Tratamiento y Transmisión de Señales

Ingenieros Electrónicos

QUINTA PRÁCTICA

Cuantificación no lineal

En esta primera parte de la práctica el objetivo es visualizar la característica entrada/salida de un cuantificador no uniforme utilizando la técnica de compansión para ley μ.

Considerar los siguientes datos:

- Número de niveles del cuantificador `L=64`.
- Valor del parámetro μ del compansor `mu=10`.
- Número de amplitudes de la entrada `N=1000`.
- Amplitud máxima del cuantificador `A_max=15`.
- Amplitud máxima de la entrada `B_max=20`.

Generar el vector de amplitudes de entrada `Entrada` utilizando la función de Matlab `linspace` con `N` puntos con amplitud máxima negativa `-B_max` y positiva `B_max`.

Usando las funciones de Matlab `quantiz` y `compand` determinar las amplitudes cuantificadas de salida `Salida` definiendo para ello las variables y vectores necesarios.

Para ver la característica entrada/salida basta con representar el vector `Salida` como función del vector `Entrada` utilizando la función de Matlab `plot`. Añadir a la figura una rejilla de referencia usando la función de Matlab `grid` y escalar, usando la función de Matlab `axis`, el eje de entradas (horizontal) usando como valor máximo negativo `-B_max` y positivo `B_max` y el eje de salidas (vertical) usando como valor máximo negativo `-A_max` y positivo `A_max`.

NOTA: probar a cambiar los parámetros `L`, `mu`, `A_max` y `B_max` de los definidos arriba observando el resultado.

Transmisión Digital en Banda Base

En esta segunda parte de la práctica el objetivo es comprobar el funcionamiento de un sistema de transmisión en banda base con receptor empleando la técnica de filtro adaptado. En particular, se va presentar por pantalla la señal transmitida, la señal recibida ruidosa y la señal recuperada a la salida del receptor. Finalmente se va a comparar la curva de probabilidad de error teórica con la obtenida experimentalmente.

Considerar los siguientes datos:

- Tasa binaria de entrada al transmisor `Rb=64000`.
- Número de muestras de la señal transmitida por bit `K=10`.
- Energía por bit `Eb=1`.
- Número de bits transmitidos `M=100000`.
- Número de niveles de ruido `N=50`.
- Nivel de ruido mínimo `N0_min=0.1`.
- Nivel de ruido máximo `N0_max=10`.

Generar un vector de densidades espectrales de ruido `N0` espaciados logarítmicamente entre `N0_min` y `N0_max` con `N` puntos usando la función de Matlab `logspace`.

Definir las siguientes variables de Matlab según corresponda:

- Duración de cada bit `Tb`.
- Duración de cada muestra de la señal transmitida `Ts`.
- Frecuencia de muestreo `fs`.
- Amplitud de la señal `A`.
- La señal binaria con `M` bits teniendo en cuenta que los ceros y unos son equiprobables. Utilizar la función de Matlab `rand`. Almacenar el resultado en el vector `bitsTx`.
- Usando el código de línea NRZ polar determinar la señal a transmitir a partir del vector `bitsTx`. Utilizar las funciones de Matlab `reshape` y `repmat`. Almacenar el resultado en el vector `senalTx`. Comprobar que `senalTx` es un vector fila con `M*K` elementos.
- Determinar el vector de varianzas de ruido `varianzas` a partir del vector de densidades espectrales `N0`.

- Generar un eje temporal `tiempo` con `M*K` elementos.

Hacer lo siguiente para cada elemento del vector de varianzas de ruido `varianzas` (utilizar un bucle `for` con `N` iteraciones):

- Generar un vector de ruido blanco Gaussiano con media cero y con varianza dada por una de las posibles varianzas (una para cada repetición del bucle `for`) y almacenarlo en el vector `ruido`. Este vector debe tener el mismo tamaño que `senalTx`.
- Determinar el vector de señal recibida `senalRuido` sumando al vector `senalTx` el vector `ruido`.
- Determinar la señal a la entrada del decisor (tras el filtro adaptado y el muestreador) y almacenarla en el vector `salida` utilizando las funciones de Matlab `reshape` y `sum`. El vector `salida` debe ser un vector fila con `M` elementos.
- Determinar la señal binaria tras el decisor y almacenar el resultado en el vector `bitsRx`.
- A partir de la señal binaria recibida `bitsRx` generar la señal NRZ polar recibida `senalRx` de la misma forma que como se hizo con `senalTx` a partir de `bitsTx`.
- Determinar la probabilidad de error contando los bits diferentes entre `bitsTx` y `bitsRx` dividiendo el resultado entre el número de bits `M`. Almacenar el resultado en el vector `Pe` (un elemento para cada iteración del bucle `for`).
- Dibujar en la misma figura los primeras 500 muestras de la señal transmitida `senalTx`, de la señal ruidosa `senalRuido` y de la señal recibida `senalRx` en tres gráficas independientes. Utilizar las funciones de Matlab `figure`, `subplot`, `plot` y `drawnow` (esta última función se utiliza para que se actualice la figura cada iteración del bucle `for`).

Después de terminar el bucle `for` el vector `Pe` debe tener las `N` probabilidades de error correspondientes a los `N` niveles de ruido. Determinar ahora el vector `EbN0` como el cociente entre la energía por bit `Eb` y la densidad espectral de ruido `N0`. Determinar la probabilidad de error teórica `PeTeorica` a partir del vector `EbN0` usando las funciones de Matlab `sqrt` y `erfc`. Pasar el vector `EbN0` a dB definiendo el vector `EbN0log` usando la función de Matlab `log10`. Finalmente, utilizando las funciones de Matlab `figure`, `semilogy` y `grid` dibujar en una segunda figura la probabilidad de error teórica `PeTeorica` y la experimental `Pe` como función del vector `EbN0log` comprobando que la curva experimental se aproxima con bastante fiabilidad a la teórica.