Capítulo 5

SIMULACIÓN DE LA CUANTIZACIÓN ESCALAR Y VECTORIAL

Como se vio en cada uno de los capítulos anteriores, los principales métodos utilizados para la cuantización están basados en la creación de un *codebook*, en el cual se localizan todos los posibles valores correspondientes a la señal de entrada de un sistema. Este *codebook* se establece tanto en el transmisor como en el receptor, con la finalidad de comparar la muestra de entrada al transmisor con el diccionario, y así asignarle el valor correspondiente, el cual será enviado al receptor, en donde se especificará únicamente la ubicación de esa palabra de código. Esto permite que la distorsión que afecta a la señal sea mínima, ya que de inicio, el *codebook* se creará a partir de valores de distorsión, lo cual por si mismo disminuye la probabilidad de error en el receptor.

Debido a que estos temas fueron tratados con anterioridad en los demás capítulos a continuación se presentará la simulación realizada en *MATLAB* del proceso de cuantización por diferentes métodos y tomando diferentes consideraciones para su implementación. Es decir, se presentarán las simulaciones de los algoritmos LGB, COVQ basándonos en la generación de vectores aleatorios para la creación del *codebook*, y tomando como parámetro determinante la distorsión.

Además de presentar los algoritmos correspondientes tanto para la cuantización vectorial como la escalar, se presentarán los resultados obtenidos después de la generación de varios diccionarios variando las condiciones iniciales.

Finalmente se presenta un anexo en donde se incluyen si excepción los diccionarios generados para todos los casos, así como sus representaciones gráficas.

5.1 ALGORITMO PARA LA CUANTIZACIÓN ESCALAR

Algoritmo:

El proceso que se seguirá para realizar la simulación de la cuantización escalar estará basado en el esquema conocido como LGB, el cual a su vez, se basa en *splitting*, es decir, en ir dividiendo cada una de las regiones en dos partes. Este tema se explicó con mayor detenimiento en el capítulo 4. El algoritmo a seguir en la programación y generación del *codebook* es el siguiente:

1. Se debe generar una secuencia aleatoria de entrenamiento la cual será la representación de los datos que se desean enviar y con ella se generará el *codebook*.
2. Se ingresan valores de entrenamiento para poder realizar la iteración de Lloyd inicial.
3. Mediante la indicación *'lloyds*[1]*'* se calculan una frontera y dos centroides, con lo cual se tiene el primer diccionario de 2 elementos (los centroides calculados).
4. A continuación, se debe calcular la distorsión generada, la cual se conocerá mediante la siguiente expresión,

$$D = \frac{1}{N}\sum_{n=1}^{N}(x_n - Q(x_n))^2 \tag{5.1}$$

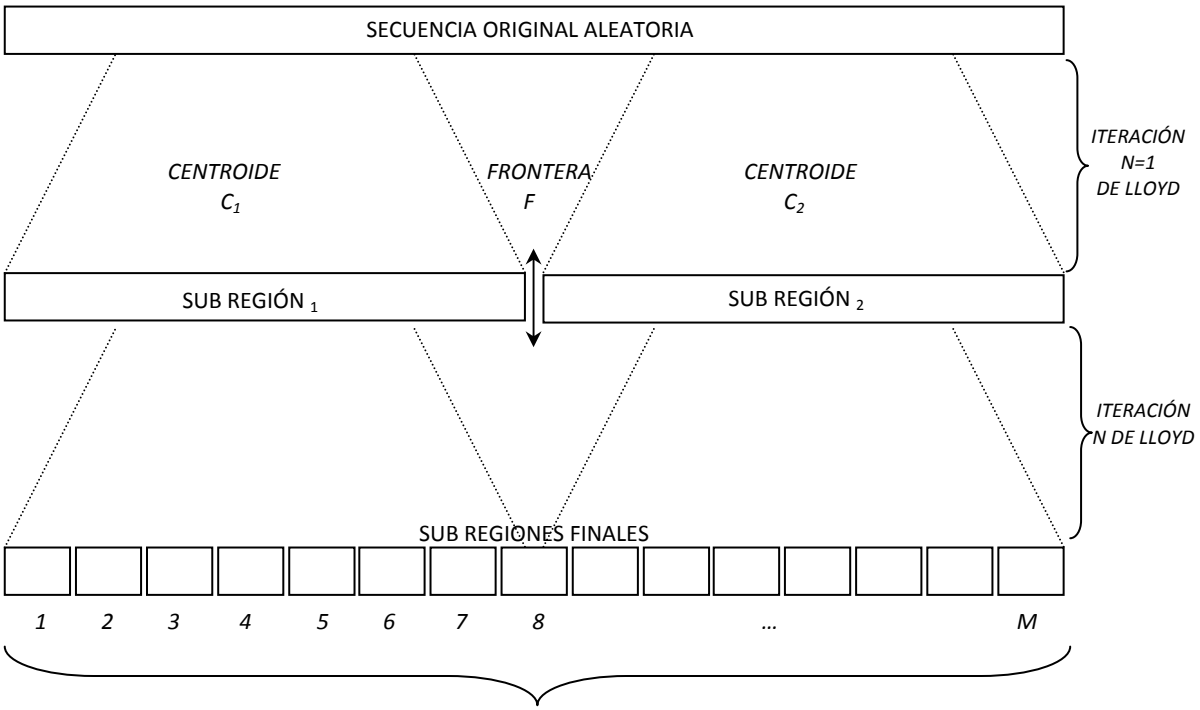
   para la cual es necesario tener la nueva secuencia cuantizada, y utilizando la instrucción '*quantiz*[2]*'* que genera una cuantización uniforme. Finalmente se calcula la distorsión.
5. A partir de este punto, se entra a un ciclo repetitivo, en el cual el primer paso debe ser calcular los límites mínimo y máximo de cada una de las regiones, para lo cual se utilizan las fronteras que se generaron anteriormente.
6. Después, para cada una de las regiones se realiza el mismo procedimiento aplicando la iteración de *'lloyds'* para generar por cada una de las regiones originales, dos nuevas regiones con dos centroides y una nueva frontera.

[1] *En el caso exclusivo de la programación decidí utilizar esta función solo por comodidad, ya que ésta realiza de manera automática el proceso de Iteración de Lloyd, la cual fue explicada en el capítulo correspondiente.*

[2] *La función quantiz se utilizó con la finalidad de comparar el valor de la distorsión generada por la función lloyds y la generada por una cuantización uniforme de la señal de entrada.*

7. Una vez obtenidas estas dos nuevas regiones con sus centroides, se debe calcular el valor de la distorsión, la cual se comparará con un valor patrón de distorsión $\boldsymbol{D_i}$ establecido al inicio del ciclo.
8. Si la distorsión $\boldsymbol{D}$ calculada en cada región es menor a $\boldsymbol{D_i}$, los valores de los centroides $\boldsymbol{c_i}$ deben guardarse en una nueva variable. Si la $\boldsymbol{D}$ calculada es mayor a $\boldsymbol{D_i}$ se debe volver a realizar el ciclo hasta que se tenga un valor de distorsión mínimo.
9. Finalmente, todos los centroides $\boldsymbol{c_i}$ guardados en la nueva variable, será el *codebook* buscado.

Ahora se presenta un diagrama que explica con más detalle el procedimiento que se espera realizar a partir de este algoritmo:



CADA UNO DE LOS CENTROIDES DE CADA REGIÓN FINAL FORMARÁ PARTE DEL CODEBOOK

FIG. 5.1 DIAGRAMA REPRESENTATIVO DEL ALGORITMO LGB

A continuación se presenta una parte del código utilizado para obtener los resultados solo para que se observe de forma detallada la aplicación del algoritmo antes mencionado.

Programa:

```
% CUANTIZADOR ESCALAR BASADO EN EL ALGORITMO LGB
%
% Secuencia de 10000 números aleatorios entre 0 y 100
```

```
% con una distribución uniforme:
%
a=100*rand(1,10000);
%
% Se realiza una función que tendrá como parámetro de
% entrada la secuencia aleatoria 'a' y como salida
% los valores correspondientes a dos nuevos sub-
% conjuntos y un valor de distorsión:
%
function [minimo,maximo,D]=div(a)
%
% Se establece un valor de distorsión inicial para
% realizar las comparaciones:
%
Di=10^-3;
co=[1,2];
minimo=[];
maximo=[];
%
% Se aplica la función 'lloyds' al conjunto original,
% generando para esta un valor de f (frontera), y dos
% de c (centroides):
%
[f,c,D]=lloyds(a,co);
%
% Se calcula la distorsión:
%
[inds,sq] = quantiz(minimo,f,c);
N=length(sq)
resta= minimo'-sq;
cuad=resta.^2;
D=mean(cuad)
%
% Se utiliza un ciclo 'for' para separar la secuencia
% original en dos nuevas regiones basándose en el
% valor de la frontera generada anteriormente:
%
for i=1:length(a)-1
    if a(i)<=f
        k=length(minimo);
        minimo(k+1)=a(i);
    else
        k=length(maximo);
        maximo(k+1)=a(i);
    end
end
minimo;
maximo;
%
% Se utiliza la instrucción 'if' para hacer que la
% función declarada se llame a sí misma y se comience
```

```
% así el ciclo repetitivo:
%
if D>Di
    div(minimo);
    div(maximo);
elseif D<=Di
% Se imprimen los valores de ‘c’ que cumplen con la
% condición:
    CODE=c;
end
% Se imprime el codebook final:
CODE
```

Inicialmente se generó una secuencia de 10 000 números aleatorios en el intervalo I[0,100], con la finalidad de obtener un *codebook* de gran calidad que permita asignar una valor a cada una de las muestras de entrada y con la menor distorsión posible, facilitando de esta forma su recepción y evitando la existencia de errores en el receptor.

```
% Secuencia aleatoria de 10000 números en el intervalo
% de 0 a 100:
x=100*rand(1,10000);
x1=rand(1,100);
subplot(2,1,1); plot(x)
subplot(2,1,2); plot(x1)
```

A continuación se muestran de forma gráfica las secuencias generadas, la primera en el intervalo entre [0,100], y la segunda entre [0,1]. Esto se realizó con la finalidad de observar de forma más detallada las muestras generadas.

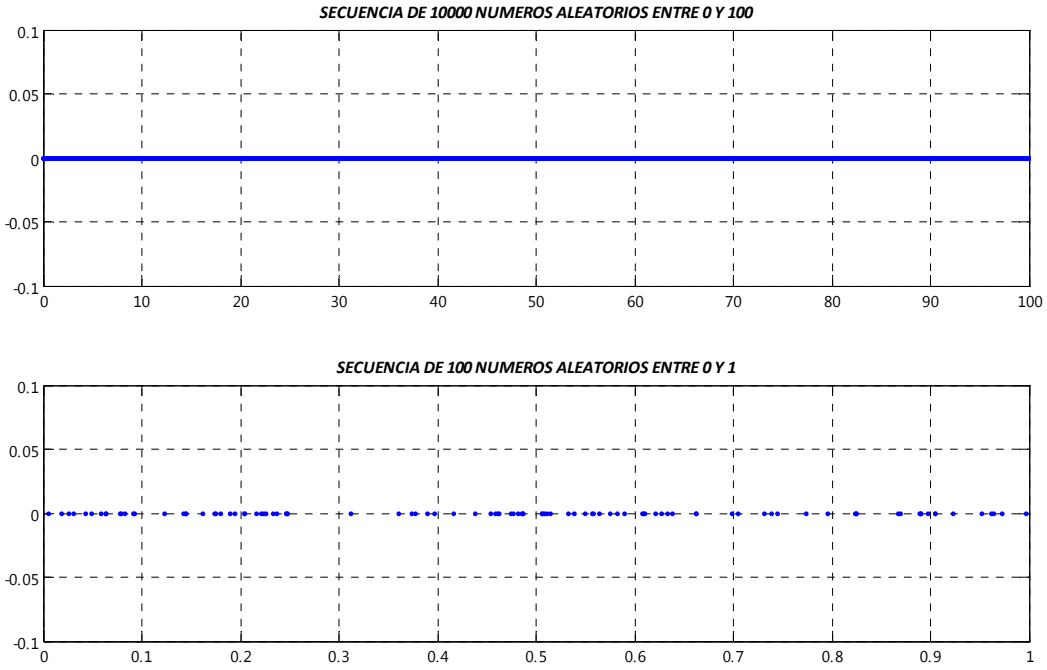


FIG. 5.2.A) SECUENCIA ALEATORIA DE 10000 NÚMEROS EN EL INTERVALO [0,100]

5.2.B) SECUENCIA ALEATORIA DE 100 NÚMEROS EN EL INTERVALO [0,1]

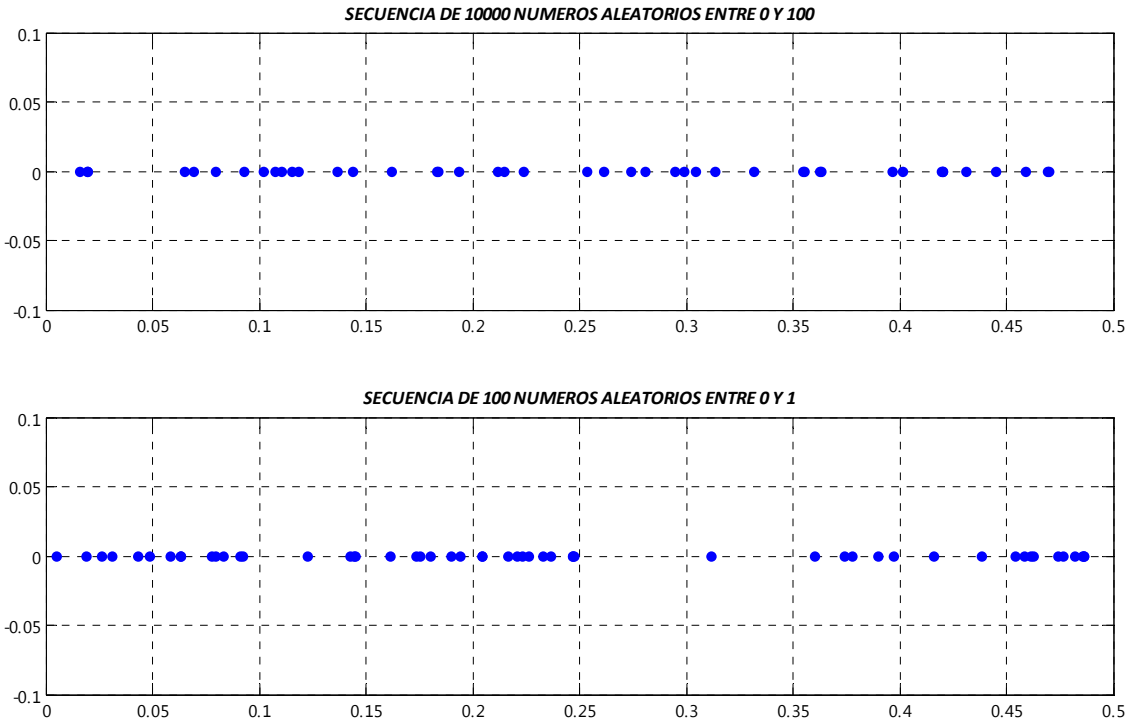


FIG. 5.3.A) SECUENCIA ALEATORIA DE 10000 NÚMEROS EN EL INTERVALO [0,100]

5.3.B) SECUENCIA ALEATORIA DE 100 NÚMEROS EN EL INTERVALO [0,1]

Después de haber aplicado el programa anterior para N=1, es decir para llevar a cabo únicamente la primera iteración de Lloyd, se pudieron obtener dos nuevos subconjuntos generados a partir de la secuencia aleatoria original, así como el valor que representa la frontera entre estas nuevas secuencia, y el valor medio (denominado centroide) de cada una de estas regiones.

```
% Valores para f y c iniciales:
f =
     49.7010
c =
     24.6902
     74.8592
% Gráfica de la secuencia con f y c:
x=100*rand(1,10000);
y=0*rand(2,10000);
x1=rand(1,100);
y1=0*rand(1,100);
     plot(x,y)
hold on
     plot(f,)
     plot(c,)
hold off
```

