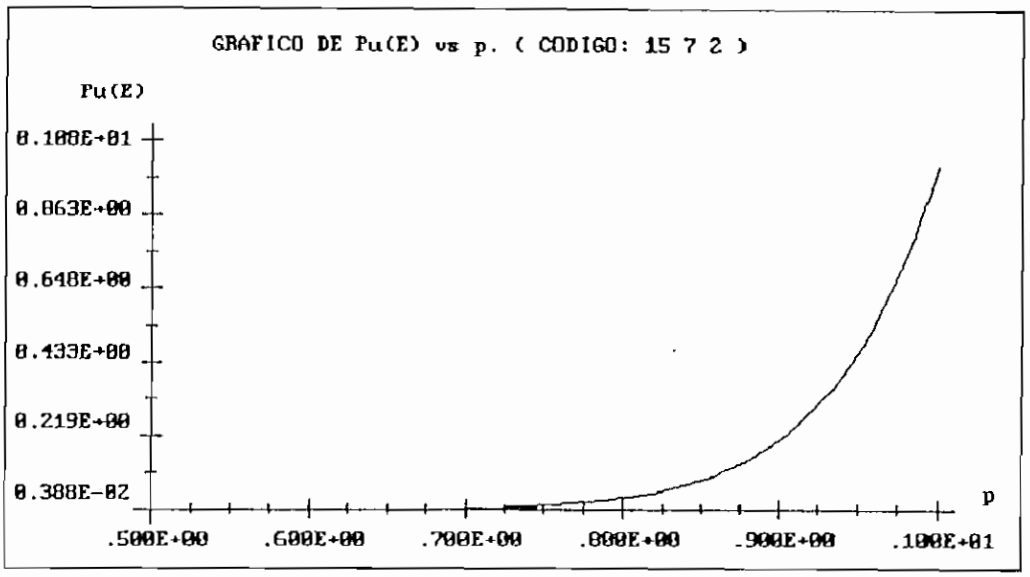
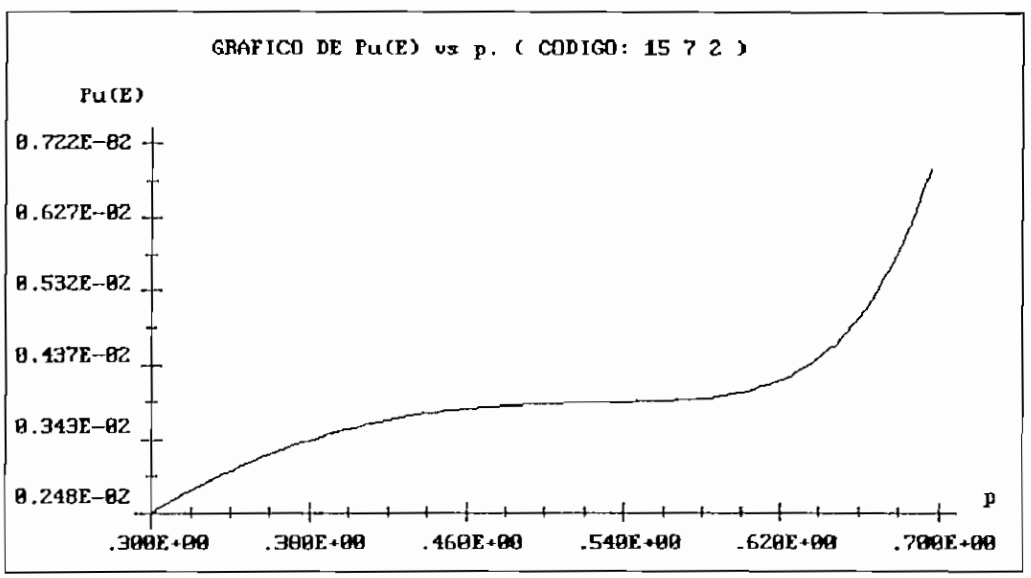


FIGURA 2.6b



(c)



(d)

FIGURA 2.6 Pu(E) vs. p PARA EL CODIGO (15,7)

(a) $0 \leq p \leq 1$

(b) $0 \leq p \leq 0.5$

(c) $0.5 \leq p \leq 1$

(d) $0.4 \leq p \leq 0.7$

Para valores inferiores de p *no se cumple estrictamente lo indicado anteriormente. Por ejemplo, para los códigos con* $t = 1$ *y con valores de* $p < 0.3$*, se tiene que al cambiar del código (7,4) al (15,11) la Pu(E) aumenta y de igual manera con los otros códigos en otros rangos, figura 2.7. En los puntos de cruce entre las curvas, se tiene que para un valor determinado de* p*, los códigos correspondientes a estas curvas dan el mismo valor de Pu(E).*

Para los códigos con $t = 2$ *ésta variación se tiene cuando* $p < 0.2$*, figura 2.9. Mientras que para los códigos con* $t = 3$ *cuando* $p < 0.16$*, figura 2.10.*

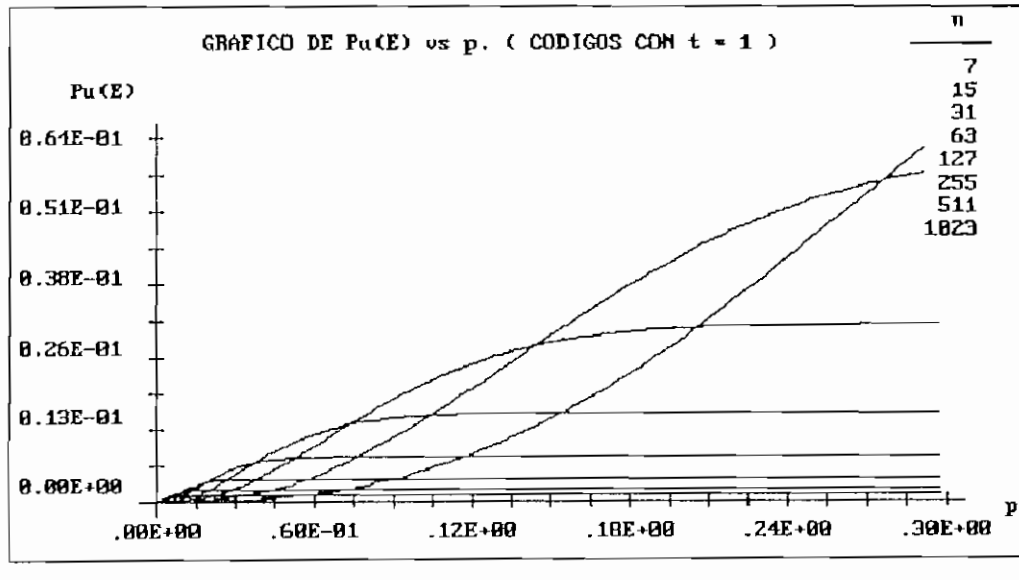


FIGURA 2.7 CODIGOS BLOQUE LINEALES CON $t = 1$

$0 \leq p \leq 0.3$

Para comprender mejor lo indicado anteriormente, se ha realizado gráficos en los que se presenta la variación de Pu(E) vs n−k vs p para códigos bloque lineales en un canal BSC.

Para los códigos con t = 1 se tiene la figura 2.8 en la que se puede apreciar como varía Pu(E) a medida que se incrementa la longitud de la palabra mensaje k y el número de bits de paridad n−k, para un rango de p igual al utilizado en la figura 2.7 (0 ≤ p ≤ 0.3). Los códigos graficados son desde el código (7,4) hasta el código (1023,1013), teniendo en cuenta que t = 1 .

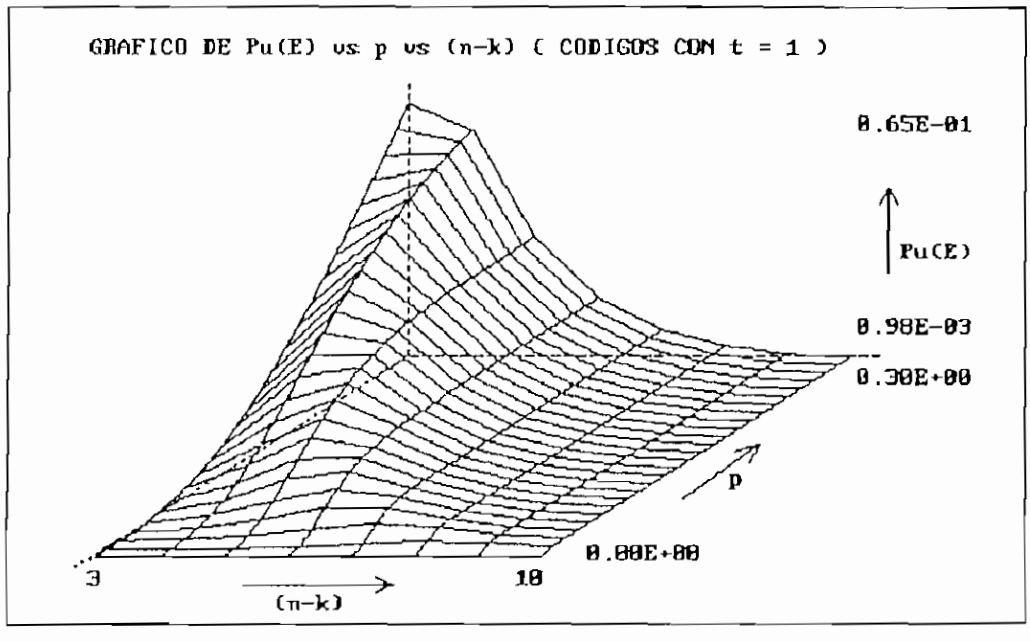


FIGURA 2.8 VARIACION DE Pu(E) vs p vs n−k

CODIGOS CON t = 1 Y 0 ≤ p ≤ 0.3

Se realiza una agrupación de curvas que se pueden obtener utilizando el programa desarrollado en 2.2, y graficándolas una a continuación de otra en función del número de bits de paridad n−k.

Las figuras 2.9 y 2.10 presentan la variación de Pu(E) para los códigos con t = 2 y con t = 3, respectivamente. En los rangos de p que han sido realizadas estas figuras (0 − 0.2 para la figura 2.9 y 0 − 0.16 para la figura 2.10) se puede apreciar como va cambiando el comportamiento de Pu(E) al aumentar n−k para ciertos valores de p.

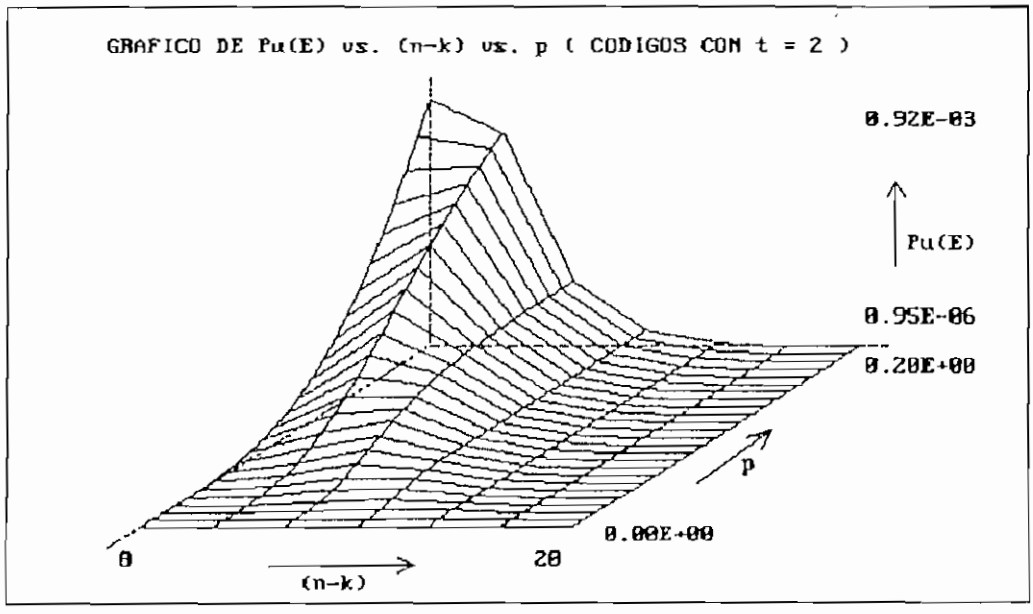


FIGURA 2.9 CODIGOS BLOQUE LINEALES CON t = 2

0 ≤ p ≤ 0.2

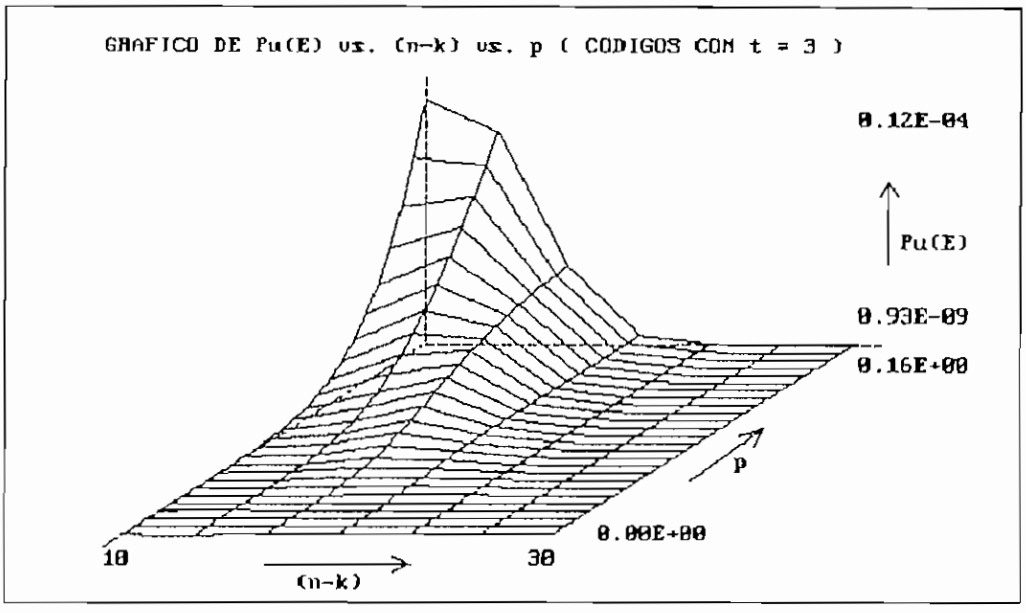


FIGURA 2.10 CODIGOS BLOQUE LINEALES CON t = 3

0 ≤ p ≤ 0.16

Como resultados, también se obtienen gráficos de la probabilidad de error en la palabra mensaje P(E). Si se transmite un flujo de bits que ha sido agrupado en palabras mensaje de longitud k y que estas a su vez han sido codificadas utilizando cualquiera de los códigos bloque lineales, a través de un canal binario simétrico BSC, se puede obtener la probabilidad de error en la palabras mensaje P(E) en función de la probabilidad de error en los bits p (probabilidad de transición) que tenga este canal. Teniendo en cuenta que la probabilidad de error en los bits p equivale a la tasa de bits erróneos BER.

En el submenú de gráficos del programa principal (ver punto 2.2.2) se presenta esta opción. La figura 2.11 presenta la variación de P(E) en función de p en el rango de 0 a 1 para el código (7,4). Mientras que la figura 2.12 para el rango de 0 a 10^{-3}.

Los equipos de comunicación normalmente incluyen alarmas para detectar una BER de 10^{-6} y de 10^{-3}. Por esta razón se realiza los gráficos con la variación de p hasta 10^{-3}. Estos gráficos nos permiten analizar los errores que se comenten en la palabra mensaje en función de la probabilidad de error en los bits p (que aquí representa la BER).

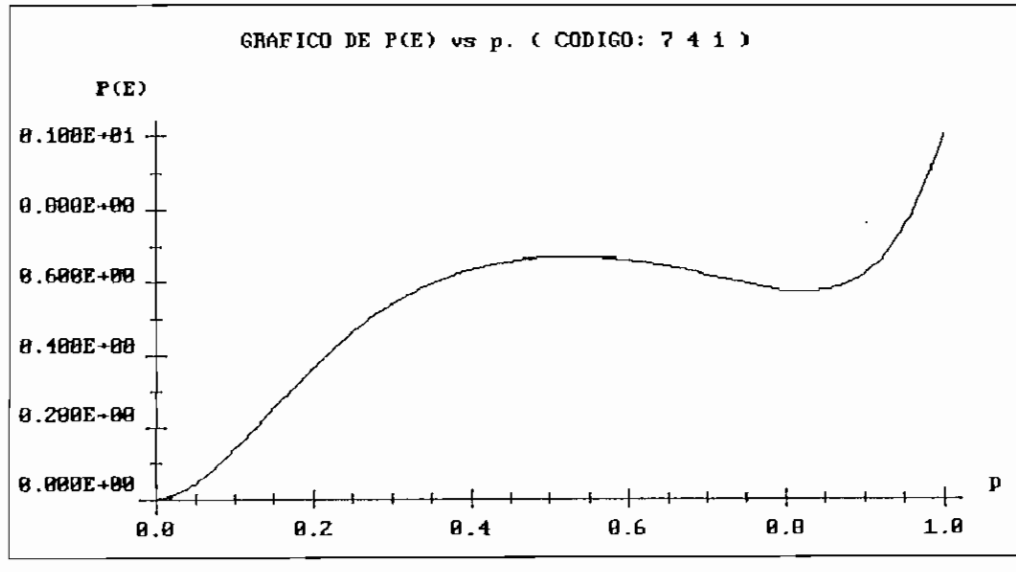


FIGURA 2.11 P(E) vs. p PARA EL CODIGO (7,4)

$0 \leq p \leq 1$

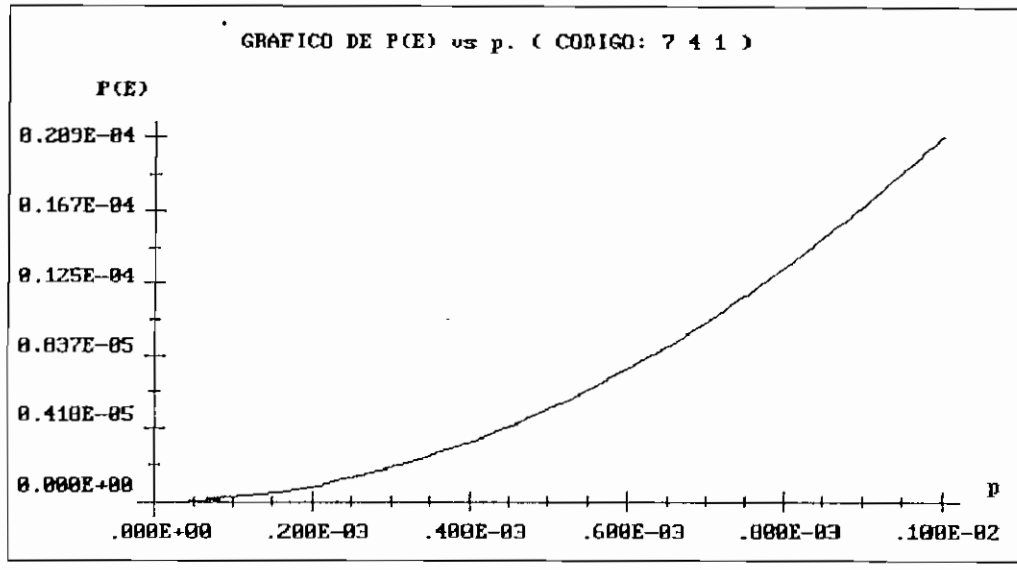


FIGURA 2.12 P(E) vs. p PARA EL CODIGO (7,4)

$0 \leq p \leq 10^{-3}$

En el punto 2.3 de este capítulo, se presenta como influye el tipo de código bloque lineal (n,k) en el cálculo de ***p****, además de otros parámetros como la potencia de transmisión, ruido, tipo de modulación y velocidad de trasmisión de los bits, entre otros.*

La figura 2.13 presenta la variación de la probabilidad de error en los bits, ***p*** *(probabilidad de transición del BSC y que también representa el valor teórico de la BER), en función de* E_b/N_o *para los códigos que corrigen un sólo error* $t = 1$*. Este gráfico esta realizado bajo las siguientes condiciones: que el canal tiene ruido aditivo blanco gaussiano y también*

que se utiliza modulación BPSK. El rango de variación de E_b/N_o es de -8 a 15 dB. Para realizar este figura se utiliza la ecuación 2.25 que la repetimos:

$$p = Q\left(\sqrt{2\left(\frac{k}{n}\right)\frac{E_b}{N_o}}\right) \qquad \textit{Sistema Codificado}$$

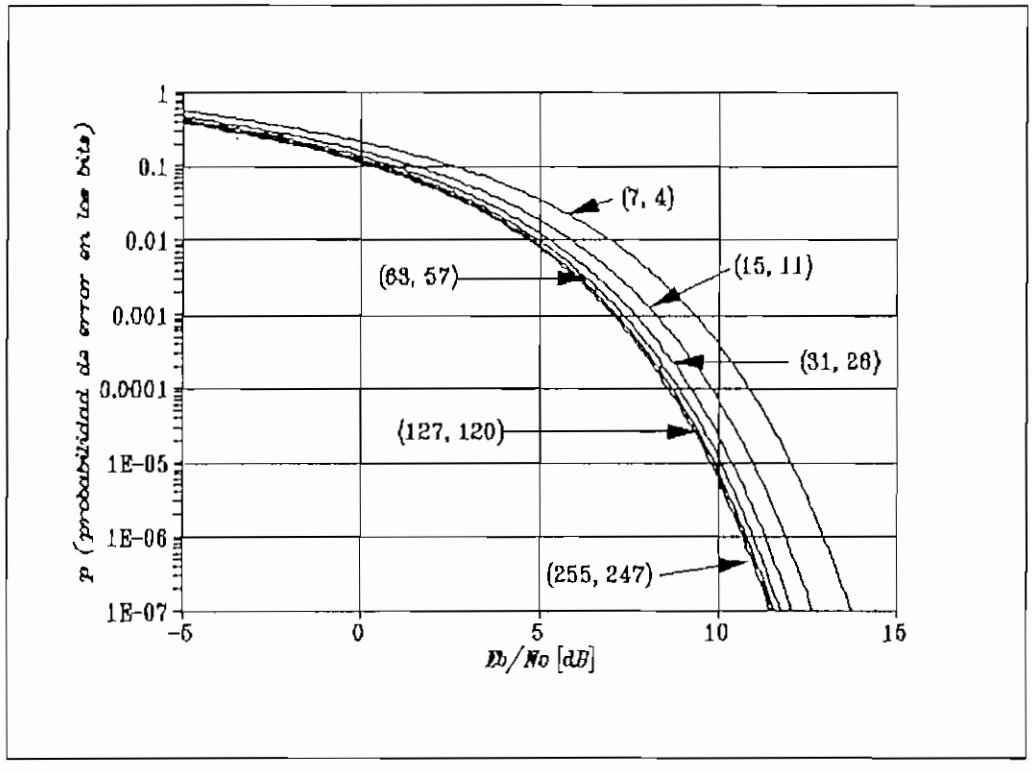


FIGURA 2.13 PROBABILIDAD DE ERROR EN LOS BITS EN FUNCION DE E_b/N_o PARA SISTEMAS CODIFICADOS CON CODIGOS BLOQUE LINEALES Y MODULACION B-PSK.

La figura 2.14 presenta las curvas de **p** en función de

E_b/N_o para el caso no codificado manteniendo las mismas condiciones y rango. Se utiliza la ecuación 2.24 que se repite a continuación:

$$p = Q\left(\sqrt{2\,\frac{E_b}{N_o}}\right) \qquad \textit{Sistema no Codificado}$$

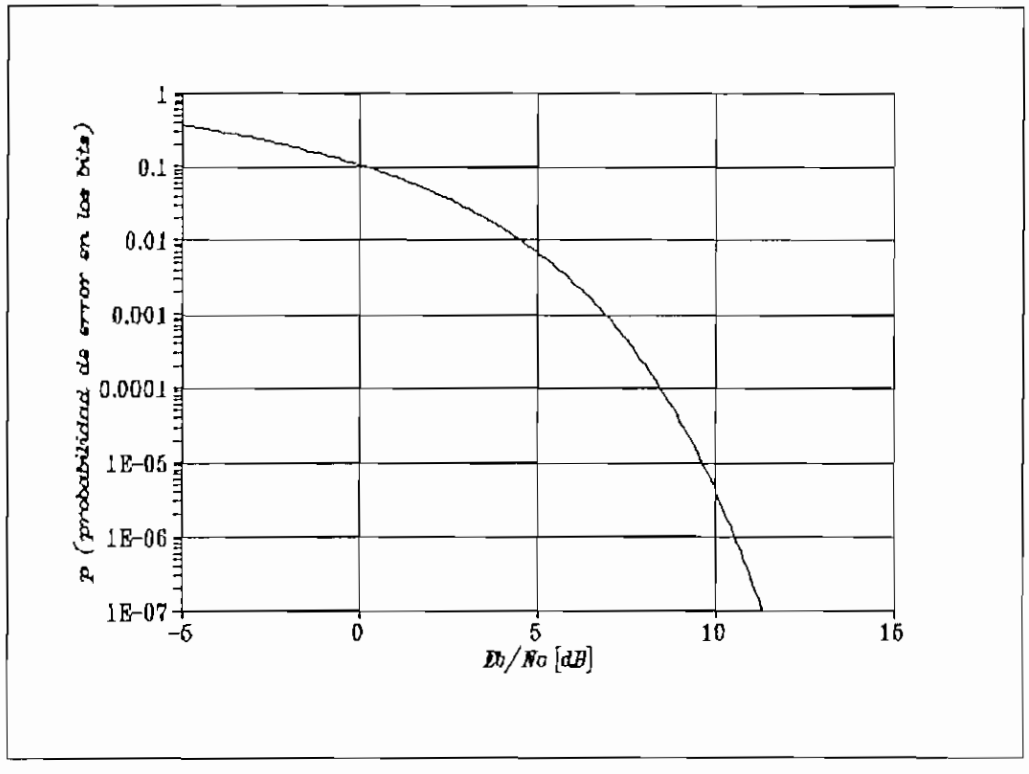


FIGURA 2.14 PROBABILIDAD DE ERROR EN LOS BITS EN FUNCION DE E_b/N_o PARA SISTEMAS NO CODIFICADOS CON MODULACION B-PSK.

Si se compara entre las figuras 2.13 y 2.14, se puede ver que la probabilidad de error en los bits del

sistema codificado se ha degradado, por lo que debe ser posible detectar más bits durante el mismo intervalo y con la misma potencia disponible. La mejora en el rendimiento se ve en el cálculo de la probabilidad de error en la palabra mensaje, P(E).

Por lo tanto, si se quiere mejora el rendimiento se puede seleccionar el código bloque lineal (n,k) que satisfaga los requerimientos de comunicación. Con el valor de E_b/N_o y determinado el tipo de código a utilizarse, se puede calcular la probabilidad de no detección de errores, Pu(E).

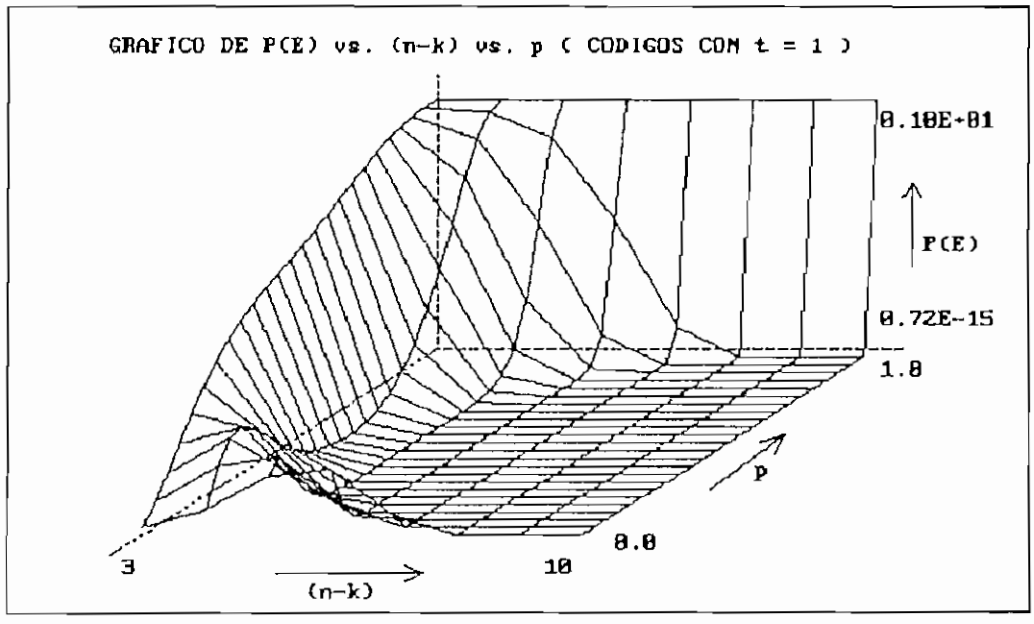


FIGURA 2.15 CODIGOS BLOQUE LINEALES CON t = 2

$0 \leq p \leq 1$

La figura 2.15 presenta la variación de P(E) en función de n−k y de p, para los códigos bloque lineales que corrigen un solo error, t = 1, se puede ver que a medida que se incrementa el valor de n−k y se disminuye p el valor de P(E) decrece, obteniéndose un mejor rendimiento de las palabras código a la salida del canal.

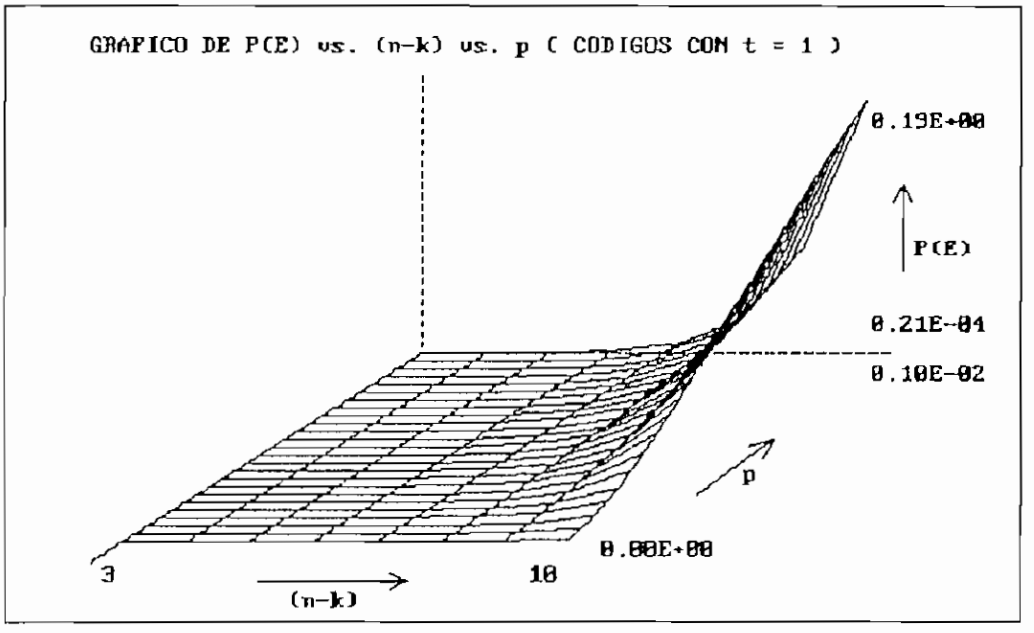


FIGURA 2.16 CODIGOS BLOQUE LINEALES CON t = 2

$0 \leq p \leq 10^{-3}$

Al igual que la probabilidad de no detección de error Pu(E), la probabilidad de error en la palabras mensaje P(E), para valores de p pequeños (próximos a cero) se tiene un comportamiento contrario al indicado

anteriormente. P(E) aumenta al aumentar n−k para un determinado valor de p. Figura 2.16.

Con la opción de gráficos del programa principal descrito en 2.2, se puede realizar los mismos análisis hechos en este punto, con los otros códigos bloque lineales.

CAPITULO III.

RELACION ENTRE LA BER, BITS DE CONTROL DE LOS CODIGOS BLOQUE LINEALES (n-k) Y LA POTENCIA DE TRANSMISION.

A partir de mediados de los años setenta se empezaron a introducir centrales de conmutación digitales, la transmisión analógica dejó de ser la solución más económica. Por ese motivo fue desarrollada una primera generación de equipos de radioenlace digitales, que

utilizaban preferentemente las modulaciones Q-PSK y 16-QAM.

Sólo en casos especiales, o sea cuando la capacidad de transmisión de los equipos de radioenlace digitales de primera generación no era suficiente se continuaron implementando sistemas SSB-AM en conjunto con transmultiplexores.

En los años venideros, la decisión de desarrollar la red de telecomunicaciones digitales existentes, que son un conjunto de redes dedicadas a servicios especiales, hacia una red de servicios digitales integrados (RSDI), revestirá una gran importancia.

3.1 MODULACION PSK Y QAM MULTINIVEL.

A. MODULACION PSK.

La modulación por desplazamiento de fase PSK (Phase Shift Keying) es un caso especial de la modulación de fase PM (Phase Modulation), la cual es posible generar una señal PSK con un modulador lineal de producto configurado para modulación DSBSC (doble banda lateral con portadora suprimida). El espectro de la señal modulada resultante es sencillamente una traslación en

frecuencia del espectro de información digital a la banda de frecuencia portadora en RF.

En la modulación PSK binaria (BPSK), los dos pulsos básicos que se tiene en cuenta son $p'(t)\cos(w_c t)$ *y*

$p'(t)\cos(w_c t + \pi)$.

