TRANSMISIÓN DE DATOS 2006/07			
Práctica 3: Codificación PCM		**Grupo**	
		Puesto	
Apellidos, nombre		**Fecha**	
Apellidos, nombre		**13/16 de noviembre de 2006**	

El objetivo de esta práctica es familiarizar al alumno con la codificación de forma de onda (con pérdidas) mediante técnicas de Modulación por Impulsos Codificados –MIC- (Pulse Code Modulation – PCM).

Para llevar a cabo la práctica, desarrolle cada ejercicio en un fichero Matlab dentro del directorio P3.

Se proporcionan dos códigos Matlab accesibles desde la página de la asignatura:

- `Grabando.m`: permite grabar un fichero de audio discreto (muestreado a 8 Khz) de 4 segundos de duración y almacenarlo en el fichero `sample_audio.mat`
- `Leyendo.m`: Es una función que permite leer un fichero de audio discreto (`sample_audio.mat`) , lo "sonifica" y dibuja sus gráficas características. Además devuelve un vector con la secuencia contenida en el fichero.

Si el equipo de desarrollo lo permite, el primer código permitirá realizar pruebas sobre un fragmento de audio grabado por cada usuario, en caso contrario, se puede usar el fragmento de audio (fichero `sample_audio.mat`) disponible también en la página de la asignatura. En el caso de hacer uso de un fichero propio para obtener resultados, se deberá incluir el mismo en el fichero de entrega de la práctica.

Todos estos ficheros están accesibles en el fichero TxDatosP3GXX.zip.

Justo antes de finalizar la práctica, comprima todo el directorio P3 en un fichero TxDatosP3GXxx.zip (siendo X el grupo –A ó B- y xx el número de pareja), conéctese al sistema de entrega de prácticas de la Intranet y entréguelo en el grupo que corresponda (A o B). Guárdese adicionalmente una **copia personal**, para posibles futuras reutilización del código en prácticas posteriores.

2.1 Codificación PCM uniforme

La codificación PCM uniforme es el método más sencillo de codificación por forma de onda y consiste en la aplicación directa de los conceptos de cuantificación uniforme.

El algoritmo básico consiste en dividir el rango dinámico en tantos intervalos uniformes como el número de niveles proporcionados por el número de bits disponibles para codificar cada muestra, y asignar un intervalo a cada muestra. La reconstrucción se hace al valor de reconstrucción de cada intervalo.

2.1.1 Ejercicio 1: Codificador Uniforme simétrico de 8 bits

A partir de `leyendo.m` escriba el código de un cuantificador uniforme simétrico, sin nivel de reconstrucción igual a 0, de n bits, que será guardado en `PCMUniforme.m`

La sintaxis de la función será la siguiente:

```
function [Scuan Srecon]=PCMUniforme(S,nbits,v)
%S:         secuencia de audio grabada
%nbits:     número de bits que usaremos para la codificación
%v:         valor de sobrecarga
%Scuan:     índice de los intervalos para la señal de entrada
%Srecon:    señal de entrada reconstruida
```

La función dibujará la representación temporal y el histograma de las siguientes señales:

- Señal original
- "Señal" cuantificada
- Señal reconstruida
- Señal de error

El programa deberá presentar por pantalla la potencia de error total calculada mediante:

```
Perror = sum(error.^2)
```

Recomendaciones:

La secuencia de entrada `S` *se obtendrá mediante la función* `Leyendo.m`

Inicie el desarrollo con un cuantificador de menos bits (2 ó 3) para verificar que el cálculo de los valores de decisión y de reconstrucción son calculados correctamente por el programa.

Dibuje e indique los valores de decisión y reconstrucción de un cuantificador uniforme simétrico, sin nivel de reconstrucción igual a 0, de 2 bits, y valor de sobrecarga V.

Dibuje e indique los valores de decisión y reconstrucción de un cuantificador uniforme simétrico, sin nivel de reconstrucción igual a 0, de 3 bits, y valor de sobrecarga V.

Elija un par de valores de muestras e inicie el desarrollo probando sobre una señal ficticia de una muestra (cuyo resultado haya calculado teóricamente).

