DIVISION DE ESTUDIOS DE POSGRADO

FACULTAD DE INGENIERIA

UNAM

CODIFICACION PERMUTACIONAL

TRABAJO FINAL

PARA PRESENTAR EXAMEN DE GRADO EN

MAESTRIA DE INGENIERIA ELECTRONICA

PRESENTA

ING. JOSE ABEL HERRERA CAMACHO

ABRIL 24 A MAYO 24 DE 1985.

RECONOCIMIENTO

A pesar de la brevedad y sencillez de este trabajo quiero constatar mi agradecimiento a los doctores:

FEDERICO KUHLMANN RODRIGUEZ
ANDRES BUZO DE LA PEÑA

por haberme motivado en este campo y sus valiosos comentarios; así como el Ing. JESUS SAVAGE por las facilidades que me otorgó en el uso de la computadora PDP-11 a su cargo.

I N D I C E

I N T R O D U C C I O N

El presente trabajo tiene como objetivo el estudio de un tipo especial de codificación vectorial llamado codificación permutacional. En particular, se aplicó a muestras con distribución laplaciana y con distribución uniforme.

El interés de la codificación permutacional reside en el hecho de que su comportamiento óptimo tasa de variación[1]-distorsión es idéntico a la mejor codificación por amplitud y, sin embargo, difiere sustancialmente de ella.

La codificación permutacional fue introducida por Slepian en 1965 como una codificación de canal. Su primera aplicación para codificación de fuentes fue hecha por Dunn en ese mismo año, para fuentes con distribución gaussiana. Posteriormente Berger, Jelinek y Wolf [1] presentaron en 1972 un estudio de la teoría de la codificación permutacional como una forma de codificación de fuentes y su aplicación a fuentes con distribución gaussiana. Berger en 1972 [2] y en 1982 [5] compara la codificación permutacional y los cuantizadores óptimos. Finalmente, en 1981, Townes [4] la aplicó a fuentes con distribución laplaciana y a voz.

Cabe destacar que la aplicación de Townes [4] a voz no fue satisfactoria, aún usando técnicas predictivas; la correlación y la variación, en el tiempo, de la estadística de los bloques o vectores desmejoran demasiado el comportamiento de esta codificación,de manera que codificadores más simples ya usados lo mejoran en la codificación de voz. Sin embargo, no han sido estudiadas suficientemente variantes en la aplicación de la codificación permutacional a señales reales, siendo un campo abierto a futuras investigaciones.

---

[1] del inglés "rate".

C A P I T U L O I

CODIFICACION PERMUTACIONAL

Sea una fuente discreta que emite una secuencia de variables x_i aleatorias reales y con una función de densidad $p(x_i)$ continua, estas salidas pueden no ser identicamente distribuidas ni independientes.

Nos interesa la codificación del vector salida $\bar{x} = (x_1, x_2, \ldots, x_n)$ por el vector codificador $\bar{y} = (y_1, y_2, \ldots, y_n)$ que pertenece a un conjunto M finito de m vectores, tal que $\bar{y}$ minimize alguna medida de distorsión $d(\bar{x},\bar{y})$ ya preestablecida. El promedio de distorsión por salida es

$$D = \frac{1}{n} E \left[\min_{\bar{y} \in M} d(\bar{x},\bar{y}) \right]$$

donde E es el valor esperado respecto a la distribución de $\bar{x}$.

La codificación óptima se haría, de la manera más simple, comparando cada vector de salida con cada vector $\bar{y}_k$ $k = 1,2,\ldots,m$ y escoger el que produzca la menor distorsión. Para valores grandes de m y n ésto es una tarea enorme. La codificación permutacional propone la sustitución óptima muy simple del vector $\bar{x}_i$ para una amplia gama de medidas de distorsión.

En la codificación permutacional, los vectores $\bar{y}_k$ son permutaciones del vector n dimensional

$$\underbrace{(\mu_1, \mu_1, \ldots, \mu_1}_{n_1}, \underbrace{\mu_2, \mu_2, \ldots, \mu_2}_{n_2}, \ldots, \underbrace{\mu_k, \ldots, \mu_k}_{n_k})$$

donde μ_i son números reales tales que $\mu_1 < \mu_2 < \cdots < \mu_k$, k y n_i son enteros positivos tales que $n_1 + n_2 + \ldots + n_k = n$. Entonces hay

$$m = \frac{n!}{n_1! \, n_2! \, \ldots, n_k!}$$

vectores codificadores, por lo que la tasa de variación (R) es

$$R = \frac{1}{n} \log_2 m \quad \text{(bits/muestra)}$$

El procedimiento óptimo de codificación permutacional se describe en el siguiente teorema.

Teorema. Dada la medida de distorsión de bloque

$$d(\bar{x},\bar{y}) = g\left[\sum_{i=1}^{n} f(|x_i - y_i|)\right]$$

donde $\bar{x} = (x_1,\ldots,x_n)$, $\bar{y} = (y_1,\ldots,y_n)$, $g(\cdot)$ es no decreciente y $f(\cdot)$ es no negativa, no decreciente y convexa hacia arriba para argumentos positivos; entonces la codificación óptima de la variante I de códigos permutacionales es acompañada con el algoritmo descrito a continuación.

1) Sustituir las n_1 componentes menores de $\bar{x}$ por μ_1
2) Sustituir las siguientes n_2 componentes menores de $\bar{x}$ por μ_2

.
.
.

3) Sustituir las mayores n_k componentes de $\bar{x}$ por μ_k

Usar el vector codificador que resulte de estas sustituciones para representar a $\bar{x}$.

Existe sólo otra variante, semejante a la primera, de la codificación permutacional, que puede consultarse al igual que la demostración del teorema, para ambas variantes, en la referencia [1].

Como un ejemplo, supóngase que se aplicará la codificación permutacional en base al vector codificador $\bar{y} = (-3, -1, -1, 0, 2, 2, 2)$ al vector $\bar{x} = (5.1, -8.0, 3.0, 1.4, 0.5, 1.4, 0.0)$

Del vector codificador se observa que $n = 7$, $k = 4$, $n_1 = n_3 = 1$, $n_2 = 2$ y $n_4 = 3$, el vector $\bar{x}$ quedará codificado como

$$(2, -3, 2, 2, -1, 0, -1)$$

La medida de distorsión del error cuadrático medio

$$d(\bar{x},\bar{y}) = \frac{1}{n}\sum_{i=1}^{n} (x_i - y_i)^2$$

es un tipo especial de las medidas consideradas en el teorema anterior. Se construirá el vector codificador, de la variante I, de la codificación permutacional para la distorsión media cuadrática.