Se generaliza la idea para el caso M-ario (M-PSK) utilizando M pulsos con el k-ésimo pulso $p'(t)\cos\left[w_c t + \left(2\frac{\pi}{M}\right)k\right]$. *De esta forma, las fases de pulsos sucesivos son cada* $2\frac{\pi}{M}$ *radianes, como se puede apreciar en la figura 3.1.*

Por la simetría de este sistema es evidente que si, debido al ruido de canal, la fase de cualquier pulso se desvía más de $\frac{\pi}{M}$ *radianes, se comete error.*

Por lo tanto, la probabilidad de error en los símbolos p_s *es la probabilidad de que la fase de cualquier pulso se desvíe más de* $\frac{\pi}{M}$ *radianes.*

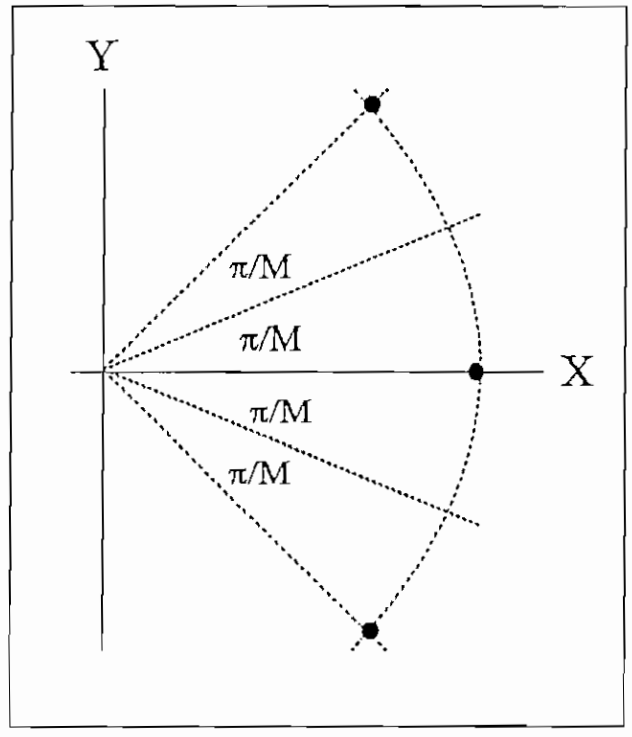


FIGURA 3.1 DIAGRAMA FASORIAL DE SEÑALES M-PSK

En el receptor se dispone de una referencia de fase coherente. La función básica del receptor es detectar la fase del pulso que se recibe. Esto puede hacerse mediante dos detectores de fase con referencias respectivas $\cos(\omega_c t)$ *y* $\text{sen}(\omega_c t)$*, junto con un circuito lógico para determinar la relación entre las dos componentes detectadas.*

B. MODULACION QAM.

Las señales DBL ocupan el doble del ancho de banda que se requiere para las señales BLU. Esta desventaja se puede salvar transmitiendo dos señales DBL utilizando portadoras de la misma frecuencia pero en cuadratura de fase (figura 3.2a).

Las dos señales moduladas ocupan el mismo ancho de banda. Las dos señales de banda base pueden separarse en el receptor mediante detección sincronizada utilizando dos portadoras locales en cuadratura de fase. Esto se puede demostrar considerando la salida del multiplicador $X_1(t)$ del canal 1 (figura 3.2a):

$$X_1(t) = 2\,[m_1(t)\,\cos(w_c t) + m_2(t)\,\sin(w_c t)]\,\cos(w_c t)$$

$$X_1(t) = m_1(t) + m_1(t)\,\cos(2w_c t) + m_2(t)\,\sin(2w_c t)$$

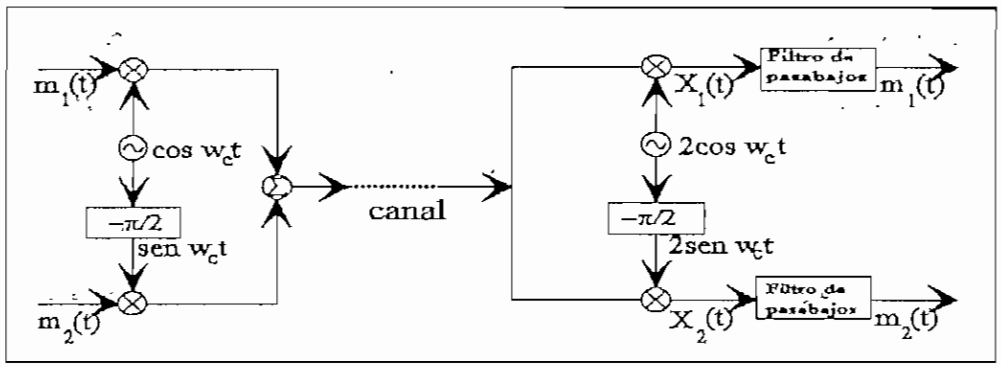
donde:

$m_1(t)$ y $m_2(t)$: *Datos de entrada al modulador.*

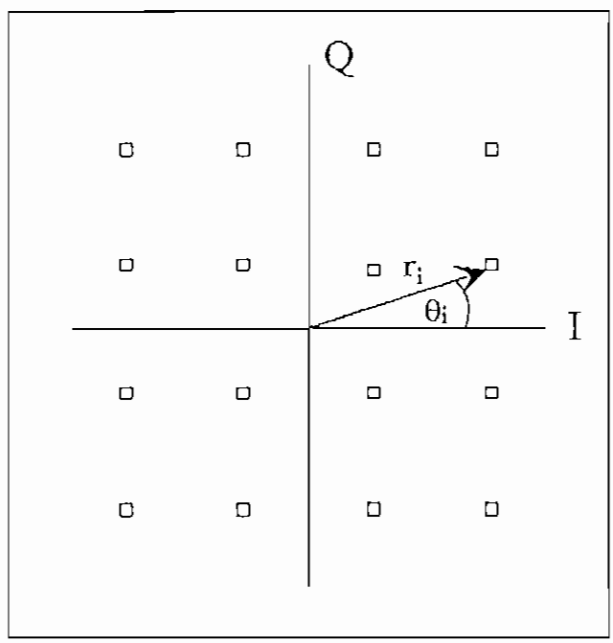
w_c : *Frecuencia de la portadora.*

Los dos últimos términos son suprimidos por el filtro de pasabajos, dando por resultado la salida deseada $m_1(t)$. En forma similar, se puede demostrar que la

salida del canal 2 es $m_2(t)$. Este esquema se conoce como modulación de amplitud en cuadratura, (QAM), o multiplexión en cuadratura.



(a)



(b)

FIGURA 3.2 (a) SISTEMA QAM O MULTIPLEXION DE CUADRATURA

(b) QAM DE 16 PUNTOS (M = 16)

señal deseada en el canal 2 y una señal de interferencia en el canal 1. Podemos aumentar el índice de transmisión aun más mediante el uso de QAM M-aria. Un caso práctico con $M = 16$ utiliza los 16 pulsos (16 símbolos) siguientes.

$$p_i(t) = r_i p(t) \cos(w_c t + \theta_i) \qquad i = 1,2,\ldots,16$$

en donde $p(t)$ es un pulso de banda base con forma apropiada. Las elecciones de r_i y θ_i para 16 pulsos se muestran gráficamente en la figura (3.2 b).

Ya que $M = 16$, cada pulso puede transmitir la información de $\log_2 16 = 4$ dígitos binarios. Esto puede hacerse en la forma siguiente: existen 16 sucesiones posibles de cuatro dígitos binarios. Por lo tanto, un sólo pulso $r_i p(t)\cos(w_c t + \theta_i)$ transmite cuatro bits. De esta manera, una línea de transmisión normal que puede transmitir 2400 bits/seg podrá ahora transmitir a 9600 bits/seg.

- <u>Sistemas de radioenlace digitales de primera generación.</u>

Los sistemas digitales de primera generación trabajan preferentemente con las modulaciones Q-PSK y 16-QAM. El aprovechamiento de banda no se aproxima a los valores de los sistemas analógicos, ni siquiera a los

modulados en frecuencia. El aprovechamiento de banda RF de un sistema 16-QAM para 90 Mbits/seg. (2 x DS3) con 21 circuitos de voz por MHz. (o de un sistema DRS 140/6700 con 22.5 circuitos de voz por MHz.) llega a sólo 2/3 del valor obtenido en un sistema FM 1800/2600.

Los sistemas con modulación Q-PSK son todavía más desfavorables en este sentido. Recientemente se consiguió desarrollar por primera vez un sistema operante con modulación 64-QAM, cuyo aprovechamiento de banda RF de 32 circuitos de voz por MHz. supera ligeramente la marca del sistema FM 1800/6200. La discriminación de canal adyacente necesaria fue alcanzada con una conformación de espectro mucho más progresiva, sin tener que recurrir a valores extremos de discriminación de polarización cruzada. En su concepción básica, el sistema de 64-QAM, no difiere significativamente de los sistemas 16-QAM conocidos.

- *Sistemas de radioenlace digitales de segunda generación.*

Los sistemas de radioenlace digitales de segunda generación se caracterizan por el empleo de procedimiento, casi siempre adaptivos, entre varios componentes de sistema, o varios sistemas, con el fin

de optimizar la transmisión. Por este motivo, los sistemas disponen de una **inteligencia** superior a la de sistemas de primera generación. Podemos entonces concluir que estos sistemas cumplirán con los objetivos de calidad y de disponibilidad del CCIR también con modulaciones de grado superior, es decir con **256-QAM**.

Hoy por hoy, las redes de radioenlaces operan básicamente con sistemas analógicos o con sistemas digitales de primera generación. El aprovechamiento de las bandas de RF tradicionales, inferior a los 11 GHz, debe girar en torno de los 20 a 25 circuitos de voz por MHz. incluyendo en la consideración la alta incidencia de sistemas de baja y mediana capacidad, con su eficiencia espectral inferior.

La importancia de los radioenlaces continuará creciendo si se consigue alcanzar dos metas principales: por un lado mantener la competitividad del medio de transmisión y por otro lado, brindar las capacidades de transmisión necesarias. Una segunda generación de sistemas digitales de radioenlace con modulaciones de más niveles y con procesos adaptivos, así como la explotación de las bandas superiores a los 11 GHz, permitirán realizar las capacidades de

transmisión que serán requeridas también en el futuro. Con respecto a la competitividad con otros medios de transmisión, ella será incrementada con la aplicación sistemática de los avances de la tecnología de circuitos integrados semidedicados VLSI; de la tecnología de componentes en microonda monolíticos e integrados; de la tecnología SAW (Surface Acoustic Wave); de la tecnología de base cerámica para resonadores y filtros; y finalmente de la tecnología GaAs para microondas.

3.1.1 DETERMINACION DE LA BER PARA SISTEMAS DE TRANSMISION DE DATOS CON MODULACION PSK Y QAM MULTINIVEL.

En circuitos de transmisión se generan varios tipos de errores que causan errores en los bits. En sistemas de microondas los ruidos que causan problemas principales son aquellos cuya generación es constante y cuya tensión eléctrica varía al azar. El ruido térmico es representativo de estos.

3.1.1.1 DEMODULACION.

La demodulación es un proceso en el cual, la información trasladada en la banda de onda portadora

mediante modulación, se traslada otra vez en la banda base.

Existen dos métodos de demodulación principales:

- *El método en el cual la componente variable contenida en la onda modulada se detecta directamente. Es un método de demodulación que no requiere la referencia de fase para demodular la onda modulada y se llama detección no coherente.*

- *En el método en el cual, se comparan la onda modulada con una onda no modulada, detectando la parte de diferencia entre ellas. Este método requiere una onda portadora de referencia cuya fase ya está conocida para lograr la onda portadora originaria antes del proceso de modulación y se llama detección coherente.*

3.1.1.2 BER DE LA SEÑAL DE DOS NIVELES.

El tren de impulsos unipolares (con niveles A y 0, donde A > 0) y el de impulsos bipolares (con niveles A/2 y -A/2) de NRZ se pueden considerar iguales uno a otro excepto por el nivel de decisión (A/2 para el unipolar, 0 para el bipolar).

Para un tren de impulsos unipolares, figura 3.3, considerando que "marca" y "espacio" (marca para 1 y espacio para 0) se generan igualmente y considerando que el nivel de decisión es A/2, la tasa media de bits erróneos p se expresa por[1]*:*

$$p = \frac{1}{2}\ erfc\left(\frac{A}{2\sqrt{2}\ \sigma}\right) \qquad (3.1)$$

donde:

p : Tasa media de bits erróneos.

A : Tensión de cresta de impulso.

σ : tensión eficaz de ruido.

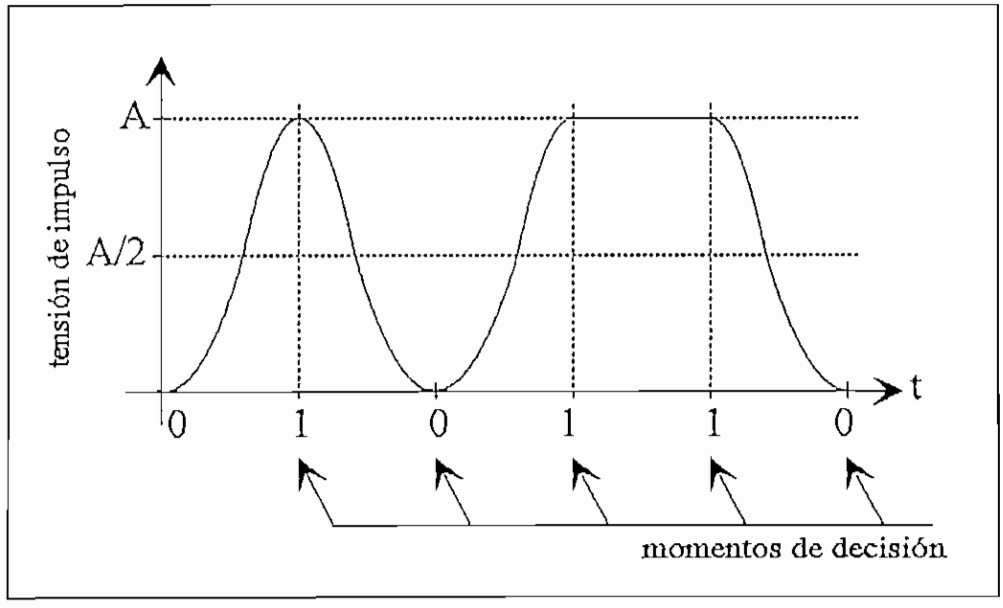


FIGURA 3.3 FORMA DE ONDA DE IMPULSOS UNIPOLARES

[1] TRANSMISION DIGITAL POR MICROONDA, IETEL, pag. 29.

Además, la función erfc(x) se define como la función de error complementaria, que para x es grande es[2]*:*

$$erfc(x) \backsim \frac{1}{x\sqrt{\pi}} \exp(-x^2) \qquad (3.2)$$

Se sabe además que la BER se decide por la relación entre la tensión de cresta del impulso ***A*** *y la tensión efectiva de ruido* σ*. Por lo general, la BER se representa por la función de la relación entre la potencia de la señal y la del ruido (S/N).*

En este caso la potencia del ruido se expresa mediante el valor efectivo σ^2 *y la de la señal se expresa por la potencia correspondiente a la tensión de cresta* ***A*** *en lugar del valor efectivo.*

3.1.1.3 SISTEMAS CON MODULACION PSK.

Al demodular una onda PSK bivalente (2-PSK) por el detector coherente, se tiene presente que en la señal PSK siempre existe una onda de señal por eso a la entrada del detector se suministran una onda de señal y ruido independiente de marca o espacio. Si la onda portadora de referencia esta completamente sincronizada con la onda de señal recibida, a la

---

[2] TRANSMISION DIGITAL POR MICROONDA, IETEL, pag. 58.

salida del detector aparece una componente del ruido cuya fase es la misma que la onda portadora de referencia. La distancia entre dos elementos de código, entre marca y espacio es 2A que es el doble de la señal de impulsos unipolares. La señal 2-PSK requiere justamente la mitad de la amplitud de esta señal (un cuarto de potencia) para lograr la misma tasa de errores, p.

$$p = \frac{1}{2}\ erfc\ (\sqrt{\rho}) \approx \frac{1}{2\sqrt{\pi\rho}}\ \exp\ (-\rho) \qquad (3.3)$$

con

$$\rho = \frac{A^2}{2\ \sigma^2} \qquad (3.4)$$

donde:

ρ : *Relación señal a ruido*

$A^2/2$: *Potencia de la señal correspondiente al valor de cresta A.*

σ^2 : *Potencia de ruido (distribución gaussiana).*

Para un sistema PSK M-ario los símbolos están colocados a intervalos fijos en una circunferencia de

radio A (amplitud de la señal) en el plano de fase como se muestra en la figura 3.4.

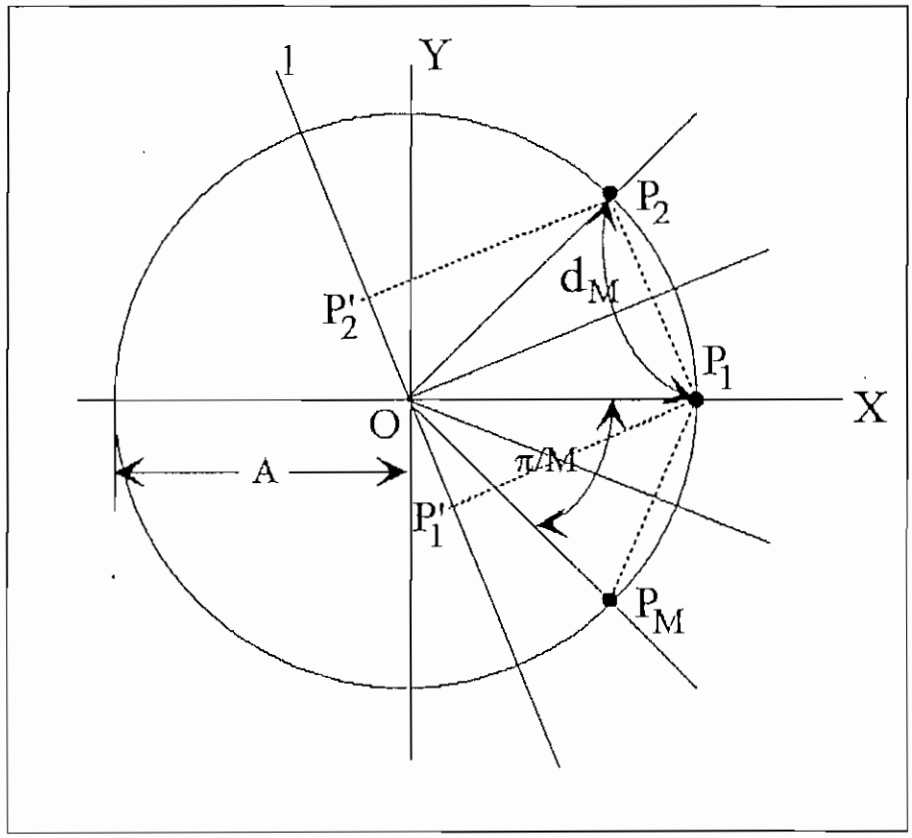


FIGURA 3.4 DIAGRAMA FASORIAL DE SEÑALES M-PSK

Si la S/N es grande, la tasa de errores del sistema depende de los errores entre los símbolos que están a la distancia más corta uno del otro. Para el símbolo P_1 basta lograr la tasa de errores relacionada con los símbolos P_2 y P_M que están adyacentes al P_1. La distancia más corta d_M, es el largo de un lado del M-ésimo polígono equilátero:

$$d_M = 2\ A \sin\left(\frac{\pi}{M}\right)$$

(3.5)

En el caso de que a la señal se añade un ruido gaussiano cuya distribución de fase es uniforme, para lograr una señal de salida del detector que hace máxima la distancia entre los del símbolos adyacentes (P_1P_2) basta detectar la señal de entrada coherentemente utilizando una onda portadora de referencia cuya fase es paralela a la línea P_1P_2.

La probabilidad p_{e1} de que el símbolo P_1 se tome erróneamente por P_2 se representa por[3]*:*

$$p_{s1} = \int_{d_M/2}^{\infty} \frac{1}{\sqrt{2\pi}\ \sigma} \exp\left(-\frac{x^2}{2\ \sigma^2}\right) dx$$

(3.6)

De la ecuación (3.5) y (3.6) se tiene:

$$p_{e1} = \frac{1}{2}\ erfc\left(\frac{d_M}{2\ \sqrt{2}\ \sigma}\right) = \frac{1}{2}\ erfc\left(\sqrt{\rho}\ \sin\frac{\pi}{M}\right)$$

(3.7)

La probabilidad de que el símbolo P_1 se tome por P_M, es el mismo resultado de la ecuación (3.7), pero en este caso, como onda portadora de referencia hay que seleccionar una onda cuya fase es paralela a la línea

---

[3] TRANSMISION DIGITAL POR MICROONDA, IETEL, pag. 64.

$P_1 P_M$. Por tanto considerando que respecto a un símbolo existen dos símbolos que están simétricamente ubicados a ambos lados de él a la distancia más corta, utilizando la ecuación (3.7) la tasa de símbolos erróneos del sistema PSK M-ario se expresa por:

$$p_s \approx 2\, p_{e1} = erfc\left(\sqrt{\rho}\, \sin\frac{\pi}{M}\right) \qquad (M > 2) \tag{3.8}$$

3.1.1.4 SISTEMAS CON MODULACION QAM.

En sistemas QAM tanto la amplitud como la fase llevan información simultáneamente, sin embargo solamente la onda QAM de 16 niveles (16-QAM) o de más altos niveles satisface esta condición, la figura 3.5 muestra la asignación de los símbolos de una onda portadora recibida. La tasa de símbolos erróneos de la onda 16-QAM se logra mediante el mismo método de cálculo que en el caso de la onda PSK multivalente, siendo A la amplitud máxima de la señal recibida.

La distancia más corta entre los símbolos dados d se expresa por:

$$d = (\sqrt{2/3})\, A \tag{3.9}$$

Por tanto la tasa de símbolos erróneos p_s que se

generan entre dos símbolos se representan por la siguiente ecuación aproximada[4]*:*

$$p_{eQ} = \int_{d/2}^{\infty} \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{x^2}{2\,\sigma^2}\right) dx = \frac{1}{2}\, erfc\left(\frac{d}{2\sqrt{2}\,\sigma}\right)$$

(3.10)

$$p_{eQ} = \frac{1}{2}\, erfc\left(\frac{A}{6\sigma}\right) = \frac{1}{2}\, erfc\left(\frac{1}{3}\sqrt{\frac{\rho}{2}}\right)$$

(3.11)

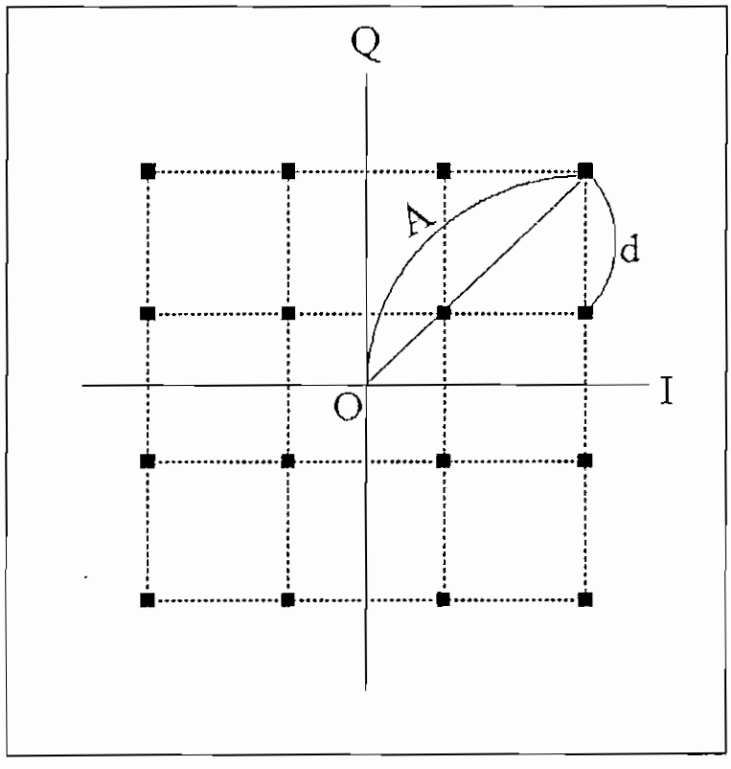


FIGURA 3.5 DIAGRAMA FASORIAL PARA LA SEÑAL 16-QAM

Como la onda 16-QAM toma tres niveles de amplitud, la

---

[4] TRANSMISION DIGITAL POR MICROONDA, IETEL, pag. 72.

potencia media se expresa por:

$$\frac{1}{2}\left[\frac{1}{4}A^2 + \frac{1}{2}\left(\frac{\sqrt{5}}{3}\right)^2 A^2 + \frac{1}{4}\left(\frac{1}{3}\right)^2 A^2\right] = \frac{5}{18}A^2$$

(3.12)

entonces la relación señal a ruido para la potencia media obtenida con (3.12) se expresa por:

$$\rho_s = \frac{5\,A^2}{18\,\sigma^2} = \frac{5\,d^2}{4\,\sigma^2} = \frac{5}{9}\,\rho$$

(3.13)

Utilizando las ecuaciones (3.11) y (3.13) se tiene:

$$p_{sQ} = \frac{1}{2}\,erfc\left(\sqrt{\frac{\rho_s}{10}}\right)$$

(3.14)

Según la posición del símbolo defiere el número de símbolos que están a la distancia d desde el símbolo contemplado, por lo que varía la tasa de símbolos erróneos conforme a la posición del símbolo. La tasa de símbolos erróneos media p_s se expresa por:

$$p_s = \frac{1}{16}\,(4 * 2p_{sQ} + 8 * 3p_{sQ} + 4 * 4p_{sQ}) = 3p_{sQ}$$

(3.15)

Reemplazando la ecuación (3.11) en (3.15) se tiene:

$$p_s = \frac{3}{2}\,erfc\left(\sqrt{\frac{\rho_s}{10}}\right) = \frac{3}{2}\,erfc\left(\frac{1}{3}\sqrt{\frac{\rho}{2}}\right)$$

(3.16)

La 16-QAM es un método de modulación que se emplea ampliamente hoy en día en sistemas de radioenlace digitales de alta eficiencia. Igualmente en el caso de 64-QAM y la 256-QAM, en las cuales el número de los símbolos aumenta bastante en comparación con el aumento de la información a transmitirse.

En cuanto a la señal QAM M-ario que tiene asignación de símbolo reticular como se muestra en la figura 3.6,

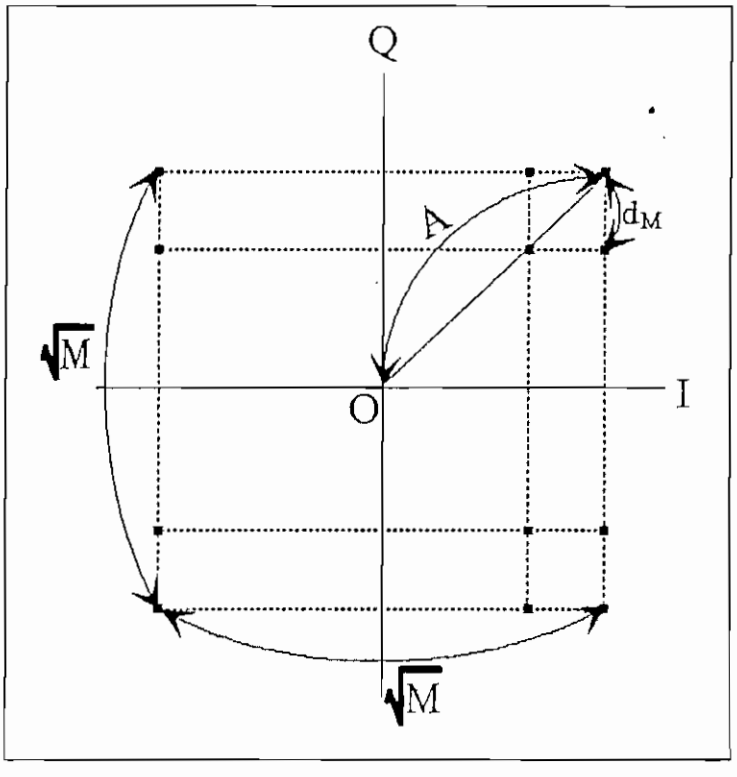


FIGURA 3.6 DIAGRAMA FASORIAL DE SEÑALES M-QAM

la distancia más corta entre dos símbolos d_M se expresa por:

$$d_M = \frac{\sqrt{2}\,A}{\sqrt{M} - 1}$$

(3.17)

$$p_{sQ} = \frac{1}{2}\, erfc\left[\frac{\sqrt{2\rho}}{2(\sqrt{M} - 1)}\right]$$

(3.18)

Existe tres tipos de probabilidades p_{sQ}, por lo que:

$$p_s = \frac{1}{M}\,(4*2p_{sQ} + 4*(\sqrt{M} - 2)*3p_{sQ} + (\sqrt{M} - 2)^2*4p_{sQ})$$

(3.19)

Obteniéndose:

$$p_s = \frac{4\,(\sqrt{M} - 1)}{\sqrt{M}}\, p_{sQ}$$

(3.20)

La calidad de transmisión de un radioenlace debe apreciarse por medio de la tasa de errores respecto a trenes de impulsos bivalentes (BER), por tanto hay que examinar la relación entre la tasa de símbolos erróneos y la BER.

En caso de que m-trenes de impulsos bivalentes se transmiten por utilizando códigos de M-símbolos donde $M = 2^m$, una de las m-combinaciones de "1" y "0" representa un símbolo, lo que significa que un símbolo tiene la información de m-bits. Si un símbolo sufre

error, el número de errores que se generan en los trenes de impulsos bivalentes después de la demodulación oscila entre 1 y m.

Para el caso general de señalización M-aria, la probabilidad de tomar una decisión incorrecta se ha denominado probabilidad de error en los símbolos, o sencillamente p_s. *A menudo es conveniente especificar el rendimiento de los sistemas por medio de la probabilidad de error en los bits p, incluso cuando las decisiones se tomen con base en los símbolos para los cuales* $m > 1$. *Para señales ortogonales,* p_s *y p están relacionadas como sigue:*

$$p = \frac{2^{m-1}}{2^m - 1} p_s$$

(3.21)

Para esquemas no ortogonales tales como la señalización M-PSK, suele utilizarse una codificación de binario a M-ario tal que las sucesiones binarias correspondientes a símbolos adyacentes (defasamientos) difieran sólo en una posición de bit; uno de tales códigos es el de Gray.

Cuando ocurre un error M-ario en los símbolos, es más probable que solamente uno de los m bits de entrada este equivocado.

Para tales señales:

$$p = \frac{p_e}{\log_2 M} = \frac{1}{m} p_s \qquad \text{para } p_e < 1$$

(3.22)

Los símbolos Gray son los que hacen mínima la BER y se emplean ampliamente en transmisión digital por microondas.

En la PSK M-ario existen varios métodos en cuanto a la correspondencia entre fases de onda portadora y símbolos.

Los más representativos son los símbolos binarios naturales y los símbolos de Gray. Una correspondencia entre símbolos y fases se muestra en la tabla 3.1.

Para el caso en que se comparan dos símbolos cuyas fases correspondientes están adyacentes una a otra en una circunferencia, en cuanto a los símbolos binarios naturales existen algunos casos en que más de dos bits son distintos entre dos símbolos adyacentes.

Por otra parte en cuanto a los símbolos de Gray sólo un bit es distinto entre cualesquier dos símbolos adyacente.

Es decir para una tasas de símbolos erróneos dada, la asignación de fases conforme a los símbolos de Gray realiza la tasa de bits erróneos menor que la conforme a los símbolos binarios naturales.

TABLA 3.1 CORRESPONDENCIA ENTRE FASE Y SIMBOLO BINARIO NATURAL Y SIMBOLO DE GRAY MODULACION 8-PSK.

fase	Símbolo bin. natural			Símbolo de Gray		
	tren 1	tren 2	tren 3	tren 1	tren 2	tren 3
0	0	0	0	0	0	0
$\pi/4$	0	0	1	0	0	1
$\pi/2$	0	1	0	0	1	1
$3\pi/4$	0	1	1	0	1	0
π	1	0	0	1	1	0
$5\pi/4$	1	0	1	1	1	1
$3\pi/2$	1	1	0	1	0	1
$7\pi/4$	1	1	1	1	0	0

*Para el sistema **QAM**, también se satisface la condición de Gray, es decir, entre cualesquier par de símbolos adyacentes (a la distancia más corta) sólo un bit es distinto, figuras 3.7 y 3.8.*

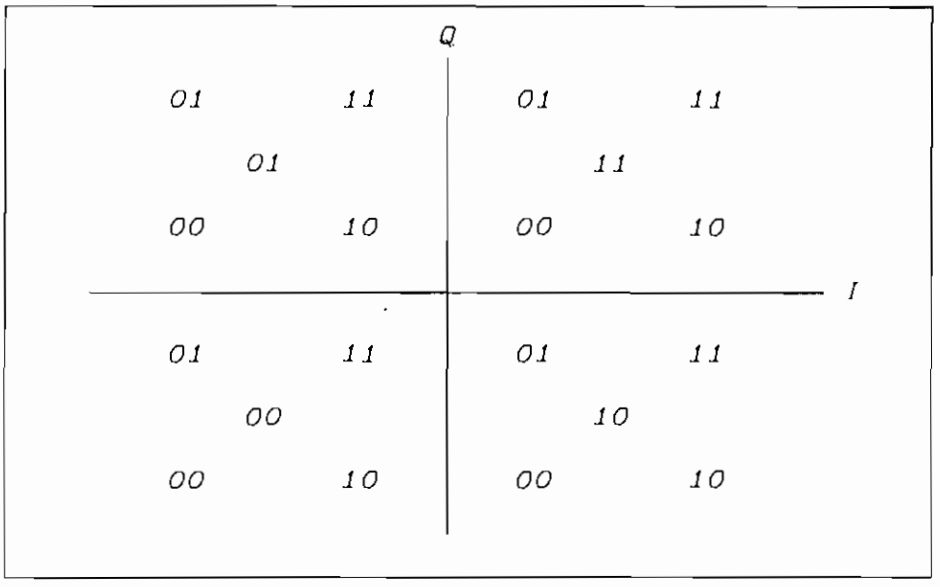


FIGURA 3.7 CODIGO POR ASIGNACION BINARIA, 16-QAM.

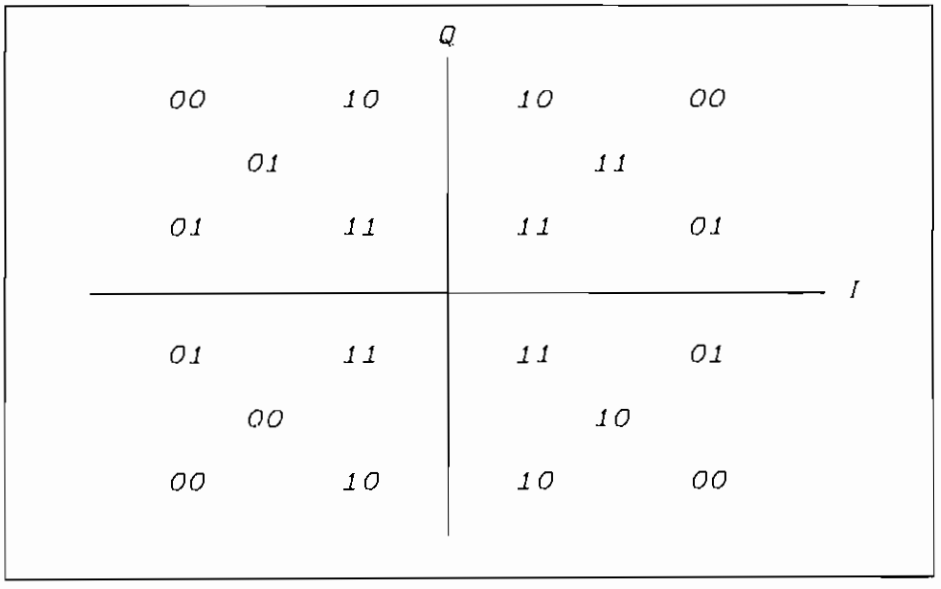


FIGURA 3.8 CODIGO POR ASIGNACION DE GRAY, 16-QAM.

