Práctica 2. Pulsos en bandabase

2.1. Objetivos

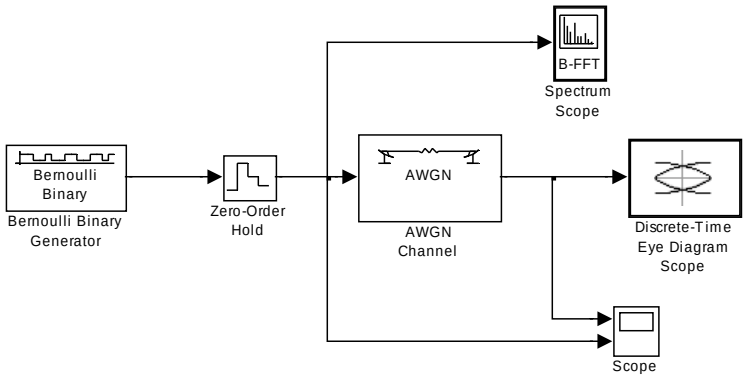
Aprender a utilizar el Simulink de MATLAB para analizar sistemas de comunicación en bandabase.

2.2. Realización práctica

2.2.1. Señal unipolar NRZ

Haciendo uso del Simulink es posible determinar las características principales de una señal unipolar NRZ. Diseñar un generador aleatorio de señales de este tipo mediante el bloque *Bernoulli Binary Generator*, disponible en el *Communications Blockset* (*Comm Sources/Random Data Sources*). Añadir un muestreador (*Simulink/Discrete/Zero-Order Hold*) para convertir los unos y ceros suministrados por el generador binario en ondas cuadradas de una duración determinada. Para ello, indicar, entre los parámetros del generador, el tiempo entre sucesivos símbolos generados (parámetro *sample time*). Dejar, en este caso, el valor por defecto igual a 1 segundo. A continuación establecer el tiempo entre muestras (*sample time*) para el *zero-order hold*, por ejemplo, a un valor de 1/16 s.

Vamos a añadir un canal AWGN (*Communications Blockset/Channels/AWGN Channel*) para comprobar el efecto del ruido sobre la señal transmitida. Estableceremos su modo (parámetro *mode*) a *signal to noise ratio (SNR)*, con un valor de *SNR* de 20 dB, para una potencia de la señal de entrada (*input signal power (watts*)) de 0,5 W. Este último valor no se ha establecido de manera arbitraria, sino que se debe a que la señal de entrada es una onda binaria con niveles de 1 voltio o cero voltios, con duraciones del pulso de 1 s. Teniendo en cuenta que existe la misma probabilidad de emitir unos que ceros, la potencia media de la señal generada, la de entrada al canal AWGN, será de 0,5 W.



Tras esto, por último, vamos a introducir en el sistema los instrumentos de visualización: un osciloscopio (*Simulink/Sinks/Scope*), un analizador de espectros

(*Signal Processing Blockset/Signal Processing Sinks/Spectrum Scope*) y un visualizador de diagramas de ojo (*Communications Blockset/Comm Sinks/Discrete-Time Eye Diagram Scope*). En el osciloscopio, establecer dos ejes de visualización (*Parameters/number of axes*) y desactivar *Limit data points to last* en *Data History*, para que almacene todas las muestras que reciba a la entrada. En el *Spectrum Scope*, establecer *Buffer size* a 1024, *Buffer overlap* a 512 y *Number of spectral averages* a 16. Por último, en el visualizador de diagramas de ojo, establecer *samples per symbol* a 16, *offset (samples)* a 8, es decir, 16/2, y *traces displayed* a 100. Lo único que estamos haciendo, en este último caso, es indicarle el número de muestras que tiene cada símbolo, esto es, 16, tal como establecimos en el circuito de muestreo, y que añada un *offset* para que el diagrama de ojo salga centrado, como veremos.

Ya para finalizar, establecer el tiempo de simulación a 1.000 segundos.

