UNIVERSIDAD DE GRANADA

Titulación: **COMPLEMENTOS DE FORMACIÓN PARA INGENIERÍA ELECTRÓNICA**

Asignatura: **TRANSMISIÓN DE DATOS**

GUIONES DE PRÁCTICAS

Ángel de la Torre Vega

Granada, Marzo de 2006

II

Práctica 1

SERIES DE FOURIER

El objetivo de esta práctica es estudiar funciones periódicas mediante series trigonométricas y exponenciales de Fourier. Esto se hará con la ayuda del software MatLab.

1.1. Análisis del tren periódico de pulsos rectangulares

1. Preparar una rutina de MatLab que proporcione un tren periódico de pulsos rectangulares. La rutina tendrá el formato:

   ```
   x = pulso(t,T_0,tau,A)
   ```

2. Obtener los coeficientes a_0, a_n y b_n de la serie trigonométrica de Fourier para n menor o igual a 20, para unos valores fijos de T_0, τ y A, a partir de las expresiones integrales:

$$a_0 = \frac{1}{T_0}\int_{t_0}^{t_0+T_0} x(t)dt$$

$$a_n = \frac{2}{T_0}\int_{t_0}^{t_0+T_0} x(t)\cos(n\omega_0 t)dt$$

$$b_n = \frac{2}{T_0}\int_{t_0}^{t_0+T_0} x(t)\sin(n\omega_0 t)dt$$

3. Obtener los coeficientes C_n y θ_n correspondientes:

$$C_0 = a_0 \qquad C_n = \sqrt{a_n^2 + b_n^2} \qquad \theta_n = -\arctan\left(\frac{b_n}{a_n}\right)$$

4. Representar gráficamente el espectro de Fourier.

5. Representar, a partir de los coeficientes, las aproximaciones a la función original obtenidas para distintos valores de N al truncar la serie de Fourier:

$$x(t) \approx a_0 + \sum_{n=1}^{N} a_n\cos(n\omega_0 t) + b_n\sin(n\omega_0 t) \equiv x_N(t)$$

6. Calcular la SNR resultante de aproximar la función $x(t)$ por su aproximación hasta el coeficiente N-ésimo $x_N(t)$:

$$\text{SNR} = \frac{\sigma_x^2}{\sigma_e^2} \qquad \text{SNR(dB)} = 10\log_{10}(\text{SNR}) \qquad e(t) = x(t) - x_N(t)$$

y representar gráficamente la SNR en función de N.

7. Calcular los coeficientes G_n de la serie exponencial de Fourier, a partir de la expresión integral:

$$G_n = \frac{1}{T_0}\int_{t_0}^{t_0+T_0} x(t)e^{-jn\omega_0 t}dt$$

8. Verificar la relación entre los coeficientes a_n, b_n y G_n:

$$a_n = (G_n + G_{-n}) \qquad b_n = j(G_n - G_{-n})$$

$$G_n = \frac{1}{2}(a_n - jb_n) \qquad G_{-n} = \frac{1}{2}(a_n + jb_n)$$

9. Repetir los apartados 2, 3, 4, 5 y 6 para trenes de pulsos rectangulares con τ cada vez menor y con A cada vez mayor, manteniendo constante el producto $A\tau$.

1.2. Linealidad del desarrollo en serie

1. Preparar una señal que sea combinación lineal de dos señales de tipo tren periódico de pulsos rectangulares con un mismo periodo:

$$x(t) = k_1 \ \texttt{pulso}\,(t, T_0, \tau_1, A_1) + k_2 \ \texttt{pulso}\,(t, T_0, \tau_2, A_2)$$

2. Calcular los coeficientes a_0, a_n y b_n para ambos trenes de pulsos y para la combinación lineal. Verificar que los coeficientes son también combinación lineal (con los mismos coeficientes k_1 y k_2) y que los espectros también son combinación lineal.

3. Representar los espectros de las tres señales involucradas.

4. Representar las aproximaciones hasta el término N-ésimo de la señal combinación lineal.

1.3. Análisis de otras señales periódicas

1. Repetir los apartados 1 a 6 de la primera sección para las señales siguientes:

   - Una señal triangular periódica.
   - Una señal rampa.
   - Una señal generada a partir de un polinomio de grado 3 en el intervalo $[-T_0/2, T_0/2]$.