En la siguiente gráfica se muestran los valores de los centroides y de la frontera en comparación a la secuencia original:

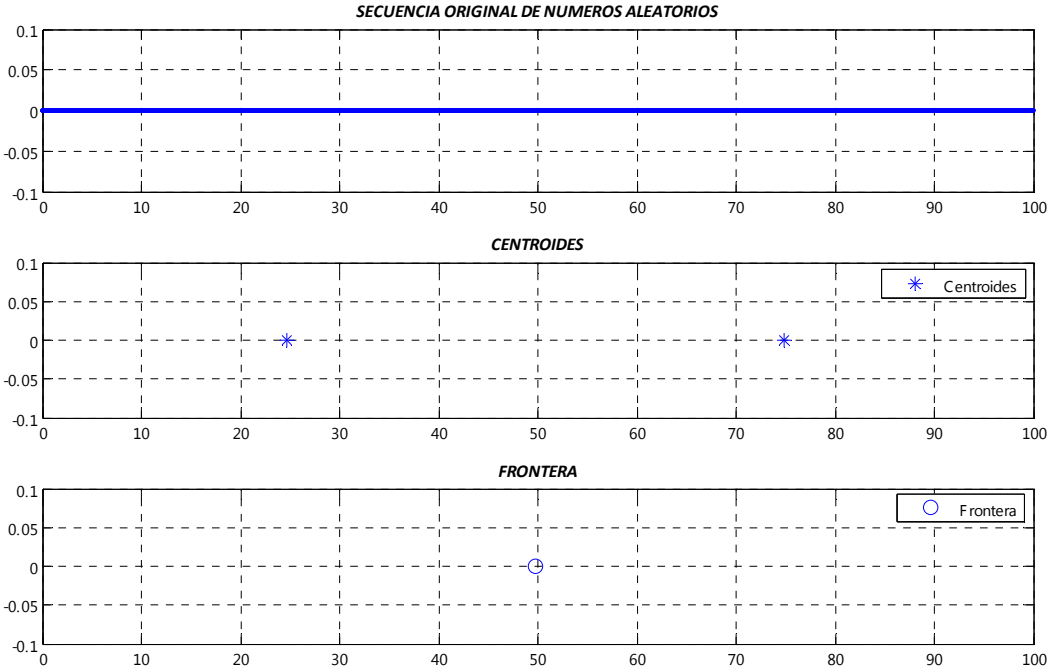


FIG. 5.4.A) SECUENCIA ORIGINAL, 5.4.B) CENTROIDES INICIALES, 5.4.C) FRONTERA INICIAL

Una vez aplicada la función para la generación del *codebook*, se obtuvo el resultado final para una distorsión de 10^{-3} que se muestra en la siguiente gráfica:

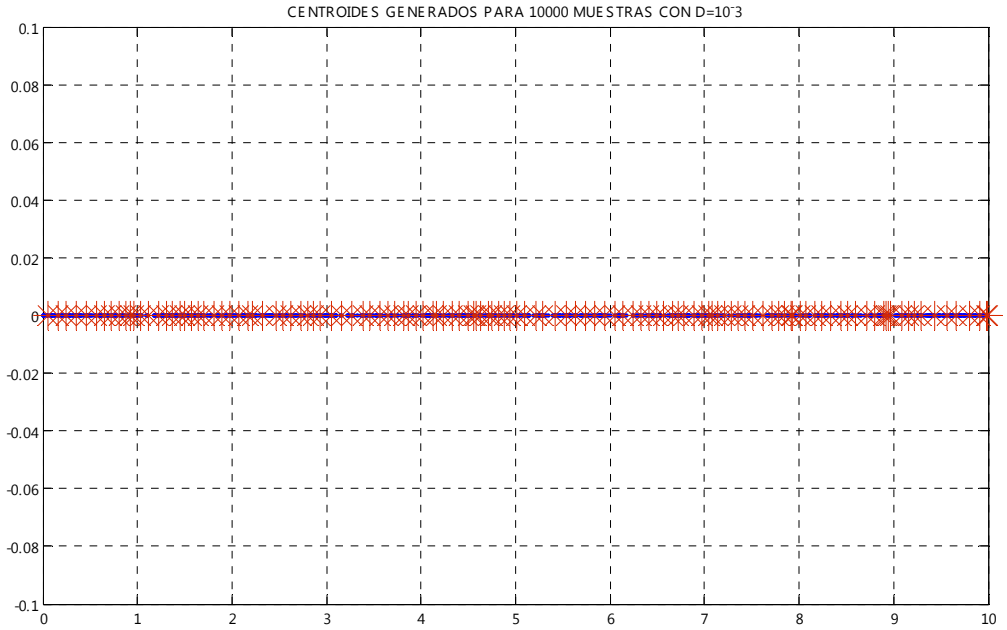


FIG. 5.5.1 REPRESENTACIÓN GRÁFICA DEL CODEBOOK FINAL

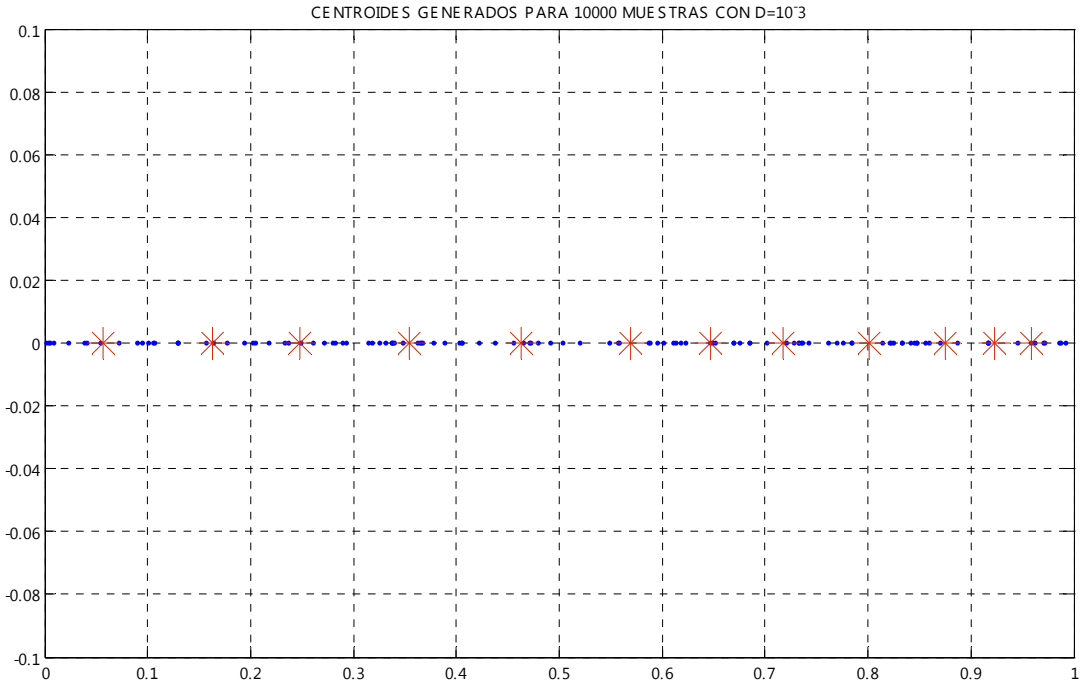


FIG. 5.5.2 CENTROIDES EN EL INTERVALO [0,1]

La tabla correspondiente al *codebook* generado a partir de las condiciones dadas se presenta al final en el Anexo 1 y 2, donde se podrá apreciar de forma detallada los 1 212 elementos que lo conforman, así como la representación gráfica en varios intervalos. Además, para fines comparativos también se presentan otras tablas generadas a partir de la modificación de las condiciones iniciales, como el valor de distorsión *D*, el número de elementos iniciales y el intervalo de trabajo.

Observaciones:

Como se puede observar con este diccionario, se consiguió reducir 10 000 números aleatorios a solo 1 212 elementos mediante la aplicación del programa utilizando y basándose en el algoritmo LGB y en el *splitting*.

Los resultados que se obtuvieron a partir de esta simulación corresponden a una distribución de probabilidad uniforme para los números aleatorios de entrada.

El código propuesto permite codificar cualquier palabra que ingrese al transmisor con gran eficacia, ya que esta creado a partir de una distorsión de 10^{-3}, es decir, casi una distorsión nula. Esto indica que cualquier palabra que entre será fácilmente asignada al centroide correspondiente de una manera sencilla y evitando cometer errores.

La generación de este diccionario fue un éxito, ya que permitió mostrar de forma analítica y gráfica los métodos propuestos al inicio del presente capítulo.

Con este diccionario de códigos, se reduce el valor de la distorsión a deseos del usuario, permitiendo con esto que los errores que puedan ocurrir en la señal se deban únicamente a las perturbaciones que se puedan dar en el medio de transmisión.

Es posible darse cuenta que la aplicación de este algoritmo es de suma facilidad, y esto depende a que este se aplica de forma única a una secuencia de números definidos conjuntamente en uno de los ejes, en este caso en el eje $\boldsymbol{x}$ (o simplemente recta numérica). Esto a su vez da la pauta para poder crear diccionarios muy variados, como los mostrados en los anexos 1 y 2 ya que con un valor mínimo de distorsión, la mayor longitud obtenida fue de 1 212 elementos, haciendo posible mostrarlo en su totalidad en el presente reporte.

Sin embargo, como se verá a continuación, para la cuantización vectorial es muy diferente, ya que al ocupar vectores que ahora estén alojados en un plano y que a cada uno de los elementos se les asignes coordenadas, nos daremos cuenta que el proceso se complica y el número de elementos finales que conforman el diccionario es del orden de $\boldsymbol{x^2}$ en comparación con la cuantización escalar,

lo que obviamente limitará las intenciones de realizar iteraciones en conjuntos con un gran rango, ya que esto generará un *codebook* por encima de los 10 000 elementos.

5.2 ALGORITMO PARA LA CUANTIZACIÓN VECTORIAL

Algoritmo:

Para el caso de la cuantización vectorial, me basaré en el algoritmo de díseño *COVQ (channel optimized vector quantization)*, la cual se explicó con anterioridad en otro capítulo. Lo que se busca al utilizar este medio de diseño, es poder crear a partir de una serie de vectores de dos dimensiones, un *codebook* que a cada una de las palabras de entrada le asigne automáticamente dos valores (de forma similar a unas coordenadas), es decir, que a cada palabra de entrada se le asigne un *'vector'* que lo represente.

Se espera que el *codebook* generado sea similar al de la cuantización escalar, ya que se utilizará nuevamente *splitting*, pero en este caso se realizará en un espacio de dos dimensiones. El algoritmo a seguir será el siguiente:

1. Se debe generar una serie de vectores aleatorios de dos dimensiones *(x,y)* de entrenamiento como base del codebook que se espera.
2. Mediante la indicación *'lloyds'* se calculan dos centroides en el plano dividiéndolo en dos nuevas regiones.

3. A continuación, se debe calcular la distorsión generada para cada uno de los puntos de la secuencia inicial, la cual se conocerá mediante la siguiente expresión,

$$D = \frac{1}{N}\sum_{n=1}^{N}(x_n - Q(x_n))^2 \tag{5.2}$$

4. Para cada una de las regiones generadas se deberá aplicar el mismo procedimiento para ir obteniendo en forma repetitiva los centroides $C_{x,y}$. La asignación de los valores a cada una de las regiones dependerá de la distancia que separe la muestra del centroide.
5. Se repite el procedimiento hasta que no se cumpla la condición (que dependerá del valor de la distorsión obtenida en cada iteración).
6. Finalmente, todos los centroides $\boldsymbol{c_{x,y}}$ serán el *codebook* buscado.

A continuación se presenta un diagrama que explica con más detalle el procedimiento que se espera realizar a partir de este algoritmo:

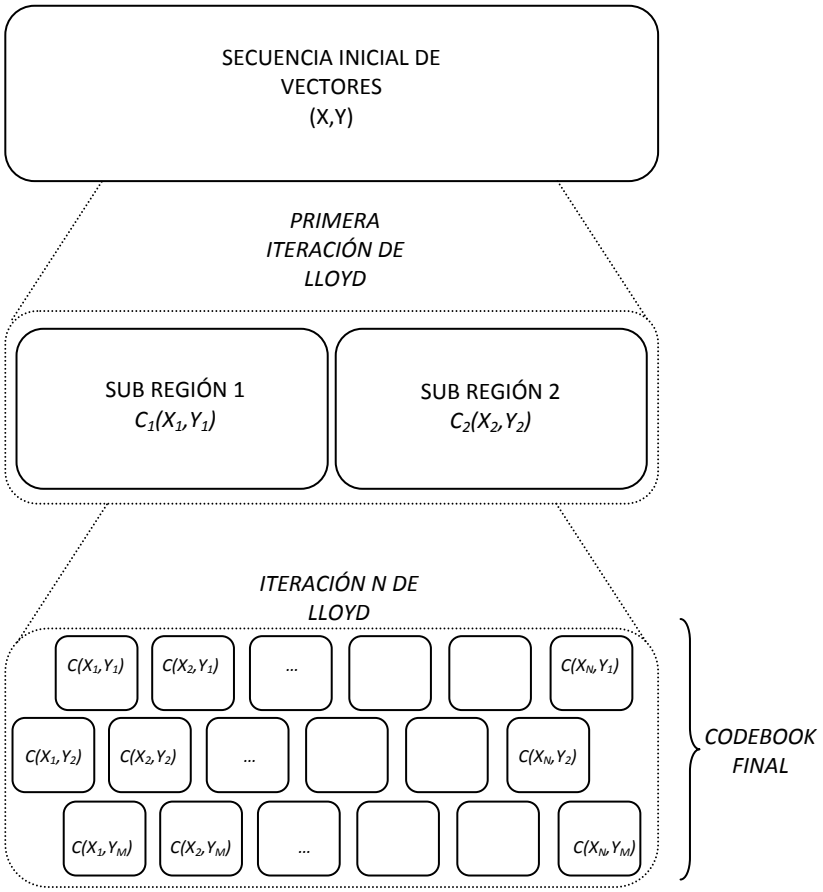


FIG. 5.6 REPRESENTACIÓN DEL ALGORITMO COVQ

Y aplicando el siguiente código, mediante el cual se imprimirán parejas ordenas correspondientes a los centroides $\boldsymbol{c}_{xy}$, se logrará generar un *codebook* que represente el algoritmo COVQ:

Programa:

```
% CUANTIZADOR VECTORIAL BASADO EN EL ALGORITMO COVQ
%
% Secuencia de 10000 vectores aleatorios entre 0 y 5
% con una distribución uniforme:
%
a=5*rand(2,10000);
%
function [minimo,maximo,D]=div(a)
%
Di=10^-3;
co=[1,2];
minimo=[];
maximo=[];
%
[f,c,D]=lloyds(a,co);
%
%
for i=1:length(a)-1
    if a(i)<=f
        k=length(minimo);
        minimo(k+1)=a(i);
    else
        k=length(maximo);
        maximo(k+1)=a(i);
    end
end
minimo;
maximo;
%
%
if D>Di
    div(minimo);
    div(maximo);
elseif D<=Di
    CODE=c;
end
% Se imprime el codebook final:
CODE
```

En las siguientes gráficas se muestra la generación de vectores de 2 dimensiones en los planos *X* y *Y*. En la figura 5.7 se muestran los 10 000 vectores aleatorios en el rango de 0 a1 para cada uno de los ejes, mientras que en la figura 5.8 se tiene un acercamiento de la misma secuencia para observar más detalladamente la localización de los puntos:

```
a=rand(500,1);
b=rand(500,1);
co=[1 2];
plot(a,b)
hold on
[f1,c1]=lloyds(a,co);
[f2,c2]=lloyds(b,co);
     plot(f1,f2)
     plot(c1,c2)
hold off
```

