TRATAMIENTO DIGITAL DE LA SEÑAL
Señales y Sistemas Discretos

Práctica 1

8 de marzo de 2010

1 Introducción

Esta práctica se centra en la generación con MATLAB de algunas señales básicas de tiempo discreto. Estas señales, usadas en el tratamiento digital de señales, son el impulso, $\delta(n)$, las exponenciales de la forma, $a^n u(n)$, las ondas sinusoidales y su generalización a exponenciales complejas.

Puesto que el único tipo posible de dato numérico en MATLAB es la matriz $M \times N$, las señales deben representarse con vectores: ya sea como matrices $M \times 1$ si son vectores columna, o como matrices $1 \times N$ si son vectores fila. En MATLAB todas las señales deben ser de longitud finita. Esto contrasta con la solución analítica de los problemas, donde una fórmula matemática puede representar señales de longitud infinita: por ejemplo, una exponencial decreciente $a^n u(n)$.

Una segunda cuestión es asociar al vector de señal los índices del dominio temporal. MATLAB asume por defecto como índices desde 1 a N, siendo N la longitud de dicho vector. Mientras que el vector es normalmente el resultado de muestrear una señal sobre algún dominio donde los índices van desde 0 a $N-1$; o quizá, el muestreo comience en algún índice arbitrario que sea negativo; por ejemplo, $-N$. La información sobre el dominio de muestreo no se puede ligar a los valores de la señal contenidos en el vector de señal. Por ello, estamos obligados a conservar separadas ambas informaciones. Normalmente, esto no será un problema hasta que se represente gráficamente la señal, en tal caso habrá que etiquetar adecuadamente el eje horizontal.

Una última cuestión que se abordará es el uso de la notación vectorial de MATLAB para generar señales. Una potencia significativa del entorno MATLAB es su notación de alto nivel para operar con vectores: los bucles `for` son casi siempre innecesarios. Cuando generemos algunas señales, por ejemplo, una onda sinusoidal, es preferible aplicar la función correspondiente. Para este ejemplo concreto sería la función `sin`, a un vector argumento formado con todas las muestras temporales. En la práctica trataremos con algunas de las señales que nos encontraremos habitualmente en tratamiento digital de la señal: impulsos, trenes de impulsos, exponenciales y sinusoides.

1.1 Recomendaciones

La representación gráfica en MATLAB de señales de tiempo discreto se realiza con la función `stem`. El siguiente código de MATLAB genera 31 puntos de una sinusoide de tiempo discreto:

```
nn = 0:30; sinus = sin(nn/2 +1);
```

Observe que el valor de tiempo discreto $n = 0$ se almacena como la primera muestra de nn, es decir: `nn(1)`, debido al esquema que utiliza MATLAB para asociar índices; del mismo modo que `sinus(1)` es el primer valor de la sinusoide almacenada en dicho vector. Cuando representemos gráficamente la onda sinusoidal utilizaremos la función `stem`, función que realiza la representación gráfica de la señal de tiempo discreto vista comúnmente en los libros de tratamiento digital de la señal: `stem(nn,sinus);`.

El primer vector argumento debe generar correctamente el eje n. Compare con `stem(sinus)`. ¿En qué se diferencian las dos gráficas?

2 Señales Básicas

2.1 Impulsos

La señal más simple es la señal impulso unidad (desplazado):

$$\delta(n - n_0) = \begin{cases} 1 & n = n_0 \\ 0 & n \neq n_0 \end{cases} \tag{1}$$

Cuando creamos un impulso en MATLAB debemos decidir qué intervalo de la secuencia nos interesa. Si el impulso $\delta(n)$ se utiliza para excitar un sistema LTI causal, y suponiendo que necesitamos ver L puntos, desde $n = 0$ a $n = L-1$ podemos crear este "impulso" con el siguiente código de MATLAB, en el cual se ha tomado $L = 31$:

```
L = 31;
nn = 0 : (L-1);
imp = zeros(L,1);
imp(1) = 1;
```

Observe que a la primera muestra, que se corresponde con $n = 0$, se accede con `imp(1)`, debido al esquema de asignación de índices de MATLAB.

