UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

Dpto. de Teoría de la señal y Comunicaciones

Ingeniería Técnica en Sistemas de Telecomunicación

Trabajo Fin de Carrera

Optimización de modulaciones caóticas codificadas mediante algoritmos genéticos

Autor: D. Luis Calvo Ortego
Profesor Tutor: D. Francisco Javier Escribano Aparicio

Presidente:

Vocal 1:

Vocal 2:

CALIFICACIÓN: ..... FECHA: .....

A mis padres y hermano
por todo su apoyo, ánimo,
confianza y comprensión.

Índice general

Índice de figuras

Índice de tablas

Índice de diagramas

Capitulo 1

1 Resumen

1. Resumen

En este trabajo se va a desarrollar la optimización de una clase de sistemas de comunicaciones basados en caos, para esto se emplearán técnicas evolutivas, en concreto del tipo algoritmo genético con el objetivo de minimizar la tasa de error producida.

A lo largo de este trabajo se detallarán las herramientas y ficheros empleados así como la caracterización del sistema de comunicaciones basado en caos dando una función de coste relacionada directamente con la cota de error que se pretende minimizar, empleando algoritmos genéticos para encontrar el juego de parámetros que minimicen dicha función de coste. También se indicará el enfoque del problema y las pruebas para su desarrollo y optimización, analizando tanto la solución tomada como su interpretación y su repercusión, para dar las pautas generales para el diseño y evaluación de dichos sistemas.

Uno de los problemas básicos asociados a la aplicación de la teoría del caos en comunicaciones digitales es la carencia de una teoría sólida y fácilmente aplicable, debido a la complejidad y no linealidad inherente de dichos sistemas. El desarrollo de este trabajo de optimización heurística permitirá avanzar en la caracterización útil de los sistemas de comunicaciones basados en caos.

Palabras clave:

Algoritmo genético, modulación caótica, función de coste, cota de la probabilidad de error binaria.

Capítulo 2

2 Summary

2. Summary

In this paper we will develop the optimization of a class of communication systems based on chaos. To this purpose, we will use evolutionary techniques, namely genetic algorithms, to minimize a bound for the bit error probability.

Throughout this paper we will detail the tools and files employed as well as the characterization of the communication system based on chaos together with the cost function directly related to the bit error probability bound we want to minimize, using genetic algorithms to find the set of parameters that minimize this cost function. We also analyze the approach to the problem and we test the development and optimization, indicating both the solution obtained as well as its interpretation and predicted effects, in order to give general guidelines for the design and evaluation of such systems.

Keywords:

Genetic algorithm, chaotic modulation, cost function, bit error probability bound.

Capítulo 3:
3 Algoritmo genético

Introducción a los algoritmos genéticos

Algoritmo utilizado

3. Algoritmos Genéticos

3.1 Introducción a los algoritmos genéticos.

Un algoritmo genético es un método de optimización ideado por John Henry Holland en la década de 1970 [5]. Holland se basó en la evolución biológica para diseñar este tipo de algoritmo en el que una población de posibles soluciones a un problema es sometida a un proceso de selección y combinación para producir mejores soluciones en cada iteración.

La capacidad de los algoritmos genéticos para explorar estructuras alejadas de los modelos convencionales ha resultado ser muy útil en la mejora y diseño de nuevos programas y dispositivos. Estos algoritmos han demostrado su capacidad para solucionar problemas combinatorios. En nuestro caso, se realizará enfocado a la obtención óptima de los parámetros y las muestras a lo largo del programa.

Para trabajar con los algoritmos genéticos se parte de una población inicial de individuos generados aleatoriamente. Los individuos son codificaciones que representan el tipo de solución que se quiere obtener. A cada individuo de esta población se le evalúa respecto al problema que se está solucionando. La función que se encarga de esto se denomina función de de ajuste, o *fitness* [4] [5]. Una vez que todos los individuos han sido evaluados, se realiza la selección. Existen muchos sistemas para seleccionar a los individuos, la mayoría de ellos favorecen la selección de aquellos cuyo resultado de la función de ajuste es mejor.

Para la selección de un individuo, éste se cruzará con otro de los individuos seleccionados para generar uno nuevo que formará parte de nueva población.

Cuando se ha obtenido la nueva población, el operador de mutación modifica alguno de los individuos, con el fin de añadir diversidad a las soluciones que mediante el cruce van convergiendo hacia una única configuración.

Se suele añadir un elemento de elitismo para copiar automáticamente al mejor de los individuos de una generación en la siguiente. De esta manera se evita que desaparezca por la aleatoriedad del sistema de selección la que puede ser la solución óptima al problema.

El diagrama típico de un algoritmo genético es el siguiente:

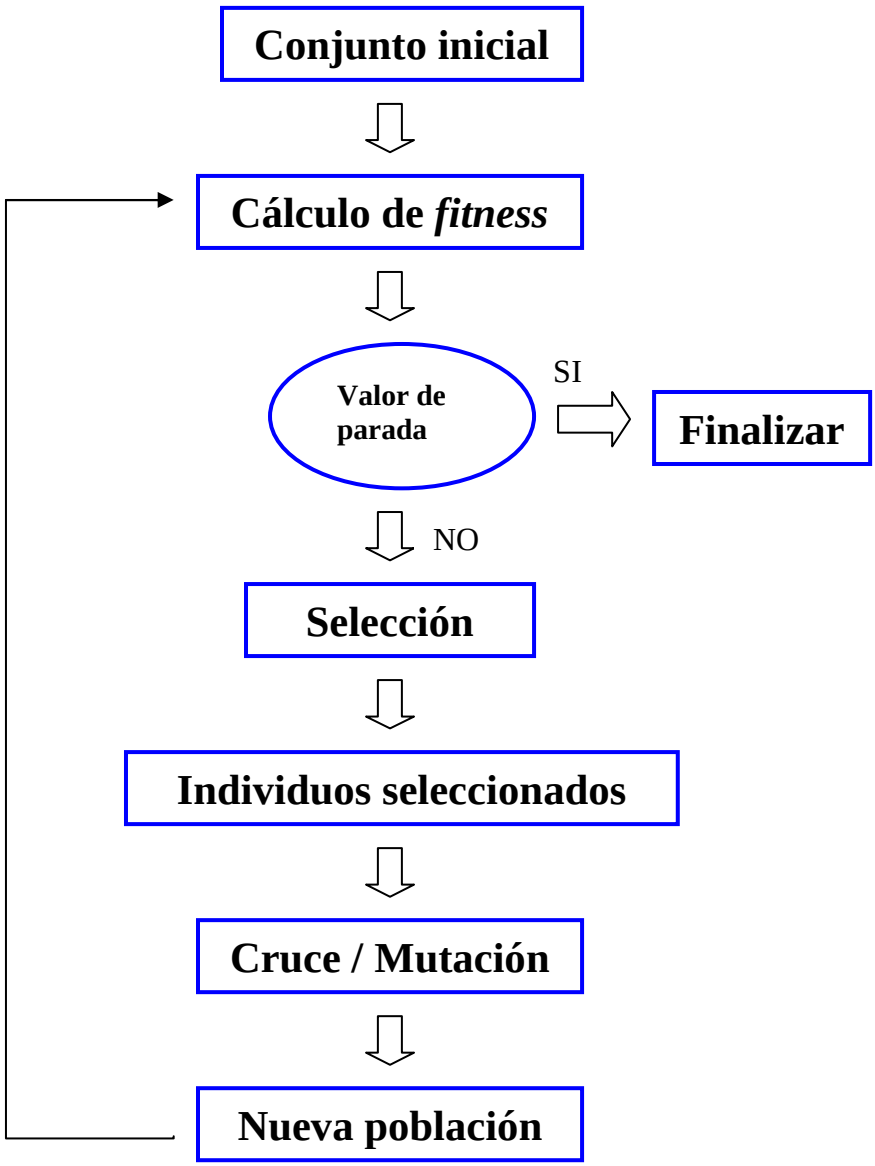

Diagrama 1 Flujo de un algoritmo genético

3.2 Algoritmo utilizado

Se utilizará una versión modificada del modelo canónico [5] del algoritmo genético. La principal diferencia vendrá dada por el sistema de selección de los individuos. Se ha de tener en cuenta que la función de ajuste se corresponderá con el procesado de la información.

Un algoritmo genético tiene su principal inconveniente en la necesidad de un conjunto suficientemente grande de individuos, para que se pueda cubrir de manera efectiva el espacio de búsqueda antes de que éste converja prematuramente.

El operador de mutación ha sido introducido para evitar la convergencia prematura del sistema y aumentar la diversidad de la población, pero hay que tener en cuenta que si la mutación es excesiva el algoritmo trabajará como una búsqueda aleatoria.

Para evitar este problema se ha modificado el sistema de selección y cruce, de esta manera se podrá utilizar una población no muy grande y una probabilidad y porcentaje de mutación altos, sin perder las propiedades del algoritmo genético.

Donde difiere realmente es en los bloques de selección, cruce, mutación y en la entrada al bloque de cálculo del *fitness*.

El bloque de selección no conlleva un proceso aleatorio de selección, sino que realiza el promedio de los valores del *fitness* de la población y toma aquellos cuyo *fitness* es mejor que la media, así una vez se han escogido los valores más aptos se desecha el resto.

En la parte de cruce y mutación se introducen nuevos individuos en la población y se selecciona aleatoriamente un par de individuos de los supervivientes, a partir de los cuales se crea un único individuo nuevo. Hay que tener en cuenta que existe una determinada probabilidad de que un nuevo miembro sufra un cierto porcentaje de mutación.

De esta forma se crea una nueva generación sobre la que se seguirá trabajando. Los elementos que fueron seleccionados como aptos se encuentran de nuevo en la población, y puesto que su función de ajuste ya ha sido calculada, no es necesario repetir el cálculo con ellos. Estos individuos permanecerán en la población hasta que la media del *fitness* de la población sea mayor que el suyo.

Dado que el sistema que se pretende optimizar corresponde a un sistema de comunicaciones basado en caos, en el apartado siguiente se caracterizará dicho sistema y se propondrá la función de coste que se usará en el cálculo del *fitness.*

Capítulo 4:
4 Modulaciones caóticas

Objetivo

4. Modulaciones caóticas

El esquema de comunicación de una modulación caótica en un canal con ruido gaussiano aditivo se basa en el siguiente diagrama:

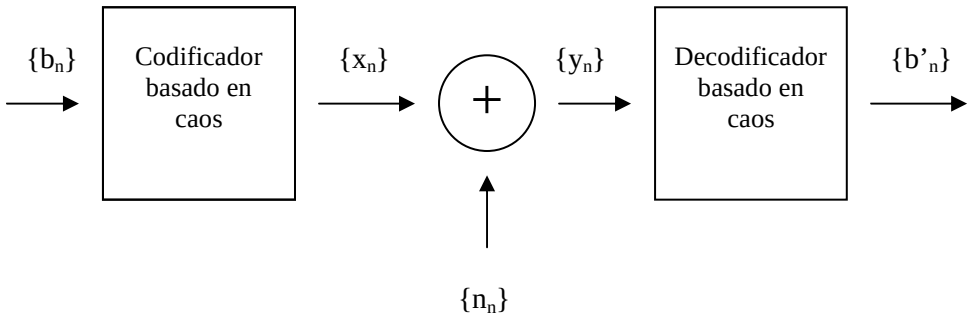

Diagrama 2 Modulación caótica

En él se muestra la llegada de una secuencia de bits a un codificador caótico, donde se generarán las muestras caóticas a partir de dicha secuencia binaria y de una aplicación caótica basada en la aplicación de Bernoulli; a la señal que se obtiene a su salida se le añade otra que corresponde al ruido aditivo blanco gausiano. Esta combinación es decodificada en el demodulador caótico donde se obtendrá una secuencia estimada de bits, con la correspondiente tasa de error [3].

Para realizar el paso de modulación codificada nos basaremos en la aplicación de Bernoulli.

$$x_{n+1} = f(x_n) = \begin{cases} 2x_n & si \quad x_n < 0.5 \\ 2x_n - 1 & si \quad x_n \geq 0.5 \end{cases}$$

Se definirá el estado simbólico del modulador como:

$$r = \sum_{n=1}^{N} b_n 2^{-n}$$

Si tomamos como condición inicial $x_0=r$, la aplicación de Bernoulli permite enviar codificada la secuencia de bits en una secuencia caótica x_n.

En un sistema real donde N es la longitud del mensaje, N puede tener un valor de miles de bits, y el proceso de modulación se vuelve impracticable, ya que requeriría una precisión infinita. En este caso se han codificado bloques de Q bits (Q<N) según la siguiente relación:

$$r'_n = \sum_{m=n}^{n+Q-1} b_m 2^{-m+n-1}$$

$$x'_n = \exp(2 \cdot \pi \cdot j \cdot g(r'_n))$$

Siendo r'_n el nuevo estado simbólico del sistema, que representa una cuantificación de r_n con Q bits. Se demuestra que, si Q>4, el resultado varía muy poco con respecto al caso con Q arbitrariamente grande [1] [2] [3]. Por otro lado, *g(r)* es una función que proyecta el intervalo [0,1] en sí mismo, parte del punto (0,0) y llega al (1,1), y ha de ser estrictamente no decreciente y continua [3]. Dicha función es la que se va a parametrizar a través de un conjunto de puntos interpolados mediante *splines* a fin de conseguir una familia completa de modulaciones caóticas que extienden el *Bernoulli shift map*.

4.1 Objetivo

El objetivo de este proyecto es alcanzar una optimización respecto de la cota de error asociada al modulador, para ello trabajaremos con la estimación de la probabilidad de error binaria.

A estos fines se trabajará con el algoritmo genético indicado anteriormente, estudiando los patrones de error típico y la evolución de cota de la probabilidad de error en función de la relación señal ruido.

Para averiguar los patrones de error típicos, se comparará la señal de entrada al modulador; indicada como b_n en el diagrama anterior; con la señal obtenida a la salida del demodulador; indicada como b'_n; para los casos de error más probables.

Ambas señales se comparan típicamente con la operación lógica XOR, la cual comparará la secuencia de bits de cada señal y mostrará un 1 en caso de que los bits comparados sean distintos y un 0 en caso de que los bits sean iguales, creando de esta forma un nuevo vector sobre el que trabajaremos, nombraremos este nuevo vector como *error* para referirnos a él en próximas referencias.

Para dicho vector *error* se ha estudiado su funcionamiento típico que corresponde a la existencia de uno o dos errores consecutivos, representado por un 1 cada uno de ellos en la forma (100…00) y (1100…00), respectivamente.

Se modularán las señales b_n y b'_n para obtener sus secuencias caóticas asociadas, siendo estas x_n y x'_n respectivamente, que serán iguales hasta que se encuentre un 1 en el vector *error*; es decir, hasta que se produce un error en la comparación debido a una decodificación errónea; de esta forma se considera que el primer error se encuentra en la posición *J*, y que en una posición *L* volverán a coincidir los valores de la secuencia caótica, finalizando el suceso de error completo, y de esta manera podremos obtener la distancia euclídea asociada al vector error típico en el canal como:

$$d^2 = \sum_{K=J}^{J+L-1} |x_K - x'_K|^2$$

De esta forma podremos evaluar la cota para un canal con ruido blanco gaussiano aditivo mediante la siguiente formula:

$$Pb = \sum_i w_i \cdot erfc(\sqrt{d^2 \cdot \frac{Eb}{N0} \cdot \frac{1}{4}})$$

Teniendo en cuenta que w_i es un factor de ponderación que depende del tipo de suceso de error, *Eb/N0* es la relación señal a ruido y d^2 es la distancia euclídea mencionada anteriormente. *erfc(x)* es la función de error complementario. La suma se extiende a todos los sucesos de error posibles del tipo mencionado [1] [2] [3].

Dado que es la que nos dará la evolución final de la cota, a esta función se la denomina función de coste, que es con la que se tendrá que trabajar durante el proceso de optimización.

Capítulo 5

5 Optimización del sistema

5. Optimización del sistema

Para realizar este programa se ha trabajado con lenguaje de programación C en el entorno de desarrollo integrado Dev-C++. Se ha escogido este lenguaje ya que es un lenguaje orientado a la implementación de sistemas operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje débilmente tipificado de nivel alto pero con muchas características de nivel bajo. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

En primer lugar antes de realizar el programa que unificará el algoritmo genético o evolutivo con la modulación caótica, se deberá instalar la librería GSL, destinada a cálculos numéricos en matemáticas y ciencia, e incorpora, entre otras, rutinas para el manejo de números complejos, funciones elementales y funciones especiales, combinatoria, álgebra lineal, integración y derivación numéricas, transformada rápida de Fourier, transformada *wavelet* discreta, generación de números aleatorios y estadística.

Cabe destacar que el programa que ha sido desarrollado para este proyecto también es compatible con el sistema operativo Linux

El conjunto del programa consta de 5 ficheros, 1 de cabecera, 3 relacionados con el algoritmo genético y otro último relacionado con las modulaciones caóticas, cada uno de ellos se realizará el siguiente cometido:

Fichero **main.c**

Es el encabezamiento del programa, además de iniciarlo y solicitar la representación del error de la cota final y las muestras obtenidas, se encarga de realizar la medida del tiempo de la operación y realiza la inicialización pseudoaleatoria de forma que no se obtenga el mismo resultado para cada ejecución con los mismos parámetros.

Fichero **alea.h**

Cuenta con dos funciones, la primera obtiene números aleatorios ente el -1 y el 1, mientras que la segunda los obtiene entre 0 y 1.

Fichero **dependiente.h**

En este fichero es donde se asigna el valor de la mayoría de los parámetros y variables que se va a utilizar a lo largo del programa, destacando el número de poblaciones sobre las que se va a trabajar y cuántos componentes tiene cada una de estas, siendo inicializados aleatoriamente estos valores en función del tipo de datos ya sean números binarios, enteros o reales, aunque preferiblemente conviene trabajar con estos últimos; también se indicará la probabilidad de que un miembro de la población sufra una mutación y su valor.

Finalmente realizará una función de evaluación donde se asignarán los valores a cada miembro de la población para su envío a la función de evaluación de cotas de la modulación caótica y conseguir la obtención de una primera tasa de error; la primera cota de error y las muestras obtenidas se enviarán al fichero del algoritmo genético como referencia para iniciar la optimización.

Fichero **AG.h**

Este fichero ejecuta el algoritmo genético o evolutivo, siendo el resultado más exacto si se realizan más iteraciones, este algoritmo irá comprobando para cada muestra si el resultado es mejor o peor que el anterior, de esta forma si el resultado es mejor lo mantendrá borrando el resultado anterior, mientras que si se trata de un resultado peor lo borrará manteniendo el resultado anterior.

Fichero **cotas.h**

La finalidad de este fichero es recrear una modulación caótica para su posterior estudio y filtrado, en este fichero se obtendrá la cota de la probabilidad de error binaria, según la fórmula dada anteriormente que se define como el número estimado de bits incorrectamente recibidos, con respecto al total de bits o bloques enviados durante un intervalo especificado de tiempo.

Finalmente realizara una función de representación que tomará los valores finales de las muestras y las cotas totales y las guardará en un archivo para su posterior análisis o visualización.

En un principio el programa se diseñó para representar *g(x)* mediante su discretización con un número fijo de muestras, tres en este caso, para realizar la simulación y la posterior variación de parámetros para ver su repercusión y darles el valor optimo para una mejor respuesta, pero una vez que se realizaron todos los estudios pertinentes se propuso incrementar el número de muestras y valorar la evolución del resultado en función de éstas y si la estimación de la utilidad de los parámetros seguía siendo la misma o en qué proporción variaba.

Para esto se modificó el programa incorporándole un bucle dinámico, al indicarle el número de muestras que se requerían las producía directamente sin tener que estar modificando cada fichero del programa.

Se impuso como condición para obtener las muestras que estas tenían que estar en prefecto orden ascendente, ninguna de ellas podía superar a la posterior, en todo caso igualar su valor, teniendo que estar comprendidos en el intervalo [0,1].

A lo largo del desarrollo del programa los principales problemas que se han encontrado han sido:

- Al trabajar con Windows y Linux en uno de los sistemas operativos el ejecutable avisaba de que se estaba accediendo a una posición no declarada, mientras en el otro, no; esto producía que se trabajase con el valor de esta posición de memoria que es desconocido y ajeno al programa.

- El dimensionamiento de los valores de los vectores de población es distinto en el fichero dependiente y en el fichero AG, lo que puede dar a error al enviar los resultados obtenidos de uno a otro.

- Cuando se realizaron las simulaciones, a medida que el numero de muestras se incrementaba, el resultado se hacía mucho más pequeño de lo esperado, debido a la existencia de tipos de eventos de error no considerados en la modulación, que se comenzaron a tener en cuenta para futuros estudios; este aspecto se comentará de manera mas profunda en el apartado de resultados de la simulación.

- Hay que tener muy claro cuál es el valor de la población que mejor aproxima la solución y dónde se encuentra para guardarlo correctamente en el archivo y poder seguir trabajando con él.

- Definir correctamente la función de la modulación caótica para la obtención de la cota de error ya que su desarrollo es de los apartados más complejos del proyecto.

- Al realizar la misma simulación con los mismos parámetros una y otra vez el resultado era siempre el mismo, sin dar lugar a una minima variación, sin embargo estos resultados deberían ser distintos para constatar el proceso estadístico de la optimización, por lo que se añadió una función que variase los valores iniciales de la población aleatoriamente.

Una vez resueltos todos los problemas y con el programa acabado se procedió a la obtención y representación de resultados.

Se ha empleado MATLAB para realizar las gráficas de la representación de la probabilidad de error en función de la relación señal ruido, para esto se usará la función *semilogy* que se encargará de colocar las cotas obtenidas en el archivo *cotatotal* de forma semilogarítmica sobre la gráfica anterior.

Con esta representación del *bit error rate* se podrá apreciar la evolución de la modulación y poder comparar sobre la misma gráfica los valores obtenidos a través de la simulación para cada juego de parámetros con respecto a los valores reales, considerando nuestra tasa de error como la variación de uno o dos bits a lo largo de la modulación en estudio estimados en la función de cuantificación.

También se representara cada función g(x) obtenida de las simulaciones del programa, puesto que MATLAB es útil para representaciones lineales o polinómicas, pero no interpola los valores, para esto se ha desarrollado una nuevo fichero.

Fichero **Funcionxy.c**

Este fichero interpola los valores a lo largo de las muestras, tomando el archivo *datosfinales* y el número de muestras, dando 200 valores interpolados dentro de la trayectoria tomada entre [0,1] en el archivo *ficherosalida.*

De esta forma se podrá apreciar su tendencia más claramente.

Estos son los dos tipos de graficas se mostrarán en el apartado de *estudio de casos* de este proyecto.

Capítulo 6

6 Estudio de casos

Parámetros

Comprobación para 5 muestras

Comprobación para 11 muestras

Consideración del error

Valoración de los resultados

6. Estudio de casos

Con la idea de realizar las simulaciones para obtener el resultado óptimo se procederá en primer lugar a comprobar cómo afectan los parámetros más destacables del programa.

Para ello mantendremos fijos el resto de parámetros y daremos diferentes valores al parámetro seleccionado para estudiar sus efectos sobre el resultado del error y los valores de las muestras del vector Y que, mediante su interpolación con *splines*, permiten representar la función *g(x)*.

Una vez realizado el estudio de la utilidad y la medida en que es más favorable cada uno de los parámetros de manera independiente, se realizará una simulación variando todos aquellos que puedan permitir una mejora adicional en el resultado, asignado a estos parámetros los valores que mejor puedan hacer que el programa sea mucho más fiable en su resultados.

Se tomarán en un comienzo los siguientes valores en los parámetros: factor de cuantificación igual a 4, una relación señal ruido; representada como eb/no; de referencia de 8 dBs, 150 iteraciones del algoritmo genético para un estudio de 200 poblaciones con 1100 valores por población y una probabilidad de mutación del 1% por población con un porcentaje de mutación sobre el elemento mutado del 1%. De esta manera en cada simulación solo variaremos una de las medidas según se indique sin variar el resto.

Se asignará al vector X el número de muestras que se requieran en el programa, estando estas fijas durante toda la ejecución y ordenadas ascendentemente y equiespaciadas en el intervalo de 0 a 1.

El vector Y mantendrá constantes las muestras superior e inferior (0 y 1), mientras que el resto las obtendrá el programa teniendo en cuenta que deberán encontrase en orden estrictamente no decreciente.

6.1 Parámetros

Los parámetros de referencia sobre los que realizaremos este estudio son los siguientes:

Factor de cuantificación

El factor de cuantificación indica el número de bits que se usan para generar las muestras caóticas, de esta manera, se puede asociar la cuantificación a la precisión, no obstante aumentar este parámetro para conseguir una mayor precisión repercute muy negativamente en el tiempo que necesita el programa para realizar la simulación, sin añadir ninguna ventaja apreciable.

Relación señal ruido

Es el principal punto de referencia para calcular el error del sistema, se mide en decibelios y se define como el margen que hay entre la potencia de la señal que se transmite y la potencia del ruido que la corrompe.

Población

La población es cada grupo de individuos o valores sobre el cual se va a realizar el estudio, se toma la mejor solución del que se ha obtenido del conjunto de grupos, por lo que es útil tener un amplio número de poblaciones, pero esto también se traduce en mucho más datos y operaciones que se deben realizar y conlleva una mayor demora.

Cantidad de valores *double*

Se corresponde con el número de miembros o valores que se estudian dentro de cada población y que son generados aleatoriamente, estos valores son evaluados eligiéndose los que proporcionan una mejor solución para el problema tratado, desechando el resto, por lo que cuanto más haya, mayor será probabilidad de que el algoritmo obtenga una solución más eficaz.

Número de iteraciones del algoritmo genético

El número de iteraciones del algoritmo genético es el número de veces que optimizamos la solución del programa dentro del algoritmo genético o evolutivo; cuantas más veces se realice el bucle el resultado se hará más preciso.

Probabilidad de mutación

Se corresponde con la cantidad de valores que va a sufrir una determinada variación dentro de cada población.

Porcentaje de mutación

Se refiere a la cuantía en que cambian los valores que van ha sufrir una mutación dentro de cada población.

Número de muestras

Se corresponde con el número de valores con que se caracteriza el vector Y, teniendo en cuenta que dos de ellos son fijos, el primero que se fija al 0 y el último que se le asignara el valor 1, entre estos se ubicarán los demás teniendo en cuenta que se colocarán de manera ascendente pudiendo varios valores ser iguales entre sí.

Estos valores intermedios se mostrarán por pantalla y se utilizará el vector final para definir una función de transformación *g(x)* que permita tener una tasa de error mínima en las transmisiones de un sistema caótico del tipo descrito.

6.2 Comprobación para 5 muestras:

En este apartado para evitar un exceso de información **redundante** se indicarán los datos de la **medida más relevante** para cada uno de los parámetros de la simulación, si la incidencia del parámetro sobre la simulación es notable se mostrarán todas la medidas realizadas.

En el apartado de **apéndice** se han indicado con todos los detalles los resultados de todas las medidas para cada uno de los parámetros.

6.2.1 Variación del factor de cuantificación

Se fijarán los valores de población a 200, el número de valores *double* a 1100, la relación señal ruido a 8 dB, se realizarán 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación será del 1%

De esta forma se darán valores al factor de cuantificación entre 3 y 5 indicando los resultados de las muestras obtenidos, así como también la estimación del error y mostrándose en una gráfica en función de la relación señal a ruido para cada caso.

Q=3

Vector Y

[0.000000, 0.324435, 0.569934, 0.884134, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000018094

Q=5

Vector Y

[0.000000, 0.244414, 0.512677, 0.797241, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000029555

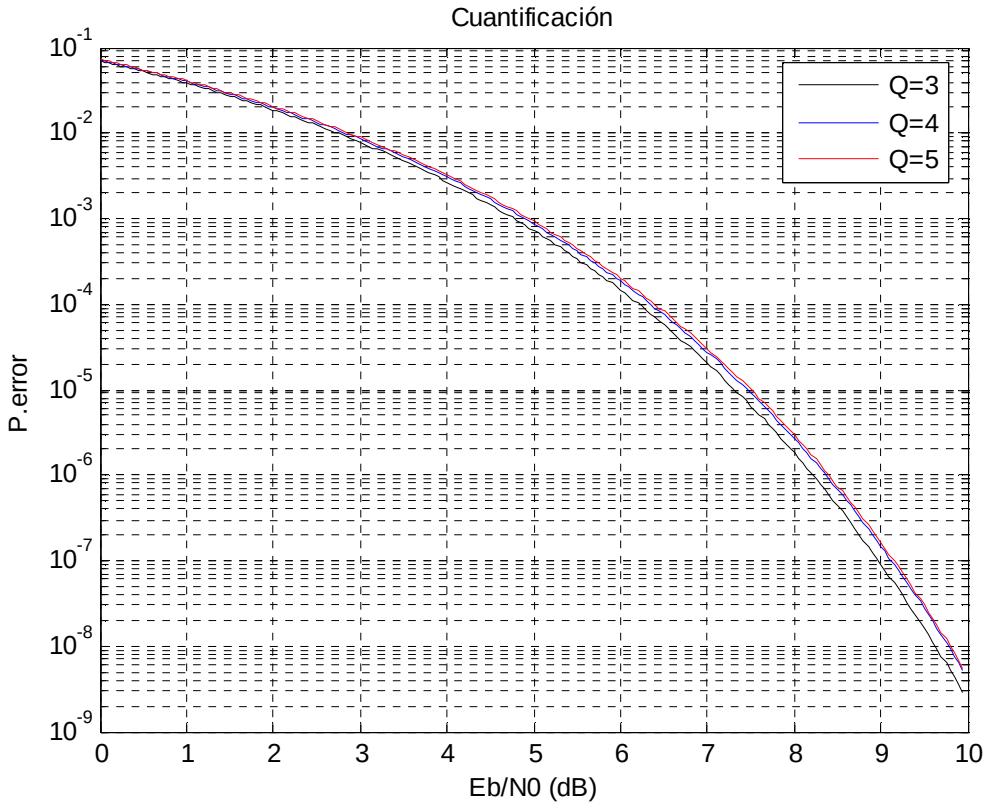

Fig. 1 Cota de la probabilidad de error respecto a la función de cuantificación, 5 muestras

6.2.2 **Variación de la relación señal ruido**

Se ha fijado el factor de cuantificación a 4, realizando 150 iteraciones en el algoritmo genético, con 1100 valores *double* y una probabilidad y porcentaje de mutación del 1% para 200 poblaciones.

Se van a dar valores a la relación señal ruido entre 3 y 9dB.

Eb/no=3 dB

Vector Y

[0.000000, 0.292895, 0.552002, 0.854847, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0084531308

Eb/no=6 dB

Vector Y

[0.000000, 0.284636, 0.545259, 0.852765, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0001826911

Eb/no=9 dB

Vector Y

[0.000000, 0.268582, 0.528110, 0.818325, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000001479

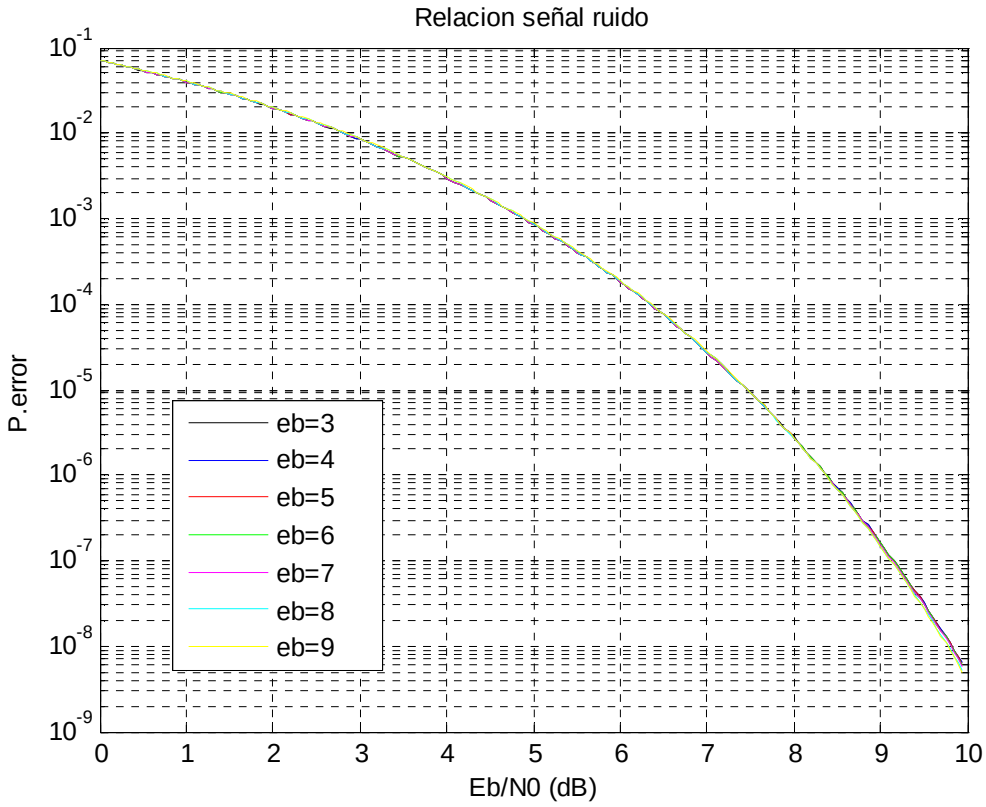

Fig. 2 Cota de la probabilidad de error respecto a la relación señal ruido, 5 muestras

6.2.3 **Variación de la población**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, realizando 150 iteraciones, con 1100 valores *double* y una probabilidad y porcentaje de mutación del 1% dando diversos valores al parámetro población.

Realizaremos medidas para 10, 50, 100 y 200 poblaciones.

TPOB=100

Vector Y

[0.000000, 0.275942, 0.533569, 0.833513, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026763

TPOB=10

Vector Y

[0.000000, 0.233707, 0.489082, 0.776686, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000031228

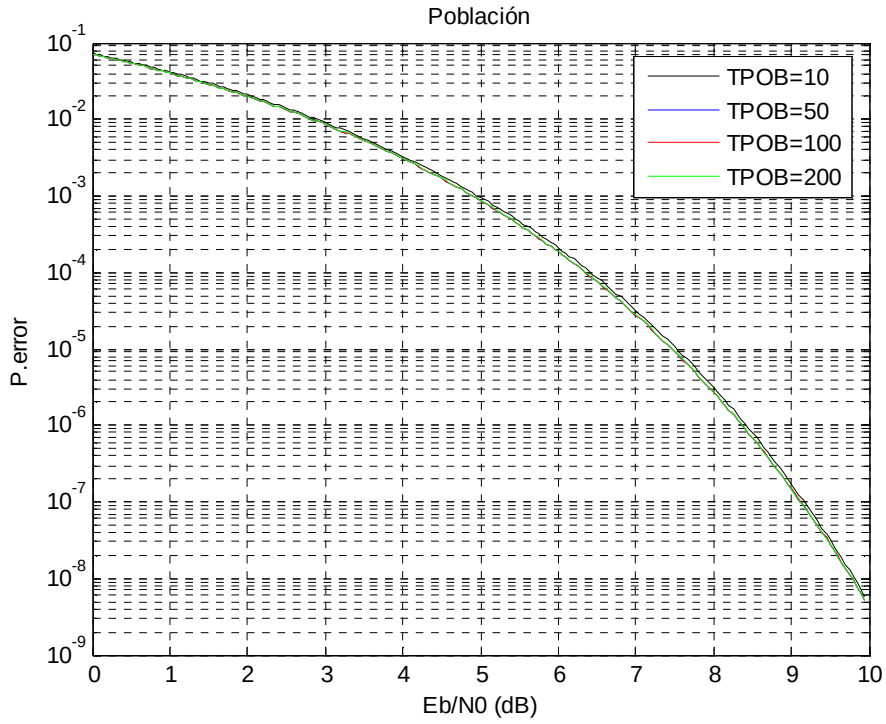

Fig. 3 Cota de la probabilidad de error respecto a la población, 5 muestras

6.2.4 **Variación de números *double***

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, realizando 150 iteraciones, para una probabilidad y porcentaje de mutación del 1% para un valor de población de 200, dando diversos valores al parámetro *numeros_double*

Las simulaciones se realizaran para 11, 110 y 1100 valores *double*.

numeros_double=11

Vector Y

[0.000000, 0.276284, 0.533785, 0.835283, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026764

numeros_double=110

Vector Y

[0.000000, 0.276415, 0.533713, 0.834282, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026763

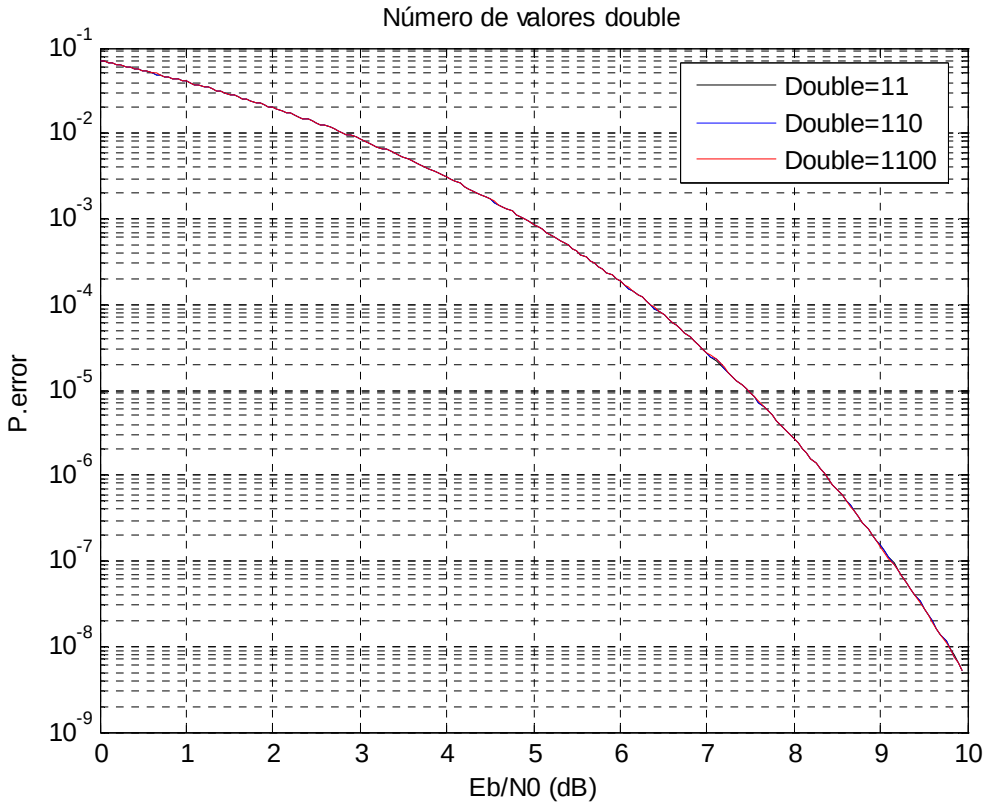

Fig. 4 Cota de la probabilidad de error respecto a los valores *double*, 5 muestras

6.2.5 **Variación del número de iteraciones en el algoritmo genético**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, introduciendo 1100 valores *double* y una probabilidad y porcentaje de mutación del 1% para un valor de población de 200, mientras se variaba el número de iteraciones en el algoritmo evolutivo.