Calcule los códigos, valores de reconstrucción y error de las siguientes muestras:

- V1(2)=0.7507 voltios con valor de sobrecarga 2 voltios (cuantificador 2 bits)

- V1(3)=0.7507 voltios con valor de sobrecarga 2 voltios (cuantificador 3 bits)

- V1(8)=0.7507 voltios con valor de sobrecarga 2 voltios (cuantificador 8 bits)

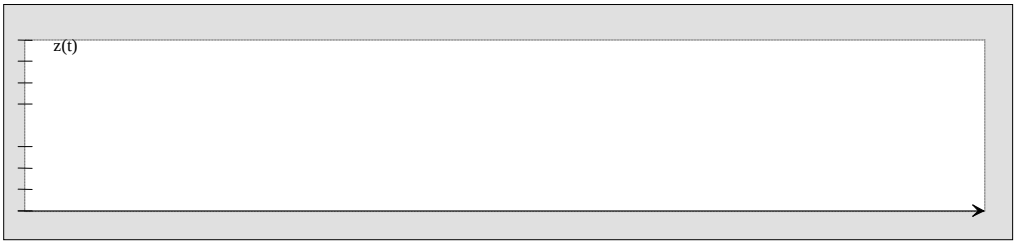
- V2(2)=0.2 voltios con valor de sobrecarga 1 voltio (cuantificador 2 bits)

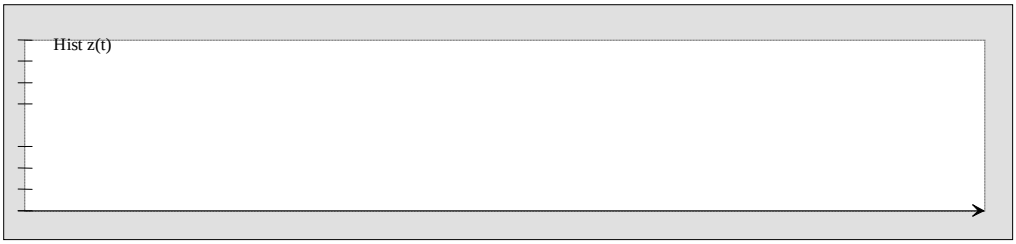
- V1(3)=0.2 voltios con valor de sobrecarga 1 voltio (cuantificador 3 bits)

- V1(8)=0.2 voltios con valor de sobrecarga 1 voltio (cuantificador 8 bits)

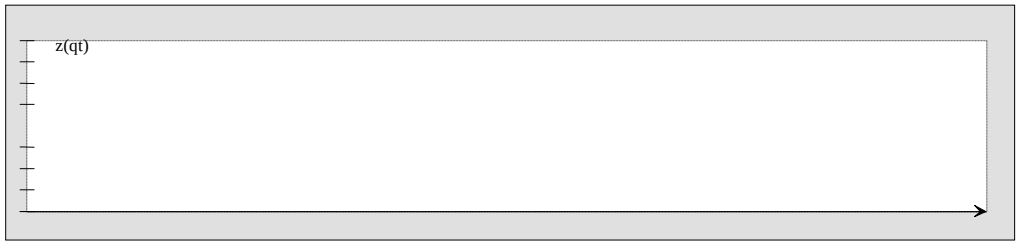
Una vez desarrollado en codificador, ejecute el programa sobre el fichero de audio `sample_audio.mat` y dibuje las gráficas correspondientes, indicando los valores de los ejes.