Apéndice: Ejemplo de rutina para implementar una función periódica con MatLab

```
% function   x = pulso(t,T0,tau,A)
% Sintaxis:  x = pulso(t,T0,tau,A)
% Devuelve en x(t) un tren periódico de pulsos,
%       de periodo T0,
%       de anchura tau,
%       de amplitud A
% El vector x tiene la misma estructura que el vector t

function x = pulso(t,T0,tau,A)

% llevamos t al intervalo [-T0/2,T0/2]:
k = floor(abs(t)/T0); t1 = sign(t).*(abs(t)-k*T0); t1 = t1 -
T0*(t1>T0/2); t1 = t1 + T0*(t1<=-T0/2);

% definimos la función en el intervalo [-T0/2,T0/2] y la aplicamos
% sobre t1:
x = (abs(t1)<=tau/2)*A;
```

Práctica 2

TRANSFORMADA DE FOURIER

El objetivo de esta práctica es construir una aproximación numérica de la transformada de Fourier y la transformada inversa de Fourier. También se van a estudiar las principales propiedades de la transformada de Fourier sobre distintas funciones.

2.1. Transformada y transformada inversa de Fourier

1. Preparar una función $x(t)$ Gaussiana:

$$x(t) = \mathcal{N}(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

   para unos valores determinados de μ y σ, tomando valores de t desde la media menos varias desviaciones estándar hasta la media más varias desviaciones estándar.

2. Preparar una rutina de MatLab que calcule la transformada de Fourier de una señal especificada mediante los vectores t y x para la frecuencia ω (o para un vector de frecuencias):

```
X = tr_four(t,x,w)
```

   donde $X(\omega)$ se aproxima del modo siguiente:

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \approx \sum_i x_i \exp(-j\omega t_i) \Delta t_i$$

3. Calcular la transformada de Fourier para la Gaussiana $x(t)$ preparada en el apartado 1, para un conjunto de adecuado de frecuencias ω (un conjunto de valores suficientemente denso y suficientemente extenso). Representar la transformada de Fourier (módulo y fase de $X(\omega)$ en función de ω).

4. Preparar una rutina de MatLab que calcule la transformada inversa de Fourier de un espectro especificado mediante los vectores ω y X, para un instante de tiempo t (o para un vector de tiempos t):

```
x = tr_inv_four(w,X,t)
```

donde $x(t)$ se aproxima del modo siguiente:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}d\omega \approx \sum_i X_i \exp(-j\omega_i t)\Delta\omega_i$$

5. Calcular la transformada inversa del espectro $X(\omega)$ obtenido en el apartado 3, y compararla con $x(t)$.

6. Repetir los puntos anteriores para una función $x(t)$ de tipo pulso rectangular de amplitud A y duración τ.

2.2. Propiedades de la transformada de Fourier

En esta parte de la práctica se estudiarán las propiedades de la transformada de Fourier con algunos ejemplos.

2.2.1. Linealidad

Preparar dos señales $x_1(t)$ y $x_2(t)$ de tipo Gaussiano, con diferentes valores de μ y σ, y una combinación lineal $x(t) = k_1x_1(t) + k_2x_2(t)$. Calcular las transformadas de Fourier de $x_1(t)$, $x_2(t)$ y $x(t)$ y verificar que $X(\omega) = k_1X_1(\omega) + k_2X_2(\omega)$.

2.2.2. Dualidad o simetría

Preparar una señal $x(t)$ de tipo Gaussiano, con media distinta de cero y una varianza distinta de 1. Calcular su transformada de Fourier $X(\omega)$ para un conjunto adecuado de valores ω. Calcular la transformada de Fourier de $X(\omega)$ (para ello, asignar $x_1 = X$ y $t_1 = \omega$ y calcular la transformada de $x_1(t_1)$, a la que llamaremos $X_1(\omega)$). Comparar $X_1(\omega)$ con $2\pi x(-t)$ mediante:

```
plot(w,abs(X1),-t,2*pi*x)
```

2.2.3. Escalado

A partir de una función $x(t)$ Gaussiana, calcular la transformada de Fourier de $x(at)$, multiplicando la variable de tiempo por una constante a (es decir, `Xa=tr_four(a*t,x,w)`). Verificar que:

$$X_a(\omega) = \frac{1}{|a|}X\left(\frac{\omega}{a}\right)$$

Hacerlo para valores de a mayores y menores que la unidad. Repetirlo para valores negativos de a.