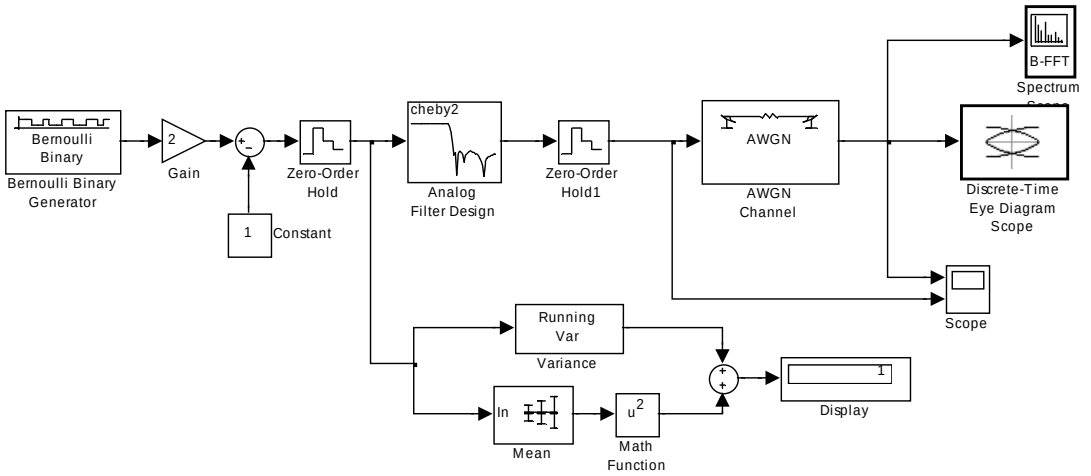
1. Visualizar el diagrama de ojo y comprobar que los valores de amplitud de la señal oscilan entre aproximadamente 1 y 0, más una componente de ruido. Obsérvese también la característica cuadrada de dichos diagramas de ojo.
2. Observar en el osciloscopio las señales unipolares con y sin ruido.
3. Observar el espectro de la señal unipolar NRZ. Por defecto, este viene representado en decibelios. En las propiedades del analizador de espectro, en la pestaña *Axis Properties*, establecer *Frequency range* a *[-Fs/2 ... Fs/2]*, *Amplitude scaling* a *Magnitude-squared*, *Minimum Y-Limit* a un valor de 10^{-5} y *Maximum Y-Limit* a 10. Ejecutar nuevamente la simulación y comprobar el nuevo aspecto del espectro de la señal unipolar NRZ. Comprobar que existe un pico del espectro en 0 Hz, lo que se corresponde perfectamente con lo predicho teóricamente. Asimismo, el espectro obtenido tiene la forma de una función sinc al cuadrado con nulos en $\pm n$ Hz, donde $n = 1, 2, \ldots$, etc.

2.2.2. Señal polar NRZ

Vamos a continuación a analizar las señales polares NRZ. Para ello, vamos a modificar ligeramente el sistema implementado anteriormente para eliminar la componente de continua de la señal unipolar y hacer que la nueva señal generada tome los valores ± 1 voltio, esto es:

$$s_{polar}(t) = 2s_{unipolar}(t) - 1 \qquad (1)$$

Por tanto, basta con multiplicar por dos la señal de salida del generador (unipolar) y restarle uno, para obtener una versión polar de esa misma señal. El nuevo sistema para el análisis de señales polares se muestra en la figura siguiente, donde también se ha añadido un filtro analógico (*Signal Processing Blockset/Filtering/Filter Design/Analog Filter Design*) y un sistema para el cálculo de la potencia media de la señal.



La potencia media de una señal aleatoria viene dada por:

$$P = P_{AC} + P_{DC} = \langle x^2(t) \rangle = \langle x^2(t) \rangle - \langle x(t) \rangle^2 + \langle x(t) \rangle^2 = \\ = \langle (x(t) - \langle x(t) \rangle)^2 \rangle + \langle x(t) \rangle^2 = \sigma_x^2 + m_x^2 \quad (2)$$

donde $\langle \cdot \rangle$ indica promedio temporal dado por:

$$\langle f(t) \rangle = \lim_{T \to \infty} \frac{1}{T} \int_{-T/2}^{T/2} f(t) dt \quad (3)$$

y donde σ_x^2 y m_x^2 son la varianza y la media de x al cuadrado, respectivamente. De aquí que se hayan utilizado los bloques mostrados en la figura para determinar la potencia media de la señal. Los bloques *Variance* y *Mean* están disponibles en *Signal Processing Blockset/Statistics.*

En el filtro analógico de la figura establecer *Design method* a *Chebyshev II*, mantener los valores por defecto para *Filter type*, *Filter order* y *Stopband attenuation in dB*, y establecer la frecuencia de corte del filtro (*Stopband edge frequency (rads/sec)*) a $2\pi \cdot 0{,}75$ rad/s, es decir, 0,75 Hz. Con esto último, lo que se busca es distorsionar la señal polar para provocar interferencia intersímbolo (ISI) en muestras consecutivas.