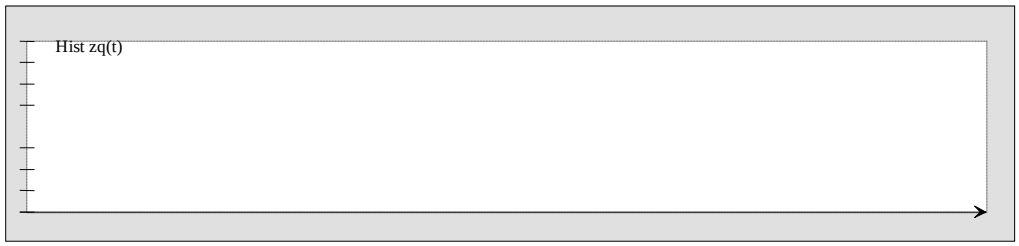
- Señal original



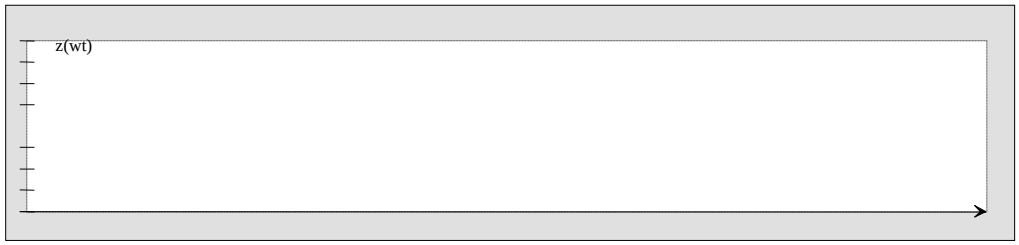


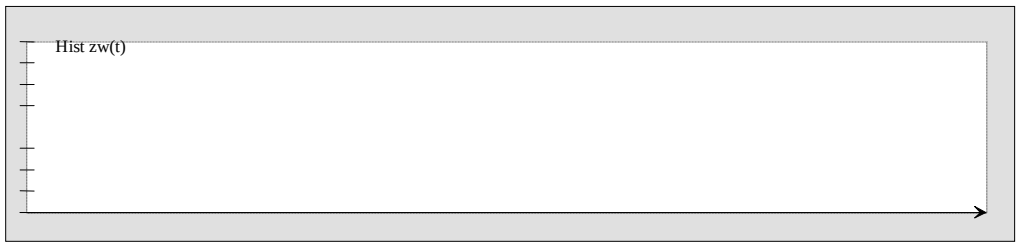
- “Señal” cuantificada



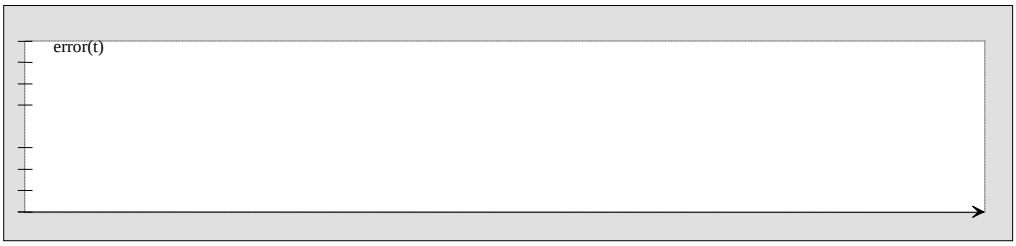


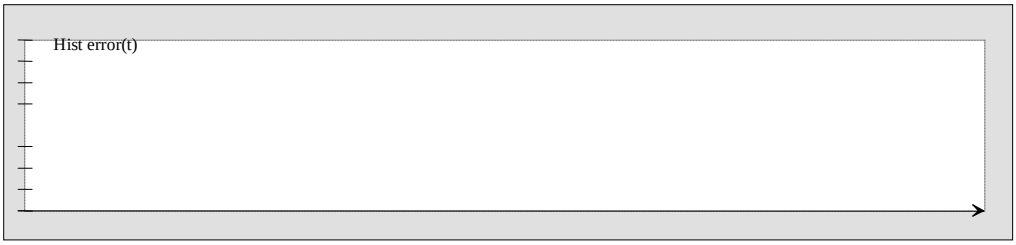
- Señal reconstruida





- Señal de error





Indique la potencia de error

Comente los resultados

2.2 Codificación PCM no uniforme

La codificación PCM no uniforme se basa en dividir el rango en intervalos de anchura variable, de forma que con el mismo número de bits (niveles) se pueda dar más precisión en las zonas donde hay una mayor probabilidad (densidad) de valores, lo que hace disminuir la potencia total de ruido al disminuir el valor de la contribución de las valores más probables. Generalmente se implementan mediante técnicas de "compansión" (compresión-expansión).

Las técnicas más usadas son los cuantificadores no uniformes robustos, que en lugar de minimizar la distorsión buscan lograr una relación señal-a-ruido de cuantificación constante. Estos cuantificadores tienen funciones de compresión logarítmicas. Adicionalmente se implementan como aproximaciones a las funciones (conocidas como *leyes*) teóricas. La norma utilizada en telefonía digital fija en Europa es la ley-A, aproximada mediante una función rectilínea que se especifica en la Recomendación G.711 de la UIT-UTI.

La Recomendación G.711, divide el rango (simétrico respecto al origen) en 13 segmentos rectilíneos. Si nos centramos en la zona positiva, se generan 7 segmentos (realmente 6+1+1). El segmento 7 ocupa la mitad del rango normalizado a 1 (Unidad de Tensión Normalizada –UTN-), esto es, de 1 a 0.5. El segmento 6 de 0.5 a 0.25, y así hasta llegar al segmento 1. El segmento 1 se divide en dos segmentos de igual tamaño (el 1 y el 0) por lo que suele hablar de 6+1 segmentos. Dentro de cada segmento no uniforme, se cuantifica el rango mediante 16 intervalos de cuantificación uniformes.