2.2.4. Desplazamiento en el tiempo

Calcular las transformadas de Fourier de funciones $x(t)$ Gaussianas con distintos valores de la media. Comparar tanto los módulos como las fases de las transformadas de Fourier.

2.2.5. Desplazamiento en frecuencia (modulación)

Preparar una función $x(t)$ que sea el producto de una Gaussiana por una exponencial compleja:

$$x(t) = \mathcal{N}(t, \mu, \sigma)e^{j\omega_0 t}$$

y calcular su transformada de Fourier. Analizar el resultado. Repetirlo para el caso en que la Gaussiana se multiplica por una función $\cos(\omega_0 t + \theta_0)$.

2.2.6. Derivada con respecto al tiempo

A partir de $x(t)$ Gaussiana, obtener la derivada con respecto al tiempo:

$$y(t) = \frac{d\,x(t)}{dt} \approx \frac{x_i - x_{i-1}}{t_i - t_{i-1}} = \frac{x_i - x_{i-1}}{\Delta t}$$

Calcular $Y(\omega)$ (la transformada de Fourier de $y(t)$) y compararla con $j\omega X(\omega)$.

2.2.7. Derivada con respecto a la frecuencia

A partir de $X(\omega)$ del apartado anterior, calcular la derivada con respecto a la frecuencia:

$$Z(\omega) = \frac{d\,X(\omega)}{d\omega} \approx \frac{X_i - X_{i-1}}{\omega_i - \omega_{i-1}} = \frac{X_i - X_{i-1}}{\Delta\omega}$$

Calcular $z(t)$ (transformada inversa de $Z(\omega)$) y compararla con $-jtx(t)$.

2.2.8. Convolución en el dominio del tiempo

Preparar una función $x_1(t)$ Gaussiana multiplicada por $\cos(\omega_0 t + \theta_0)$. Preparar una función $x_2(t)$ Gaussiana. Calcular la convolución $x(t) = x_1(t) * x_2(t)$. Calcular las transformadas de Fourier $X_1(\omega)$ y $X_2(\omega)$ y calcular la convolución $x(t)$ como la transformada inversa del producto de los espectros. Comparar ambas aproximaciones a la convolución.

2.2.9. Producto en el dominio del tiempo (ventanas temporales)

Considerar $x_1(t)$ una función Gaussiana y $x_2(t)$ una función senoidal. Calcular el producto de ambas, $x(t)$ y su transformada de Fourier $X(\omega)$. Comparar esta última con la convolución de los espectros $X_1(\omega) * X_2(\omega)$.

2.2.10. Transformadas de las partes par/impar y real/imaginaria

Preparar la función siguiente:

$$x(t) = \mathcal{N}(t, \mu_1, \sigma_1)\cos(\omega_1 t + \theta_1) + j\mathcal{N}(t, \mu_2, \sigma_2)\cos(\omega_2 t + \theta_2)$$

y a partir de $x(t)$ preparar las siguientes funciones:

- $x_p(t) = (x(t) + x(-t))/2$ (parte par de $x(t)$)
- $x_i(t) = (x(t) - x(-t))/2$ (parte impar de $x(t)$)
- $x_{re}(t) = real(x(t))$ (parte real de $x(t)$)
- $x_{im}(t) = imag(x(t))$ (parte imaginaria de $x(t)$)
- $x_{re-p}(t)$ (parte real-par de $x(t)$)
- $x_{re-i}(t)$ (parte real-impar de $x(t)$)
- $x_{im-p}(t)$ (parte imaginaria-par de $x(t)$)
- $x_{im-i}(t)$ (parte imaginaria-impar de $x(t)$)

Obtener la transformada de Fourier de cada una de estas funciones y estudiar la paridad/imparidad de sus partes reales e imaginarias para cada una de ellas.