3.1.2 RELACION ENTRE LA BER Y LA POTENCIA DE TRANSMISION.

Al evaluar el rendimiento de un sistema la cantidad más importante no es la potencia recibida P_r, sino la relación señal a ruido SNR. Esto se debe a que la restricción fundamental del sistema es la capacidad de detectar la señal, con una probabilidad de error en los bits p aceptable, en presencia de ruido. El valor de cresta de la envolvente de señal queda mantenido casi igual a la amplitud de la onda no modulada aunque una parte de la potencia de las ondas laterales incluidas en la señal se pierde debido al filtro de pasabanda, por lo cual se puede considerar que la relación señal a ruido expresada en la ecuación (3.4) es igual a la relación portadora a ruido, se habla de una razón de potencia sobre ruido de la portadora (C/N o CNR), como la S/N (P_r/N) de interés particular.

$$\frac{P_r}{N} = \frac{EIRP\ (G_r/T_e)}{L_p\ L_{os}\ L_{oi}\ K\ B} \qquad (3.23)$$

Como se trata de valores teóricos, los valores reales medidos son algunos dB. mayores. En realidad la diferencia entre el valor teórico C/N para una BER determinada y el valor de C/N medido es, la figura de ruido NF de receptor.

En sistemas analógicos, el ancho de banda del ruido suele ser mayor que el de la señal, y P_r/N es el parámetro principal para medir la detectabilidad de la señal y la calidad del funcionamiento. Sin embargo, en los receptores digitales suelen utilizarse correlacionadores o filtros de acoplamiento (compensación) en los que el ancho de banda de la señal y la del ruido se consideran iguales. En vez de considerar la potencia del ruido de entrada, un planteamiento común para los enlaces digitales es sustituir la potencia del ruido por la densidad del ruido ya que la potencia de ruido $N = N_o B$, donde B es el ancho de banda equivalente de ruido; de la ecuación (3.23) se obtiene:

$$\frac{P_r}{N_o} = \frac{EIRP\ (G_r/T_s)}{L_p\ L_{os}\ L_{ol}\ K}$$

(3.24)

Considerando las ecuaciones anotadas en el punto 3.1.1 de este capítulo, el parámetro ρ *correspondiente a la relación señal a ruido SNR (P_r/N) expresado en la ecuación (3.23), son posibles de obtener las siguientes relaciones:*

$$\rho = \frac{S}{N} = \frac{P_r}{N} = \frac{P_r}{N_o B} = \left(\frac{E_b}{N_o}\right)\left(\frac{R_b}{B}\right)$$

(3.25)

donde:

P_r : Potencia de la señal recibida en la entrada del detector.

N_o : Densidad espectral de ruido.

S : Potencia promedio de la señal de modulación.

N : Potencia de ruido.

R_b : Velocidad de transferencia de datos.

B : Ancho de banda de la señal

Como se puede analizar en el capítulo 1 (punto 1.4), para sistemas de radioenlaces digitales, la relación E_b/N_o a más de otros parámetros, depende en forma directa de la potencia de transmisión P_t por medio del $EIRP = P_t G_t$.

3.1.3 INTERPRETACION DE RESULTADOS.

La señalización M-aria se describió como un esquema de modulación o codificación que procesa m bits a la vez. El sistema dirige al modulador para que elija uno de sus $M = 2^m$ ondas de forma para cada sucesión de m bits, donde M es el tamaño de conjunto de símbolos y m es el número de dígitos binarios que representa cada

símbolo. La señalización M-aria sola, para el caso en que m > 1, puede considerarse un procedimiento de codificación de la onda de forma que afecta el rendimiento del sistema.

Para los sistemas con modulación M-PSK se puede obtener la probabilidad de error en los bits p en función de la relación E_b/N_o; considerando las ecuaciones (3.2), (3.22), (3.24) y (3.25), las ecuaciones (3.3) y (3.8) toman las siguientes formas:

a) Para modulación B-PSK.

$$p = \frac{\frac{1}{2} \exp\left[-\left(\frac{E_b}{N_o}\right)\left(\frac{R_b}{B}\right)\right]}{\sqrt{\pi\left(\frac{E_b}{N_o}\right)\left(\frac{R_b}{B}\right)}}$$

(3.26)

b) Para modulación M-PSK, con M > 2.

$$p = \frac{\frac{1}{m} \exp\left[-\left(\frac{E_b}{N_o}\right)\left(\frac{R_b}{B}\right) \sin^2\left(\frac{\pi}{M}\right)\right]}{\sqrt{\pi\left(\frac{E_b}{N_o}\right)\left(\frac{R_b}{B}\right) \sin\left(\frac{\pi}{M}\right)}}$$

(3.27)

De igual manera, para sistemas con modulación M-QAM se puede obtener p en función de la relación E_b/N_o a

partir de las ecuaciones (3.16) y (3.20) pero en este caso se utiliza la ecuación (3.21) en lugar de la (3.22), se tiene:

a) Para modulación 16–QAM.

$$p = \frac{9\left(\frac{2^{m-1}}{2^m - 1}\right)}{\sqrt{2\,\pi\left(\frac{E_b}{N_o}\right)\left(\frac{R_b}{B}\right)}}\ \exp\left[\frac{-\,(E_b/N_o)\ (R_b/B)}{18}\right]$$

(3.28)

b) Para modulación M–QAM, con $M > 16$.

$$p = \frac{4\,(\sqrt{M} - 1)^2\left(\frac{2^{m-1}}{2^m - 1}\right)}{\sqrt{2\,M\,\pi\left(\frac{E_b}{N_o}\right)\left(\frac{R_b}{B}\right)}}\ \exp\left[\frac{-\,(E_b/N_o)\ (R_b/B)}{2\,(\sqrt{M} - 1)^2}\right]$$

(3.29)

Las señales ortogonales manifiestan una p mejorada a expensas del ancho de banda a medida que m aumenta; las señales no ortogonales manifiestan eficiencia de ancho de banda mejorada (R_b/B) a expensas de la potencia o del rendimiento de p. La relación R_b/B se mide en bits por segundo por hertz (bits/seg/Hz); para la señalización binaria, el valor típico de la eficiencia de ancho de banda es aproximadamente 1 bit/seg/Hz. Sin embargo, los sistemas de manipulación por desplazamiento de fases múltiples (M–PSK) y de

modulación de amplitud en cuadratura (QAM) a menudo presentan eficiencias de ancho de banda de 3 bits/seg/Hz o más.

Utilizando las ecuaciones (3.26), (3.27), (3.28) y (3.29) se realizan las figuras 3.9 y 3.10 en las que se presenta la variación de p (BER) en función de la relación señal a ruido (S/N o ρ), ecuación (3.25),

para varios valores de M en las modulaciones M-arias de PSK y QAM respectivamente. Sistemas no codificados.

En la figura 3.9 se muestran las curvas de las tasas de bits erróneos para varios valores de niveles en cuanto a la detección coherente M-PSK, mientras que la figura 3.10 presenta las curvas para M-QAM. Según se aumenta el número de niveles aumenta también la S/N requerida para lograr una tasa de errores dada. Sin embargo también aumenta la magnitud de información contenida en un símbolo, por tanto en caso de que la velocidad de información es fija, la modulación multivalente requiere de menor velocidad de transmisión de símbolos que la bivalente. La velocidad de transmisión de símbolos es uno de los parámetros que deciden la forma de la envolvente de espectro de la onda portadora.

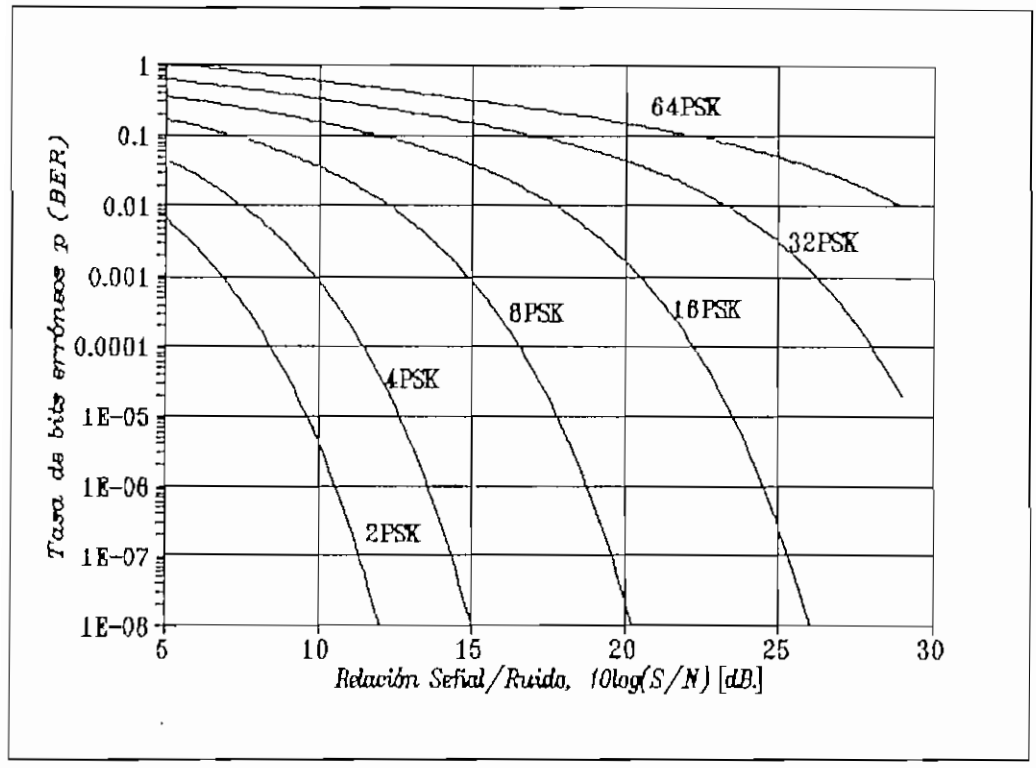


FIGURA 3.9 TASA DE BITS ERRONEOS PARA DIVERSOS MODOS DE DETECCION COHERENTE PSK (ANCHURA DE BANDA ES CONSTANTE).

Se puede apreciar en la figura 3.9 que si se tiene una BER = 10^{-3}, para mejorarla a un valor de BER = 10^{-8} es necesario realizar un incremento en la relación señal a ruido S/N. La tabla 3.2 resume el valor a incrementarse en la relación S/N para mejorar la BER desde el valor que correspondiente a la BER = 10^{-3} al valor que corresponde a la BER = 10^{-8} para los sistemas no codificados M-PSK y M-QAM.

TABLA 3.2 INCREMENTO A LA S/N PARA VARIAR LA BER DE 10^{-3} A 10^{-6}.

Tipo de modulación	Incremento de la S/N
M-PSK	5.5 dB.
M-QAM	5.0 dB.

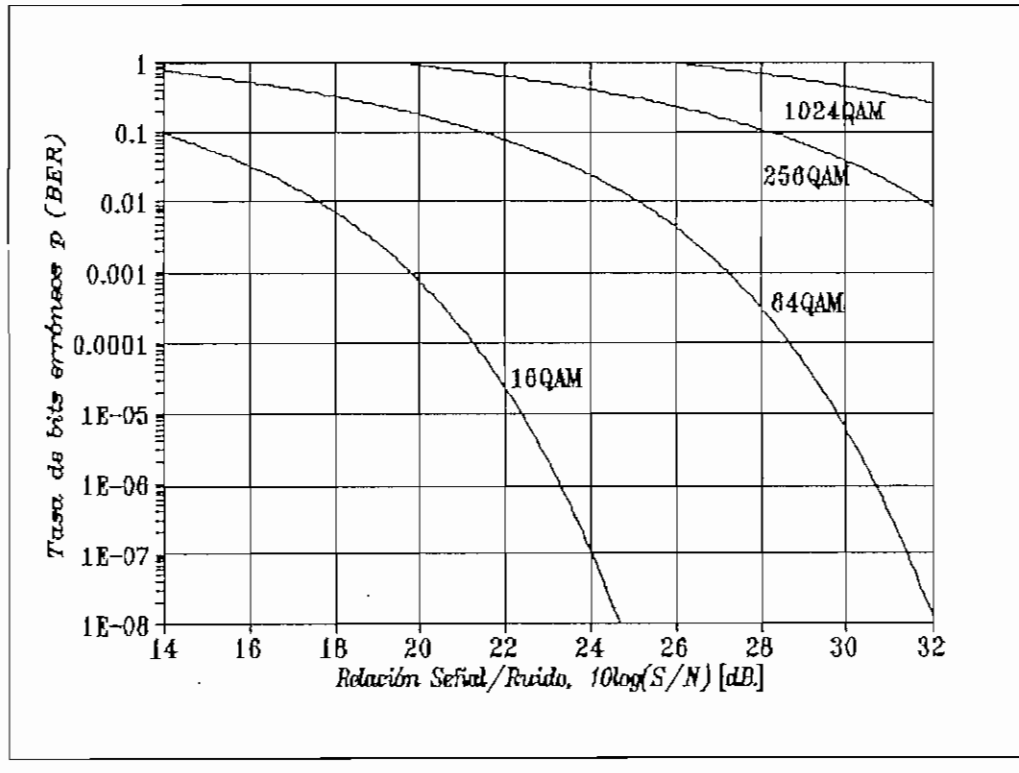


FIGURA 3.10 TASA DE BITS ERRONEOS PARA DIVERSOS MODOS DE DETECCION COHERENTE QAM (ANCHURA DE BANDA ES CONSTANTE).

Para esta variación de la BER, se tiene un incremento

3.2 RELACION ENTRE LA BER, BITS DE PARIDAD DE LOS CODIGOS BLOQUE LINEALES Y POTENCIA DE TRANSMISION.

Si analizamos las ecuaciones presentadas en el punto 1.4 del capítulo 1, las ecuaciones del punto 2.3 del capitulo 2 y las ecuaciones presentadas en el punto 3.1.3 de este capítulo, podemos darnos cuenta que existen varios parámetros que pueden ser variados para obtener mejoras en el radioenlace.

La figura 3.11 presenta un enlace de comunicaciones, en el que se puede apreciar la localización de los diferentes parámetros que se han estado estudiando. La obtención de cada uno de ellos ha sido plenamente determinado con las respectivas ecuaciones que ya están presentadas. Por el objetivo de este trabajo, consideraremos variables únicamente a los siguientes parámetros:

k : Longitud de la palabra mensaje a ser codificada con los códigos bloque lineales.

n : Longitud de la palabra código, de los códigos bloque lineales.

p : Probabilidad de error en los bits (valor teórico de la BER).

P_t : Potencia de transmisión.

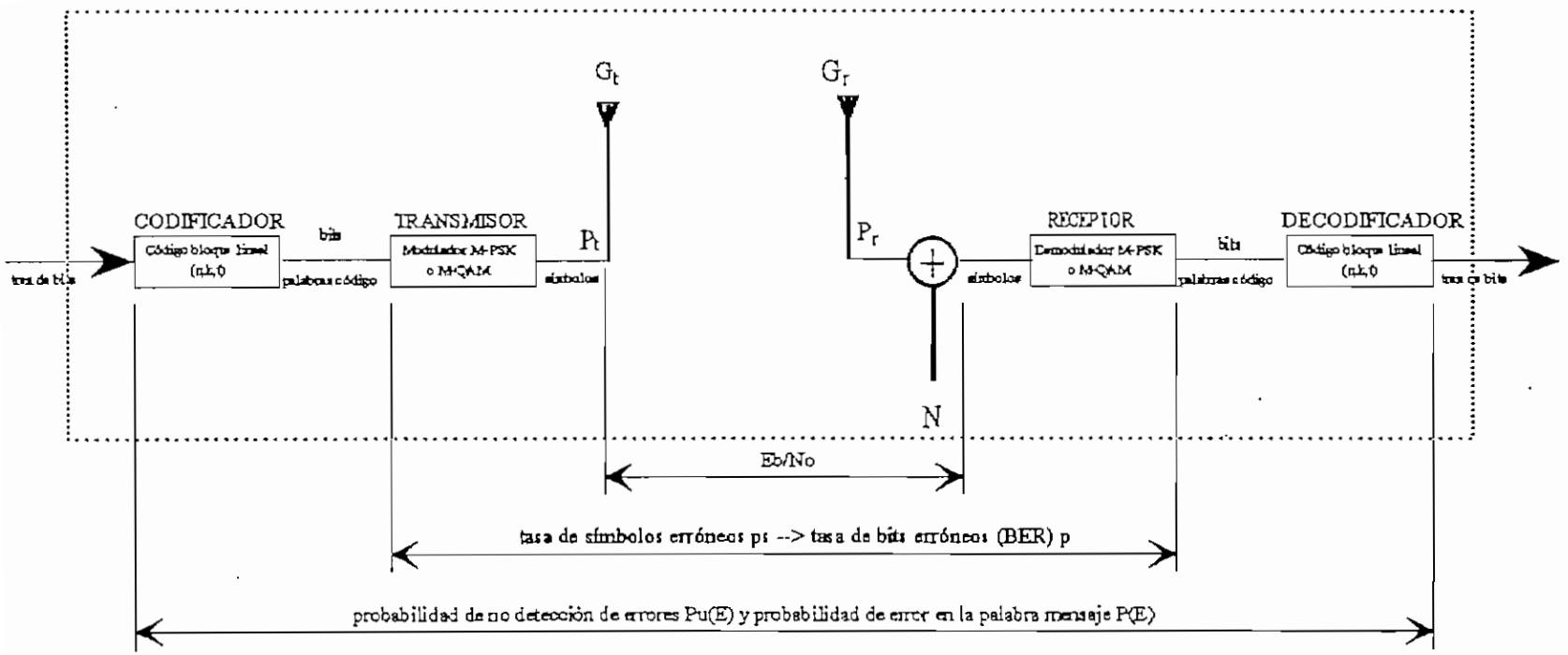


FIGURA 3.11 MODELO SIMPLIFICADO DE UN ENLACE DE COMUNICACIONES

Mientras que los demás parámetros los consideraremos fijos. También se considerará los resultados para las modulaciones 4-PSK, 8-PSK, 16-PSK, 16-QAM, 64-QAM que son los tipos de modulación más utilizados en la práctica; ya que teóricamente se puede tener una gran variedad de modulación M-aria.

En el punto 2.3 del capitulo 2, se realiza un análisis de la relación de la BER con los parámetros n y k de los códigos bloque lineales para el caso de una modulación B-PSK. Se indica que al utilizar un código bloque lineal, la relación E_b/N_o varía con respecto al sistema no codificado en un factor k/n.

Este mismo criterio podemos utilizar para los sistemas con modulación M-arios. De las ecuaciones (3.24) y (3.25) podemos obtener la relación E_b/N_o como se indica a continuación:

$$\frac{E_b}{N_o} = \frac{P_r}{R_b\, N_o}$$

(3.30)

En el sistema codificado el factor k/n incide inversamente en la velocidad de transmisión de los bits R_b. Teniendo en cuenta que n > k, la velocidad de transmisión de los bits aumentaría en un factor n/k.

Se debe tener presente que en las ecuaciones indicadas en el punto 3.1.3 de este capítulo, la relación E_b/N_o se refiere al caso no codificado.

Utilizando las ecuaciones (3.27) para M-PSK y (3.29) para M-QAM, para sistemas codificados se obtienen las siguientes ecuaciones:

a) Para sistemas con modulación M-PSK, M > 2.

$$p = \frac{\frac{1}{m} \exp\left[-\left(\frac{k}{n}\right)\left(\frac{E_b}{N_o}\right)\left(\frac{R_b}{B}\right)\sin^2\left(\frac{\pi}{M}\right)\right]}{\sqrt{\pi\left(\frac{k}{n}\right)\left(\frac{E_b}{N_o}\right)\left(\frac{R_b}{B}\right)\sin\left(\frac{\pi}{M}\right)}}$$

(3.31)

b) Para sistemas con modulación M-QAM, M ≥ 16.

$$p = \frac{4\left(\sqrt{M}-1\right)^2\left(\frac{2^{m-1}}{2^m-1}\right)}{\sqrt{2\,M\,\pi\left(\frac{k}{n}\right)\left(\frac{E_b}{N_o}\right)\left(\frac{R_b}{B}\right)}} \exp\left[\frac{-\,(k/n)\;(E_b/N_o)\;(R_b/B)}{2\left(\sqrt{M}-1\right)^2}\right]$$

(3.32)

Utilizando las ecuaciones (3.25) y (3.31) se obtienen las figuras 3.12, 3.13 y 3.14 en las que se presenta la variación de p (BER) en función de la relación S/N, utilizando codigos bloque lineales que corrigen un

solo error (t = 1), en sistemas que tienen modulación 4–PSK, 8–PSK y 16–PSK respectivamente.

Si comparamos la figura 3.9, que presenta curvas de la probabilidad de error en los bits (BER) para sistemas no codificados en función de la relacion S/N para modulación M–PSK, con las figuras 3.12, 3.13 y 3.14 que son la misma clase de curvas pero para sistemas codificado; se puede ver que en los sistemas codificados, en sus respectivo tipo de modulación, la probabilidad de error en los bits se degenera ya que para los mismos valores de S/N la BER aumenta. Esto se debe a que la velocidad de transmisión de los bits a

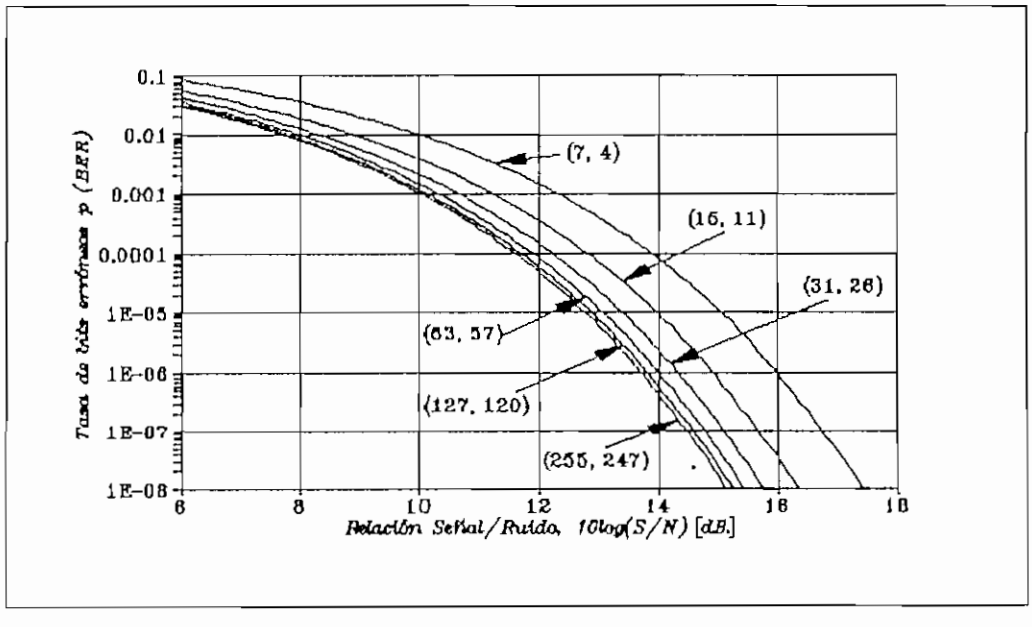


FIGURA 3.12 BER vs. S/N PARA 4–PSK.

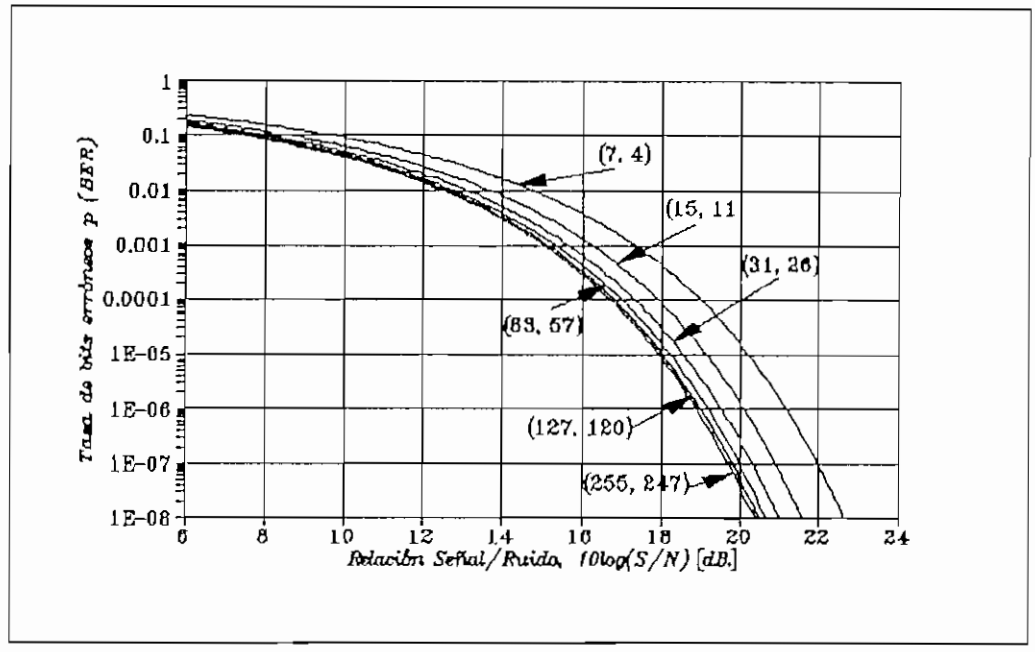


FIGURA 3.13 BER vs. S/N PARA 8-PSK.

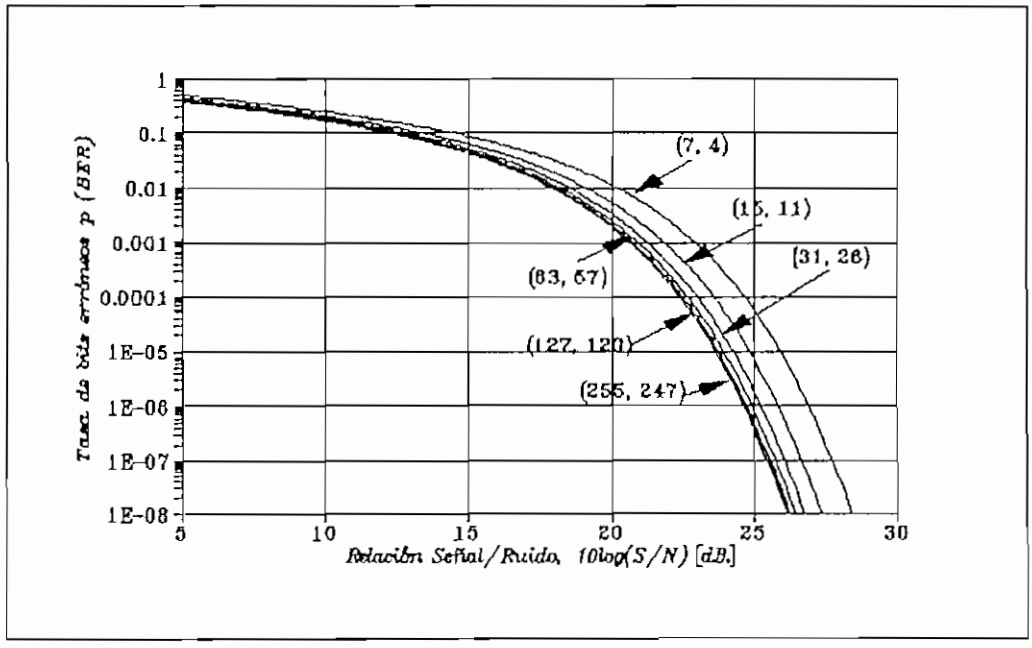


FIGURA 3.14 BER vs. S/N PARA 16-PSK.

variado en un factor n/k para los sistemas codificados, debe ser posible detectar más bits en el mismo intervalo y con la misma potencia disponible. Es por eso que para sistemas codificados se debe realizar el análisis con la probabilidad de error en el mensaje (bits de información) P(E), que será equivalente a la BER-Información (BER de los bits de información).

Considerando únicamente la variación de la Potencia de transmisión P_t (por lo tanto S/N) se puede apreciar, en las curvas de las figuras 3.12, 3.13 y 3.14 para modulación 4, 8 y 16-PSK respectivamente, que el valor del incremento de la S/N es igual que para los sistemas no codificados y además es independiente del código bloque lineal utilizado y del número de niveles de la modulación, para obtener una mejora de la BER de 10^{-3} a 10^{-6}. Es decir, el incremento es 5.5 dB.

Para sistemas codificados con modulaciones 16-QAM y 64-QAM se tiene las figuras 3.15 y 3.16 respectivamente, obtenidas con las ecuaciones (3.25) y (3.32). En ellas se presentan curvas que relacionan la BER con S/N utilizando códigos bloque lineales que corrigen un solo error (t = 1). Si comparamos estas figuras con la figura 3.10, se tiene la misma conclusión que para la modulación M-PSK. Es decir, hay

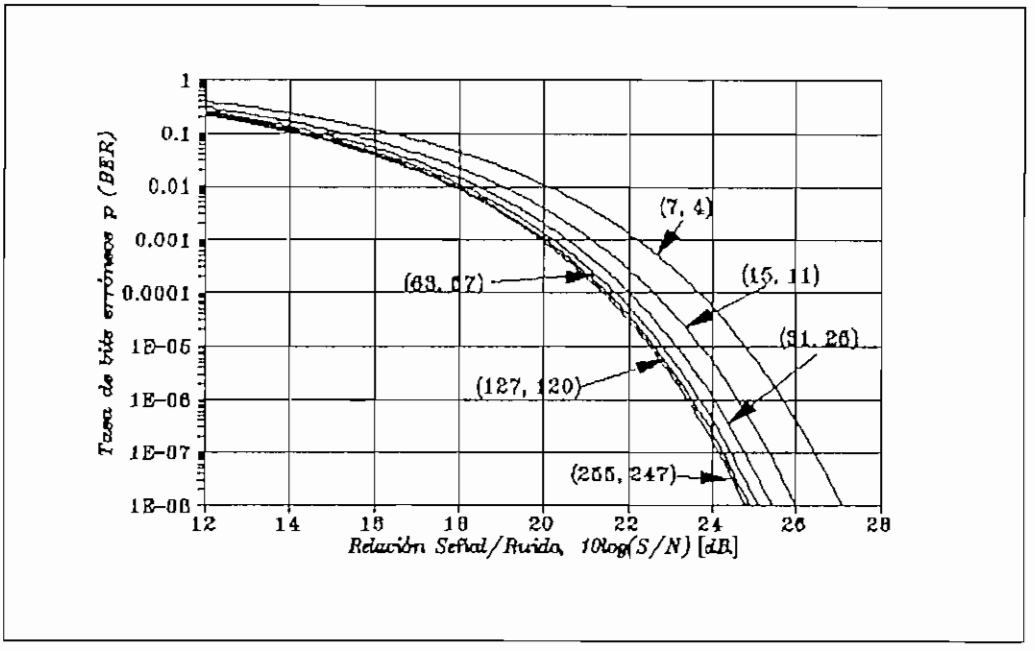


FIGURA 3.15 BER vs. S/N PARA 16-QAM.

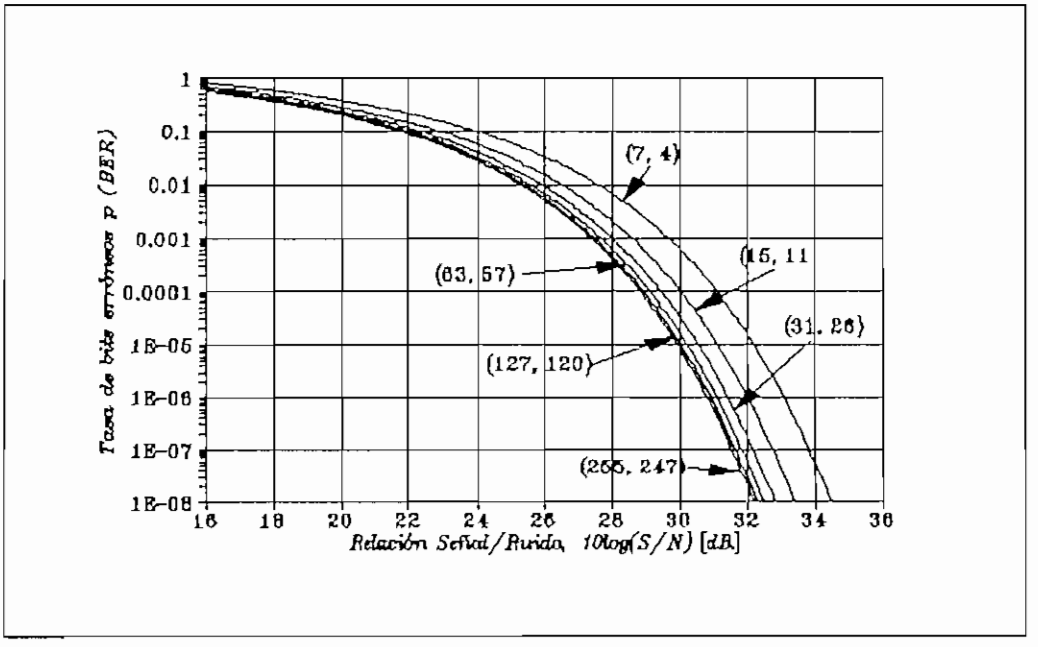


FIGURA 3.16 BER vs. S/N PARA 64-QAM.

una degradación en el valor de p debido al incremento de la velocidad de transmisión de los bits.

Realizando las mismas consideraciones hechas para los sistemas codificados M-PSK, se tiene que para los sistemas codificados M-QAM, de las figuras 3.15 y 3.16 de las modulaciones 16 y 64-QAM respectivamente, el valor que se incrementa a la relación S/N para obtener una mejora de la BER de 10^{-3} a 10^{-8} es igual al valor obtenido en los sistemas no codificados, es decir es de 5.0 dB. Este valor es independiente del código bloque lineal utilizado y del número de niveles de la modulación. En las figuras 3.15 y 3.16 se puede apreciar este valor, para la variación de la BER.

La mejora del rendimiento debido a la codificación se presenta a la salida del decodificador (si es un sistema codificado) o a la salida del receptor (si es un sistema no codificado. Por lo que para sistemas codificados es necesario calcular la probabilidad de error en la palabra mensaje P(E) (BER-Información) y la probabilidad de no detección de errores Pu(E).