Sea z_j; $j = 1,2,\ldots,n$ la j-ésima componente mayor del vector de salida $\overline{x}$, y sea $S_i = n_1 + n_2 + \ldots + n_i$ y $S_o = 0$; entonces, la distorsión media cuadrática es

$$D = \frac{1}{n} E \left[\sum_{i=1}^{k} \sum_{j=S_{i-1}+1}^{S_i} (z_j - \mu_i)^2 \right]$$

como $\sum_{k=1}^{n} z_k^2 = \sum_{k=1}^{n} x_k^2$, entonces

$$D = \frac{1}{n} \left[E |\overline{x}|^2 - 2 \sum_{i=1}^{k} \mu_i \sum_{j=S_{i-1}+1}^{S_i} E z_j + \sum_{i=1}^{k} n_i \mu_i^2 \right]$$

derivando con respecto a μ_i , Berger, Wolf y Jelinek [1] demostraron que los valores óptimos de μ_i son

$$\mu_i = \frac{1}{n_i} \sum_{j=S_{i-1}+1}^{S_i} E z_j$$

entonces, la distorsión es

$$D = \frac{1}{n} \left[E |\overline{x}|^2 - \sum_{i=1}^{k} n_i \mu_i^2 \right]$$

el algoritmo para obtener los valores de k y n_i que minimizen a D es iterativo y se encuentra desarrollado en el apéndice A.

C A P I T U L O II

CODIFICACION PERMUTACIONAL DE UNA FUENTE LAPLACIANA

La codificación permutacional de una fuente con distribución laplaciana es importante, entre otras cosas, porque su función de densidad es semejante a la distribución de muestras de una onda típica de voz. A pesar de que otras densidades como la gámma se aproximan más a una onda típica de voz, es preferible trabajar con la función laplaciana por su mayor sencillez.

Las pruebas realizadas fueron teóricas, no simuladas, para una fuente laplaciana que genera variables aleatorias con la densidad

$$p(x) = \frac{1}{2} e^{-|x|}$$

esto es, con media o y variancia 2.

Se aplicó la variante I del algoritmo mencionado en el capítulo anterior y desarrollado en el apéndice A para generar el vector norma, en el caso de vectores de 100 componentes. Además, como entradas al algoritmo fue necesario calcular los valores esperados de las estadísticas de orden 100 para la densidad laplaciana. Los resultados de éstas estadísticas fueron:

-4.741113186	-3.494235039	-2.994234562	-2.660900593
-2.410900354	-2.210900307	-2.044233084	-1.901375890
-1.776375771	-1.655264487	-1.565264463	-1.474355340
-1.391021967	-1.314098716	-1.242670059	-1.176003218
-1.113503098	-1.054679513	-0.999123812	-0.946492136
-0.896491945	-0.848872840	-0.803418219	-0.759939909
-0.718273103	-0.678272843	-0.639810681	-0.602772295
-0.567054451	-0.532563567	-0.499212056	-0.466915637
-0.435598419	-0.405137122	-0.375454664	-0.346412927
-0.317857891	-0.289610624	-0.261477560	-0.233275384
-0.204872623	-0.176245376	-0.147541121	-0.119135521
-0.091663830	-0.066010721	-0.043244599	-0.024499970
-0.010820538	-0.002996055	0.002996055	0.010820538
0.024499970	0.043244599	0.066010721	0.091663830
0.119135521	0.147541121	0.176245376	0.204872623
0.233275384	0.261477560	0.289610624	0.317857891
0.346412927	0.375454664	0.405137122	0.435598419
0.466915637	0.499212056	0.532563567	0.567054451
0.602772295	0.639810681	0.678272843	0.718273103
0.759939909	0.803418219	0.848872840	0.896491945
0.946492136	0.999123812	1.054679513	1.113503098
1.176003218	1.242670059	1.314098716	1.391021967
1.474355340	1.565264463	1.665264487	1.776375771
1.901375890	2.044233084	2.210900307	2.410900354
2.660900593	2.994234562	3.494235039	4.741113186

sus detalles de cálculo se encuentran en el apéndice B.

Los resultados obtenidos para el vector norma se resumen en la tabla 1, para valores distintos de la tasa de variación (R) y de la distorsión normalizada (D/σ^2), en cada uno de ellos se indican las distintas componentes (μi) del vector norma, su número (k) y las veces que cada una se presenta (ni).

R	D/σ^2	k	μi	ni
2.9484	0.0645	10	-3.05 -1.83 -0.99 -0.50 -0.15 0.11 0.47 0.99 1.83 3.05	3 8 12 13 14 14 13 12 8 3
2.7029	0.0381	8	-2.75 -1.44 -0.65 -0.19 0.12 0.60 1.36 2.64	4 12 16 17 17 16 13 5
2.2526	0.1126	6	-2.75 -1.25 -0.34 0.21 1.06 2.53	5 18 25 26 20 6
1.8067	0.2025	4	-1.90 -0.46 0.43 1.19	15 35 35 15

Tabla 1. Parámetros del código permutacional para distintas tasas de variación para un vector de 100 componentes.

De la tabla 1 puede observarse la tendencia hacia una simetría en los valores positivos y negativos, obtenida por Townes [4] en sus resultados.