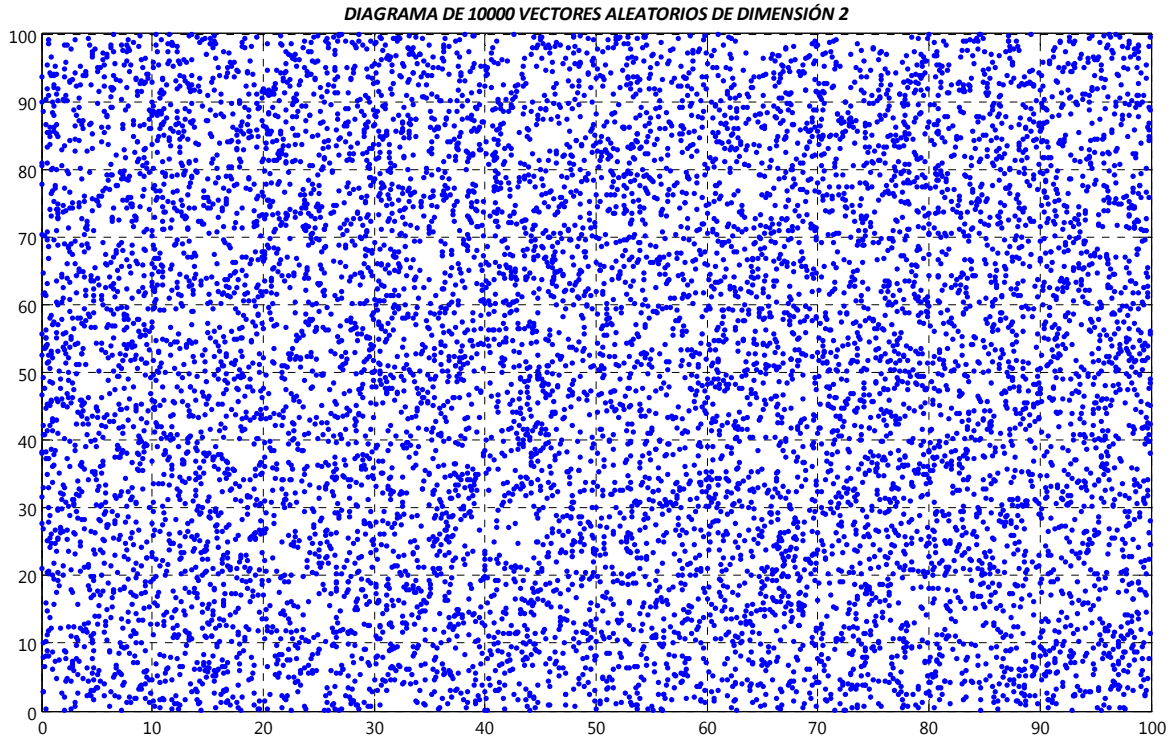


FIG. 5.7 VECTORES ALEATORIOS EN DIMENSIÓN 2

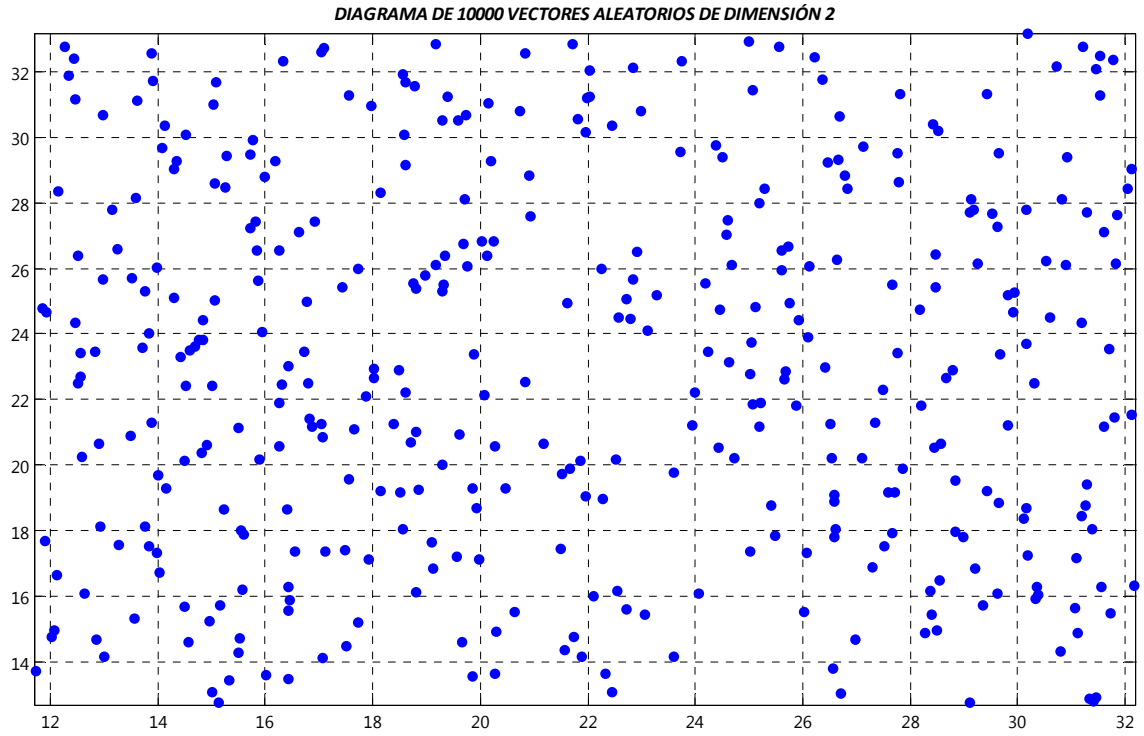


FIG. 5.8 VECTORES ALEATORIOS EN DIMENSIÓN 2

Inicialmente, al contar con los vectores en los ejes X y Y, se optó por conocer primeramente la localización del centroide inicial $\boldsymbol{C_i}$, con la finalidad de obtener el promedio general de toda la secuencia. Esto se muestra en la siguiente gráfica:

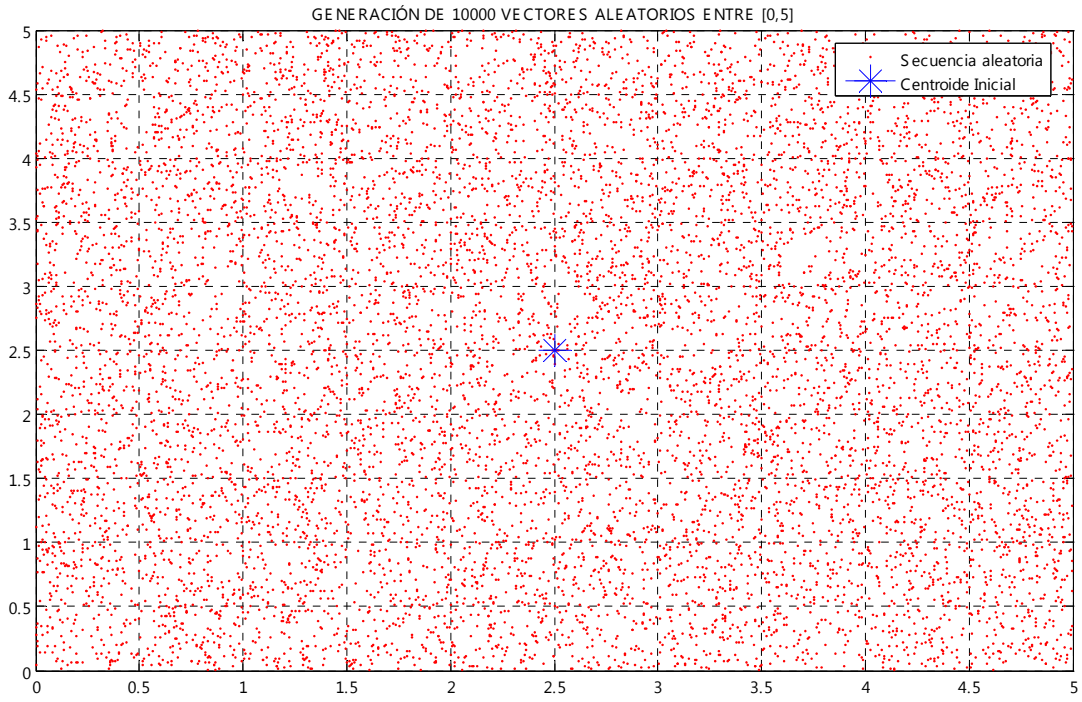


FIG. 5.9 VECTORES ALEATORIOS EN DIMENSIÓN 2 CON EL CENTROIDE INICIAL

Como se indica en el código utilizado, el *codebook* que se generará para describir el algoritmo COVQ tendrá las siguientes condiciones iniciales:

1. Se utilizarán 10 000 vectores iniciales con una distribución uniforme, esto con la finalidad de que los centroides obtenidos tengan mayor fidelidad al poder representar a un mayor cantidad de puntos en el plano.
2. La distorsión que me manejará será la misma que para el algoritmo LGB mostrado anteriormente, es decir, un valor de 10^{-3}, esto para crear una lista de datos con la menor probabilidad de error al momento de asignarle valores a la señal de entrada.
3. Para el caso de la cuantización escalar, se tiene que el número de elementos que forman parte del *codebook* es prácticamente el cuadrado del obtenido para la escalar, es decir, que si en este caso se manejará el mismo intervalo utilizado en el LGB en donde se obtuvieron 1 200 elementos, el *codebook* generado para este caso rebasaría los el millón de elementos, lo cual haría imposible plasmar dichos valores en el presente trabajo. Por tales razones se optó por manejar intervalos pequeños que a su vez fueran representativos y mostraran el resultado en una forma *'manejable'* para el lector. Los intervalos que se usarán variarán entre [0,5] y [0,1].

En la siguiente gráfica se puede observar el resultado después de la primera iteración de Lloyd, la cual generó los dos primero centroides dividiendo la región inicial en dos nuevos subconjuntos:

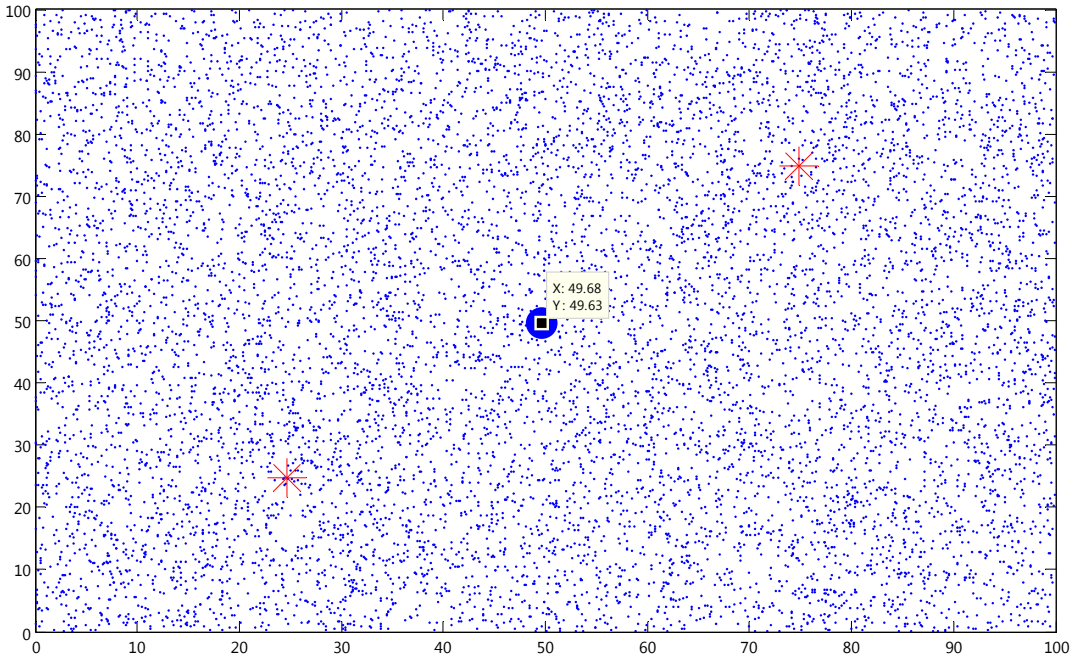


FIG. 5.10 RESULTADO DE LA PRIMER ITERACIÓN DE LLOYD

Como se puede apreciar, a partir del centroide inicial, se crean dos nuevas regiones, las cuales a su vez cuentan con su propio centroide, lo que indica que en ese momento existen dos regiones en el plano, y cada una contiene los elementos más cercanos. En este método es posible apreciar la aplicación del teorema del *'vecino más cercano'*, ya que cada pareja ordenada se agregará al conjunto cuya distancia sea mínima en comparación en el otro. Este método iterativo se repetirá nuevamente, generando regiones con la siguiente periodicidad N=1, 2, 4, 8, 16…

Después de la aplicación del programa creado, se generó de manera gráfica el *codebook* final, mostrado a continuación:

FIG. 5.11 REPRESENTACIÓN GRÁFICA DEL ALGORITMO COVQ

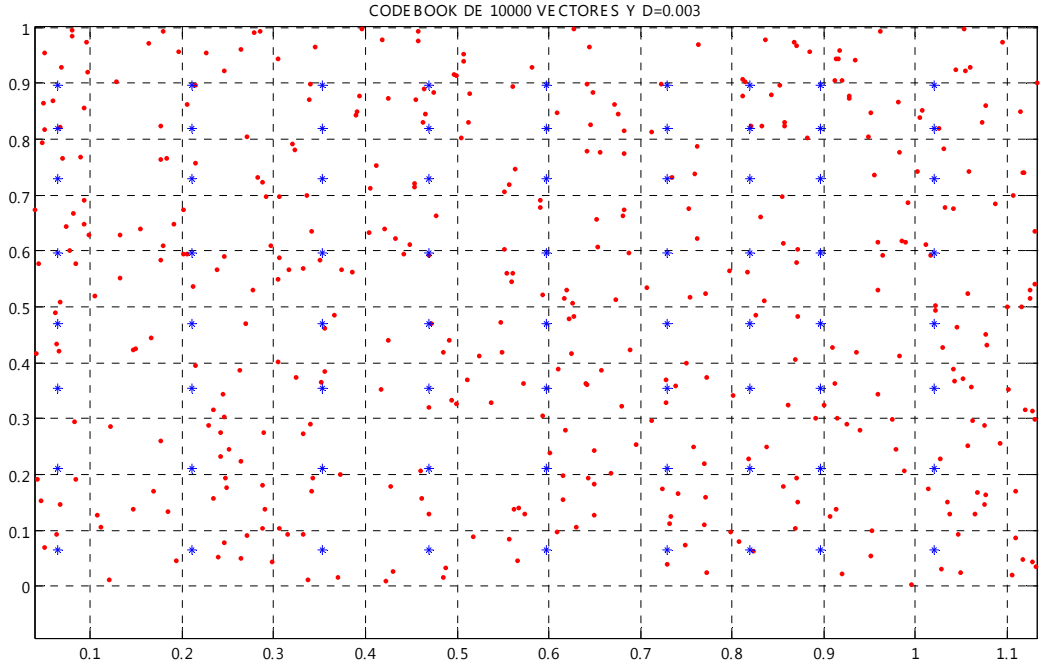


FIG. 5.12 ELEMENTOS GENERADOS EN EL INTERVALO [0,1]

Observaciones:

Como se puede observar en ambas gráficas, se generan una gran cantidad de centroides que pueden representar a los 10 000 vectores de entrada. En la figura 5.11 se puede observar el diccionario completo, el cual resulto estar formado de 1 156 parejas ordenas $\boldsymbol{C_{xy}}$, es decir, para cada muestra de entrada al transmisor, se le asignará una pareja *(x,y)* que la representará al momento de enviarla al receptor.