1. Establecer el tiempo de simulación a 2.000 segundos. Observar el diagrama de ojo distorsionado por el ruido y la ISI producida por el filtrado del canal. Para comprobar el efecto del filtrado del canal sobre el espectro de la señal recibida, establecer la frecuencia de corte del filtro a $2\pi \cdot 20$ rad/s en una simulación y luego a $2\pi \cdot 0{,}75$ rad/s en la siguiente. Observar cómo los lóbulos secundarios y las frecuencias más altas del lóbulo principal son fuertemente atenuadas por el filtro. Para observar mejor los cambios establecer *Amplitude scaling* en *Axis Properties* del analizador de espectros a *dB*. Comprobar también que el pico en 0 Hz ha

desaparecido para el caso de la señal polar, lo que era de esperar pues este tipo de señal presenta un valor nulo de DC (m_x = 0). De hecho, para el cálculo de su potencia promedio se tendría que:

$$P_{polar} = \sigma_x^2 + m_x^2 = \sigma_x^2$$

2. Con la frecuencia de corte del filtro establecida a $2\pi \cdot 0{,}75$ rad/s, eliminar la componente de ruido estableciendo *SNR* a 100 dB en las propiedades del bloque *AWGN Channel*. Asimismo, establecer la potencia de la señal de entrada a 1 W (la señal polar tiene el doble de potencia promedio que la señal unipolar como se puede comprobar durante la simulación en el *display* conectado a la salida de los bloques incluidos en el sistema para tal fin). Comprobar en el diagrama de ojo el efecto debido únicamente a la ISI inducida por el canal (filtro). Para compararlo con el caso ideal (sin filtro), establecer nuevamente la frecuencia de corte a $2\pi \cdot 20$ rad/s y realizar una nueva simulación. Obsérvese que la ISI, al aplicar el filtrado a 0,75 Hz, reduce muchísimo la apertura del ojo respecto al caso ideal.
3. Observar la forma de onda temporal, mediante el osciloscopio, de las señales tanto filtradas como no filtradas. Para ello, conectar a una de las entradas del osciloscopio la señal de entrada al filtro, y a la otra entrada del osciloscopio la señal de salida del filtro.
4. ¿Qué ocurriría si hacemos la frecuencia de corte igual a 0,6 Hz? ¿Y si la hacemos igual a 0,5 Hz?

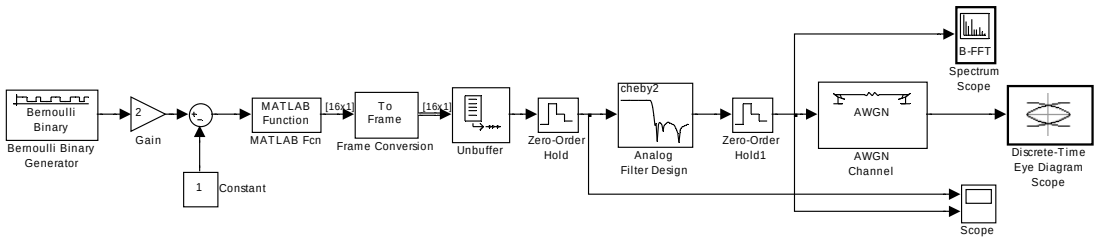
2.2.3. Transmisión mediante pulsos con forma de onda sinc

Vamos a modificar la forma del pulso para que siga la forma de una onda sinc. Para ello, en el sistema anterior añadiremos una serie de bloques que se encargarán de llevar a cabo la generación de este tipo de pulsos. Un pulso con forma de onda sinc, viene dado por:

$$f(t) = \frac{\text{sen}[\pi(t - t_0)/T]}{\pi(t - t_0)/T} = \text{sinc}[\pi(t - t_0)/T] \qquad (4)$$

donde T es el tiempo transcurrido entre nulos sucesivos de la forma de onda y t_0 es el instante de tiempo donde se produce el máximo de la señal. Haremos uso de una función de MATLAB para generar este tipo de forma de onda, y añadiremos un bloque *Simulink/User-Defined Functions/MATLAB Fcn* que llame a esta función para llevar a cabo dicha tarea (ver figura siguiente sobre el nuevo esquema del sistema). Dicho bloque llamará a una función, que denominaremos *fsinc*, con tres argumentos de entrada (la entrada al propio bloque, que se especifica con la letra *u*, el tiempo de duración del pulso o intervalo entre nulos de la forma de onda, que estableceremos en 1 segundo, y el número de muestras por tiempo de símbolo que estableceremos a 16). Para ello, en las propiedades del bloque hay que establecer el parámetro *MATLAB function* a