En la figura 1 se grafica R v.s. D/σ^2 para datos de la tabla anterior. También se muestran: la función tasa de variación-distorsión (RDF) establecida por Shannon y que representa una cota inferior para cualquier codificación. Su fór

mula es:

$$R(D) = \frac{1}{2} \log_2(\sigma^2/D)$$

además, el comportamiento del óptimo cuantizador escalar [4] y el comportamiento del óptimo cuantizador escalar por error medio cuadrático (MSSE) para una fuente laplaciana [3]

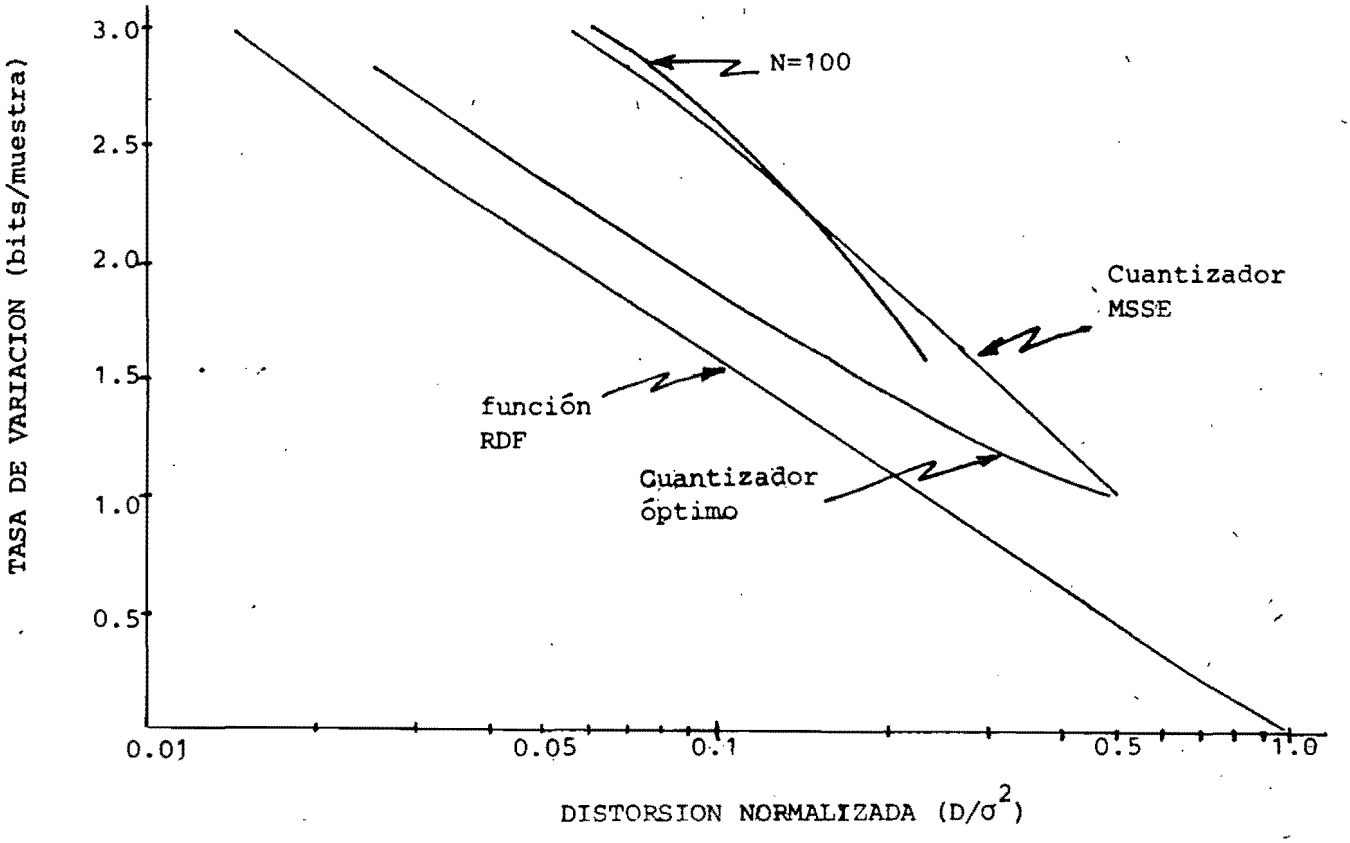

Figura 1. Comportamiento del código permutacional para fuente laplaciana de 100 muestras. Curva RDF y óptimo y MSSE cuantizadores.

De la figura 1 se observa que para valores bajos de la tasa de variación mejora el comportamiento de la codificación permutacional con respecto a si misma y a la codificación escalar.

Por otro lado, Townes [4] mostró que al aumentar la longitud del vector mejora el comportamiento de la codificación permutacional, pero también se incrementa el tiempo en obtener al vector norma lo que en aplicaciones de tiempo real es indeseable.

C A P I T U L O III

CODIGO PERMUTACIONAL DE UNA FUENTE UNIFORME

Se aplicó la codificación permutacional en forma teórica y simulada a muestras con distribución uniforme en el intervalo (0,1)

Se uso también la variante 1 del código permutacional mencionado en el primer capítulo y desarrollado en el apéndice A, para generar el vector norma en el caso de 60 y 20 componentes. Como entrada al algorítmo fué necesario calcular los valores esperados de las estadísticas de orden para la densidad uniforme en (0,1), éste cálculo no es tan complicado como en el caso de densidad laplaciana.

De acuerdo a David [6] las medias de orden para muestras idénticamente distribuidas en forma continua están dadas por

$$\mu_{r:n} = n \binom{n-1}{r-1} \int_{-\infty}^{\infty} x \left[F(x)\right]^{r-1} \left[1 - F(x)\right]^{n-r} dF(x)$$

donde F(x) es la función distribución de las muestras.

En el caso de distribución uniforme en (0,1), F(x)=x, se tiene que

$$\mu_{r:n} = n \binom{n-1}{r-1} \int_{0}^{1} x^{r}(1 - x)^{n-r} dx$$

y después de algunos cálculos

$$\mu_{r:n} = n \binom{n-1}{r-1} \sum_{k=0}^{n-r} (-1)^{k} \frac{1}{k+r+1}$$

lo que puede escribirse como

$$\mu_{r:n} = n \binom{n-1}{r-1} \frac{1}{\binom{n}{r}} = \frac{r}{n+1}$$

Para el caso de 60 componentes, los valores obtenidos fueron:

0.0163934	0.0327869	0.0491803	0.0655738
0.0819672	0.0983607	0.1147541	0.1311475
0.1475410	0.1639344	0.1803279	0.1967213
0.2131147	0.2295082	0.2459016	0.2622951
0.2786885	0.2950819	0.3114754	0.3278688
0.3442623	0.3606557	0.3770491	0.3934426
0.4098361	0.4262295	0.4426229	0.4590164
0.4754098	0.4918033	0.5081967	0.5245901
0.5409836	0.5573770	0.5737705	0.5901639
0.6065574	0.6229508	0.6393442	0.6557377
0.6721311	0.6885245	0.7049180	0.7213115
0.7377049	0.7540983	0.7704918	0.7868852
0.8032786	0.8196721	0.8360655	0.8524590
0.8688524	0.8852459	0.9016393	0.9180328
0.9344262	0.9508196	0.9672130	0.9836065

en el caso de 20 componentes, los resultados fueron:

0.0476190	0.0952381	0.1428571	0.1904762
0.2380952	0.2857143	0.3333333	0.3809524
0.4285714	0.4761905	0.5238096	0.5714286
0.6190476	0.6666667	0.7142857	0.7619048
0.8095238	0.8571429	0.9047619	0.9523810

Los resultados obtenidos para el vector norma se encuentran en las tablas 2 y 3 para 60 y 20 componentes, respectivamente, para valores distintos de la tasa de variación (R) y de la distorsión normalizada (D/σ^2); en cada una de ellas se indican las distintas componentes (μ_i) del vector norma, su número (k) y las veces en cada una se presenta (n_i).