La parte encerrada con línea segmentada en la figura 3.11, representará el canal binario simétrico, al cual ingresa un tren de bits (1's y 0's), internamente son

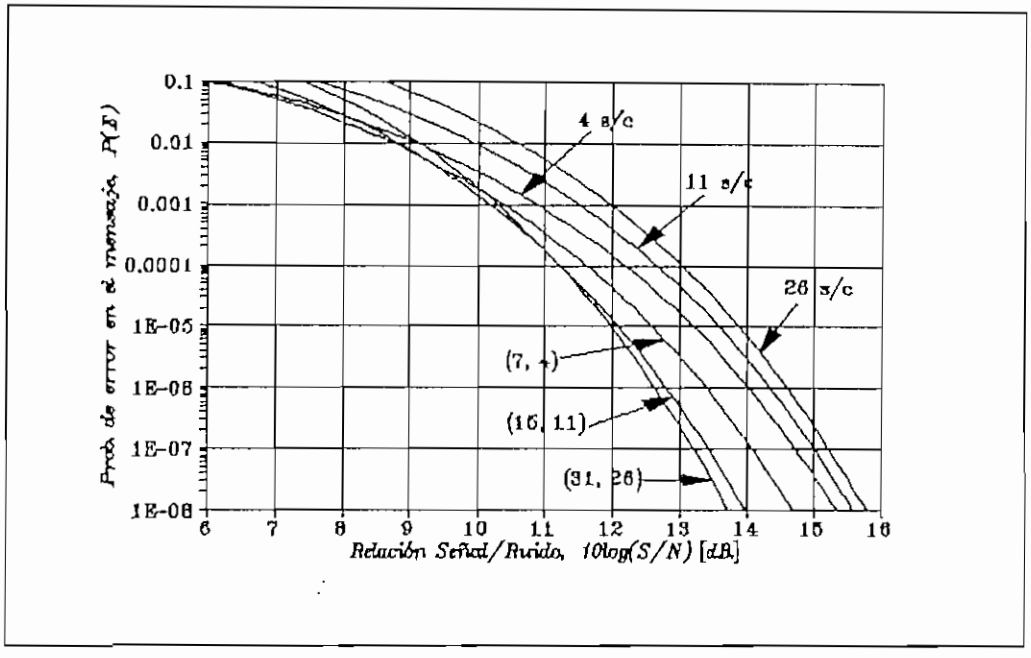


FIGURA 3.17 P(E) vs. S/N PARA 4-PSK.

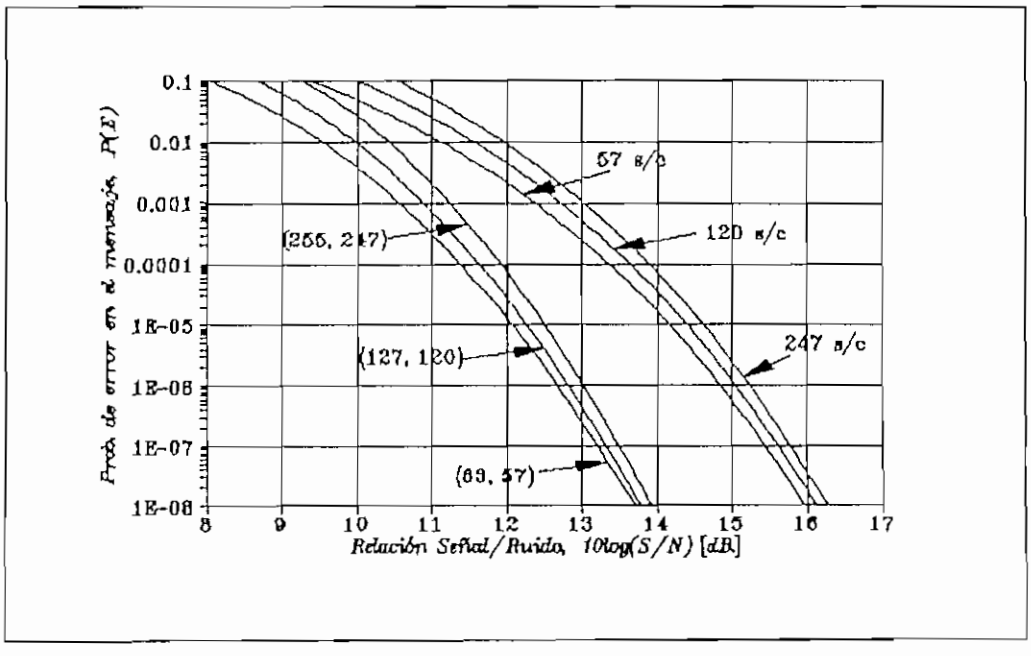


FIGURA 3.18 P(E) vs. S/N PARA 4-PSK.

agrupado en palabras mensaje de **k** bits los cuales son enviados primero al codificador y luego al transmisor (en el caso de sistemas codificados) o directamente al transmisor (en el caso no codificado). Es decir que a la salida del canal obtendremos el tren de bits en grupos de **k** bits. La probabilidad de transición de este canal es la probabilidad de error en los bit (tasa de bits erróneos) **p**.

Las figuras 3.17 y 3.18 presentan curvas de la probabilidad de error en la palabra mensaje **P(E)** respecto a la relación **S/N**, para los sistemas codificados y no codificados que utilizan modulación 4-PSK. Los sistemas codificados utilizan los códigos bloque lineales que corrigen un solo error ($t = 1$).

Cualquiera que sea el tipo de sistema, codificado o no codificado, al transmisor ingresa un tren de bits. Para comparar la **P(E)** en los dos sistemas se considerará que al canal ingresan en grupos de **k** bits.

Por ejemplo, si tenemos una $P(E) = 10^{-4}$ enviando palabras mensaje de 26 bits y se desea disminuir este valor manteniendo constante los demás parámetros del enlace como la **S/N** = 13 dB., si utilizamos el código bloque lineal (31,26) que corrige un solo error, se

puede apreciar que la se obtiene una disminución considerable, $P(E) = 2*10^{-7}$. Figura 3.17.

La figura 3.17 presenta curvas para valores de **k** de 4, 11 y 26 bits y la figura 3.18 para 57, 120 y 247 bits.

Las figuras 3.19 y 3.20 presenta las curvas para modulación 8-PSK y las figuras 3.21 y 3.22 para modulación **16-PSK**, tanto para sistemas codificados como para no codificados de la probabilidad de error en la palabra mensaje P(E) en función de la relación **S/N**. Se presenta curvas para diferentes longitudes de la palabra mensaje **k**. Las siglas **s/c** representan a los sistemas no codificados.

Para el otro sistema de modulación (M-ario), se tiene las figuras 3.23 y 3.24 para la modulación **16-QAM** y las figuras 3.25 y 3.26 para la modulación **64-QAM.**

Todas las curvas presentadas de P(E), tanto para sistemas codificados como para los no codificados, respecto de la relación **S/N** son obtenidas previo el cálculo de la tasa de bits erróneos p (indicado en el punto 3.1.3 de este capítulo). El respectivo valor de p se aplica a la ecuación (2.20) para el caso no codificado o a la (2.23) para el caso codificado.

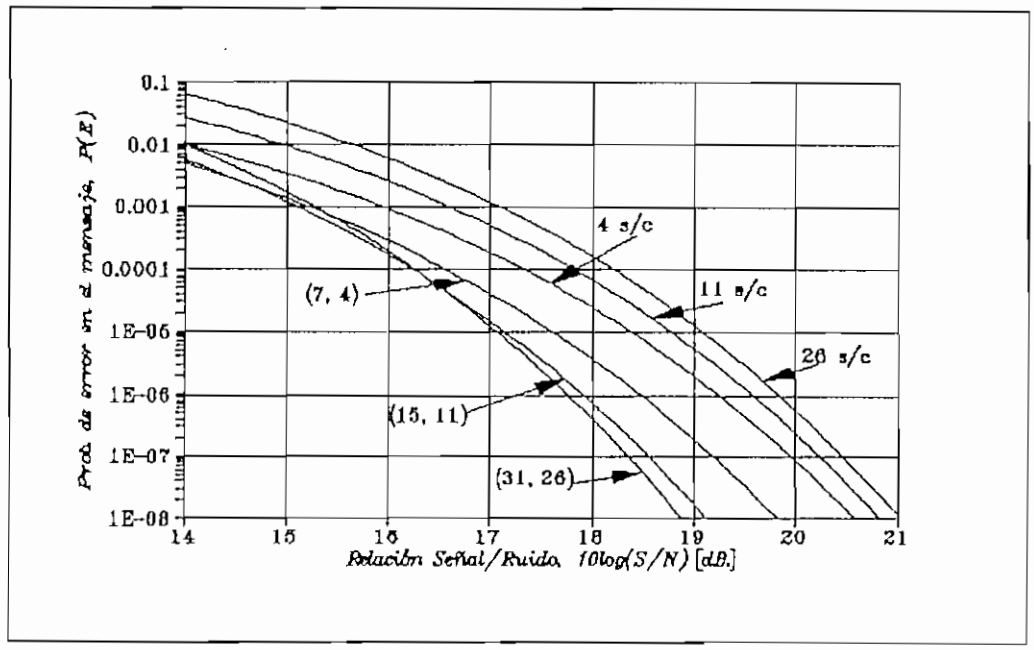


FIGURA 3.19 P(E) vs. S/N PARA 8-PSK.

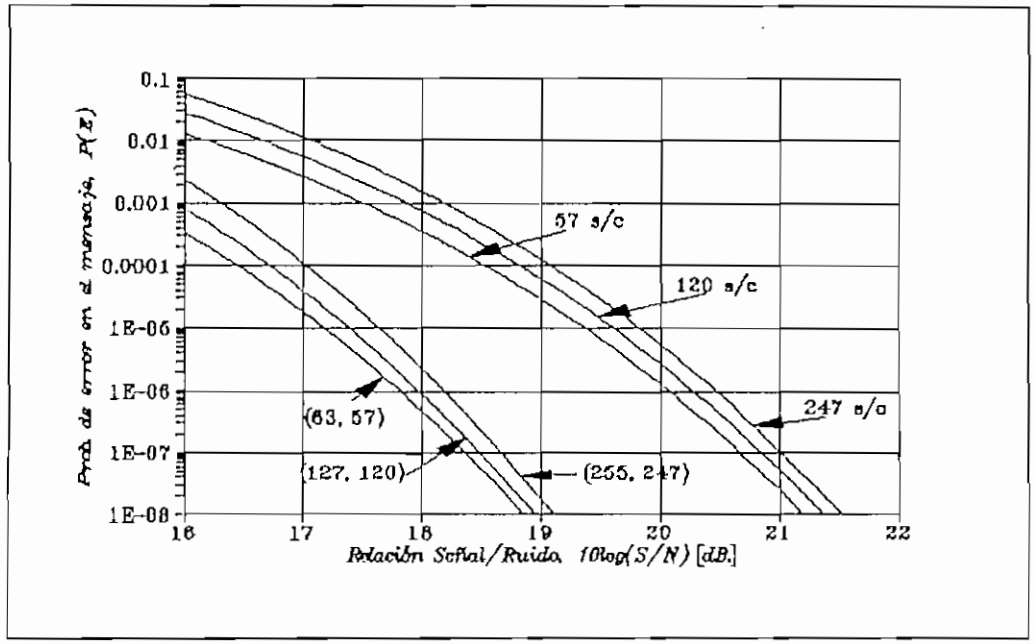


FIGURA 3.20 P(E) vs. S/N PARA 8-PSK.

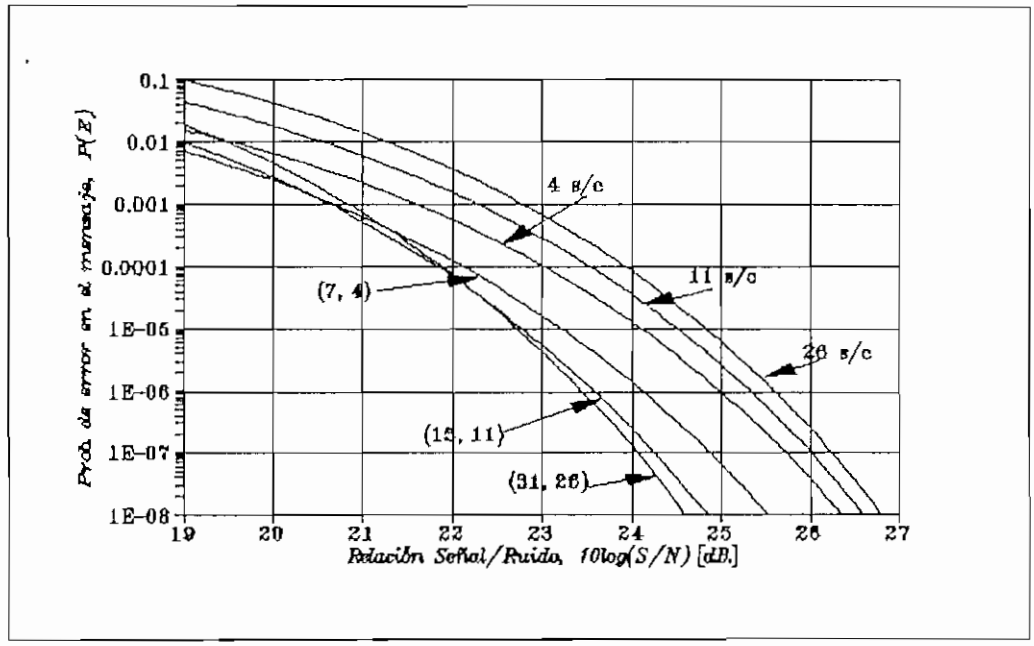


FIGURA 3.21 P(E) vs. S/N PARA 16-PSK.

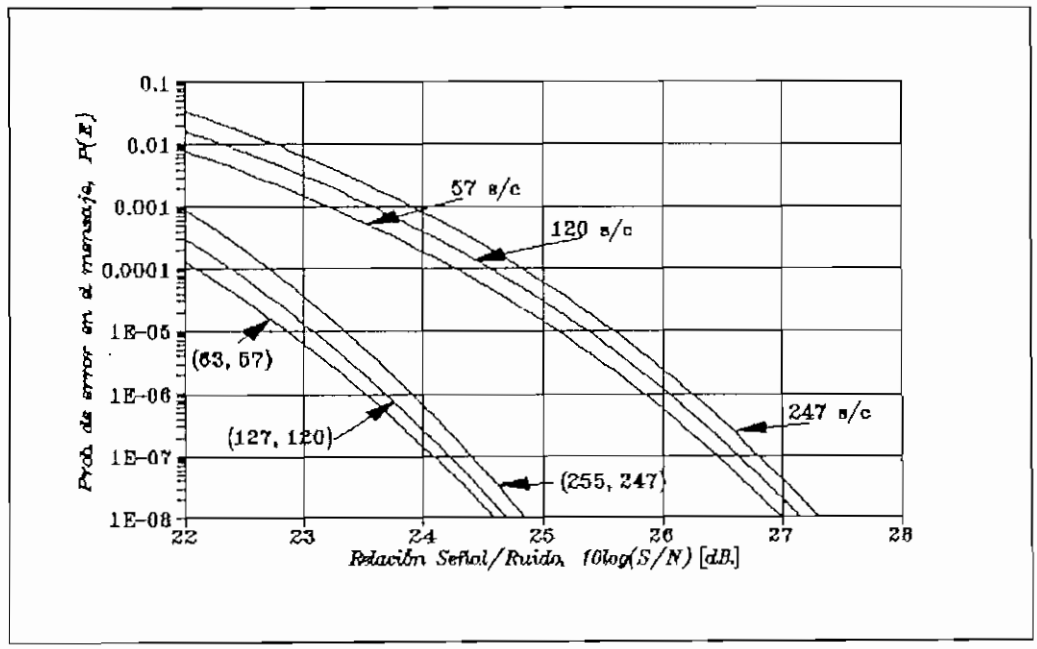


FIGURA 3.22 P(E) vs. S/N PARA 16-PSK.

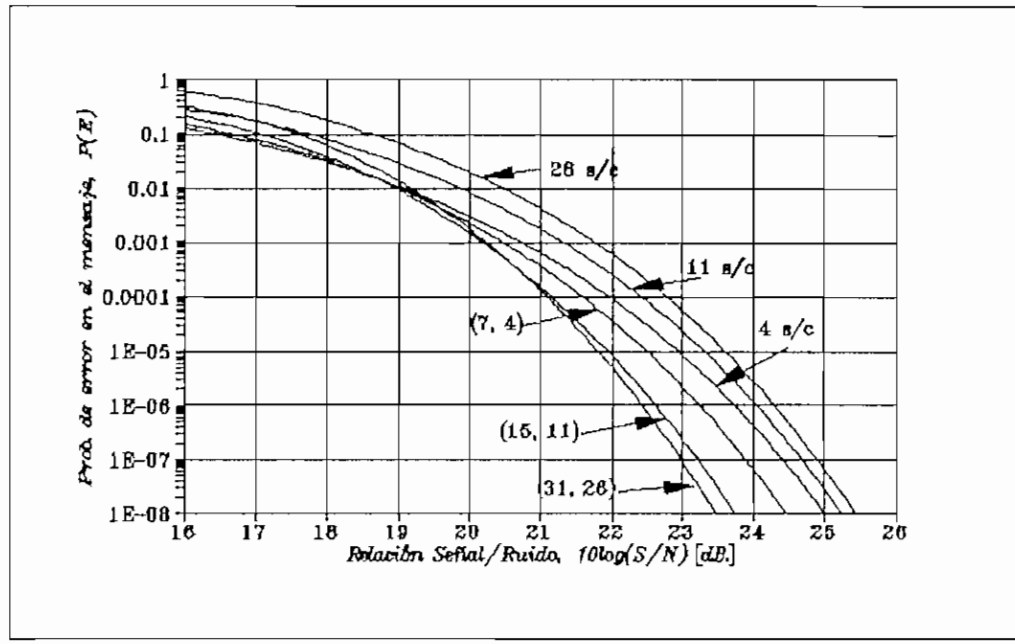


FIGURA 3.23 P(E) vs. S/N PARA 16-QAM.

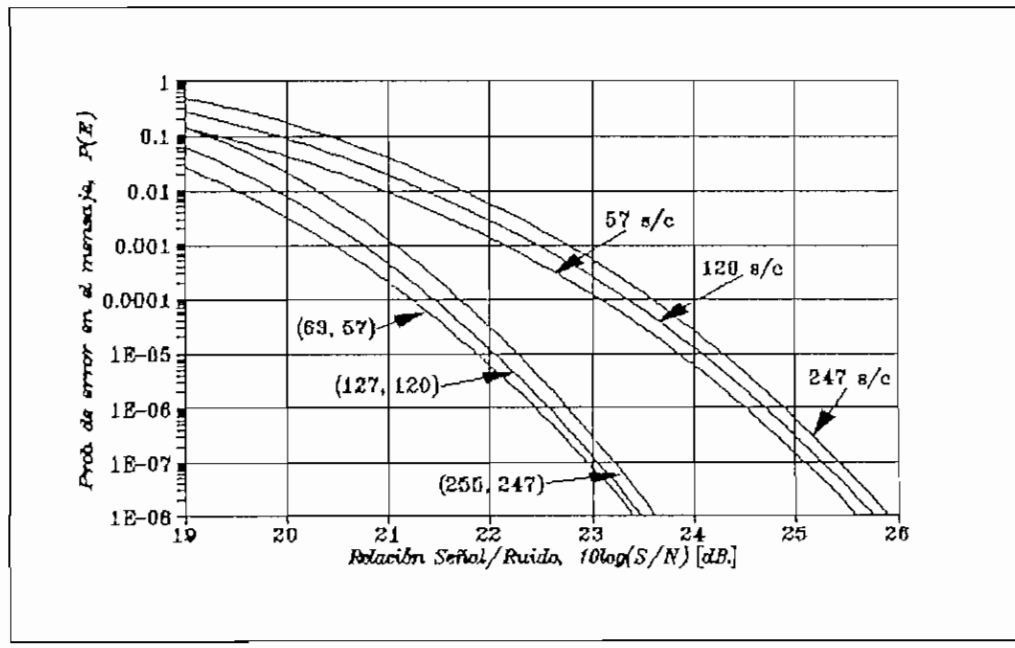


FIGURA 3.24 P(E) vs. S/N PARA 16-QAM.

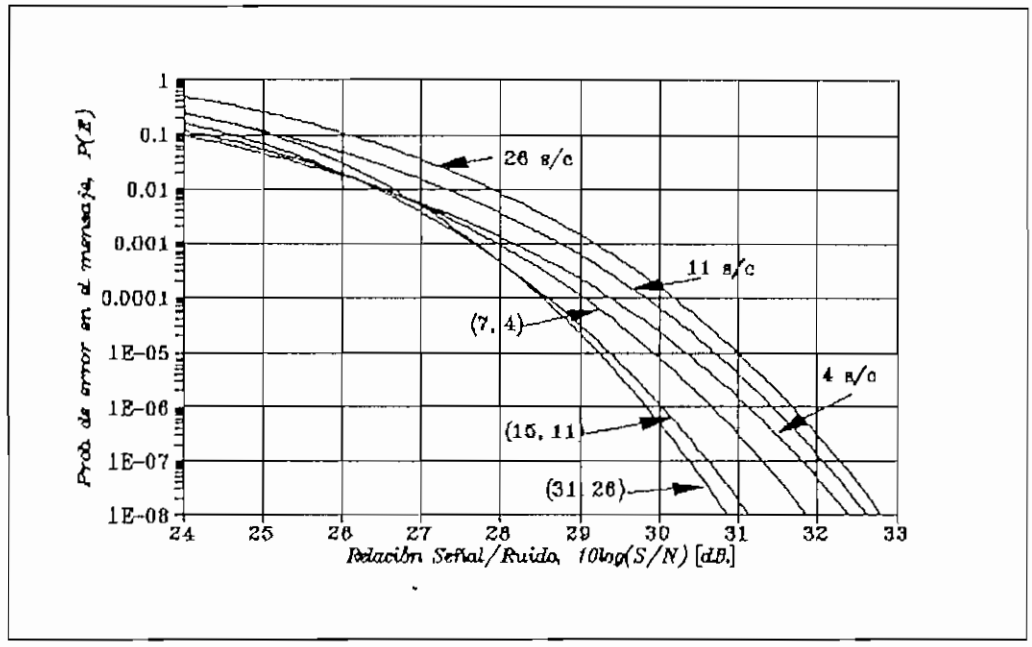


FIGURA 3.25 P(E) vs. S/N PARA 64-QAM.

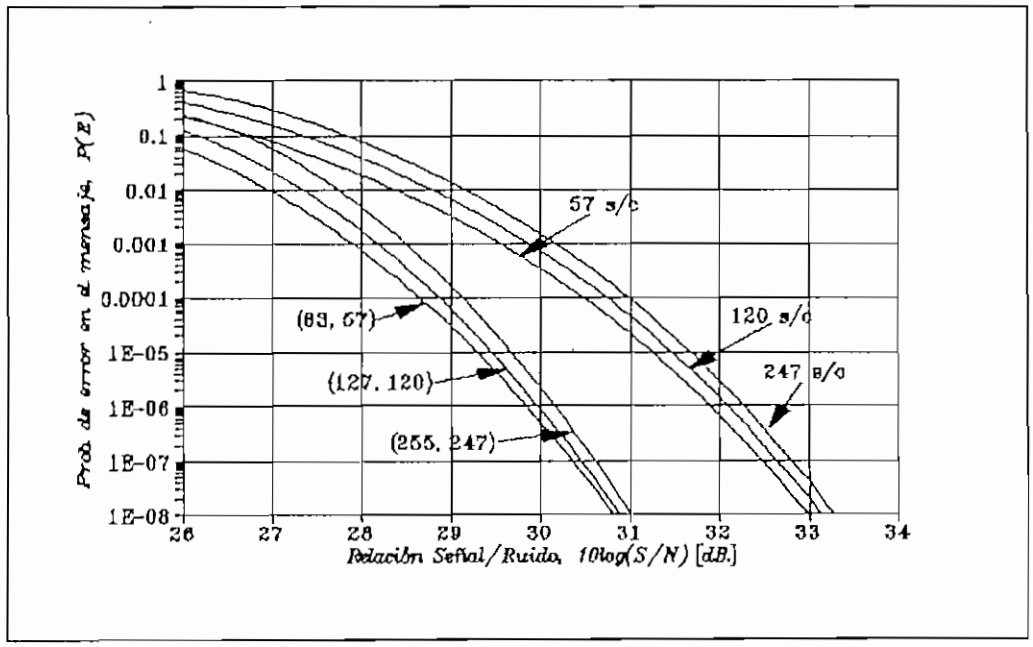


FIGURA 3.26 P(E) vs. S/N PARA 64-QAM.

En base a estas curvas se puede analizar mejor la relación que existe entre el código bloque lineal y la potencia de transmisión para obtener una variación en la tasa de bits erróneos p y con esta a su vez la mejora de la probabilidad de error en la palabra mensaje P(E), en un enlace de comunicación.

Si consideramos un sistema no codificado M-PSK y utilizando la figura 3.9, obtenemos la tabla 3.3 con los valores de la relación S/N para varios valores de la tasa de bits erróneos BER.

TABLA 3.3 VARIACION DE LA S/N RESPECTO DE LA BER PARA MODULACIONES 4-PSK, 8-PSK Y 16-PSK

BER	Relación señal a ruido S/N		
	4-PSK	8-PSK	16-PSK
$2*10^{-4}$	11.0	16.0	21.6
$1*10^{-4}$	11.5	16.6	22.3
$4*10^{-5}$	12.0	17.1	22.8
$2*10^{-5}$	12.4	17.5	23.2
$1.5*10^{-5}$	12.7	17.9	23.6
$1*10^{-5}$	13.0	18.2	23.9

Con estos valores de S/N podemos utilizar las figuras 3.17, 3.18, 3.19, 3.20, 3.21 y 3.22 para obtener los valores de la P(E) para sistemas no codificados (s/c) si consideramos que para cada fila de la tabla 3.3 se

envía mensajes de longitud k igual a 4, 11, 26, 57, 120 y 247 respectivamente.

Los valores de la BER han sido escogidos para que con los valores obtenidos de S/N se obtenga el valor de $P(E) = 10^{-3}$, Como se puede apreciar en las figuras indicadas, tal como se presenta en la tabla 3.4.

TABLA 3.4 VARIACION DE LA S/N RESPECTO DEL MENSAJE PARA MODULACIONES 4-PSK, 8-PSK Y 16-PSK.

Longitud del mensaje	Relación señal a ruido S/N			P(E) sistema no cod.
	4-PSK	8-PSK	16-PSK	
4	11.0	16.0	21.6	10^{-3}
11	11.5	16.6	22.3	10^{-3}
26	12.0	17.1	22.8	10^{-3}
57	12.4	17.5	23.2	10^{-3}
120	12.7	17.9	23.6	10^{-3}
247	13.0	18.2	23.9	10^{-3}

Estos resultados de P(E) para sistemas no codificados son importantes obtenerlos ya que hay que compararlos con los respectivos valores al realizar la codificación teniendo en cuenta la misma longitud de la palabra mensaje tanto para el sistema no codificado como para el codificado.

De las curvas de las figuras 3.17, 3.18, 3.19, 3.20,

3.21 y 3.22 se obtiene la tabla 3.5, la cual presenta los valores de la P(E) que se obtienen al codificar el flujo de bits con códigos bloque lineales que corrigen un solo error, si se tiene como referencia el valor de la P(E) inicial de 10^{-3} para sistemas no codificados obtenidos en la tabla 3.4. Además, en esta tabla se puede apreciar que el valor de la S/N para cada caso se mantiene.

TABLA 3.5 VARIACION DE LA P(E) CON LOS CODIGOS BLOQUE

Código Bloque Lineal (n,k) t = 1	Probabilidad de error en la palabra mensaje P(E)					
	4-PSK		8-PSK		16-PSK	
	P(E)	S/N	P(E)	S/N	P(E)	S/N
(7,4)	$3*10^{-4}$	11.0	$3*10^{-4}$	16.0	$3*10^{-4}$	21.6
(15,11)	$4*10^{-5}$	11.5	$4*10^{-5}$	16.6	$4*10^{-5}$	22.3
(31,26)	$1*10^{-5}$	12.0	$1*10^{-5}$	17.1	$1*10^{-5}$	22.8
(63,57)	$3*10^{-6}$	12.4	$3*10^{-6}$	17.5	$3*10^{-6}$	23.2
(127,120)	$2*10^{-6}$	12.7	$2*10^{-6}$	17.9	$2*10^{-6}$	23.6
(255,247)	$1*10^{-6}$	13.0	$1*10^{-6}$	18.2	$1*10^{-6}$	23.9

Se puede notar también que en las tablas 3.4 y 3.5, se tiene que para cada tipo de código se utiliza un valor de la S/N obtenido en la tabla 3.3.

Si se utiliza la modulación M-QAM y se realiza las mismas consideraciones utilizadas para M-PSK, se obtendrá las mismas conclusiones, es decir que para

*determinado valor de P(E) de un sistema no codificado, se obtiene una mejora de este valor al realizar la codificación, independientemente del valor de los niveles de la modulación, ya sea **16-QAM** o **64-QAM**.*

En el siguiente punto de este capítulo, se presenta resultados para P(E) considerando que se tiene un valor fijo de potencia de transmisión P_t y por lo tanto fija la relación S/N.

3.3 INTERPRETACION DE RESULTADOS Y FACTIBILIDAD DE IMPLEMENTACION PRACTICA.

Considerando el enlace presentado en la figura 3.11, es posible aplicar el objetivo planteado de este trabajo. Para lo cual se realizará los siguientes pasos:

A) Cálculo del valor de la relación E_b/N_o del radioenlace para valores determinados de la potencia de transmisión P_t, la velocidad de transmisión de los bits R_b, ganancias y pérdidas. De acuerdo a lo indicado en el punto 1.4 del capitulo 1.

B) Una vez que se ha determinado el valor de E_b/N_o

y teniendo en cuenta el tipo de modulación utilizado en el radioenlace, se calculará el valor de la probabilidad de error en los símbolos p_s y a su vez la probabilidad de error en los bits p (BER). Las ecuaciones (3.26) y (3.27) para modulación PSK mientras que las ecuaciones (3.28) y (3.29) para modulación QAM permiten calcular ya el valor de p para los sistemas no codificados. Para sistemas codificados (con códigos bloque lineales) se utilizará las ecuaciones (3.31) para M-PSK (M > 2) y (3.32) para M-QAM (M ≥ 16).

C) *Determinación de la calidad del radioenlace. Para los sistemas codificados será necesario calcular la probabilidad de no detección de errores Pu(E) y la probabilidad de error en el mensaje P(E), para lo cual se utilizará el programa desarrollado en el punto 2.2.2 del capítulo 2. Para los sistemas no codificados únicamente se calculará la probabilidad de error en los bits p (BER). Se puede realizar una comparación del valor de P(E) para los dos sistemas, para lo cual se considerará que se va ha transmitir k bits. en el caso no codificado se enviará estos bits al transmisor, mientras que para el codificado se enviará al codificador y se obtendrá n bits que*

ingresarán al trasmisor. Utilizando las ecuaciones (2.20) para el no codificado y (2.23) para el codificado se obtendrá la comparación entre estos dos sistemas.

Conocidos ya estos valores, se realizará el estudio de los ajustes que se crean necesarios. Para nuestro caso se considerará la posibilidad de variar la potencia de transmisión P_t y el código bloque utilizado (n,k). Si es un sistema no codificado, primero se insertará un codificador.

Para mayor claridad de los resultados obtenidos, se presenta como ejemplo el enlace, que corresponde a un enlace ascendente satelital[5].

Datos:

P_t = 13.5 W (11.3 dBW).

G_t = 35.2 dB.

frecuencia = 13.775 GHz. (banda Ku)

distancia = 38.000 Km

Pérdida en el espacio libre L_p

L_p (dB) = 32.5 + 20 log f (MHz) + 20 log d (Km)

L_p = 206.8 dB.

---

[5] Enciclopedia de la ELECTRONICA Ingeniería y Técnica, pags 1536 – 1538.

Pérdida por el direccionamiento de la antena = 0.5 dB.

Pérdida por la polarización de la antena = 0.2 dB.

Ganancia de la antena receptora de la Estación Espacial = 50.1 dB.

Temperatura de ruido del sistema receptor de la Estación Espacial, T_s (440°K) = 26.4 dB°K.

Constante de Boltzmann, K = -228.6 dBW/°K-Hz.

Pérdida en el circuito receptor de la Estación Espacial = 2.5 dB.

Velocidad de transferencia de los bits, R_b (22.22 Mb/s) = 73.5 dBHz.

Utilizando la ecuación (1.40) se tiene:

E_b/N_o verdadero = 15.3 dB.

E_b/N_o necesario para una BER (valor práctico) de 10^{-5} = 12.3 dB.

Margen de desvanecimiento = 3 dB.

Considerando modulación Q-PSK y el valor verdadero de E_b/N_o, calculamos la probabilidad de error en los bits p (valor teórico de la BER), ecuación (3.27) para el sistema no codificado. Se considera una eficiencia del ancho de banda R_b/B = 3 b/s/Hz.

$$p = 3.3391 * 10^{-24}$$

Para el sistema codificado se considerará el código

(63,57) que corrige un solo error. La ecuación (3.31) permite calcular el valor de p.

$p = 4.4424 * 10^{-24}$

La eficiencia del sistema, luego de agregar un codificador, se apreciará al calcular la probabilidad de error en el mensaje P(E) para cada sistema.

Con el sistema no codificado, ecuación (2.20), $k = 57$:

$P(E) = 1.9033 * 10^{-22}$

Con el sistema codificado, ecuación (2.23), $n = 63$ y $k = 57$:

$P(E) = 3.8543 * 10^{-40}$

Comparando estos valores, se puede observar que la probabilidad de error en el mensaje ha mejorado por un factor de aproximadamente $5 * 10^{17}$, gracias al empleo de un código de bloques.

Si se desea obtener esta mejora sin utilizar un código de bloques, sino únicamente incrementando la potencia de transmisión. Con el valor de $P(E) = 3.8543 * 10^{-40}$

utilizando la ecuación (2.20) se obtiene el valor de $p = 6.7046 * 10^{-42}$ y de éste con la ecuación (3.27) se tiene $E_b/N_o = 17.843$ dB.

Por lo tanto el incremento que se realizará a la potencia de transmisión anterior será de 2.543 dbW.

Consideremos también el siguiente ejemplo:

Datos:

$P_r/N_o = 43776$

$R_b = 4800$ b/s.

$R_b/B = 1$ b/s/Hz

Modulación B-PSK y código bloque lineal (15,11).

Se tiene:

- Para el sistema no codificado:

$p = 1.02 * 10^{-5}$

$P(E) = 1.12 * 10^{-4}$

- Para el sistema codificado:

$p = 1.36 * 10^{-4}$

$P(E) = 1.94 * 10^{-4}$

En este ejemplo, la probabilidad de error en el mensaje ha mejorado en un factor de 58, con el código bloque.

Esta mejora se puede lograr sin utilizar el código bloque lineal, pero aumentando la potencia de 0.7 dBW. sobre el valor anterior.