La codificación consiste en asignar un bit al signo (1 si positivo), 3 bits al segmento, y 4 bits al intervalo de cuantificación.

2.2.1 Ejercicio 2: Codificador G.711

A partir de `leyendo.m` y `PCMUniforme.m` escriba el código de un cuantificador G.711, que será guardado en `PCMG711.m`

La sintaxis de la función será la siguiente:

```
function [Scuan Srecon]= PCMG711(S)
%S:         secuencia de audio grabada
%Scuan:     secuencia de valores 0-255 [signo(1bit)  segmento(3bits)
%intervalo(4bits)]
%Srecon:    señal de entrada reconstruida(normalizada)
```

El programa dibujará la representación temporal y el histograma de las siguientes señales:

- Señal original
- Señal reconstruida
- Señal cuantificada
- Señal de error

El programa deberá presentar por pantalla la potencia de error total calculada mediante:

```
Perror = sum(error.^2)
```

El programa deberá permitir ver el código que se asignaría (esto es, al menos los valores decimales del signo, segmento e intervalo de cuantificación).

Recomendaciones:

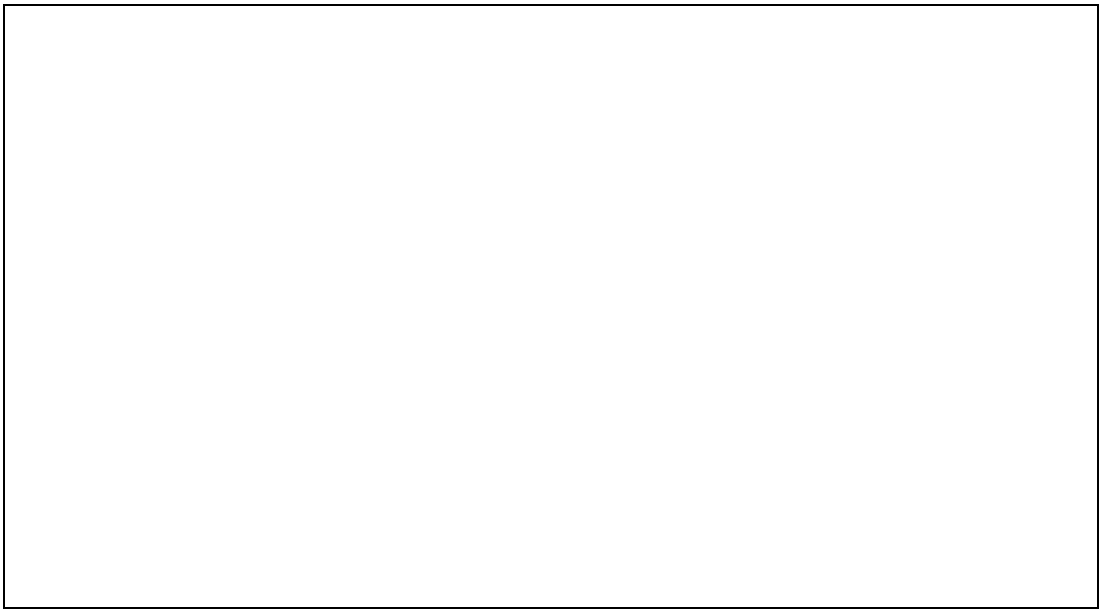
Elija un par de valores de muestras e inicie el desarrollo probando sobre una señal ficticia de una muestra (cuyo resultado haya calculado teóricamente).