R	D/σ^2	k	μ_i	n_i
2.7612	0.0621	15	0.02	14
			0.14	13
			0.35	9
			0.52	6
			0.62	4
			0.70	3
			0.75	2
			0.80	2
			0.83	1
			0.86	1
			0.89	1
			0.92	1
			0.93	1
			0.95	1
			0.97	1
1.6478	0.2634	8	0.02	30
			0.31	20
			0.67	4
			0.79	2
			0.85	1
			0.89	1
			0.93	1
			0.95	1
1.4578	0.2366	4	0.02	31
			0.30	21
			0.67	5
			0.84	2
			0.93	1

Tabla 2. Parámetros del código permutacional para distintas tasas de variación y para un vector de 60 componentes.

R	D/σ^2	K	μ_i	n_i
2.4661	0.1072	10	0.05 0.10 0.19 0.33 0.45 0.55 0.64 0.74 0.81	4 3 3 3 2 2 1 1 1
2.0208	0.1264	6	0.05 0.14 0.31 0.50 0.67 0.79	6 5 4 3 1 1
1.6058	0.1672	4	0.05 0.19 0.45 0.69	8 7 4 1
1.3460	0.2608	4	0.05 0.29 0.62 0.79	10 7 2 1

Tabla 3. Parámetros del código permutacional para distintas tasas de variación y para un vector de 20 componentes.

En la figura 2 se gráfica R vs. D/σ^2 para datos de las tablas 2 y 3, así como la función RDF y el comportamiento del óptimo cuantizador escalar [4].

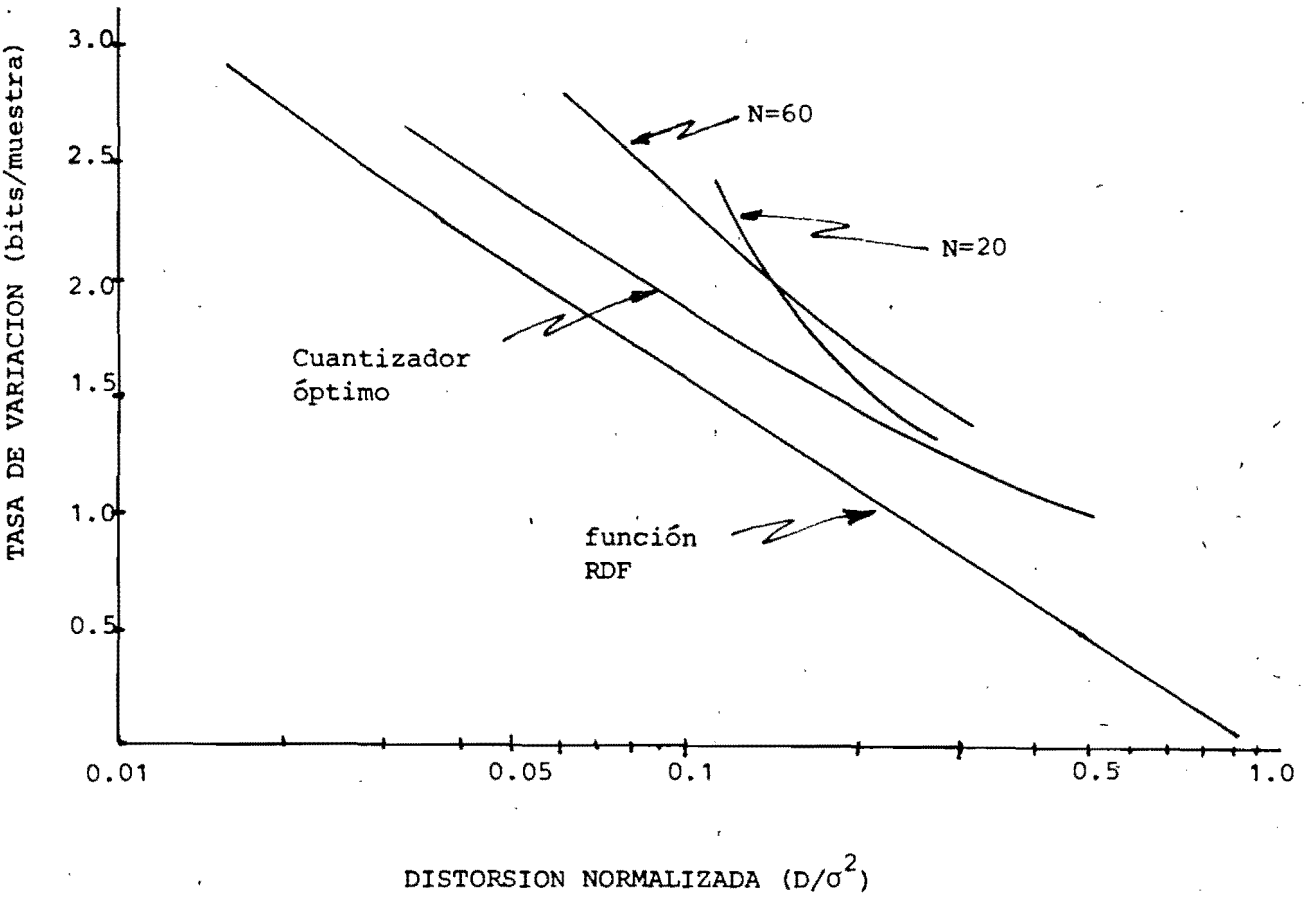

figura 2. Comportamiento del código permutacional para fuente uniforme de 60 y 20 muestras.

De la figura 2 se obtienen conclusiones semejantes a la fuente laplaciana del capítulo anterior.