Otro tipo de ejemplo sería en el cual se tenga la potencia de transmisión P_t fija, es decir también fijo el valor de la relación señal a ruido S/N. Consideremos que después de todos los cálculos de propagación con modulación 4-PSK se obtiene que S/N = 13 dB., por lo tanto con este valor se verificará si se es posible tener una mejora en el rendimiento del enlace utilizando codificación con códigos bloque lineales.

Cuando el enlace es no codificado, para S/N = 13 dB. la probabilidad en los bits $BER = 1*10^{-5}$ (figura 3.9).

Al utilizar códigos bloque lineales, para codificar la señal, se puede apreciar en la figura 3.11 que la probabilidad en los bits se degrada debido a que se

tiene un aumento en la velocidad de transmisión de los bits conforme el código utilizado.

La mejora en el rendimiento al utilizar la codificación se aprecia en la probabilidad de error en la palabra mensaje P(E), comparando los valores que se obtiene tanto para el sistema codificado como para el sistema no codificado. Figuras 3.17 y 3.18.

Los resultados obtenidos para el valor de S/N = 13 dB. se presentan en la tabla 3.6 en la que se resume los valores de la probabilidad de error en la palabra mensaje P(E) de los dos sistemas codificado y no codificado.

TABLA 3.6 VARIACION DE P(E) AL UTILIZAR CODIFICACION

CODIGO BLOQUE (n,k) t = 1	MODULACION 4-PSK	
	P(E) sin codif.	P(E) con codif.
(7,4)	$2*10^{-5}$	$3*10^{-6}$
(15,11)	$4*10^{-5}$	$5*10^{-7}$
(31,26)	$1*10^{-4}$	$2*10^{-7}$
(63,57)	$2*10^{-4}$	$2*10^{-7}$
(127,120)	$4*10^{-4}$	$4*10^{-7}$
(255,247)	$1*10^{-3}$	$1*10^{-6}$

Cuando los valores de n y k son altos, se tiene un

Utilizamos los datos anteriores con modulación 16-PSK se tiene:

- Para el sistema no codificado:

$p = 1.5 * 10^{-4}$

$P(E) = 2 * 10^{-3}$

- Para el sistema codificado:

$p = 1 * 10^{-3}$

$P(E) = 6 * 10^{-3}$

Se tiene una mejora en un factor de 33.

Este ejemplo nos indica que los sistemas con modulación QAM son mejores que los sistemas PSK de igual número de niveles M-ario de modulación, bajo las mismas características del enlace.

Los resultados de estos ejemplos y las figuras presentadas en el punto anterior de este capítulo, nos permiten concluir que no se puede obtener una relación general entre el tipo de código bloque utilizado (o el

aumento de los bits de paridad) con el incremento de potencia en el transmisor para obtener una mejoría en el rendimiento de un enlace, disminuyendo la probabilidad de error que se pueda obtener a la salida.

Por la no linealidad de las curvas, el análisis se deberá realizar para condiciones iniciales conocidas, como:

- Relación E_b/N_o, (Potencia de transmisión P_t)
- Tipo de modulación M-PSK o M-QAM.
- Eficiencia del ancho de banda R_b/B.

Para los sistemas codificados, se puede utilizar el programa desarrollado en el capítulo 2 para obtener el valor de P(E) conocido el valor de p.

El estudio realizado, se lo puede implementar en forma práctica en enlaces donde los equipos transmisores tienen la potencia de transmisión fija, o con un rango de variación pequeño. Si se quiere disminuir los errores producidos en este tipo de enlaces, se tiene las siguientes alternativas:

- Incrementar la potencia si el equipo lo permite.

- Cambio del equipo si no tiene variación de la potencia con uno que si tenga (o uno que incluya codificación para el control de errores).
- Aumentar un codificador y decodificador externos en el enlace.

Con la última opción se necesita conseguir el codificador y decodificador de acuerdo a los resultados obtenidos al realizar el estudio que se indica en esta Tesis. Es posible que estos equipo no existan y se tenga que diseñarlos con los costos y tiempo que esto implica.

GLOSARIO DE TERMINOS UTILIZADOS.

$Pu(E)$: Probabilidad de no detección de errores.

$P(E)$: Probabilidad de error en la palabra mensaje.

p : Probabilidad de transición del BSC.

Probabilidad de error en los bits.

Tasa de bits erróneos, BER.

n : Longitud de la palabra código.

k : Longitud de la palabra mensaje.

E_b : Energía en los bits.

N_o : Densidad de ruido.

R_b : Velocidad de transmisión de los bits.

B : Ancho de banda a la entrada del detector.

CAPITULO IV.

CONCLUSIONES.

4.1 CONCLUSIONES.

En un radioenlace, el margen de desvanecimiento se obtiene respecto de una referencia que presenta el fabricante del equipo. Esta puede ser nivel mínimo de la potencia de recepción P_r, relación E_b/N_o, o cualquier que tenga que ver con la señal recibida respecto de la tasa de bits erróneos BER.

El margen de desvanecimiento nos permite determinar la confiabilidad del radioenlace.

Para calcular un enlace, se debe disponer de todos los parámetros necesarios para con estos determinar la relación S/N que interesa para realizar el estudio teórico de la tasa de bits erróneos.

Existe varias causas para que se produzcan errores en la transmisión digital, de las cuales la causa más predominante es el ruido blanco gaussiano.

Se debe tener presente que el Teorema de Shannon, algunas veces llamado teorema fundamental de la teoría de la información, establece que, para un canal discreto sin memoria (cada símbolo es perturbado por ruido, independientemente de los demás símbolos) con capacidad C y una fuente con velocidad positiva R_b, donde $R_b < C$, existe un código tal que la salida de la fuente puede transmitirse por el canal con una probabilidad de error arbitrariamente pequeña.

De este modo, el teorema de Shannon predice la transmisión esencialmente libre de error en presencia de ruido.

Por desgracia, este teorema solo menciona la existencia de códigos y no especifica la forma de construirlos.

Algunos resultados intermedios como el cálculo de la probabilidad de error en los bits, en sistemas codificados, dan una apariencia contradictoria sobre el objetivo de disminuir la tasa de errores. El objetivo de los sistemas codificados, es la disminución de los errores en la palabra mensaje P(E), que equivale a la BER de los bits de información.

En la práctica es posible realizar estas variaciones en un mismo equipo ya que están diseñados para dar estas facilidades, como por ejemplo el ComStream PSK Digital Satellite Modem. Este modem trabaja con velocidades de 9.6 Kbps a 2.048 Mbps, con modulación B-PSK o Q-PSK, con codificación convolucional soporta códigos con tasa 1/2 y 3/4 y decodificación secuencial y también puede operar sin codificación.

En esta Tesis se puede comprobar que si se aumenta un codificador en el enlace, se disminuye la probabilidad de error en la palabra mensaje. Mientras el código tenga una longitud tal que la eficiencia k/n tienda a 1, disminuye más la probabilidad de error en la palabra mensaje (BER en los bits de información) sin necesidad de variar la potencia de transmisión. Esto se puede apreciar en las curvas para sistemas codificados.

Al utilizar un código bloque lineal (n,k), se tiene ya el número de bits de control n−k que se está utilizando y también la eficiencia k/n correspondiente a este código. Es este valor de eficiencia el que influye en la disminución de la probabilidad de error en la palabra mensaje P(E).

Cuando se utiliza sistemas con modulación M-arios, mientras mayor es el número de niveles M de la modulación, se necesita aumentar la potencia (por lo tanto aumenta la relación S/N) ya que el número de bits por símbolo aumenta, para mantener la misma BER.

Cuando se utiliza códigos bloque lineales para disminuir los errores, se debe estudiar la probabilidad de error en la palabra mensaje P(E), tanto para sistemas codificados como no codificados. Consideremos que se desea disminuir la P(E) de 10^{-3} a 10^{-6} o a $4*10^{-7}$, utilizando codificación, para lo que se debe tener en cuenta que se envía mensajes de longitud k bits. Las figuras 3.18, 3.20, 3.22, 3.24 y 3.26 permiten determinar el tipo de código, correspondiente para cada tipo de modulación, que satisface el requerimiento planteado y también la equivalencia en el incremento de la relación S/N que se necesitaría para obtener el mismo efecto sin

codificación manteniendo los mensajes de longitud k.

Modulación	S/N	P(E)	bits de mensaje k	Para obtener $P(E) = 10^{-6}$ se requiere	
				Incrementar S/N	Utilizar código
4-PSK	13.00	10^{-3}	247	2.25	(255,247)
8-PSK	18.20	10^{-3}	247	2.50	(255,247)
16-PSK	23.90	10^{-3}	247	2.40	(255,247)
16-QAM	22.75	10^{-3}	247	2.10	(255,247)
64-QAM	30.20	10^{-3}	247	2.10	(255,247)

Modulación	S/N	P(E)	bits de mensaje k	Para $P(E) = 4*10^{-7}$ se requiere	
				Incrementar S/N	Utilizar código
4-PSK	13.00	$4*10^{-4}$	120	2.25	(127,120)
8-PSK	18.20	$4*10^{-4}$	120	2.50	(127,120)
16-PSK	23.90	$4*10^{-4}$	120	2.40	(127,120)
16-QAM	22.75	$4*10^{-4}$	120	2.10	(127,120)
64-QAM	30.20	$4*10^{-4}$	120	2.10	(127,120)

Se debe tener presente que cada tipo de modulación tiene su propio valor de S/N donde se cumple esta variación. Utilizando estos valores de S/N, en las figuras 3.9 y 3.10, se obtiene BER = $4 * 10^{-6}$ fijo.

El tabla que se presenta a continuación, contiene la equivalencia entre un código bloque lineal y el incremento de la S/N (incremento de P_e) para disminuir la probabilidad P(E), teniendo en cuenta un valor

inicial de la S/N = 13 dB. y modulación 4-PSK. Obviamente que para cada código utilizado se debe tener en cuenta la longitud k del mensaje.

P(E) Inicial	bits de mensaje, k	Incremento de S/N para P(E) final	Código (n,k) para P(E) final	P(E) final
$2*10^{-3}$	4	0.7	(7,4)	$3*10^{-6}$
$5*10^{-3}$	11	1.5	(15,11)	$5*10^{-7}$
$1*10^{-4}$	26	2.0	(31,26)	$3*10^{-7}$
$2*10^{-4}$	57	2.1	(63,57)	$2*10^{-7}$
$4*10^{-4}$	120	2.25	(127,120)	$4*10^{-7}$
$1*10^{-3}$	247	2.25	(255,247)	$1*10^{-6}$

Se debe tener presente que para el cálculo de la distribución de anchos de Hamming para códigos con valores grandes de n y k, se necesita gran cantidad de tiempo de utilización de un computador. O si se desea estos resultados, hay que recurrir a relaciones que proporcionan valores aproximados de esta distribución.

4.2 RECOMENDACIONES.

Se puede continuar con este estudio para otros sistemas de codificación, como los que utilizan códigos convolucionales. También para los diferentes tipos de diversidad en la recepción de la señal (diversidad de frecuencia, de espacio, etc.).

I.1 PROGRAMA COE_HEXA.BAS

Este programa permite presentar los polinomios geradores de los código BCH presentados en forma octal[1] a forma hexadecimal, pasándolos primeramente a forma binaria. los resultados son almacenados en el archivo TIPO_COD.TBL.

```
DECLARE SUB polingener (gl$, polin$)
DECLARE SUB polinhex (polinomio$, polin$)
DEFINT A-O, Q-Y
'
     CLS
     RESTORE Datos
     OPEN "A:\ARCH_TBL\TIPO_COD.TBL" FOR OUTPUT AS #1
          FOR J = 1 TO 232
               gl$ = ""
               READ n, k, t, poli
               FOR dat = 1 TO poli
                       READ datin$
                       gl$ = gl$ + datin$
               NEXT dat
               CALL polingener(gl$, polin$)
               PRINT #1, n, k, t
               PRINT #1, polin$
          NEXT J
     CLOSE #1
'
     END
'
Datos[2]:

"Aqui se debe ingresar los datos de n, k, t y del polinomio generador tal como se encuentran en el Anexo III"
```

---

[1] ERROR CONTROL CODING. Lin Shu, Appendix C. Polinomios Generadores de Códigos Primitivos Binarios BCH de ongitud de hasta n = $2^{10} - 1$, presentados en forma octal.

[2] Se utiliza los valores de los coeficientes de los códigos BCH de hasta $2^{10} - 1$ de longitud de la palabra código, presentados en el Anexo III.

```
SUB polingener (gl$, polin$)
'
     polinbin$ = ""
     FOR i = 1 TO LEN(gl$)
        IF MID$(gl$, i, 1) = "0" THEN
              Binario$ = "000"
           ELSEIF MID$(gl$, i, 1) = "1" THEN
             Binario$ = "001"
           ELSEIF MID$(gl$, i, 1) = "2" THEN
             Binario$ = "010"
           ELSEIF MID$(gl$, i, 1) = "3" THEN
             Binario$ = "011"
           ELSEIF MID$(gl$, i, 1) = "4" THEN
             Binario$ = "100"
           ELSEIF MID$(gl$, i, 1) = "5" THEN
             Binario$ = "101"
           ELSEIF MID$(gl$, i, 1) = "6" THEN
             Binario$ = "110"
           ELSEIF MID$(gl$, i, 1) = "7" THEN
             Binario$ = "111"
           ELSE PRINT "Error, numero es mayor que 7"
        END IF
        polinbin$ = polinbin$ + Binario$
     NEXT i
     indi = 1: cont = 1
     DO
          IF MID$(polinbin$, indi, 1) = "1" THEN EXIT DO
          indi = indi + 1: cont = cont + 1
     LOOP
     polinomio$ = MID$(polinbin$, cont, LEN(polinbin$) - cont + 1)
     z = LEN(polinomio$) / 4
     IF z - INT(z) = 0 THEN
          CALL polinhex(polinomio$, polin$)
        ELSE
          polinomio$ = MID$("0000", 1, (1 + INT(z)) * 4 - LEN(polinomio$)) + polinomio$
          CALL polinhex(polinomio$, polin$)
     END IF
'
END SUB

SUB polinhex (polinomio$, polin$)
'
     polin$ = ""
     FOR i = 1 TO LEN(polinomio$) STEP 4
        IF MID$(polinomio$, i, 4) = "0000" THEN
              hexa$ = "0"
           ELSEIF MID$(polinomio$, i, 4) = "0001" THEN
             hexa$ = "1"
           ELSEIF MID$(polinomio$, i, 4) = "0010" THEN
             hexa$ = "2"
           ELSEIF MID$(polinomio$, i, 4) = "0011" THEN
             hexa$ = "3"
```

```
            ELSEIF MID$(polinomio$, i, 4) = "0100" THEN
              hexa$ = "4"
            ELSEIF MID$(polinomio$, i, 4) = "0101" THEN
              hexa$ = "5"
            ELSEIF MID$(polinomio$, i, 4) = "0110" THEN
              hexa$ = "6"
            ELSEIF MID$(polinomio$, i, 4) = "0111" THEN
              hexa$ = "7"
            ELSEIF MID$(polinomio$, i, 4) = "1000" THEN
              hexa$ = "8"
            ELSEIF MID$(polinomio$, i, 4) = "1001" THEN
              hexa$ = "9"
            ELSEIF MID$(polinomio$, i, 4) = "1010" THEN
              hexa$ = "A"
            ELSEIF MID$(polinomio$, i, 4) = "1011" THEN
              hexa$ = "B"
            ELSEIF MID$(polinomio$, I, 4) = "1100" THEN
              hexa$ = "C"
            ELSEIF MID$(polinomio$, i, 4) = "1101" THEN
              hexa$ = "D"
            ELSEIF MID$(polinomio$, i, 4) = "1110" THEN
              hexa$ = "E"
            ELSEIF MID$(polinomio$, i, 4) = "1111" THEN
              hexa$ = "F"
        END IF
        polin$ = polin$ + hexa$
    NEXT i
'
END SUB
```

I.2 PROGRAMA ANCH_BLQ.BAS

Programa para calcular la distribución de los códigos bloque utilizando los polinomios generadores del programa anterior realizando una codificación sistemática. También toma en cuenta la longitud de la palabra mensaje **k** respecto del número de bits de paridad **n-k** para utilizar el código dual o no.

Si **n-k** < **k** se utilizará el código dual, para lo que habra que calcular el polinomio generador de este código.

Los resultados son almacenados en el archivo DATOS1.TBL

```
DECLARE SUB CalcAnchHam (nn%, kk%, G%(), N1%(), B%(), A#(), An$)
DECLARE SUB NumMensajes (N1%(), nn%, kk%)
DECLARE SUB PalabCodigo (nn%, kk%, G%(), N1%(), B%(), An$)
DECLARE SUB PolinGener (polinhex$, polinbin$)
DEFINT A-O, Q-Y
DEFDBL A
DIM N(250), K(250), T(250), G1$(250), A(1025), B(1025), G(1025), N1(1025)
'
   CLS
   j = 1
   OPEN "A:\ARCH_TBL\TIPO_COD.TBL" FOR INPUT AS #1
   DO WHILE NOT EOF(1)
      INPUT #1, N(j), K(j), T(j)
      INPUT #1, G1$(j)
      j = j + 1
   LOOP
   CLOSE #1
'
otrocod:
   INPUT "Ingresar el número de Código: "; Indi
```

```
    COLOR 0, 15: PRINT "  ** (ESCAPE) --> Termina  **  "
    COLOR 0, 15: PRINT "  ** (0) ó (o) --> Otro código  **  ": COLOR 7, 0
'
    OPEN "A:\ARCH_TBL\DATOS1.TBL" FOR APPEND AS #1
        DO WHILE Indi <= 232
            nn = N(Indi): kk = K(Indi)
            polinhex$ = G1$(Indi)
            CALL Polin Gener(polinhex$, polinbin$)
            j = LEN(polinbin$)
            FOR i = j TO 1 STEP -1
                G(i - j) = VAL(MID$(polinbin$, i, 1))
                j = j - 2
            NEXT i
'
            IF nn - kk < kk THEN
                Dual$ = "1": DG$ = "0"
                FOR Du = 1 TO nn - 1
                    Dual$ = Dual$ + "0"
                    IF Du <= kk THEN DG$ = DG$ + "0"
                NEXT Du
'
                Dual$ = Dual$ + "1":
                Ult = 0: Ult1 = nn + 1
                DO
                    i = Ult1
                    DO WHILE i >= 1
                        IF MID$(Dual$, i, 1) = "1" THEN Ult = i: i = 1
                        i = i - 1
                    LOOP
'
                    IF Ult = 0 THEN EXIT DO
                    IA = Ult - nn + kk: j = 0
                    MID$(DG$, IA, 1) = "1"
'
                    DO WHILE j <= nn - kk
                        IF G(j) = 1 THEN MID$(Dual$, IA + j, 1) = MID$(STR$(VAL(MID$(Dual$, IA + j, 1))
XOR G(j)), 2, 1)
                        j = j + 1
                    LOOP
                    Ult1 = Ult: Ult = 0
                LOOP
'
                j = kk + 1
                FOR i = 0 TO kk
                    G(i) = VAL(MID$(DG$, j, 1))
                    j = j - 1
                NEXT i
                kk = nn - kk
            END IF
'
            CALL CalcAnchHam(nn, kk, G(), N1(), B(), A(), An$)
            IF An$ = CHR$(27) OR An$ = "0" OR An$ = "o" THEN EXIT DO
```

```
'
                AnchHam$ = "1"
                FOR i = 1 TO nn
                    AnchHam$ = AnchHam$ + STR$(A(i))
                NEXT i
'
                IF N(Indi) - K(Indi) < K(Indi) THEN
                    PRINT #1, "Dual"
                  ELSE
                    PRINT #1, "NDual"
                END IF
                PRINT #1, N(Indi), K(Indi), T(Indi)
                PRINT #1, AnchHam$
                Indi = Indi + 1
            LOOP
        CLOSE #1
        IF An$ = "O" OR An$ = "o" THEN GOTO otrocod
        PRINT " Número del último código calculado: "; Indi - 1
'
        END


SUB CalcAnchHam (nn, kk, G(), N1(), B(), A(), An$)
'
        PRINT " --> CALCULANDO <-- "
        FOR i = 1 TO nn
            A(i) = 0
            IF i >= nn - kk + 1 THEN N1(i - 1) = 0
        NEXT i
'
        DO
            CALL PalabCodigo(nn, kk, G(), N1(), B(), An$)
            CALL NumMensajes(N1(), nn, kk)
'
            Contad = 0: ContAux = 0
            FOR IJ = 0 TO nn - 1
                IF B(IJ) = 1 THEN Contad = Contad + 1
                IF B(IJ) = 1 AND IJ >= nn - kk THEN ContAux = ContAux + 1
            NEXT IJ
            A(Contad) = A(Contad) + 1
        LOOP UNTIL ContAux = kk OR An$ = CHR$(27) OR An$ = "O" OR An$ = "o"
'
END SUB


SUB NumMensajes (N1(), nn, kk)
'
        Num = nn - 1
        DO WHILE Num >= nn - kk AND N1(Num) = 1
            N1(Num) = 0
            Num = Num - 1
        LOOP
```

```
    IF Num >= nn - kk THEN N1(Num) = 1
'
END SUB


SUB PalabCodigo (nn, kk, G(), N1(), B(), An$)
'
    Grado = nn - kk
    FOR i = 0 TO nn - 1
        B(i) = 0
        IF i >= nn - kk THEN B(i) = N1(i)
    NEXT i
'
    Ult1 = nn - 1
    DO
       DO WHILE B(Ult1) <> 1 AND Ult1 >= Grado
            Ult1 = Ult1 - 1
       LOOP
       IF Ult1 < Grado THEN EXIT DO
       IA = Ult1 - Grado: j = 0
       DO WHILE j <= Grado AND An$ <> CHR$(27) AND An$ <> "0" AND An$ <> "o"
            IF G(j) = 1 THEN B(IA + j) = B(IA + j) XOR 1
            j = j + 1
            An$ = INKEY$
       LOOP
    LOOP UNTIL An$ = CHR$(27) OR An$ = "0" OR An$ = "o"
'
    FOR Ind = nn - kk TO nn - 1
        B(Ind) = N1(Ind)
    NEXT Ind
'
END SUB


SUB PolinGener (polinhex$, polinbin$)
'
     polinbin$ = ""
     FOR i = 1 TO LEN(polinhex$)
        IF MID$(polinhex$, i, 1) = "0" THEN
              binario$ = "0000"
           ELSEIF MID$(polinhex$, i, 1) = "1" THEN
              binario$ = "0001"
           ELSEIF MID$(polinhex$, i, 1) = "2" THEN
              binario$ = "0010"
           ELSEIF MID$(polinhex$, i, 1) = "3" THEN
              binario$ = "0011"
           ELSEIF MID$(polinhex$, i, 1) = "4" THEN
              binario$ = "0100"
           ELSEIF MID$(polinhex$, i, 1) = "5" THEN
              binario$ = "0101"
           ELSEIF MID$(polinhex$, i, 1) = "6" THEN
              binario$ = "0110"
```

```
            ELSEIF MID$(polinhex$, i, 1) = "7" THEN
              binario$ = "0111"
            ELSEIF MID$(polinhex$, i, 1) = "8" THEN
              binario$ = "1000"
            ELSEIF MID$(polinhex$, i, 1) = "9" THEN
              binario$ = "1001"
            ELSEIF MID$(polinhex$, i, 1) = "A" THEN
              binario$ = "1010"
            ELSEIF MID$(polinhex$, i, 1) = "B" THEN
              binario$ = "1011"
            ELSEIF MID$(polinhex$, i, 1) = "C" THEN
              binario$ = "1100"
            ELSEIF MID$(polinhex$, i, 1) = "D" THEN
              binario$ = "1101"
            ELSEIF MID$(polinhex$, i, 1) = "E" THEN
              binario$ = "1110"
            ELSEIF MID$(polinhex$, i, 1) = "F" THEN
              binario$ = "1111"
            ELSE PRINT "Error, numero es mayor que F"
        END IF
        polinbin$ = polinbin$ + binario$
    NEXT i
'
END SUB
```

I.3 PROGRAMA ANCH_BCH.BAS

Programa para calcular la distribución de anchos de los códigos dual BCH que corrigen doble y triple errores, utilizando relaciones derivadas por Kasami[3].

Los resultados son almacenados en el archivo DATOS2.TBL

```
DECLARE SUB impar3 (m%, B#())
DECLARE SUB impar5 (m%, B#())
DECLARE SUB par4 (m%, B#())
DECLARE SUB par6 (m%, B#())
DEFINT A, C-O, Q-Z
DEFDBL B
DIM N(250), K(250), T(250), G1$(250), B(1030)
'
   CLS
   j = 1
   OPEN "A:\ARCH_TBL\TIPO_COD.TBL" FOR INPUT AS #1
      DO WHILE NOT EOF(1)
         INPUT #1, N(j), K(j), T(j)
         INPUT #1, G1$(j)
         j = j + 1
      LOOP
   CLOSE #1
'
   j = 12
   OPEN "A:\ARCH_TBL\DATOS2.TBL" FOR OUTPUT AS #1
      DO WHILE j <= 232
          IF T(j) = 2 OR T(j) = 3 THEN
                nn = N(j): tt = T(j)
                FOR i = 0 TO nn
                     B(i) = 0
                NEXT i
'
                m = LOG(nn + 1) / LOG(2)
                p = m / 2
                IF p - INT(p) = 0 THEN
                     IF m >= 4 AND tt = 2 THEN
```

---

[3] ERROR CONTROL CODING, Lin Shu, pags. 177 y 178

```
                        CALL par4(n, B())
                      ELSEIF n >= 6 AND tt = 3 THEN
                        CALL par6(n, B())
                    END IF
                  ELSE
                    IF n >= 3 AND tt = 2 THEN
                        CALL impar3(n, B())
                      ELSEIF n >= 5 AND tt = 3 THEN
                        CALL impar5(n, B())
                     END IF
                END IF
'
                AnchHan$ = ""
                FOR i = 0 TO nn
                     AnchHan$ = AnchHan$ + STR$(B(i))
                NEXT i
'
                PRINT #1, "Dual"
                PRINT #1, N(j), K(j), T(j)
                PRINT #1, AnchHan$
'
            END IF
            j = j + 1
        LOOP
    CLOSE #1
'
    END


SUB impar3 (n, B())
'
    aux = 0
    B(aux) = 1
    aux = 2 ^ (n - 1) - 2 ^ ((n + 1) / 2 - 1)
    B(aux) = (2 ^ (n - 2) + 2 ^ ((n - 1) / 2 - 1)) * (2 ^ n - 1)
    aux = 2 ^ (n - 1)
    B(aux) = (2 ^ n - 2 ^ (n - 1) + 1) * (2 ^ n - 1)
    aux = 2 ^ (n - 1) + 2 ^ ((n + 1) / 2 - 1)
    B(aux) = (2 ^ (n - 2) - 2 ^ ((n - 1) / 2 - 1)) * (2 ^ n - 1)
'
END SUB

SUB impar5 (n, B())
'
    aux = 0
    B(aux) = 1
    aux = 2 ^ (n - 1) - 2 ^ ((n + 1) / 2)
    B(aux) = 2 ^ ((n - 5) / 2) * (2 ^ ((n - 3) / 2) + 1) * (2 ^ (n - 1) - 1) * (2 ^ n - 1) / 3
    aux = 2 ^ (n - 1) - 2 ^ ((n - 1) / 2)
    B(aux) = 2 ^ ((n - 3) / 2) * (2 ^ ((n - 1) / 2) + 1) * (5 * 2 ^ (n - 1) + 4) * (2 ^ n - 1) / 3
    aux = 2 ^ (n - 1)
    B(aux) = (9 * 2 ^ (2 * n - 4) + 3 * 2 ^ (n - 3) + 1) * (2 ^ n - 1)
```

```
    aux = 2 ^ (n - 1) + 2 ^ ((n - 1) / 2)
    B(aux) = 2 ^ ((n - 3) / 2) * (2 ^ ((n - 1) / 2) - 1) * (5 * 2 ^ (n - 1) + 4) * (2 ^ n - 1) / 3
    aux = 2 ^ (n - 1) + 2 ^ ((n + 1) / 2)
    B(aux) = 2 ^ ((n - 5) / 2) * (2 ^ ((n - 3) / 2) - 1) * (2 ^ (n - 1) - 1) * (2 ^ n - 1) / 3
'
END SUB

SUB parm4 (n, B())
'
    aux = 0
    B(aux) = 1
    aux = 2 ^ (n - 1) - 2 ^ ((n + 2) / 2 - 1)
    B(aux) = 2 ^ ((n - 2) / 2 - 1) * (2 ^ ((n - 2) / 2) + 1) * (2 ^ n - 1) / 3
    aux = 2 ^ (n - 1) - 2 ^ (n / 2 - 1)
    B(aux) = 2 ^ ((n + 2) / 2 - 1) * (2 ^ (n / 2) + 1) * (2 ^ n - 1) / 3
    aux = 2 ^ (n - 1)
    B(aux) = (2 ^ (n - 2) + 1) * (2 ^ n - 1)
    aux = 2 ^ (n - 1) + 2 ^ (n / 2 - 1)
    B(aux) = 2 ^ ((n + 2) / 2 - 1) * (2 ^ (n / 2) - 1) * (2 ^ n - 1) / 3
    aux = 2 ^ (n - 1) + 2 ^ ((n + 2) / 2 - 1)
    B(aux) = 2 ^ ((n - 2) / 2 - 1) * (2 ^ ((n - 2) / 2) - 1) * (2 ^ n - 1) / 3
'
END SUB

SUB parm6 (n, B())
'
    aux = 0
    B(aux) = 1
    aux = 2 ^ (n - 1) - 2 ^ ((n + 4) / 2 - 1)
    B(aux) = (2 ^ (n - 1) + 2 ^ ((n + 4) / 2 - 1)) * (2 ^ n - 4) * (2 ^ n - 1) / 960
    aux = 2 ^ (n - 1) - 2 ^ ((n + 2) / 2 - 1)
    B(aux) = 7 * (2 ^ (n - 1) + 2 ^ ((n + 2) / 2 - 1)) * 2 ^ n * (2 ^ n - 1) / 48
    aux = 2 ^ (n - 1) - 2 ^ (n / 2 - 1)
    B(aux) = 2 * (2 ^ (n - 1) + 2 ^ (n / 2 - 1)) * (3 * 2 ^ n + 8) * (2 ^ n - 1) / 15
    aux = 2 ^ (n - 1)
    B(aux) = (29 * 2 ^ (2 * n) - 4 * 2 ^ n + 64) * (2 ^ n - 1) / 64
    aux = 2 ^ (n - 1) + 2 ^ (n / 2 - 1)
    B(aux) = 2 * (2 ^ (n - 1) - 2 ^ (n / 2 - 1)) * (3 * 2 ^ n + 8) * (2 ^ n - 1) / 15
    aux = 2 ^ (n - 1) + 2 ^ ((n + 2) / 2 - 1)
    B(aux) = 7 * (2 ^ (n - 1) - 2 ^ ((n + 2) / 2 - 1)) * 2 ^ n * (2 ^ n - 1) / 48
    aux = 2 ^ (n - 1) + 2 ^ ((n + 4) / 2 - 1)
    B(aux) = (2 ^ (n - 1) - 2 ^ ((n + 4) / 2 - 1)) * (2 ^ n - 4) * (2 ^ n - 1) / 960
'
END SUB
```

I.4 PROGRAMA UNIFICA.BAS

Este programa permite unificar el contenido de los archivos DATOS1.TBL y DATOS2.TBL en el archivo TIP_COD1.TBL. Se tiene en cuenta si se ha realizado el cálculo con el código dual o no.

```
DIM N(250), K(250), T(250), AnchHam$(250), RP$(250)
'
   CLS
   j = 1
   OPEN "A:\ARCH_TBL\DATOS1.TBL" FOR INPUT AS #1
      DO WHILE NOT EOF(1)
         INPUT #1, RP$(j)
         INPUT #1, N(j), K(j), T(j)
         INPUT #1, AnchHam$(j)
         j = j + 1
      LOOP
   CLOSE #1
'
   OPEN "A:\ARCH_TBL\DATOS2.TBL" FOR INPUT AS #1
      DO WHILE NOT EOF(1)
         INPUT #1, RP$(j)
         INPUT #1, N(j), K(j), T(j)
         INPUT #1, AnchHam$(j)
         j = j + 1
      LOOP
   CLOSE #1
'
   K = j - 1
   FOR i = 1 TO K
      FOR j = i + 1 TO K
           IF N(i) > N(j) THEN
                AuxN = N(i): N(i) = N(j): N(j) = AuxN
                AuxK = K(i): K(i) = K(j): K(j) = AuxK
                AuxT = T(i): T(i) = T(j): T(j) = AuxT
                AuxRP$ = RP$(i): RP$(i) = RP$(j): RP$(j) = AuxRP$
                AuxAH$ = AnchHam$(i): AnchHam$(i) = AnchHam$(j): AnchHam$(j) = AuxAH$
              ELSEIF N(i) = N(j) THEN
                IF T(i) > T(j) THEN
                     AuxT = T(i): T(i) = T(j): T(j) = AuxT
                     AuxK = K(i): K(i) = K(j): K(j) = AuxK
                     AuxRP$ = RP$(i): RP$(i) = RP$(j): RP$(j) = AuxRP$
                     AuxAH$ = AnchHam$(i): AnchHam$(i) = AnchHam$(j): AnchHam$(j) = AuxAH$
                END IF
```

```
            END IF
        NEXT j
    NEXT i
'
    OPEN "A:\ARCH_TBL\TIP_COD1.TBL" FOR OUTPUT AS #1
        j = 1
        DO WHILE j <= K
            PRINT #1, RP$(j)
            PRINT #1, N(j), K(j), T(j)
            PRINT #1, AnchHa#$(j)
            j = j + 1
        LOOP
    CLOSE #1
'
    END
```