fsinc(*u*,1,16). Asimismo, estableceremos las dimensiones del vector de salida (*Output dimensions*) a [16,1], es decir, la salida del bloque será un vector columna de 16 filas. Para convertir esta señal en una secuencia temporal de valores hay que introducir un bloque *Frame Conversion* y un bloque *Unbuffer*. De esta manera, los 16 valores de salida del bloque *MATLAB Fcn* se convertirán en 16 muestras temporales correctamente ordenadas en el tiempo según la tasa de muestreo establecida por el bloque *Zero-Order Hold* incluido a continuación.



Lo primero que haremos será crear una función que genere una forma de onda sinc en el rango de valores indicado a su entrada. Dicha función en MATLAB puede definirse simplemente como:

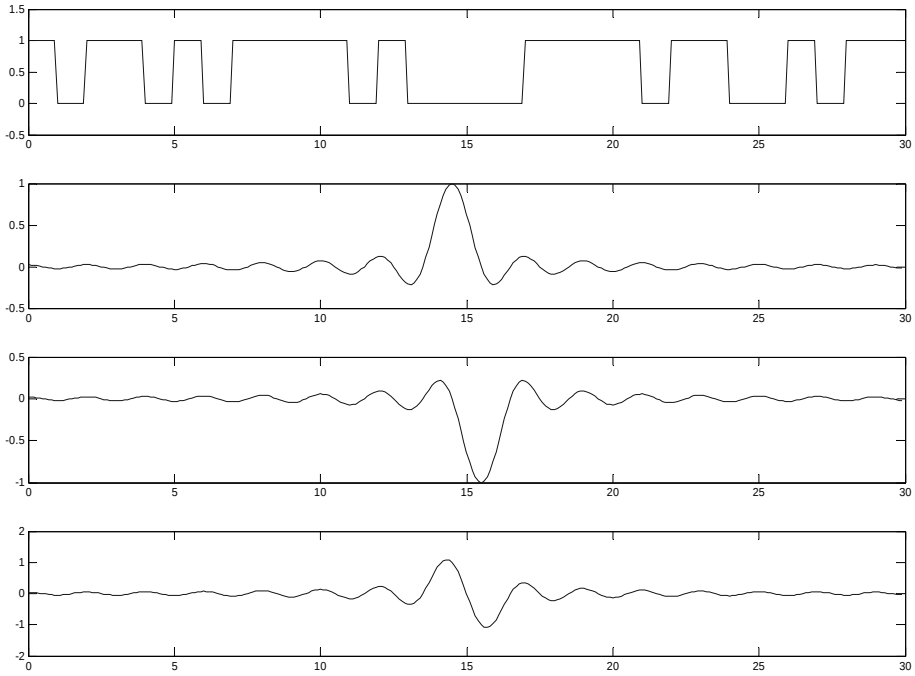
```
function y=sinc(x)
%
% Calcula sinc(x) para el rango de valores dado en el vector x
N=length(x);
y=zeros(1,N);
for i=1:N
    if x(i)==0
        y(i)=1;
    else
        y(i)=sin(x(i))/x(i);
    end
end
```

Los comentarios introducidos debajo de la declaración de la función aparecerán siempre que ejecutemos en la línea de comandos del MATLAB:

```
help sinc
```

Hay que indicar que esta función existe como función incorporada en las librerías del MATLAB, pero hemos definido nuestra propia función para ayudar a la comprensión del sistema a implementar. La función sinc(x) se caracteriza por presentar un máximo igual a uno en $x = 0$, y ser nula en los valores $x = \pm n\pi$, donde n es un número natural distinto de cero. Al contrario que con los pulsos rectangulares, donde la amplitud del pulso sólo depende del valor del bit actual (0 ó 1) durante los T segundos de duración del pulso, cuando se hace uso de formas de onda sinc se tiene que, idealmente, éstas tienen que tener una duración infinita tanto a instantes de tiempo negativos como positivos. Como no se pueden crear formas de onda a instantes de tiempo negativos, pues en ese caso el sistema sería no causal, podemos aprovechar la característica de atenuación de la amplitud de la señal sinc a lo largo del tiempo. Se tiene que, aproximadamente, a partir del decimoquinto nulo