A pesar de las diferentes longitudes tomadas para las codificaciones laplaciana y uniforme, el comportamiento de la codificación uniforme puede considerarse mejor para tasas de variación altas (mayores a 2 bits/muestra), pero tienden a ser semejantes para tasas de variación bajas.

Se codificaron por permutaciones algunos vectores de 60 componentes usando el vector norma para la tasa de 2.7612 bits/muestra, desgraciadamente, no los suficientes para comparar las distorsiones obtenidas en la simulación y en forma teórica.

Se generaron vectores aleatorios con distribución uniforme en (0,1); como un ejemplo se muestra la codificación del siguiente vector.

+++++ VECTOR NORMA +++++

0.97	0.95	0.93	0.92	0.89	0.86	0.83	0.80	0.80	0.75
0.75	0.70	0.70	0.70	0.62	0.62	0.62	0.62	0.52	0.52
0.52	0.52	0.52	0.52	0.35	0.35	0.35	0.35	0.35	0.35
0.35	0.35	0.35	0.14	0.14	0.14	0.14	0.14	0.14	0.14
0.14	0.14	0.14	0.14	0.14	0.14	0.02	0.02	0.02	0.02
0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02

+++++ VECTOR ALEATORIO +++++

0.53	0.35	0.32	0.78	0.83	0.91	0.01	0.91	0.33	0.80
0.84	0.81	0.37	0.89	0.02	0.07	0.29	0.04	0.74	0.11
0.98	0.93	0.71	0.89	1.00	0.95	0.69	0.63	0.55	0.66
0.02	0.17	0.84	0.57	0.82	0.77	0.25	0.74	0.91	0.78
0.49	0.94	0.22	0.85	0.10	0.92	0.62	0.44	0.06	0.42
0.96	0.03	0.47	0.61	0.39	0.86	0.71	0.49	0.52	0.75

+++++ VECTOR RESULTADO +++++

0.14	0.14	0.02	0.52	0.62	0.75	0.02	0.80	0.14	0.52
0.62	0.52	0.14	0.70	0.02	0.02	0.02	0.02	0.35	0.02
0.95	0.86	0.35	0.75	0.97	0.92	0.35	0.35	0.14	0.35
0.02	0.02	0.62	0.14	0.62	0.52	0.02	0.35	0.80	0.52
0.14	0.89	0.02	0.70	0.02	0.83	0.35	0.14	0.02	0.14
0.93	0.02	0.14	0.35	0.14	0.70	0.35	0.14	0.14	0.52

El algoritmo se encuentra en el apéndice C.

C O N C L U S I O N E S

1.- Los resultados obtenidos para la densidad laplaciana son semejantes a los obtenidos por Townes [4] para la misma longitud del vector. Considero que las diferencias existentes se deben a que para el algoritmo propuesto en [1] y usado tanto aquí como por Townes, use la aproximación de la ecuación 4 del apéndice A, lo que provoca también que los resultados de μ_i no sean simétricos.

2.- En la obtención del vector norma, tanto para densidad laplaciana como para densidad unfirme, aumenta el tiempo de respuesta del algoritmo al aumentar la longirud del vector, y para algunas tasas de variación hay problemas de convergencia, lo que sugiere la revisión del algoritmo propuesto en [1] para disminuir el tiempo de respuesta; en consecuencia, su uso en tiempo real sea más factible.

3.- El comportamiento de la codificación permutacional mejora el comportamiento de los mejores cuantizadores escalares para ciertas tasas de variación. Sin embargo, Townes [4] mostró que su comportamiento con señales reales (voz) es más pobre comparado con otros sistemas más simples, pero sigue siendo un campo aún no totalmente explorado.

4.- Resulta atractivo el probar la codificación permutacional con señales reales ahora en el dominio de la frecuencia y observar resultados. Así como su combinación con otras técnicas o bien la codificación por bloques, surgen como alternativas.

A P E N D I C E A

Para la variante I del código permutacional, la tasa de variación está dada por

$$R = \frac{m!}{n_1! \ldots n_k!} \qquad \ldots\ldots\ldots\ldots (1)$$

y la distorsión por

$$D = \frac{1}{n}\left[E\left(\sum_{j=1}^{n} x_j^2\right) - \sum_{i=1}^{k} n_i \mu_i^2\right] \qquad \ldots\ldots\ldots\ldots (2)$$

los valores de μ_i son tales que D es mínima para n n_i y k fijos, y están dados por

$$\mu_i = \frac{1}{n_i} \sum_{j=S_{i-1}+1}^{S_i} E(Y_{j:n}) \qquad \ldots\ldots\ldots\ldots (3)$$

donde $Y_{j:n}$ es la estadística j-ésima de orden.

El problema es obtener los parámetros n_i y k que minimizan a D para una tasa R* y longitud del vector dadas. El problema es de optimización multivariable. Para resolverlo en [1] se definió

$$p_i = n_i/n$$

y se aproximó

$$\hat{R} = -\sum_{i=1}^{k} p_i \log_2 p_i \qquad \ldots\ldots\ldots\ldots (4)$$

Ahora, se minimiza (2) con respecto a p_i, se obtiene que

$$\sum_{i=1}^{n} p_i = 1 \quad y \quad p_i = \frac{2^{-\beta\mu^2 i}}{\left(\sum_{j=1}^{k} 2^{-\beta\mu_j^2}\right)} \qquad \ldots\ldots\ldots\ldots (5)$$

donde β se escoge de manera que $\hat{R}$ sea igual a un R* dado.

El algoritmo descrito en [1] se presenta a continuación:

1) Entradas n, R^* y $E(Y_{j:n})$ $j = 1, \ldots, n$

2) Se establece k igual al menor valor entero cuya $\log_2$ exceda R^*.

3) Se establece $\beta = 1$ y se fijan los n_i aproximadamente iguales de forma tal que si k divide n: $n_i = n/k$ para toda i.

4) Se calculan $\mu_1, \mu_2, \ldots, \mu_k$ de (3).

5) Se obtienen p_i según (5) y se ajusta β hasta que $\hat{R} = R^*$. Aqui se recomienda el método de Newton-Raphson.

6) Se calculan los nuevos n_i como los enteros más cercanos a np_i tales que $\Sigma_{i=1}^{k} n_i = n$.

7) Si para alguna i, $n_i = 0$, ir al paso 11).

8) Si los nuevos n_i y los anteriores coinciden ir al paso 9). En caso contrario ir al paso 4).