2.2.11. Teorema de Parseval

Preparar la función (real) siguiente:

$$x(t) = \mathcal{N}(t, \mu_1, \sigma_1)\cos(\omega_1 t + \theta_1)$$

Calcular su energía en el dominio del tiempo:

$$E_x = \int_{-\infty}^{\infty} x^2(t)dt \approx \sum_i x_i^2 \Delta t_i$$

Compararla con la energía calculada en el dominio de la frecuencia:

$$E_X = \frac{1}{2\pi}\int_{-\infty}^{\infty} |X(\omega)|^2 d(\omega) \approx \frac{1}{2\pi}\sum_i (abs(X_i))^2 \Delta\omega_i$$

Apéndice 1: Ejemplo de rutina para generar funciones Gaussianas

```
% function x = gaussiana(t,mu,sig)

function x = gaussiana(t,mu,sig)

lim=log(1e100); % (es importante poner este limite)
var=sigˆ2;
E=-(t-mu).ˆ2/(2*var);
norma=sqrt(2*pi*var);
E0=max(E); umbral=E0-lim;
E=E.*(E>umbral)+umbral.*(E<umbral);
x=exp(E)/norma;
```

Apéndice 2: Rutina para obtener la transformada de Fourier

```
% function X = tr_four(t,x,w)
% presupone que t, x y w son vectores fila
% presupone muestreo uniforme de t (no necesariamente de w)

function X = tr_four(t,x,w)

deltat=t(2)-t(1);
A=(-j*w'*t);
B=exp(A);
X=deltat*(x*B');
```

Apéndice 3: Rutina para obtener la transformada inversa de Fourier

```
% function x = tr_inv_four(t,X,w)
% presupone que t, X y w son vectores fila
% presupone muestreo uniforme de w (no necesariamente de t)

function x = tr_inv_four(w,X,t)

deltaw=w(2)-w(1);
A=(j*t'*w);
B=exp(A);
X=2*pi*deltat*(X*B');
```

Apéndice 4: Rutina para calcular la convolución de dos funciones

```
% function x = convolucion(t,x1,x2)
% calcula la integral de convolucion de x1(t)*x2(t) para cada valor (t)
% presupone que x1, x2 estan muestreados para los mismos valores de t
% presupone muestreo uniforme

function x = convolucion(t,x1,x2)

deltat=t(2)-t(1);
K=length(t);
i0=-round(t(1)/deltat);   % porque   i(t)=round((t-t(1))/deltat)+1
                          %          t(i)=t(1)-deltat+i*deltat
for k=1:K            % indice para t
   i1=1:K;           % indice para tau      tau=t(i1)
   i2=k-i1+1+i0;     % indice para (t-tau)  t-tau=t(i2)
                     %                 donde i2=i(t-tau)=i(t(k)-t(i1))
   i_ini=k-K+1+i0;   % nos limitamos a las muestras para las que tenemos
   if i_ini<1        %                 definidas las funciones
      i_ini=1;
   end
   i_fin=k+i0;
   if i_fin>K
      i_fin=K;
   end

   x(k)=deltat*sum(x1(i1(i_ini:i_fin)).*x2(i2(i_ini:i_fin)));
end
```

Práctica 3

ESTUDIO DE LOS EFECTOS DE CANAL

En esta práctica estudiaremos los efectos del canal sobre los sistemas de comunicación en banda base. En concreto simularemos los efectos de distorsión lineal y no-lineal y el efecto multi-path.

3.1. Distorsión lineal

El primer sistema que simularemos es un canal paso-baja. Estudiaremos su comportamiento cuando la entrada es una señal rectangular periódica. El diagrama de bloques es como el de la figura siguiente.

3.1.1. Creación Del modelo

Para crear un nuevo modelo utilice la opción File/New/Model de la ventana de comando de Matlab. Una vez creado el modelo, añada y conecte los bloques entre si en la forma que muestra la figura 3.1.

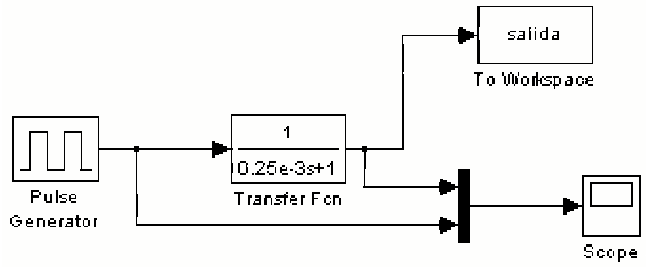

Figura 3.1: *Modelo de canal paso-baja.*

3.1.2. Bloques de diseño

Pulse Generator Este bloque genera un tren de pulsos de periodo y ancho variables. Los parámetros se fijan en la forma:

```
Periodo(secs): 1e-3 Ancho(secs): 0.5e-3
```