Además en esta misma gráfica se puede ver como los centroides forman una especie de *'reja'* la cual no es totalmente simétrica, esto debido a que se generó a partir de vectores cuya posición fue aleatoria, justo como se observa en la figura 5.12 donde se tiene un acercamiento al intervalo de [0,1]. En ella nos podemos percatar de que los centroides (puntos azules) se generaron únicamente en las zonas de mayor concentración de muestras iniciales (puntos rojos).

Cabe mencionar que estas gráficas corresponden al caso más óptimo obtenido, ya que con los más de mil elementos (en parejas ordenadas) y en un intervalo de [0,5] con D=0.003 es posible representar cualquier muestra de la señal de entrada.

Para observarlos resultados de una manera más detallada, la tabla correspondiente a estas gráficas se localizan en el Anexo 5, además de otras tablas para fines comparativas obtenidas a partir de la variación del intervalo de trabajo y el valor nominal de *D*. Finalmente, en el anexo 6 se muestran las representaciones gráficas correspondientes a las tablas mencionadas.

5.3 ESQUEMA COMPARATIVO

5.3.1 CUANTIZACIÓN ESCALAR

De acuerdo a los resultados obtenidos en la simulación escalar basada en el algoritmo LGB se pueden llevar a cabo una serie de comparación mediante la cual será posible determinar cual de todos los *codebooks* calculados resultará ser el código más óptimo que permita el intercambio adecuado de información entre un transmisor y un receptor. A continuación se llevará a cabo una comparación basándose en los siguientes aspectos:

1. Distorsión,

Como se mencionó en los capítulos anteriores, la distorsión es una medida que indica la diferencia que existe entre una señal que ha sufrido algún tratado (ya sea que se hayan modificado algunas de sus características por necesidad o por que hayas sido creadas por alteraciones en el medio), y la señal original.

En el caso de las simulaciones realizadas para el algoritmo LGB correspondiente a la cuantización escalar, fue posible determinar una serie de resultados a partir de la variación de este parámetro de forma manual, lo que permitió crear un *codebook* único para cada valor de distorsión fijado.

2. Numero de elementos finales de codebook,

Otro punto de comparación de los resultados obtenidos radica en el número de elementos finales (o códigos de palabra) que forman el *codebook* o diccionario. Este aspecto es de suma importancia, ya que además de tener en cuenta que este parámetro es dependiente del valor de distorsión asignado, en una aplicación real debe tenerse en cuenta la capacidad de memoria y la velocidad con que se cuente en el sistema, ya que al tenerse un diccionario muy grande, puede que este supere la capacidad de memoria que esta reservada para esta tarea, tanto en el transmisor como en el receptor, ya que en ambos dispositivos, la tarea va más allá de solo almacenar el diccionario.

En ambos elementos, el sistema debe tener la capacidad de almacenar el *codebook*, pero al mismo tiempo, debe contar con una gran velocidad para poder asignarle a cada una de las muestras de entrada un valor fijo, y enviarlo. Lo mismo sucede en la recepción. Por este motivo, el número de elementos finales debe estar acoplado a las características iniciales del sistema. En este caso, solo se hará la comparación para fines prácticos, ya que los diccionarios propuestos están diseñados con la finalidad de mostrar la variabilidad de resultados al modificar las condiciones iniciales de un sistema cualquiera, es decir, no están especificadas las condiciones de un sistema único.

3. Numero de iteraciones que lograron ese codebook,

Este también resulta ser un parámetro importante, y que al igual que el anterior, depende en gran parte de la distorsión. Su importancia radica en que también depende de la capacidad del sistema, ya que el sistema puede no soportar el programa, y por lo tanto no permitir que el resulta final se conozca.

Como se mencionó, ninguno de estos parámetros son independientes, sino que son dependientes uno con otro, siendo aquel código que posea las condiciones más óptimas en todos los casos el diccionario mejor desarrollado.

En la siguiente tabla se muestran de una manera concentrada los resultados obtenidos después de las simulaciones:

Distorsión	***Numero de elementos finales***	***Numero de elementos iniciales***	***Intervalo [0,]***
10^{-3}	1212	10000	100
10^{-3}	136	1000	10
10^{-3}	16	10000	1
10^{-3}	16	1000	1
10^{-3}	14	100	1
0.5	64	10000	100
0.5	6	1000	10
0.5	2	10000	1
0.5	2	1000	1
0.5	2	100	1
1	32	10000	100
1	4	1000	10
1	2	10000	1
1	2	1000	1
1	2	100	1
10	16	10000	100
10	2	1000	10
50	8	10000	100

TABLA 5.1 RESULTADOS FINALES DEL ALGORITMO LGB

A continuación se muestra de forma gráfica la relación entre la distorsión y el número de elementos que formarán el *codebook* buscado:

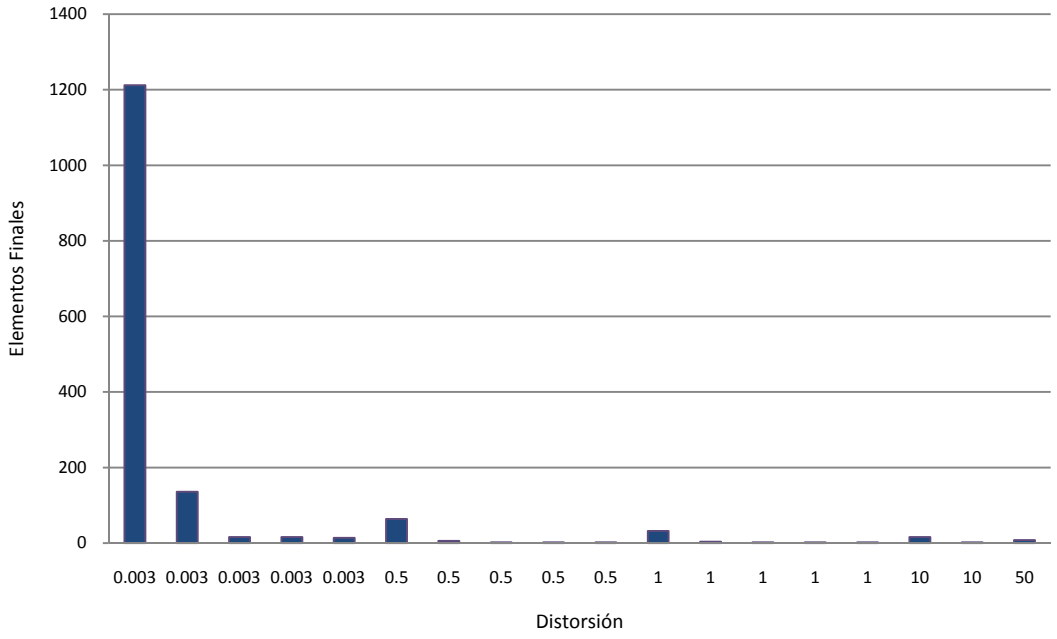


FIG. 5.13 RELACIÓN DISTORSIÓN VS ELEMENTOS FINALES

Sin embargo, para poder apreciar de una forma más detallada, se verán las gráficas por separada a partir del valor de distorsión. Se comenzará tomando como patrón D=0.003:

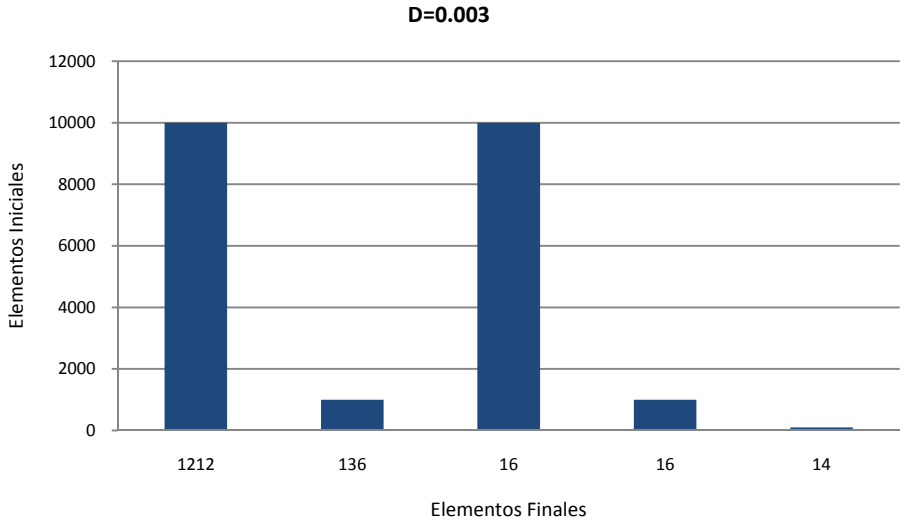


FIG. 5.14 RELACIÓN ELEMENTOS FINALES VS INICIALES PARA D=0.003

Como podemos observar, para una distorsión de 0.003, el número de elementos que forman el *codebook* es muy grande, sobre todo cuando el rango en que se generan los números aleatorios aumenta.

Esto se debe a que mientras en valor de distorsión disminuye, el rango de *'efectividad'* de cada una de las muestras disminuye, provocando a la vez que su número aumente. Es decir, el aumento de la distorsión generará un diccionario de una longitud menor y viceversa.

En la gráfica se observa que el *codebook* que presenta mayor cantidad de elementos es aquel que se propuso realizar en un intervalo mayor, de [0,100], esto con la finalidad de darles a las muestras de entrada la capacidad de encontrar un punto que las represente con mayor eficacia, y al mismo tiempo que disminuya su probabilidad de error.

Sin embargo, como se mencionó al inicio de esta comparación de resultados, para la aplicación de este código se debe contar con un sistema que cuente con una memoria que soporte la cantidad de datos y que a su vez tenga una velocidad considerable parta localizar el punto que represente a la señal de entrada.

A continuación se presenta la gráfica que corresponde a una distorsión D=0.5:

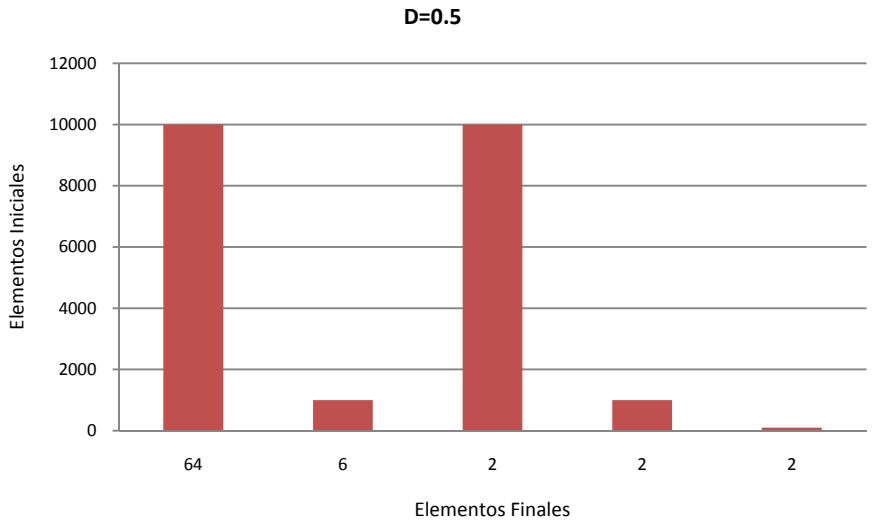


FIG. 5.15 RELACIÓN ELEMENTOS FINALES VS INICIALES PARA D=0.5

Como se puede observa, el número de elementos del diccionario disminuye de forma considerable en comparación con la anterior, esto se debe a que este valor de *D* cuenta con un rango mayor para representar las muestra, pero al mismo tiempo disminuye la calidad con que lo hace, es decir, aumenta la probabilidad de error. Esto se repite para la siguiente gráfica en donde se muestran los resultados para D=1:

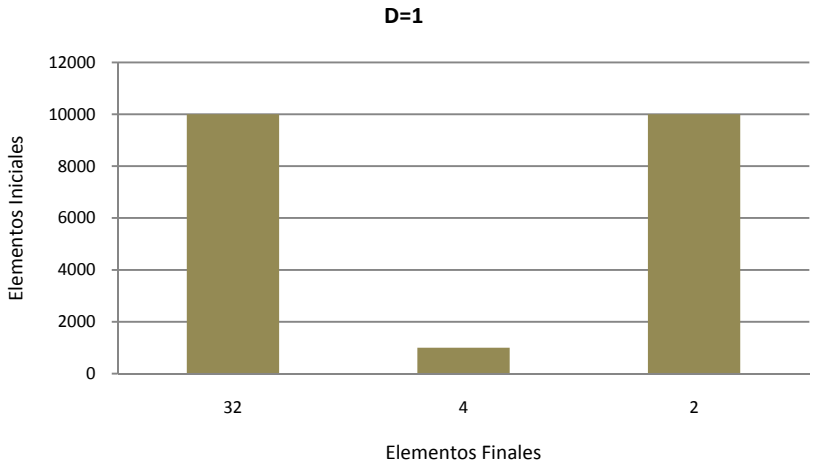


FIG. 5.16 RELACIÓN ELEMENTOS FINALES VS INICIALES PARA D=1

De esta comparación se puede concluir que mientras menor sea la distorsión, el número de elementos generados aumenta, por lo que la probabilidad de error al tratar de recuperar la señal original en el receptor es menor, ya que los errores ahora solo dependen de las perturbaciones que se generen en el medio de transmisión. Por lo que es de esperarse que para los casos de D=10 y D=50 el número de elementos será demasiado pequeño (dividirá la secuencia en 2 o 4 muestras) teniendo con esto que la señal de entrada quede, justo en el momento de la cuantización, con una forma muy diferente a la original, lo que dificulte su correcta recuperación.

A continuación se presentan los resultados pero desde el punto de vista del intervalo inicial en donde se generaron las muestras. Esto con la finalidad de determinar cual es el rango más óptimo en el que la señal de entrada puede cuantizarse de forma correcta:

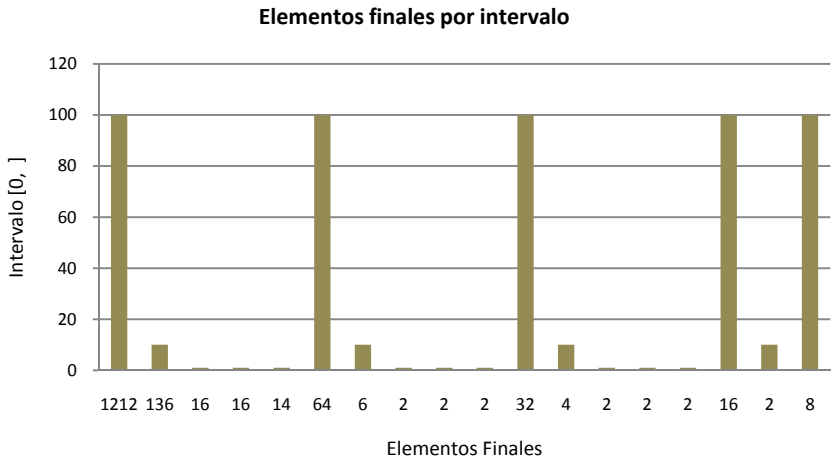


FIG. 5.17 ELEMENTOS FINALES POR INTERVALO

El primer intervalo a examinar comprende números generados entre 0 y 100. Como se puede observar en la siguiente gráfica, y tomando en consideración que el intervalo es muy amplio para poder tomar muestras que representen a la señal de entrada, es posible percatarse que la mayor concentración de estos se da cuando la distorsión es muy pequeña, es decir, que con un amplio rango y una distorsión casi nula, se generan 1212 elementos a partir de 10 000 elementos para cuantizar las señales (esto se observa en la primer barra de la gráfica).

Sin embargo, si el valor de la distorsión se varía, en este caso, si se aumenta, el número de elementos que conformen el diccionario tiende a disminuir de manera drástica, por lo que para este intervalo en concreto, conviene utilizar el *codebook* que se propone al inicio de este capítulo, ya que cualquiera de los demás en este intervalo podría provocar mayor cantidad de errores.