9) Se guardan los valores de n_i, D y R (valor exacto).

10) Se reemplaza k por k + 2, se reduce el mayor n_i en 2, se sustituye n_i por n_{i+1} para $i = 1,2,\ldots, k-2$, y se definen $n_1 = n_k = 1$, ir al paso 4).

11) Se impriman $\{n_i\}$, R y D fijados en el paso 9). Si k es par ir al paso 13, en caso contrario continuar.

12) Se establece k igual al menor valor entero par cuyo $\log_2$ exceda R^* e ir al paso 3).

13) Fin.

Un diagrama de flujo de este algoritmo se puede observar en [1] ó [4]. A continuación se muestra la realización del algoritmo en FORTRAN.

```
CCCC            ESTE PROGRAMA GENERA LA SERIE PARA
CCCC            PARA UNA SECUENCIA DE ENTRENAMIENTO.
CCCC
CCCC            CALCULOS PRELIMINARES
CCC
CC
      REAL*8 P(100),SUMA,NSUMA,U(100)
      DIMENSION NN(100),NO(100),S(100),PRR(100),PRO(100),NR(100)
      ACCEPT 2,N,R,D
2     FORMAT(I3,2(F5.2))
      CALL ASSIGN(1,'MEDIA.DAT;1',11)
      DEFINE FILE 1(1,200,U,IVAR)
      CALL ASSIGN(2,'LPO:',4)
      K=2.0**R
      J=K
10    J=J-2
      IF(J.GT.0)GO TO 10
      IF(J.LT.0)GO TO 12
      K=K+1
      GO TO 14
12    K=K+2
```

```
      READ(1,1)(PRO(I),I=1,60)
      DO 7 I=1,60
      PRR(I)=PRO(I)
7     CONTINUE
      WRITE(2, 3)(PRR(I),I=1,N)
3     FORMAT(///,2X,'VALORES MEDIOS:',//,15(4(2X,F13.7),/))
14    WRITE(2, 80)K
80    FORMAT(///,2X,'VALOR DE K=',2X,I3)
      L=K-1
      DO 33 I=1,L
      NN(I)=N/K
33    CONTINUE
      NN(K)=N-(K-1)*NN(1)
      WRITE(2, 82)(NN(I),I=1,K)
82    FORMAT(///,2X,'VALORES DE NN(I) INICIALES:',15(/,30X,I3))
13    S(1)=0
      S(2)=NN(1)
      MM=K+1
      DO 34 I=3,MM
      S(I)=NN(I-1)+S(I-1)
34    CONTINUE
      WRITE(2, 83)(S(I),I=1,MM)
83    FORMAT(///,2X,'VALORES DE S(I) INICIALES',16(/,30X,F10.5))
      DO 35 I=1,K
      J=S(I)+1
      JJ=S(I+1)
      SUM=0
      DO 36 L=J,JJ
      SUM=PRR(L)+SUM
36    CONTINUE
      U(I)=SUM/NN(I)
35    CONTINUE
      WRITE(2, 84)(U(I),I=1,K)
84    FORMAT(///,2X,'VALORES DE U(I) INICIALES:',15(/,30X,F5.2))
      A=1
      B=1
18    SUMA=0
CCCC
CC
      DO 39 J=1,K
      SUMA=2.0**(-B*U(J)**2)+SUMA
39    CONTINUE
      DO 40 I=1,K
      IF(SUMA.GT.0)GO TO 61
      P(I)=0.0
      GO TO 40
61    P(I)=2.0**(-B*U(I)**2)/SUMA
      IF(P(I).GT.0)GO TO 40
      P(I)=0.0
40    CONTINUE
      SUMB=0
      DO 41 L=1,K
      IF(P(L).NE.0)GO TO 62
      GO TO 41
62    SUMB=-P(L)*ALOG(P(L))/ALOG(2.0)+SUMB
41    CONTINUE
      F=ABS(SUMB-R)
      FF=SUMB-R
      IF(F.LT.0)GO TO 20
      SUMAB=0
      SUMAA=0
      DO 42 I=1,K
      SUMAA=P(I)*U(I)**2+SUMAA
      SUMAB=P(I)*U(I)**4+SUMAB
42    CONTINUE
      SUMAC=SUMAB-SUMAA**2
      SUMAD=-B*SUMAC/ALOG(2.0)
      B=B-FF/SUMAD
      GO TO 18
20    WRITE(2, 85)(P(I),I=1,K)
85    FORMAT(///,2X,'VALORES DE P(I):',15(/,30X,F15.8))
      WRITE(2, 86)B
86    FORMAT(///,2X,'VALOR FINAL DE B=',//,30X,F15.8)
      CALL AJUSTE(K,P,D,N,NQ,NSUMA)
      WRITE(2, 87)NSUMA
87    FORMAT(///,2X,'VALOR DE LA SUMA=',2X,F8.4)
      WRITE(2, 88)(NQ(I),I=1,K)
88    FORMAT(////,10X,'VALORES AJUSTADOS DE NN(I)',15(/,30X,I8))
```

```
      DO 43 I=1,K
      IF(NQ(I).EQ.0)GO TO 2000
43    CONTINUE
      DO 44 I=1,K
      IF(NQ(I).EQ.NN(I))GO TO 44
      DO 45 J=1,K
      NN(J)=NQ(J)
45    CONTINUE
      GO TO 13
44    CONTINUE
CC
CC
CC              CALCULO DE DISTORSION Y TASA DE VARIACION
CC
      D1=2.D0*FLOAT(N)
      DO 46 I=1,K
      D1=D1-FLOAT(NN(I))*U(I)**2
46    CONTINUE
      D1=D1/(FLOAT(N)*2.D0)
      R1=0
      DO 47 I=1,N
      R1=R1+DLOG10(FLOAT(I))
47    CONTINUE
      DO 48 I=1,K
      DO 49 J=1,NN(I)
      R1=R1-DLOG10(FLOAT(J))
49    CONTINUE
48    CONTINUE
      R1=R1/(DLOG10(2.D0)*FLOAT(N))
CC                  PASO 10
CC
      K=K+2
      KKK=K-2
      DO 53 I=1,KKK
      NN(K-I)=NN(K-I-1)
53    CONTINUE
      KR=K-1
      DO 50 I=2,KR
      NR(I)=NN(I)
50    CONTINUE
      DO 51 I=3,KR
      IF(NR(2).GE.NR(I))GO TO 51
      TEMP=NR(1)
      NR(I)=NR(2)
      NR(2)=TEMP
51    CONTINUE
      DO 52 I=2,KR
      IF(NR(2).NE.NN(I))GO TO 52
      NN(I)=NN(I)-2
      I=KR
52    CONTINUE
      NN(1)=1
      NN(K)=1
      WRITE(2,93)(NN(I),I=1,K)
93    FORMAT(///,10X,'VALORES NUEVOS DE NN(I)',15(/20X,I8))
      WRITE(2, 92)
92    FORMAT(//,30X,'++++ AQUI ESTOY +++++')
      GO TO 13
2000  WRITE(2, 89)R1
89    FORMAT(///,2X,'TASA DE VARIACION=',F8.5)
      WRITE(2, 90)D1
90    FORMAT(///,2X,'DISTORSION=',F8.5)
      IF((K/2*2).EQ.K)GO TO 65
      K=2.0**R
      J=K
75    J=J-2
      IF(J.GT.0)GO TO 75
      IF(J.GE.0)GO TO 66
CC
      K=K+1
      GO TO 67
66    K=K+2
67    CONTINUE
      GO TO 14
65    CONTINUE
      CALL EXIT
      END
```

```
CC
      SUBROUTINE AJUSTE(K,P,A,M,NQ,SUM)
CC
CC                AJUSTE DE N(I) A N*P(I)
CC
      DIMENSION DD(100),DDD(100),NQ(100)
      REAL*8 P(100),SN(100),D(100),TEMP,A,SUM
      DO 30 I=1,K
      NQ(I)=IDINT(M*P(I))
30    CONTINUE
      SUM=0
      DO 32 I=1,K
      SUM=NQ(I)+SUM
32    CONTINUE
      DO 34 I=1,K
      SN(I)=M*P(I)
      D(I)=SN(I)-NQ(I)
      DD(I)=D(I)
34    CONTINUE
      MM=INT(ABS(M-SUM))
      KK=K-1
      DO 36 I=1,KK
      N=I+1
      DO 36 J=N,K
      TEMP=0.0
      IF(DD(I)-DD(J))14,36,36
14    TEMP=DD(I)
      DD(I)=DD(J)
      DD(J)=TEMP
36    CONTINUE
      IF(SUM.GE.M)GO TO 40
      L=MM
      DO 40 I=1,L
      DO 40 J=1,K
      SUM=0
      DO 44 L=1,K
      SUM=SUM+NQ(L)
44    CONTINUE
      IF((ABS(ABS(SN(J)-DD(I))-NQ(J)).GE.A).OR.(SUM.EQ.M))GO TO 40
      NQ(J)=NQ(J)+1
40    CONTINUE
      IF(SUM.LE.M)GO TO 48
      DO 46 I=1,K
      DDD(I)=DD(K-I+1)
46    CONTINUE
      DO 48 I=1,MM
      DO 48 J=1,K
      SUM=0
      DO 52 L=1,K
      SUM=SUM+NQ(L)
52    CONTINUE
      SOF=ABS(ABS(SN(J)-DDD(I))-NQ(J))
      IF((SOF.GE.A).OR.(NQ(J).EQ.(0.0)).OR.(SUM.EQ.M))GO TO 48
      NQ(J)=NQ(J)-1
48    CONTINUE
      RETURN
      END
```