Transfer Fcn Este bloque implementa una función de transferencia arbitraria H(s). Los parámetros del modelo son los coeficientes de los polinomios en s del numerador y denominador de la función de transferencia (recuerde que $s = j\omega$). Por ejemplo, la función de transferencia:

$$H(s) = \frac{as^2 + bs + c}{ds^3 + es^2 + f}$$

se describiría en la forma:

```
Numerator: [a b c] Denominator: [d e 0 f]
```

Nótese que los coeficientes se introducen en orden descendente de potencias de s, y que también hay que especificar los nulos. En el ejemplo que nos ocupa, consideraremos la función de transferencia

$$H(s) = \frac{1}{1 + j\frac{\omega}{\omega_o}} \quad \Rightarrow \quad H(s) = \frac{1}{1 + \frac{s}{\omega_o}}$$

Para el ejemplo considere un valor $\omega_0 = 4000$, para este caso, los parámetros deberían ser:

```
Numerator: [1] Denominator: [0.25e-3 1]
```

Mux Este bloque multiplexa las diferentes entradas generando un vector de salida. Los parámetros son:

```
Number of inputs: 2
```

Este bloque se utiliza para multiplexar dos o más señales. La salida es un vector (bus) en lugar de una línea escalar. En este caso se utiliza para insertar dos señales (entrada y salida de H(s)) al osciloscopio de forma que se visualicen simultáneamente.

Scope Este bloque simula un osciloscopio. Es decir, visualiza las señales que se aplican a su entrada frente al tiempo de simulación del sistema.

```
Horizontal range: 0.004 Vertical range: 2
```

To Workspace Este bloque muestrea la señal que se aplica a su entrada y almacena las muestras en una variable que es accesible a Matlab al terminar la simulación.

```
Variable name: salida Save Format: array Maximum number of rows
(time steps): [4000,1,1/20000]
```

Estos parámetros especifican que la variable salida almacenará valores en instantes temporales muestreados al Periodo 1/20000 (frecuencia 20KHz), de uno en uno, con un máximo de 4000. Al finalizar la simulación, la variable aparecerá Matlab, y se podrá utilizar para ulterior análisis de los datos.

3.1.3. Simulación

Para iniciar la simulación, primero elegiremos los parámetros. Para esta simulación fijaremos los parámetros:

```
Simulation algorithm: Runge Kutta 5
Start Time: 0
Stop Time: 0.004
Min Step Size: 1e-9
Max Step Size: 10
Tolerance: 1e-5
Return Variables:
```

Una vez insertados los bloques y fijados los parámetros de la simulación, ya pude simular el modelo, pero antes sálvelo con `File/Save...` con el nombre `cpbaja.m`. Simule ahora el modelo eligiendo `Simulation/Start`. Según va avanzando la simulación, en el osciloscopio de puede ver la evolución temporal de la entrada y salida al sistema.

Análisis temporal

Una vez la simulación ha finalizado, en la ventana de comandos de Matlab puede ejecutar *whos* y encontrará que se ha creado una variable denominada salida que debe contener 81 elementos reales, correspondientes a las muestras de la salida. Visualícela con `plottime (salida,20000)`. El primer argumento es la variable y el segundo la frecuencia con que fue muestreada (20KHz en nuestro caso). Deberá obtener una gráfica de la señal de salida. Si ejecuta ahora `zoomtool` (no disponible en versiones nuevas de Matlab, en este caso use solo `plot(salida)`), a la gráfica se le superponen una serie de botones y campos de texto más dos cursores. Su significado es el siguiente:

```
> Desplazar el cursor una muestra a la derecha
< Ídem a la izquierda
>> Desplazar al siguiente máximo/mínimo a la derecha del cursor
<< Ídem a la izquierda
>< Ampliar la señal entre los dos cursores
<> Restaurar la señal a su tamaño original [] Retrazar la señal S
Crear una nueva figura con la vista actual Q Terminar zoomtool
dejando la figura con su aspecto actual
```

También se pueden desplazar los cursores arrastrándolos con el ratón (haga clic sobre la línea del cursor y desplace el ratón; luego suelte el botón del ratón). Además, en la parte inferior izquierda se muestran los valores X e Y de cada cursor así como la diferencia entre estos valores (en el centro).

Cuestiones

Con ayuda de `zoomtool`, mida los valores máximo (A1) y mínimo (A2) de la señal y calcule la relación (A1-A2)/A1, que es la separación relativa entre niveles (una medida de la interferencia inter-simbólica introducida por el canal), para valores de ancho de pulsos de 0.25e-3, 0.5e-3 y 0.75e-3. Compruebe que los resultados concuerdan con los previstos en teoría. Para una mayor precisión en las medidas, realice estas sobre los últimos periodos de la señal. De esta forma evitará los efectos del transitorio inicial.