I.5 PROGRAMA CODIGOS.BAS

Programa que permite obtener la codificación en forma sistemática, cálculo (gráficos) de la probabilidad de no detección de errores Pu(E) y de la probabilidad de error en la palabra mensaje P(E) en función de la probabilidad de transición del canal binario simétrico p (probabilidad de error en los bits, BER).

```
'
'    Programa Principal
'
DECLARE SUB cuadro1 (j1%, j2%, j3%, j4%, j5%, j6%)
DECLARE SUB Cuadro2 (i1%, i2%, j1%, j2%, codigo$, pinf!, psup!, tecla$)
DECLARE SUB DigitEscala (i%, j%, p1$)
DECLARE SUB Ingre1 (camino$, n%(), k%(), t%())
DECLARE SUB Ingre2 (camino$, n%(), k%(), t%(), g1$())
DECLARE SUB Ingre3 (camino$, nn%(), kk%(), tt%(), kg$(), AnchHam$())
DECLARE SUB Ingre4 (camino$, Ater$())
DECLARE SUB MovBloque (AM$(), ap$, xi1%, xf1%, yi1%, yf1%, y1%, y2%, y3%, y4%)
DECLARE SUB MovCursor (AM$, ap$, xi%, xf%, yi%, yf%, y%)
DECLARE SUB PalabCodigo (nn%, kk%, G%(), n1%(), b%())
DECLARE SUB PolinGener (polinhex$, polinbin$)
DECLARE SUB PuCodBlq (DistAnch$, dualid$, nn%, kk%, pinf!, psup!, PMayor!, PMenor!, Pu!())
DECLARE SUB PuCodHam (nn%, kk%, pinf!, psup!, PMayor!, PMenor!, Pu!())
DEFINT B-D, Q-Y
DEFDBL A
REM $STATIC
DIM g1$(233), AnchHam$(40), Ater$(10)
DIM n(233), k(233), t(233), Pu(105), PE(105), nn(40), Acombi(1025)
DIM kk(40), tt(40), kg$(40), a(1025), b(1025), G(1025), n1(1025)
'
    CLS
    CALL cuadro1(0, 15, 9, 25, 54, 19)
    LOCATE 11, 41: PRINT 'CODIGOS BLOQUE LINEALES'
    LOCATE 13, 31: PRINT 'Codificación en forma sistemática y gráficos'
    LOCATE 14, 31: PRINT 'de la probabilidad de no detección de error'
    LOCATE 15, 35: PRINT 'utilizando códigos bloque lineales.'
    LOCATE 19, 31: PRINT 'Realizado por: Luis A. Rivera S.    -    1994'
    COLOR 7, 0
    camino$ = '': video = 12: pfact1 = 1: pfact2 = 1
'
    DO
```

```
        LOCATE 22, 1: PRINT STRING$(80, " ")
        LOCATE 23, 1: PRINT STRING$(80, " ")
        LOCATE 22, 2: PRINT "ARCHIVO DE DATOS: Drive ->"
        LOCATE 23, 2: PRINT "                Camino ->"
        DO
            drive$ = INKEY$
        LOOP UNTIL drive$ = "C" OR drive$ = "c" OR drive$ = "A" OR drive$ = "a" OR drive$ = "B" OR drive$
= "b"
        LOCATE 22, 29: PRINT drive$
        LOCATE 23, 29: INPUT "", direct$
        DO
            salir$ = INKEY$
        LOOP UNTIL salir$ = CHR$(27) OR salir$ = CHR$(13)
        IF salir$ = CHR$(13) THEN EXIT DO
    LOOP
    camino$ = drive$ + ":\" + direct$
'
    AM$(1) = " 1 --> Tipos de Códigos (n,k,t) "
    AM$(2) = " 2 --> Codificación Sistemática "
    AM$(3) = " 3 --> Gráficos                 "
    AM$(4) = " 4 --> Salir del Programa       "
'
    AM1$(1) = " 1 -> Pu(E) para un código (n,k,t) "
    AM1$(2) = " 2 -> Pu(E) vs. (n-k) vs. p        "
    AM1$(3) = " 3 -> P(E) para un código (n,k,t)  "
    AM1$(4) = " 4 -> P(E) vs. (n-k) vs. p         "
'
MenuPrincip:
    CLS
    k4 = 20: k5 = 40: k6 = 17: y1 = 10
    GOSUB entrada
'
    LOCATE 7, 33: PRINT "MENU  PRINCIPAL"
    mensaje$ = "Seleccione una opción con " + CHR$(24) + " " + CHR$(25)
    mensaje1$ = "y presione <ENTER>"
    LOCATE k6 - 1, k4 + 2: PRINT TAB(k4 + (k5 - LEN(mensaje$)) / 2); mensaje$
    LOCATE k6, k4 + 2: PRINT TAB(k4 + (k5 - LEN(mensaje1$)) / 2); mensaje1$
'
    CALL MovBloque(AM$(), ap$, 25, 56, 10, 13, y1, 25, 4, 9)
    IF y1 - 9 = 4 OR ap$ = CHR$(27) THEN GOTO Fin
    IF y1 - 9 = 1 THEN
        CALL Ingre1(camino$, n(), k(), t())
        ELSEIF y1 - 9 = 2 THEN
        CALL Ingre2(camino$, n(), k(), t(), g1$())
    END IF
    ON (y1 - 9) GOSUB bloque1, bloque2, bloque3
'
bloque1:
    CLS
'
    k4 = 17: k5 = 44: k6 = 17
    GOSUB entrada
```

```
'
    LOCATE 7, 21: PRINT "Listado de los Códigos Bloque Lineales"
    LOCATE 10, 24: PRINT "Se indica los tipos de códigos"
    LOCATE 11, 24: PRINT "bloque lineales (n, k, t), BCH"
    LOCATE 12, 25: PRINT "de longitud menor que 1023."
    LOCATE 14, 24: COLOR 15, 0: PRINT " -> Listado de los códigos <- ": COLOR 0, 15
    LOCATE 16, 24: PRINT "Presione <ENTER> para continuar"
    LOCATE 17, 24: PRINT "<ESC> regresa pantalla anterior"
    COLOR 7, 0
'
    DO
        ap$ = INKEY$
    LOOP UNTIL ap$ = CHR$(13) OR ap$ = CHR$(27)
    IF ap$ = CHR$(27) THEN
        ERASE n, k, t
        RETURN MenuPrincip
    END IF
'
    CLS
    CALL cuadro1(15, 0, 1, 3, 73, 2)
'
    LOCATE 2, 18: PRINT "* *  C O D I G O S   ( n ,  k ,  t )  * *"
    FOR i = 0 TO 3
        CALL cuadro1(15, 0, 4, 20 * i + 1, 17, 21)
        CALL cuadro1(15, 0, 4, 20 * i + 2, 15, 5)
        LOCATE 5, 20 * i + 3: PRINT "  n    k    t"
    NEXT i
    COLOR 7, 0
'
    iaux = 0: Pagina = 1
    DO
        FOR i = 7 TO 21
            ival = i + iaux - 6
            FOR j = 0 TO 3
                LOCATE i, 20 * j + 2: PRINT "                "
                IF ival <= 232 THEN
                    LOCATE i, 20 * j + 2: PRINT n(ival)
                    LOCATE i, 20 * j + 8: PRINT k(ival)
                    LOCATE i, 20 * j + 14: PRINT t(ival)
                END IF
                ival = ival + 15
            NEXT j
        NEXT i
'
        COLOR 0, 15
        LOCATE 23, 2: PRINT " ** <ESC> Retorna pantalla anterior ** "
        LOCATE 23, 42: PRINT "  ** [M] ó [■] -->  Más Códigos **  "
        DO
            control$ = INKEY$
        LOOP UNTIL control$ = "M" OR control$ = "■" OR control$ = CHR$(27)
        COLOR 7, 0
        IF control$ = CHR$(27) THEN EXIT DO
```

```
'
        iaux = iaux + 60
        Pagina = Pagina + 1
        IF Pagina > 4 THEN Pagina = 1: iaux = 0
    LOOP
    GOTO bloque1
'
bloque2:
    CLS
    k4 = 17: k5 = 44: k6 = 17
    GOSUB entrada
'
    LOCATE 7, 29: PRINT "Codificación Sistemática"
    LOCATE 10, 25: PRINT "Ingrese el código a utilizarce:"
    LOCATE 12, 25: PRINT "long. del código,  n : "
    LOCATE 13, 25: PRINT "long. del mensaje, k : "
    LOCATE 14, 25: PRINT "bits que corrige,  t : "
    LOCATE 16, 25: PRINT "Ingresar los valores correctos"
    LOCATE 17, 25: PRINT "de las variables para continuar"
'
    j = 1
    DO
        LOCATE 12, 48: PRINT " "; n(j)
        LOCATE 13, 48: PRINT " "; k(j)
        LOCATE 14, 48: PRINT " "; t(j)
'
        ter$ = INKEY$
        IF ter$ = CHR$(13) OR ter$ = CHR$(27) THEN EXIT DO
        IF LEN(ter$) = 2 THEN
            IF ASC(RIGHT$(ter$, 1)) = 80 THEN
                j = j - 1
                IF j < 1 THEN j = 232
            END IF
            IF ASC(RIGHT$(ter$, 1)) = 72 THEN
                j = j + 1
                IF j > 232 THEN j = 1
            END IF
        END IF
    LOOP
    COLOR 7, 0
    IF ter$ = CHR$(27) THEN
        ERASE n, k, t, g1$
        RETURN MenuPrincip
    END IF
'
    nn = n(j): kk = k(j): tt = t(j): polinhex$ = g1$(j)
Otromen:
    CLS
    LOCATE 2, 1: COLOR 0, 15: PRINT " CODIGO SELECIONADO (n, k, t): "; nn; kk; tt: COLOR 7, 0
    LOCATE 4, 1: PRINT " Palabra Mensaje a Codificar ("; kk; " bits máximo ): "
    i = nn - kk: n1$ = "": palab$ = ""
    FOR ii = 0 TO kk
```

```
        n1(ii) = 0
    NEXT ii
'
    DO WHILE i <= nn - 1
        n1$ = INKEY$
        IF n1$ = CHR$(13) OR n1$ = CHR$(27) THEN EXIT DO
        IF n1$ = "0" OR n1$ = "1" THEN
            n1(i) = VAL(n1$)
            palab$ = palab$ + STR$(VAL(n1$))
            LOCATE 5, 1: PRINT palab$
            i = i + 1
        END IF
    LOOP
    IF n1$ = CHR$(27) THEN
        ERASE n, k, t, g1$
        RETURN MenuPrincip
    END IF
'
    CALL PolinGener(polinhex$, polinbin$)
    j = LEN(polinbin$)
    FOR i = j TO 1 STEP -1
        G(i - j) = VAL(MID$(polinbin$, i, 1))
        j = j - 2
    NEXT i
    CALL PalabCodigo(nn, kk, G(), n1(), b())
    PaCodigo$ = ""
    FOR i = 0 TO nn - 1
        PaCodigo$ = PaCodigo$ + STR$(b(i))
    NEXT i
    PRINT
    PRINT " Palabra Código ( de longitud "; nn; " bits ): "
    PRINT PaCodigo$
'
    PRINT
    COLOR 0, 15: PRINT "   ** <ESC> Retorna pantalla anterior **     ** [0] ó [o] —> Otro mensaje **  "
    COLOR 7, 0
    DO
        control$ = INKEY$
    LOOP UNTIL control$ = "0" OR control$ = "o" OR control$ = CHR$(27)
    IF control$ = "0" OR control$ = "o" THEN GOTO Otromen
    GOTO bloque2
'
bloque3:
    CLS
    k4 = 19: k5 = 42: k6 = 18: y1 = 11
    GOSUB entrada
'
    LOCATE k6 - 11, k4 + 11: PRINT "CODIGOS BLOQUE LINEALES"
    LOCATE k6 - 10, k4 + 14: PRINT "GRAFICOS DE Pu(E)"
    mensaje$ = "Seleccione con " + CHR$(24) + " " + CHR$(25) + " y con <ENTER>"
    mensaje1$ = "<ESC> regresa pantalla anterior"
    LOCATE k6 - 1, k4 + 2: PRINT TAB(k4 + 1 + (k5 - LEN(mensaje$)) / 2); mensaje$
```

```
    LOCATE k6, k4 + 2: PRINT TAB(k4 + 1 + (k5 - LEN(mensaje1$)) / 2); mensaje1$
'
    CALL MovBloque(AM1$(), ap$, 23, 57, 11, 14, y1, 23, 4, 10)
    IF ap$ = CHR$(27) THEN RETURN MenuPrincip
    IF y1 - 10 = 1 OR y1 - 10 = 2 THEN CALL Ingre3(camino$, nn(), kk(), tt(), kg$(), AnchHam$())
    IF y1 - 10 = 3 OR y1 - 10 = 4 THEN
        CALL Ingre1(camino$, n(), k(), t())
        CALL Ingre4(camino$, Ater$())
    END IF
    ON (y1 - 10) GOSUB grafico1, grafico2, grafico3, grafico4
'
grafico1:
    CLS
    k4 = 17: k5 = 44: k6 = 18
    GOSUB entrada
'
    LOCATE 7, 24: PRINT "Probabilidad de no detección de"
    LOCATE 8, 26: PRINT "errores Pu(E) vs. p del BSC"
    LOCATE 10, 29: PRINT "Gráfico de Pu(E) vs. p"
    LOCATE 11, 25: PRINT "Ingrese el código a utilizarce:"
    LOCATE 13, 25: PRINT "long. del código,  n : "
    LOCATE 14, 25: PRINT "long. del mensaje, k : "
    LOCATE 15, 25: PRINT "bits que corrige,  t : "
    LOCATE 17, 24: PRINT "Ingresar los valores correctos"
    LOCATE 18, 24: PRINT "de las variables para continuar"
'
    jselec = 1
    DO
        LOCATE 13, 48: PRINT " "; nn(jselec); "   "
        LOCATE 14, 48: PRINT " "; kk(jselec); "   "
        LOCATE 15, 48: PRINT " "; tt(jselec); "   "
'
        ter$ = INKEY$
        IF ter$ = CHR$(13) OR ter$ = CHR$(27) THEN EXIT DO
        IF LEN(ter$) = 2 THEN
            IF ASC(RIGHT$(ter$, 1)) = 80 THEN
                jselec = jselec - 1
                IF jselec < 1 THEN jselec = 40
            END IF
            IF ASC(RIGHT$(ter$, 1)) = 72 THEN
                jselec = jselec + 1
                IF jselec > 40 THEN jselec = 1
            END IF
        END IF
    LOOP
    COLOR 7, 0
    IF ter$ = CHR$(27) THEN
        ERASE nn, kk, tt, kg$, AnchHam$
        RETURN bloque3
    END IF
'
Otrorang1:
```

```
    codigo$ = 'código:' + STR$(nn(jselec)) + ' ' + STR$(kk(jselec)) + ' ' + STR$(tt(jselec))
    CALL Cuadro2(15, 16, 47, 47, codigo$, pinf, psup, tecla$)
    IF tecla$ = CHR$(27) THEN GOTO grafico1
'
    IF tt(jselec) = 1 THEN
        CALL PuCodHam(nn(jselec), kk(jselec), pinf, psup, PMayor, PMenor, Pu())
        ELSE
        CALL PuCodBlq(AnchHam$(jselec), kg$(jselec), nn(jselec), kk(jselec), pinf, psup, PMayor, PMenor,
Pu())
    END IF
'
    CLS
    SCREEN video
    LOCATE 2, 1: PRINT CHR$(201); STRING$(78, 205); CHR$(187)
    FOR j = 3 TO 26
        LOCATE j, 1: PRINT CHR$(186); SPC(78); CHR$(186)
    NEXT j
    LOCATE 27, 1: PRINT CHR$(200); STRING$(78, 205); CHR$(188)
    men$ = 'GRAFICO DE Pu(E) vs p. ( CODIGO:' + STR$(nn(jselec)) + STR$(kk(jselec)) + STR$(tt(jselec)) +
' )'
    LOCATE 4, 2: PRINT TAB((78 - LEN(men$)) / 2); men$
'
    LINE (100 * pfact1, 110 * pfact2)-(100 * pfact1, 380 * pfact2)
    LINE (90 * pfact1, 370 * pfact2)-(590 * pfact1, 370 * pfact2)
    FOR i = 120 TO 345 STEP 25
        LINE (97 * pfact1, i * pfact2)-(103 * pfact1, i * pfact2)
    NEXT i
    FOR i = 124 TO 580 STEP 24
        LINE (i * pfact1, 367 * pfact2)-(i * pfact1, 373 * pfact2)
    NEXT i
    FOR i = 1 TO 5
        LINE (94 * pfact1, (70 + 50 * i) * pfact2)-(106 * pfact1, (70 + 50 * i) * pfact2)
        LINE ((100 + 96 * i) * pfact1, 364 * pfact2)-((100 + 96 * i) * pfact1, 376 * pfact2)
    NEXT i
'
    x1 = 100: y1 = 370 - ((251 / (PMayor - PMenor)) * (Pu(1) - PMenor))
    IndProb = 2: ProbEsc = pinf + (psup - pinf) / 50
    DO WHILE ProbEsc <= psup
        x2 = 100 + (480 * (ProbEsc - pinf) / (psup - pinf)): y2 = 370 - ((251 / (PMayor - PMenor)) *
(Pu(IndProb) - PMenor))
        LINE (x1 * pfact1, y1 * pfact2)-(x2 * pfact1, y2 * pfact2)
        x1 = x2: y1 = y2
        IndProb = IndProb + 1: ProbEsc = ProbEsc + (psup - pinf) / 50
    LOOP
'
    Paso1 = PMenor
    FOR i = 23 TO 8 STEP -3
        LOCATE i, 3: PRINT USING '#.###^^^^'; Paso1
        Paso1 = Paso1 + ((PMayor - PMenor) / 5)
    NEXT i
    LOCATE 6, 8: PRINT 'Pu(E)'
'
```

```
    IF psup - pinf = 1 THEN
        paso = pinf
        FOR i = 0 TO 10 STEP 2
            LOCATE 25, i * 6 + 12: PRINT USING "#.#"; paso
            paso = paso + ((psup - pinf) / 5)
        NEXT i
        ELSE
        paso = pinf
        FOR i = 0 TO 10 STEP 2
            LOCATE 25, i * 6 + 10: PRINT USING ".###^^^^"; paso
            paso = paso + ((psup - pinf) / 5)
        NEXT i
    END IF
    LOCATE 23, 77: PRINT "p"
'
    GOSUB parada
    IF control$ = CHR$(27) THEN GOTO grafico1
    GOTO Otrorang1
'
grafico2:
    CLS
    k4 = 15: k5 = 46: k6 = 17
    GOSUB entrada
'
    LOCATE 7, 17: PRINT "Probabilidad de no detección de errores Pu(E)"
    LOCATE 9, 29: PRINT "Pu(E) = f(n-k,p)"
    LOCATE 10, 22: PRINT "Para códigos que corrigen t errores"
    LOCATE 11, 20: PRINT "n-k : bits de paridad"
    LOCATE 12, 20: PRINT "p   : probabilidad de transición del BSC"
    LOCATE 16, 23: PRINT "Se agrupa las curvas Pu(E) vs. p que"
    LOCATE 17, 22: PRINT "corrigen t errores en función de n-k"
    LOCATE 14, 18: COLOR 15, 0: PRINT " Número de errores que corrigen, t:      "
    LOCATE 14, 54: INPUT "", t: COLOR 0, 15
    LOCATE 16, 17: PRINT "      Presione <ENTER> para continuar      "
    LOCATE 17, 17: PRINT "      <ESC> regresa pantalla anterior      "
    COLOR 7, 0
'
    DO
        ap$ = INKEY$
    LOOP UNTIL ap$ = CHR$(13) OR ap$ = CHR$(27)
    IF ap$ = CHR$(27) THEN
        ERASE nn, kk, tt, kg$, AnchHam$
        RETURN bloque3
    END IF
'
Otrorang2:
    codigo$ = "Códigos que corrigen" + STR$(t) + " errores"
    CALL Cuadro2(15, 16, 47, 47, codigo$, pinf, psup, tecla$)
    IF tecla$ = CHR$(27) THEN GOTO grafico2
'
    prob = psup: paso = 25
    PMayor = 0: PMenor = 1: cont = 0
```

```
'
    CLS
    SCREEN video
    LOCATE 2, 1: PRINT CHR$(201); STRING$(78, 205); CHR$(187)
    FOR k = 3 TO 26
        LOCATE k, 1: PRINT CHR$(186); SPC(78); CHR$(186)
    NEXT k
    LOCATE 27, 1: PRINT CHR$(200); STRING$(78, 205); CHR$(188)
    men$ = "GRAFICO DE Pu(E) vs. (n-k) vs. p ( CODIGOS CON t =" + STR$(t) + " )"
    LOCATE 4, 2: PRINT TAB((78 - LEN(men$)) / 2); men$
'
    LINE (275 * pfact1, 75 * pfact2)-(275 * pfact1, 250 * pfact2), 7
    LINE (540 * pfact1, 250 * pfact2)-(275 * pfact1, 250 * pfact2), 7
    LINE (86 * pfact1, 385 * pfact2)-(275 * pfact1, 250 * pfact2), 7
'
    LINE (183.33 * pfact1, 395 * pfact2)-(266.67 * pfact1, 395 * pfact2)
    LINE (545 * pfact1, 196.67 * pfact2)-(545 * pfact1, 143.33 * pfact2)
    LINE (429.67 * pfact1, 340 * pfact2)-(473.33 * pfact1, 305 * pfact2)
    LINE (256.67 * pfact1, 390 * pfact2)-(266.67 * pfact1, 395 * pfact2)
    LINE (256.67 * pfact1, 400 * pfact2)-(266.67 * pfact1, 395 * pfact2)
    LINE (540 * pfact1, 153.33 * pfact2)-(545 * pfact1, 143.33 * pfact2)
    LINE (550 * pfact1, 153.33 * pfact2)-(545 * pfact1, 143.33 * pfact2)
    LINE (459.33 * pfact1, 310 * pfact2)-(473.33 * pfact1, 305 * pfact2)
    LINE (467.33 * pfact1, 317 * pfact2)-(473.33 * pfact1, 305 * pfact2)
'
    LOCATE 12, 70: PRINT "Pu(E)": LOCATE 26, 26: PRINT "(n-k)": LOCATE 21, 58: PRINT "p"
'
    isb = 0: itm = 0
    DO
        j = 1: nk = 1
        DO
            IF tt(j) = t THEN
                Paridad(nk) = nn(j) - kk(j)
                Pu(Paridad(nk)) = 0: al = 1
                AnchHam$ = ""
                IF kg$(j) = "NDual" THEN
                    AnchHam$ = MID$(AnchHam$(j), 3, LEN(AnchHam$(j)) - 2): Ind = 1
                    ELSE
                    AnchHam$ = AnchHam$(j): Ind = 0
                END IF
'
                DO WHILE Ind <= nn(j)
                    Ham$ = ""
                    DO
                        Ham$ = Ham$ + MID$(AnchHam$, al, 1)
                        al = al + 1
                    LOOP UNTIL MID$(AnchHam$, al, 1) = " " OR MID$(AnchHam$, al, 1) = ""
                    Ancho = VAL(Ham$)
'
                    IF Ancho <> 0 THEN
                        IF kg$(j) = "NDual" THEN
                            Pu(Paridad(nk)) = Pu(Paridad(nk)) + ((prob ^ Ind) * ((1 - prob) ^ (nn(j)
```

```
- Ind)) * Ancho)
                                ELSE
                                Pu(Paridad(nk)) = Pu(Paridad(nk)) + Ancho * ((1 - 2 * prob) ^ Ind)
                            END IF
                        END IF
                        Ind = Ind + 1
                    LOOP
                    IF kg$(j) = "Dual" THEN Pu(Paridad(nk)) = (2 ^ (kk(j) - nn(j))) * Pu(Paridad(nk)) - ((1
- prob) ^ nn(j))
'
                    IF cont = 0 THEN
                        IF Pu(Paridad(nk)) > PMayor THEN PMayor = Pu(Paridad(nk))
                        IF Pu(Paridad(nk)) < PMenor THEN PMenor = Pu(Paridad(nk))
                    END IF
                    nk = nk + 1
                END IF
                j = j + 1
            LOOP UNTIL j = 40
'
            x1 = 275 - isb
            IF PMayor = PMenor THEN
                y1 = itm + 90
                ELSE
                y1 = 250 + itm - (160 * (Pu(Paridad(1)) - PMenor) / (PMayor - PMenor))
            END IF
'
            im = 2
            DO WHILE im <= nk - 1
                x2 = 275 - isb + (250 * (im - 1) / (nk - 2))
                IF PMayor = PMenor THEN
                    y2 = itm + 90
                    ELSE
                    y2 = 250 + itm - (160 * (Pu(Paridad(im)) - PMenor) / (PMayor - PMenor))
                END IF
                IF prob <> psup THEN
                    LINE (pfact1 * !Xaux(im - 1), pfact2 * !Yaux(im - 1))-(x1 * pfact1, y1 * pfact2), 15
                END IF
                !Xaux(im - 1) = x1: !Yaux(im - 1) = y1
                LINE (x1 * pfact1, y1 * pfact2)-(x2 * pfact1, y2 * pfact2), 15
                x1 = x2: y1 = y2
                IF im = nk - 1 AND prob <> psup THEN
                    LINE (pfact1 * !Xaux(im), pfact2 * !Yaux(im))-(x1 * pfact1, y1 * pfact2), 15
                END IF
                im = im + 1
            LOOP
'
            !Xaux(im - 1) = x1: !Yaux(im - 1) = y1
            IF PMayor = PMenor THEN
                cont = 0
                ELSE
                cont = 1
            END IF
```

```
        prob = prob - ((psup - pinf) / paso)
        isb = isb + 175 / paso: itm = itm + 125 / paso
        icont = icont + 1
    LOOP UNTIL prob < pinf
'
    LOCATE 15, 67: PRINT USING "#.##^^^^"; PMenor: LOCATE 7, 67: PRINT USING "#.##^^^^"; PMayor
    LOCATE 25, 11: PRINT USING "###"; Paridad(1): LOCATE 25, 42: PRINT USING "###"; Paridad(nk - 1)
    IF psup - pinf = 1 THEN
        LOCATE 24, 48: PRINT USING "#.#"; pinf: LOCATE 17, 67: PRINT USING "#.#"; psup
        ELSE
        LOCATE 24, 48: PRINT USING "#.##^^^^"; pinf: LOCATE 17, 67: PRINT USING "#.##^^^^"; psup
    END IF
'
    GOSUB parada
    IF control$ = CHR$(27) THEN GOTO grafico2
    GOTO Otrorang2
'
grafico3:
    CLS
    k4 = 17: k5 = 44: k6 = 18
    GOSUB entrada
'
    LOCATE 7, 20: PRINT "Probabilidad de error en los mensajes en"
    LOCATE 8, 23: PRINT "códigos (n,k,t), P(E) vs. p del BSC"
    LOCATE 10, 21: PRINT "p  : probabilidad de error en los bits"
    LOCATE 11, 25: PRINT "Ingrese el código a utilizarce:"
    LOCATE 13, 25: PRINT "long. del código,  n : "
    LOCATE 14, 25: PRINT "long. del mensaje, k : "
    LOCATE 15, 25: PRINT "bits que corrige,  t : "
    LOCATE 17, 24: PRINT "Ingresar los valores correctos"
    LOCATE 18, 24: PRINT "de las variables para continuar"
'
    jselec = 1
    DO
        LOCATE 13, 48: PRINT " "; n(jselec); "   "
        LOCATE 14, 48: PRINT " "; k(jselec); "   "
        LOCATE 15, 48: PRINT " "; t(jselec); "   "
'
        ter$ = INKEY$
        IF ter$ = CHR$(13) OR ter$ = CHR$(27) THEN EXIT DO
        IF LEN(ter$) = 2 THEN
            IF ASC(RIGHT$(ter$, 1)) = 80 THEN
                jselec = jselec - 1
                IF jselec < 1 THEN jselec = 232
            END IF
            IF ASC(RIGHT$(ter$, 1)) = 72 THEN
                jselec = jselec + 1
                IF jselec > 232 THEN jselec = 1
            END IF
        END IF
    LOOP
    COLOR 7, 0
```

```
    IF ter$ = CHR$(27) THEN
        ERASE n, k, t, Ater$
        RETURN bloque3
    END IF
'
Otrorang3:
    codigo$ = "código:" + STR$(n(jselec)) + " " + STR$(k(jselec)) + " " + STR$(t(jselec))
    CALL Cuadro2(15, 16, 47, 47, codigo$, pinf, psup, tecla$)
    IF tecla$ = CHR$(27) THEN GOTO grafico3
'
    Alsab$ = ""
    IF n(jselec) = 7 THEN
        Alsab$ = Ater$(1)
        ELSEIF n(jselec) = 15 THEN
        Alsab$ = Ater$(2)
        ELSEIF n(jselec) = 31 THEN
        Alsab$ = Ater$(3)
        ELSEIF n(jselec) = 63 THEN
        Alsab$ = Ater$(4)
        ELSEIF n(jselec) = 127 THEN
        Alsab$ = Ater$(5)
        ELSEIF n(jselec) = 255 THEN
        Alsab$ = Ater$(6)
        ELSEIF n(jselec) = 511 THEN
        Alsab$ = Ater$(7)
        ELSEIF n(jselec) = 1023 THEN
        Alsab$ = Ater$(8)
    END IF
'
    IndProb = 1: prob = pinf: PMayor = 0: PMenor = 1
    DO WHILE prob <= psup
        PE(IndProb) = 0
        FOR i = 2 TO INT(n(jselec) / 2)
            al = 1: Acombina$ = ""
            DO
                Acombina$ = Acombina$ + MID$(Alsab$, al, 1)
                al = al + 1
            LOOP UNTIL MID$(Alsab$, al, 1) = " " OR MID$(Alsab$, al, 1) = ""
            Acombi(i) = VAL(Acombina$)
        NEXT i
        ii = 1
        FOR icont = i TO n(jselec)
            Acombi(icont) = Acombi(icont - ii)
            ii = ii + 1
        NEXT icont
        Acombi(n(jselec) - 1) = n: Acombi(n(jselec)) = 1
'
        Ind = t(jselec) + 1
        DO WHILE Ind <= n(jselec)
            PE(IndProb) = PE(IndProb) + (Acombi(Ind) * (prob ^ Ind) * ((1 - prob) ^ (n(jselec) - Ind)))
            Ind = Ind + 1
        LOOP
```

```
'
        IF PE(IndProb) > PMayor THEN PMayor = PE(IndProb)
        IF PE(IndProb) < PMenor THEN PMenor = PE(IndProb)
'
        prob = prob + ((psup - pinf) / 50)
        IndProb = IndProb + 1
    LOOP
'
    CLS
    SCREEN video
    LOCATE 2, 1: PRINT CHR$(201); STRING$(78, 205); CHR$(187)
    FOR j = 3 TO 26
        LOCATE j, 1: PRINT CHR$(186); SPC(78); CHR$(186)
    NEXT j
    LOCATE 27, 1: PRINT CHR$(200); STRING$(78, 205); CHR$(188)
    men$ = "GRAFICO DE P(E) vs p. ( CODIGO:" + STR$(n(jselec)) + STR$(k(jselec)) + STR$(t(jselec)) + " )"
    LOCATE 4, 2: PRINT TAB((78 - LEN(men$)) / 2); men$
'
    LINE (100 * pfact1, 110 * pfact2)-(100 * pfact1, 380 * pfact2)
    LINE (90 * pfact1, 370 * pfact2)-(590 * pfact1, 370 * pfact2)
    FOR i = 120 TO 345 STEP 25
        LINE (97 * pfact1, i * pfact2)-(103 * pfact1, i * pfact2)
    NEXT i
    FOR i = 124 TO 580 STEP 24
        LINE (i * pfact1, 367 * pfact2)-(i * pfact1, 373 * pfact2)
    NEXT i
    FOR i = 1 TO 5
        LINE (94 * pfact1, (70 + 50 * i) * pfact2)-(106 * pfact1, (70 + 50 * i) * pfact2)
        LINE ((100 + 96 * i) * pfact1, 364 * pfact2)-((100 + 96 * i) * pfact1, 376 * pfact2)
    NEXT i
'
    x1 = 100: y1 = 370 - ((251 / (PMayor - PMenor)) * (PE(1) - PMenor))
    IndProb = 2: ProbEsc = pinf + (psup - pinf) / 50
    DO WHILE ProbEsc <= psup
        x2 = 100 + (480 * (ProbEsc - pinf) / (psup - pinf))
        y2 = 370 - ((251 / (PMayor - PMenor)) * (PE(IndProb) - PMenor))
        LINE (x1 * pfact1, y1 * pfact2)-(x2 * pfact1, y2 * pfact2)
        x1 = x2: y1 = y2
        IndProb = IndProb + 1: ProbEsc = ProbEsc + (psup - pinf) / 50
    LOOP
'
    Paso1 = PMenor
    FOR i = 23 TO 8 STEP -3
        LOCATE i, 3: PRINT USING "#.###^^^^"; Paso1
        Paso1 = Paso1 + ((PMayor - PMenor) / 5)
    NEXT i
    LOCATE 6, 9: PRINT "P(E)"
'
    IF psup - pinf = 1 THEN
        paso = pinf
        FOR i = 0 TO 10 STEP 2
            LOCATE 25, i * 6 + 12: PRINT USING "#.#"; paso
```

```
            paso = paso + ((psup - pinf) / 5)
        NEXT i
        ELSE
        paso = pinf
        FOR i = 0 TO 10 STEP 2
            LOCATE 25, i * 6 + 10: PRINT USING ".###^^^^"; paso
            paso = paso + ((psup - pinf) / 5)
        NEXT i
    END IF
    LOCATE 23, 77: PRINT "p"
'
    GOSUB parada
    IF control$ = CHR$(27) THEN GOTO grafico3
    GOTO Otrorang3
'
grafico4:
    CLS
    k4 = 15: k5 = 46: k6 = 17
    GOSUB entrada
'
    LOCATE 7, 20: PRINT "Probabilidad de error en los mensajes"
    LOCATE 9, 29: PRINT "P(E) = f(n-k,p)"
    LOCATE 10, 22: PRINT "Para códigos que corrigen t errores"
    LOCATE 11, 20: PRINT "n-k : bits de paridad"
    LOCATE 12, 20: PRINT "p   : probabilidad de error en los bits"
    LOCATE 16, 23: PRINT "Se agrupa las curvas P(E) vs. p que"
    LOCATE 17, 22: PRINT "corrigen t errores en función de n-k"
    LOCATE 14, 18: COLOR 15, 0: PRINT " Número de errores que corrigen, t:      "
    LOCATE 14, 54: INPUT "", t: COLOR 0, 15
    LOCATE 16, 17: PRINT "      Presione <ENTER> para continuar      "
    LOCATE 17, 17: PRINT "      <ESC> regresa pantalla anterior      "
    COLOR 7, 0
'
    DO
        ap$ = INKEY$
    LOOP UNTIL ap$ = CHR$(13) OR ap$ = CHR$(27)
    IF ap$ = CHR$(27) THEN
        ERASE n, k, t, Ater$
        RETURN bloque3
    END IF
'
Otrorang4:
    codigo$ = "Códigos que corrigen" + STR$(t) + " errores"
    CALL Cuadro2(15, 16, 47, 47, codigo$, pinf, psup, tecla$)
    IF tecla$ = CHR$(27) THEN GOTO grafico4
'
    prob = psup: paso = 25
    PMayor = 0: PMenor = 1: cont = 0
'
    CLS
    SCREEN video
    LOCATE 2, 1: PRINT CHR$(201); STRING$(78, 205); CHR$(187)
```

```
FOR k = 3 TO 26
    LOCATE k, 1: PRINT CHR$(186); SPC(78); CHR$(186)
NEXT k
LOCATE 27, 1: PRINT CHR$(200); STRING$(78, 205); CHR$(188)
men$ = "GRAFICO DE P(E) vs. (n-k) vs. p ( CODIGOS CON t =" + STR$(t) + " )"
LOCATE 4, 2: PRINT TAB((78 - LEN(men$)) / 2); men$
'
LINE (275 * pfacL1, 75 * pfacL2)-(275 * pfact1, 250 * pfact2), 7
LINE (540 * pfacL1, 250 * pfact2)-(275 * pfact1, 250 * pfact2), 7
LINE (86 * pfact1, 385 * pfacL2)-(275 * pfact1, 250 * pfact2), 7
'
LINE (183.33 * pfact1, 395 * pfact2)-(266.67 * pfact1, 395 * pfact2)
LINE (545 * pfact1, 196.67 * pfact2)-(545 * pfact1, 143.33 * pfact2)
LINE (429.67 * pfact1, 340 * pfact2)-(473.33 * pfact1, 305 * pfact2)
LINE (256.67 * pfact1, 390 * pfact2)-(266.67 * pfact1, 395 * pfact2)
LINE (256.67 * pfact1, 400 * pfact2)-(266.67 * pfact1, 395 * pfact2)
LINE (540 * pfact1, 153.33 * pfact2)-(545 * pfact1, 143.33 * pfact2)
LINE (550 * pfact1, 153.33 * pfacL2)-(545 * pfacL1, 143.33 * pfact2)
LINE (459.33 * pfact1, 310 * pfact2)-(473.33 * pfact1, 305 * pfact2)
LINE (467.33 * pfact1, 317 * pfact2)-(473.33 * pfact1, 305 * pfact2)
'
LOCATE 12, 70: PRINT "P(E)": LOCATE 26, 26: PRINT "(n-k)": LOCATE 21, 58: PRINT "p"
'
isb = 0: itm = 0
DO
    j = 1: nk = 1
    DO
        IF L(j) = t THEN
            Paridad(nk) = n(j) - k(j): PE(Paridad(nk)) = 0
'
            Alsab$ = ""
            IF n(j) = 7 THEN
                Alsab$ = Aler$(1)
                ELSEIF n(j) = 15 THEN
                Alsab$ = Aler$(2)
                ELSEIF n(j) = 31 THEN
                Alsab$ = Aler$(3)
                ELSEIF n(j) = 63 THEN
                Alsab$ = Aler$(4)
                ELSEIF n(j) = 127 THEN
                Alsab$ = Aler$(5)
                ELSEIF n(j) = 255 THEN
                Alsab$ = Aler$(6)
                ELSEIF n(j) = 511 THEN
                Alsab$ = Aler$(7)
                ELSEIF n(j) = 1023 THEN
                Alsab$ = Aler$(8)
            END IF
'
            FOR i = 2 TO INT(n(j) / 2)
                al = 1: Acombina$ = ""
                DO
```

```
                    Acombina$ = Acombina$ + MID$(Alsab$, al, 1)
                    al = al + 1
                LOOP UNTIL MID$(Alsab$, al, 1) = " " OR MID$(Alsab$, al, 1) = ""
                Acombi(i) = VAL(Acombina$)
            NEXT i
            ii = 1
            FOR icont = i TO n(j)
                Acombi(icont) = Acombi(icont - ii)
                ii = ii + 1
            NEXT icont
            Acombi(n(j) - 1) = n(j): Acombi(n(j)) = 1
'
            Ind = t(j) + 1
            DO WHILE Ind <= n(j)
                PE(Paridad(nk)) = PE(Paridad(nk)) + (Acombi(Ind) * (prob ^ Ind) * ((1 - prob) ^
(n(j) - Ind)))
                Ind = Ind + 1
            LOOP
'
            IF cont = 0 THEN
                IF PE(Paridad(nk)) > PMayor THEN PMayor = PE(Paridad(nk))
                IF PE(Paridad(nk)) < PMenor THEN PMenor = PE(Paridad(nk))
            END IF
            nk = nk + 1
        END IF
        j = j + 1
    LOOP UNTIL j = 232
'
    x1 = 275 - isb
    IF PMayor = PMenor THEN
        y1 = itm + 90
        ELSE
        y1 = 250 + itm - (160 * (PE(Paridad(1)) - PMenor) / (PMayor - PMenor))
    END IF
'
    im = 2
    DO WHILE im <= nk - 1
        x2 = 275 - isb + (250 * (im - 1) / (nk - 2))
        IF PMayor = PMenor THEN
            y2 = itm + 90
            ELSE
            y2 = 250 + itm - (160 * (PE(Paridad(im)) - PMenor) / (PMayor - PMenor))
        END IF
        IF prob <> psup THEN LINE (pfact1 * !Xaux(im - 1), pfact2 * !Yaux(im - 1))-(x1 * pfact1, y1
* pfact2), 15
        !Xaux(im - 1) = x1: !Yaux(im - 1) = y1
        LINE (x1 * pfact1, y1 * pfact2)-(x2 * pfact1, y2 * pfact2), 15
        x1 = x2: y1 = y2
        IF im = nk - 1 AND prob <> psup THEN LINE (pfact1 * !Xaux(im), pfact2 * !Yaux(im))-(x1 *
pfact1, y1 * pfact2), 15
        im = im + 1
    LOOP
```

```
'
        IXaux(ia - 1) = x1: IYaux(ia - 1) = y1
        IF PMayor = PMenor THEN
            cont = 0
            ELSE
            cont = 1
        END IF
        prob = prob - ((psup - pinf) / paso)
        isb = isb + 175 / paso: ita = ita + 125 / paso
        icont = icont + 1
    LOOP UNTIL prob < pinf
'
    LOCATE 15, 67: PRINT USING "#.##^^^^"; PMenor: LOCATE 7, 67: PRINT USING "#.##^^^^"; PMayor
    LOCATE 25, 11: PRINT USING "###"; Paridad(1): LOCATE 25, 42: PRINT USING "###"; Paridad(nk - 1)
    IF psup - pinf = 1 THEN
        LOCATE 24, 48: PRINT USING "#.#"; pinf: LOCATE 17, 67: PRINT USING "#.#"; psup
        ELSE
        LOCATE 24, 48: PRINT USING "#.##^^^^"; pinf: LOCATE 17, 67: PRINT USING "#.##^^^^"; psup
    END IF
'
    GOSUB parada
    IF control$ = CHR$(27) THEN GOTO grafico4
    GOTO Otrorang4
'
entrada:
     CALL cuadro1(7, 0, 1, 1, 76, 22)
     COLOR 0, 15: LOCATE 1, 26: PRINT "  CODIGOS BLOQUE LINEALES  "
     CALL cuadro1(0, 15, 6, k4, k5, k6)
     LOCATE k6 - 9, k4 + 1: PRINT STRING$(k5, 196)
     LOCATE k6 - 2, k4 + 1: PRINT STRING$(k5, 196)
     RETURN
'
parada:
     BEEP: BEEP
     LOCATE 28, 2: PRINT " **  (ESC) Retorna al Menú anterior  ** "
     LOCATE 28, 42: PRINT " **  (P) o (p) Otro rango de p  ** "
     DO
          control$ = INKEY$
     LOOP UNTIL control$ = "P" OR control$ = "p" OR control$ = CHR$(27)
     SCREEN 0, 0, 0
     RETURN
'
Fin:
     CLS
     END

SUB cuadro1 (j1, j2, j3, j4, j5, j6)
    COLOR j1, j2: LOCATE j3, j4: PRINT CHR$(201); STRING$(j5, 205); CHR$(187)
    FOR j = j3 + 1 TO j6
        LOCATE j, j4: PRINT CHR$(186); SPC(j5); CHR$(186)
    NEXT j
    LOCATE j6 + 1, j4: PRINT CHR$(200); STRING$(j5, 205); CHR$(188)
```

```
END SUB

SUB Cuadro2 (i1, i2, j1, j2, codigo$, pinf, psup, tecla$)
    CALL cuadro1(15, 0, 7, 35, 34, 19)
    LOCATE 17, 36: PRINT STRING$(34, 196)
    LOCATE 9, 38: PRINT "Códigos bloque lineales (n,k,t)"
    LOCATE 10, 37: PRINT TAB(37 + (32 - LEN(codigo$)) / 2); codigo$
    LOCATE 12, 43: COLOR 15, 0: PRINT "Escala Horizontal :"
    LOCATE 13, 42: PRINT "(0 ≤ pinf < psup ≤ 1)"
    LOCATE 15, 44: COLOR 7, 0: PRINT "pinf = "
    LOCATE 16, 44: COLOR 7, 0: PRINT "psup = "
    LOCATE 18, 37: COLOR 15, 0: PRINT "Ingresar los valores límites de"
    LOCATE 19, 37: PRINT "las probabilidades, pinf y psup"
'
valores:
    LOCATE 15, 43: COLOR 16, 7: PRINT " pinf =           "
    CALL DigitEscala(15, 51, p1$)
    pinf = VAL(p1$)
    LOCATE 15, 43: COLOR 7, 0: PRINT " pinf = "; p1$
    LOCATE 16, 43: COLOR 16, 7: PRINT " psup =           "
    CALL DigitEscala(16, 51, p1$)
    psup = VAL(p1$)
    LOCATE 16, 43: COLOR 7, 0: PRINT " psup = "; p1$
    IF pinf >= psup THEN GOTO valores
'
    LOCATE 18, 37: COLOR 15, 0: PRINT "Presione <ENTER> para continuar"
    LOCATE 19, 37: PRINT "<ESC> regresa pantalla anterior"
    DO
        tecla$ = INKEY$
    LOOP UNTIL tecla$ = CHR$(27) OR tecla$ = CHR$(13)
    IF tecla$ = CHR$(27) THEN GOTO salida1
    CALL cuadro1(0, 15, 14, 20, 29, 17)
    LOCATE 16, 23: PRINT "gráfico en proceso.....!"
salida1:
END SUB

SUB DigitEscala (i, j, p1$)
otro1:
    LOCATE i, j: PRINT "          "
    p11$ = "": punto = 1
    DO
otro:     p$ = INKEY$
          IF p11$ = "sobreflujo" THEN p11$ = "": punto = 1
          IF (p$ < "0") OR (p$ > "9") THEN
              IF p$ <> "." THEN
                  IF p$ = CHR$(13) AND p11$ = "" THEN GOTO otro1
                  IF p$ = CHR$(13) THEN EXIT DO
                  IF p$ = CHR$(27) THEN GOTO otro1
                  GOTO otro
                  ELSE
                  IF punto = -1 THEN GOTO otro
                  punto = -1
```

```
            END IF
        END IF
        p11$ = p11$ + p$
        IF LEN(p11$) > 10 THEN p11$ = 'sobreflujo'
        p1$ = MID$('          ', 1, 11 - LEN(p11$)) + p11$
        LOCATE i, j: PRINT p1$
    LOOP
    IF VAL(p1$) > 1 THEN GOTO otro1
    p1$ = p1$ + ' '
END SUB

SUB Ingre1 (camino$, n(), k(), t())
    i = 1
    OPEN camino$ + '\TIPOS.TBL' FOR INPUT AS #1
        DO WHILE NOT EOF(1)
            INPUT #1, n(i), k(i), t(i)
            i = i + 1
        LOOP
    CLOSE #1
END SUB

SUB Ingre2 (camino$, n(), k(), t(), g1$())
    i = 1
    OPEN camino$ + '\TIPO_COD.TBL' FOR INPUT AS #1
        DO WHILE NOT EOF(1)
            INPUT #1, n(i), k(i), t(i)
            INPUT #1, g1$(i)
            i = i + 1
        LOOP
    CLOSE #1
END SUB

SUB Ingre3 (camino$, nn(), kk(), tt(), kg$(), AnchHam$())
    i = 1
    OPEN camino$ + '\TIP_COD1.TBL' FOR INPUT AS #1
        DO WHILE NOT EOF(1)
            INPUT #1, kg$(i)
            INPUT #1, nn(i), kk(i), tt(i)
            INPUT #1, AnchHam$(i)
            i = i + 1
        LOOP
    CLOSE #1
END SUB

SUB Ingre4 (camino$, Ater$())
    i = 1
    OPEN camino$ + '\DATCOMBI.TBL' FOR INPUT AS #1
        DO WHILE NOT EOF(1)
            INPUT #1, Ater$(i)
            i = i + 1
        LOOP
    CLOSE #1
```

```
END SUB

SUB MovBloque (AM$(), ap$, xi1, xf1, yi1, yf1, y1, y2, y3, y4)
    COLOR 0, 15
    FOR ren = 1 TO y3
        LOCATE y4 + ren, y2: PRINT AM$(ren)
    NEXT ren
    LOCATE y1, y2: COLOR 15, 0: PRINT AM$(y1 - y4)
'
    DO
        AM$ = AM$(y1 - y4)
        CALL MovCursor(AM$, ap$, xi1, xf1, yi1, yf1, y1)
    LOOP UNTIL ap$ = CHR$(13) OR ap$ = CHR$(27)
    COLOR 7, 0
END SUB

SUB MovCursor (AM$, ap$, xi, xf, yi, yf, y)
Curs1:
    ap$ = INKEY$
    IF LEN(ap$) = 2 THEN GOTO Cursor1
    IF ap$ = CHR$(13) OR ap$ = CHR$(27) THEN GOTO Cursor3
    GOTO Curs1
Cursor1:
    ay = ASC(RIGHT$(ap$, 1))
    IF ay <> 80 AND ay <> 72 THEN GOTO Cursor3
    COLOR 0, 15: LOCATE y, xi: PRINT AM$: AM$ = ""
    IF ay = 80 THEN y = y + 1 ELSE y = y - 1
    IF y > yf THEN y = yi ELSE IF y < yi THEN y = yf
'
    FOR x = xi TO xf
        AM$ = AM$ + CHR$(SCREEN(y, x))
    NEXT x
    COLOR 15, 0: LOCATE y, xi, 0: PRINT AM$
    COLOR 0, 15: IF xi - 1 > 0 THEN LOCATE y, xi - 1, 0
Cursor3:
END SUB

SUB PalabCodigo (nn, kk, G(), n1(), b())
    Grado = nn - kk
    FOR j = 0 TO nn - 1
        b(j) = 0
        IF j >= nn - kk THEN b(j) = n1(j)
    NEXT j
    Ult1 = nn - 1
    DO
        DO WHILE b(Ult1) <> 1 AND Ult1 >= Grado
            Ult1 = Ult1 - 1
        LOOP
        IF Ult1 < Grado THEN EXIT DO
'
        IA = Ult1 - Grado: j = 0
        DO WHILE j <= Grado
```

```
            IF G(j) = 1 THEN b(IA + j) = b(IA + j) XOR 1
                j = j + 1
        LOOP
    LOOP
    FOR Ind = nn - kk TO nn - 1
        b(Ind) = n1(Ind)
    NEXT Ind
END SUB

SUB PolinGener (polinhex$, polinbin$)
    polinbin$ = ""
    FOR i = 1 TO LEN(polinhex$)
        IF MID$(polinhex$, i, 1) = "0" THEN
            binario$ = "0000"
            ELSEIF MID$(polinhex$, i, 1) = "1" THEN
            binario$ = "0001"
            ELSEIF MID$(polinhex$, i, 1) = "2" THEN
            binario$ = "0010"
            ELSEIF MID$(polinhex$, i, 1) = "3" THEN
            binario$ = "0011"
            ELSEIF MID$(polinhex$, i, 1) = "4" THEN
            binario$ = "0100"
            ELSEIF MID$(polinhex$, i, 1) = "5" THEN
            binario$ = "0101"
            ELSEIF MID$(polinhex$, i, 1) = "6" THEN
            binario$ = "0110"
            ELSEIF MID$(polinhex$, i, 1) = "7" THEN
            binario$ = "0111"
            ELSEIF MID$(polinhex$, i, 1) = "8" THEN
            binario$ = "1000"
            ELSEIF MID$(polinhex$, i, 1) = "9" THEN
            binario$ = "1001"
            ELSEIF MID$(polinhex$, i, 1) = "A" THEN
            binario$ = "1010"
            ELSEIF MID$(polinhex$, i, 1) = "B" THEN
            binario$ = "1011"
            ELSEIF MID$(polinhex$, i, 1) = "C" THEN
            binario$ = "1100"
            ELSEIF MID$(polinhex$, i, 1) = "D" THEN
            binario$ = "1101"
            ELSEIF MID$(polinhex$, i, 1) = "E" THEN
            binario$ = "1110"
            ELSEIF MID$(polinhex$, i, 1) = "F" THEN
            binario$ = "1111"
            ELSE PRINT "Error, numero es mayor que F"
        END IF
        polinbin$ = polinbin$ + binario$
    NEXT i
END SUB

SUB PuCodBlq (DistAnch$, dualid$, nn, kk, pinf, psup, PMayor, PMenor, Pu())
    IndProb = 1: prob = pinf: PMayor = 0: PMenor = 1
```

```
    DO WHILE prob <= psup
        Pu(IndProb) = 0: al = 1
        IF dualid$ = "NDual" THEN
            AnchHam$ = MID$(DistAnch$, 3, LEN(DistAnch$) - 2): Ind = 1
            ELSE
            AnchHam$ = DistAnch$: Ind = 0
        END IF
'
        DO WHILE Ind <= nn
            Ham$ = ""
            DO
                Ham$ = Ham$ + MID$(AnchHam$, al, 1)
                al = al + 1
            LOOP UNTIL MID$(AnchHam$, al, 1) = " " OR MID$(AnchHam$, al, 1) = ""
            Ancho = VAL(Ham$)
'
            IF Ancho <> 0 THEN
                IF dualid$ = "NDual" THEN
                    Pu(IndProb) = Pu(IndProb) + ((prob ^ Ind) * ((1 - prob) ^ (nn - Ind)) * Ancho)
                    ELSE
                    Pu(IndProb) = Pu(IndProb) + Ancho * ((1 - 2 * prob) ^ Ind)
                END IF
            END IF
            Ind = Ind + 1
        LOOP
        IF dualid$ = "Dual" THEN Pu(IndProb) = (2 ^ (kk - nn)) * Pu(IndProb) - ((1 - prob) ^ nn)
'
        IF Pu(IndProb) > PMayor THEN PMayor = Pu(IndProb)
        IF Pu(IndProb) < PMenor THEN PMenor = Pu(IndProb)
'
        prob = prob + (psup - pinf) / 50
        IndProb = IndProb + 1
    LOOP
END SUB

SUB PuCodHam (nn, kk, pinf, psup, PMayor, PMenor, Pu())
    prob = pinf: PMayor = 0: PMenor = 1: IndProb = 1
    DO WHILE prob <= psup
        Pu(IndProb) = 0
        m = nn - kk
        Pu(IndProb) = (2 ^ (-m)) * (1 + ((2 ^ m) - 1) * (1 - 2 * prob) ^ (2 ^ (m - 1))) - (1 - prob) ^ ((2
^ m) - 1)
'
        IF Pu(IndProb) > PMayor THEN PMayor = Pu(IndProb)
        IF Pu(IndProb) < PMenor THEN PMenor = Pu(IndProb)
'
        prob = prob + (psup - pinf) / 50
        IndProb = IndProb + 1
    LOOP
END SUB
```