En el programa anterior, distorsión está càlculada para media 0 y varian cia 2, para el cálculo condensidad uniforme la media es un medio y la varian- cia 1/12, por lo que hicieron los cambios pertinentes al programa.

A P E N D I C E B

Los valores esperados para una variable aleatoria $Y_{i:n}$ con densidad laplaciana de medio y variancia 2 pueden calcularse, de acuerdo a Govindarajulu [6] como

$$E(Y_{i:n}) = \frac{1}{2^n}\left[\sum_{k=0}^{i-1}\binom{n}{k}E(V_{i-k:n-k}) - \sum_{k=i}^{n}\binom{n}{k}E(V_{k-i+1:k})\right]$$

para un lado de la densidad. Además, $E(V_{i:n}) = \sum_{j=n-i+1}^{n} J^{-1}$, entonces

$$E(Y_{i:n}) = \frac{1}{2^n}\left[\sum_{k=0}^{i-1}\binom{n}{k}\left(\sum_{j=n-i+1}^{n=k} J^{-1}\right) - \sum_{k=i}^{n}\binom{n}{k}\left(\sum_{j=i}^{k} J^{-1}\right)\right]$$

que puede simplificarse haciendo $\binom{n}{k} = \binom{n}{n-k}$, $k \leq$ entero $(\frac{n}{2})$. La ecuación anterior puede entonces escribirse finalmente como:

$$E(Y_{i:n}) = -\left(\sum_{j=1}^{n-i} J^{-1}\right)\left(\sum_{k=0}^{i-1}\binom{n}{k}2^{-n}\right) - \sum_{k=i}^{n-i}\binom{n}{k}2^{-n}\left(\sum_{j=i}^{k} J^{-1}\right)$$

que fue la forma usada en el cálculo de los valores esperados, y sólo para $n/2$ valores, ya que por la simetría de la densidad: $E(Y_{i:n}) = -E(Y_{n-i+1:n})$.

Los valores de $\binom{n}{k}2^{-n}$ fueron calculados parcial y sucesivamente para prevenir problemas de saturación en la computadora.