3.2. Distorsión no-lineal

En esta segunda parte estudiaremos los efectos de la característica no lineal de transferencia del canal. Consideraremos un modelo como el de la figura.

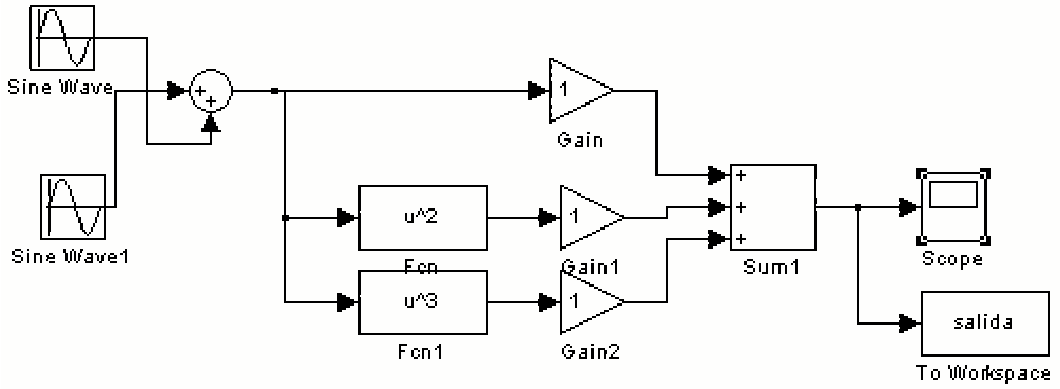

Figura 3.2: *Modelo de canal no-lineal.*

Este modelo genera una señal de la forma

$$x(t) = a_1 \sin(\omega_1 t) + a_2 \sin(\omega_2 t)$$

que es transmitida a través del canal cuya función de transferencia en amplitud es de la forma:

$$y(t) = g_1 x(t) + g_2 x^2(t) + g_3 x^3(t)$$

3.2.1. Bloques de diseño

Los nuevos bloques de diseño son:

Sin Wave Genera una señal seno de amplitud, frecuencia y fase constantes.

Fcn Aplica una transformación arbitraria a la entrada.

Gain Amplificador de ganancia constante.

3.2.2. Simulación

Ajuste los parámetros del modelo para conseguir una entrada:

$$x(t) = \sin(1000\pi t) + \sin(4000\pi t)$$

y las ganancias para conseguir una característica lineal del canal y una frecuencia de muestreo para la salida de 25600 Hz. Simule el sistema durante 0.05 segundos. Al terminar la simulación, puede visualizar la señal de salida con `plottime(salida,25600)` y ampliar una parte con `zoomtool` para ver los detalles (o cualquier otra herramienta de Matlab).

Análisis en frecuencia

Para analizar en frecuencia la señal de salida, utilizaremos la función `cpsd`. Esta función obtiene una estimación de la densidad de potencia de una señal así como la fase de la misma. En el formato más sencillo se invoca con `cpsd(x,fs,nfft)` donde x es la señal, fs la frecuencia de muestreo y nfft el numero de puntos que se calculan de ésta. La resolución espectral está dada por fs/nfft. Con este formato, `cpsd` visualiza la psd de la señal en decibelios. Si se invoca con el formato `cpsd(x,fs,nfft,1)` visualiza además la fase de la señal. Se puede invocar también con el formato `[pxx,fxx,f]=psd(x,fs,nfft)` para obtener los vectores de potencia pxx, fase fxx y frecuencia f. De esta forma se puede trazar la psd en escala lineal con `plot(f,pxx)`. Utilizando `cpsd(salida,25600,256)` y `zoomtool` obtenga el espectro de la salida del sistema. Este debería contener dos picos centrados en frecuencias 500Hz y 2000Hz. Sus amplitudes deben ser de -6dB. Esto es debido a que cada señal seno de amplitud 1 genera dos deltas de amplitud 1/2 centradas en la frecuencia positiva y negativa de la sinusoide. Cada una de estas deltas contribuye a la densidad de potencia espectral.