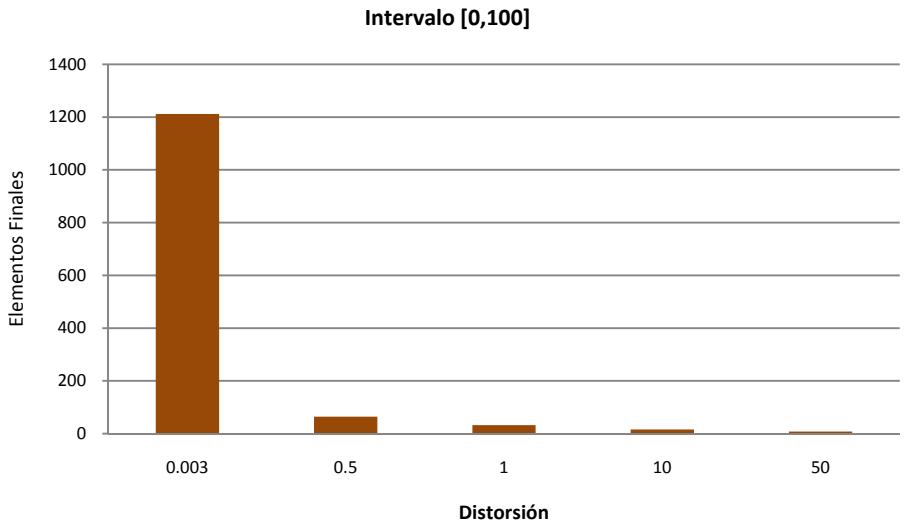


FIG. 5.18 RELACIÓN DISTORSIÓN VS ELEMENTOS FINALES PARA [0,100]

En la siguiente gráfica se muestran los resultados correspondientes para el intervalo de números entres 0 y 10 en donde es posible darse cuenta que ocurre los mismo que en caso anterior, la mayor concentración de elementos se localiza cuando la distorsión es mínima, y estos van disminuyendo conforme la distorsión es mayor.

Además, como se mencionó, la diferencia entre los dos valores mínimos de *D* generan un diccionario con una cantidad de elementos que varia de una forma drástica, por lo que es recomendable utilizar el diccionario en con la menos distorsión.

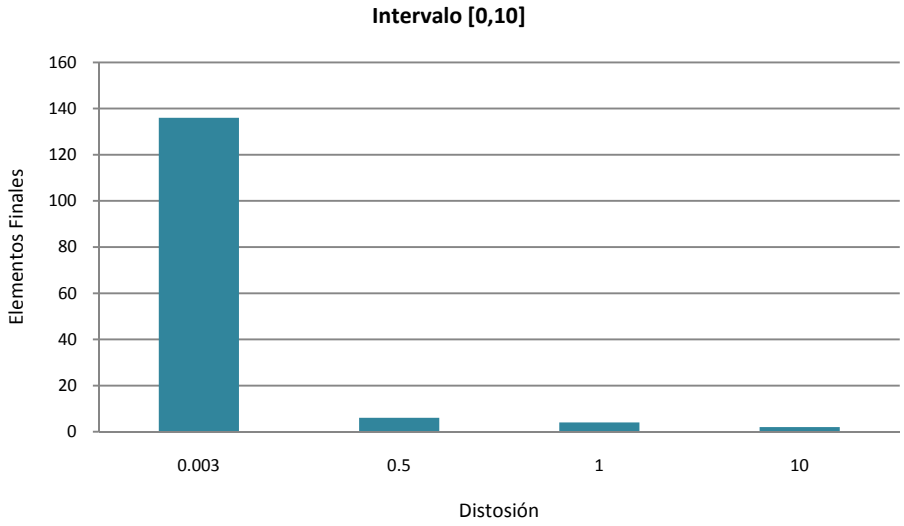


FIG. 5.19 RELACIÓN DISTORSIÓN VS ELEMENTOS FINALES PARA [0,10]

Finalmente, sucede lo mismo para el intervalo de 0 y 1. En este caso el número de elementos para todos los valores de distorsión resultó ser muy pequeña en comparación a todos los demás resultados:

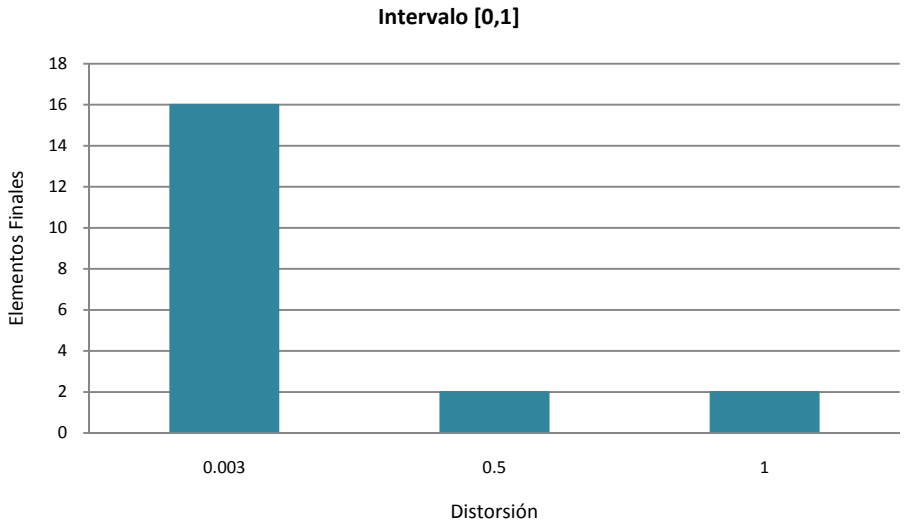


FIG. 5.20 RELACIÓN DISTORSIÓN VS ELEMENTOS FINALES PARA [0,1]

De esta comparación puedo concluir que el *codebook* que cumple con los requerimientos necesarios para llevar a cabo una cuantización escalar más óptima fue el propuesto al inicio, es

decir, el diccionario formado por 1 212 generado a partir de una distorsión D=0.003 y en un intervalo de [0,100].

Esto debido a que presenta un cantidad impresionante de elementos que permitirán a cualquier valor de entrada al sistema encontrar un valor que lo represente y que prácticamente mantenga el 100% de fidelidad, evitando que aparezcan errores en el transmisor, y por ende en el receptor, y los errores que surjan se deberán únicamente al medio por el que se envía la señal.

Distorsión	***Numero de elementos finales***	***Numero de elementos iniciales***	***Intervalo [0,]***
10^{-3}	1212	10000	100
10^{-3}	136	1000	10

TABLA 5.2 CARACTERÍSTICAS DE LOS DICCIONARIOS SELECCIONADOS

Otro código que puede resultar útil, es el generado a partir de 1 000 números con D=0.003 y en un intervalo de 0 y 10. Esto debido a que el número de elementos finales es una décima parte del total, lo cual en cierta forma garantiza una cuantización buena.

Cabe señalar que no se eligió ningún diccionario con una distorsión mayor, ya que aunque se facilite la búsqueda, no se garantiza una comunicación adecuada y de calidad.

Hasta este punto se ha realizado la comparación para los diccionarios obtenidos al aplicar la función de Lloyd a una secuencia de números aleatorios con una distribución uniforme, pero ¿y si se utiliza una distribución normal?

5.3.2 LGB: DISTRIBUCIÓN NORMAL

En este apartado se generará un diccionario a partir de una secuencia de números aleatorios pero con una distribución normal, es decir, una secuencia donde se localicen en el centro una concentración mayor de muestras (véase capítulo 2).

Para realizar la simulación se utilizará una secuencia con una distribución Gaussiana estándar, es decir, una distribución cuyos parámetros son media μ = 0 y varianza σ = 1. A continuación se presenta la secuencia generada formada por 1 000 elementos:

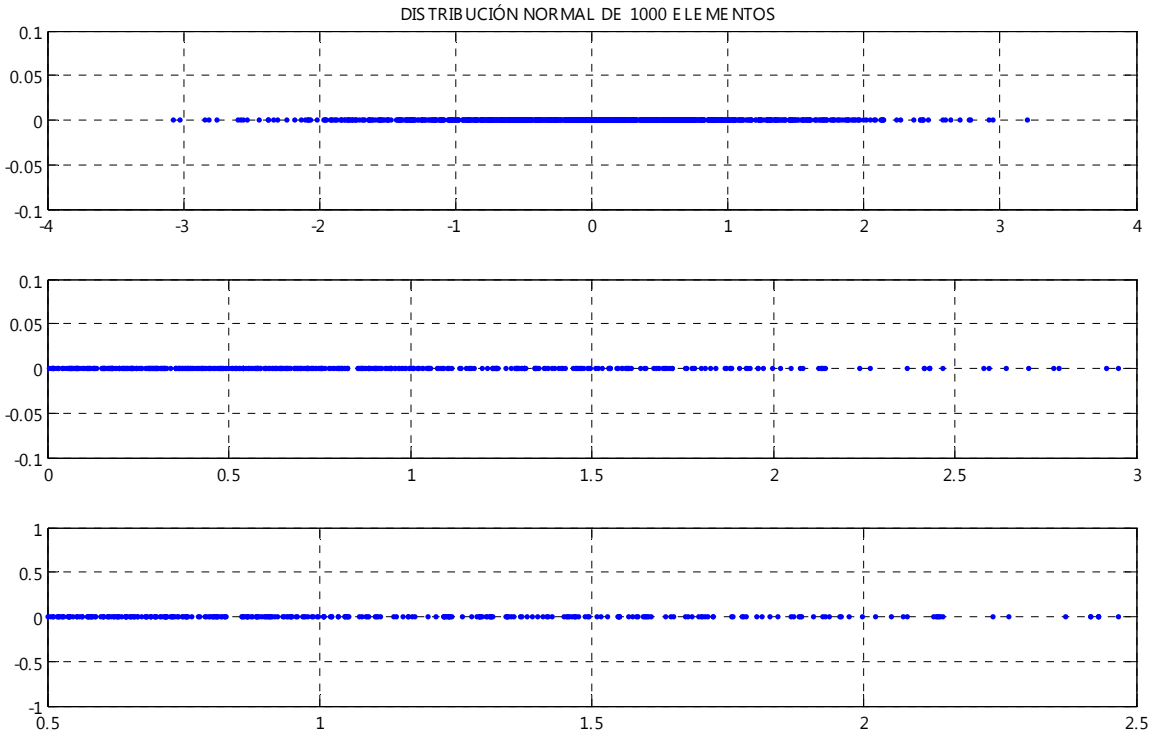


FIG. 5.21 DISTRIBUCÓN NORMAL DE 1000 ELEMENTOS

Como se puede apreciar en la gráfica anterior, los números generados pueden ser tanto positivos como negativos, además de que la mayor concentración de ellos se localiza cerca del origen, mientras que en los costados los puntos generados son mínimos. Una vez más se aplica la función diseñada para obtener los primeros centroides:

```
a=randn(1000,1);
[f,c]=lloyds(a,co)
f =
    0.0298
c =
   -0.7833    0.8515
```

En la siguiente gráfica se observa la localización de los centroides *C* en comparación con la secuencia original con distribución normal. Los centroides generados en la primera iteración se localizan cerca del origen, es decir en donde la concentración de números es mayor:

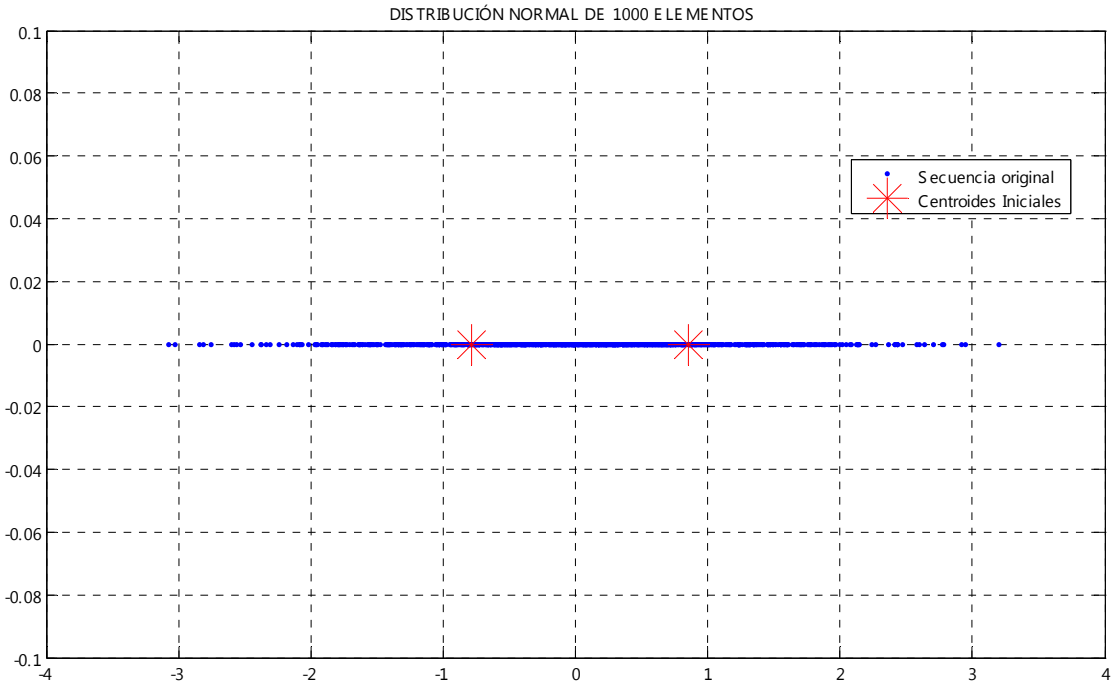


FIG. 5.22 DISTRIBUCIÓN NORMAL DE 1000 ELEMENTOS

Aquí se observan los primero dos centroides generados a partir de la secuencia original de 1000 elementos. A diferencia de los resultados obtenidos en la distribución uniforme, aquí es posible darse cuenta que los centroides no se encuentran distribuidos de una manera equidistante, es decir, en este caso se localizan cerca de donde se encuentra la mayor concentración de elementos, mas cerca del origen, por lo tomando en cuenta que la frontera esta casi en 0, el centroide no está precisamente en el centro de la nueva subregión.