Se realizaran 10, 50, 150 y 300 iteraciones en cada simulación.

Iteraciones=10

Vector Y

[0.000000, 0.313521, 0.595063, 0.861248, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000040506

Iteraciones=300

Vector Y

[0.000000, 0.275364, 0.532452, 0.832498, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026763

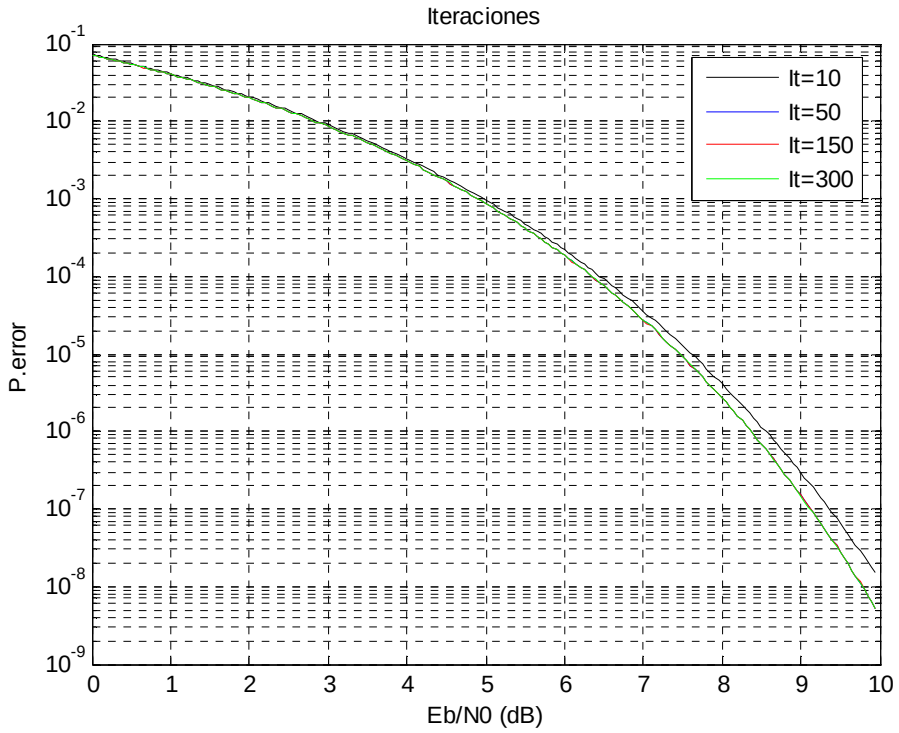

Fig. 5 Cota de la probabilidad de error respecto al número de iteraciones, 5 muestras

6.2.6 **Variación de la probabilidad de mutación**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, para 150 iteraciones del algoritmo genético, introduciendo 1100 valores *double* con un porcentaje de mutación del 1% para un valor de población de 200, mientras se daban diversos valores a la probabilidad de mutación.

Las medidas de probabilidad de mutación serán para el 10%, 40%, 65% y 100%

Probabilidad =40%

Vector Y

[0.000000, 0.275277, 0.533180, 0.833264, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026761

Probabilidad =100%

Vector Y

[0.000000, 0.274937, 0.532158, 0.832169, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026762

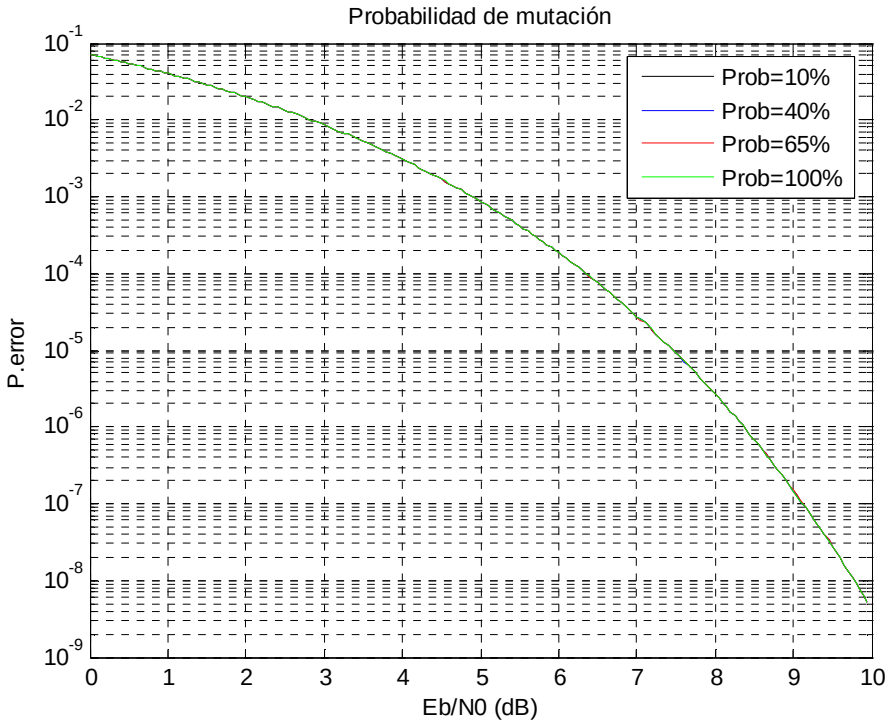

Fig. 6 BER respecto a la probabilidad de mutación, 5 muestras

6.2.7 **Variación del porcentaje de mutación**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, para 150 iteraciones del algoritmo genético, introduciendo 1100 valores *double* con una probabilidad de mutación del 100% para un valor de población de 200 modificando el porcentaje de mutación para dicha probabilidad.

Las medidas del porcentaje de mutación serán para una variación del 10%, 40%, 65% y 100%

Porcentaje=40%

Vector Y

[0.000000, 0.277287, 0.534828, 0.836343, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026770

Porcentaje=100%

Vector Y

[0.000000, 0.270054, 0.530499, 0.829289, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026787

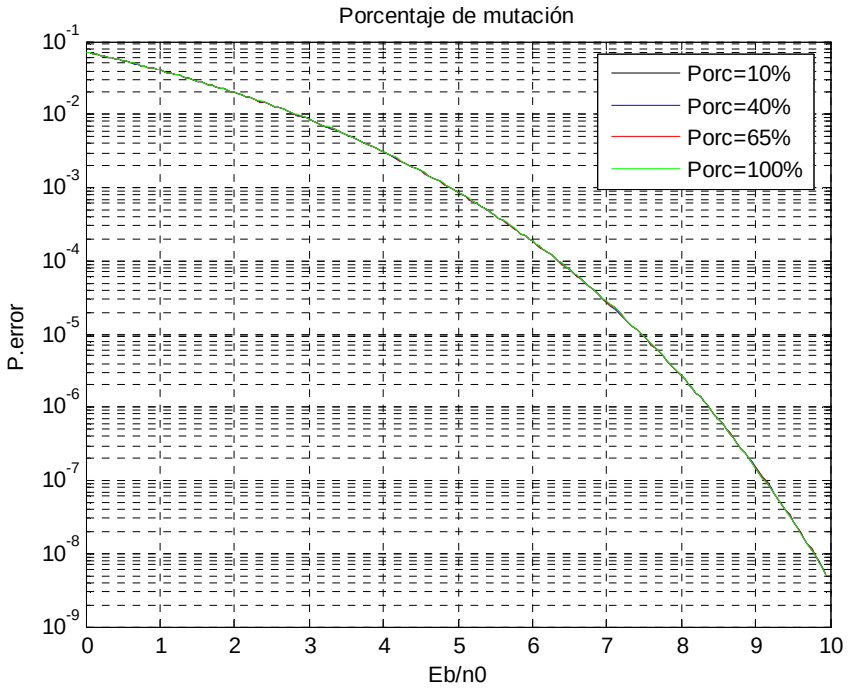

Fig. 7 Cota de la probabilidad de error respecto al porcentaje de mutación, 5 muestras

6.2.8 **Valor óptimo**

Puesto que estamos tomando pocos valores del vector Y, apreciamos que los resultados son bastante constantes tanto en la cota de error final como en la obtención de los valores del vector, sin embargo hay dos parámetros que parecen afectar de forma más relevante al resultado.

Estos parámetros son la población y las iteraciones del algoritmo genético, teniendo esto en cuenta vamos a realizar una simulación optimizando todo lo posible los parámetros, de esta forma se elegirán:

300 iteraciones, el factor de cuantificación a 4, relación señal ruido de 8 dB, 1100 valores *double* y tanto el porcentaje como la probabilidad de mutación al 40%, para 200 poblaciones.

Vector Y

[0.000000, 0.275106, 0.533334, 0.833297, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026761

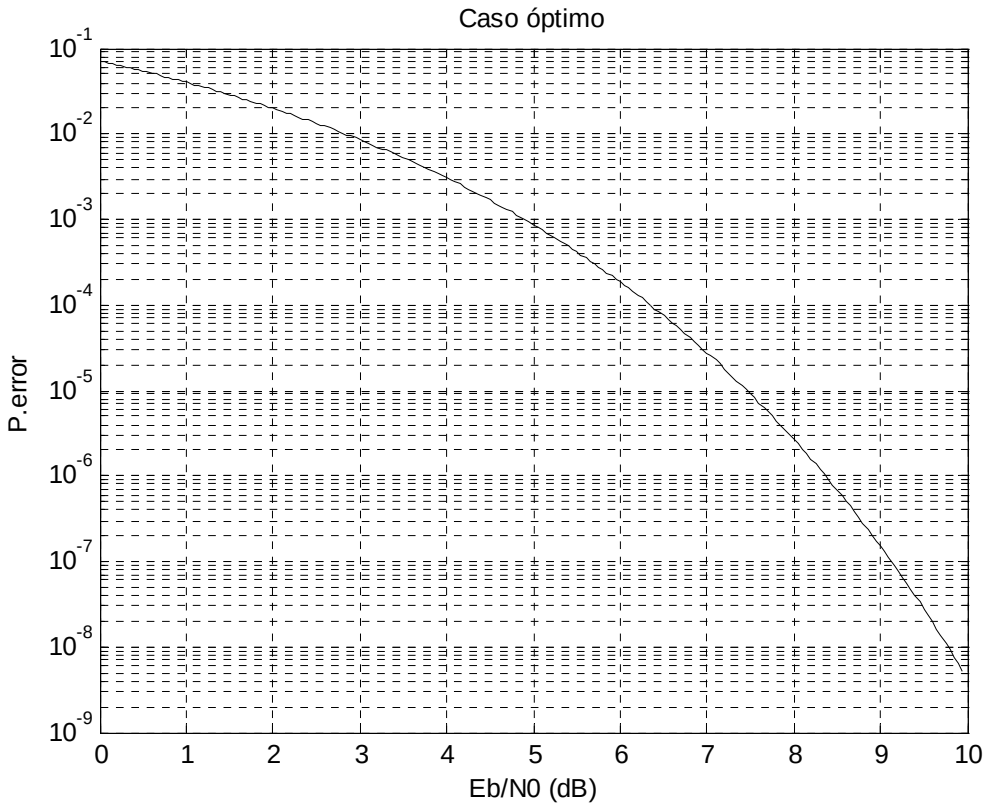

Fig. 8 Cota de la probabilidad de error óptimo, 5 muestras

Pese a haber elegido los mejores valores de los parámetros, la cota de error final es bastante aproximada a las obtenidas anteriormente, por lo que se realizarán las mismas comprobaciones para otras discretizaciones de la función *g(x)*, con más puntos y, por tanto, más grados de libertad. De esta forma, se podrá verificar la influencia de cada parámetro y la posibilidad de lograr o no una mayor optimización.

Dado que los valores de los vectores X e Y son prácticamente iguales para todos los casos anteriores, solo se realizara una representación de los mismos con los valores de este ultimo caso.

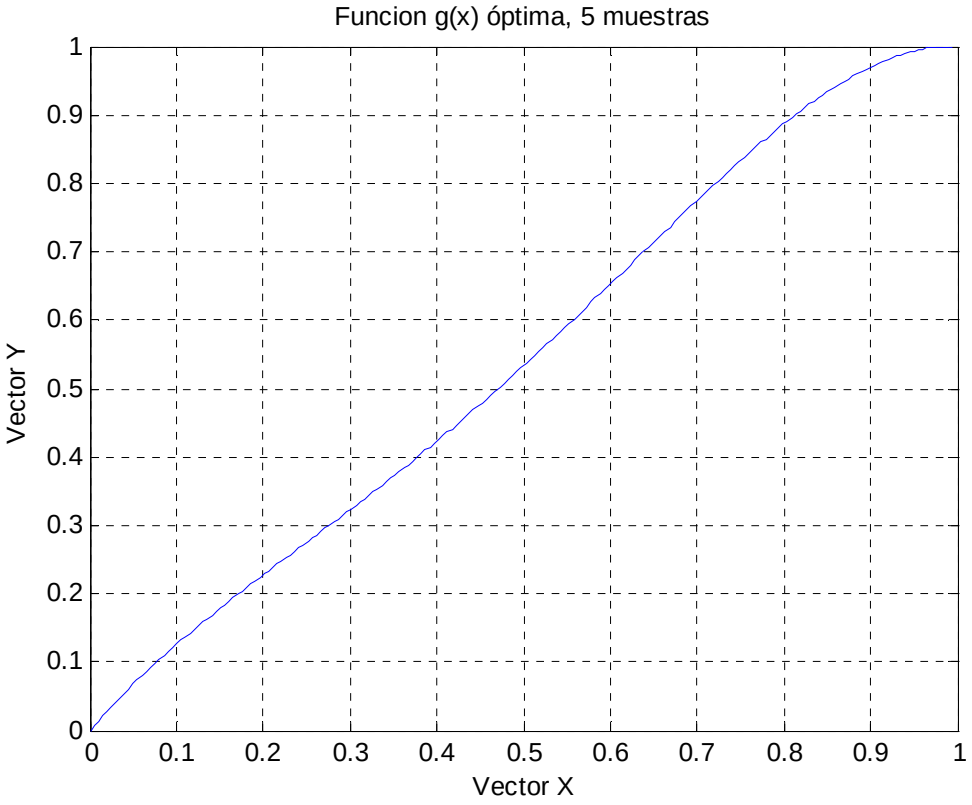

Fig. 9 Función *g(x)* óptima, 5 muestras

6.3 Comprobación para 11 muestras:

En este apartado para evitar un exceso de información **redundante** se indicarán los datos de la **medida más relevante** de cada uno de los parámetros de la simulación, si la incidencia del parámetro sobre la simulación es notable se mostrarán todas la medidas realizadas.

En el apartado de **apéndice** se han recogido con todo detalle los resultados de todas las medidas para cada uno de los parámetros.

A diferencia de las pruebas con la discretización de *g(x)* con 5 muestras los valores del vector Y son muy diferentes unos de otros por lo que se acompañará cada simulación de una representación entre los valores del vector X e Y obtenidos, de esta forma será mucho más sencillo interpretar los resultados y la tendencia de sus valores.

6.3.1 Variación del factor de cuantificación

Se fijará el valor de población a 200, el número de *double* a 1100, el ruido a 8 dB, se realizaran 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación será del 1%.

De esta forma se darán valores al factor de cuantificación entre 3 y 6, y se incorporarán las representaciones de los vectores usando el tipo de interpolación empleado en la optimización y una gráfica con el valor de la cota de error en función de la relación señal ruido para cada conjunto de valores que se haya asignado al parámetro.

Q=3

Vector Y

[0.000000, 0.005671, 0.013059, 0.014603, 0.327950, 0.327950, 0.674588, 0.677857, 0.750224, 0.778452, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000018264

De esta forma obtendremos la siguiente representación gráfica:

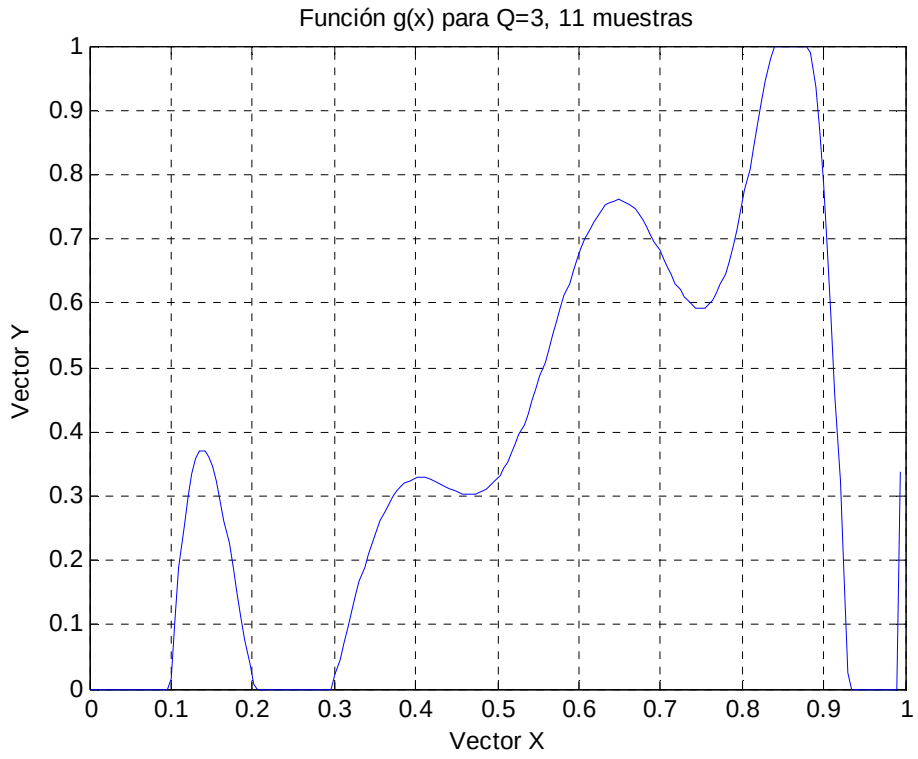

Fig. 10 Función *g(x)* para Q=3, 11 muestras

Q=6

Vector Y

[0.000000, 0.076102, 0.106779, 0.106779, 0.357823, 0.398607, 0.554437, 0.624793, 0.624793, 1.000000, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000044372

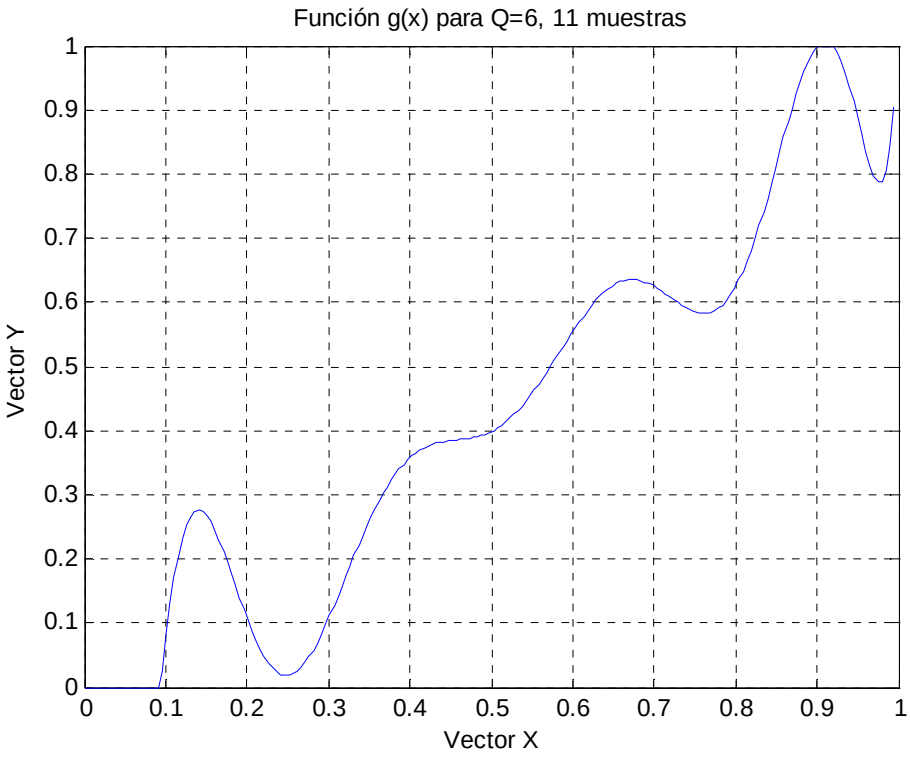

Fig. 11 Función *g(x)* para Q=6, 11 muestras

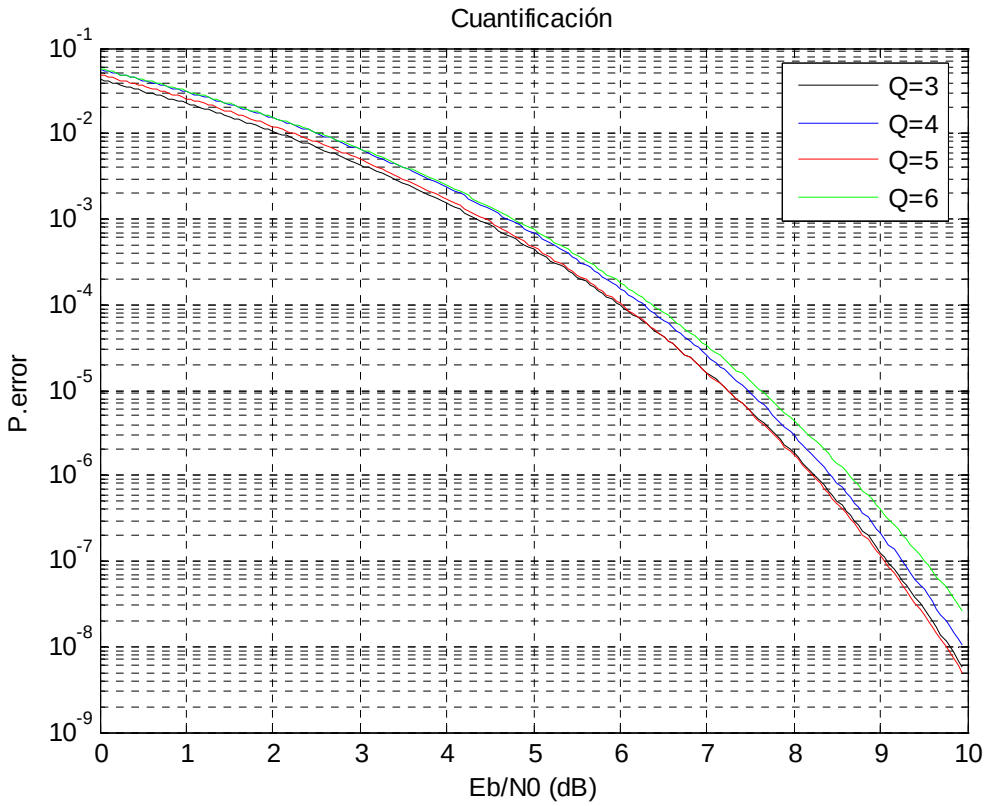

Fig. 12 Cota de la probabilidad de error respecto a la función de cuantificación, 11 muestras

6.3.2 **Variación de la relación señal ruido**

Se fijará el valor de población a 200, el número de valores *double* a 1100, el factor de cuantificación a 4, se realizarán 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación serán del 1%.

Se van a dar valores a la relación señal ruido de 3 a 9 dB.

Eb/no=3 dB

Vector Y

[0.000000, 0.007640, 0.008883, 0.008909, 0.340457, 0.340457, 0.616596, 0.617684, 0.617684, 0.712644, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0060541435

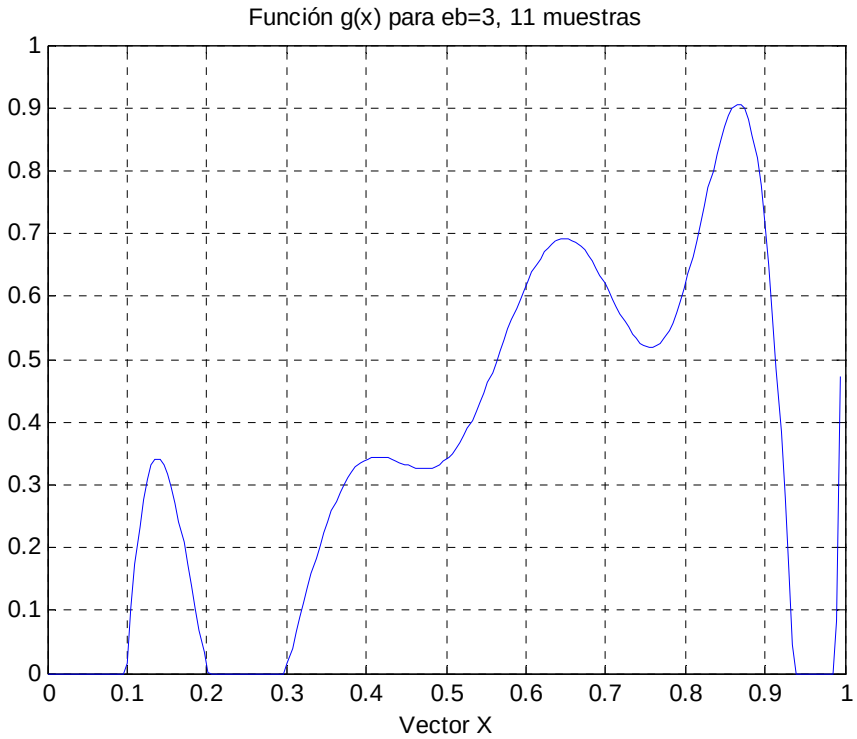

Fig. 13 Función *g(x)* para eb=3, 11 muestras

Eb/no=6 dB

Vector Y

[0.000000, 0.087463, 0.122214, 0.147427, 0.147427, 0.361764, 0.368256, 0.631080, 0.634220, 0.980571, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0001880654

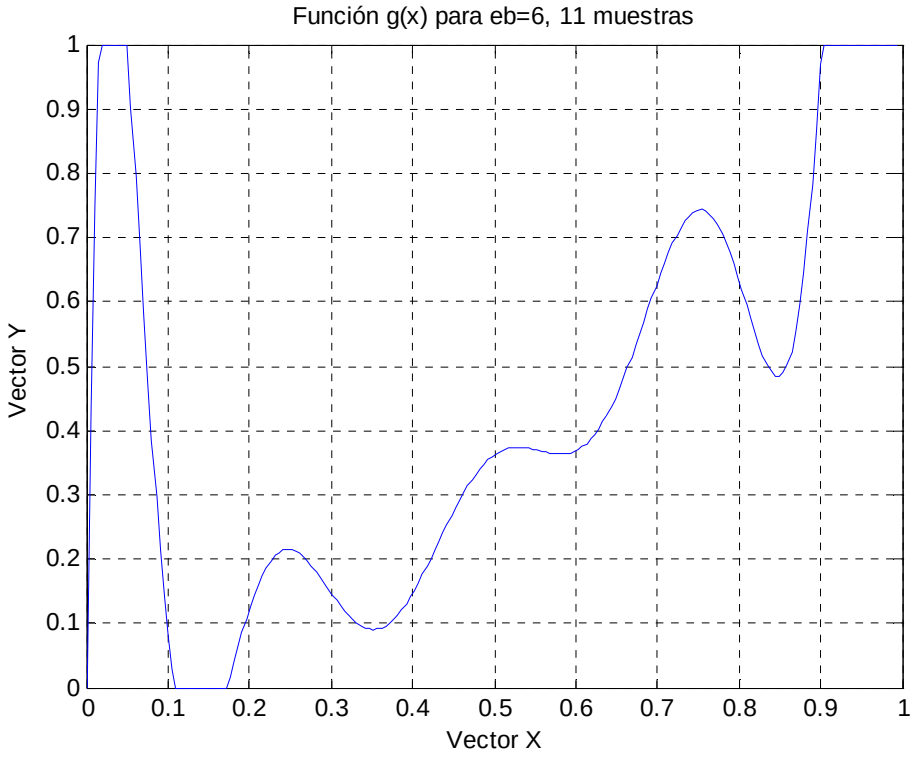

Fig. 14 Función *g(x)* para eb=6, 11 muestras

Eb/no=9 dB

Vector Y

[0.000000, 0.041474, 0.062989, 0.098876, 0.098876, 0.352002, 0.365503, 0.590145, 0.591029, 0.895972, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000007721

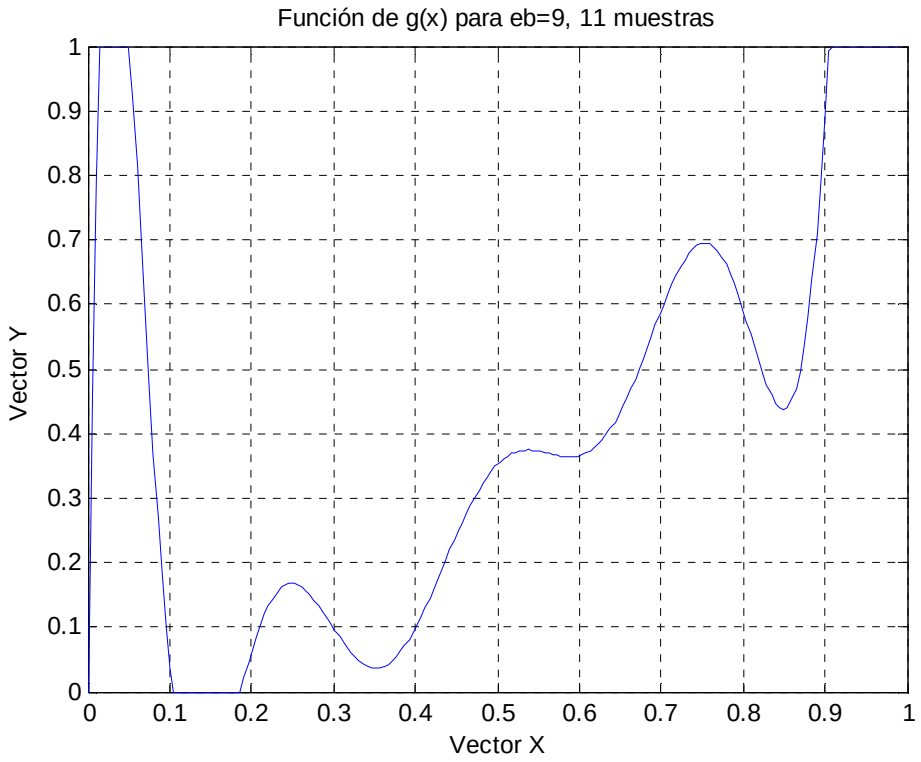

Fig. 15 Función *g(x)* para eb=9, 11 muestras

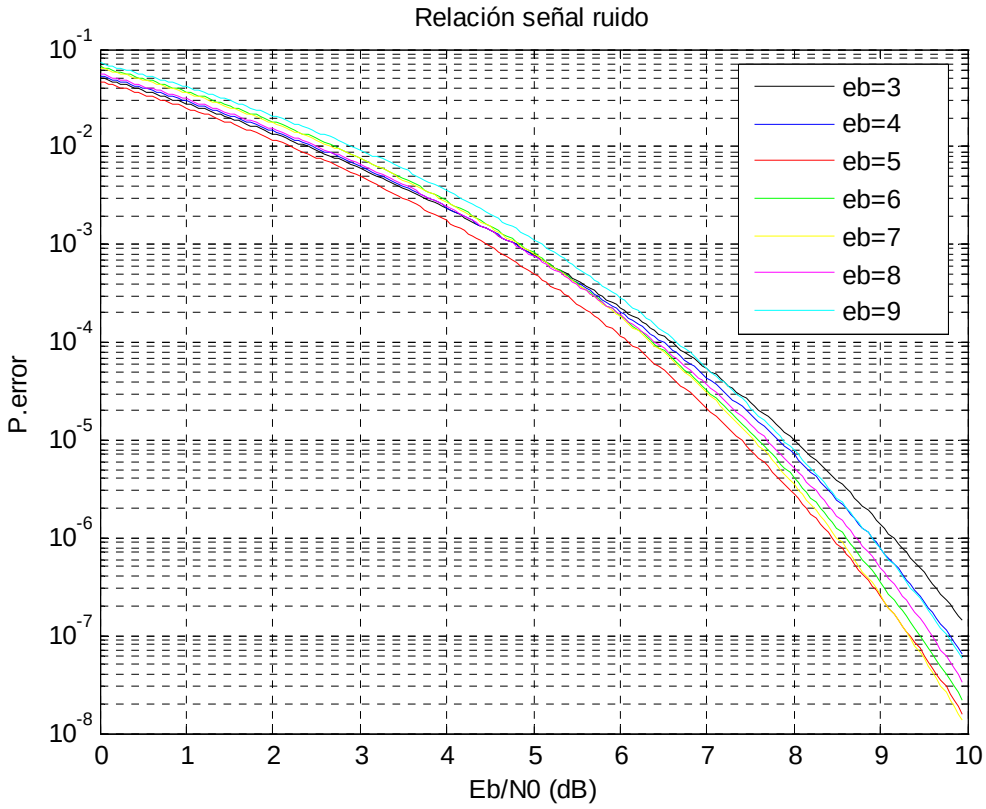

Fig. 16 Cota de la probabilidad de error respecto a la relación señal ruido, 11 muestras

6.3.3 **Variación la población**

Se fijará el valor del factor de cuantificación a 4, el número de valores *double* a 1100, la relación señal ruido a 8 dB, se realizarán 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación serán del 1%.

Realizaremos medidas para 10, 50, 100 y 200 poblaciones.