Cuestiones

Simule el sistema con funciones de transferencia en amplitud

$$a) \quad y(t) = x(t) + \frac{1}{2}x^2(t) \qquad b) \quad y(t) = x(t) + \frac{1}{2}x^2(t) + \frac{1}{4}x^3(t)$$

Compruebe que los resultados concuerdan con las predicciones teóricas. ¿Cuál es el ancho de banda de la señal de salida en cada caso? ¿Qué armónicos aparecen a la salida?

3.3. Efecto *multi-path*

En este último apartado consideraremos el efecto multi-path causado por la superposición de un eco retardado y atenuado de la señal. Consideraremos un modelo como el de la figura.

3.3.1. Bloques de diseño

Los nuevos bloques son los siguientes:

Transport Delay Implementa un retardo temporal fijo entre su entrada y salida.

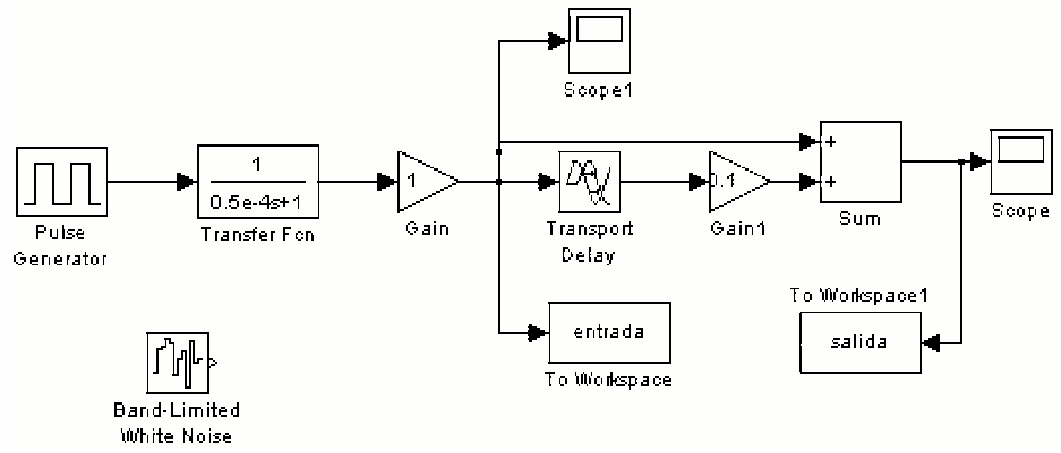

Figura 3.3: *Modelo de canal multipath.*

Band-Limited Wait Noise Genera un ruido blanco (densidad de potencia espectral uniforme) de ancho de banda limitado. El ancho de banda se elige de forma que si el período de muestreo es T, el ancho de banda resultante es $B = 1/2T$.

3.3.2. Simulación

Ajuste el generador de pulsos para un Periodo de 1ms y un ancho de 0.5ms. Simule el sistema durante 0.1 segundos. Muestree la salida a una frecuencia de 25600Hz y fije un máximo número de muestras superior a 2561 (p.e. 3000). Con estos parámetros, observe en el osciloscopio la señal de salida para diferentes valores de retardo y ganancia del eco.

Análisis en frecuencia

A continuación obtendremos una caracterización de la función de transferencia del sistema. Desconecte el bloque que implementa la función de transferencia H(s) y alimente ahora el sistema con el generador de ruido blanco. Ajuste sus parámetros para generar un ruido limitado en banda a 12800Hz (Período de muestreo de 1/25600) y una potencia de 1e-4 (para un rango de amplitudes aceptable en el osciloscopio). Con valores de 0.25ms de retardo y 0.1 de ganancia, simule el sistema durante 0.1s. Al terminar la simulación, visualice la función de transferencia con

```
ctfe(entrada,salida,25600,256,1)
```

(el formato es muy similar al de la función `cpsd` salvo que tiene dos variables de entrada en lugar de una). Observará el comportamiento oscilatorio tanto de la amplitud como de la fase del sistema. Si ahora obtiene los vectores de amplitud y fase con:

```
[txx,fxx,f]=ctfe(entrada,salida,25600,256)
```

podrá visualizar el módulo con `plot(f,txx)` y la fase con `plot(f,fxx)`, pudiendo utilizar `zoomtool` para realizar medidas sobre ellos.