Para observar las diferencias con mayor apreciación, se muestra en la gráfica de abajo la localización de los dos primeros centroides correspondientes tanto a la distribución normal como uniforme, esto con la finalidad de mostrar el efecto que tiene la distribución de la secuencia original para la determinación de dichos puntos:

```
x=randn(1000,1);
subplot(2,1,1); plot(x)
hold on
   co=[2,1];
   [f,c]=lloyds(x,co);
   c=[-0.7833 0.8515];
   plot(c)
hold off
a=rand(1000,1);
subplot(2,1,2); plot(a)
```

```
hold on
   co=[2,1];
   [f,c]=lloyds(a,co);
   c=[0.2475 0.7489];
   plot(c)
hold off
```

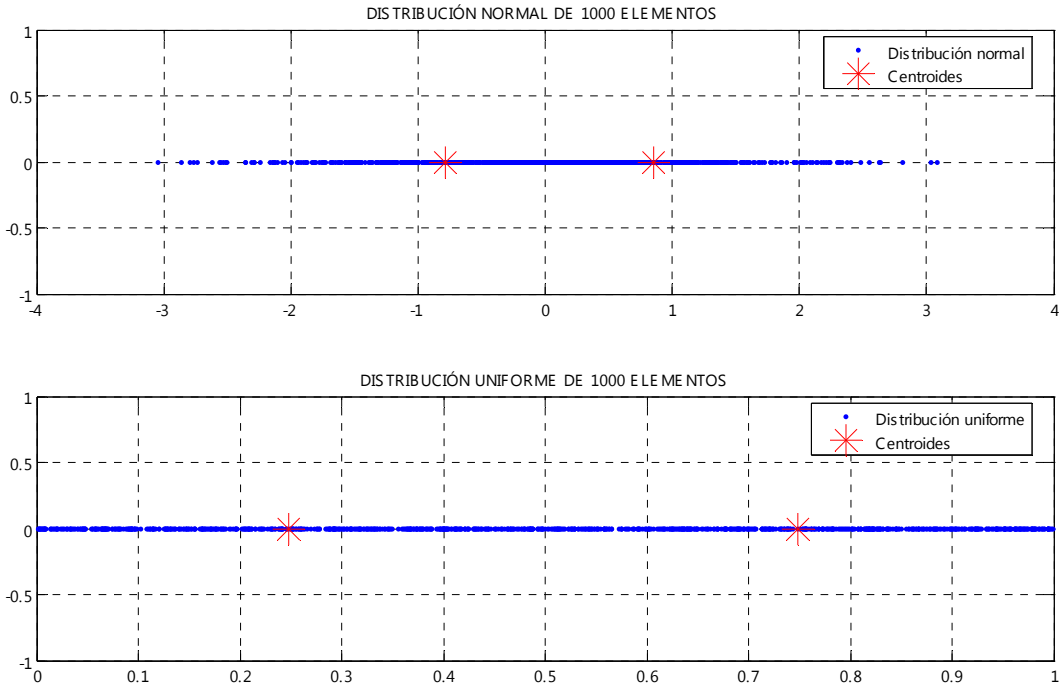


FIG. 5.23.A) CENTROIDES PARA UNA DISTRIBUCIÓN NORMAL

5.23.B) CENTROIDES PARA UNA DISTRIBUCIÓN UNIFORME

Según la gráfica 5.23 y como se había predicho anteriormente, la localización de estos puntos iniciales *C* varía en gran medida para cada una de las distribuciones utilizadas, teniendo para una distribución uniforme zonas equidistantes donde la probabilidad de que ocurra un evento es casi la misma para todos los puntos, mientras que para una distribución normal se localizan en la zona con mayor concentración de punto los cuales cuentan con una probabilidad mayor de que sucedan.

Finalmente, aplicando la función correspondiente se tiene el siguiente diccionario para la distribución normal mostrado a continuación para 1 000 elementos aleatorios con distribución normal y una distorsión mínima de 0.003:

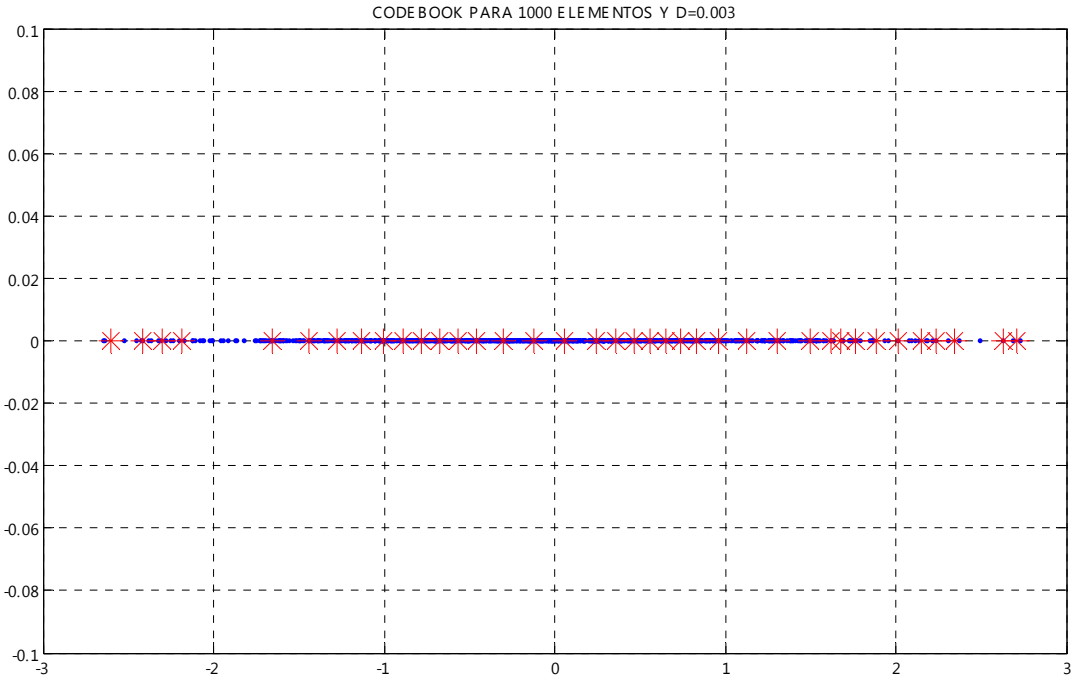


FIG. 5.24 CENTROIDES PARA UNA DISTRIBUCIÓN NORMAL

Como se observa en la figura 5.24 la distribución del diccionario final cuenta con una gran cantidad de elementos al igual que el caso uniforme, pero se puede observar también que no están distribuidos de uniformemente, sino que la mayor cantidad de puntos se localizan en el centro de la gráfica, y en los extremos hay, pero en menos cantidad.

Esto se debe a que en los extremos de la secuencia original las probabilidades de ocurrencia son menores a las del centro, pero aún así el *codebook* debe estar condicionado a que se ocupen con frecuencia o no, debe contener elementos que puedan representar a las muestras ocasionales que puedan surgir en esas regiones. Esto se verá con mayor claridad en el algoritmo COVQ con esta distribución, ya que deberán existir forzosamente centroides en las regiones en donde ni siquiera la secuencia original aparezca, sin embargo el que esta no lo haga no evita que la señal de entrada que es totalmente aleatoria pueda llegar a caer en esos lugares.

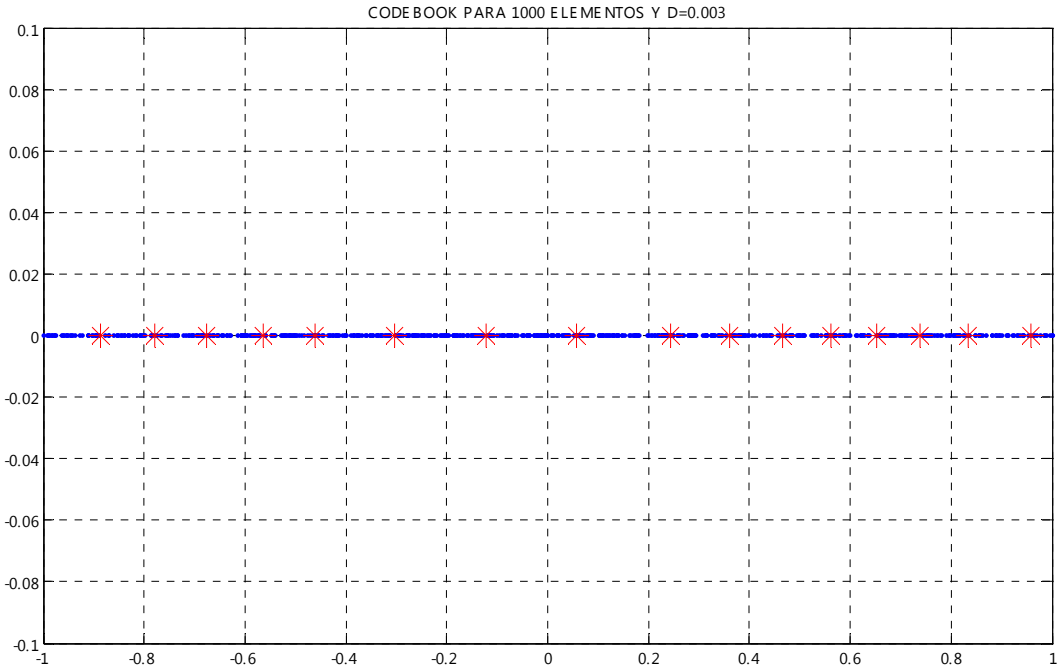


FIG. 5.25 CENTROIDES PARA UNA DISTRIBUCIÓN NORMAL [0,1]

En la figura 5.25 se puede apreciar el mismo diccionario generado pero con un acercamiento en el intervalo positivo [0,1] con la finalidad de observar el comportamiento de los centroides. Como se observa la distancia de cada uno de ellos no es la misma debido a que la señal de entrada tiene una función de probabilidad que hace las muestras más dispersas en ciertas zonas. Sin embargo para este intervalo en particular, donde la secuencia podría tener un comportamiento cercano al de una distribución uniforme, la localización de los centroides es muy similar a las que se observaron para el primer caso del algoritmo LGB con la secuencia uniforme debido a que en esta zona se localizan la mayor cantidad de muestras de la señal original.

Por otro lado, se si observa el comportamiento del diccionario en un intervalo mayor [0,3] es posible percatarse de las diferencias, justo como se muestra en la figura 5.26 donde la cantidad de centroides generados fue menor en el extremo positivo que en el centro (cerca del origen) por las razones que se han venido explicado. Además es fácil comprender que la existencia de estos centroides en específico es de suma importancia, ya que no se puede garantizar que la señal de entrada no vaya a caer específicamente en esas zonas, y si se omiten centroides únicamente por la poca probabilidad de ocurrencia que se tenga, pueden generarse problemas más adelante ya que se tendrían que colocar las muestras de acuerdo a los centroides generados en el resto de la distribución, aumentando significativamente el valor de la distorsión.

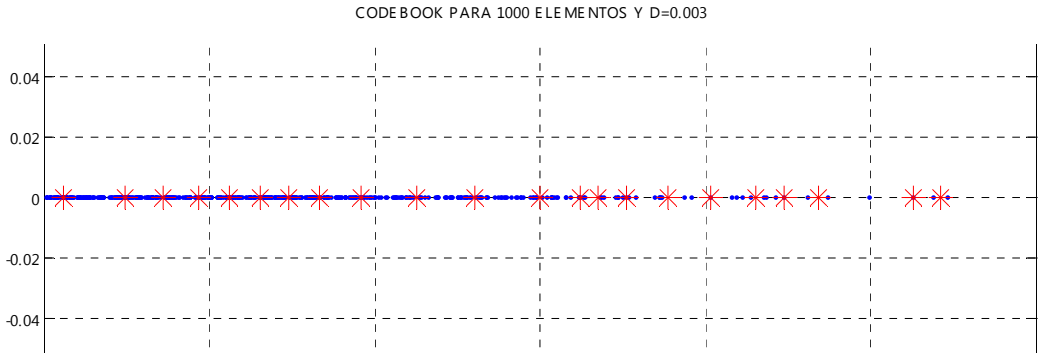


FIG. 5.26 CENTROIDES PARA UNA DISTRIBUCIÓN NORMAL [0,3]

Hasta este momento se han presentado únicamente los resultados obtenidos a partir de una secuencia original de 1 000 elementos y una distorsión de D=0.003 con una distribución normal. Para este caso se obtuvo un *codebook* de 64 elementos tanto negativos como positivos (véase anexo 3) en el intervalo de I[-3,3] y de igual forma se han presentado sus representaciones gráficas.

A continuación se presentan los resultados obtenidos para el mismo experimento variando únicamente el numero de elementos que componen la secuencia original de 500 elementos, logrando con esto que el numero de elementos por zona disminuya a la mitad, esto con la finalidad de observar el comportamiento de los centroides que se generan en las zonas de menos de probabilidad (en los extremos de la secuencia).

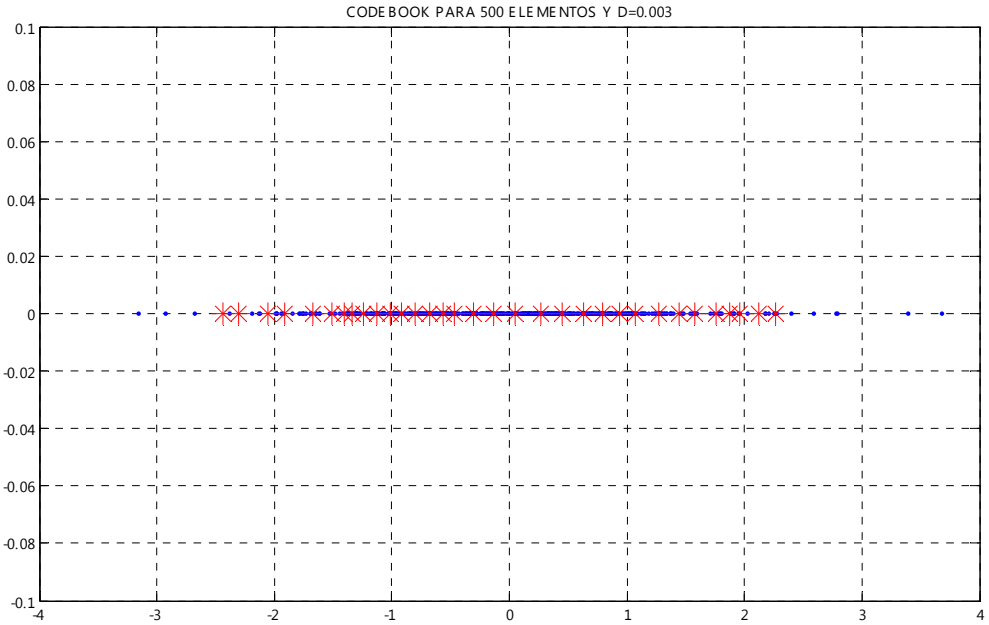


FIG. 5.27 CENTROIDES PARA UNA DISTRIBUCIÓN NORMAL DE 500 ELEMENTOS [-3,3]

En la figura 5.27 se muestra el diccionario generado para las condiciones mencionadas con anterioridad y es posible darse cuenta de que el comportamiento presenta ciertos cambios en comparación con las características anteriores, sobre todo en los extremos donde de inicio, el número de elementos iniciales disminuyó a la mitad. El cambio más significativo es que ahora no hay centroides en las regiones de poca probabilidad, lo cual es un gran problema ya que no se garantiza que esas regiones nunca vayan a ser ocupadas. A continuación se presenta un acercamiento de estas regiones:

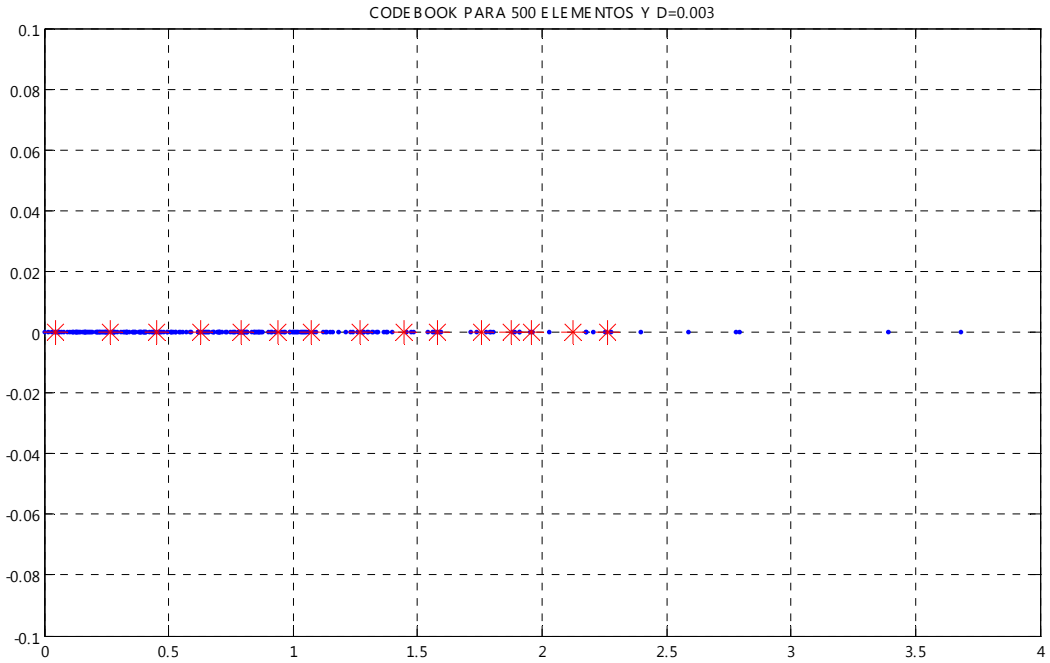


FIG. 5.28 CENTROIDES PARA UNA DISTRIBUCIÓN NORMAL DE 500 ELEMENTOS [0,4]

De acuerdo a los resultados, si una muestra de la señal de entrada se posiciona en el intervalo [2.5, 4] el diccionario no tendrá posibilidades de asignarle un valor con el que pueda garantizar una distorsión mínima, por lo que a todas las muestras en este inérvalo no tendrá más opción de asignar el valor de c = 2.2609, aunque en el resto de la secuencia se tenga un buen comportamiento.