TPOB =200

Vector Y

[0.000000, 0.112359, 0.112359, 0.112359, 0.315153, 0.316594, 0.599053, 0.599053, 0.650854, 0.656655, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000038635

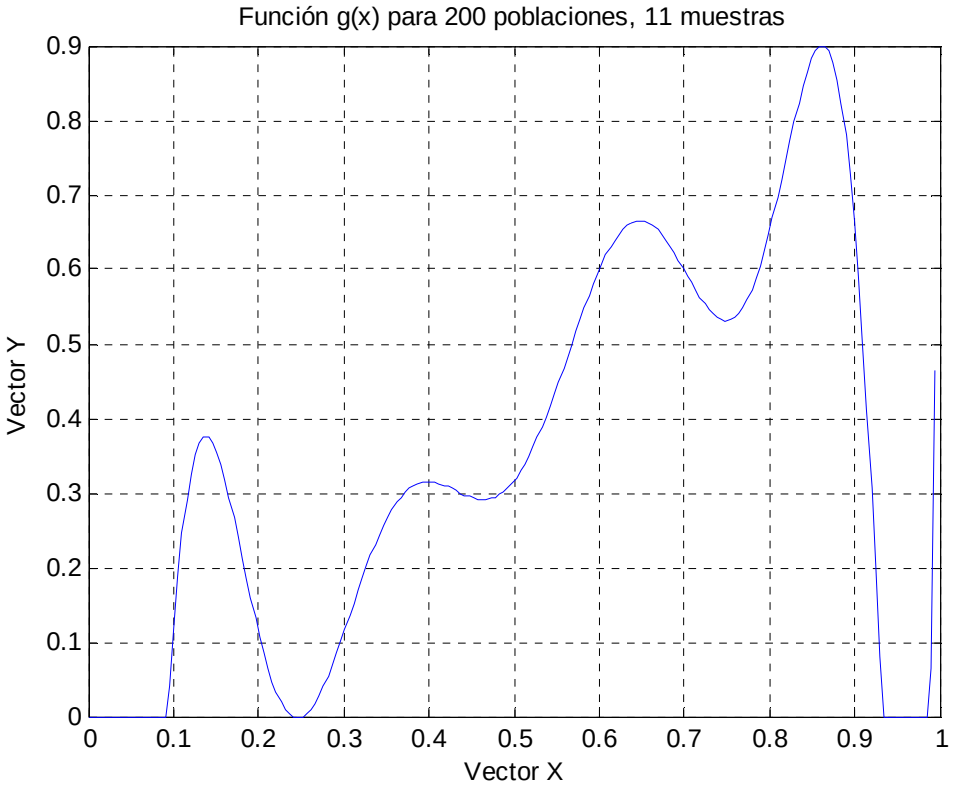

Fig. 17 Función *g(x)* para 200 poblaciones, 11 muestras

TPOB=100

Vector Y

[0.000000, 0.032103, 0.041895, 0.042026, 0.316597, 0.316597, 0.594872, 0.594872, 0.609030, 0.623181, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000081347

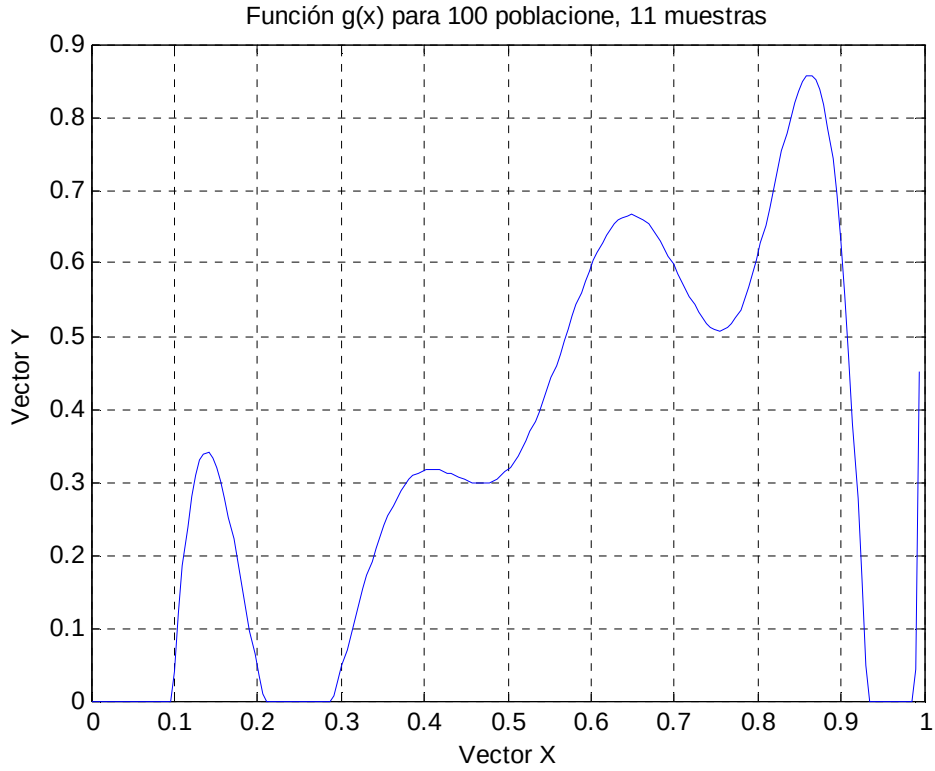

Fig. 18 Función *g(x)* para 100 poblaciones, 11 muestras

TPOB=50

Vector Y

[0.000000, 0.065551, 0.070940, 0.070940, 0.237776, 0.237776, 0.425597, 0.425597, 0.468665, 0.498910, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000716813

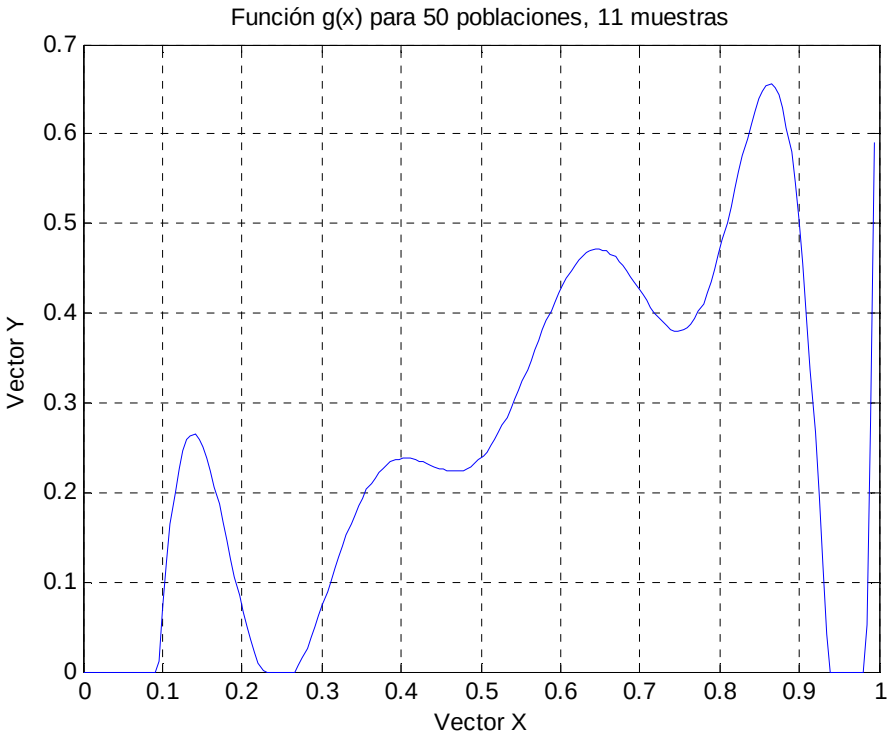

Fig. 19 Función *g(x)* para 50 poblaciones, 11 muestras

TPOB=10

Vector Y

[0.000000, 0.004400, 0.005976, 0.005982, 0.028087, 0.028087, 0.030193, 0.307741, 0.322091, 0.726028, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0274905063

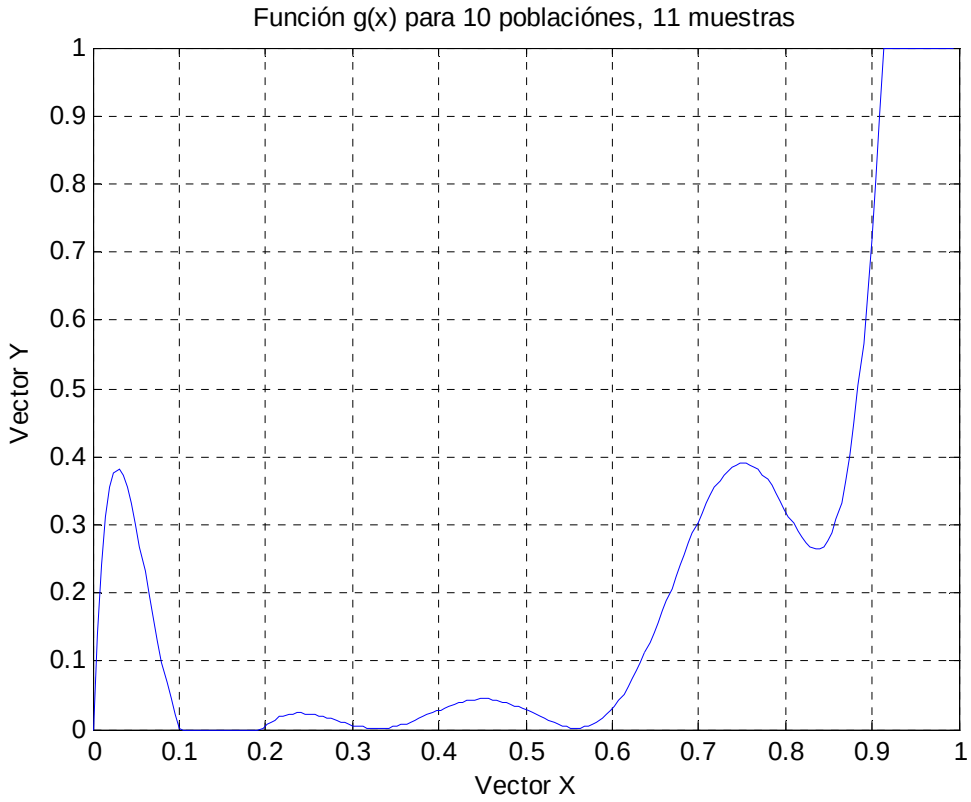

Fig. 20 Función *g(x)* para 10 poblaciones, 11 muestras

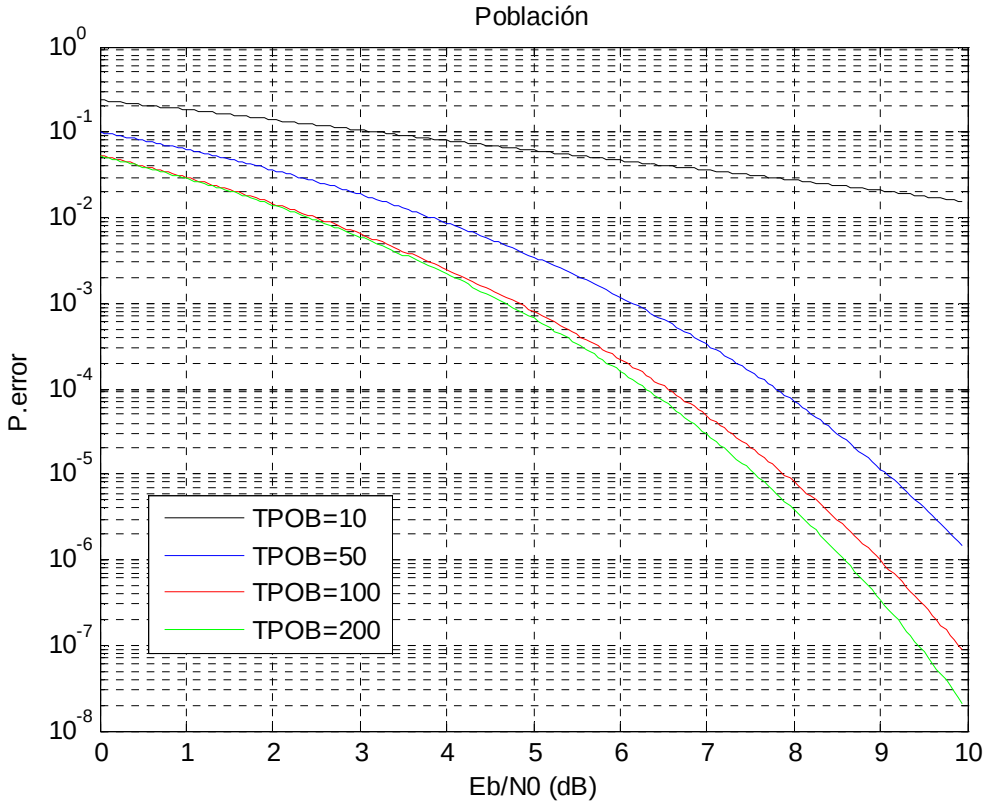

Fig. 21 Cota de la probabilidad de error respecto a la población

6.3.4 **Variación de valores *double***

Se fijará el valor del factor de cuantificación a 4, la relación señal ruido a 8 dB, se realizaran 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación será del 1% para 200 poblaciones.

Las simulaciones se realizaran para 11, 110 y 1100 valores *double.*

numeros_double=110

Vector Y

[0.000000, 0.010295, 0.070773, 0.071150, 0.071150, 0.372398, 0.372398, 0.581424, 0.581424, 0.991350, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.000006855

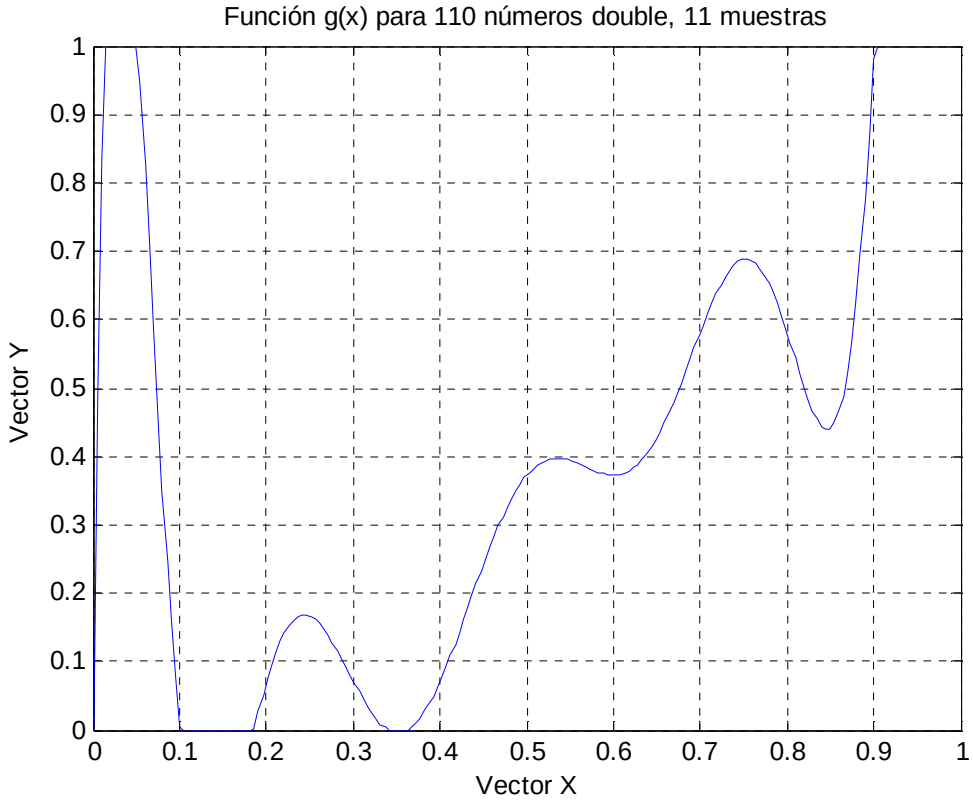

Fig. 22 Función *g(x)* para 110 valores *double*, 11 muestras

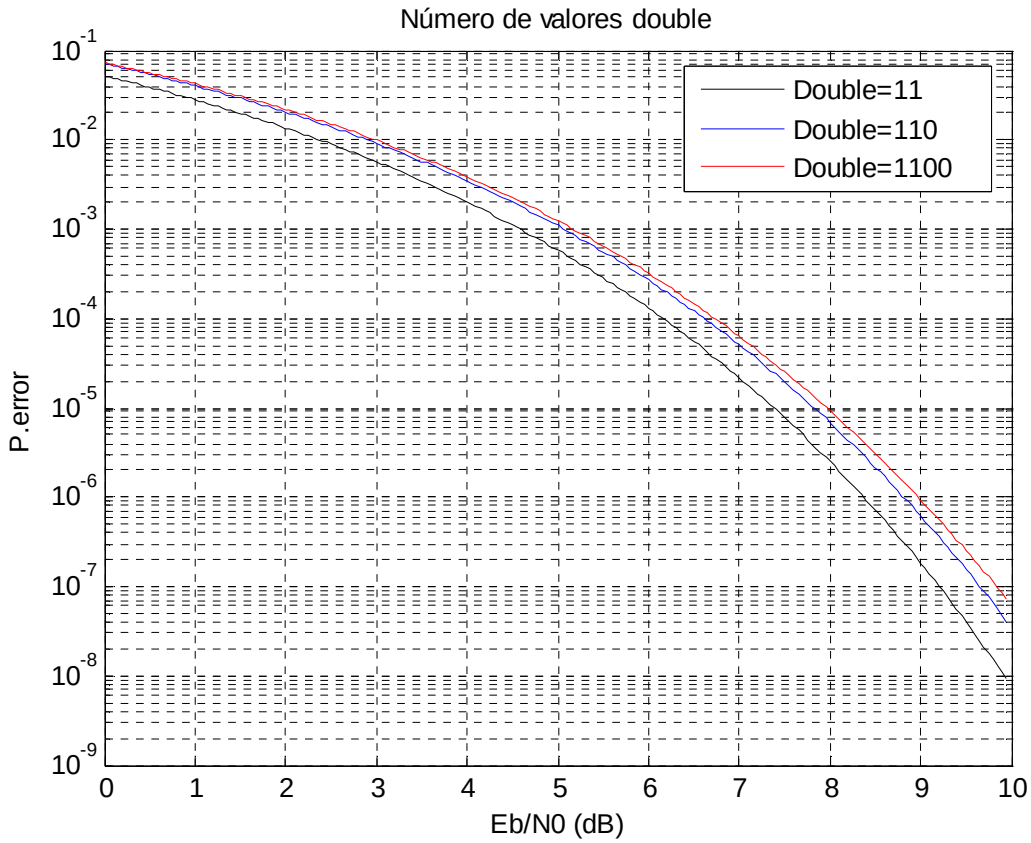

Fig. 23 Cota de la probabilidad de error respecto a los valores *double*, 11 muestras

6.3.5 **Variación del número de iteraciones en el algoritmo genético**

Se fijará el valor del factor de cuantificación a 4, la relación señal ruido a 8 dB, se asignan 1100 valores *double* y tanto la probabilidad como el porcentaje de mutación será del 1% para 200 poblaciones.

Se realizaran 10,50, 150 y 300 iteraciones en cada simulación.

Iteraciones=10

Vector Y

[0.000000, 0.054846, 0.054846, 0.054846, 0.054846, 0.280930, 0.280930, 0.417349, 0.443163, 0.772499, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000706371

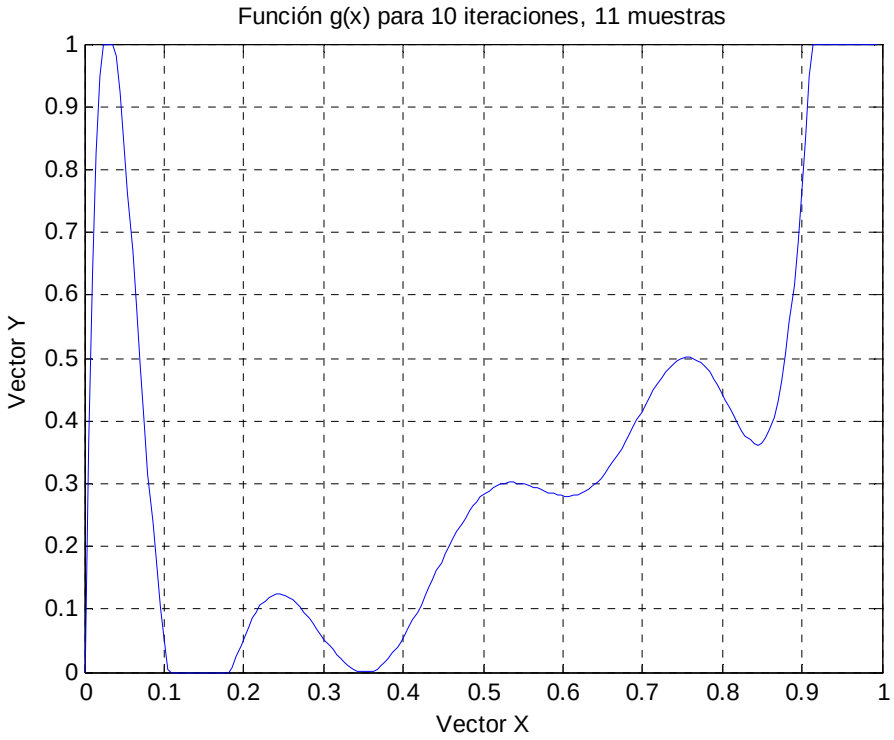

Fig. 24 Función *g(x)* para 10 iteraciones, 11 muestras

Iteraciones=50

Vector Y

[0.000000, 0.000807, 0.002179, 0.213948, 0.213948, 0.362191, 0.362191, 0.648381, 0.850985, 0.958312, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000087068

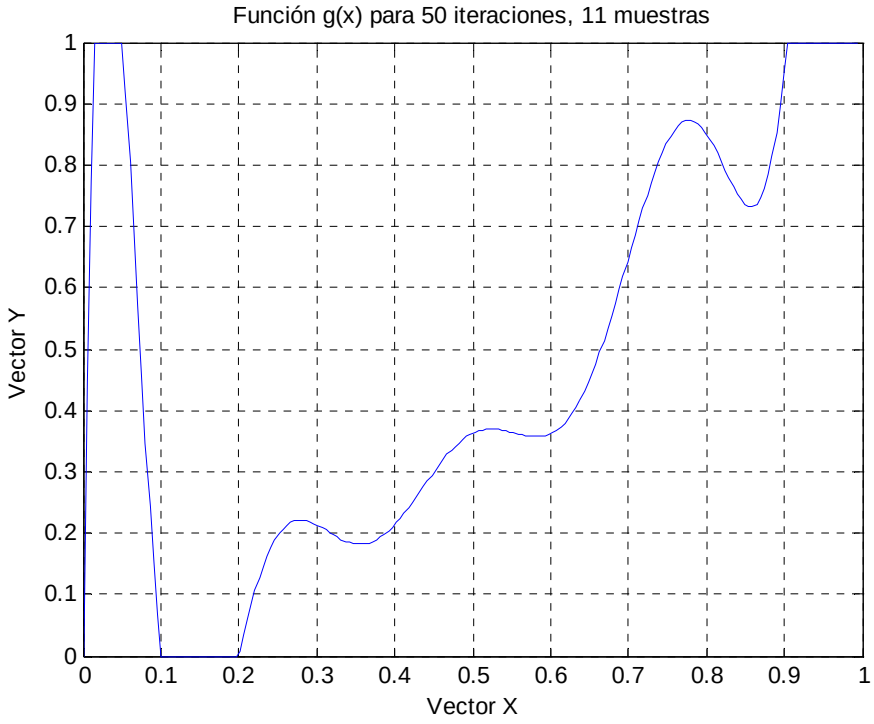

Fig. 25 Función *g(x)* para 50 iteraciones, 11 muestras

Iteraciones=150

Vector Y

[0.000000, 0.005440, 0.161553, 0.161553, 0.413265, 0.413265, 0.598971, 0.600113, 0.684541, 0.697313, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000020901

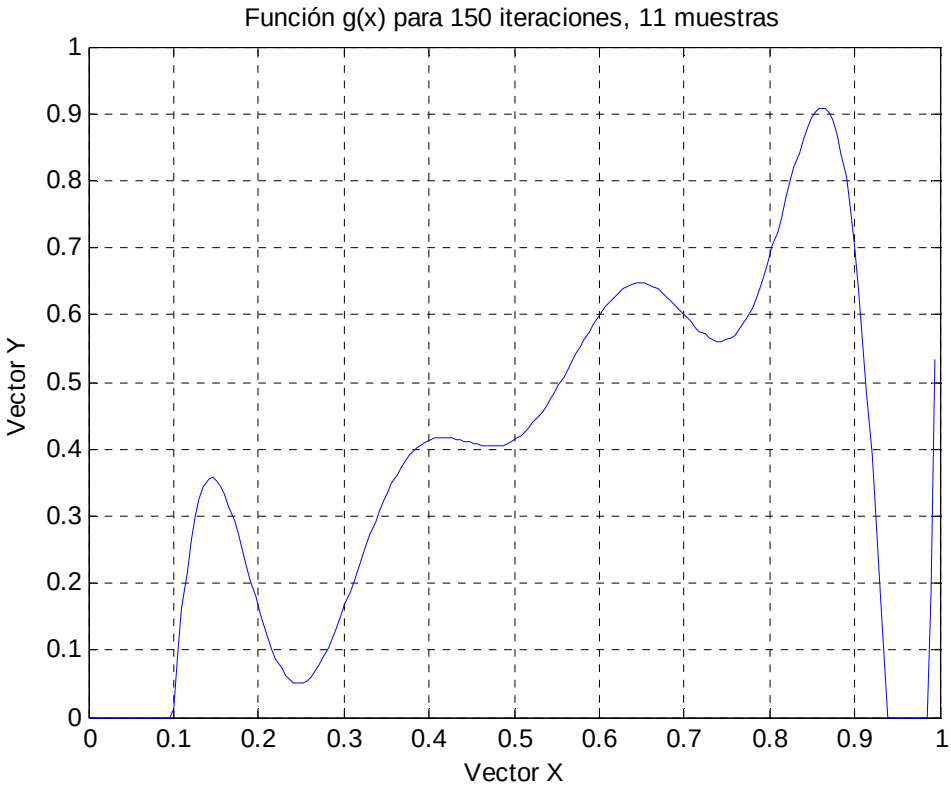

Fig. 26 Función *g(x)* para 150 iteraciones, 11 muestras

Iteraciones=300

Vector Y

[0.000000, 0.086415, 0.099563, 0.349014, 0.349014, 0.379936, 0.379936, 0.786241, 0.903218, 0.997196, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000015506

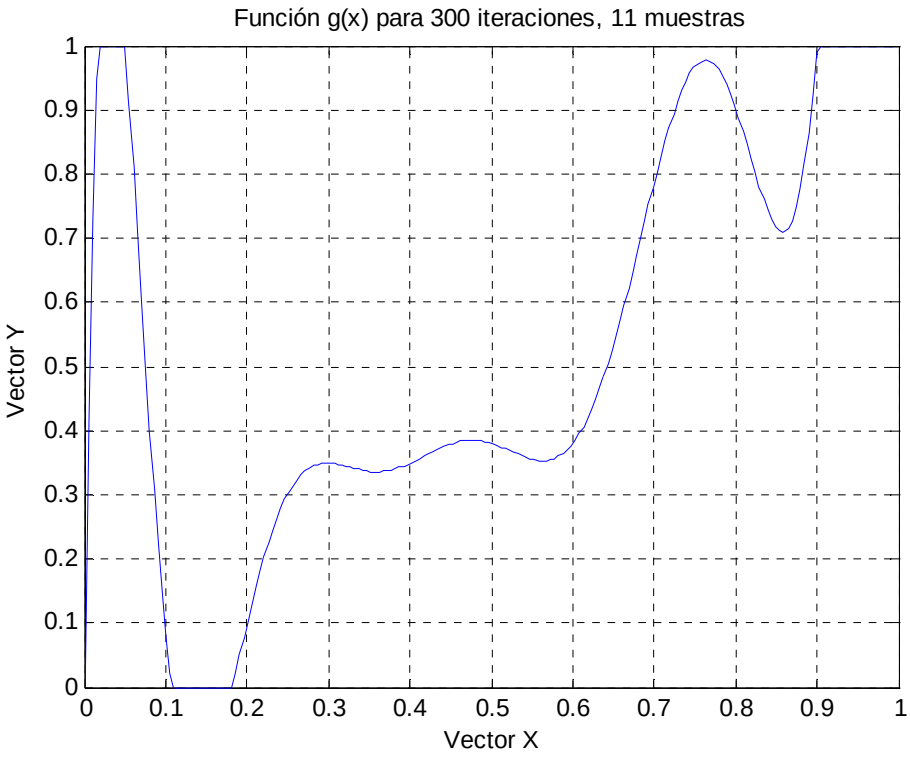

Fig. 27 Función *g(x)* para 300 iteraciones, 11 muestras

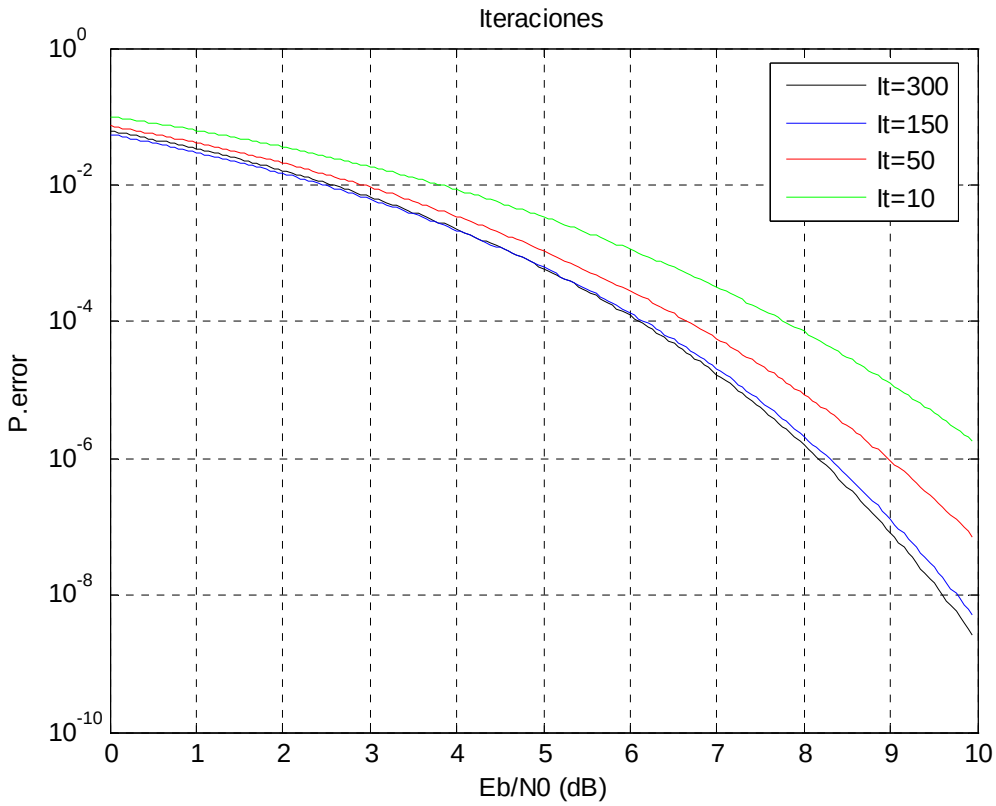

Fig. 28 Cota de la probabilidad de error respecto a las iteraciones, 11 muestras

6.3.6 **Variación de la probabilidad de mutación**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, para 150 iteraciones del algoritmo genético, introduciendo 1100 valores *double* con un porcentaje de mutación del 1% para un valor de población de 200 modificando la probabilidad de mutación.

Las medidas de probabilidad de mutación serán del 10%, 40%, 65% y 100%.

Probabilidad =40%

Vector Y

[0.000000, 0.013735, 0.082995, 0.313098, 0.401356, 0.424853, 0.471338, 0.569323, 0.847191, 0.973885, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000041439

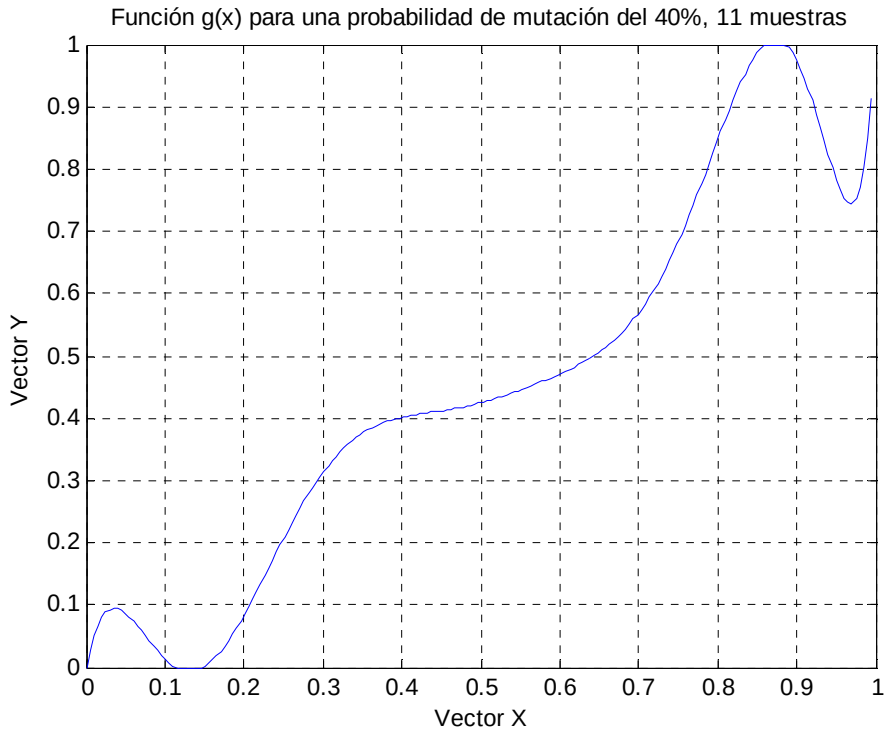

Fig. 29 Función *g(x)* para una probabilidad de mutación del 40%, 11 muestras

Probabilidad =100%

Vector Y

[0.000000, 0.018850, 0.172591, 0.241558, 0.463771, 0.463771, 0.659981, 0.659981, 0.761201, 0.838206, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.000000985

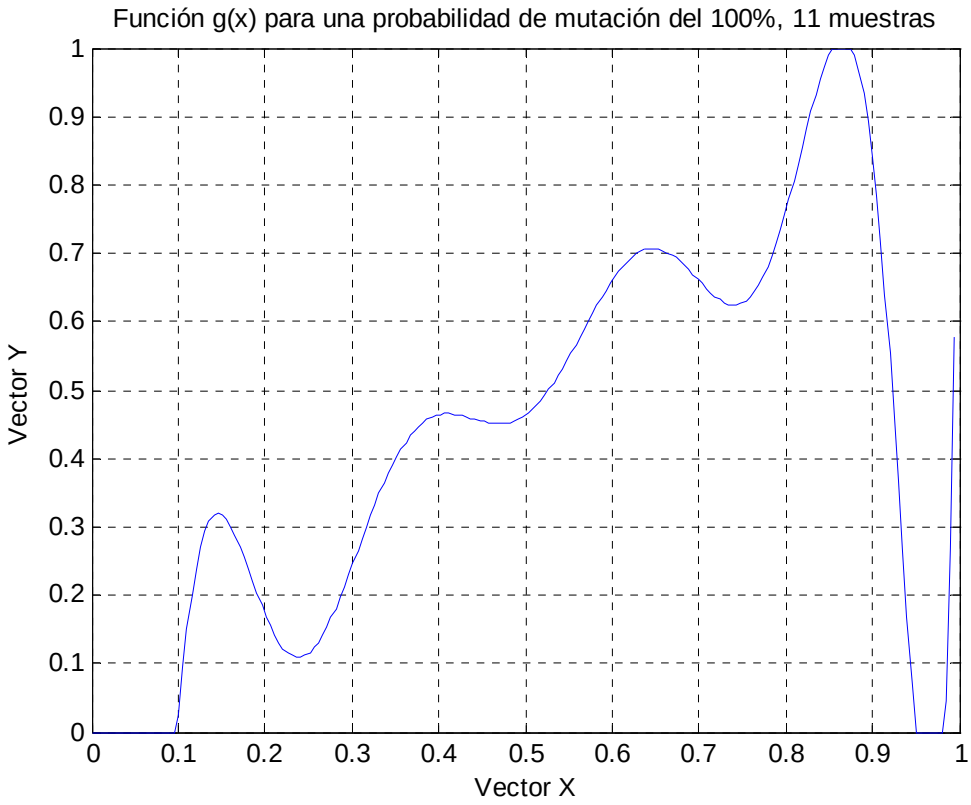

Fig. 30 Función *g(x)* para una probabilidad de mutación del 100%, 11 muestras

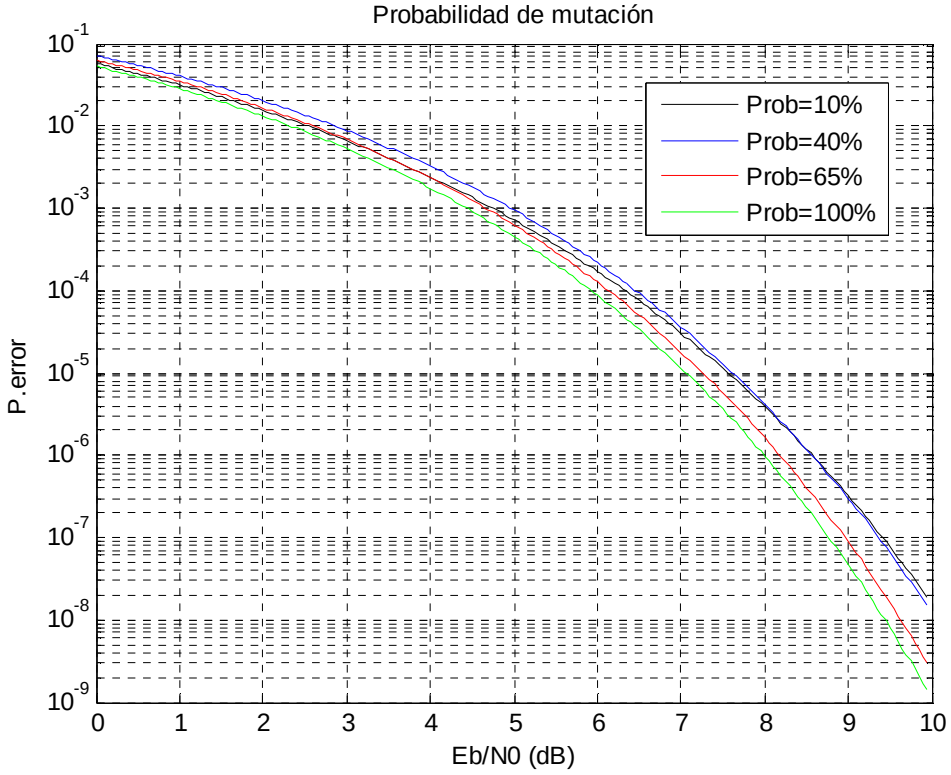

Fig. 31 Cota de la probabilidad de error respecto a la probabilidad de mutación, 11 muestras

6.3.7 **Variación del porcentaje de mutación**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, para 150 iteraciones del algoritmo genético, introduciendo 1100 valores *double* con una probabilidad de mutación del 100% para un valor de población de 200 modificando el porcentaje de mutación dentro de esta probabilidad.