Cuestiones

Mediante el proceso descrito anteriormente, obtenga la función de transferencia del sistema multipath para ganancia 0.1 y valores de retardo de 0.5ms, 0.25ms y 0.125ms. Compruebe que los resultados concuerdan con los predichos por la teoría. ¿Qué ocurre si se aumenta la ganancia de 0.1 a 0.75? Explique el efecto que se produce.

Apéndice 1: Rutina para obtener la PSD

```
function [pxx,fxx,f] = cpsd(x,fs,nfft,fasesw)
%function [pxx,fxx,f] = cpsd(x,fs,nfft,fasesw)
% Estima 'nfft' puntos de la PSD de 'x' muestreada a 'fs' Hz
% cpsd(x,fs,nfft) Traza la PSD en escala logaritmica
% cpsd(x,fs,nfft,fasesw) Si fasesw==1 también se traza la fase
% [pxx,fxx,f] = cpsd(x,fs,nfft) Devuelve la PSD en 'pxx',
%   la fase en 'fxx' y el vector de frecuencias en 'f'
  if nargin < 3
    disp('Formato: cpsd(x,fs,nfft)');
  else
  if nargin == 3, fasesw = 0; end
% Cálculo de la PSD y FASE
    npts = length(x);
    w=hanning(nfft);
    h = fft(x(1:nfft).*w,nfft);
    psd = abs(h).^2;
    fase = unwrap(angle(h));
    n = nfft;
    np = 1;
    while n <= npts - nfft;
      h = fft(x(n+1:n+nfft).*w,nfft);
      psd = psd + abs(h).^2;
      fase = fase + unwrap(angle(h));
      n = n + nfft/2;
      np = np + 1;
    end
    psd  = psd(1:nfft/2+1) / (norm(w)^2 * np);
    fase = fase(1:nfft/2+1) / np;
    fvec = fs*(0:nfft/2)/nfft;

    if nargout == 0
      psd = 10*log10(psd*norm(w)^2/sum(w)^2);
      fase = fase / pi;
      if fasesw == 1
```

```
        subplot(2,1,1)
          plot(fvec,psd);
          set(gca,'XLim',[0,fs/2]);
          xlabel('Frecuencia (Hz)');
          ylabel('PSD (dB)');
        subplot(2,1,2);
          plot(fvec,fase);
          set(gca,'XLim',[0 fs/2]);
          xlabel('Frecuencia (Hz)');
          ylabel('Fase (xPI)');
      else
        plot(fvec,psd);
        set(gca,'XLim',[0,fs/2]);
        xlabel('Frecuencia (Hz)');
        ylabel('PSD (dB)');
      end
    else
      pxx = psd / length(psd);
      fxx = fase;
      f   = fvec;
    end
  end
end
```

Apéndice 2: Rutina para obtener la función de transferencia

```
function [txx,fxx,f] = ctfe(x,y,fs,nfft,fasesw)
%function [txx,fxx,f] = ctfe(x,y,fs,nfft,fasesw)
% Estima 'nfft' puntos de la TFE de 'x' e 'y' muestreada a 'fs' Hz
% ctfe(x,y,fs,nfft) Traza la TFE en escala lienal
% ctfe(x,y,fs,nfft,fasesw) Si fasesw==1 también se traza la fase
% [txx,fxx,f] = ctfe(x,y,fs,nfft) Devuelve la TFE en 'txx',
%   la fase en 'fxx' y el vector de frecuencias en 'f'
  if nargin < 4
    disp('Formato: ctfe(x,y,fs,nfft)');
  else
  if nargin == 4, fasesw = 0; end
% Cálculo de la TFE y FASE
    w=hanning(nfft);
    [p,fvec] = psd(x,nfft,fs,w,nfft/2,'none');
    c = csd(x,y,nfft,fs,w,nfft/2,'none');
    tfe = abs(c)./abs(p);
    fase = unwrap(angle(c));
    if nargout == 0
      fase = fase / pi;
```

```
      if fasesw == 1
        subplot(2,1,1)
          plot(fvec,20*log10(tfe));
          ylabel('TFE (dB)');
          xlabel('Frecuencia (Hz)');
        subplot(2,1,2);
          plot(fvec,fase);
          xlabel('Frecuencia (Hz)');
          ylabel('Fase (xPI)');
      else
        plot(fvec,20*log10(tfe));
        ylabel('TFE (dB)');
        xlabel('Frecuencia (Hz)');
      end
    else
      txx = tfe;
      fxx = fase;
      f   = fvec;
    end
  end
```