II.1 ARCHIVO TIPO_COD.TBL

Archivo que contiene los valores (n,k,t) y los coeficientes del polinomio generador expresados en forma hexadecimal para cada código bloque lineal de longitud hasta de $n = 2^{10} - 1$.

```
 7          4          1
B
 15         11         1
13
 15         7          2
1D1
 15         5          3
537
 31         26         1
25
 31         21         2
769
 31         16         3
8FAF
 31         11         5
1626D5
 31         6          7
32DEA27
 63         57         1
43
 63         51         2
1539
 63         45         3
782CF
 63         39         4
1DB2777
 63         36         5
86EB113
 63         30         6
37CD0EB67
 63         24         7
F69AC20921
 63         18         10
2F30B529D3D5
 63         16         11
CD930BDD3B2B
 63         10         13
2759262D5D506D
```

63 7 15
153225B1D0D73DF
127 120 1
89
127 113 2
4377
127 106 3
26D9E3
127 99 4
1C9C26B9
127 92 5
CA76024D7
127 85 6
58E24F9A4BB
127 78 7
2C98011D8B04D
127 71 9
195A08E5AACAFEB
127 64 10
A1AB815BC7EC8025
127 57 11
6EAD55545E28AE9FD1
127 50 13
2C9352AA6CC054468311
127 43 14
1FD1FD22F5297A8A42F8E3
127 36 15
CCC3CDB54B7A24FA5F3A3DD
127 29 21
40C99317BEE307B7AC184FC6D
127 22 23
14DFF87104E954A3B25CB67F4E23
127 15 27
121788A4B84B67E2A60BF923F08EB
127 8 31
E275A0ABD218D4CF928B9BBF6CB08F
255 247 1
11D
255 239 2
16F63
255 231 3
1BBA185
255 223 4
1EE5B42FD
255 215 5
1337DD3AD11
255 207 6
1C7EB85DF3C97
255 199 7
1F36195C443A4E1
255 191 8
16CE707E26B6F9977

```
 255            187            9
157B5976000B493CE9
 255            179            10
12CA7239EE08D439812D
 255            171            11
1B0E46229C4EE1F8C7319F
 255            163            12
1E810DA40F70569BE7529981
 255            155            13
1FBE960583ED41BD2A34D37F9B
 255            147            14
1D1160F75F3AD55887562C8413C9
 255            139            15
13180F68607DB8DE3A5D853CAEEF25
 255            131            18
11BCB6CCE6906958AA17F2231050EB39
 255            123            19
143182A510D807CF4435A9C614B2EA8CB7
 255            115            21
1855B6B7A2029D679E826017CEAB732E75DF
 255            107            22
1242FE9A4365732A1EC04EB9E207EBE7A0D921
 255            99             23
11AEDBAAE767C497861C81BE36955091F4698719
 255            91             25
1BD0B50C35E487AE9E67A9DAA48F6D1F2E8751C971
 255            87             26
120BDF38678D3D1D4CE718E7A1321BA5655DB27A2CF
 255            79             27
1B7003B9FD7D0020B87221DEC7CC8835ADC9FB585C4ED
 255            71             29
140A722A1A468D36D87A25364E685922A1E56FD1A478C1D
 255            63             30
11EC9E8B4E7646AB351EEFE380F6C49EB4B56F8BD770AC6C1
 255            55             31
1D9B1541D04805B06AF58C1A1635618D6F6822DE248B076778F
 255            47             42
156EC40F1969AE61B10BFE0C9B3E94DB86593094 0A360A5AAC67B
 255            45             43
6A085C2D4E1C4B241733FA27C1B9EE02938F93EC3682378545361
 255            37             45
52F3615A3703C30FF2752C7EB110EEF18F8D3BD39F48ECEFC0C4803
 255            29             47
615E6D7B76399AD5C680A78BFC9AE25135102 7C260C159AF8440EEA1F
 255            21             55
55C8D578C1B5AE00FECD787C510D2EE182EAC7962A89ACA38C9B52E77D5
 255            13             59
4D0F680A2ABA5922D7BE62A06C046C6FE4B3EB6C0CF9BF45DE162E4C28167
 255            9              63
6F582A8F9D4CD021911AB5DA5CC61C9EE8A120F2CA4AFB136CFC5B6EFE9C2F
 511            502            1
211
```

511 493 2
495C9
511 484 3
D612B79
511 475 4
1CC2B989A1
511 466 5
24AEA6C8E3F3
511 457 6
4C5BFB4377EBC7
511 448 7
870A4DF3A3E92805
511 439 8
1B8BA069B8B1FFE26E5
511 430 9
3D780318AF903DCC63397
511 421 10
59FC9CAADD69887B6938F4B
511 412 11
B7EB0748431794CE3DFDCCE41
511 403 12
1BDC7987AF64FCBADAFEA6A507A9
511 394 13
242A4390B0DB762E20F3C62BB5381B
511 385 14
7FB660AFFF94FA96CA8BEC0AC3664819
511 376 15
93BC9C6A8F56B4973961CCC8E4411BB683
511 367 17
12B6BD0545DB34C1E01D5296E58C8ED2701AD
511 358 18
37C971B2CB288676E1C7EAE9B8701FF8425F6AD
511 340 19
65A0A4E6F7D6A21E506DBAF7F818EB51B7BF850CB
511 340 20
B787B17194F2A690D909CA589DFBCE92F4241169FFF
511 331 21
1B0693A246145E1C5FAB293CFF2B4697B67BF98E865ED1
511 322 22
26BC66F68C4944BF3EDBDCA6E3DA9EA9E9465BF18FE3BA3B
511 313 23
43E2302B9A1242FE53A4E271C77704D74A67E235EEF2A69F1F
511 304 25
D23BC8A5D16DD74275579A1C38419D2ABB3A37C7E700EAB04EE9
511 295 26
1256627F580D9E04954F3F5FEF5AF509B15831FCE854A7DCF926E1D
511 286 27
2CF99C0D6BA5110D3FAD6127F7BF0EE60E7DC0EE654F5012EE5160339
511 277 28
49254F514ACE4070C4BF7B0612B930CFF4A9617CF61E1FBF16B0D59AC25
511 268 29
AB4E2A734BDCEB8D7D04CF0915A45D2D1FB59CC47438F323331560F750B81

511 259 30
128B30927220D69F1D251F2419336DCFD5C1514EBB18ECE5FC364E88DA334C4F
511 250 31
2075F61E33247165296BFBD7A93F355FB229AFE6AFBFE1DD4EB75BC6F39E7DCEA5
511 241 36
98B31AAC94358A16985964698837O546BFCD1DB0BAED0F219EC3C0658954CDDC3D1
511 238 37
2B626FD488EF96747B51047FB6CF092FF1F1FE5AA0C2FB4372AAD21D22D15B3885AFB
511 229 38
6433C2ABFF31D39B590784479D5D6549182A3F8ADDE5EEBF757DF52A06A9D1FBF800645
511 220 39
B61D1A810A98BF047CD8567F3BC740C66AADB4BE0D6582F50607F068F063E12CD238B6A2D
511 211 41
1AFACB7265F767A72B944CD8262F8E5DA4FC52ABDA8671E540398O3464E5C000FEDE5061FC03
511 202 42
2ED1E0C733B4F46A358054DA50ACFB841C7C3FCED6926A87E409C7DF0E976EC10F8AB2B5FDC93F
511 193 43
4D34265B3E644DE782A00DA161A39C9DF8DB50C099F774A12F6AF580967715DEEEF77451ECE22A2D
511 184 45
B42D3FEFB3F1C1FA4DCC3C1D7B45A90B58DBD5EAAED59720C91F0E52F7D4E8ABD21E96A404766E8BB9
511 175 46
1412C2E9BC6546FC7D45342996400DBE1FDAEEC7E9C989989C76A065132FA3101F25C5FEFC5DE6F725D43
511 166 47
4BBF497783E999F119244C3E99736521B693DFE34C0FD2B92FFBBA90ABBCA70949C1EEEE01A52E9191BC4B
511 157 51
790793CC0308B48E764C1C9A8BE3D7A400A2627B6B3D74704850DA6E51907C38C00094F7DE0FCFCDA6196EB43
511 148 53
C30B366283E55B61C186305812A68BF5DD07C5B8B5CBA5B86E38F67238E2BB7F94417643D133D09526BFC298531
511 139 54
1FEB78C62B1CD9FF2F9BD34C7C67D68F3F14B9954EE15B1F52E18C009BCE15BE29096BAF4F4AB6703C59C5C682D3F3
511 130 55
2564444333B8EDACA8F25BFB556D001E4EAEF2913D36296D9AC3768597798284 1D7F289A29862A159EE5D8A9FA2D0CD1
511 121 58
3615F977EB7F555C2118484EB51E0BB239E85C7D32AD32AB51AFBFF1354A2BF3DDAEF0FB8CF82BF193EEFA3BEB3E26EFEAF
511 112 59
AD13D203F95ACB31E81417BEEC6345D4B2028EE8154315F2CA21C2F4CAC5164D465926748F75F6288AC4FF5BE94A94C356F5
511 103 61
2B7A120E3ED7CF27D0420268A37E45ECD33A634862573DDB7063BCF5D11AAC740B6F14F0706F4E496C16B3B63EAEF63BE8111F
511 94 62
21CCA5B2A43E2F58DFCA65F50615E97B24E8280E065520752C53C6D3C1C7A5CBAFACD77B04DC1BAEC9E225FAD60B3DFA887B2C6DD
511 85 63
593136E2E8BB1E1132A00C576BC3176BDBF424B060769FEAA273F0D17CC0CDF912319135B1AB21C0071A100C25279294A4DACC1C8CD
511 76 85
AD9BBF95A7F2ABBA971BBA5F0C3B524C0F6F91DBE79D49B207848FFAD3B37791732BC9184E87A6E7CF2E2CBAAAAE28D8C59A7E1153E45
511 67 87
1BD14F93F5736AFF6A9F8AA73A02856842B2EA071AD9BDC0D11DE9842FBDC0459C1024BFB0E5DFBD44B01D21DF55A5C18035AA69E
7680621
511 58 91
22FA8FC00877144B18D63464937C9F24869AA4A04647F6A4C15A6E976EC0A5249C266B2C40F0D1317A5A9A4B9461DDBFB1C2AB698
18261B181
511 49 93
600FB2AA594B8E8E7E023298E7F39A231AAA69F6249C6DFD5C996546336BB55D37694F47BFC6F39C6E5D6F4DF777EC27E23FD0C7D

C205D50F5BB
511 40 95
8D746B0C0C2C186438E9F2CCB3B6D3156E644B25D384225BDEB08027BF490E54E82F927B85A04FDB27A73D4262C58F3D49005140F987F96F817D93
511 31 109
1020681744D0B066A19097EFB6FF4CA0C1850F7402E340EBF973FC9C99A4E7694FEE05F294F207260F4A807BF6958B9FFA86BAD4E22ADFA6FBA40CBE5
511 28 111
D1A288F07117F2C92B56772CDAF81B25CBB949041E74441BD723EA0A321041860F263DAC60AA228E4B34824B986FF91FC4A987207BD75C0AC8145BB59
511 19 119
169A12F978D89D11503EDAD09813826278BE039F7AD70D8632FF927610E24FA785A9FEB07D94ADDA483834B13071544089522430DE38F767A1E665B44DCB
511 10 127
3C23D3242F19EDD450DA31F88B0AD7EAA0A5F224A7D1070CB28E5D0169D673F99A9B025B640D2BD762684F2AC6F3BDFEEE6E2A4E3B57260C3A46B7D98BB21F
1023 1013 1
409
1023 1003 2
101877
1023 993 3
50A91113
1023 983 4
182EBE91E9B
1023 973 5
6F21CE1015FF9
1023 963 6
36C85772845CAAD
1023 953 7
168BE3CF3DB3D2C70CB
1023 943 8
1F0F22579AB8400128CE5
1023 933 9
5A756D8A96A24B479C95A19
1023 923 10
104D3F9B412624870B9B662B93
1023 913 11
7F6D77A4A8C6FB2E25D84213860D
1023 903 12
1F939D5CD2128FFDE767C26B988EEAF
1023 893 13
620F2F23E56CD665C03D9BCE350D0F511
1023 883 14
13F79793D7AA292C354FA652E006661177CB
1023 873 15
5F7BB5C05D5AC00289B97E1040FF529180151B
1023 863 16
1C81746E78492774F633AE021BAD004C082FCC64D
1023 858 17
2D2B6CD8CD692ECD13A2F46786713IE5DEEEBD4061
1023 848 18
EC5EA1AA4AF7CF31EFC0F10BC5CB2410B5EA57928OB3

1023 838 19
2A232FBEAD1FB7FEE9B7AB443CCF40C99038E4713C4A275
1023 828 20
A2D2504B267A6C03E42F419120B8BFD6581D0265E98FE260B
1023 818 21
246357C48E640AF510F7F4D0C8B9D7D2BC65B031E8E6A2E6DF07
1023 808 22
B6AF44431FEF25D35FB0108CB058628C216A63C4EB5D90E61ECC4F
1023 798 23
3AAEA2A02517E25294CF2D022706F9D330D67C8BD6CF854D60282235F
1023 788 24
94C3656EB015EFA98D0C2697228A3D1A9BC3ACD02931E4F1EBDCDBAB9B5
1023 778 25
3C0F31BF4831DE91D570E06959C943D1BBCF25AD647C72E04811856B79A271
1023 768 26
6202F84737823FDB93A6B177A5D2A4AD2395AF33E42205ED9D2A9B5AB3E1B0837
1023 758 27
3B26CFCA73CE28B1CA88688F1B26CC1516FF81518226EE9BCB2F3A87751303589BD
1023 748 28
A700EBA018193D3F10E3CD414B60E0DD55AD8A17C02308577AAC4A5F17ECC5FFFD38F
1023 738 29
3F1154D79F7C94B23686883CEBC671415A0892BDE1B00E06BE192C7BA1BF88DE5F1F2ABF
1023 728 30
B8A20A033D0B2CF2C7AA9A9B1B220A01F3502A4A94D11B4FB523A2650CBEF0439FE824BE57
1023 718 31
344403676B3C0517DE7AD2B7F754CB67B7E863FD12501CA6AE11A7C1CBC01E7E779355AFAF629
1023 708 34
A475FE99E3EFB04FFBB89BB0E12AE63359AF7F7CA74BD69B6FC9ED9AACA28961B045792DAD7BB65
1023 698 35
24E21F1EA7CBEAAF4599326DC9DBF9E8D76F74D903CB4D9CB27971FB679FDE886A4D33A3F88DBC62A5
1023 11 255
1C7626A215C2D5F402AA178B734A614E606AB33582C7B9375908A8CC4462B17C2479DAD32EE968B3A7EB6821CE44F0D8D3BC80C72
B5EF69051D1BC12AE84C60F8DA9A1A3EA4CF2917502E3235CBF36EFB258654F44B9EC0470FC49D766F96C20EA97AFE92187702714
AF327CFF24A287AB7A6CFBB7EFFB6CB2833837103C7F

II.2 ARCHIVO DATOS1.TBL

Contiene los valores (n,k,t) de los códigos bloque lineales, la distribución de ancho de Hamming de los códigos y además la indicación si ésta distribución fue calculada con el código dual o no.