Se ha realizado esta última medida con la probabilidad al 100% debido a que en el caso anterior al aumentar la probabilidad de mutación mejora mucho el error y partiendo de este dato conocido queremos seguir mejorándolo.

Las medidas del porcentaje de mutación serán para una variación del 10%, 40%, 65% y 100%

Porcentaje=40%

Vector Y

[0.000000, 0.105593, 0.140998, 0.140998, 0.341461, 0.342436, 0.652383, 0.666417, 0.666417, 0.774045, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000029098

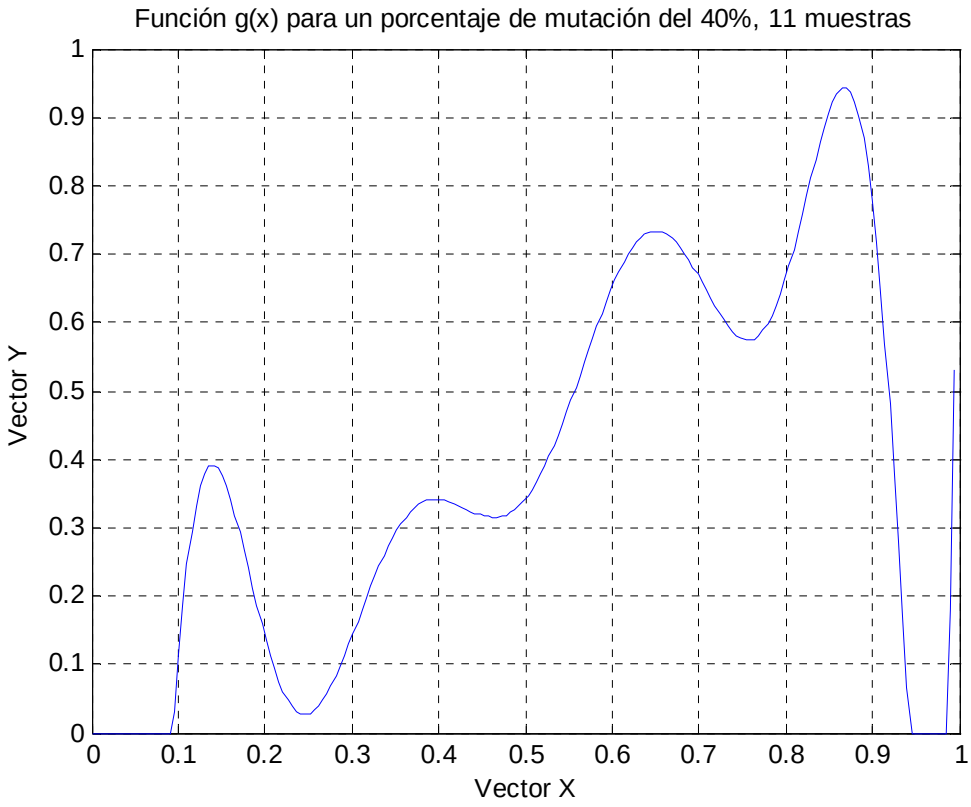

Fig. 32 Función *g(x)* para un porcentaje de mutación del 40%, 11 muestras

Porcentaje=100%

Vector Y

[0.000000, 0.023318, 0.217275, 0.217275, 0.536890, 0.536890, 0.737432, 0.737432, 0.850927, 0.962516, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.000000746

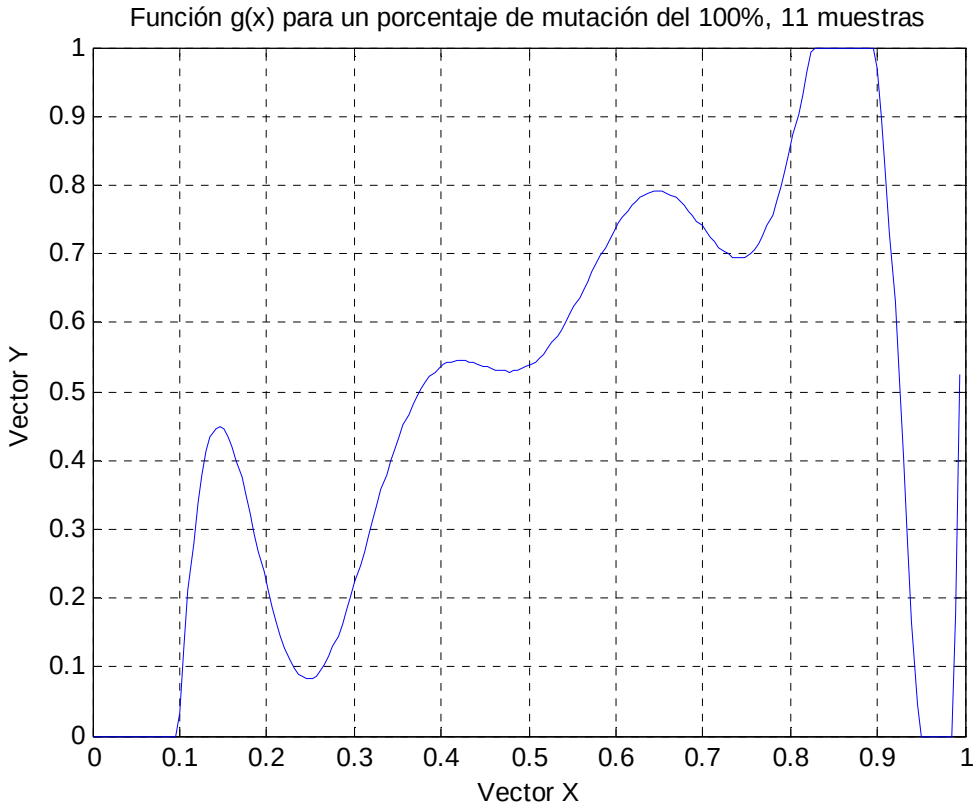

Fig. 33 Función *g(x)* para un porcentaje de mutación del 100%, 11 muestras

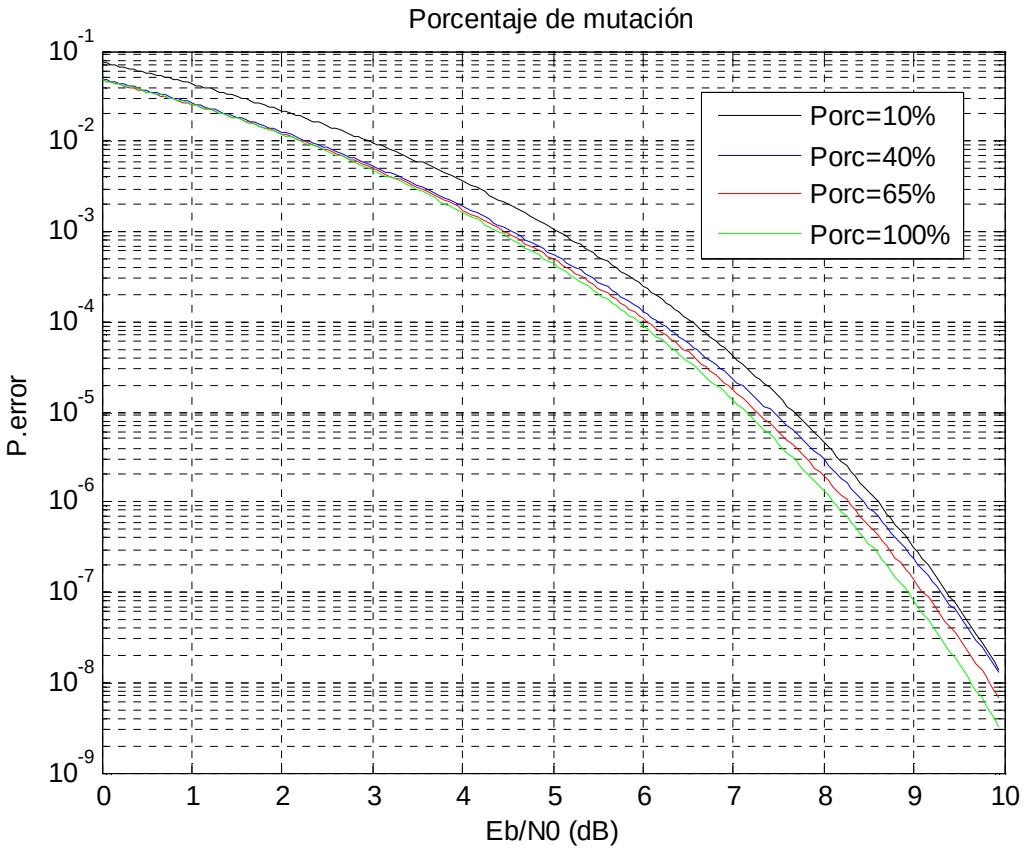

Fig. 34 Cota de la probabilidad de error respecto al porcentaje de mutación, 11 muestras

6.3.8 **Optimización**

Caso 1

Para este caso tomo un valor de población de 200, para 1100 valores de números *double*, 300 iteraciones en el algoritmo genético, el factor de cuantificación a 4 y tanto una probabilidad como un porcentaje de mutación del 100% para una relación señal ruido de 9 dB.

Vector Y

[0.000000, 0.061086, 0.193351, 0.193351, 0.466003, 0.466003, 0.625745, 0.625745, 0.726003, 0.737844, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000000907

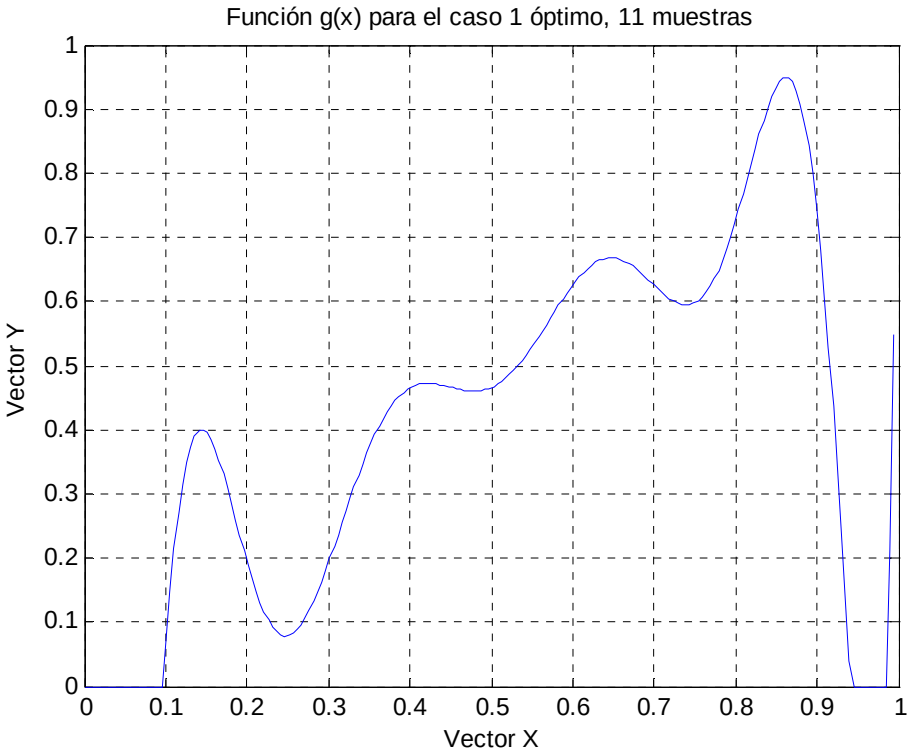

Fig. 35 Función *g(x)* para el caso 1 óptimo, 11 muestras

Caso 2

Se tomarán los mismos valores que en el caso anterior pero para una relación señal ruido se fija en 8 dB.

Vector Y

[0.000000, 0.114297, 0.191818, 0.261955, 0.508769, 0.508769, 0.719967, 0.719967, 0.761880, 0.866806, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000008209

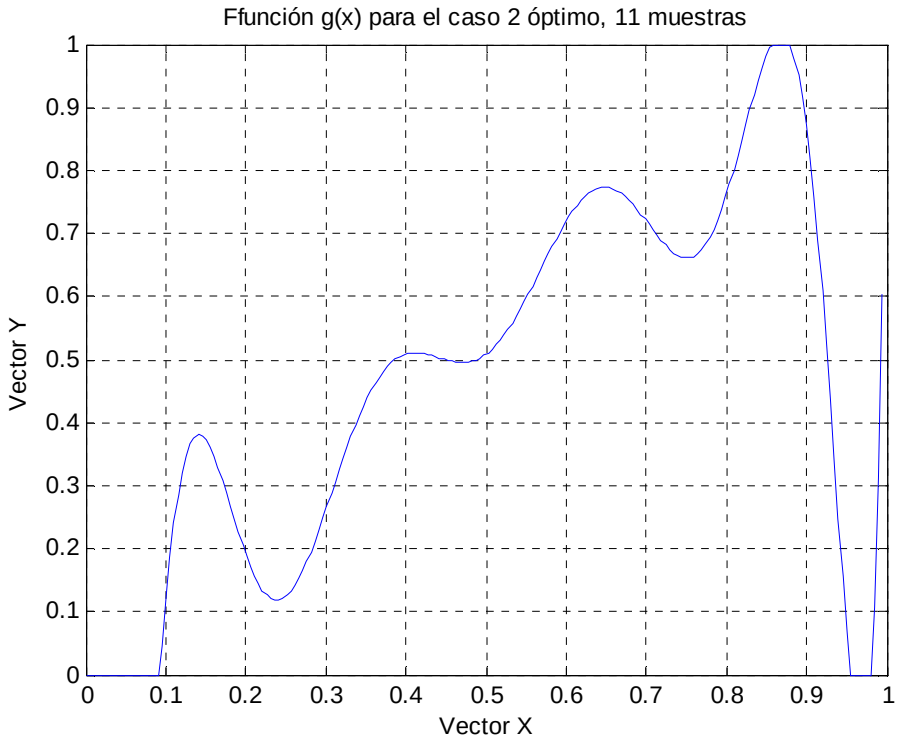

Fig. 36Función *g(x)* para el caso 2 óptimo, 11 muestras

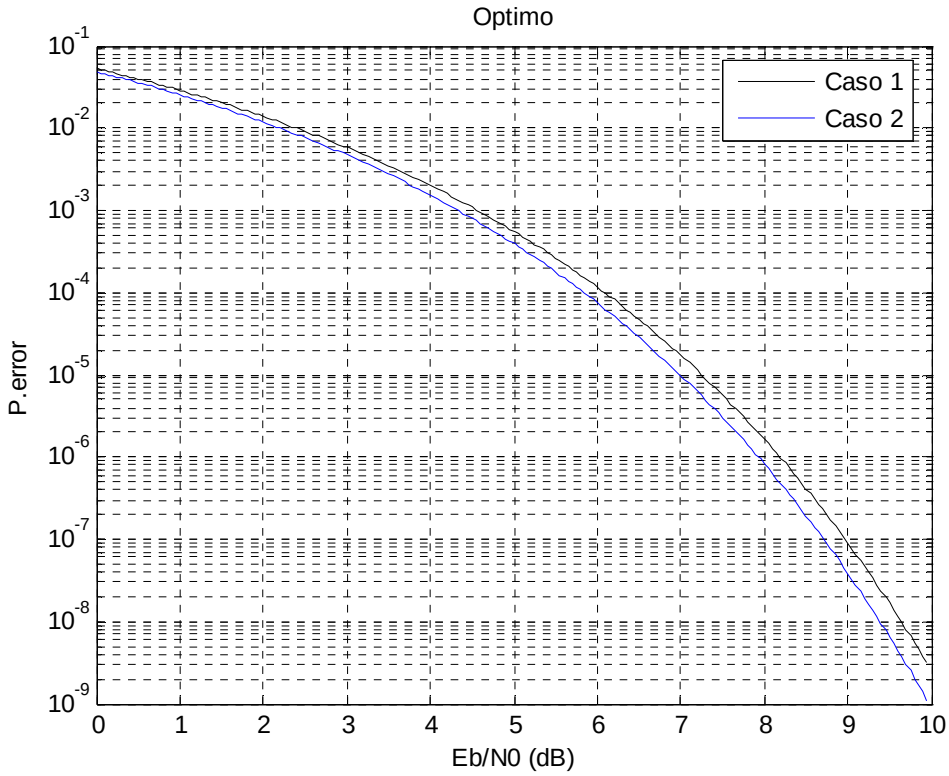

Fig. 37 Cota de la probabilidad de error respecto a los casos óptimos, 11 muestras

6.4 Consideración del error

A medida que ampliamos el número de muestras con que se discretiza *g(x)*, se hace más visible una gran variabilidad entre las mismas simulaciones, por lo que se tratará de compensar esta dispersión aparente incrementando la población y el número de iteraciones en el algoritmo ya que son los parámetros que en mayor medida están afectando al programa y se estima que podría mejorar el resultado.

Para mostrar más claramente esta situación se van a mostrar los datos obtenidos en dos situaciones con los mismos parámetros para ver cómo pueden llegar a variar en una situación límite y cuál seria su valor óptimo aumentando la población o las iteraciones del bucle genético:

6.4.1 Situación 1

Se fijará el valor del factor de cuantificación 4, el número de valores *double* será 1100, la relación señal ruido a 8 dB, se realizaran 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación será del 1%, tomando 200 poblaciones.

Caso1

Vector Y

[0.000000, 0.080126, 0.104281, 0.121916, 0.361772, 0.361772, 0.537197, 0.537197, 0.538346, 0.988469, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000084063

Fig. 38 Función *g(x)* para la situación 1, caso 1 de error, 11 muestras

Caso 2

Vector Y

[0.000000, 0.112359, 0.112359, 0.112359, 0.315153, 0.316594, 0.599053, 0.599053, 0.650854, 0.656655, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000038635

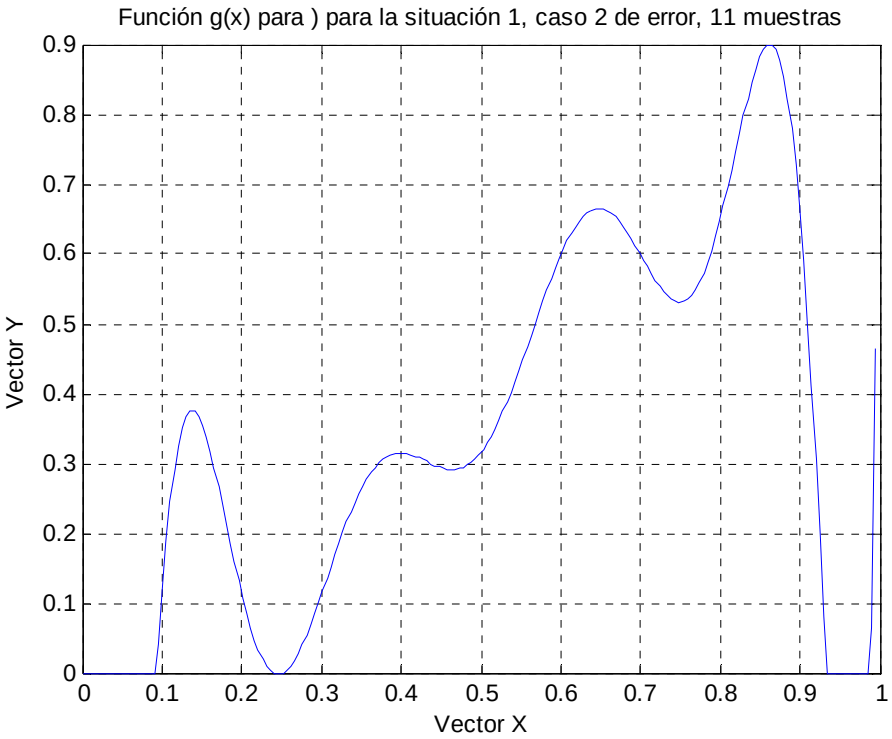

Fig. 39Función *g(x)* para la situación 1, caso 2 de error, 11 muestras

Caso 3

Dada la gran diferencia respecto al error realizaremos nuevamente la simulación con el doble de iteraciones (300) en el algoritmo genético, puesto que este es el mejor parámetro para solucionarlo, ya que a medida que crece el bucle de optimización más adecuada es la solución obtenida.

Vector Y

[0.000000, 0.094488, 0.153744, 0.262258, 0.464648, 0.464648, 0.641578, 0.641966, 0.792242, 0.792242, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000011568

Fig. 40 Función *g(x)* para la situación 1, caso 3 de error, 11 muestras

En la siguiente gráfica se puede apreciar la evolución de la cota de error en función de la relación señal ruido, siendo los casos 1 y 2 los obtenidos con un alto número de muestras y pocos ciclos en el algoritmo genético, para conseguir un resultado más fiable se han aumentado los ciclos en el tercer caso lo que ha propiciado un error más bajo.

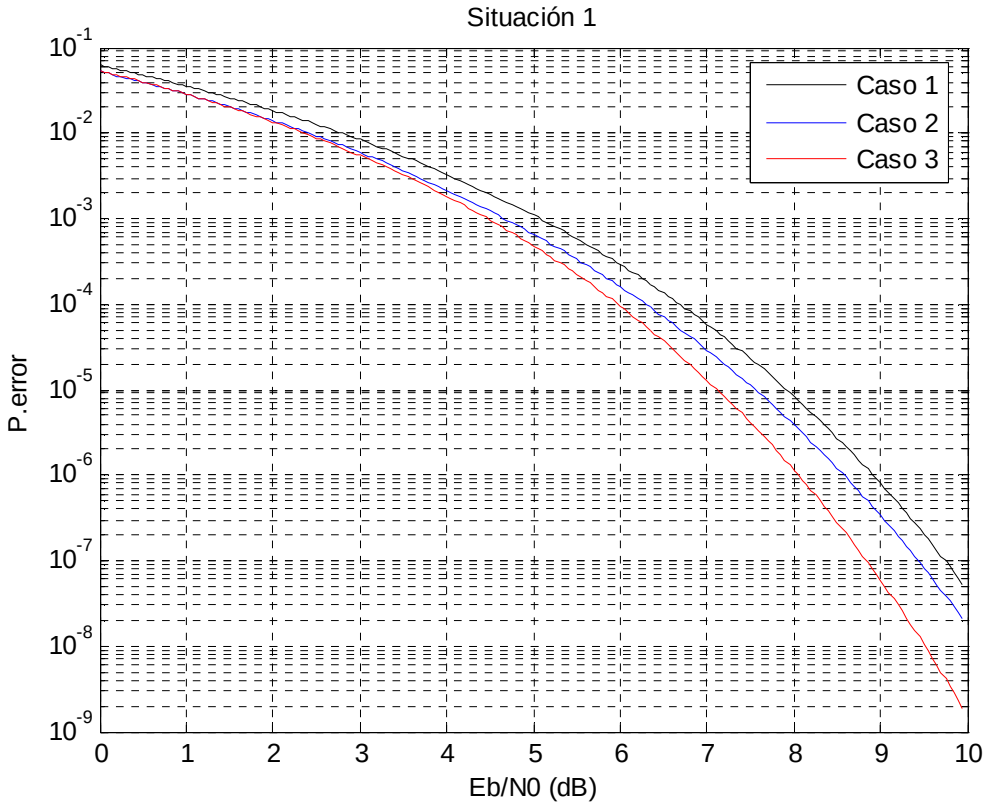

Fig. 41 Cota de la probabilidad de error respecto a la situación 1 de error, 11 muestras

6.4.2 **Situación 2**

Se fijará el valor del factor de cuantificación a 4, el número de valores *double* a 1100, la relación señal ruido a 8 dB, se realizaran 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación será del 100%, tomando 200 poblaciones.

Caso 1

Vector Y

[0.000000, 0.014376, 0.057248, 0.109464, 0.109464, 0.358259, 0.358259, 0.588553, 0.588553, 1.000000, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000064398

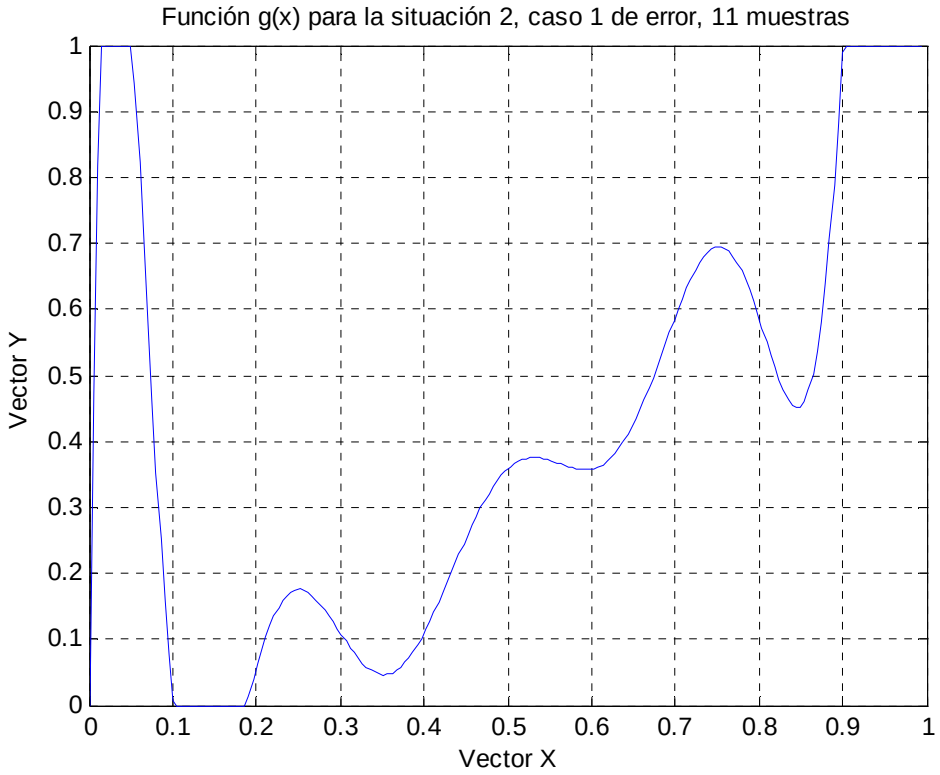

Fig. 42 Función *g(x)* para la situación 2, caso 1 de error, 11 muestras

Caso 2

Vector Y

[0.000000, 0.019235, 0.176821, 0.182644, 0.431485, 0.438919, 0.679007, 0.689743, 0.752963, 0.7847290, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000013178

Fig. 43 Función *g(x)* para la situación 2, caso 2 de error, 11 muestras

Caso 3

Esta vez también tomaremos el doble de iteraciones en el algoritmo evolutivo y aumentaremos la población a 300.

Vector Y

[0.000000, 0.023318, 0.217275, 0.217275, 0.536890, 0.536890, 0.737432, 0.737432, 0.850927, 0.962516, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.000000746

Fig. 44 Función *g(x)* para la situación 2, caso 3 de error, 11 muestras

Al igual que en la situación 1 los valores obtenidos en la cota de la probabilidad de error son mayores de lo esperado, por lo que los optimizamos en este caso aumentado la población y las iteraciones del algoritmo genético para conseguir un resultado más fiable.

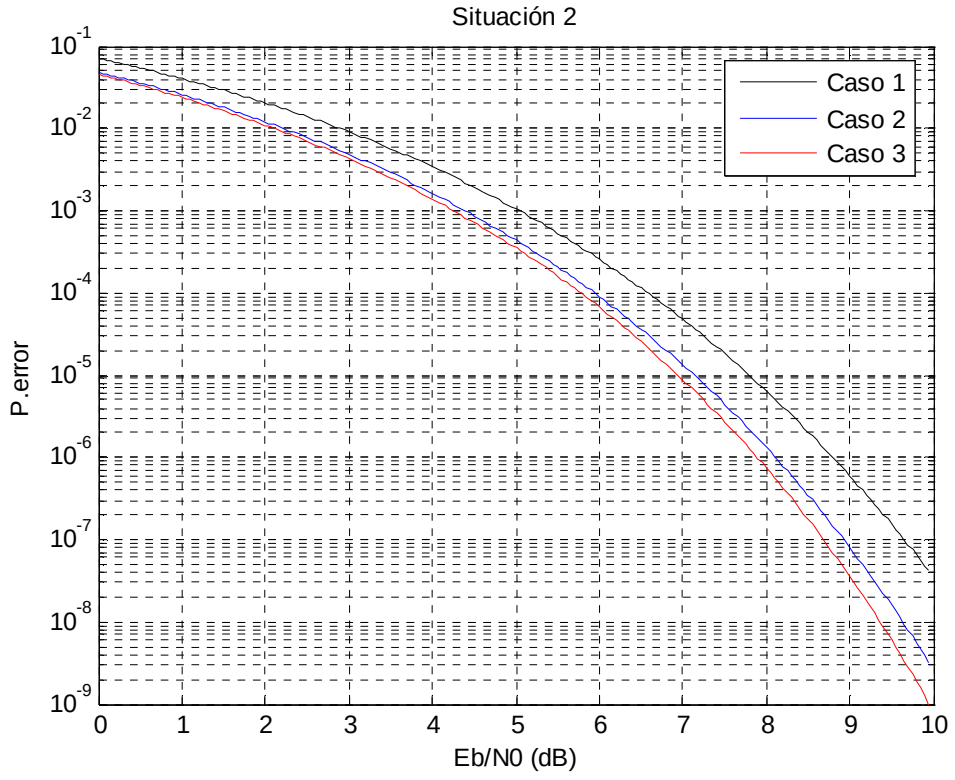

Fig. 45 Cota de la probabilidad de error respecto a la situación 2 de error, 11 muestras

6.5 Valoración de los resultados

Pese a mostrar solo los resultados más destacables para 5 y 11 muestras en el apartado anterior y tener todas las simulaciones para estas muestras en el apéndice, también se ha realizado la simulación de todos los parámetros para 4, 6, 7, y 9 muestras para confirmar el funcionamiento regular en cada caso, no obstante estos resultados no se indicarán en este escrito para evitar la redundancia de resultados, solo se indicará la relevancia de los mismos.

En primer lugar se ha realizado el estudio de cada parámetro de manera independiente, tomando como constantes el resto, de esta forma se ha procedido a asignarle diferentes valores y estudiar su variación.

Posteriormente se ha realizado una simulación alterando todos los parámetros en función de su rendimiento estudiado anteriormente, optimizando así el programa para que el resultado sea lo más eficiente y fiable posible.

Al estudiar de manera independiente cada parámetro se ha podido comprobar su relevancia, indicando esta a continuación con el nombre del parámetro que se ha dado en el programa:

- TPOB: el parámetro TPOB indica la población con la que se cuenta en la simulación, de esta forma se han realizado estudios para 10, 50, 100 y 200 poblaciones, de esta manera se ha podido comprobar que a medida que el número de poblaciones crecía el resultado de la simulación era cada vez mejor, siendo la probabilidad de error más baja. Esto se debe a que a medida que crecen las poblaciones, el algoritmo genético cuenta con una mayor cantidad de valores sobre los que iterar y aproximarse al resultado.

- Iteraciones: las iteraciones se refieren al número de veces que se ejecutará el algoritmo genético, en este caso se ha probado con 10, 50, 150 y 300 iteraciones, de esta forma se ha apreciado que a medida que crecía el número de iteraciones el resultado tenía una menor valor de la cota de la probabilidad de error, esto es debido a que cuantas más veces se ejecute el algoritmo genético más preciso se hará el resultado que obtendremos.

- Ndatos_double: se refiere al número de valores *double* que asignamos a cada población, por lo que a medida que aumenta su valor el algoritmo es más fiable y por lo tanto se obtiene un mejor resultado; en los ejemplos mostrados la mejora respecto a este parámetro es muy limitada ya que se ha trabajado con un alto número de iteraciones y poblaciones, pero en estudios con valores más bajos de estos parámetros se ha mostrado mucho más claramente la mejora.

- Pmut_double: se corresponde con la probabilidad de los miembros de la población que sufran una determinada mutación; se estima que la mayor parte de la población se mantendrá estable. No obstante, las pruebas se han realizado también para que afecte a un amplio número de miembros de la misma; tras numerosas pruebas se ha constatado que el resultado de la simulación es aceptable para probabilidades de mutación bajas en los casos en los que se

discretiza *g(x)* con pocos puntos, mientas que ha de ser más alto para las simulaciones en que se usa una discretización con más puntos.

- P_mut_double: este parámetro se corresponde con el porcentaje de cambio que sufre cada valor de la población sobre el que recae una mutación, la variación de cada valor en un caso real no suele ser muy alta, sin embargo también se han supuesto casos en los que produce una variación del 100%. Como conclusión de las pruebas se puede decir que, en las simulaciones con un número bajo de grados de libertad es preferible una variación baja, mientras que en las simulaciones con un alto número de grados de libertad es preferible que la variación sea alta.

- Q: se corresponde con el factor de cuantificación, nos indica el número de bits que se toman para comparar los resultados y tener una mayor precisión en cada simulación, se estima que el error que produce la cuantificación no es muy grande [3], este parámetro a partir de un valor no muy alto mantiene el comportamiento caótico de la secuencia, de seguir elevándolo solo se perderá tiempo en la ejecución de programa sin obtener una mejoría apreciable.

- Eb: se corresponde con el valor de referencia de la relación señal ruido que se ha tomado para realizar cada simulación, de esta manera no afectará a la simulación si no se produce un error excesivo, como para el caso de 5 muestras; teóricamente en el caso de 11 muestras también debería coincidir, sin embargo esto no es así debido a la aparición de un error que aumenta al crecer el número de muestras. Debido a que su efecto es de una notable relevancia se comentará con más profundidad en los siguientes apartados, sobre resultados y conclusiones.

- Cuantos: es el número de puntos con que se discretiza g(x), este valor menos 2 es el número de grados de libertad. Se han realizado simulaciones de 3 hasta 11 muestras, o de 1 a 9 grados de libertad, observando que en el caso de simular de 3 a 5 muestras se consiguen valores entre 0 y 1 prácticamente lineales, sin embargo esta proporción no se mantiene si las simulaciones se han realizado con un número mayor de muestras.

Tras el estudio individualizado de cada uno de los parámetros podemos concluir que para obtener el resultado mas fiable conviene trabajar con un elevado número de iteraciones del algoritmo genético, una alta cantidad de poblaciones y miembros o valores de cada una, mientras que es preferible utilizar un bajo número puntos en la discretización de *g(x)* y por lo tanto de probabilidad y porcentaje de mutación, mientras que la relación señal ruido y el factor de cuantificación no resultan muy significativos.

En la mayoría de las simulaciones se ha trabajado con un alto número de iteraciones y poblaciones, siendo estos los parámetros que más afectan al resultado del programa, por lo que la mayoría de las soluciones son muy aproximadas a los valores esperados.

Para una mayor aproximación del resultado se han realizado simulaciones variando varios parámetros a la vez, conforme a las especificaciones anteriormente indicadas.

Estas simulaciones se indican en los estudios anteriores como casos *óptimos* y se aprecia cómo el valor final de la función de coste es inferior al de los resultados anteriores.

Puesto que para los estudios de las simulaciones con un elevado número de grados de libertad los resultados pueden llegar a ser muy dispares, se ha optado por elevar aquellos parámetros que lo hacen más fiable, en particular el número de iteraciones y el número de poblaciones, de esta forma los resultados son mucho mejores a costa de aumentar el tiempo que emplea el programa para ejecutarse.