En la siguiente gráfica se observa el diccionario en el intervalo positivo [0, 2], intervalo que se podría denominar ‘óptimo’ ya que en él, el proceso de cuantización se da de manera normal ya que todos los centroides garantizan una distorsión máxima de 0.003 tanto para el eje positivo como el negativo:

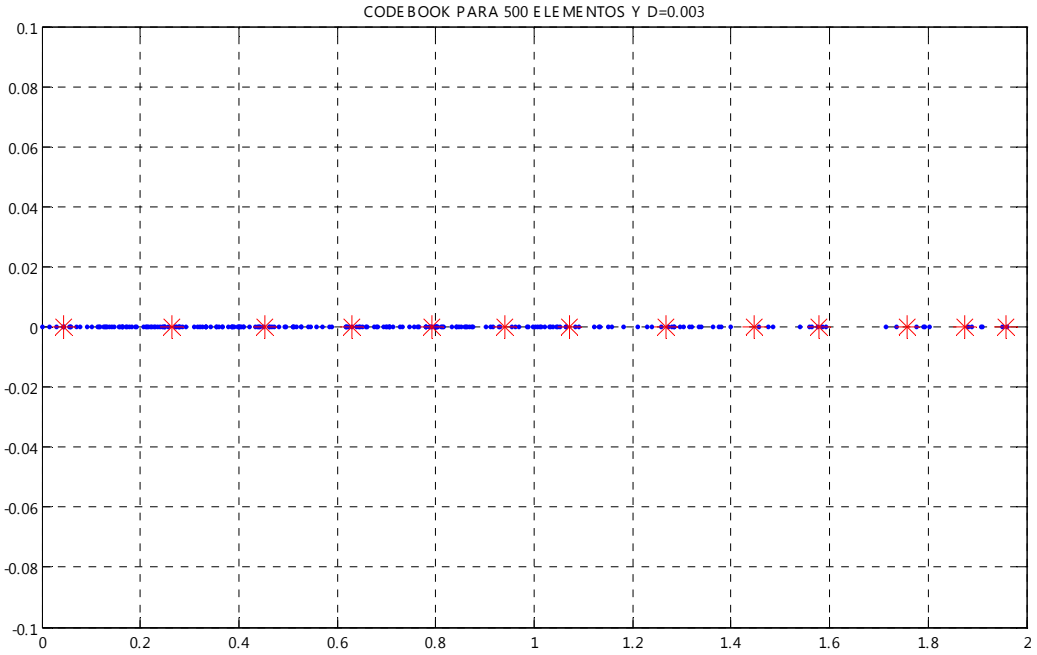


FIG. 5.29 CENTROIDES PARA UNA DISTRIBUCIÓN NORMAL DE 500 ELEMENTOS [0,2]

Finalmente, las características de este diccionario son: se creó a partir de 500 elementos con distribución normal y con una distorsión máxima de 0.003 y el *codebook* final estuvo formado de 64 elementos en el eje positivo y negativo (véase anexo 4).

De esto se puede ver fácilmente que de los dos diccionarios creados para este tipo de distribución, el más óptimo fue el primero generado a partir de 1 000 elementos. Se podría pensar que si de ahorrar elementos se trata, el caso de 500 elementos es el mejor ya que ambos generan la misma cantidad de elementos finales, sin embargo el primer caso es el de mayor eficiencia debido a que permite que las muestras de poca probabilidad sean cuantizadas de manera óptima respetando el margen de distorsión mínimo, a diferencia del segundo caso donde la mayor concentración de centroides se localiza al centro de la distribución dejando los extremos sin puntos que representen a las posibles muestras de entrada.

5.3.3 CUANTIZACIÓN VECTORIAL

Para saber cual de los diccionarios es el más óptimo, a continuación se realizará una comparación basándose en los mismos aspectos que se trataron con la cuantización escalar para el algoritmo LGB, es decir, en el número de elementos finales que se crearon dependiendo del valor de la distorsión.

Como se puede ver en los resultados del Anexo 5, para la cuantización basada en el algoritmo COVQ únicamente se utilizaron de manera inicial 10 000 vectores y solo existen dos intervalos de comparación [0,5] y [0,1], esto debido a que, como se mencionó anteriormente, si se maneja un intervalo más amplio, por ejemplo entre 0 y 10, el número de elementos finales crece de manera exponencial, sin mencionar que el caso de 0 y 100 generaría un diccionario imposible de plasmar.

La tabla que sume los resultados obtenidos es la siguiente:

Distorsión	***Numero de elementos finales***	***Numero de elementos iniciales***	***Intervalo [0,]***
10^{-3}	1156	10000	5
0.5	16	10000	5
1	4	10000	5
10^{-3}	64	10000	1
0.5	4	10000	1
1	4	10000	1

TABLA 5.3 RESULTADOS FINALES DEL ALGORITMO COVQ

Y en la siguiente gráfica se muestra la relación entre los elementos generados a partir de la variación de la distorsión, recordando que un aumento en la distorsión provoca que cada uno de los centroides generados abarque en este caso, un área mayor, por lo que algunos puntos que se encuentren alejados sí podrán asociarse a una pareja ordenada, pero esto provocará que las probabilidades de error aumenten ya que el margen es mayor. La gráfica corresponde al intervalo [0,1]:

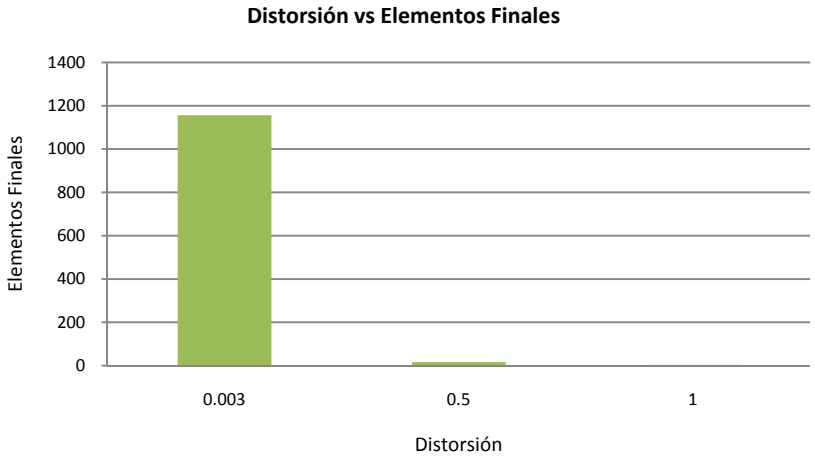


FIG. 5.30 RELACIÓN DISTORSIÓN VS ELEMENTOS FINALES PARA [0,5]

Como se puede observar, para un valor de distorsión casi nulo (D=0.003) y para el intervalo de 0 a 5, el número de elementos que conforman el *codebook* es muy parecido al generado en el algoritmo LGB para un intervalo de 0 a 100. Esto da la pauta para evitar hacer iteraciones en intervalos mayores.

Además se observa que al variar la distorsión el número de elementos finales disminuye en gran cantidad, lo que indica que existe un gran margen de error.

Cabe mencionar que para e caso de la cuantización escalar, la distorsión se media a lo más en dos direcciones, por lo que el error generado consistía en pasar de un punto C_1 a un punto C_2.

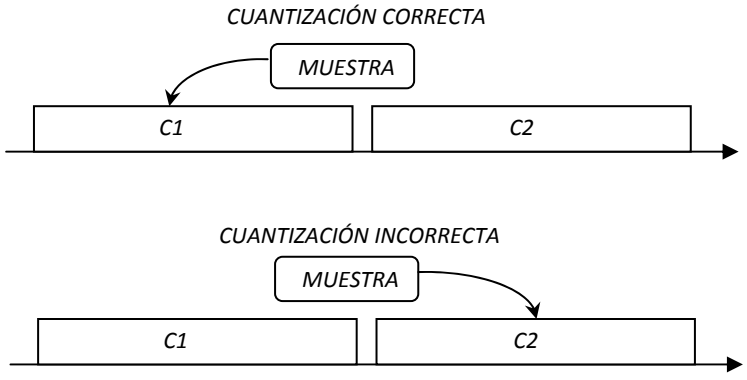


FIG. 5.31.A) CUANTIZACIÓN LGB CORRECTA

5.31.B) CUANTIZACIÓN LGB INCORRECTA

Pero para el caso de la cuantización vectorial un error no consiste en solo dar un punto diferente, sino varias al mismo tiempo, ya que el transmisor puede equivocarse en asignar ambas

coordenadas y terminar por proporcionar un dato que no represente en lo absoluto a la muestra que entra:

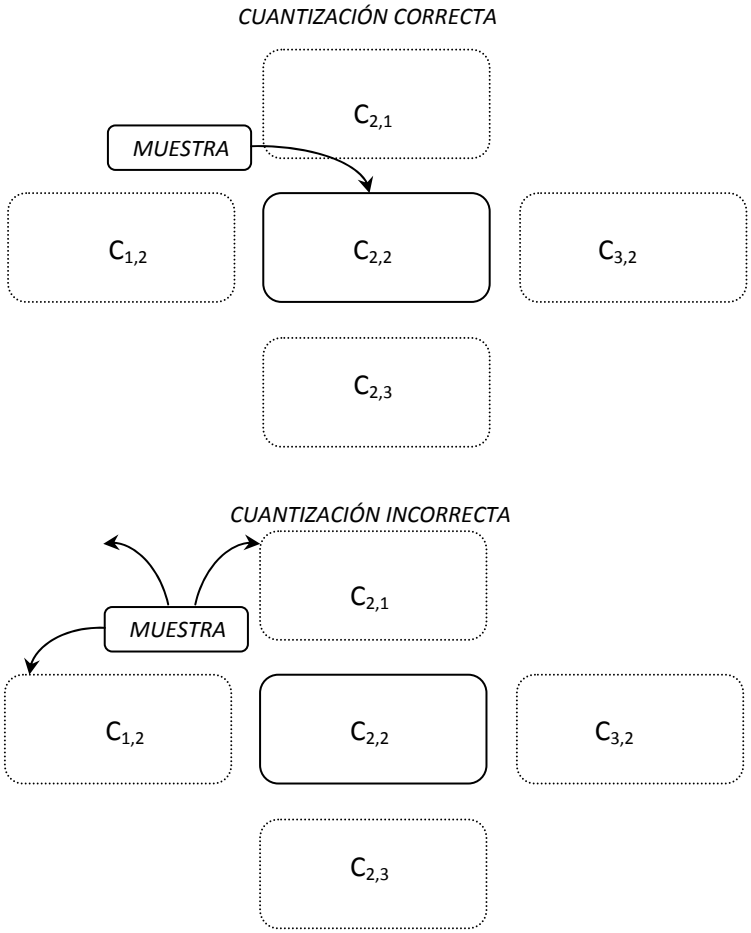


FIG. 5.32.A) CUANTIZACIÓN COVQ CORRECTA

5.32.B) CUANTIZACIÓN COVQ INCORRECTA

Por tal motivo es considerablemente bueno pensar que a pesar de tener un *codebook* con muchos elementos puede disminuir la velocidad de proceso, pero al mismo tiempo ahorra problemas al evitar crear errores que sucederían si el proceso fuero más rápido y tuviera que buscar la asociación en una base de datos más pequeña.

En la siguiente gráfica se muestran los resultados obtenidos para la comparación de los diccionarios correspondientes al intervalo [0,1]:

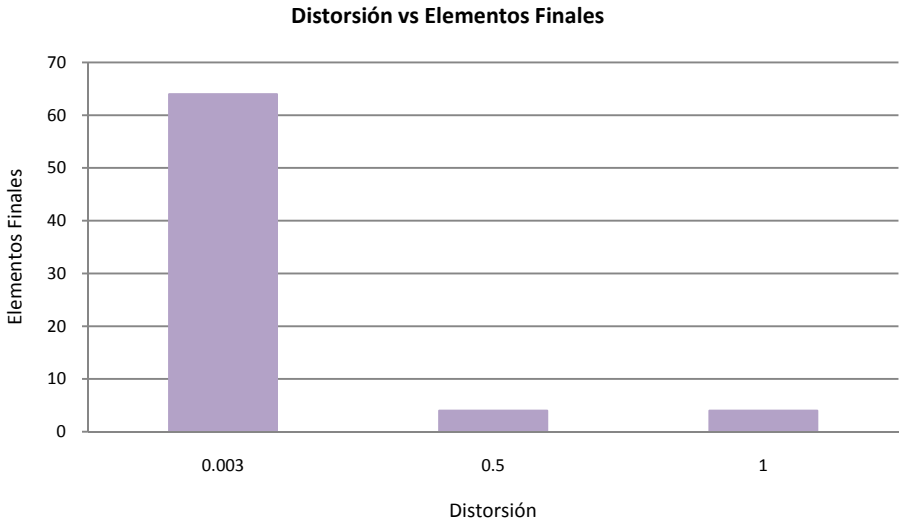


FIG. 5.33 RELACIÓN DISTORSIÓN VS ELEMENTOS FINALES PARA [0,1]

Como se aprecia, el resultado es similar al anterior, es decir, para el valor mínimo de distorsión se generó un diccionario con un número elevado de elementos, y este fue disminuyendo conforme el valor de *D* aumenta.

Sin embargo también se puede ver la modificación de este parámetro deja de tener efectos significativos, es decir, debido a que el intervalo es pequeño, no es necesario crear muchos centroides ya que para codificar tanto 10 000 vectores como 100 en el intervalo de [0,1] es posible hacer mediante los mismos centroides, esto debido a que ya el margen de error permitido es muy grande.

De este conjunto de diccionarios creados para la representación del algoritmo COVQ, puedo decir que los siguientes son los más óptimos:

Distorsión	***Numero de elementos finales***	***Numero de elementos iniciales***	***Intervalo [0,]***
10^{-3}	1156	10000	5
10^{-3}	64	10000	1

TABLA 5.4 CARACTERÍSTICAS DE LOS DICCIONARIOS SELECCIONADOS

El resultado es claro, de inicio se deben evitar diccionarios que presenten mayor vulnerabilidad a los errores, por lo tanto los creados a partir de D=0.5 y D=1 quedan omitidos. Esto reduce la lista a los dos presentados.

En ellos se observan las mejores características ya que ambos ofrecen gran cantidad de elementos para representar cualquier valor de las señales de entrada. Para el caso del primero, su cantidad de elementos es mayor, pero debe considerarse que su intervalo también lo es. Mientras que para el segundo, su número de elementos disminuye, pero su intervalo puede perfectamente contener todo tipo de señales. Además de que ambos fuero diseñados a partir de 10 000 vectores aleatorios, lo que sirve como referencia para saber que se realizaron las iteraciones de forma correcta y cada uno de los centroides representan la mayor cantidad de puntos posibles.