La secuencia de entrada `S` *se obtendrá mediante la función* `Leyendo.m`

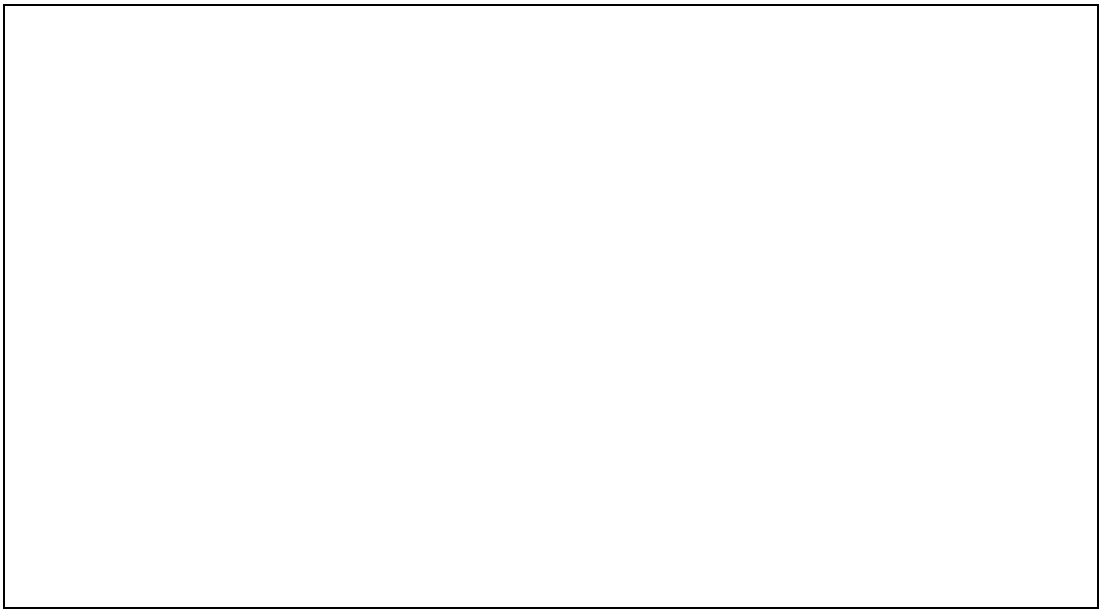
La función espera una señal de entrada que varíe entre -1 y 1.

Calcule los códigos, valores de reconstrucción y error de las siguientes muestras:

- V1(G)=0.7507 voltios con valor de sobrecarga 2 voltios

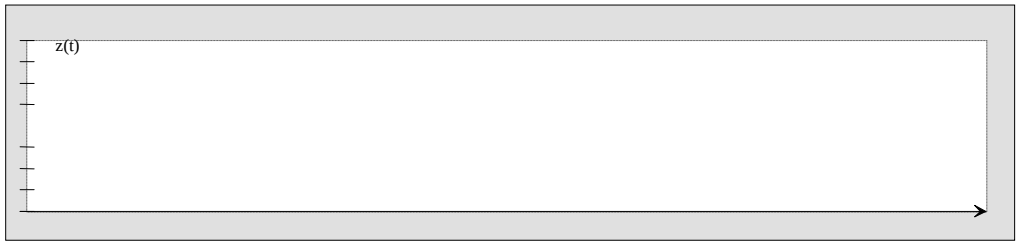


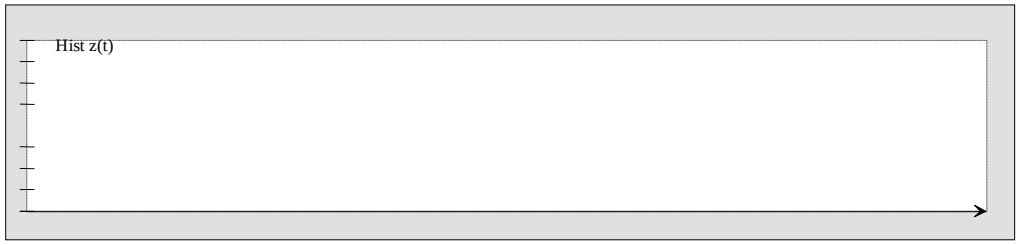
- V1(G)=0.2 voltios con valor de sobrecarga 1 voltio (cuantificador 8 bits)



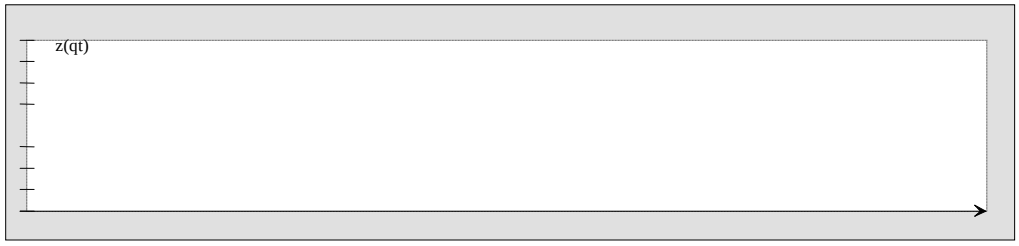
Una vez desarrollado en codificador, ejecute el programa sobre el fichero de audio `sample_audio.mat` y dibuje las gráficas correspondientes, indicando los valores de los ejes.