Capítulo 7

7 Resultados globales

7. Resultados globales

Una vez ejecutado programa, estudiado la relevancia de cada parámetro y obtenido el valor optimizado de la cota de error, se procederá a compararla con la tasa de error binaria (BER) de una simulación realizada con los parámetros obtenidos para *g(x)*; ver apartado de Modulación caótica:

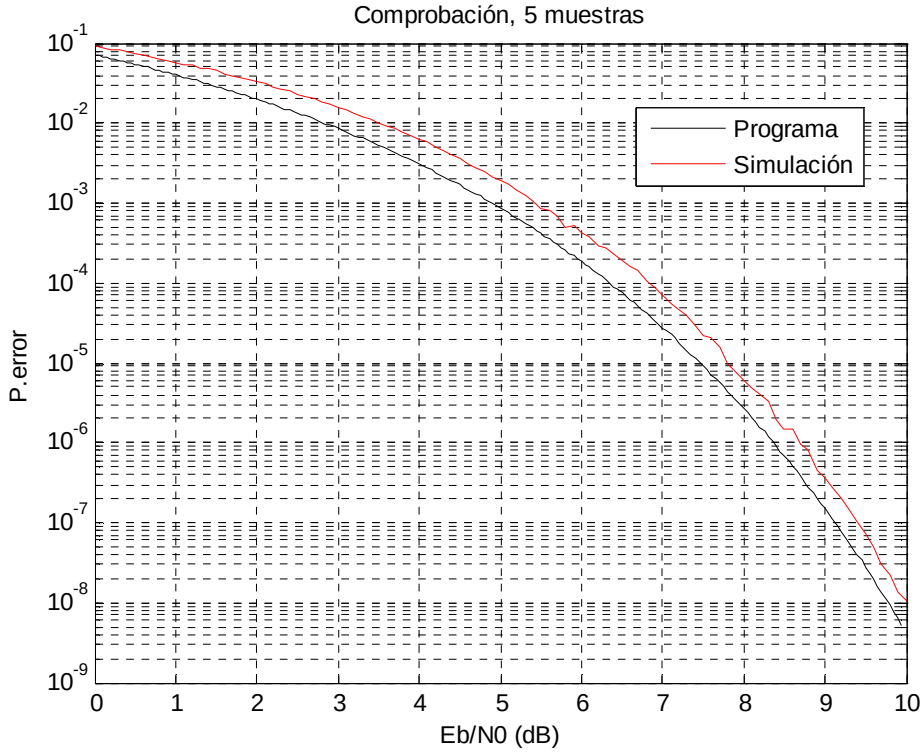

Fig. 46 Comparación, 5 muestras

Mientras que para el caso de 11 muestras la comprobación quedaría de la siguiente manera:

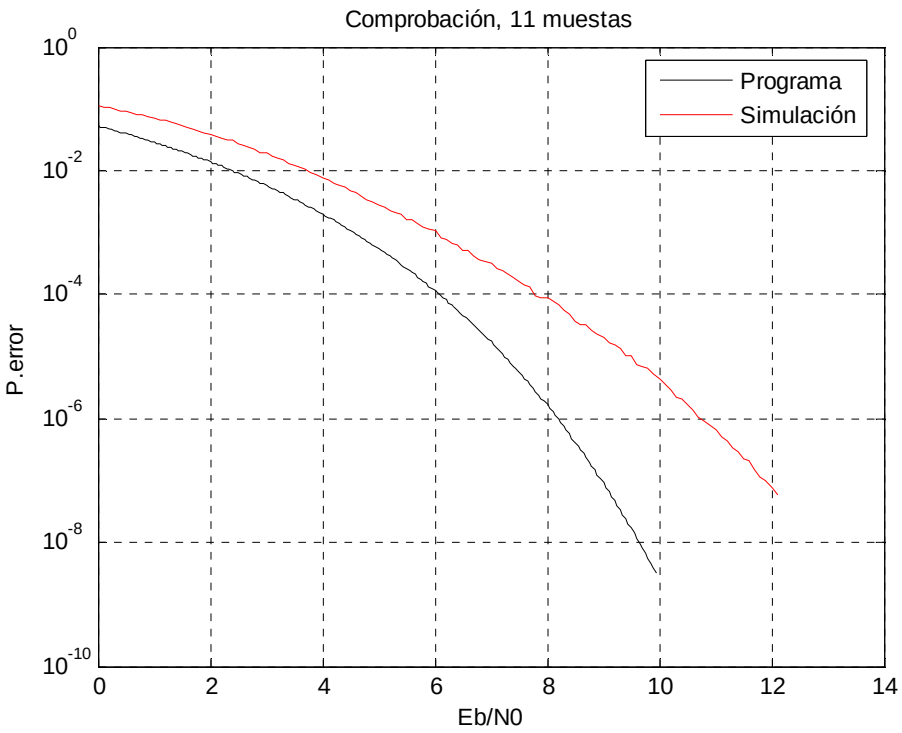

Fig. 47 Comparación, 11 muestras

Como se puede apreciar, en ambos casos, la cota de error es inferior en la obtenida mediante el programa con respecto a la implementación directa de la modulación caótica, esto es debido al efecto de aplicar una tasa de error distinta para cada caso, mientras que en la simulación real toma en cuenta todos los posibles errores en la modulación, en el caso realizado mediante el programa se han estimado los casos más generales, es decir, la existencia de uno o dos errores consecutivos en la modulación, como se indico en el apartado de Modulaciones caóticas.

También se observa que mientras en el caso de 5 muestras (3 grados de libertad) las funciones de las cotas de error están muy próximas, mientras que en el caso de 11 muestras (9 grados de libertad) estas funciones tienden a distanciarse a medida que aumenta la relación señal ruido lo que indica que el resultado de la optimización no es coherente.

Tras un estudio pormenorizado del problema se llegó a la conclusión de que este error se debía a que la forma de uso de la aproximación por *splines* no era la adecuada para representar la función *g(x).*

Por ello se representó la función *g(x)* para los casos en que se tomaban pocas muestras o grados de libertad:

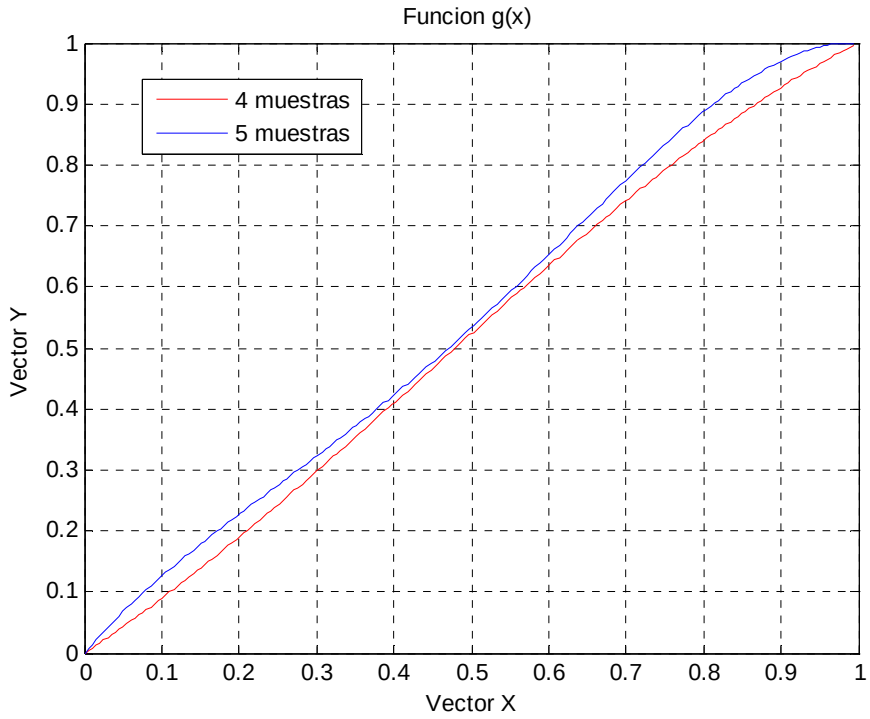

Fig. 48 Función *g(x)*, 4 y 5 muestras

Y se contrastó con los casos de un número mayor de muestras o grados de libertad:

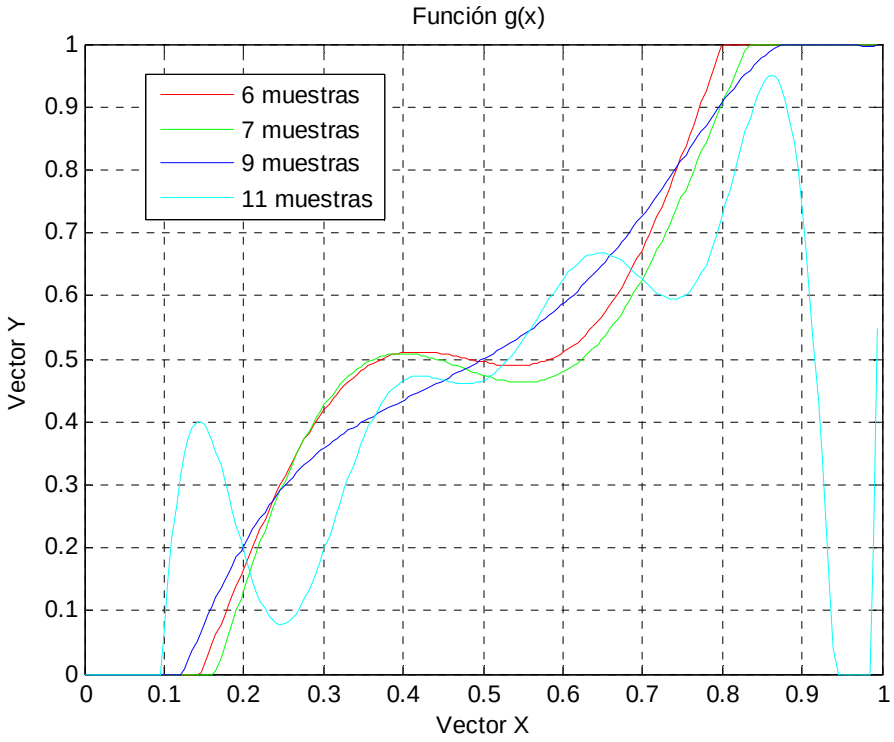

Fig. 49 Función *g(x)*; 6, 7, 9 y 11 muestras

Como se puede apreciar, mientras que en el estudio de la función para casos de pocas muestras la función que se consigue es prácticamente lineal, en los casos en los que se han tomado más muestras la función comienza a realizar escalones hasta que acaba teniendo una forma que no es estrictamente no decreciente y, por lo tanto, no válida.

Esto es debido a que los *splines* se han realizado con una interpolación no lineal que produce que la función varíe de forma polinómica entre estos puntos y no sea una función estrictamente no decreciente.

Se muestra a continuación el resultado obtenido para la cota de la probabilidad de error (en negro) y para el BER experimental (en azul) en el caso de discretización con 11 muestras de forma completamente lineal. Se muestra también el resultado del BER experimental obtenido con el supuesto juego de muestras óptimo correspondiente a ese caso (en rojo).

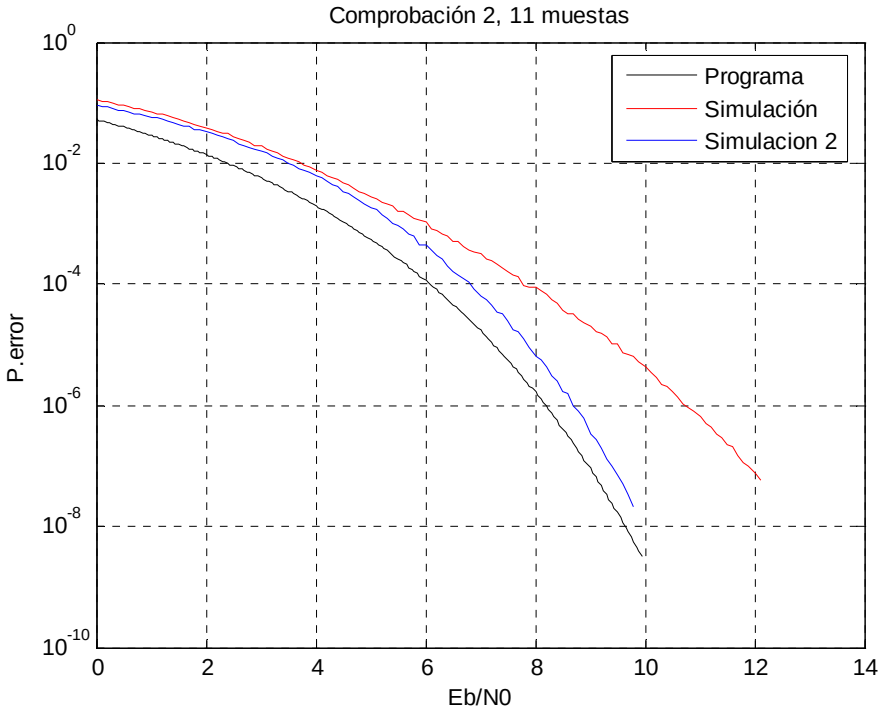

Fig. 50 Comparación 2, 11 muestras

De esta forma se demuestra que, al no asegurar que la función sea estrictamente no decreciente, la solución óptima aportada no es utilizable.

Capítulo 8
8 Conclusiones

Futuras líneas de trabajo

8. Conclusiones

Tras el estudio de cada parámetro se llega a la conclusión de que si quiere conseguir una consistente optimización de la modulación caótica en primer lugar se deberá incrementar el número de poblaciones de estudio y las iteraciones del algoritmo genético, debido a que son los parámetros que influyen en mayor modo en esta operación.

En segundo lugar conviene incrementar los miembros de la población y tener una probabilidad y porcentaje de mutación de esta no muy alto.

Hay que tener en cuenta que, si la mutación es excesiva, el algoritmo comienza a funcionar como una búsqueda aleatoria, con lo que pierde las propiedades iniciales del algoritmo genético.

Finalmente la relación señal ruido no influye en el proceso, al igual que el factor de cuantificación si su valor se encuentra por encima de 3.

El objetivo del algoritmo genético es obtener las mejores muestras en el intervalo de estudio para el diseño de la función *g(x),* con la que se trabajará para obtener la cota de error de la modulación caótica.

Entre las muestras obtenidas de la modulación se realizo una interpolación polinómica para la obtención los tramos intermedios, y así ajustar la función *g(x),* para esto se ha tenido en cuenta que estos valores no pueden ser superiores a 1, ni inferiores a 0.

Esta acotación produce que el estudio para pocas muestras corresponda con la estimación esperada, sin embargo para los casos en que el número de muestras es mayor (a partir de 6) produce que la función *g(x)* no esté correctamente representada, esto se produce ya que debería condicionarse también que los valores fueran de manera estrictamente no decreciente.

8.1 Futuras líneas de trabajo

La teoría del caos en comunicaciones digitales no dispone de una teoría sólida y fácilmente aplicable, debido a la complejidad y no linealidad inherente de dichos sistemas. Este proyecto ha permitido avanzar en la caracterización útil de los sistemas de comunicaciones basados en caos.

Una vez fijadas las bases de este proyecto se abren innumerables posibilidades que desarrollar.

Uno de los desarrollos que podrían tratarse en primer lugar es el estudio de la interpolación por los *splines*, cómo se comporta la función según el tipo de interpolación, también se podrá estudiar la variación de la cota en función de la cantidad de tipos de error que estimemos de la función *error* de la modulación caótica.

El desarrollo de la modulación caótica se ha tratado con la aplicación de Bernoilli, pero existen otras maneras con las que tratar la modulación, como la aplicación tienda, con las que se puede contrastar los resultados aquí obtenidos para conseguir el mejor resultado para una modulación de este tipo.

Capítulo 9

9 Pliego de condiciones

Herramientas

Requisitos mínimos

9. Pliego de condiciones

En este apartado se incluirá una descripción de las herramientas empleadas para desarrollar el proyecto, así como los requisitos mínimos de *hardware* y *software* que se requieren para su uso.

9.1 Herramientas

Las herramientas de trabajo que se han utilizado para implementar el programa son:

9.1.1 Matlab

Abreviatura de "laboratorio de matrices" es un *software* matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio. Está disponible para las plataformas Unix, Windows y Apple.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario y la comunicación con programas en otros lenguajes y con otros dispositivos *hardware*. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, Simulink para la simulación multidominio y GUIDE para interfaces de usuario.

MATLAB puede llamar funciones y subrutinas escritas en C o Fortran. Se crea una función envoltorio que permite que sean pasados y devueltos tipos de datos de MATLAB. Los archivos objeto dinámicamente cargables creados compilando esas funciones se denominan "MEX-files", aunque la extensión de nombre de archivo depende del sistema operativo y del procesador.

Sus funcionalidades de Matlab se agrupan en más de 35 cajas de herramientas y paquetes de bloques, siendo estos los siguientes:

MATLAB	Simulink
Matemáticas y Optimización	Modelado de punto fijo
Estadística y Análisis de datos	Modelado basado en eventos
Diseño de sistemas de control y análisis	Modelado físico
Procesado de señal y comunicaciones	Gráficos de simulación
Procesado de imagen	Diseño de sistemas de control y análisis
Pruebas y medidas	Procesado de señal y comunicaciones
Biología computacional	Generación de código
Modelado y análisis financiero	Prototipos de control rápido y SW/HW HIL

Desarrollo de aplicaciones	Tarjetas integradas
Informes y conexión a bases de datos	Verificación, validación y comprobación

Tabla 1: Bloques de Matlab y simulink

9.1.2 **Lenguaje C**

Se ha trabajado con lenguaje de programación C [15] en el entorno de desarrollo integrado Dev-C++.

Es un lenguaje orientado a la implementación de sistemas operativos, es apreciado por la eficiencia del código que produce y es uno de los lenguajes de programación más populares para crear *software* de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje débilmente tipificado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

Sus principales ventajas son el uso de un lenguaje muy eficiente puesto que es posible utilizar sus características de bajo nivel para realizar implementaciones óptimas, además es el lenguaje más portado en existencia, habiendo compiladores para casi todos los sistemas conocidos y proporciona facilidades para realizar programas modulares y/o utilizar código o bibliotecas existentes.

Sus principales inconvenientes es la gran diferencia en velocidad de desarrollo, es más lento programar en C debido que el compilador de C se limita a traducir código sin apenas añadir nada, en otros lenguajes la memoria es gestionada de forma transparente para el programador. Esto alivia la carga de trabajo humano y en muchas ocasiones previene errores, aunque también supone mayor carga de trabajo para el procesador, el mantenimiento también es más difícil y costoso que con lenguajes de más alto nivel.

Su proceso de compilación se realiza en varias fases que normalmente son automatizadas y ocultadas por los entornos de desarrollo, la primera es el preprocesado consistente en modificar el código fuente en C según una serie de instrucciones simplificando de esta forma el trabajo del compilador, posteriormente la compilación que genera el código objeto a partir del código ya preprocesado y por último el enlazado que une los códigos objeto de los distintos módulos y bibliotecas externas para generar el programa ejecutable final.

9.2 Requisitos mínimos

9.2.1 Software

- Dev-C ++
- Matlab 2007 o 2010
- Microsoft Windows o Linux
- Librería GSL
- Microsoft Office 2003, 2007 o 2010 u Open Office
- Adobe Reader

9.2.2 Hardware

- Ordenador con RAM de 2 Gb
- Placa base compatible.
- Monitor
- Impresora HP Deskjet 710C o superior
- Ratón
- Unidad CD-ROM
- Teclado

Capítulo 10

10 Presupuesto

Coste humano

Coste material

Coste final

10. Presupuesto

En este apartado se van a describir los costes materiales y humanos de derivados de la realización de este proyecto de fin de carrera.

10.1 Coste humano

Se corresponde con la relación de horas trabajadas con respecto al salario por hora del personal que ha intervenido en el desarrollo del proyecto.

Personal	Tiempo trabajado	Sueldo	Total
Ing. de Telecomunicación (Esp. Sistemas caóticos)	100 horas	60€/hora	6000 €
Ing. de Telecomunicación (Esp. Algoritmos genéticos)	10 horas	60€/hora	600 €
Ing. Técnico en Sistemas de Telecomunicación	900 horas	50€/hora	45000 €
Total			**51600 €**

Tabla 2 Coste Humano

10.2 Coste material

Se corresponde con el coste del *software*, *hardware* y la memoria final del proyecto, teniendo en cuenta que el precio de las licencias de los sistemas y herramientas utilizados están abonados por la universidad, lo que implica que el coste es cero para los usuarios universitarios.

Concepto	Precio	Unidades	Total
Ordenador (incluidos periféricos)	1195 €	1	1195 €
Impresora	95 €	1	95 €
Dev-C++	0 €	1	0 €
Matlab	0 €	1	0 €
Librería GSL	0 €	1	0 €
Microsoft Windows	0 €	1	0 €
Microsoft Office	0 €	1	0 €
Adobe Reade	0 €	1	0 €
Impresión	60 €	3	180 €
Encuadernación	20 €	3	60 €
Total			**1530 €**

Tabla 3 Coste material

10.3 Coste final

El coste final es la suma de los costes humanos y materiales, incluyendo un sobrecoste del 18% debido al IVA

No se realiza un beneficio industrial ya que el proyecto es punto de partida para nuevas desarrollos y será publicado de manera gratuita en la universidad para que cualquier usuario pueda consultarlo y continuar su desarrollo.

Concepto	Porcentaje	Total
Coste humano	100%	51600 €
IVA (Coste humano)	18%	9288 €
Coste material	100%	1530 €
IVA (Coste material)	18%	275.40 €
Total		**62693.40 €**

Tabla 4 Coste total

El coste total del proyecto esta estimado en: **62693.40 €**

Capítulo 11
11 Manual de usuario

Proyecto

Librería GSL

Funcionxy

Programa del proyecto

11. Manual de usuario

En este apartado se indicara cómo hay que realizar la creación de un proyecto en Dev-C++ y cómo se debe incluir la librería GSL.

11.1 Proyecto

Para crear un proyecto se ha abrirá la aplicación Dev-C++ y se seguirán los siguientes pasos:

- Se seleccionará en la barra de herramientas *Archivo* y dentro de esta sección *Nuevo.*
- Se tomará la opción de *Proyecto* y *Empty Project*, aceptándose esta ultima opción.
- El nuevo proyecto se guardara donde el usuario estime para su uso, indicándole una dirección y un nombre al proyecto.
- En el margen izquierdo de la pantalla aparecerá un icono con el nombre asignado al proyecto.
- Sobre este icono se pulsará con el botón derecho del ratón, lo que desplegará un menú contextual.
- Se seleccionará *Añadir a proyecto*, y se incluirán los ficheros correspondientes.

11.2 Librería GSL

Una vez creado el proyecto se procederá a incluir la librería GSL.

- Se obtendrá el fichero de la librería y se guardará en el PC del usuario.
- Se abrirá el proyecto en el que se quiere incluir.
- Se seleccionará el botón de *Proyecto* en la barra de herramientas, seleccionando la opción *Opciones del proyecto*.
- En la barra de herramientas del nuevo menú se seleccionará *Compilador.*
    - Dentro de esta opción se pulsará sobre *Linker* y se indicará *Yes* sobre *Generar información de Debug*

- Sobre la barra de herramientas del menú anterior selecciono *Parámetros*

    - Añado las librerías del archivo GSL:
        - GnuWin32/lib/libgsl.dll.a
        - GnuWin32/lib/libgslcblas.a
        - GnuWin32/lib/libgslcblas.dll.a

- Sobre la barra de herramientas anterior selecciono *Directorios*

    - En *directorio de bibliotecas* añado
        - GnuWin32/lib
        - GnuWin32/bin
    - En *directorio de incluye* añado
        - GnuWin32/bin
        - GnuWin32/include

Una vez finalizados estos pasos el proyecto estará listo para compilarse y ejecutarse correctamente.

11.3 Funcionxy

Para utilizar esta función se deberán seguir los siguientes pasos:

- Se creará un proyecto con este fichero como se ha indicado en el apartado 9.1.

- Se incluirá la librería GSL.

- Se creara un ejecutable del proyecto (.exe).

- Se accederá mediante la línea de comandos a la dirección donde se halla guardado el ejecutable introduciendo:

    - Nombredelproyecto.exe datosfinales muestras ficherosalida

El *Nombredelproyecto* corresponde con el nombre que le haya puesto el usuario, *datosfinales* es el archivo que crea en el programa del proyecto, *muestras* es el número de grados de libertad con los que se haya trabajado y *ficherosalida* es el nombre que tendrá el fichero que guardará el resultado de la interpolación por *splines*.

Se debe trabajar con el fichero *datosfinales* en la misma dirección donde se encuentra el ejecutable del proyecto, al igual que la librería *gslcblas.dll.*

11.4 Programa del proyecto

Para saber trabajar con el programa del proyecto:

- Se creará un proyecto con los ficheros:
  - Main.c
  - Dependiente.h
  - Alea.h
  - AG.h
  - Cotas.h

- Se incluirá la librería GSL.

- Se ajustarán los parámetros que se estimen oportunos.

- Se compilará y ejecutará el programa.

Una vez compilado y ejecutado el proyecto nos proporcionará por pantalla:

- La secuencia de valores para los grados de libertad.

- El valor de la función de coste.

- Una *D* cada vez que se ha realizado un bucle de optimización dentro del algoritmo genético.

- Una *b* una vez ha finalizado de ejecutarse el programa.

También los siguientes archivos:

- Datosfinales: donde obtendremos los valores de los grados de libertad finales tras la optimización.

- Cotatotal: que indicará la cota de la probabilidad de error binaria.

Capítulo 12
12 Bibliografía

12. Bibliografía

[1] Francisco J. Escribano, Luis López y Miguel A. F. Sanjuán; "Performance evaluation of parallel concatenated chaos coded modulation", Journal of Communications Software and Systems, volumen 4, nº 2, páginas 159-165, 2008

[2] Francisco J. Escribano, Slobodan Kozic, Luis López, Miguel A. F. Sanjuán y Martin Hasler; "Turbo-like structures for chaos encoding and decoding", IEEE transactions on Communications, volumen 57, nº 3, páginas 597-601, Marzo 2009

[3] Francisco J. Escribano, Luis López y Miguel A. F. Sanjuán, "Evaluation of channel coding and decoding algorithms using discrete chaotic maps", Chaos, volumen 16, 2006.

[4] Sancho Salcedo Sanz, "Algoritmos de computación evolutiva y su aplicación a problemas de optimización en física", Revista española de física, volumen 22, nº 4, páginas 6-12, 2008

[5] Leopoldo Carro Calvo, "Aplicación de algoritmos genéticos al diseño y síntesis de dispositivos fotónicos", proyecto de fin de carrera, Universidad de Alcalá de Henares, Escuela Politécnica, volumen 1, 2009

[6] Ángel Manuel Pérez Bellido, "Asignación de recursos en redes de telecomunicación mediante algoritmos evolutivos y subasta tipo *vickey*", tesis de master, Universidad de Alcalá de Henares, 2007

[7] Emilio Gedeón Ortiz García, "Desarrollo de heurísticos emergentes para la resolución de juegos de lógica", tesis de master, Universidad de Alcalá de Henares, 2007

[8] M. P. Kennedy, R. Rovatti, and G. Setti, "Chaotic Electronics in telecommunications", CRC Press, Boca Raton, 2000.

[9] F. C. M. Lau and C. K. Tse, "Chaos-Based Digital Communication Systems", Springer, Berlin, 2003.

[10] A. Baranovsky and D. Daems, "Int. J. Bifurcation Chaos Appl.", Sci, volumen 5, pag 1585, 1995.

[11] Wiliam Ditto y Toshinori, ”Principles and aplication of chaotic systems”, Comunication of the ACM, volumen 38, nº 11, 1995

[12] Cambel, A.B. “Applied Chaos Theory: A Paradigm for Complexity”, Academic Press, San Diego, California, 1993.

[13] Juliany, J., and Vose, M.D. “The genetic algorithm fractal”, In The Fifth International Conference on Genetic Algorithms (Urbana- Champaign, 1993), pag. 639.

[14] Ott, E., Sauer, T., and Yorke, J.A. “Coping with Chaos: Analysis of Chaotic Data and the Exploitation of Chaotic Systems”, Wiley, New York 1994.

[15] Francisco J. Ceballos, “C/C++ Curso de programación”, Ra-Ma, 2ª edición, 2002

Capítulo 13

13 Apéndice

Medidas completas para 5 muestras

Mediadas completas para 11 muestras

13. Apéndice

13.1 Medidas completas para 5 muestras

13.1.1 Variación del factor de cuantificación

Se fijará el valor de población a 200, el número de valores *double* a 1100, la relación señal ruido a 8 dB, se realizarán 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación serán del 1%.

De esta forma se darán valores al factor de cuantificación entre 3 y 5 indicando los resultados de las muestras obtenidos, así como también el error y mostrándose en una gráfica en función de la relación señal a ruido para cada muestra.

Q=3

Vector Y

[0.000000, 0.324435, 0.569934, 0.884134, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000018094

Q=4

Vector Y

[0.000000, 0.273557, 0.531142, 0.833031, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026771

Q=5

Vector Y

[0.000000, 0.244414, 0.512677, 0.797241, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000029555

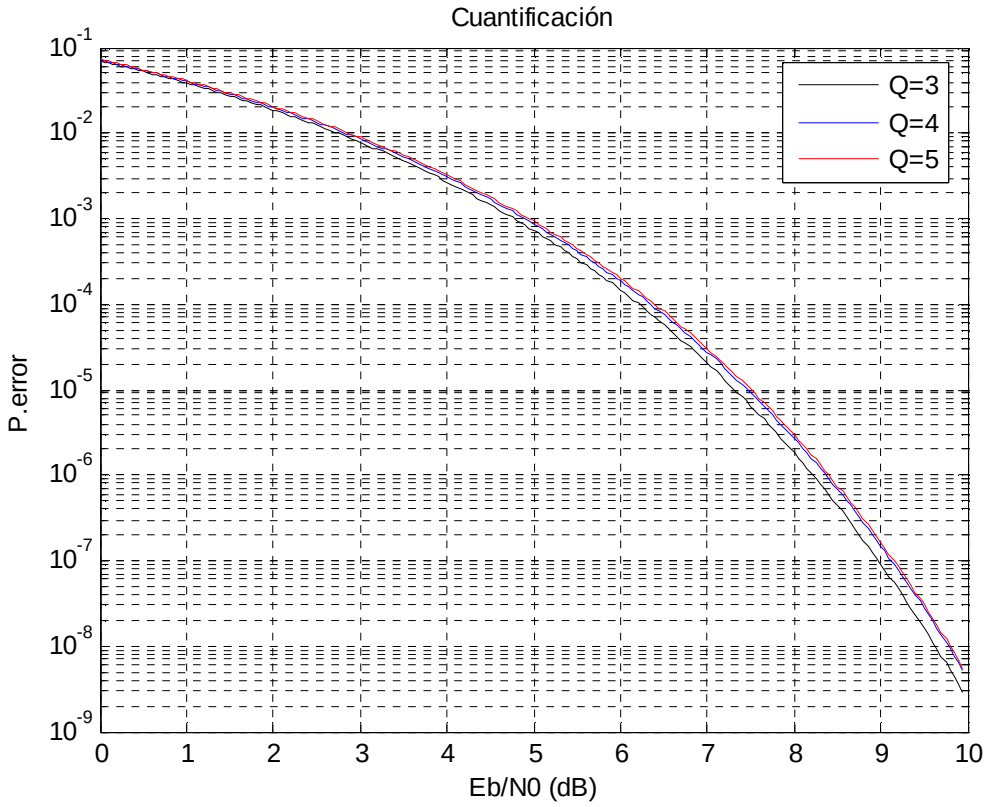

Fig. 51 Cota de la probabilidad de error respecto a la función de cuantificación, 5 muestras

13.1.2 **Variación de la relación señal ruido**

Se ha fijado el factor de cuantificación a 4, realizando 150 iteraciones en el algoritmo genético, con 1100 valores *double* y una probabilidad y porcentaje de mutación del 1% para 200 poblaciones.

Eb/no=3 dB

Vector Y

[0.000000, 0.292895, 0.552002, 0.854847, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0084531308

Eb/no=4 dB

Vector Y

[0.000000, 0.289276, 0.550319, 0.854755, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0030101731

Eb/no=5 dBs

Vector Y

[0.000000, 0.287138, 0.548592, 0.854225, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0008536854

Eb/no=6 dB

Vector Y

[0.000000, 0.284636, 0.545259, 0.852765, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0001826911

Eb/no=7 dBs

Vector Y

[0.000000, 0.284372, 0.542315, 0.850840, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000275722

Eb/no=8 dB

Vector Y

[0.000000, 0.275043, 0.532392, 0.832256, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026762

Eb/no=9 dB

Vector Y

[0.000000, 0.268582, 0.528110, 0.818325, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000001479

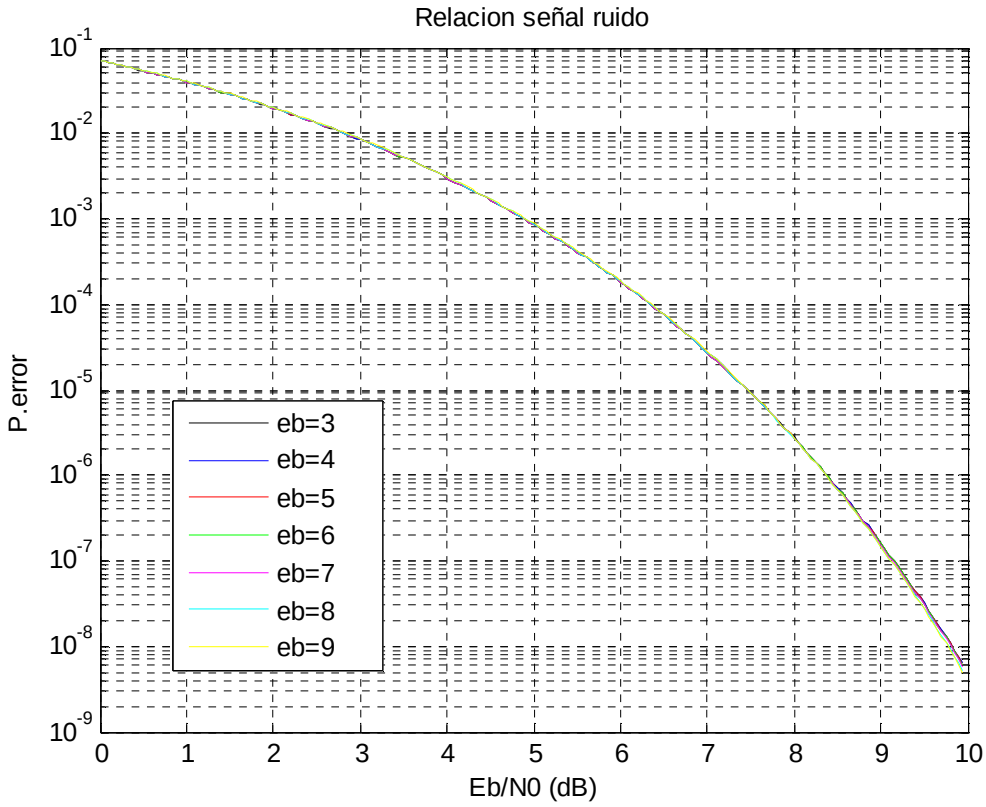

Fig. 52 Cota de la probabilidad de error respecto a la relación señal ruido, 5 muestras

13.1.3 **Variación de la población**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, realizando 150 iteraciones, con 1100 valores *double* y una probabilidad y porcentaje de mutación del 1% dando diversos valores de población.