5.3.4 COVQ: DISTRIBUCIÓN NORMAL

Para el caso del algoritmo COVQ también se optó por hacer muestras variando la distribución de la secuencia de entrada con la finalidad de observar el comportamiento y localización de los centroides finales del diccionario. Inicialmente se utilizaron 1 000 vectores de dos dimensiones con distribución normal (gaussiana). A continuación se presenta la secuencia inicial:

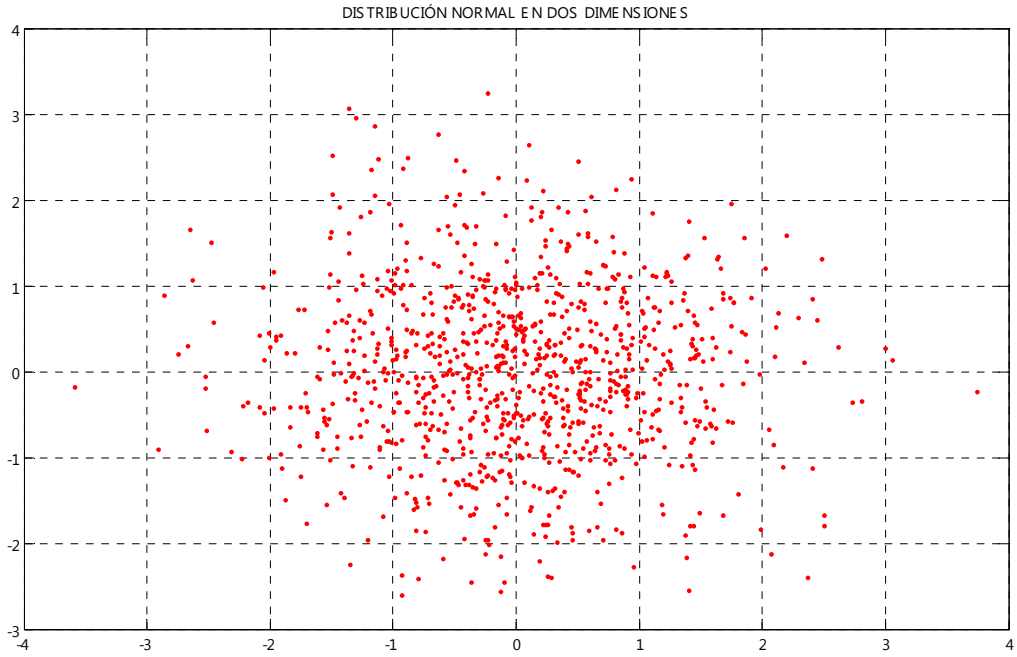


FIG. 5.34 SECUENCIA CON DISTRIBUCIÓN NORMAL DE 1000 ELEMENTOS EN 2D

Debido a que en este caso se esta trabajando en dos dimensiones en fácil darse cuenta de varios aspectos que en el caso de LGB para la misma distribución no se apreciaban a simple vista,

el primer aspecto importante es que la distribución entre los cuatro cuadrantes no es uniforme ni simétrica, es decir, solo en los intervalos [-1,1] tanto para el eje X y Y se parecía un comportamiento parecido, pero fue de estos la distribución varía mucho.

Otros aspectos importantes es que la mayor concentración esta alrededor del origen y esta va disminuyendo conforme se aleja de el en todas las direcciones.

Al igual que al resto de las distribuciones, se aplica la función creada y se obtiene la siguiente representación del diccionario buscado para los 1 000 vectores de entrada:

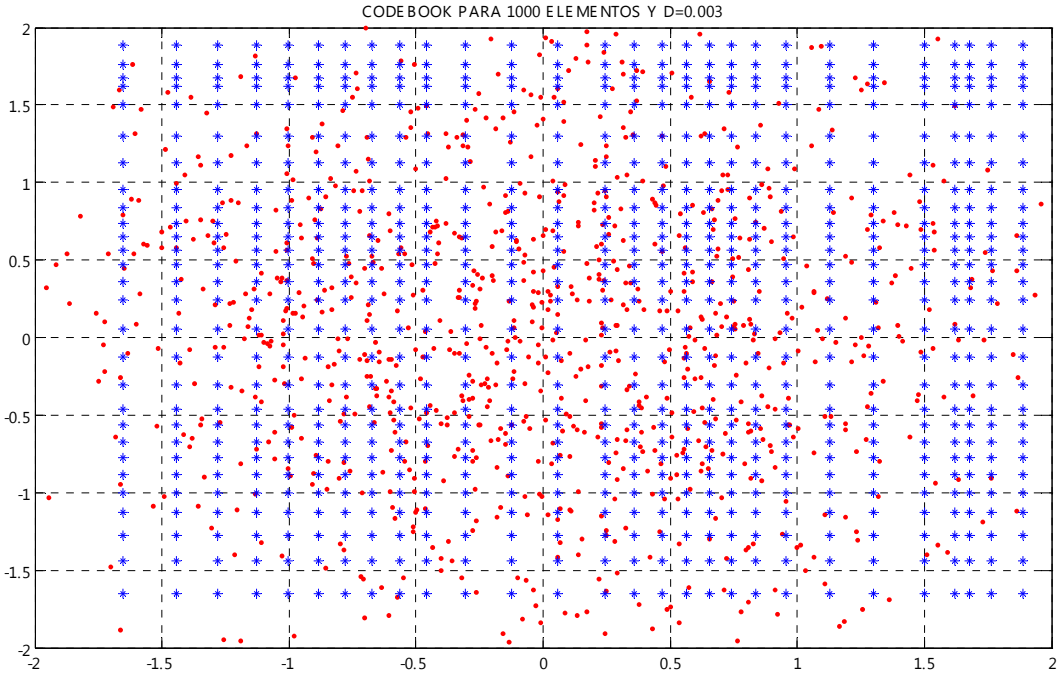


FIG. 5.35 DICCIONARIO PARA 1000 ELEMENTOS Y D=0.003

En la figura anterior se muestra el *codebook* generado a parir de la secuencia original de 1000 vectores. Como se observa el resultado es muy similar al obtenido para el caso de la distribución uniforme, pero si existen pequeñas diferencias que se explicarán a continuación.

La diferencia más notable que se puede ver a simple vista del resultado obtenido es que existe una menor presencia de centroides en las orillas de la gráfica, es decir, en los extremos de la señal original donde el número de muestras es menor debido a la probabilidad de ocurrencia. Además de que nuevamente se tiene un comportamiento similar al de una reja que cubre cada uno de los puntos del plano en su mayoría de forma uniforme en los intervalos [-2,2].

Al igual que se mencionó en el algoritmo LGB para esta distribución, después de ver el resultado obtenido se generaron centroides en zonas del plano en donde de forma inicial no había

señal de entrada, pero esto se debió a que son necesarios, ya que para una señal de entrada cualquiera deben existir todas las posibilidades para poder cuantizarla de forma correcta. En la gráfica 5.36 se observa el diccionario obtenido en un intervalo más pequeño de valores:

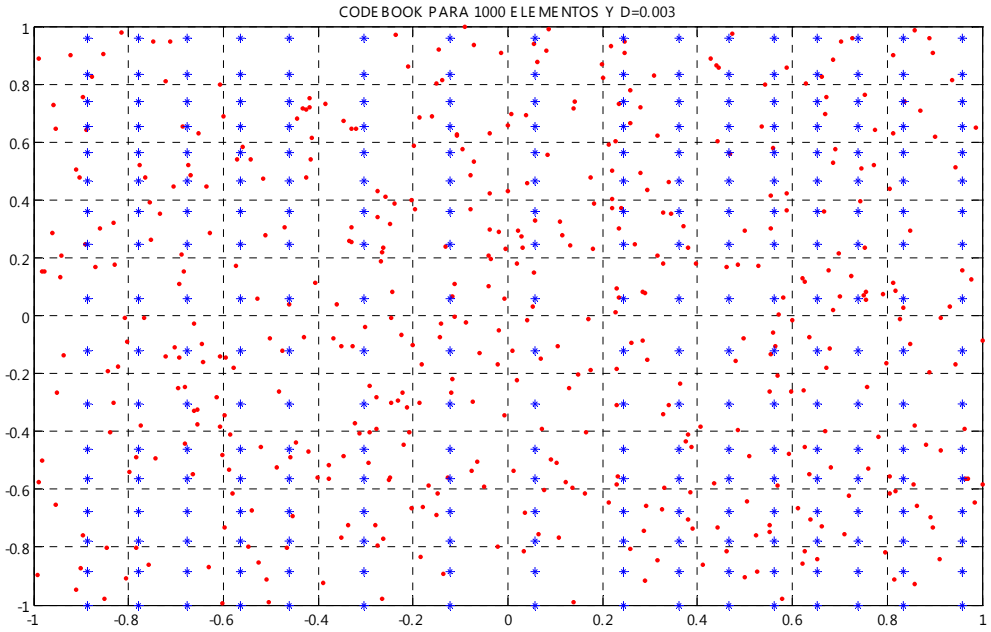


FIG. 5.36 DICCIONARIO PARA 1000 ELEMENTOS Y D=0.003 EN [-1,1]

Para los 1 000 vectores con distribución normal de entrada y con D=0.003 se obtuvo un diccionario final de 1 156 elementos, sin embargo para el caso de 500 vectores aleatorios de obtuvieron las misma cantidad de parejas ordenadas (véase anexo 7). A continuación s presentan los resultados para este segundo caso:

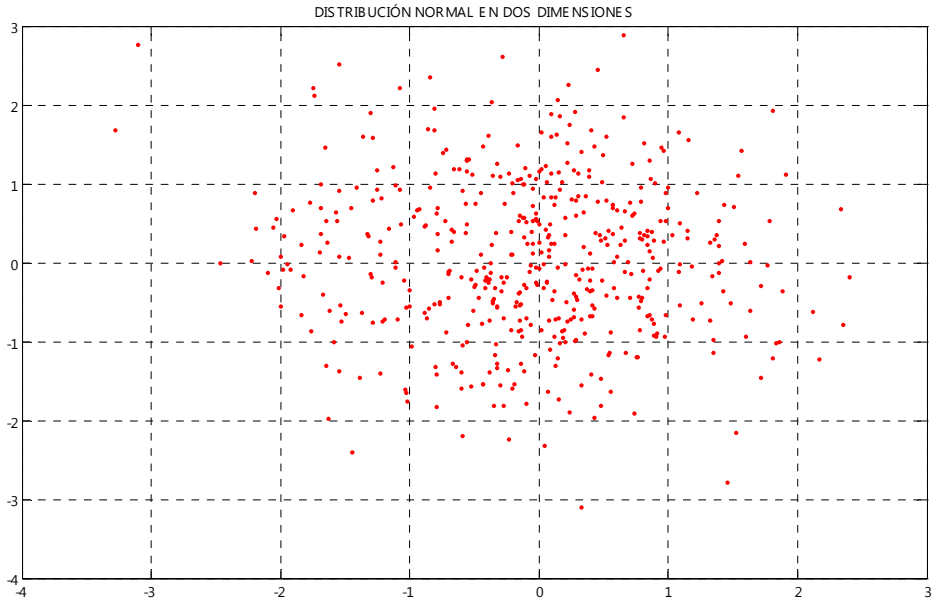


FIG. 5.37 SECUENCIA CON DISTRIBUCIÓN NORMAL DE 500 ELEMENTOS

A diferencia de la secuencia de 1 000 vectores, en esta figura se puede observar una distribución mucho más dispersa y con mas 'huecos', además de que las zonas de probabilidad baja presentan menos cantidad de muestras.

Y el diccionario se obtuvo mediante la aplicación de la función diseñada y el resultado final se muestra a continuación:

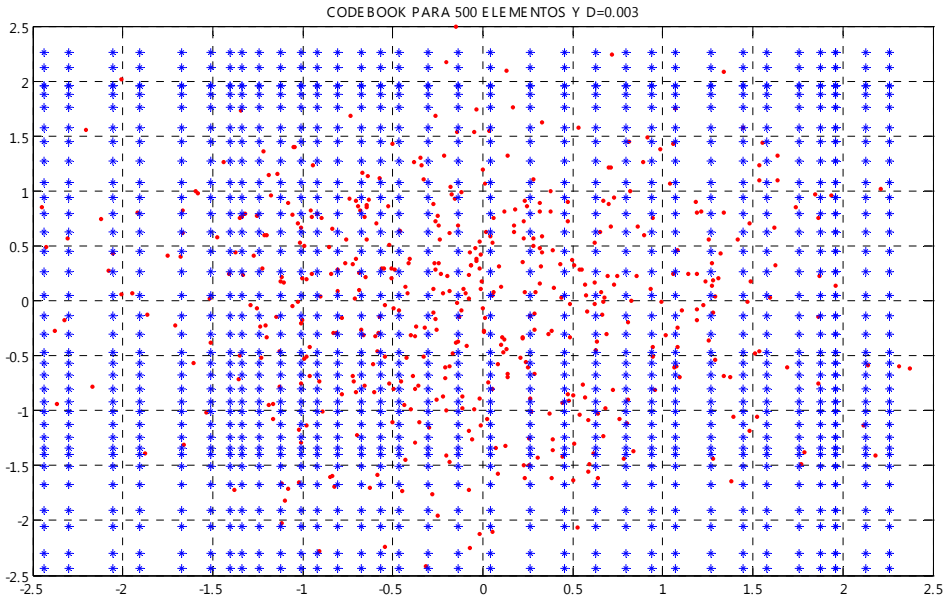


FIG. 5.38 DICCIONARIO PARA 500 ELEMENTOS Y D=0.003

Para este caso es más factible utilizar el diccionario obtenido con estas características ya que además de tener el mismo número de elementos que el del primer caso, ahora si permite la correcta cuantización de muestras posiblemente localizadas en los extremos de la distribución aún proviniendo de una distribución con una cantidad pequeña de muestras en estas zonas.

5.4 RESULTADOS

En el desarrollo del presente capítulo, se realizaron las simulaciones correspondientes para obtener un *codebook* representativo de la cuantización escalar (algoritmo LGB) y vectorial (algoritmo COVQ). Además se incluyó el *codebook* seleccionado como el más óptimo a partir de las especificaciones dadas en la comparación.

Sin embargo, para cuestiones de comparación, a continuación se presentan los diferentes *codebook* obtenidos con cada una de estas variaciones. Estos se presentan en forma de anexos, divididos en ocho apartados.

Para establecer cual de todos los resultados obtenidos, se compararon varios aspectos como el intervalo de trabajo, el número de elementos finales que conformaron el diccionario, el valor de distorsión a partir del cual estos se crearon.

En el caso de cuantización escalar del algoritmo LGB se seleccionó el diccionario creado a partir de un valor de distorsión D=0.003, es decir, un valor muy pequeño. Además se manejó en el, un intervalo mayor [0,100] para dar más oportunidad a las señales de entrada a ser representadas con más detalle, y el número de elementos finales fue de 1212. Estas tablas se localizan en el anexo 1 de forma numérica y de forma gráfica están contenidos en el anexo 2 para el caso de la distribución uniforme.

Los primeros dos anexos corresponden exclusivamente a los resultados del algoritmo LGB, en donde se presentan las tablas con diferentes *codebook* generados a partir de diferentes valores de distorsión, desde D=[10^{-3}, 0.5, 1, 10, 50]. Otro parámetro importante fue el intervalo en que se generaron las muestras I={[0,100], [0,10], [0,1]}, e incluso el número total de muestras iniciales. Estos dos primero anexos corresponden a una distribución de probabilidad uniforme para las secuencias de entrada al sistema. El anexo dos incluye estos mismo resultados, pero de una forma gráfica, con la finalidad de observar detalladamente la localización de cada uno de los valores.

También se agregan en los anexos 3 y 4 las tablas y gráficas correspondientes para la distribución normal en el caso de cuantización escalar.

En el caso de la cuantización vectorial del algoritmo COVQ, se seleccionó el diccionario creado con las siguientes condiciones iniciales: una distorsión D=0.003, en un rango de [0,5] y con 1156 parejas ordenadas, las cuales pueden ser consultadas en los Anexos 5 y 6 tanto de forma numérica como gráfica.

Finalmente, los anexos 7 y 8 contienen los resultados correspondientes a la cuantización vectorial del algoritmo COVQ pero con una distribución de probabilidad normal para los valores de entrada.