Estos parámetros han sido agrupados y almacenados para cada código.

```
Dual
 7        4        1
1 0 0 0 7 0 0 0
Dual
 15       11       1
1 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0
NDual
 15       7        2
1 0 0 0 0 18 30 15 15 30 18 0 0 0 0 1
NDual
 15       5        3
1 0 0 0 0 0 0 15 15 0 0 0 0 0 0 1
Dual
 31       26       1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
 31       21       2
1 0 0 0 0 0 0 0 0 0 0 0 310 0 0 0 527 0 0 0 186 0 0 0 0 0 0 0 0 0 0 0
Dual
 31       16       3
1 0 0 0 0 0 0 0 465 0 0 0 8680 0 0 0 18259 0 0 0 5208 0 0 0 155 0 0 0 0 0 0 0
NDual
 31       11       5
1 0 0 0 0 0 0 0 0 0 0 186 310 0 0 527 527 0 0 310 186 0 0 0 0 0 0 0 0 0 0 1
NDual
 31       6        7
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 31 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
Dual
 63       57       1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
Dual
```

63 51 2
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 210 0 0 0 1512 0 0 0 1071 0 0 0 1176 0 0 0 126 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NDual
63 10 13
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 196 252 0 0 63 63 0 0 252 196 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
NDual
63 7 15
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 63 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1
Dual
127 120 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 127 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NDual
127 8 31
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 127 127 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
Dual
255 247 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NDual
255 9 63
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
Dual
511 502 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NDual
511 10 127
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0

II.3 ARCHIVO DATOS2.TBL

Igual que el archivo anterior, contiene los valores (n,k,t) de los códigos bloque lineales, la distribución de ancho de Hamming de los códigos y además la indicación que ésta distribución fue calculada con el código dual. Estos datos corresponden a códigos BCH cuya distribución de ancho de Hamming se pueden obtener con relaciones ya determinadas.

Estos parámetros han sido agrupados y almacenados para cada código.

```
Dual
 63          45          3
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 189 0 0 0 0 0 0 0 23520 0 0 0 60480 0 0 0 116739 0 0 0 47040 0 0 0 14112
0 0 0 0 0 0 0 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
 127         113         2
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 4572 0 0 0 0 0 0 0 8255 0 0 0 0 0 0 0 3556 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
 127         106         3
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26670 0 0
0 0 0 0 0 493776 0 0 0 0 0 0 0 1176655 0 0 0 0 0 0 0 384048 0 0 0 0 0 0 0 16002 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
 255         239         2
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 3060 0 0 0 0 0 0 0 23120 0 0 0 0 0 0 0 16575 0 0 0 0 0 0 0 20400 0 0 0 0 0 0 0 2380 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
 255         231         3
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10710 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1370880 0 0 0 0 0 0 0 3588224 0 0 0 0 0 0 0 7568655 0 0 0 0 0 0 0 3166080 0 0 0 0 0 0 0
```

1066240 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6426 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 626076 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 83105792 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 221819136 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 485999679 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 208375552 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 73328640 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 486948 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

II.4 ARCHIVO TIP_COD1.TBL

Contiene los datos de los archivos DATOS1.TBL y DATOS2.TBL, los que se encuentran ordenados en forma ascendente de n. Formándose grupos que contienen el mismo valor de n y a los cuales simultáneamente se ha realizado un ordenamiento en forma ascendente de t.

```
Dual
 7          4          1
1 0 0 0 7 0 0 0
Dual
 15         11         1
1 0 0 0 0 0 0 0 15 0 0 0 0 0 0 0
NDual
 15         7          2
1 0 0 0 0 18 30 15 15 30 18 0 0 0 0 1
NDual
 15         5          3
1 0 0 0 0 0 0 15 15 0 0 0 0 0 0 1
Dual
 31         26         1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
 31         21         2
1 0 0 0 0 0 0 0 0 0 0 0 310 0 0 0 527 0 0 0 186 0 0 0 0 0 0 0 0 0 0 0
Dual
 31         16         3
1 0 0 0 0 0 0 0 465 0 0 0 8680 0 0 0 18259 0 0 0 5208 0 0 0 155 0 0 0 0 0 0 0
NDual
 31         11         5
1 0 0 0 0 0 0 0 0 0 0 186 310 0 0 527 527 0 0 310 186 0 0 0 0 0 0 0 0 0 0 1
NDual
 31         6          7
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 31 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
Dual
 63         57         1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
Dual
 63         51         2
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 210 0 0 0 1512 0 0 0 1071 0 0 0 1176 0 0 0 126 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
```

```
 63            45            3
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 189 0 0 0 0 0 0 0 23520 0 0 0 60480 0 0 0 116739 0 0 0 47040 0 0 0 14112 0
0 0 0 0 0 0 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NDual
 63            10            13
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 196 252 0 0 63 63 0 0 252 196 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
NDual
 63            7             15
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 63 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
Dual
 127           120           1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 127 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
 127           113           2
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 4572 0 0 0 0 0 0 0 8255 0 0 0 0 0 0 0 3556 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
 127           106           3
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 493776 0 0 0 0 0 0 0 1176655 0 0 0 0 0 0 0 384048 0 0 0 0 0 0 0 16002 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NDual
 127           8             31
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 127 127 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
Dual
 255           247           1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
 255           239           2
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 3060 0 0 0 0 0 0 0 23120 0 0 0 0 0 0 0 16575 0 0 0 0 0 0 0 20400 0 0 0 0 0 0 0 2380 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
 255           231           3
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1370880 0 0 0 0 0 0 0 3588224 0 0 0 0 0 0 0 7568655 0 0 0 0 0 0 0 3166080 0 0 0 0 0 0 0
1066240 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6426 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

0 0 0 0 0 0 0 0 0 0
NDual
255 9 63
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
Dual
511 502 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
511 493 2
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 69496 0 0 0 0 0 0 0 0 0 0 0 0 0 0 131327 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 61320 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Dual
511 484 3
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1563660 0 0 0 0 0 0 0 0 0 0 0 0 0 29744288 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 75448639
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 26244960 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1216180 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
NDual
511 10 127
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 511 511 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0

```
0 0 0 
```

APPENDIX C

Generator Polynomials of Binary Primitive BCH Codes of Length up to $2^{10}-1$

The generator polynomials of all the binary primitive BCH codes of length up to $2^{10} - 1$ are given. They are given in an octal representation. Each digit in the representation is coded as follows:

$0 \longleftrightarrow 000$ $\quad 2 \longleftrightarrow 010$ $\quad 4 \longleftrightarrow 100$ $\quad 6 \longleftrightarrow 110$

$1 \longleftrightarrow 001$ $\quad 3 \longleftrightarrow 011$ $\quad 5 \longleftrightarrow 101$ $\quad 7 \longleftrightarrow 111.$

When the octal representation of a generator polynomial is expanded in binary, the binary digits are the coefficients of the polynomial, with the high-order coefficients at the left. For example, consider the (63, 45) BCH code in the table. Its generator polynomial in octal form is

1701317.

Expanding this in binary using the code above, we obtain

001 111 000 001 011 001 111

The generation polynomial is then

$$g(X) = X^{18} + X^{17} + X^{16} + X^{15} + X^9 + X^7 + X^6 + X^3 + X^2 + X + 1.$$

Primitive Binary BCH Codes of Length up to $2^{10} - 1$

n	k	t	Generator polynomial
7	4	1	13
15	11	1	23
15	7	2	721
15	5	3	2467
31	26	1	45
31	21	2	3551
31	16	3	107657
31	11	5	5423325
31	6	7	313365047
63	57	1	103
63	51	2	12471
63	45	3	1701317
63	39	4	166623567
63	36	5	1033500423
63	30	6	157464165547
63	24	7	17323260404441
63	18	10	1363026512351725
63	16	11	6331141367235453
63	10	13	472622305527250155
63	7	15	5231045543503271737
127	120	1	211
127	113	2	41567
127	106	3	11554743
127	99	4	3447023271
127	92	5	624730022327
127	85	6	130704476322273
127	78	7	26230002166130115
127	71	9	6255010713253127753
127	64	10	1206534025570773100045
127	57	11	335265252505705057517721
127	50	13	54446512523314012421501421
127	43	14	17721772213651227521220574343
127	36	15	3146074666522075044764574721735
127	29	21	403114461367670603667530141176155

n	
127	
127	
127	
255	2
255	2
255	2
255	2
255	2
255	2
255	1
255	1
255	1
255	1
255	1
255	1
255	1
255	1
255	1
255	1
255	1
255	9
255	9
255	8
255	7
255	7
255	6
255	5

n	k	t	Generator polynomial
127	22	23	12337607040472252243544562663764 7043
127	15	27	22057042445604554770523013762217604353
127	8	31	704726405275103065147622427156773313021 7
255	247	1	435
255	239	2	267543
255	231	3	156720665
255	223	4	75626641375
255	215	5	23157564726421
255	207	6	16176560567636227
255	199	7	7633031270420722341
255	191	8	26634701761153337145 67
255	187	9	52755313540001322236351
255	179	10	22624710717340432416300455
255	171	11	15416214212342356077061630637
255	163	12	75004155100756025515747245146 01
255	155	13	37575130054076650157225064646776 33
255	147	14	16421301735371655253041653054410117 11
255	139	15	46140173206017556157072273024745356744 5
255	131	18	21571313147151015126125027744214202416547 1
255	123	19	12061405224206600371721032651614122627250 6267
255	115	21	60526665572100247263636404600276352556313 472737
255	107	22	22205772322066256312417300235347420176574 750154441
255	99	23	10656667253473174222741416201574332252411 07643230343 1
255	91	25	67502650303274441727236317247325110755507 627207243445 61
255	87	26	11013676341474323643523163430717204620672 2545273311721317
255	79	27	66700035637657500020270344207366174621015 3326711766541342355
255	71	29	24024710520644321515554172112331163205444 4250362557643221706035
255	63	30	10754475055163544325315217357707003666111 7264552676136567025433 01
255	55	31	73154252035011001330152753060320543254 14

n	k	t	Generator polynomial
			32675501055704442603547361 7
255	47	42	25335420170626465630330413774062331751233341454460450050660245525431 73
255	45	43	15202056055234161131101346376423701563670024470762373033202157025051541
255	37	45	513633025506700741417744724543753042073570617432343334764435473740304 4003
255	29	47	3025715536673071465527064012361377115342242324201174114060254657410403565037
255	21	55	12562152570693265600177315360761210322734140565307454252115312161446651347372 5
255	13	59	4641732005052564544426573714250066004330677445476561403174677213570261344605005 47
255	9	63	15726025217472463201031043255355134614162367212044074545112766115547705561677516057
511	502	1	1021
511	493	2	1112711
511	484	3	1530225571
511	475	4	1630256304641
511	466	5	1112724662161763
511	457	6	1142677410335765707
511	448	7	1034122337164372224005
511	439	8	1561350064670543777423345
511	430	9	1727400306127620173461431627
511	421	10	1317711625267264610360644707513
511	412	11	1337530164410305712316173767147101
511	403	12	1573436303657311762726657724651203651
511	394	13	1102510344130333354270407474305341234033
511	385	14	1775545025777713737245545210730053033144 4031
511	376	15	1116744706521725533227162607146216210106733203
511	367	17	1126657202505666327301700165224556261443551160065 5
511	358	18	1574456154545450414733416176535150070037760411373255
511	349	19	1455012234675753242074501355657377036165521557376050313
511	340	20	1036075427062474623220662047122611677363511364110105517777

n
511
511
511
511
511
511
511
511
511
511
511
511
511
511
511
511
511
511

n	k	t	Generator polynomial
511	331	21	1540644721106050570342772405117177453215 1366636774616414573 21
511	322	22	1153614675506111211374366675624670755236 5236450626770617707350773
511	313	23	1037043005346411102774516447047070775602 3272246374215767362515174377
511	304	25	1510736212272130353502352536320703410147 2252730643370771600352540473351
511	295	26	1112620477530037170044524747727767532752 0466126030774720522476717444670355
511	286	27	1317463403265645042064775326044775737416 7140717560167145235500227345054014771
511	277	28	1111170752122547100341422773660302256230 3177512454137173036077374264015263256045
511	268	29	1241160247151767165615372023170221264427 2264376531637043503436310631425301735295601
511	259	30	1121314111162101573237072224371101446333 4772560250516566143547137606623504332146 46117
511	250	31	1007276607431444342624512277577275223746527731051537632575776073523533533615716763734724 5
511	241	36	1142630652622241570502646054544323040670124327763216605656641710317303600602611251467341721
511	238	37	1266115765104357454721732420217755547411137707517745524060575503345253220722132126634205537 3
511	229	38	1441474125377630733466544074104363527262511060250774255674573537565237245201524721767740003105
511	220	39	1330350650041246137409371541263763570720143157533327603264502752030077406436030760454544342665055
511	211	41	1537262671145755536471271211466022057474566447705125165503161712400714003214471340000077557120303760003
511	202	42	1355074061631664750550654005233724126373550161741774732644455207710047076760727737540417425312657744477
511	193	43	1151502312317462115717012400337054150716235761555206011476735120457325726004547342567567367350507947042505 5
511	184	45	1320551776766374341777223460740727550552041306675275255055262710144437034513676516425264417226510021654375056 71
511	175	46	1202260566674312437743724246412313100673370376656730772744671461151665006234221 37

n	k	t	Generator polynomial
			21420076227057757613571573445503
511	166	47	1105764456740764621742144442303723134662 4415551173770646017645044577670522052736 1070224470173567201512272211433361137
511	157	51	1710171171400604264434731140711521370753 6443012114236665478350701102415515624310 1741614000045173736937477155141455655503
511	148	53	1414131546120371255541603030602601124642 7727350174267055345645560670707547107070 5355774504056620750463641124465774123024 61
511	139	54	1775336143053070047771371572232076147655 0747705134625235605337076513414300020363 4 1267612204557536475125547007426342706405 51763
511	130	55	1126210420631670733262507415577025266400 0744725674510475154245554654156641313571 4050203537624233123030520531734566124772 13206321
511	121	58	1541271357655772552504106044116552170175 4434750270766625331253243027677423251 21 2771735535703734317405374311756764357501 74233577257
511	112	59	1264236440177126554561720120275756614321 2522620050735005241425745450416057221261 2131152145444635107565754242126117753372 24522460653765
511	103	61	1267502203437327536237202040115050677105 7315147230644142256367333406167475350432 5307201333612360344967516222665026547330 7 6535730737201043?
511	94	62	1034624554522076136543376246276501412751 3662275012007006252201651305170664740707 5134565753153573011560335354474211375326 0263677242075454303 5
511	85	63	1311423305612705436044231240030555360613 5536677204454060166477652423477032177140 3157621106134423201520207000161502002022 447445122446654632443215
511	76	85	1266305771250774445612456156722760607324 4460173771073371716611544036044377532354 6726213462571106044226351563717124262125? 527050661426327741052371 05
511	67	87	1572123711765346653777251761251635002412 6410254565097065546756015043572302057577 4010547010044577034567736504540164416765 5264502000655246473320030 41
511	58	91	1057521760004167050454306543214444576277 1110323251120106217732446025315645653300 5122220411465454201703211427513204445624 3035667754341253323906023033060 1
511	49	93	1400373125226245616437770021451433477a715 0420652515175422234337765344626250514664 2652723355123643637615716343345655723373

n
511
511
511
511
511
1023
1023
1023
1023
1023
1021
1023
1023
1023
1023
1023
1023
1023
1023
1023

n	k	t	Generator polynomial
			5677302374217750307670201352417267 3
511	40	95	1065643260601413014144161047625313166664 6125563104544564702042267527604002367722 2071247202762236702640231554475163650230 5426171724440024240371417745574027662 3
511	31	109	1004032013504641301465031022773733377231 2030141207564005615307277456377116271511 6355127767095745123620162301722500173755 1261347775206565234212557646767220145745
511	28	111	1506424217016105771311126531671315537006 6227135624440407472104067534437249506204 0406060362307553063752105071191510111346 0677710774225141420173656560126201213355 31
511	19	119	1323204574570661164218403733264114023404 6117057401637365536154143137744473020704 4764741324776540766743375511016032261143 7052430104522110303151617354750363145550 46713
511	10	127	1702172311027431733521266437277042605327 7252024574422247642034145430713500264725 3477467246601133713266536565611502362530674 7473677673467052234355271140607221533731 42701037
1023	1013	1	2011
1023	1003	2	4014167
1023	993	3	12052210423
1023	983	4	30135372217233
1023	973	5	67441634100257771
1023	963	6	155441273452021342055
1023	953	7	321370747475547513070313
1023	943	8	760744225715270200004506345
1023	933	9	1723526661245521113217162255031
1023	923	10	2023237673202230444160563331425627
1023	913	11	7755535722250515754561135410204703015
1023	903	12	1762347256320450777571663702727143167257
1023	893	13	3040745710762554654627001731571615206417 2421
1023	883	14	4773627447536521222606523723312270030314603 4273713
1023	873	15	1373673270027255303030504571774101007765122140012433
1023	863	16	3440272156360444435173063534010335535011 1403027714311 5

n	k	t	Generator polynomial
1023	859	17	1322555466145551135464235057214741470461713573565724014 1
1023	849	18	3542752065223573717143677007420570562622022055365127445003263
1023	839	19	5210617676532176677755405752642074636403114403434423611211665
1023	829	20	1213222404504163616603711027501442202705775313007201145723077423013
1023	819	21	2214325742216310353650417376464144271657512743132006172163242715574077
1023	809	22	5552750420617757113515375401021454054142420205523074235327310346075461 17
1023	799	23	1652724250022427704512246362640211603371644630326371057256370251530024042153 7
1023	789	24	2246066255654012757523054141151344242436432467416546402446171120753671556534655
1023	778	25	7403630677220307364435256070064531622417215674744553262174345601100430255376321151
1023	768	26	1420057410715701076671164654273645645222551071265714762242013656351251553254760660406 7
1023	758	27	3544663745163634242516250415043615446630124267770052140423356467454571650356504601530467 5
1023	748	28	5160035350014231172374207074650122560340672526554241370010604127365261122761375461377775161 7
1023	738	29	1761052465717574451310664156407472743161202550104453674154007006574144543672067742157137076252 77
1023	728	30	2705040500317205454745436524651543310405091746500522251232136047665107211450313736020717750111371 27
1023	718	31	6421001547326760050675717264533767251455475576414377211120071232560432370162740036374736232532765730 51
1023	708	34	1221657751474373730117767342335416045271431531536775745164572646667711733152545050454154042571133265734145
1023	698	35	2234207617247622765257211144622334473376364727336723310074551547131171343755474775721032446463507742155706124 5
1023	688	36	6062331567761473445053573264026122102305320144000665606245030755554423340530027140456114002742376254456531124177
1023	678	37	1173340210112354267372717132007746772117243024207506006027243526735623176571427007034326134214711531560844542303567 5

1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023
1023

n	k	t	Generator polynomial
1023	669	38	2726674363712255027500744177372151741454 5645473073762151476755130543142214595356 4055326754775417026420350254437447064123
1023	658	39	4526553271372400577644374714763070706741 1031171237615543024630303133511357737123 3160732343127163117047027566041001443541 05
1023	649	41	1053012656042512406741405367113520216337 3335151361445051301722153706537340534000 0442526605347576637101417411054700433515 223273
1023	638	42	3760125774555145576316166607300752402515 4405641372077331770676406123070777066044 5666510673072260546441216265142152656464 533676375
1023	628	43	6421516056255057702552057134724545361312 2357027016417012220103012744645753301127 6724371741000104347301751703445213150012 1 350470437063
1023	618	44	1436767275404102542246475266313571611600 5027177636022064473162157041345644331322 7260670220051221512570532647243752555477 5754664246713353
1023	608	45	3205632662501732326372612360531506221660 1642671653635152412144657627270612401323 5612466273254637300536451672472606153046 216157047010356654 1
1023	599	46	5175076274225613532026031146621612144714 3656373757601671144563673511345562033377 2363115242327614411151057011653316234767 2420714406612507752125
1023	588	47	1425717410513115113640172157602722701146 3011170047424744030123121316566534441253 5602227567425313345476331147701436161533 0 0452703626243361464515 2063
1023	578	49	3177430361201520403652734206337256777357 1514665321603455147706422374545553322214 5746177477620442053330427030577251714537 6 3640742553727574413301614 137
1023	573	50	1355677017526536535146475314700031707456 3026607614213013666657500772520767133533 7304265646727506154132227032714377077023 2341637207202520573521417420615
1023	563	51	2772361504424321676653141026604362002113 2553454106501612101536430475161545700052 0475027073570114161514602647142254455044 6255557776515151167236204235327161
1023	553	52	6042334623077055525431324656261417515014 2274125133435505177502357332260547747255 3162040767437444775076764454714640345671 0027561254427647334263055360460303115
1023	543	53	1055532301605012610131175414035550021456 2763276437435575056231126247124212102427 6704204717425615057004044214740647640450 0651662113626166471160577340552720505314

n	k		Generator polynomial
			3
1023	533	54	3011252271024007601256355614410110631241 4476043760457564721111127466077665562665 4217667430146671642613617516247055413742 6636741420639626224221121140152405654631 6753
1023	523	55	6173100556046353561760536040447255721247 4310077565636320561601165471476057132554 2322473307203231122443324706157714547610 2204305664142377402105560521763116372354 2140361
1023	513	57	1442056607323310761242261470690262276017 5101364127621557726777406532310570775461 5277217655753122205474040207573055575465 4557253141504105143444175702256556213675 47711533103
1023	503	58	3526111340256374267000600572256610352600 0577617220337007505520131512037473367771 3400541207320422505347425362014026650010 0004257043231653414626355376624351306710 0231156525073 1
1023	493	59	7525405755220775623405534445735062435500 7256600605234131432506166510202242405073 3356624720337475566117114661531204767172 3111426350630167625357243023251(30010175 4522605016124500 1
1023	483	60	1010227406455200037623641637202403117556 7 5534053006400654636246221506521032564174 6035764603621630460402150663270555030262 2471552611215057442745721707226536776112 3250716264476677241 33
1023	473	61	3275255734777512716766130324236300561602 3266541073200606432722454332243070233312 1056616662644203305305353270024721414270 6252777265242372623345602666164360155136 2 3601147120220715635225 41
1023	463	62	4257710770354177520700120310175131741452 2646316040516304114052431351313706134426 2044226555241044230276332312417700133134 5011373217527767452307776456313356256212 2754010715255456170275521 61
1023	453	63	1531237560135257466053045644313751237706 0514513604361025341247434430553661067700 1052325072756200027154257505607050746074 2347574503151051424560227041423702376426 0257036602570516146760636153261
1023	443	73	3051462775160646577510534745450470242247 4353060560266203436041336276663363011462 7247152315630145650707511617214040605320 4277716062400570267044607301501125151777 3374535517744726746624051117614557
1023	433	74	6551077013564477321042777730023704151601 7521127762157247770233225337240166276371 2515477552103412502262702245120233465315 5173725652335621405465526246704034012300 7073271127075457754155273245775745013

n	k	t	Generator polynomial
1023	423	75	13714035457403256597347206626414456055720647077141650392015063042536766727141035356347426042441521145714214012236710237416461244413037947910215037201037260662794604725046607414253305030001515377207422 7
1023	413	77	31305145510133715020775103957325573214011760073161356760432560144747314167124255537617324764260251735701062542725777645577342664455451200737327744750516575245501423614017265745456752500412104234734710756 7
1023	403	78	65305037066255621241415646754247347562575145101332262333026444047152073731026412746244005571621101627774403723066305531641276532452533535412724676631061133755763741616153022635545907234742671051423067665 7
1023	393	79	16535067014014227112444215052132557412027657471014596751746344362763209264534655662553450526474576514340077272347752464237175651200140713732734754325241174056635606472676113552676751152425134176727371570113232 51
1023	383	82	21015563503162746367217740325213463747550416045710713235213260025017310331101075115245351779026643055165034363360301633035437162222460442722136365276712407113617655761511013226737136510205127510377572757105036743057
1023	378	83	12717272324756730326454163410276443220544742747152233707376724600630560464152160615441501471172331343574553307577366042631000165112524534543065554666652167725040134511004147527430357700216203600436576356231446723126 7
1023	368	85	26671102443067175541166705357723525557220775135077721751514126771255024621053205131402256746373217236124117120141202616036533337504716610551526334725345224307400302522242423267144120672542411721301423732163174126032000111
1023	358	86	74351213725645604161644234055125311312734223331115032441547715500032344047574426171255246153400574361641566444466317375416017412505517110423075644367410210075051557253227575466534415533264056346651307314476261042544230245
1023	348	87	11521505704365102657666741361413034742714526012535312616667574523032536722700050262520703520133277114107013243424325560616444535277531163014473676344247112447577501500116162562637711717707510221321717043446340305445612706534703
1023	338	89	34354047442656775023203515551341723046330127646153113200255460143043276470736246612540101534107337707374677154110230773340256661536205570426770537032370455260654431671722472654734753225325512764361134 1

n	k	t	Generator polynomial
			662013445777124440573650334127
1023	328	90	437607556645646444772552331015705572505133647524031750615371350142265255155355275615544726357754527476450766455170760505427007214641140377770237266130173760004726506446215750756233157425776556517457041307254743603644556731767667760215 57
1023	318	91	170657037431513074456621534174535767354465065170740573076067577417165717702306704666124321175656667111343514112511526132576723461212753733656711052041200623667306520253655456236102664653735764157357076757337145232061011411327037625642457 1
1023	308	93	264212032004771762223114063167326340740465466227174626344411403442435052126716431357154502540667346217774070517733634072115442255530645251537001215776545771547624746324557115247157356012040303003625161137157136535674357547562725561237524 7
1023	298	94	760731512576143364252630445036354562647256117437215207330563431571502255405523160767511126112777171451211352175276770466732556644427514042164046110636365656566455034067154627370421533112512724113526716703011246441157106440703162714663311067 73
1023	288	95	170160406663341677472644472526165055106721244107460021414162674652011221024551676755376270304423013326176347761574051225216105112245612754346505266663174200542522666553175534607545317323256676205467564036613660503377260277430301175367621203555
1023	278	102	256111241320400457251232342701512154042432371230711017550372160652527753082235474065127152454745331164320065161070571534437124732460771702704144626140302447623427511131234135442316432174203352577141500236421250623237270512605242017233245075531042463 5
1023	268	103	565403573775664446033441413303011114631001461526216217547461442050120553305270775303715417371641000727253627610744441051570344631626533260251275302723026125225702305666676751206333125030100761713443777330061102755376502137425117533736240637177561306317
1023	258	106	147271656065455350447611442733020015011511734721116611464727105364572640746167306212707640513421202420267426242134553637042751770405346314032145302444637747011207615064307673471464265724037671255733022035561045771534136727635500077572471130110646647103743
1023	248	107	346147771037654324141437523163651657472450653162301374561204613235745415257260412555673562616526111127656576320525763510564402431214565262713275606350620145456005036572154411721004024763065314306235 26

n	k	t	Generator polynomial
			5426422633275763410271325014165633012203 241479550462445315
1023	239	109	610705550557200726271447241566544360172747733215354726555667655160743012005321165555655214627411766175714545116221415404212440253115426514574524637271755667277474213367104512121722437505553242361346525470577265727407254047417505073237114455226722521136165545455
1023	229	110	164559527260235964631771217243713545732434317660426124677350457914236351070225325405654403507065023411771574425452536142770713564045442575566207222015607240167503450874704032334543505710424407412757710407046017740711025700727262205172152134764771653006051570762272065
1023	219	111	25327277770671464561273040354055633452457027263257763231533011541245200437451417442212456466152657101207725274527152722757241753564030274560724415144761371307516636246255510507206460643660473725334277617777131471461106011622523472737560215335235114737632637244307010474131206751707707365720257526643
1023	209	115	766624255442712461342450735326550617777064311756474450153204750201344173536375164336117420142562704175424403275607244151447613713075166362462555105072064606436604737253427761377771314714611060116225234727375602153352351147376326372440307010474131206751707707365720257526643
1023	203	117	2600721761002761741110660757163503757520652515753726740735733666613116364712327365275645254765562351403631501114054010175722401727732201771677650255740570006741171204577100724037757230158754132761535701153221436207760470741237213426270541570460222541522736535252277560237265234
1023	193	118	705331572703026700507653766244662660136130617771711501237035176023236343301601224404504733611375240564516452123165075255244134130153077772431444405634017667015257142421560732575542156515504017777245056754520716611074711622221374361754031733166505125521764903215222751413571734
1023	183	119	13474734747752700447205570040555432536415414261074530641366160044747074270440647635172546333706616423742574131000301015133453374232473772046203334555337543714510705407216647156127045077052715450171630223326100135517107647241473151161146625775631040701323452210223740720176570172124 5
1023	173	122	2361302433102534366303174057425000312611540771444611324127036701625054551212710741460774212445352751521455704353777541205555475532041352234100663235537577223261445755353554040347110214711676430632176060321150534032173701762406103076641731454011366670403057642710113361765614553157553

n	k	t	Generator polynomial
1023	163	123	4113171037325201434290011477507442016774417325213737663530655624555754314734527735542541130534076025251223426330456733674367764165157746277175651756475025772370722175615025353051266010214741762003013355457103141643542557151144254001725320301746564422054740357115621457437242510375164660607
1023	153	125	1473037520530655305664534611401066464651065664662407502726452471015327170011327355125525040144624177221035154710077234137302372610647546174210411267454304252746474517754075216775076710674212553623331766443174253062657273071331153660071250532432044341796551447506642664766242225300324447505 11
1023	143	126	2705672534137537656134605524750654047625427542325751256210276141606761777054277114600136642475746100406176263715432660573554365713174265772110154773404721770511137337474461477465030623324775750377403151165171374252046732556770526522266602575274624053121143125413151517111550525107627337477461525
1023	133	127	4617241122502533137567475531722511566715742570164460235417241347347776724542114554447464632770001337721543511324751679261615450366572057420352543740750634363666271454562205232753334923261432570550116220123731674052034715703701044271032426760725120016714416477507015325167127425643765453221150665
1023	123	170	1606122475314452440364114065506377666426534361070276027731002455270534360042332024243331051461443927327126143431675314002300402653162631026523102264715527745315037416221137636061555727344465325561056616142031043474435405337273112044660354352112534202311077160374362047252557520614465106727272047164066065
1023	121	171	5222675662147725741314744210723377001541225327251472137417015727453225320357006155550017334624741443045642452515062144317113141412153431714127171741654024767214313645376763737222402344427761205012731025245611735266752042303471766736003205226765225617177275521033336364525016762247631720444446365514222 13
1023	111	173	1522057216652345524151261131756011401100241745735741155626725743064423544552377231414143626430623756035136777711661732620240003715057120763521556703412721351506644410720607694354466343145651543147763137320403036366427176704273476157627362355036760052146207425047255426724234354457567034466340442617646443465517
1023	101	175	3545472772647733506027327007274644770357707647401611203443340224220721243341250342125356260652701426057761347173573614656417762011214767217553555554653572111747

n	k	t	Generator polynomial
			4551152615662076355010433437412255115177 6272636505203446412116131004765025047506 2051240151567124466146277070142637071755 557546772101677262472416431
1023	91	191	4055177762504121412577610205372007777645 6673574005261545600620674271417162617312 1596256105240557423647764737643012341017417 7126672067376365721700727456241223731307 2756306103623641555001215465766126437034173 7306700214455577041500501577212157321576 7374543304701503707161451114254726044161 1735616377403703647742331023117
1023	85	193	2264761771543214471677727410753621637726 1160424142731011126327103066573247413075 4432257727721744044647425016445437057617 3267260210057106377253361731750152677206 0076374246566643157105441055355777515755 3364312466543275541283126163776745507642751 4370702513763060274463513676021143763534 5342217036156026623166234073223
1023	76	197	5504413226010767557023243061737064215573 4421653247737231247224371647516170367102 2332107021677452321437245440364276246075 0627166653459647114065537041525547007537 0021311306574650326702510673440306147402 0127524575753074316742263029463722341655 5043324273005722150256144415377275030403205027514351137164171510053060456776027
1023	66	199	1335505423444174140615752575457717024302734241736254711233675513427142342234405354423121174602724207250656224065152764744252257354041261416250666831614143346370023253213327457376454172114672756045050327145741474335547265552510727366670371354456044605352522750777052251520016127100460052306423411704545764370377417042755706037
1023	56	191	3533660424156746206407772444526075003401332704443011175576405421367126140763772177651500664529355030207820316170137045061451154164441245647737260347755041337423134332421423334727666016525177347306160053437677231213602456402116243524676655254225620424553427435741752070224364277643622113551667237042145430541554174713125325173331
1023	46	219	6301541625665623020521356602744545517117552653111473567026914175574471746710331370637137663547267635501752647104447676250677337042247442410501466045557603305365763031370740762367221675415372064403277036042232014600011121247511373712657055313174645762574576401050005016160506604517273022627407551221052152200421636255324376075
1023	36	223	1152711551526005136004274504217435302475043722143566497300517123722015114006470754419340557473710166551273015664141311316371170423356177113076413423724466722315076131665466450263251276057160641615276437407144501652522375667325070030347777735054571207710715200022172125536140461363

n	k	t	Generator polynomial
			221165744342510506161537616206140423274129275021 43
1023	26	239	3476655146625052703141431762032154723323 2124522103643311317642327332020241645172551 3129645425737600220635005146600630723670 4734745250307771910643301537623156667417 3 2473210524773623011063773741204666445005522 2530732634726564174115140770457327674476075 7766775101312115727255145732774036010471 3655061034476144410965213573204720215255 6455751231633
1023	16	247	6036262767731160101141105063701021363565 3763274052211422564557762162121174115786 0750767504220703652612124677754243717505 4060630501092657505104156311506724744425 5411743547305156755340244015309107719436 3513041170531505737691274562266054152255 5633231652623573617766360711322000755023 0251544067462673641704456634322574227213 4736703723114761
1023	11	255	3435432424205341325756012523570556322460 1630065267150113075623353102124314210612 6137022171665514357225426351765550103471 0474154327571027371265731664405072157011 5641142017433246415376511474510565005614 4327127466737311303120752456366004341761 1165603713302035245727751103035601161 57144763771112120752675154767337577666 6240634967040361 77

Inde

BIBLIOGRAFIA GENERAL

1. Shu Lin, Costelo, ERROR CONTROL CODING FUNDAMENTALS AND APLICATIONS, Prentice Hall, 1983.

2. Gallager R. G., INFORMATION THEORY AND REALIABLE COMUNICATION, John Wiley, New York, 1968.

3. IETEL, TRANSMISION DIGITAL POR MICROONDAS, Centro Nacional de Capacitación en Telecomunicaciones, Japan International Cooperation.

4. Roberto Angel Ares, SISTEMAS DE RADIOENLACES DIGITALES. EQUIPOS, MEDICIONES Y CALCULOS, Lenkurt Telecomunicaciones S. A., Santiago de Chile, 1989, SIEMENS TELECOMUNICAZIONI S. P. A.

5. C. Belove, ENCICLOPEDIA DE LA ELECTRONICA, INGENIERIA Y TECNICA, Vol. 6, Oceano, España, 1990.

6. IEEE Transactions on Communications, Vol. COM-32, No. 4, April 1984.

7. *IEEE Transactions on Communications, Vol. COM-30, No. 2, February 1982.*

8. *IEEE Transactions on Communications, Vol. COM-30, No. 5, May 1982.*

9. *Telcom report 11 (1988) Special <<Transmisión por Radioenlaces Digitales>>, SIEMENS, Enero 1988.*

10. *Fabian Luna, RED DE RADIOCOMUNICACIONES PARA EL IESS, Tesis E. P. N., Enero 1991.*

11. *Kamilo Feher, DIGITAL COMMUNICATIONS.*