TPOB =200

Vector Y

[0.000000, 0.273253, 0.530717, 0.830507, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026769

TPOB=100

Vector Y

[0.000000, 0.275942, 0.533569, 0.833513, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026763

TPOB=50

Vector Y

[0.000000, 0.275366, 0.533248, 0.833194, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026761

TPOB=10

Vector Y

[0.000000, 0.233707, 0.489082, 0.776686, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000031228

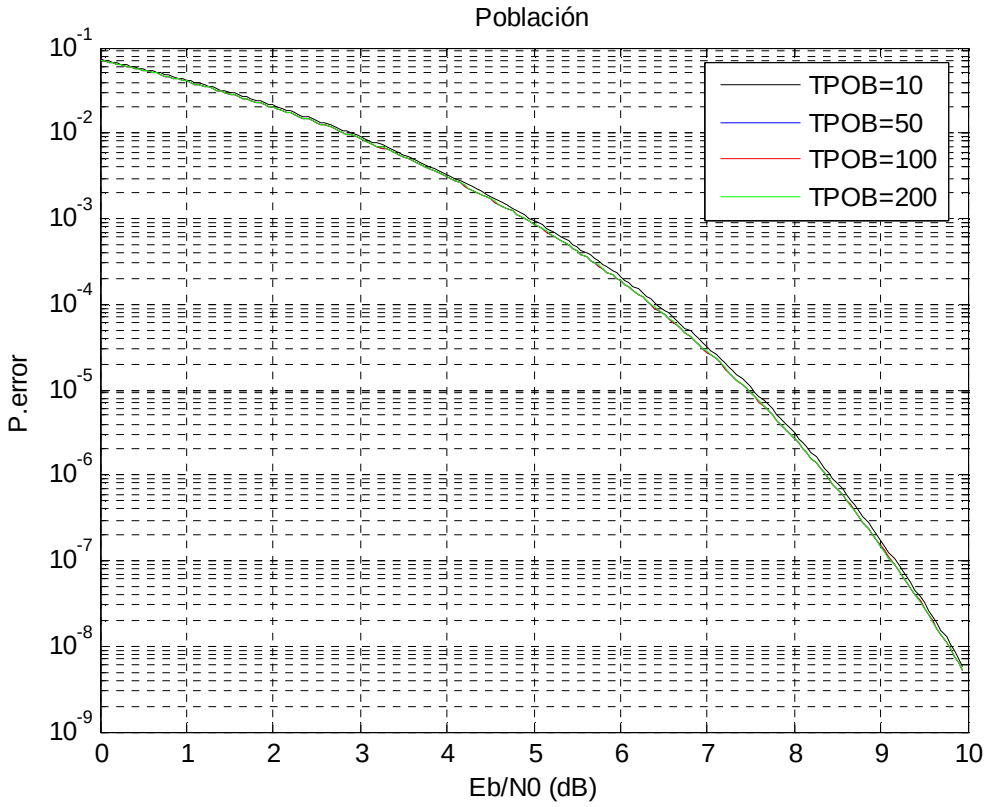

Fig. 53 Cota de la probabilidad de error respecto a la población, 5 muestras

13.1.4 **Variación de números double**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, realizando 150 iteraciones, para una probabilidad y porcentaje de mutación del 1% para un valor de población de 200, dando diversas cantidades de valores al parámetro *numeros_double.*

numeros_double=11

Vector Y

[0.000000, 0.276284, 0.533785, 0.835283, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026764

numeros_double=110

Vector Y

[0.000000, 0.276415, 0.533713, 0.834282, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026763

numeros_double=1100

Vector Y

[0.000000, 0.275039, 0.531932, 0.832042, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026764

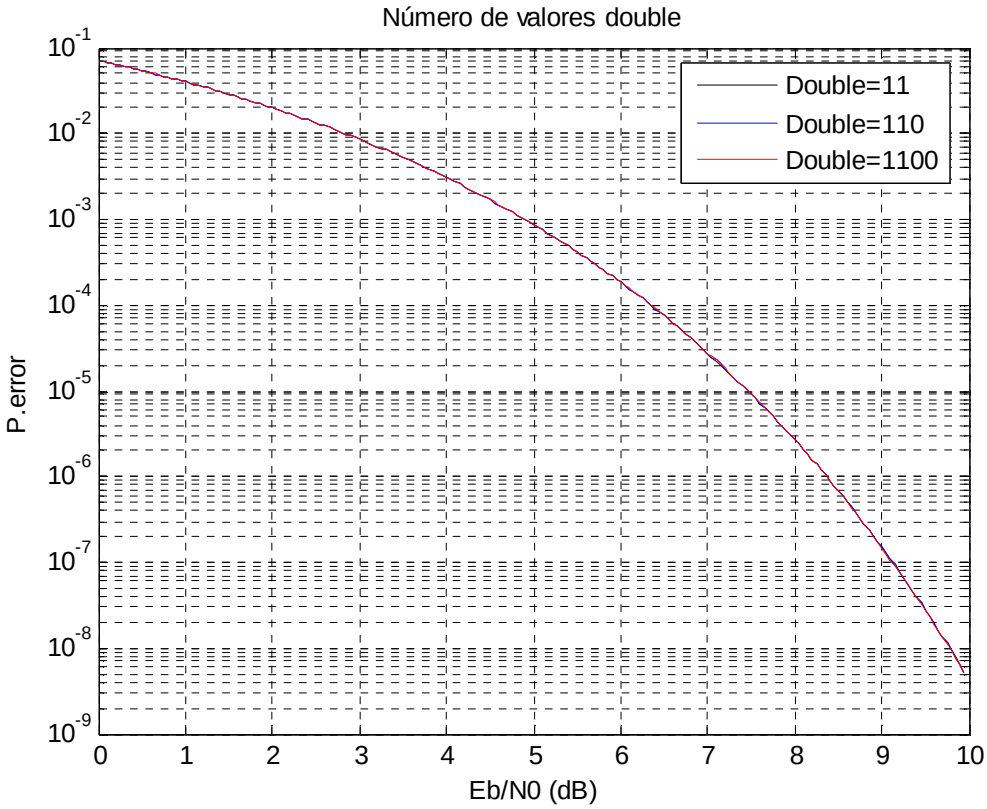

Fig. 54 Cota de la probabilidad de error respecto a los números *double*, 5 muestras

13.1.5 **Variación del número de iteraciones en el algoritmo genético**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, introduciendo 1100 valores *double* y una probabilidad y porcentaje de mutación del 1% para un valor de población de 200, realizando diversas iteraciones en el algoritmo evolutivo.

Iteraciones=10

Vector Y

[0.000000, 0.313521, 0.595063, 0.861248, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000040506

Iteraciones=50

Vector Y

[0.000000, 0.275236, 0.533615, 0.833291, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026763

Iteraciones=150

Vector Y

[0.000000, 0.276055, 0.532565, 0.834616, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026768

Iteraciones=300

Vector Y

[0.000000, 0.275364, 0.532452, 0.832498, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026763

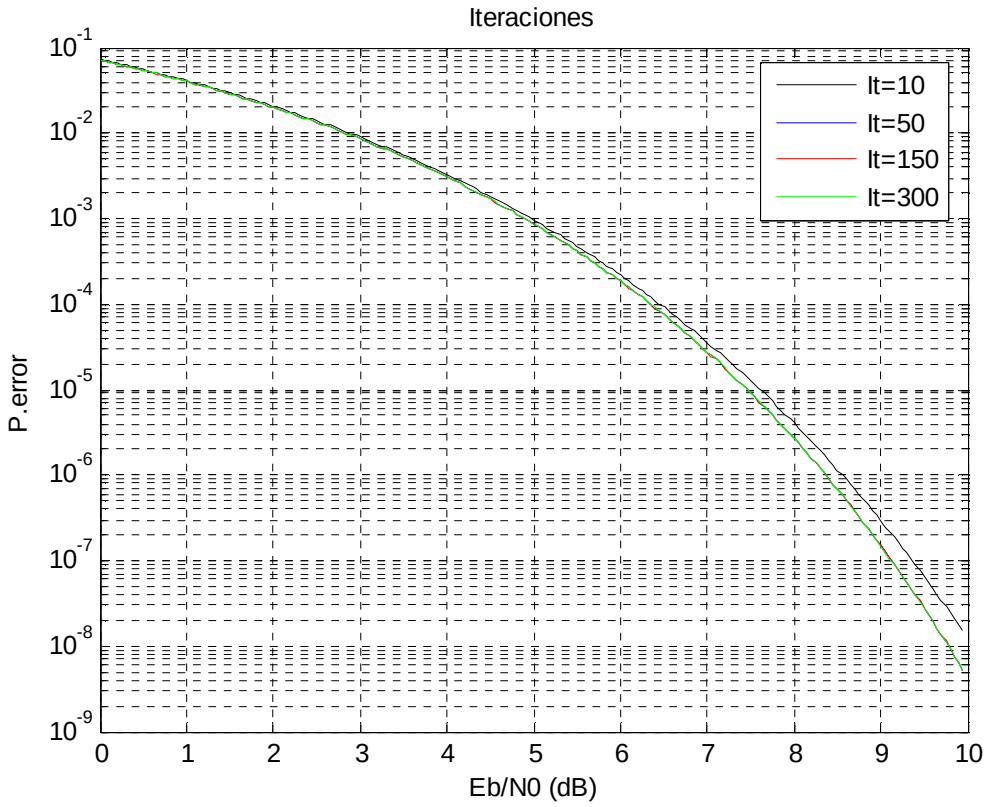

Fig. 55 Cota de la probabilidad de error respecto a las iteraciones, 5 muestras

13.1.6 **Variación de la probabilidad de mutación**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, para 150 iteraciones del algoritmo genético, introduciendo 1100 valores *double* con un porcentaje de mutación del 1%, para un valor de población de 200 modificando el probabilidad de mutación.

Probabilidad =10%

Vector Y

[0.000000, 0.275408, 0.532973, 0.832858, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026761

Probabilidad =40%

Vector Y

[0.000000, 0.275277, 0.533180, 0.833264, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026761

Probabilidad =65%

Vector Y

[0.000000, 0.274943, 0.531777, 0.834107, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026771

Probabilidad =100%

Vector Y

[0.000000, 0.274937, 0.532158, 0.832169, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026762

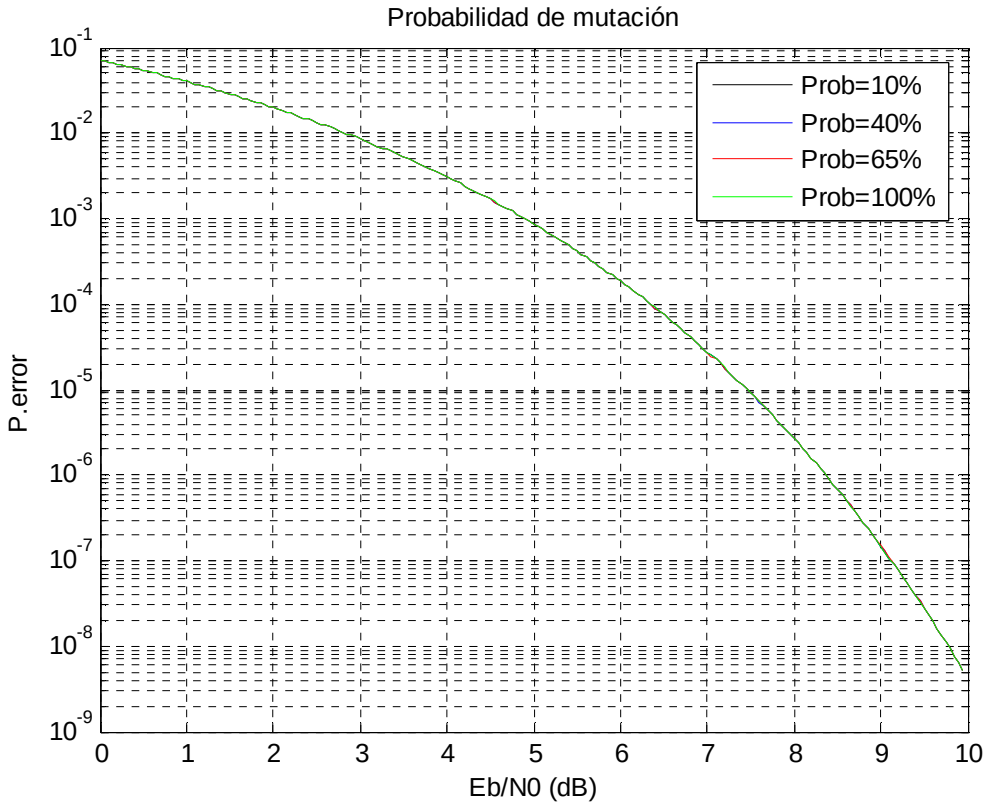

Fig. 56 Cota de la probabilidad de error respecto a la probabilidad de mutación, 5 muestras

13.1.7 **Variación del porcentaje de mutación**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, para 150 iteraciones del algoritmo genético, introduciendo 1100 valores *double* con una probabilidad de mutación del 100%, para un valor de población de 200 modificando el porcentaje de mutación dentro de esta probabilidad.

Porcentaje=10%

Vector Y

[0.000000, 0.274937, 0.532158, 0.832169, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026762

Porcentaje=40%

Vector Y

[0.000000, 0.277287, 0.534828, 0.836343, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026770

Porcentaje=65%

Vector Y

[0.000000, 0.273787, 0.532024, 0.834495, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026774

Porcentaje=100%

Vector Y

[0.000000, 0.270054, 0.530499, 0.829289, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000026787

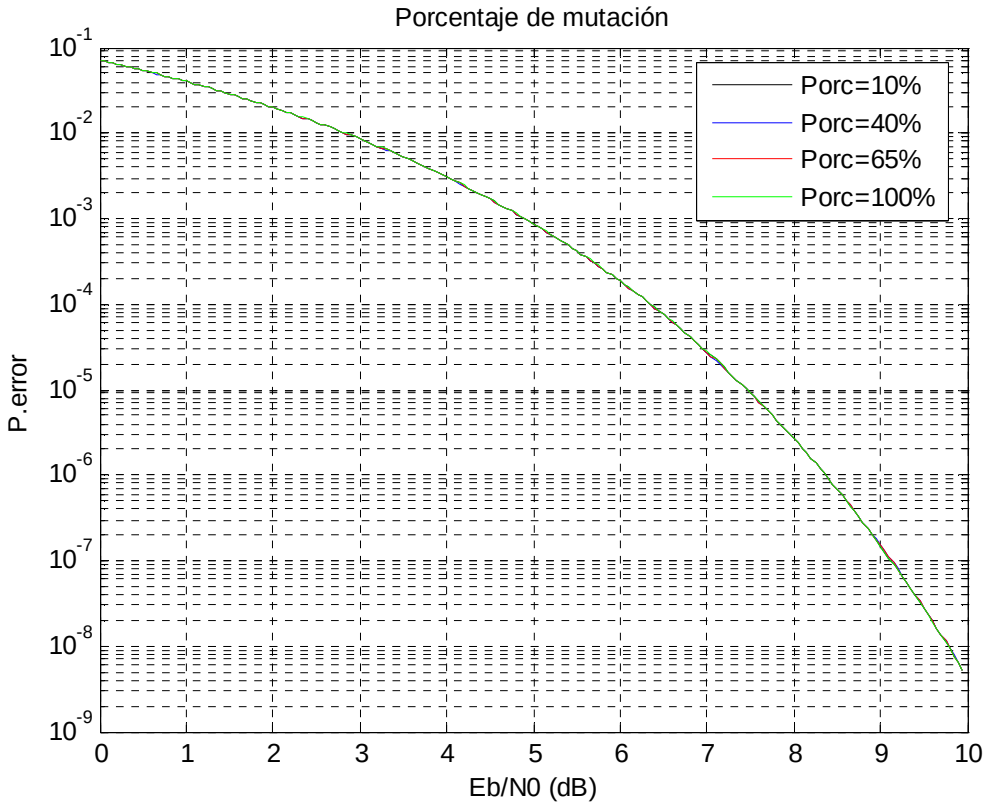

Fig. 57 Cota de la probabilidad de error respecto al porcentaje de mutación, 5 muestras

13.2 Medidas completas para 11 muestras:

13.2.1 Variación del factor de cuantificación

Se fijará el valor de población a 200, el número de *double* a 1100, el ruido a 8 dB, se realizarán 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación serán del 1%.

De esta forma se darán valores al factor de cuantificación entre 3 y 6, y se incorporarán las representaciones de los vectores de la función *g(x)* y una gráfica con la probabilidad de error en función de la relación señal a ruido para cada valor.

Q=3

Vector Y

[0.000000, 0.005671, 0.013059, 0.014603, 0.327950, 0.327950, 0.674588, 0.677857, 0.750224, 0.778452, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000018264

De esta forma obtendremos la siguiente representación gráfica:

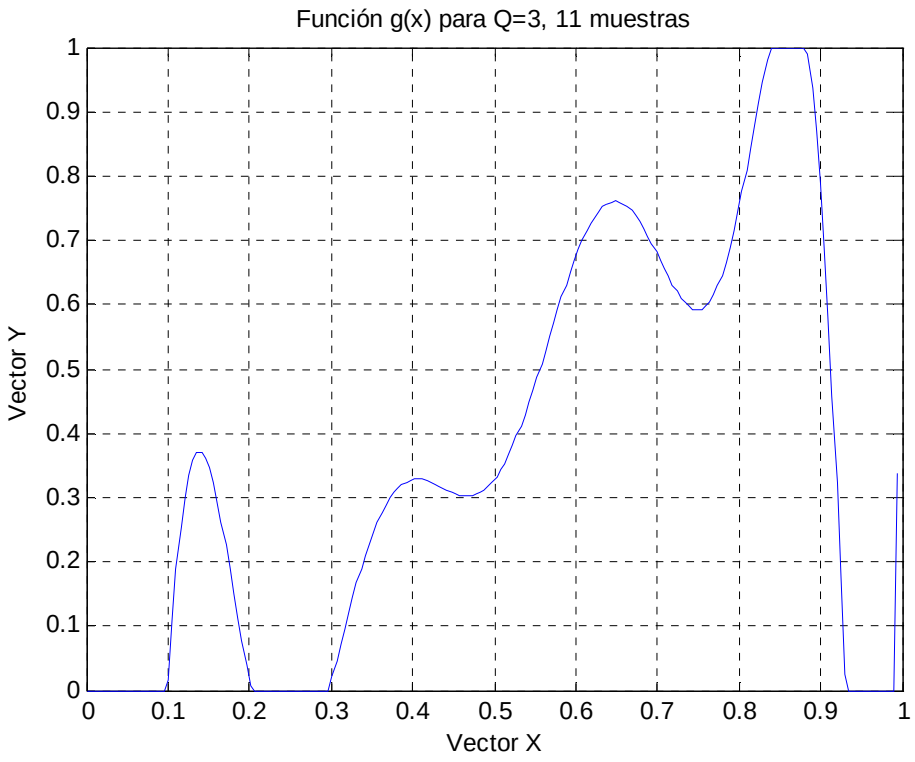

Fig. 58 Función *g(x)* para Q=3, 11 muestras

Q=4

Vector Y

[0.000000, 0.086453, 0.154107, 0.154107, 0.369250, 0.371427, 0.586234, 0.586234, 0.660904, 0.684016, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000029187

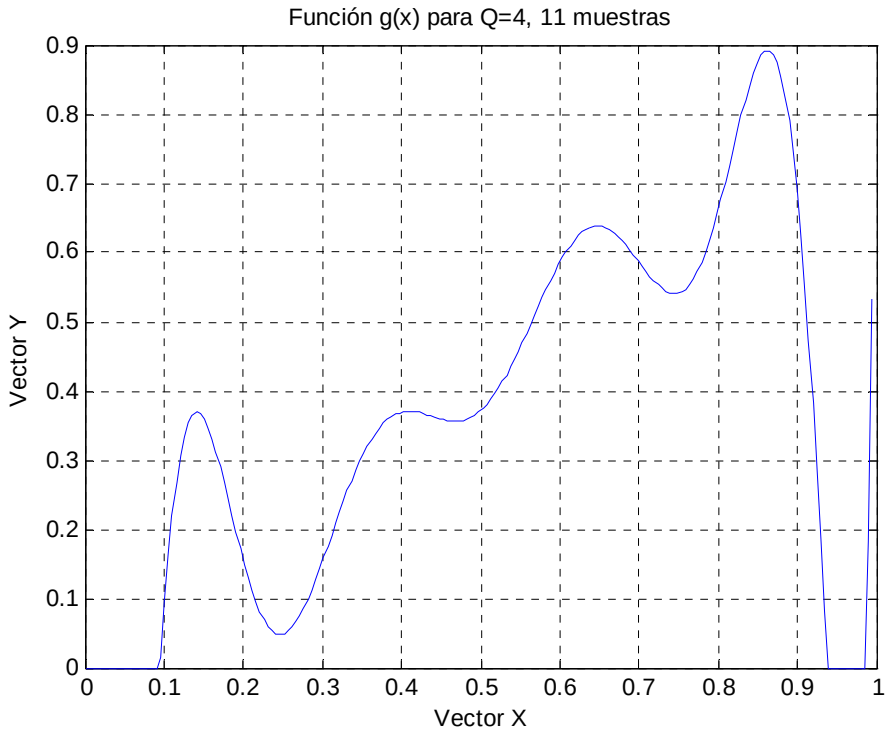

Fig. 59 Función *g(x)* para Q=4, 11 muestras

Q=5

Vector Y

[0.000000, 0.009287, 0.188536, 0.188536, 0.385847, 0.385847, 0.647659, 0.647659, 0.690889, 0.790998, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000017217

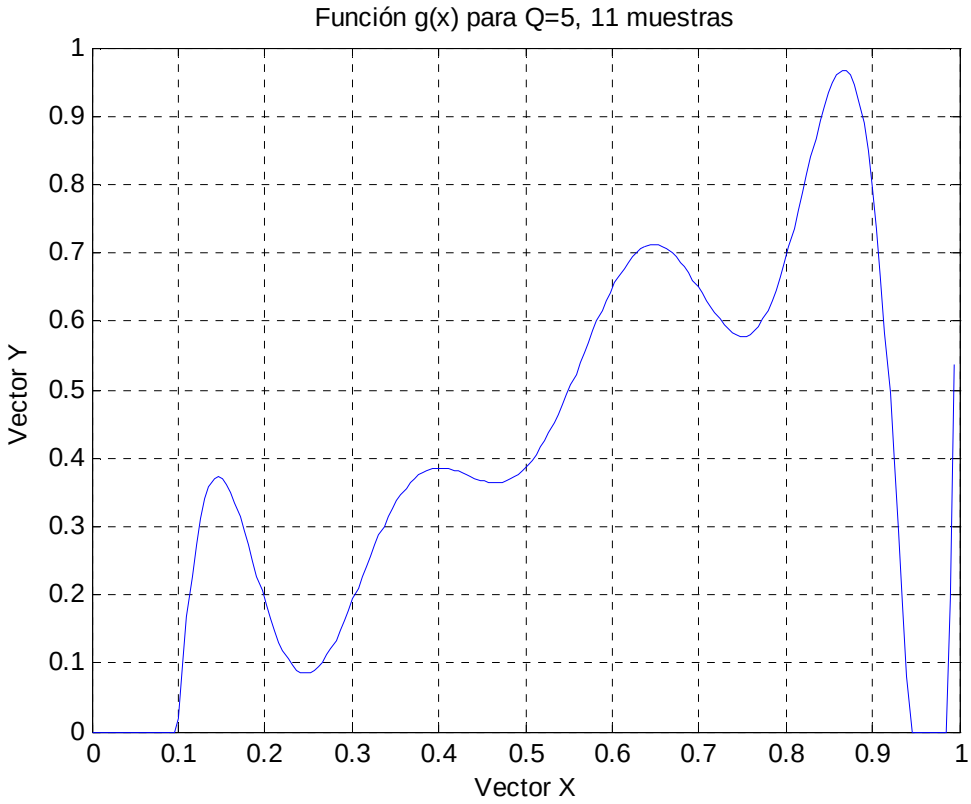

Fig. 60 Función *g(x)* para Q=5, 11 muestras

Q=6

Vector Y

[0.000000, 0.076102, 0.106779, 0.106779, 0.357823, 0.398607, 0.554437, 0.624793, 0.624793, 1.000000, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000044372

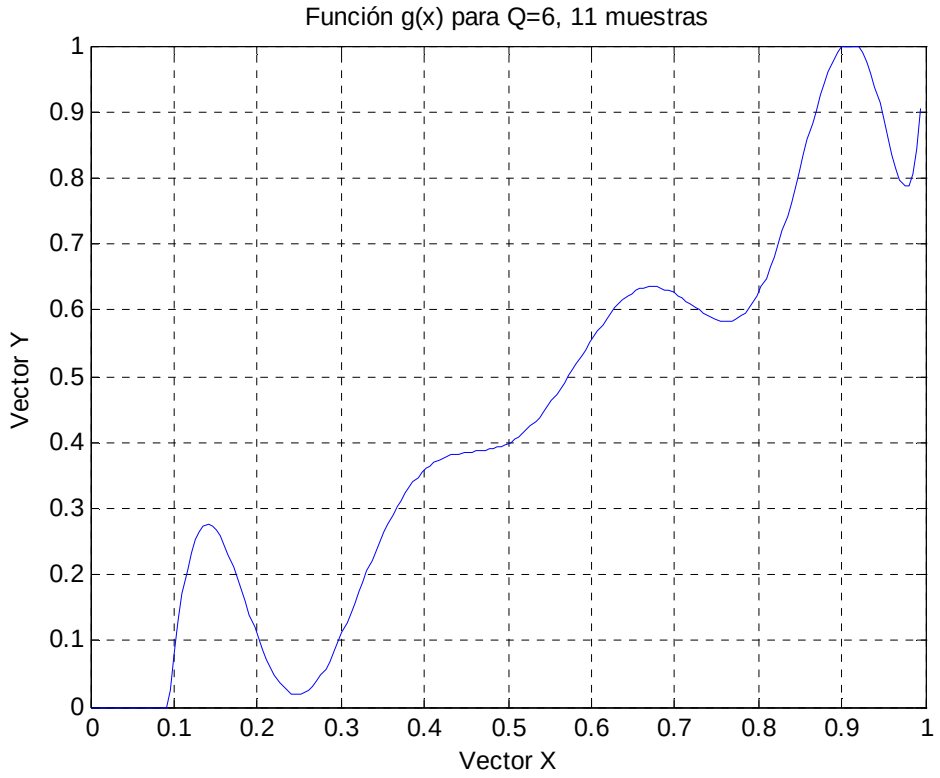

Fig. 61 Función *g(x)* para Q=6, 11 muestras

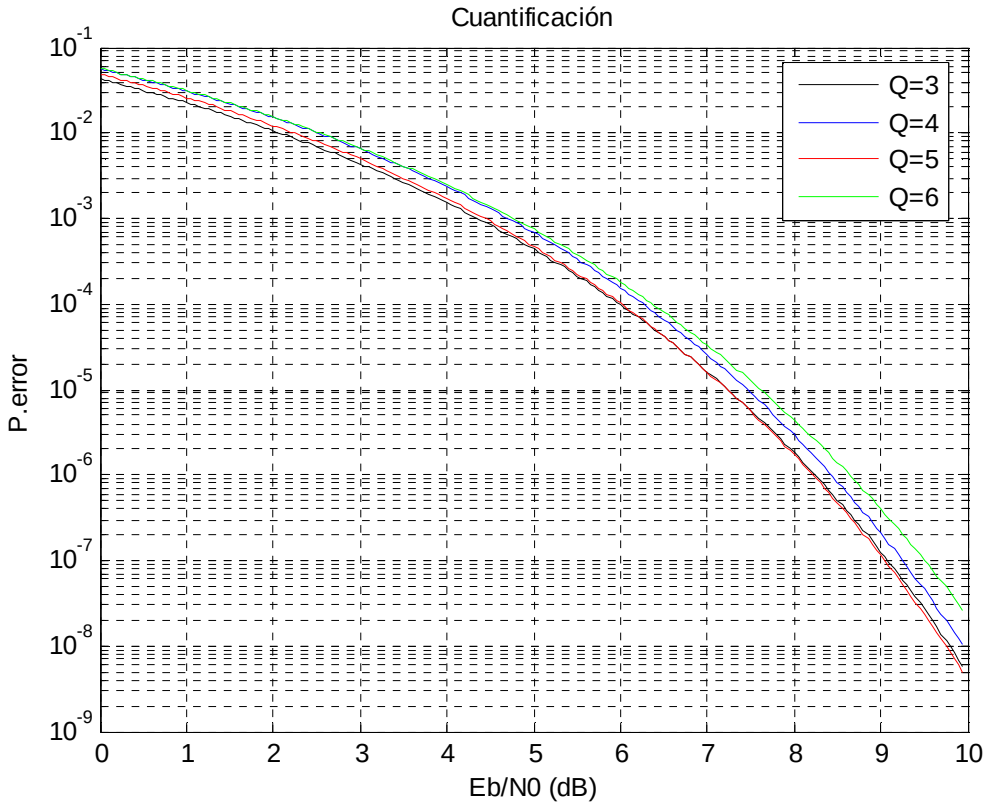

Fig. 62 Cota de la probabilidad de error respecto a la función de cuantificación, 11 muestras

13.2.2 **Variación de la relación señal ruido**

Se fijará el valor de población a 200, el número de valores *double* a 1100, el factor de cuantificación a 4, se realizarán 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación serán del 1%.

Eb/no=3 dB

Vector Y

[0.000000, 0.007640, 0.008883, 0.008909, 0.340457, 0.340457, 0.616596, 0.617684, 0.617684, 0.712644, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0060541435

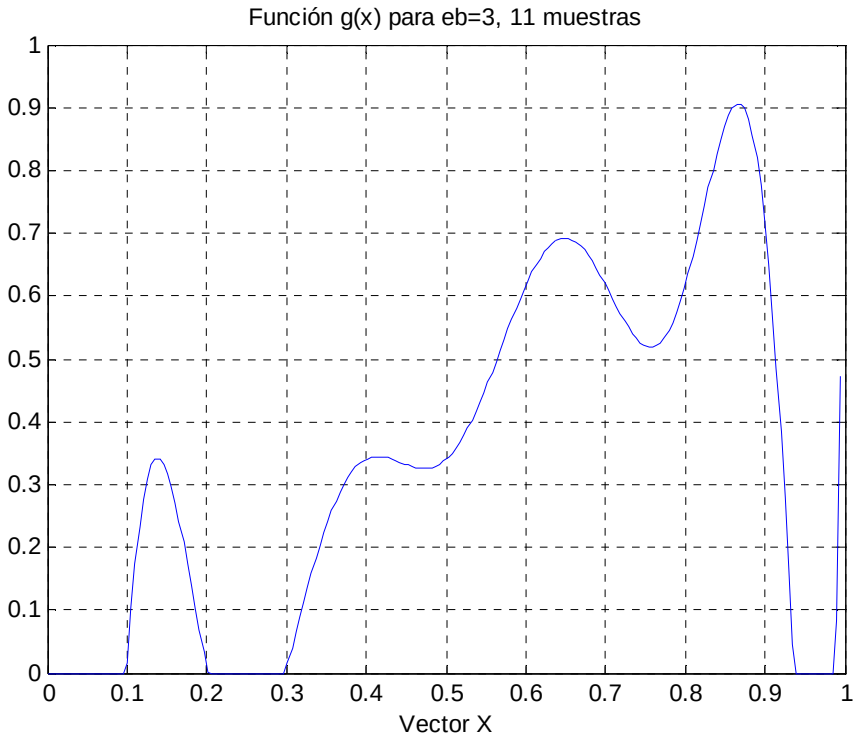

Fig. 63 Función *g(x)* para eb=3, 11 muestras

Eb/no=4 dB

Vector Y

[0.000000, 0.057478, 0.057956, 0.057956, 0.304117, 0.304142, 0.595146, 0.595146, 0.599455, 0.691832, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0024121164

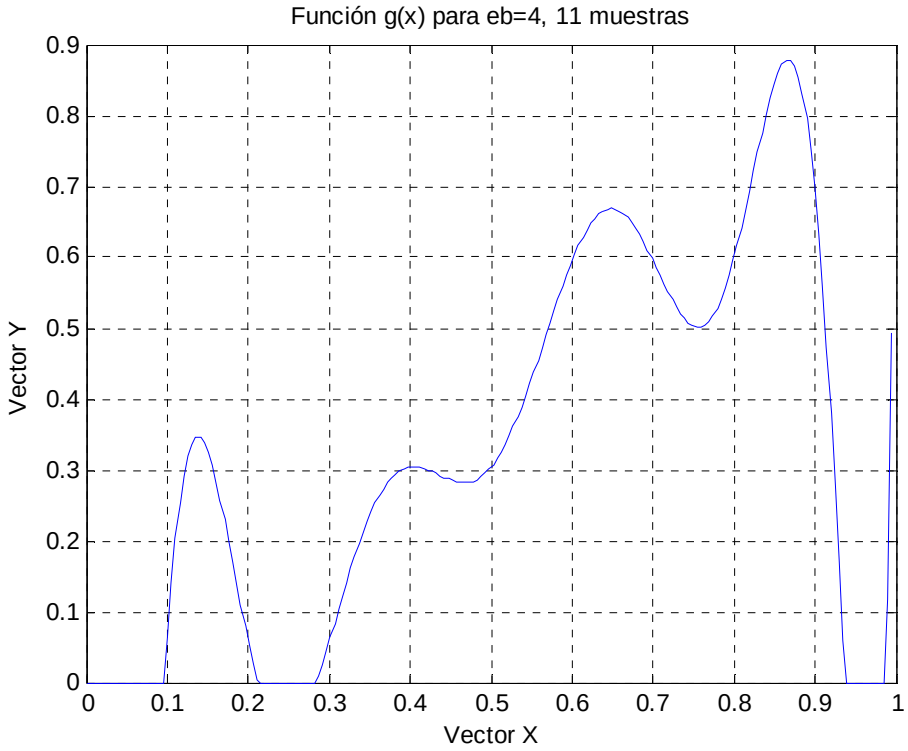

Fig. 64 Función *g(x)* para eb=4, 11 muestras

Eb/no=5 dB

Vector Y

[0.000000, 0.012389, 0.124360, 0.126202, 0.408453, 0.410043, 0.657155, 0.657548, 0.686154, 0.745775, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0005046768

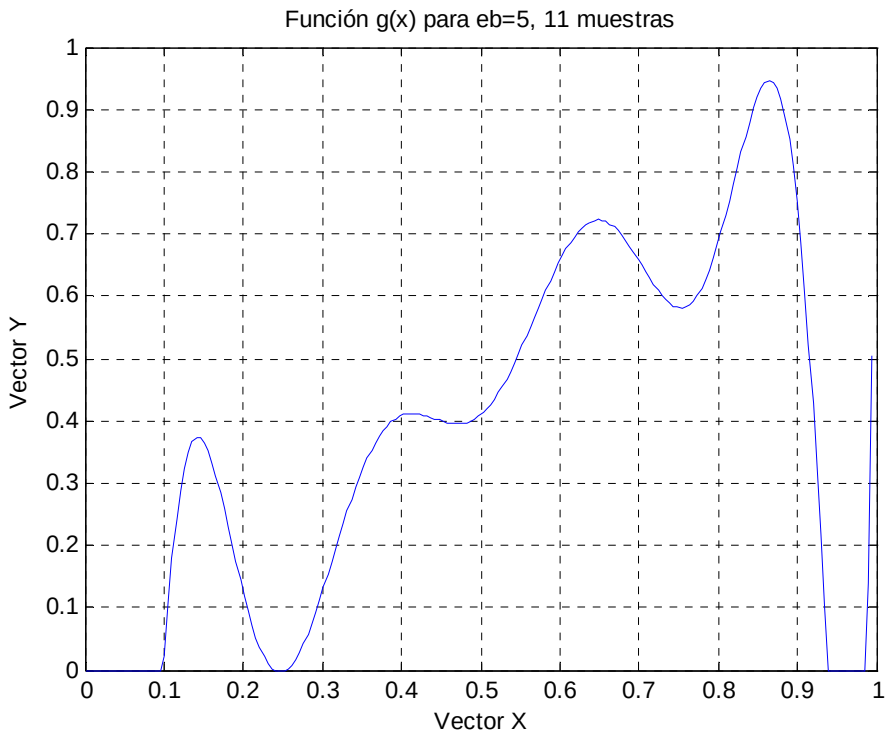

Fig. 65 Función *g(x)* para eb=5, 11 muestras

Eb/no=6 dB

Vector Y

[0.000000, 0.087463, 0.122214, 0.147427, 0.147427, 0.361764, 0.368256, 0.631080, 0.634220, 0.980571, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0001880654

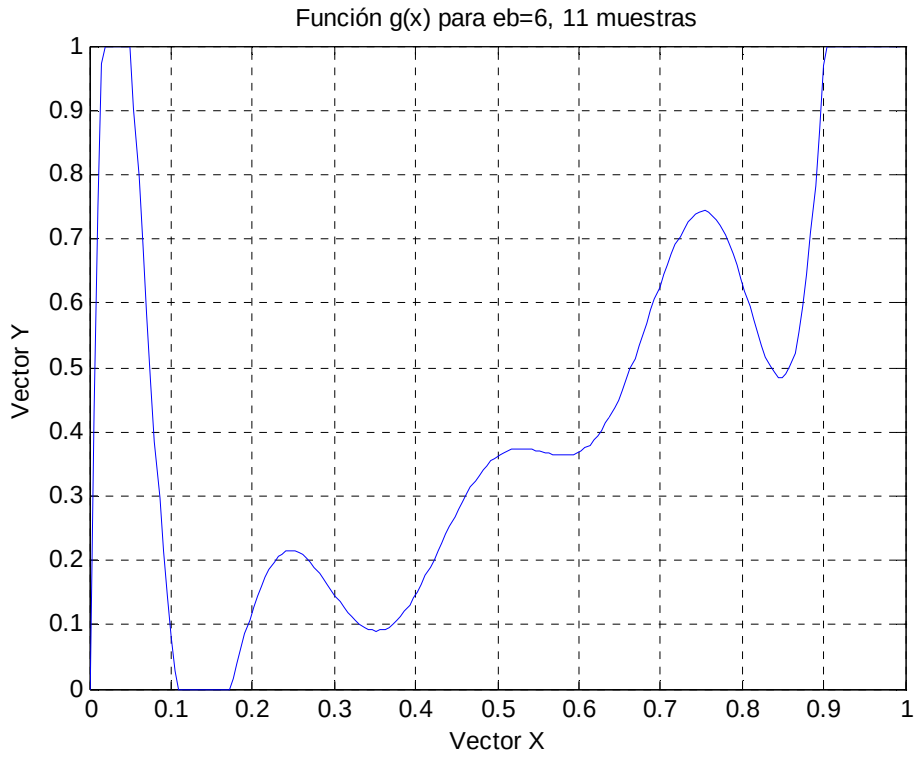

Fig. 66 Función *g(x)* para eb=6, 11 muestras

Eb/no=7 dB

Vector Y

[0.000000, 0.073831, 0.073831, 0.307653, 0.307653, 0.309445, 0.345384, 0.731320, 0.758497, 0.992873, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000305779