El programa fue hecho en FORTRAN y se muestra a continuación.


```
CC
CC
CC                CALCULO DE LOS VALORES ESPERADOS
CC                DE UNA DISTRIBUCION LAPLACIANA
CC                CON MEDIA CERO Y VARIANCIA UNITARIA
CC
      REAL*8 SPP,SBS,S,SC,RR,R,PP,RP
      DIMENSION RRQ(100)
      CALL ASSIGN (1,'LAPLA.DAT;1',11)
      DEFINE FILE 1(1,200,U,IVAR)
      CALL ASSIGN(2,'LPO:',4)
      N=100
      NNN=N/2
      DO 40 M=1,NNN
      I=M
CC                PRIMER  SUMANDO
      NN=N-I
      RO=0
```

```
      DO 41 J=1,NN
      RA=J
      RO=1.0/RA+RO
41    CONTINUE
      II=I-1
      LM=20
      LLM=4
      LN=LM/LLM
      NL=N/LN
      RR=((10.0**LLM)/(2.0**NL))*LN
      IF(II.EQ.0)GO TO 16
      DO 42 K=1,II
      P=N
      Q=K
      PP=2.0**(P/Q)
      R=1.0
      DO 43 J=1,K
      QQ=J
      R=R*((P-Q+QQ)/(PP*QQ))
43    CONTINUE
      R=R*(10.0*LLM)
      RR=(RR/(10.0**LN)+R)/(10.0**LLM)
42    CONTINUE
16    RP=(RO*RR)
CC
CC              SEGUNDO SUMANDO
CC
      IM=N-I
      SC=0
      DO 44 K=I,IM
      SP=N
      SQ=K
      SPP=2.0**(SP/SQ)
      S=1
      DO 45 J=1,K
      SQQ=J
      S=S*((SP-SQ+SQQ)/(SPP*SQQ))
45    CONTINUE
      SB=0
      DO 46 L=I,K
      SS=L
      SB=1/SS+SB
46    CONTINUE
      SC=(S*SB)+SC
44    CONTINUE
      PRO(I)=-(RP+SC)
40    CONTINUE
      DO 47 I=1,50
      RRO(N-I+1)=-PRO(I)
47    CONTINUE
      I1=1
      WRITE(1'I1)(PRO(I),I=1,100)
      CALL EXIT
      END
```

A P E N D I C E C

Algoritmo para la generación de vectores aleatorios con distribución uniforme en (0,1) y su codificación permutacional por un vector norma contenido en el programa.

```
CC
CC
CC                EN ESTE PROGRAMA SE SUSTITUYE
CC                      - UN VECTOR ALEATORIO DE 60 COMPONENTES
C                         DE DISTRIBUCION UNIFORME (0,1)
CC                      - POR EL VECTOR NORMA Y(I) YA GENERADO
CC                        ANTERIORMENTE.
CC
CC
      DIMENSION X(60),Y(60),Z(60),XX(60),NN(60),T(300)
      CALL ASSIGN(1,'RANDOM.DAT;1',12)
      DEFINE FILE 1(1,600,U,IVAR)
      CALL ASSIGN(2,'LP0:',4)
      READ(1'1)(T(I),I=1,300)
CC
CC
      DO 3 I=47,60
            Y(I)=0.02
3     CONTINUE
      DO 4 I=34,46
            Y(I)=0.14
4     CONTINUE
      DO 5 I=25,33
            Y(I)=0.35
5     CONTINUE
      DO 6 I=19,24
            Y(I)=0.52
6     CONTINUE
      DO 7 I=15,18
            Y(I)=0.62
7     CONTINUE
      DO 8 I=12,14
            Y(I)=0.70
8     CONTINUE
      Y(11)=0.75
      Y(10)=Y(11)
      Y(9)=0.80
      Y(8)=Y(9)
      Y(7)=0.83
      Y(6)=0.86
      Y(5)=0.89
      Y(4)=0.92
      Y(3)=0.93
      Y(2)=0.95
      Y(1)=0.97
      WRITE(2,70)
70    FORMAT(///,20X,'+++++ VECTOR NORMA +++++')
      WRITE(2,71)(Y(I),I=1,60)
71    FORMAT(6(//,10(3X,F4.2)))
CC
CC
      DO 9 K=1,5
            DO 10 I=1,60
                  J=60*(K-1)+I
                  X(I)=T(J)
10          CONTINUE
            WRITE(2,72)
72          FORMAT(///,20X,'+++++ VECTOR ALEATORIO +++++')
            WRITE(2,73)(X(I),I=1,60)
73          FORMAT(6(//,10(3X,F4.2)))
```

```
CC
CC
CC                    REACOMODO DEL VECTOR ALEATORIO DADO
CC
                DO 12 I=1,60
                      Z(I)=X(I)
12              CONTINUE
                DO 14 J=1,59
                      N=J+1
                      DO 16 I=N,60
                            IF(Z(J)-Z(I))15,16,16
15                          TEMP=Z(J)
                            Z(J)=Z(I)
                            Z(I)=TEMP
16                    CONTINUE
14              CONTINUE
CC
CC
CC                    SUSTITUCION DEL VECTOR DADO POR
CC                    EL VECTOR NORMAL
CC
                DO 18 M=1,60
                      NN(M)=0
18              CONTINUE
                DO 20 L=1,60
                      DO 22 I=1,60
                            MM=1
                            DO 24 M=1,60
                                  IF(MM.EQ.0)GO TO 24
                                  IF(NN(M).EQ.I)GO TO 25
                                  MM=1
                                  GO TO 24
25                                MM=0
24                          CONTINUE
                            IF(MM.EQ.0)GO TO 22
                            IF(Z(L).EQ.X(I))GO TO 23
                            GO TO 22
23                          XX(I)=Y(L)
                            J=L
                            NN(J)=I
                            GO TO 20
22                    CONTINUE
20              CONTINUE
CC
CC
                WRITE(2,74)
74              FORMAT(///,20X,'+++++ VECTOR RESULTADO +++++')
                WRITE(2,75)(XX(I),I=1,60)
75              FORMAT(5(//,10(3X,F4.2)))
9       CONTINUE
        END
```

R E F E R E N C I A S

[1] Berger, T., Jelinek, F. y Wolf, J. K.
"Permutation Codes for Sources".
IEEE Transactions on Information Theory.
Vol. IT-18, No. 1, Enero de 1972, pp. 160 a 169.

[2] Berger, Toby.
"Optimum Quantizers and Permutation Codes".
IEEE Transactions on Information Theory.
Vol. IT-18, No. 6, Noviembre de 1972, pp. 759 a 765

[3] Paez, M. D. y Glisson, T.H.
"Minimum Mean-Squared-Error Quantization in Speech PCM and DPCM Systems".
IEEE Transactions on Communications
Vol. COM-20, No. 4, Abril de 19872. pp. 225 a 230

[4] Townes, Stephen Allen.
"Permutation Coding of Speech".
Tesis de Doctorado, Universidad del Estado de Carolina del Norte
Raleigh, 1981.

[5] Berger, Toby
"Minimum Entropy Quantizers and Permutations Codes".
IEEE Transactions on Information Theory.
Vol. IT-28 No.2, Marzo de 1982, pp. 149 a 157

[6] David, Herbert A.
"Order Statistics"
John Wiley & Sons, Inc.
2a. edición, 1981