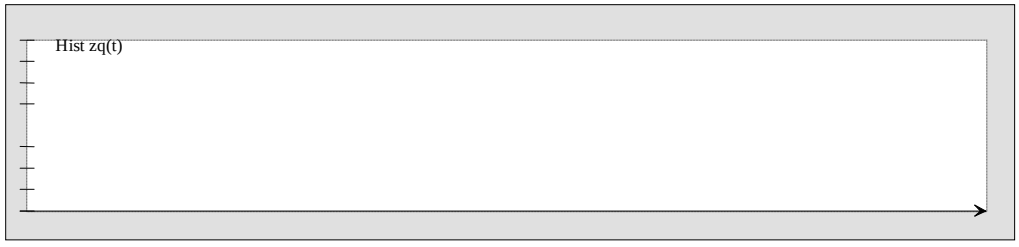
- Señal original



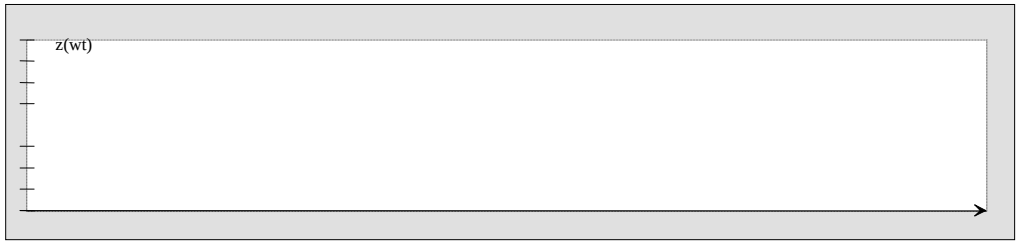


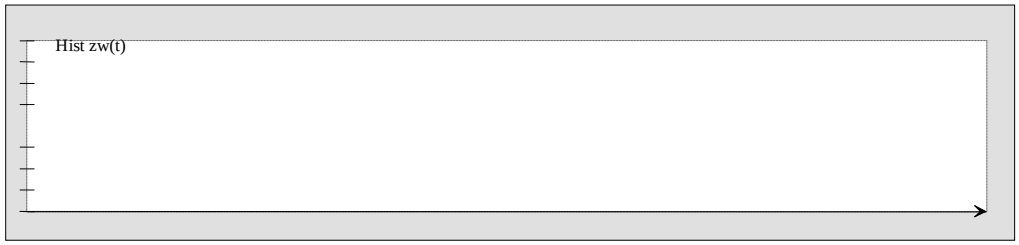
- "Señal" cuantificada



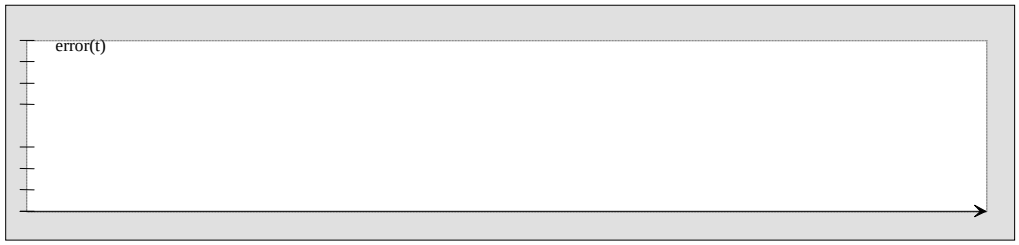


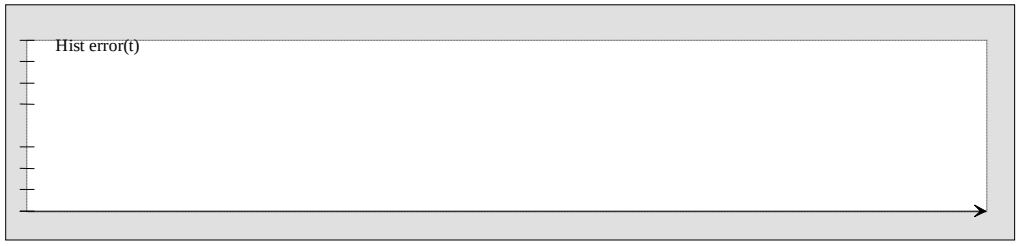
- Señal reconstruida





- Señal de error





Indique la potencia de error

Comente los resultados y compárelos con los del cuantificador uniforme