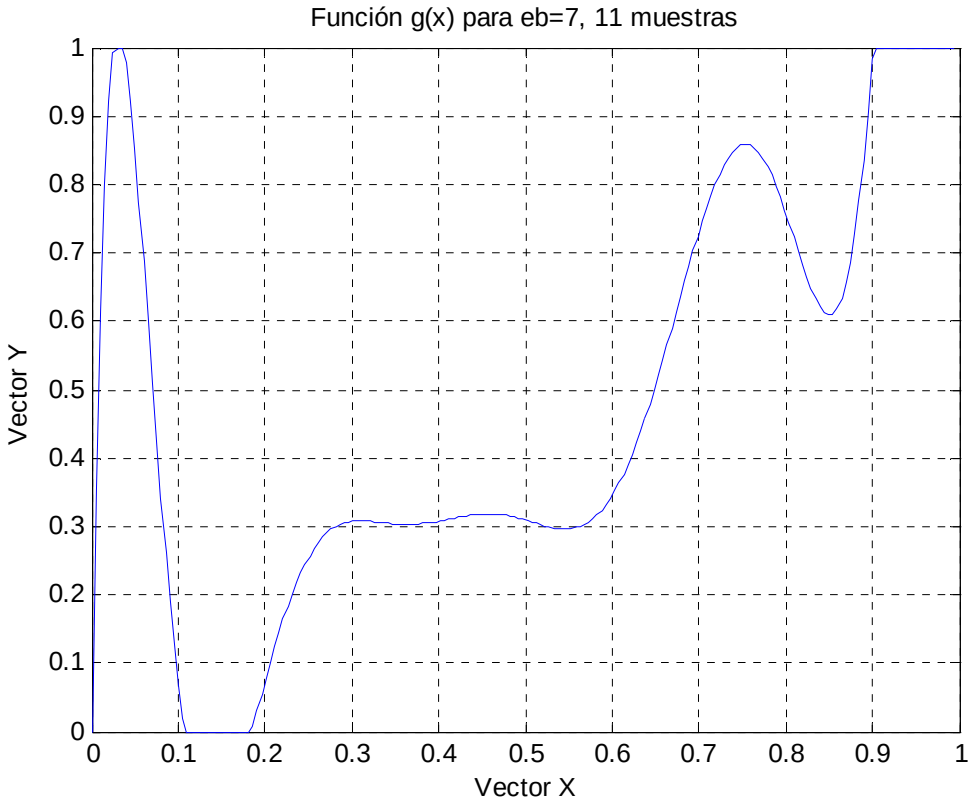

Fig. 67 Función *g(x)* para eb=7, 11 muestras

Eb/no=8 dB

Vector Y

[0.000000, 0.098864, 0.098864, 0.098864, 0.322757, 0.323613, 0.566444, 0.576336, 0.650259, 0.709602, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000052009

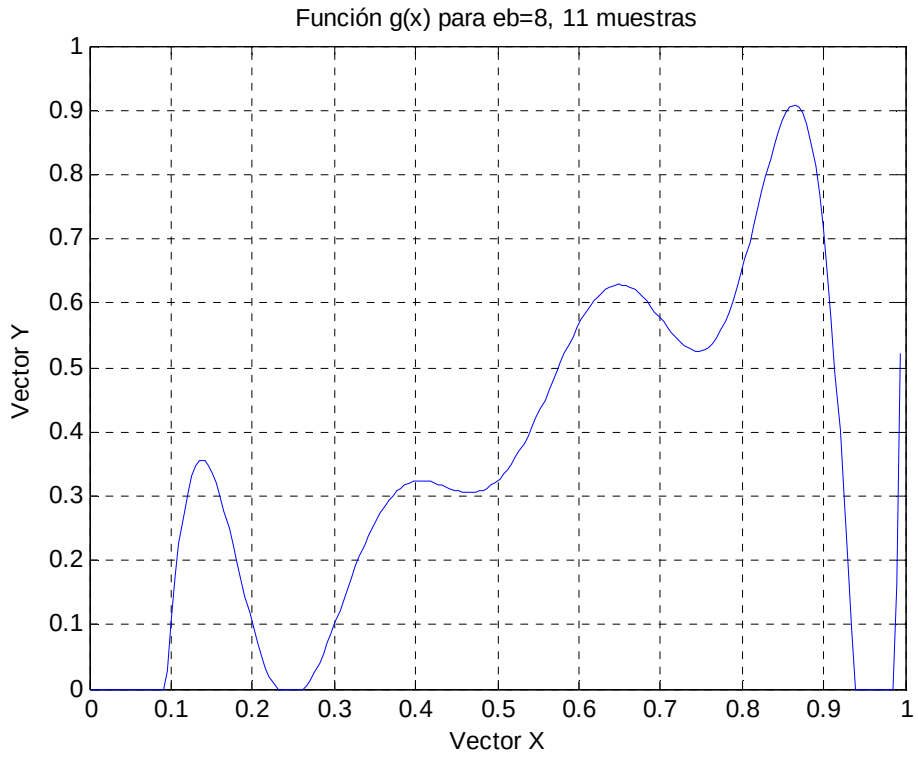

Fig. 68 Función *g(x)* para eb=8, 11 muestras

Eb/no=9 dB

Vector Y

[0.000000, 0.041474, 0.062989, 0.098876, 0.098876, 0.352002, 0.365503, 0.590145, 0.591029, 0.895972, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000007721

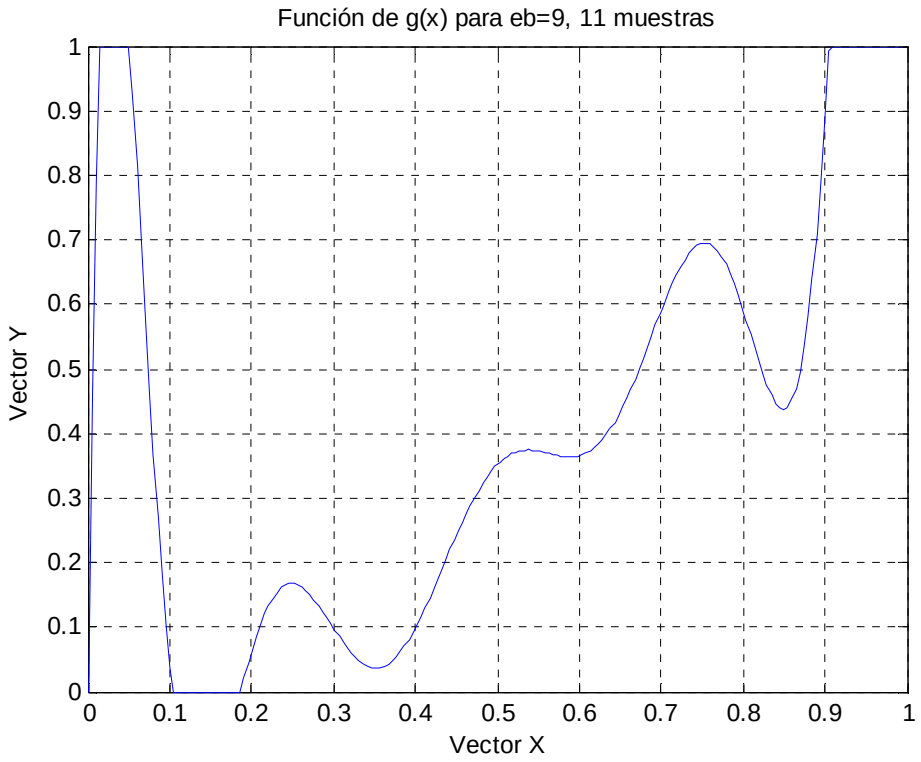

Fig. 69 Función *g(x)* para eb=9, 11 muestras

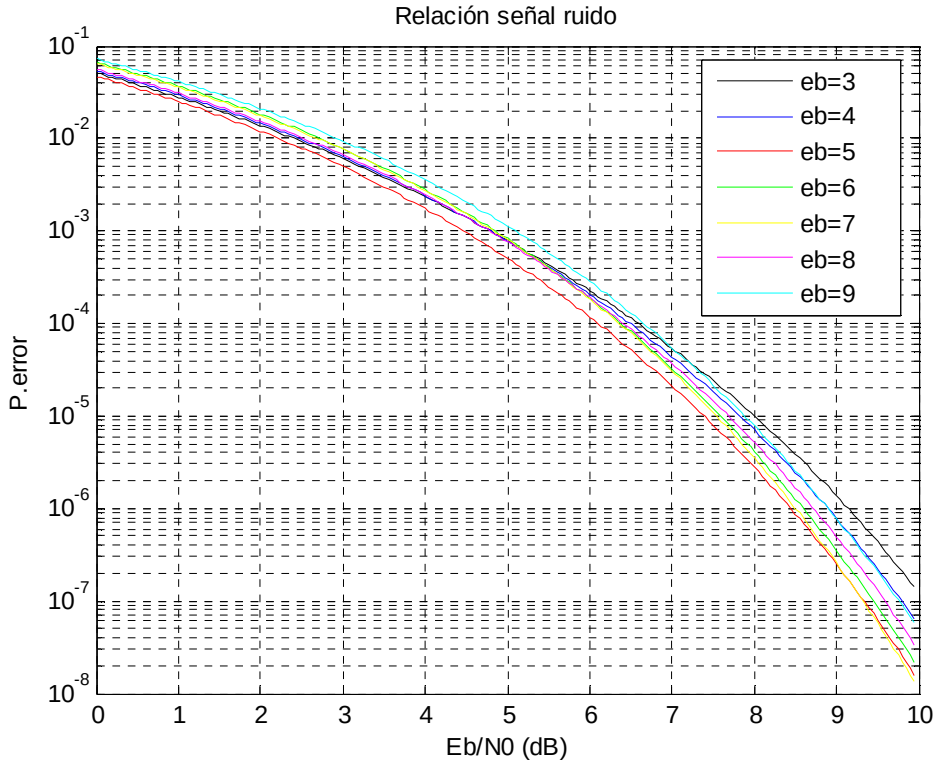

Fig. 70 Cota de la probabilidad de error respecto a la relación señal ruido, 11 muestras

13.2.3 **Variación la población**

Se fijará el valor del factor de cuantificación a 4, el número de valores *double* a 1100, la relación señal ruido a 8 dB, se realizarán 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación serán del 1%.

TPOB =200

Vector Y

[0.000000, 0.112359, 0.112359, 0.112359, 0.315153, 0.316594, 0.599053, 0.599053, 0.650854, 0.656655, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000038635

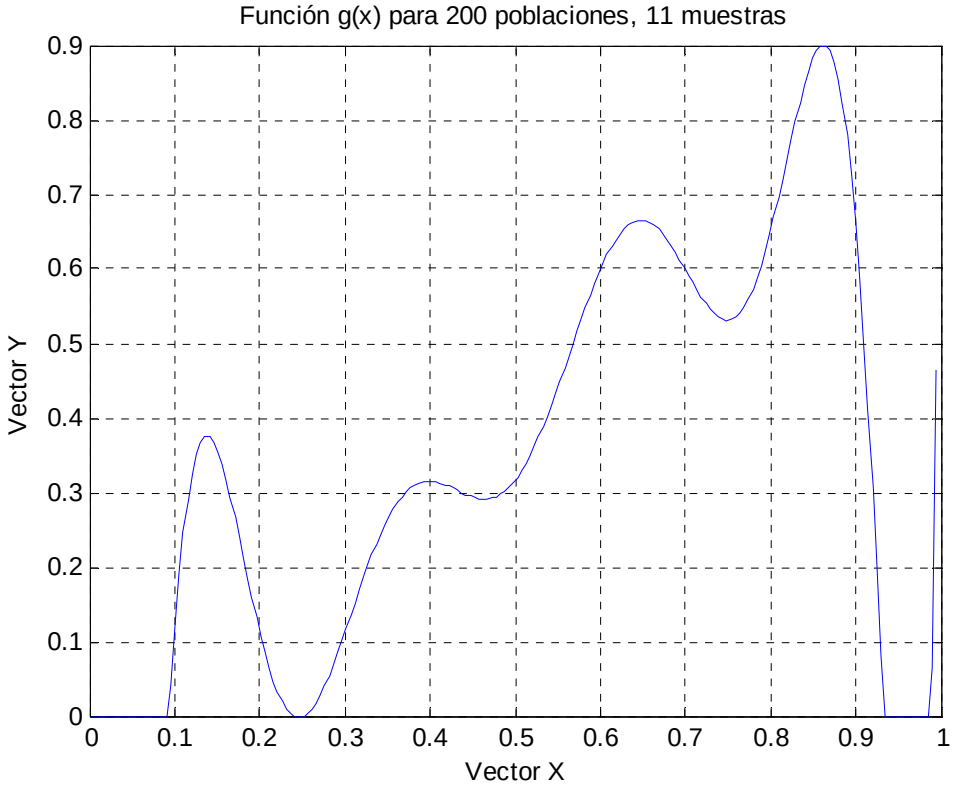

Fig. 71 Función *g(x)* para 200 poblaciones, 11 muestras

TPOB=100

Vector Y

[0.000000, 0.032103, 0.041895, 0.042026, 0.316597, 0.316597, 0.594872, 0.594872, 0.609030, 0.623181, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000081347

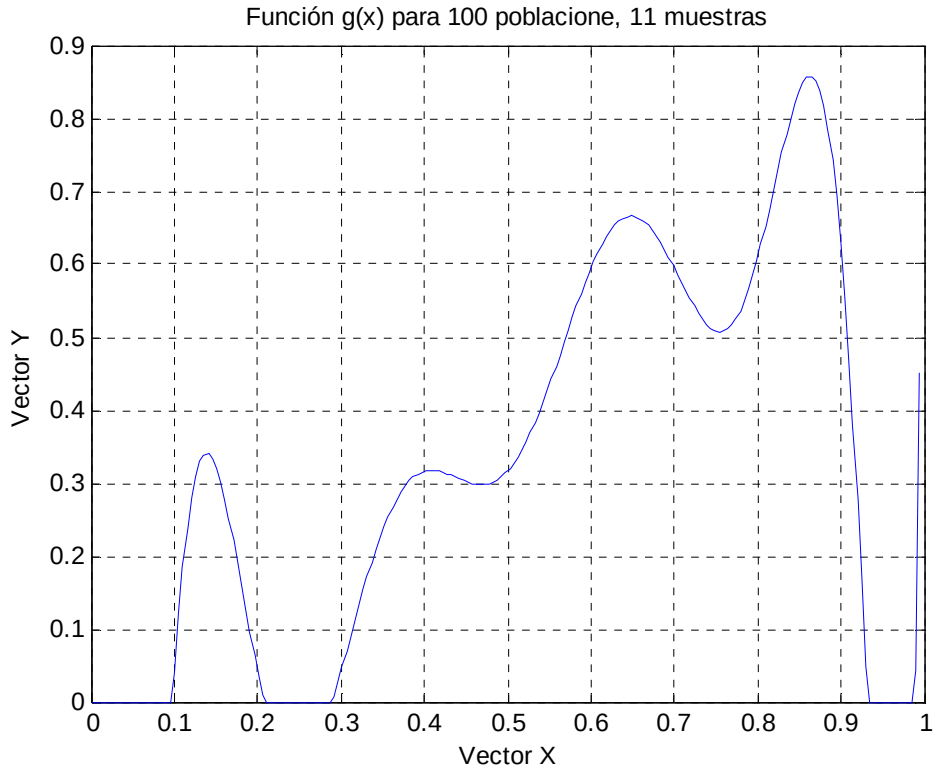

Fig. 72 Función *g(x)* para 100 poblaciones, 11 muestras

TPOB=50

Vector Y

[0.000000, 0.065551, 0.070940, 0.070940, 0.237776, 0.237776, 0.425597, 0.425597, 0.468665, 0.498910, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000716813

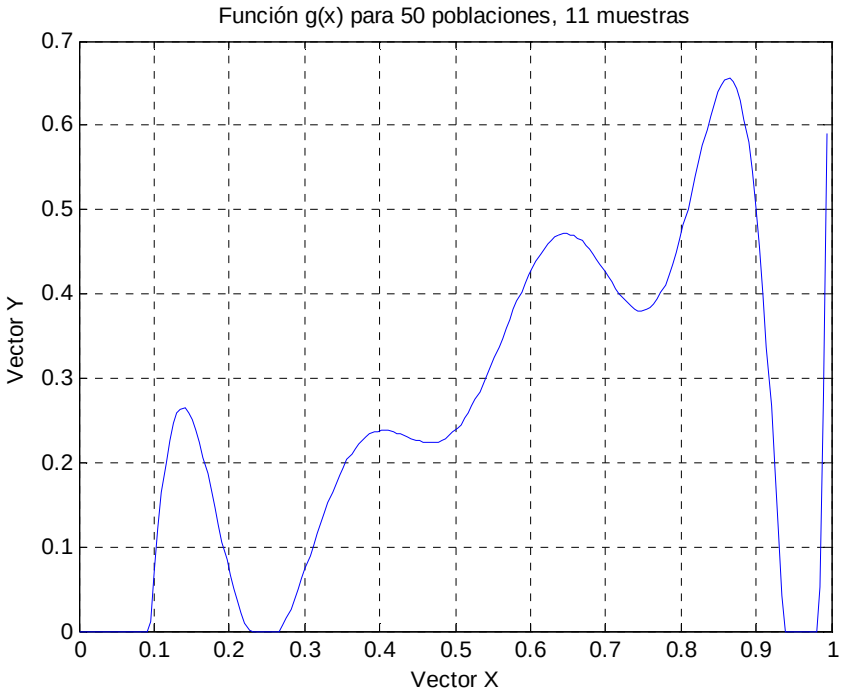

Fig. 73 Función *g(x)* para 50 poblaciones, 11 muestras

TPOB=10

Vector Y

[0.000000, 0.004400, 0.005976, 0.005982, 0.028087, 0.028087, 0.030193, 0.307741, 0.322091, 0.726028, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0274905063

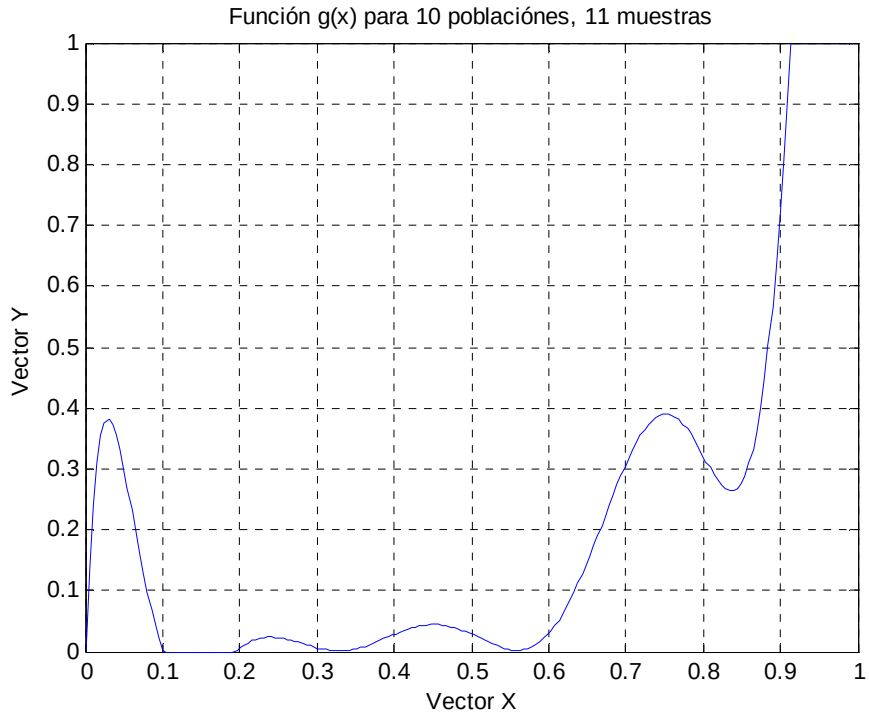

Fig. 74 Función *g(x)* para 10 poblaciones, 11 muestras

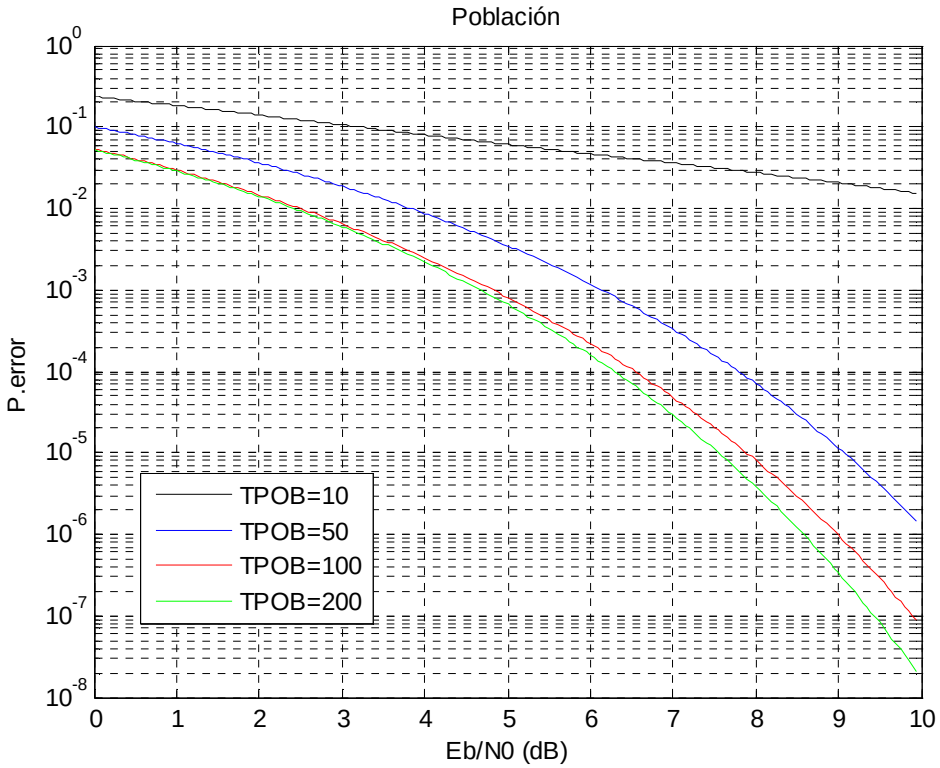

Fig. 75 Cota de la probabilidad de error respecto a la población, 11 muestras

13.2.4 **Variación de valores *double***

Se fijará el valor del factor de cuantificación a 4, la relación señal ruido a 8 dB, se realizaran 150 iteraciones en el algoritmo genético y tanto la probabilidad como el porcentaje de mutación será del 1% para 200 poblaciones.

numeros_double=11

Vector Y

[0.000000, 0.139457, 0.148829, 0.149381, 0.354414, 0.354414, 0.610886, 0.610886, 0.684270, 0.684270, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000024980

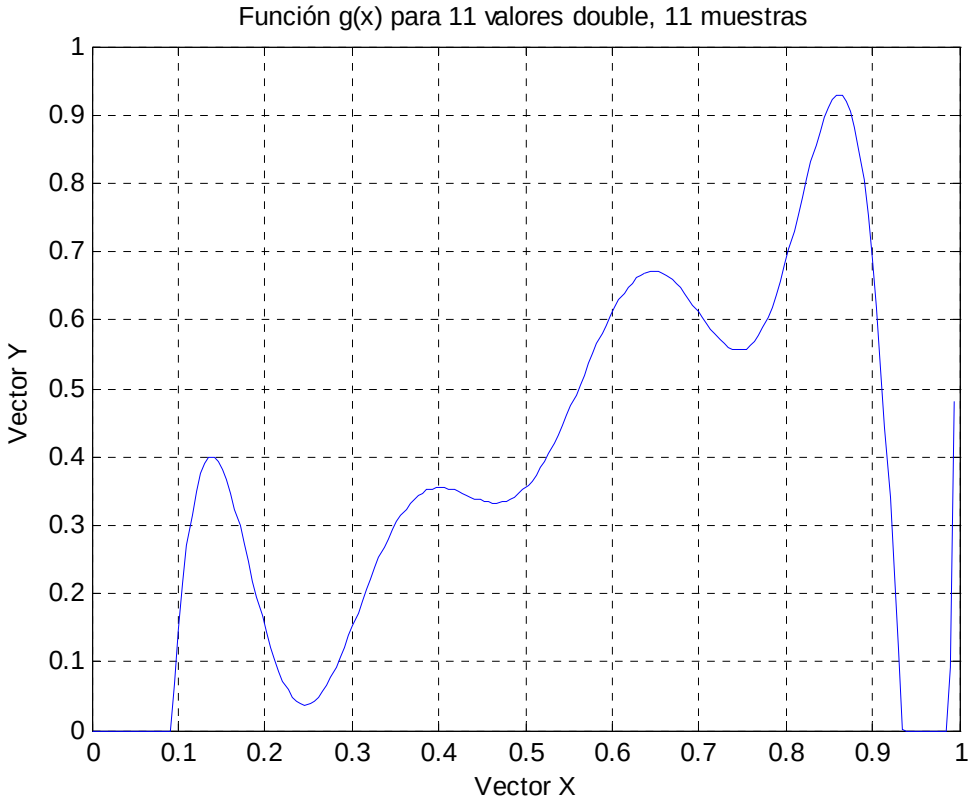

Fig. 76 Función *g(x)* para 11 valores *double*, 11 muestras

numeros_double=110

Vector Y

[0.000000, 0.010295, 0.070773, 0.071150, 0.071150, 0.372398, 0.372398, 0.581424, 0.581424, 0.991350, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.000006855

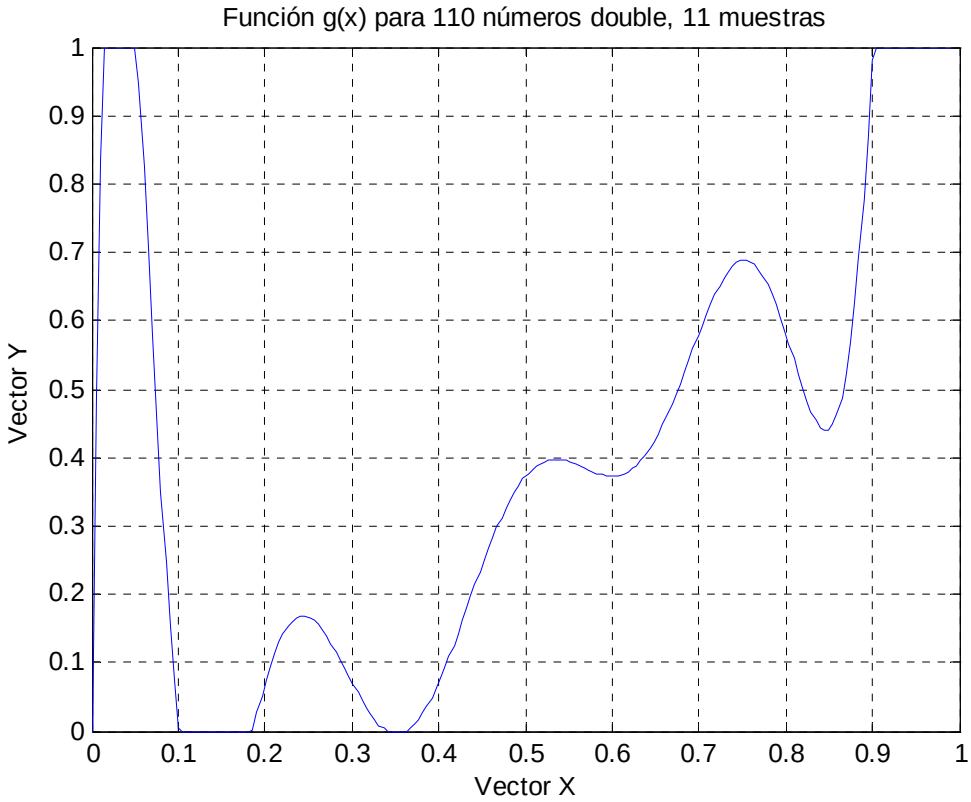

Fig. 77 Función *g(x)* para 110 valores *double*, 11 muestras

numeros_double=1100

Vector Y

[0.000000, 0.010984, 0.060551, 0.066810, 0.066810, 0.355581, 0.355581, 0.549134, 0.549134, 0.906141, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000092175

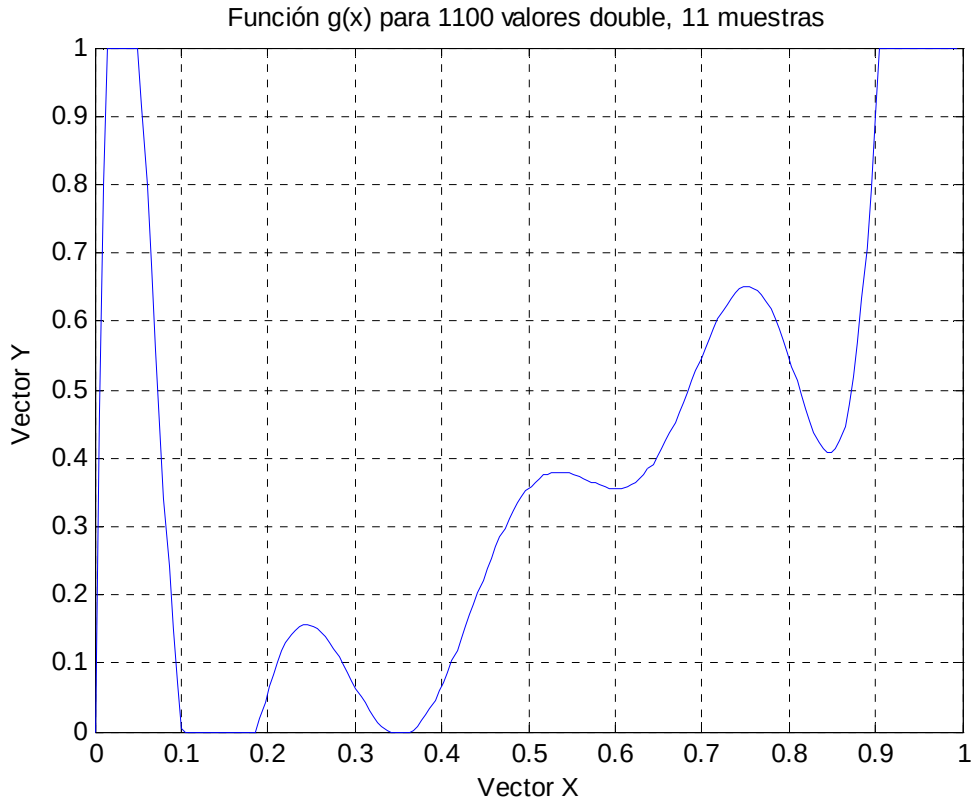

Fig. 78 Función *g(x)* para 1100 valores *double*, 11 muestras

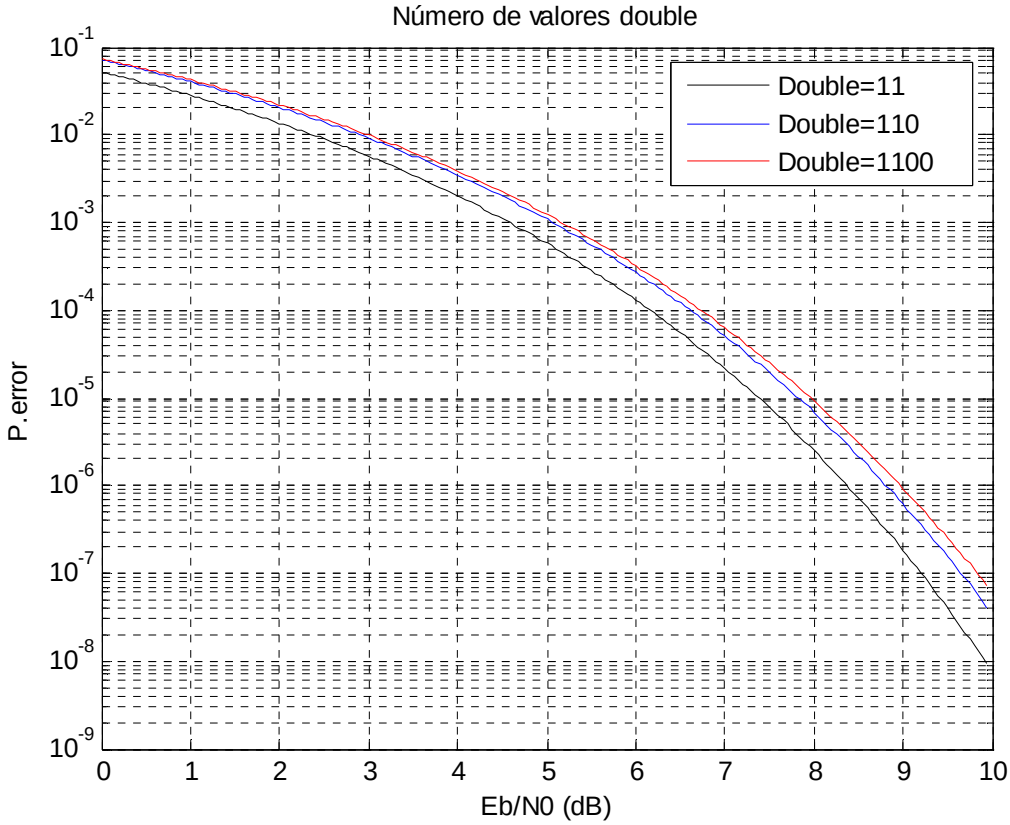

Fig. 79 Cota de la probabilidad de error respecto al número de valores double, 11 muestras

13.2.5 **Variación del número de iteraciones en el algoritmo genético**

Se fijará el valor del factor de cuantificación a 4, la relación señal ruido a 8 dB, se asignan 1100 valores *double* y tanto la probabilidad como el porcentaje de mutación serán del 1% para 200 poblaciones.