después del máximo la amplitud ha decaído considerablemente (es inferior al 2% del valor de pico de la señal). Es importante comprobar que, aunque en la práctica despreciemos estas amplitudes, si fuéramos rigurosos habría que considerar lapsos de tiempo aún mayores. Dado que la función sinc es simétrica respecto del pico, esto quiere decir que habrá que considerar tanto los valores hasta el decimoquinto nulo a la izquierda como a la derecha del pico de la forma de onda. En la figura siguiente se muestra una sucesión de datos binarios, junto con las formas de onda debidas a los dos primeros datos, más la suma de estas dos señales (gráfica inferior). Evidentemente, para obtener la forma de onda total habría que sumar las formas de ondas sinc correspondientes a cada uno de los datos binarios. Una de las características evidentes es que el pico de la forma de onda del pulso sinc está retardada casi 15 segundos (14,5 segundos exactamente) respecto de su dato binario correspondiente. Esto es necesario para hacer que el sistema de generación de pulsos sea causal, como hemos indicado anteriormente. La representación de señales mostrada también nos da cierta idea de las características que debe tener la función en MATLAB que diseñemos para generar la señal pulsada con formas sinc. Para empezar, habrá que almacenar los valores de los 29 datos binarios previos pues su duración en el tiempo es de $30T$, donde T es el tiempo de duración del dato (en este caso, un segundo). Esto es, en el ejemplo de la figura, el trigésimo dato es un uno y habría que generar una función sinc que partiera desde ese mismo instante, pero las formas de onda sinc de hasta la vigésimo novena muestra anterior serían necesarias para sumar sus contribuciones en el instante de tiempo actual. Así, la influencia de la primera muestra se extinguiría tras este instante de tiempo, y así sucesivamente con el resto de las muestras siguientes.



A continuación se muestra un posible código en MATLAB para generar las formas de onda sinc a cada entrada de un nuevo dato binario:

```
function salida=fsinc(nuevo_dato,T,N)
persistent datos;
M=30;
if isempty(datos)
    datos=zeros(1,M);
end
datos=[datos(2:M) nuevo_dato];
salida=zeros(1,N);
t=0:T/N:T*(1-1/N);
for i=1:M
    x=pi*((t-T*(i-M/2-0.5))/T);
    salida=salida+datos(i)*sinc(x);
end
salida=salida.';
```

Lo primero que se hace es declarar una variable de tipo persistente (*persistent*) de tal forma que no borre su información cuando se salga de la función y se vuelva a entrar. Esta variable *datos* almacenará los datos binarios de entrada al bloque cada T segundos de tiempo, por lo que inicialmente (la primera vez que se llama a la función) se establecerá toda a cero. A continuación el nuevo dato introducido al bloque, *nuevo_dato*, se incorpora a la memoria de *datos*, desplazando a la izquierda todos los datos previos, eliminándose por tanto el primer dato introducido 30 instantes de tiempo atrás. A continuación se calcula la forma de onda con N muestras durante los T segundos actuales mediante la suma de las contribuciones debidas a cada uno de los datos, tanto el actual como los previos:

$$f(t) = \sum_{i=1}^{M} x_i \cdot \mathrm{sinc}\left\{\frac{\pi[t - T(i - M/2 + 0.5)]}{T}\right\}, \quad 0 \le t < T$$

donde $x_i = \pm 1$ según el dato binario sea un uno o un cero. El índice i va desde el primer dato introducido ($i = 1$) hasta el dato actual ($i = M$). Obsérvese que con esta función sólo se calcula la forma de onda durante el intervalo de tiempo actual, no en los sucesivos intervalos de tiempo, motivo por el cual es necesario almacenar los datos previos durante un gran lapso de tiempo. Finalmente la forma de onda generada se almacena en la variable *salida*, la cual es devuelta por la función y constituye el vector de salida del bloque *MATLAB Fcn*.

1. Con el generador de pulsos sinc que se ha introducido, comprobar las formas de onda de la señal antes y después del filtro. ¿A qué se debe el gran parecido entre ellas?
2. Comprobar el espectro de frecuencias de la señal sinc antes y después del filtro pasabajas con frecuencia de corte 0,75 Hz. ¿Qué ancho de banda tiene aproximadamente la señal original sinc?
3. Comprobar el efecto de la ISI cuando se utilizan pulsos de este tipo. ¿Qué características destacarías en contraposición a lo observado con pulsos rectangulares?