Iteraciones=10

Vector Y

[0.000000, 0.054846, 0.054846, 0.054846, 0.054846, 0.280930, 0.280930, 0.417349, 0.443163, 0.772499, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000706371

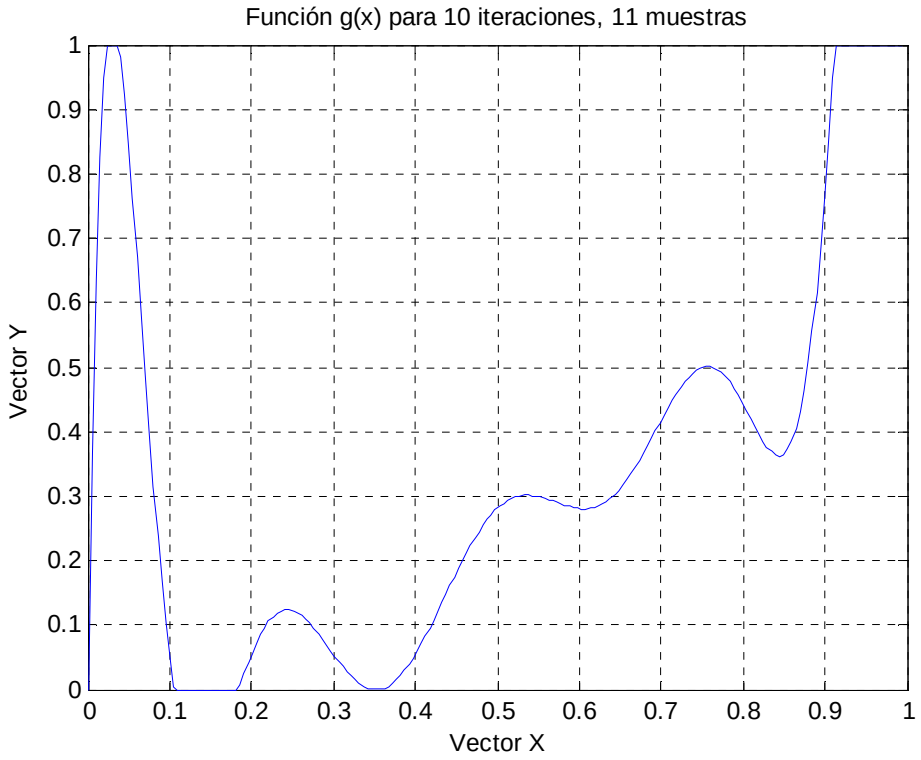

Fig. 80 Función *g(x)* para 10 iteraciones, 11 muestras

Iteraciones=50

Vector Y

[0.000000, 0.000807, 0.002179, 0.213948, 0.213948, 0.362191, 0.362191, 0.648381, 0.850985, 0.958312, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000087068

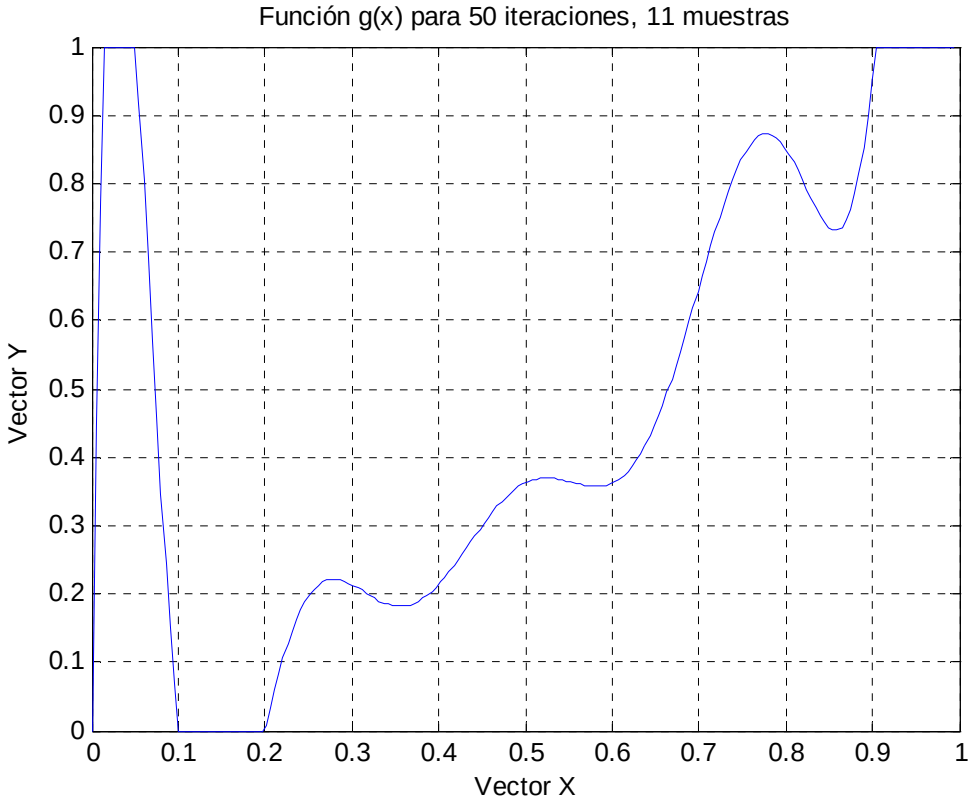

Fig. 81 Función *g(x)* para 50 iteraciones, 11 muestras

Iteraciones=150

Vector Y

[0.000000, 0.005440, 0.161553, 0.161553, 0.413265, 0.413265, 0.598971, 0.600113, 0.684541, 0.697313, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000020901

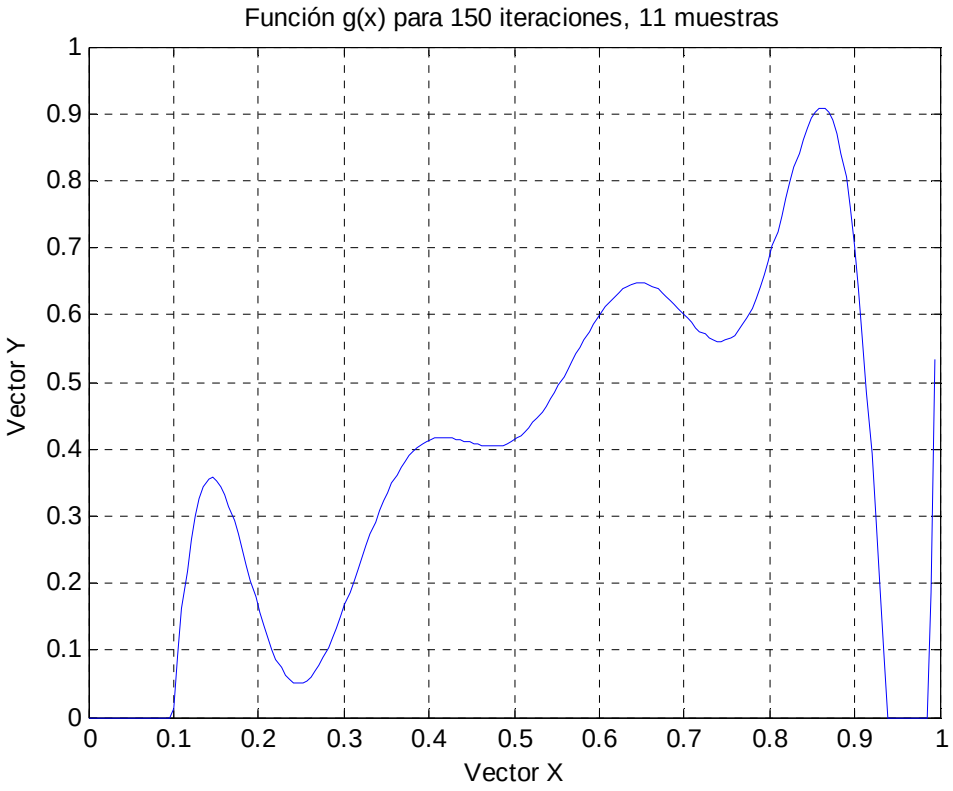

Fig. 82 Función *g(x)* para 150 iteraciones, 11 muestras

Iteraciones=300

Vector Y

[0.000000, 0.086415, 0.099563, 0.349014, 0.349014, 0.379936, 0.379936, 0.786241, 0.903218, 0.997196, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000015506

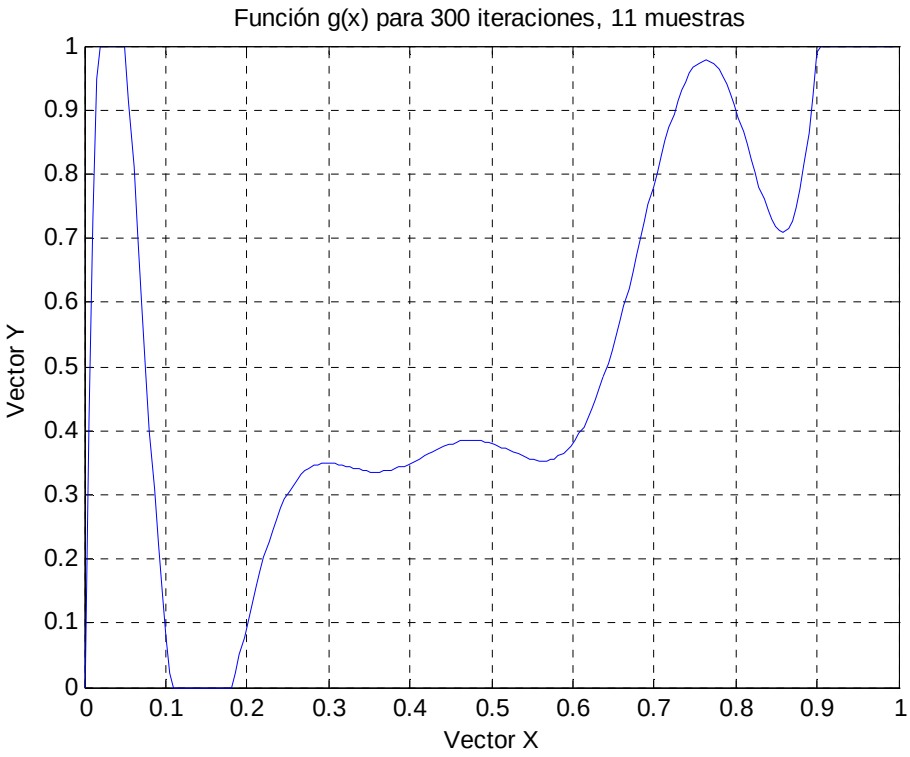

Fig. 83 Función *g(x)* para 300 iteraciones, 11 muestras

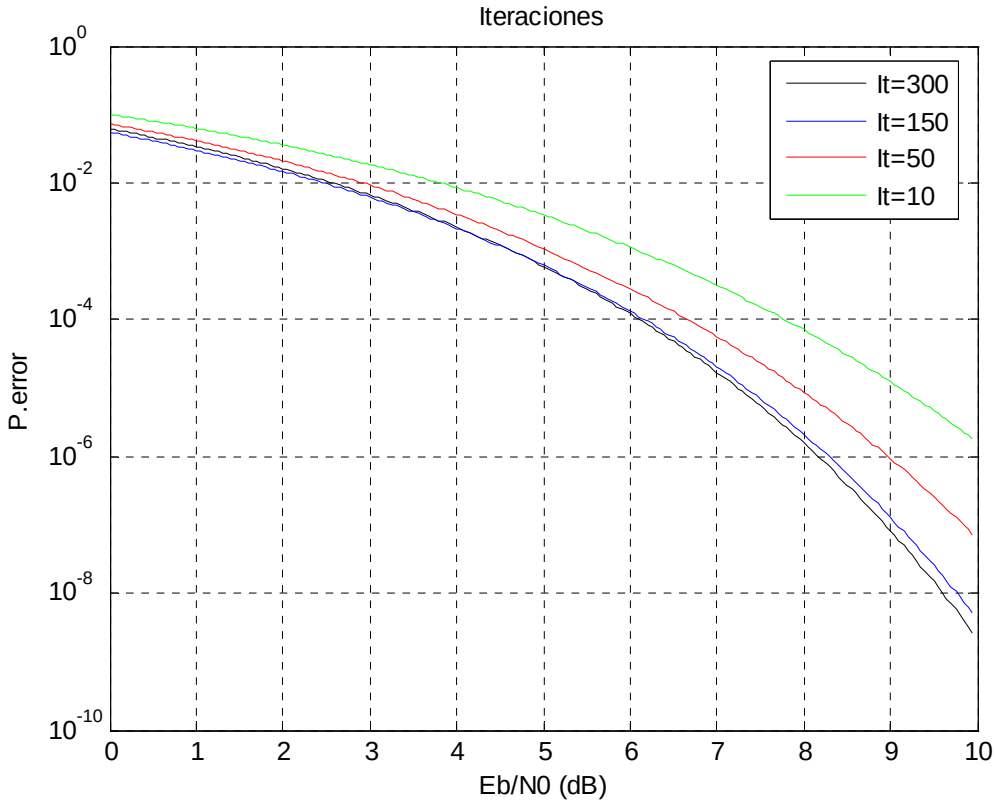

Fig. 84 Cota de la probabilidad de error respecto a las iteraciones, 11 muestras

13.2.6 **Variación de la probabilidad de mutación**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, para 150 iteraciones del algoritmo genético, introduciendo 1100 valores *double* con un porcentaje de mutación del 1%, para un valor de población de 200 modificando la probabilidad de mutación.

Probabilidad =10%

Vector Y

[0.000000, 0.027184, 0.127526, 0.127526, 0.348116, 0.348116, 0.588473, 0.589436, 0.616201, 0.747831, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.000003878

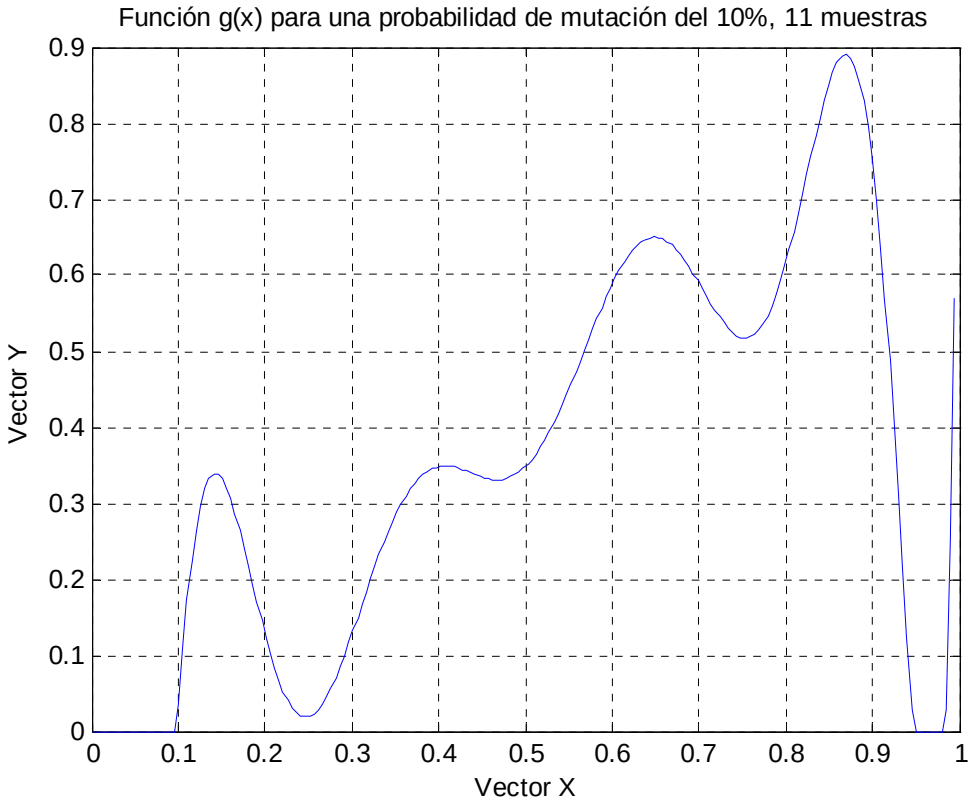

Fig. 85 Función *g(x)* para una probabilidad de mutación del 10%, 11 muestras

Probabilidad =40%

Vector Y

[0.000000, 0.013735, 0.082995, 0.313098, 0.401356, 0.424853, 0.471338, 0.569323, 0.847191, 0.973885, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000041439

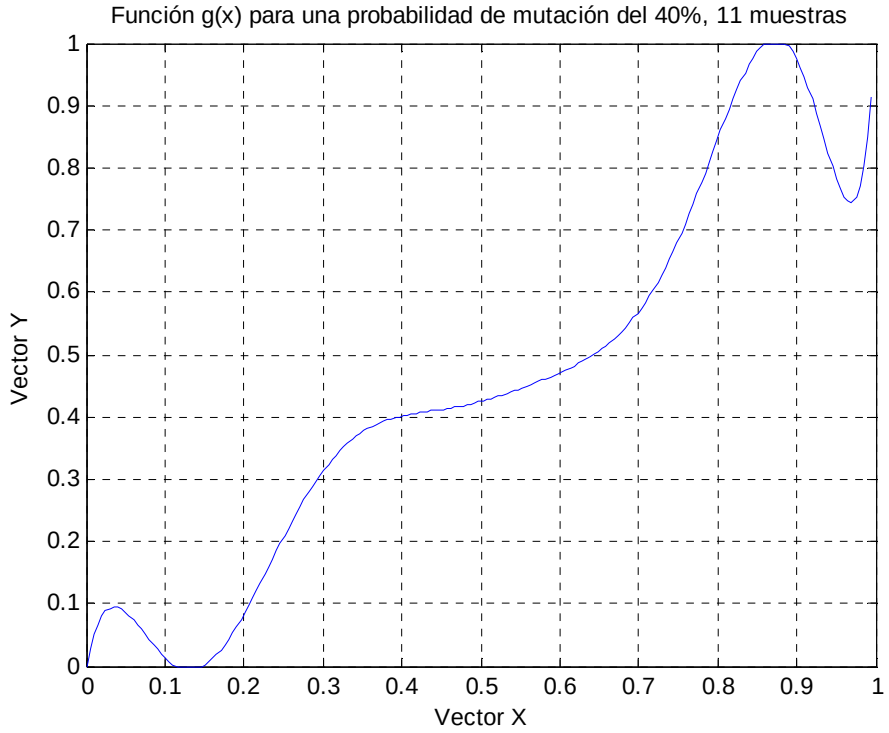

Fig. 86 Función *g(x)* para una probabilidad de mutación del 40%, 11 muestras

Probabilidad =65%

Vector Y

[0.000000], 0.185278, 0.185278, 0.185278, 0.185278, 0.399434, 0.399434, 0.682169, 0.709947, 1.000000, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000016402

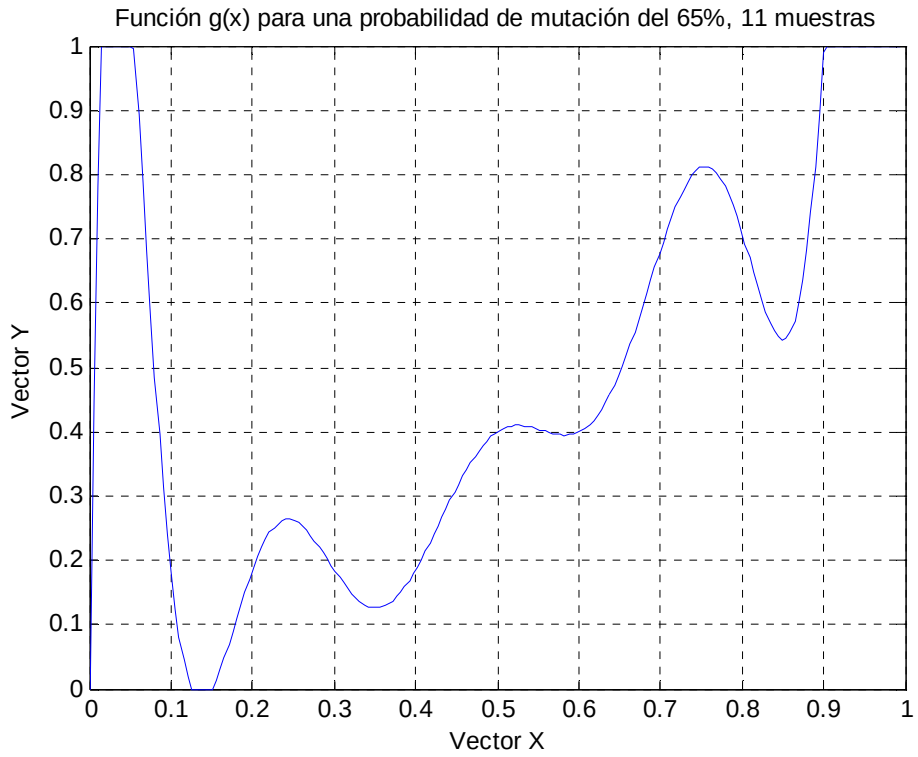

Fig. 87 Función *g(x)* para una probabilidad de mutación del 65%, 11 muestras

Probabilidad =100%

Vector Y

[0.000000, 0.018850, 0.172591, 0.241558, 0.463771, 0.463771, 0.659981, 0.659981, 0.761201, 0.838206, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.000000985

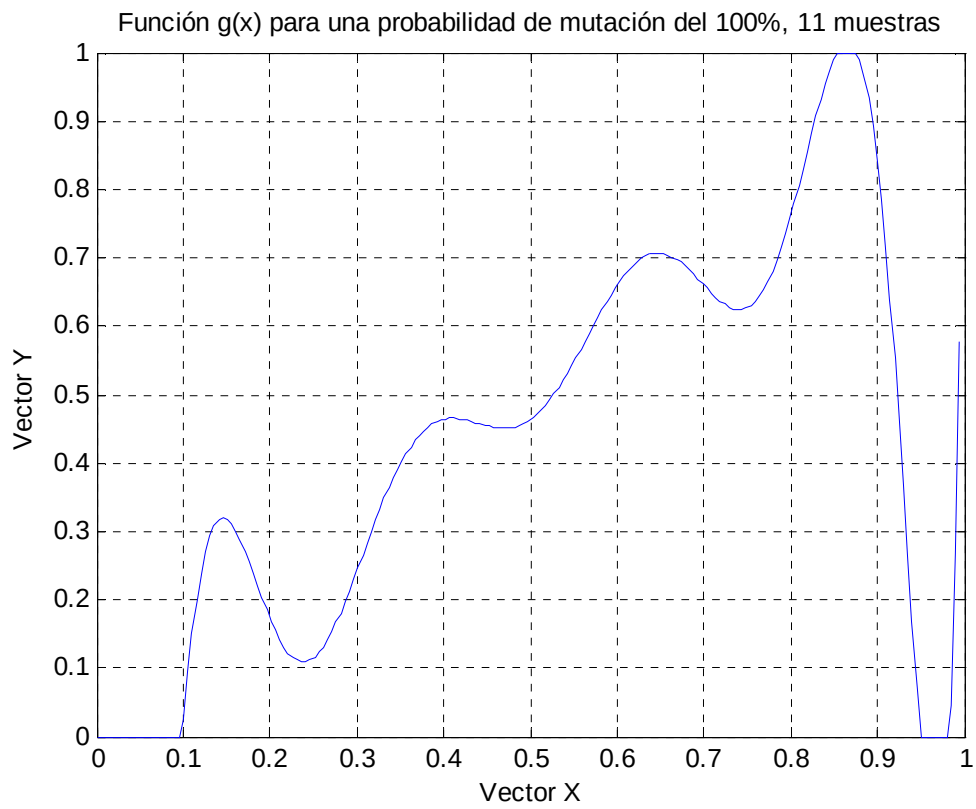

Fig. 88 Función *g(x)* para una probabilidad de mutación del 100%, 11 muestras

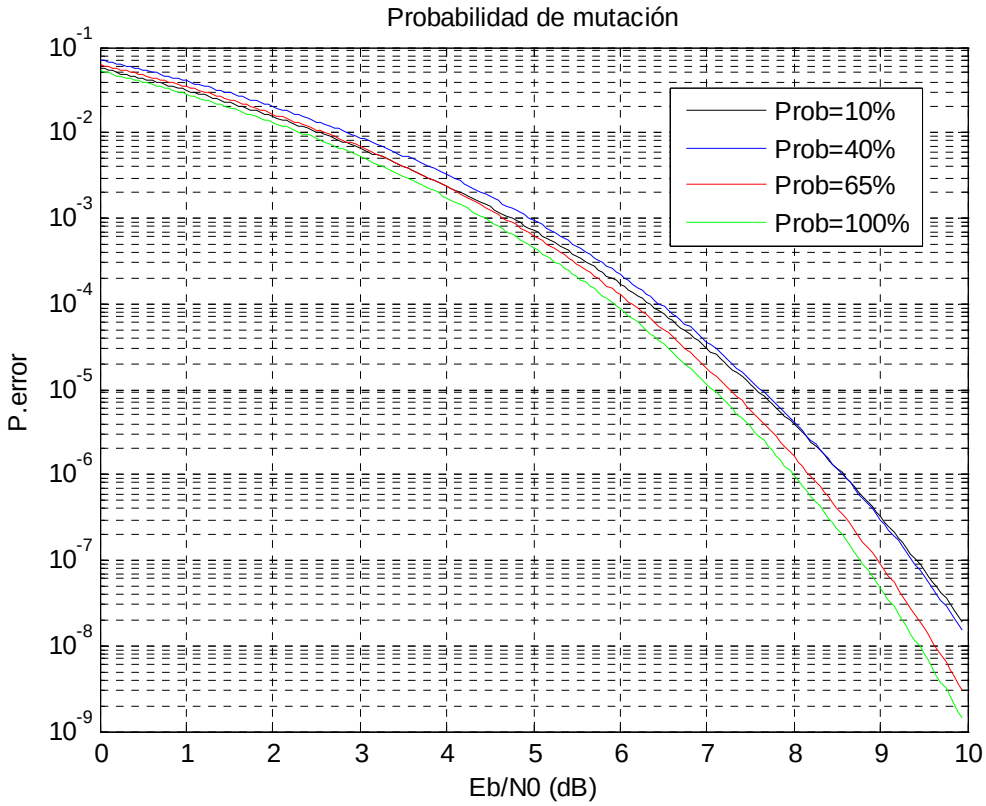

Fig. 89 Cota de la probabilidad de error respecto a la probabilidad de mutación, 11 muestras

13.2.7 **Variación del porcentaje de mutación**

Se ha fijado el factor de cuantificación a 4, la relación señal ruido a 8 dB, para 150 iteraciones del algoritmo genético, introduciendo 1100 valores *double* con una probabilidad de mutación del 100% para un valor de población de 200 modificando el porcentaje de mutación dentro de esta probabilidad.

Se ha realizado esta ultima medida con la probabilidad al 100% debido a que en el caso anterior al aumentar la probabilidad de mutación mejora mucho el error y partiendo de este dato conocido queremos seguir mejorándolo.

Porcentaje=10%

Vector Y

[0.000000, 0.001108, 0.143086, 0.352677, 0.352677, 0.391114, 0.461081, 0.594523, 0.970853, 0.972193, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000046354

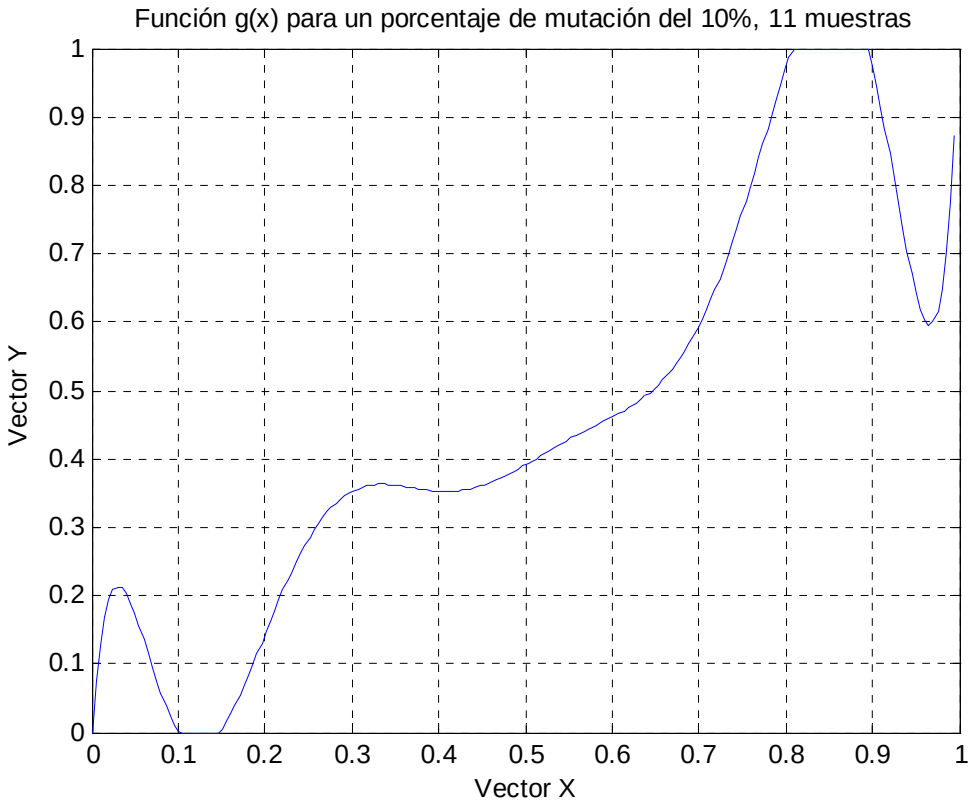

Fig. 90 Función *g(x)* para un porcentaje de mutación del 10%, 11 muestras

Porcentaje=40%

Vector Y

[0.000000, 0.105593, 0.140998, 0.140998, 0.341461, 0.342436, 0.652383, 0.666417, 0.666417, 0.774045, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000029098

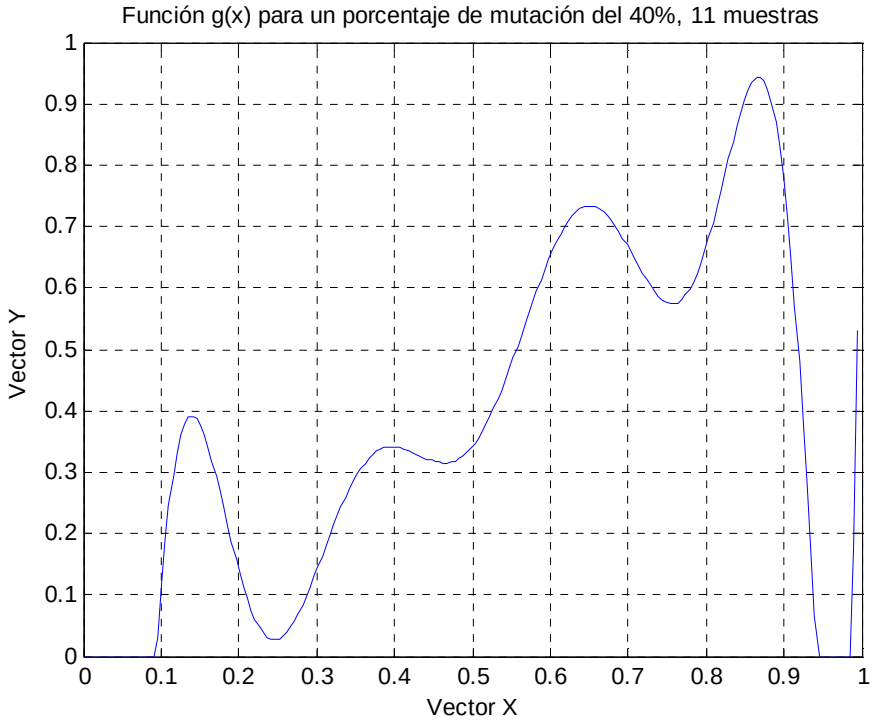

Fig. 91 Función *g(x)* para un porcentaje de mutación del 40%, 11 muestras

Porcentaje=65%

Vector Y

[0.000000, 0.091377, 0.157809, 0.159492, 0.416294, 0.416294, 0.668636, 0.668636, 0.684166, 0.684166, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.0000019341

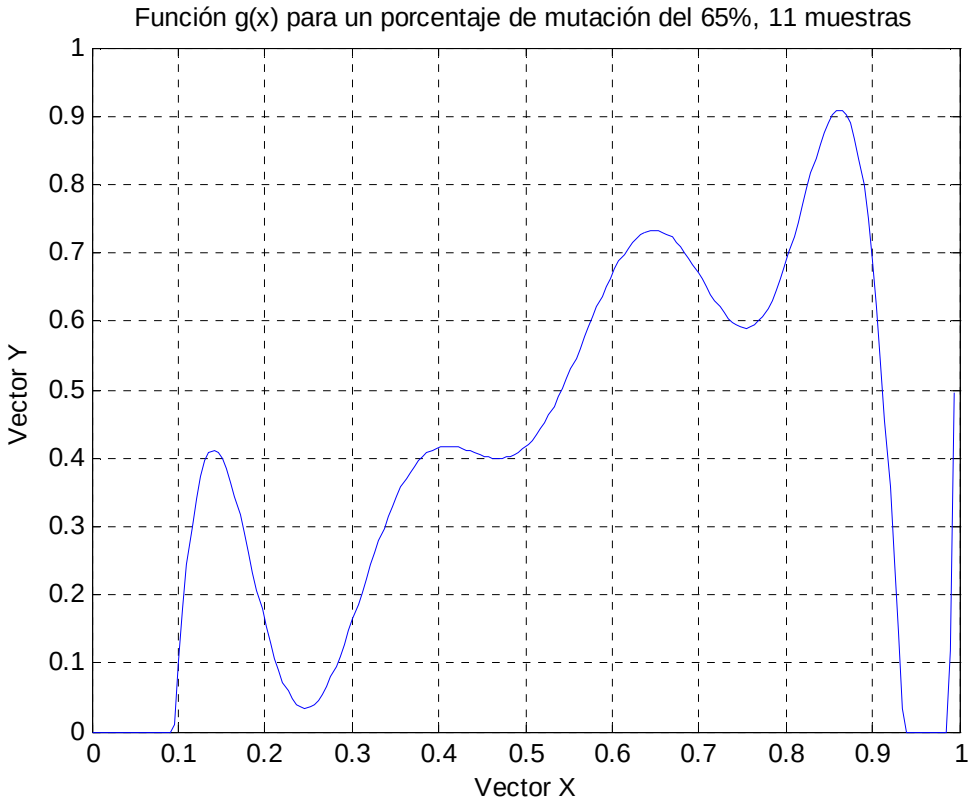

Fig. 92 Función *g(x)* para un porcentaje de mutación del 65%, 11 muestras

Porcentaje=100%

Vector Y

[0.000000, 0.023318, 0.217275, 0.217275, 0.536890, 0.536890, 0.737432, 0.737432, 0.850927, 0.962516, 1.000000]

Valor final de la función de coste (cota de la probabilidad de error): 0.000000746

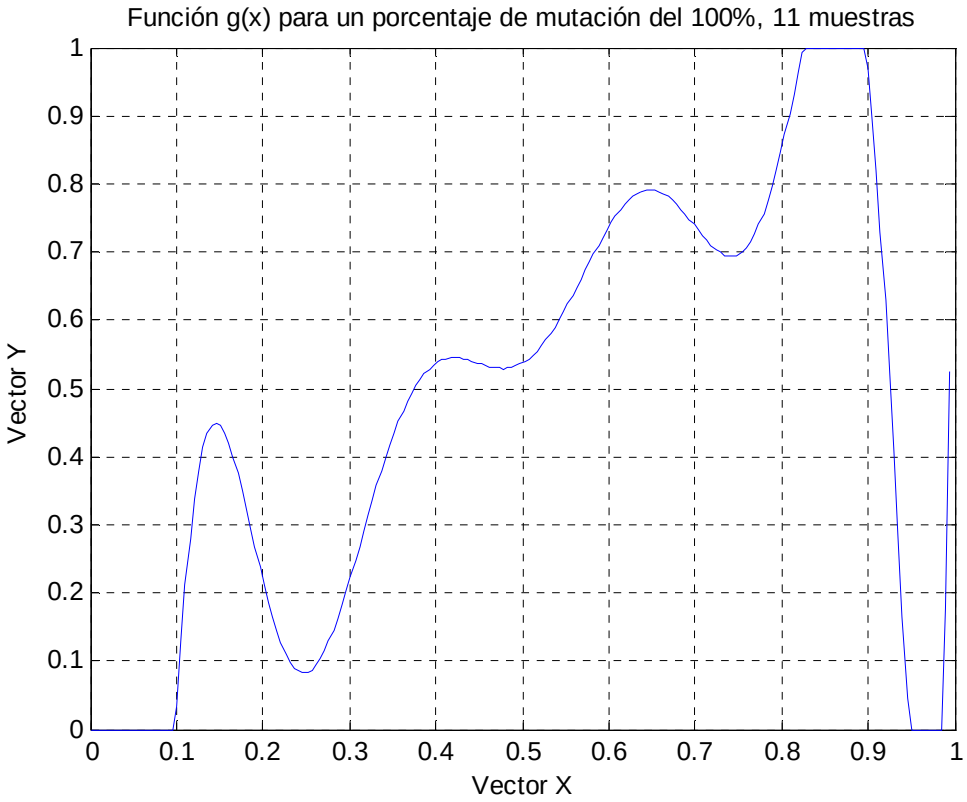

Fig. 93 Función *g(x)* para un porcentaje de mutación del 100%, 11 muestras

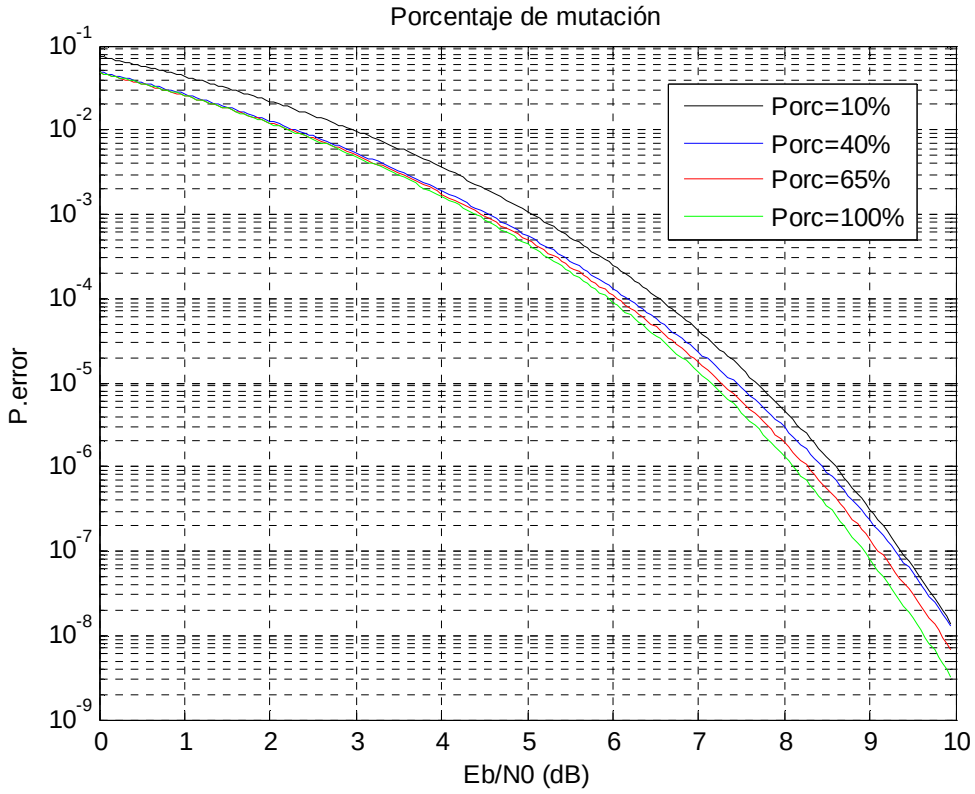

Fig. 94 Cota de la probabilidad de error respecto al porcentaje de mutación, 11 muestras