Se pide:

1. Genere y represente gráficamente las siguientes secuencias:

$$\begin{array}{cc} x_1(n) = 0.9\delta(n-5) & 1 \leq n \leq 20 \\ x_2(n) = 0.8\delta(n) & -15 \leq n \leq 15 \\ x_3(n) = 1.5\delta(n-333) & 300 \leq n \leq 350 \\ x_4(n) = 4.5\delta(n+7) & -10 \leq n \leq 0 \end{array}$$

   En cada caso el eje horizontal, n, debe extenderse solamente sobre el intervalo indicado y numerarse de manera adecuada. Cada secuencia deberá visualizarse, mediante `stem`, como una señal de tiempo discreto.

2. El siguiente código de MATLAB creará una señal repetitiva en el vector **x**:

   ```
   x = [0; 1; 1; 0; 0; 0] * ones(1,7);
   x = x(:);
   size(x);      % Devuelve la longitud de la señal
   ```

   Represente **x** para ver su forma y, a continuación, obtenga una fórmula matemática similar a (2) que describa esta señal. ¿Qué hace la instrucción $x(:)$?

3. Los impulsos desplazados, $\delta(n - n_0)$, pueden usarse para construir trenes de impulsos ponderados, con período P y longitud total MP:

$$s(n) = \sum_{l=0}^{M-1} A_l \delta(n - lP - n_0) \tag{2}$$

   Si los pesos, A_l, son todos iguales el tren de impulsos es periódico con período P. Genere y represente gráficamente un tren de impulsos periódicos, cuyo período sea $P = 5$, longitud 50, $n_0 = 0$ y $A_l = 3$ para $l = 0, \cdots, M-1$. ¿Con qué valor de M se consigue esto? La señal debe comenzar en $n = 0$. Intente utilizar una o dos operaciones con vectores en vez de un lazo `for`, para situar los impulsos. ¿Cuántos impulsos contiene esta señal de longitud finita?

2.2 Sinusoides

Otra señal básica es la onda cosenoidal. En general, se tendrán tres parámetros básicos para describir completamente una señal sinusoidal: la amplitud (A), la frecuencia (w) y la fase (ϕ).

$$x(n) = Acos(w_0 n + \phi) \tag{3}$$

Se pide:

1. Genere y represente gráficamente cada una de las siguientes secuencias. Utilice la capacidad vectorial de MATLAB para hacerlo con la llamada a la función que realiza el coseno (o seno) de un argumento vector. En cada caso, el eje horizontal, n, se extenderá solamente sobre el intervalo indicado y deberá ser numerado adecuadamente. Cada secuencia se representará aplicando la función `stem`.

$$\begin{array}{cc} x_1(n) = sen(\frac{\pi}{17}n) & 0 \leq n \leq 25 \\ x_2(n) = sen(\frac{\pi}{17}n) & -15 \leq n \leq 25 \\ x_3(n) = sen(3\pi n + \frac{\pi}{2}) & -10 \leq n \leq 11 \\ x_4(n) = cos(\frac{\pi}{\sqrt{23}}n) & 0 \leq n \leq 50 \end{array}$$

   Observando las gráficas anteriores, obtenga la fórmula sencilla para $x_3(n)$, que no haga uso de funciones trigonométricas. ¿Es esta secuencia periódica? ¿Por qué? Utilizando dos señales iguales a (2), ¿qué valores de A_l, M y P tendría que considerar en éstas para que $x_3(n) = s(n)$?

2. Escriba una función en MATLAB (`help function` para ver su sintaxis) que genere una sinusoide de longitud finita. La función necesitará un total de cinco argumentos de entrada: tres para los parámetros (amplitud, frecuencia y fase) y dos más para especificar el primer y el último valor del índice n de la secuencia de longitud finita. La función deberá devolver un vector columna conteniendo los valores de la sinusoide y se llamará `seno.m`. Esta función se comprobará representando gráficamente los resultados que se obtienen para diferentes parámetros de entrada. En concreto, compruebe la función con la señal $2sen(\pi n/11)$, para $-20 \leq n \leq 20$.

3. Vuelva a escribir la función del apartado anterior para que devuelva dos argumentos: un vector de índices en el intervalo de n y un vector con los valores de la señal correspondientes a esos índices. Llame a dicha función `seno2.m`.

2.3 Exponenciales

La exponencial decreciente es una señal básica en tratamiento digital de señal porque aparece como solución a las ecuaciones en diferencias de coeficientes constantes.

Se pide:

1. Estudie la función de MATLAB presentada a continuación para ver cómo se genera una señal exponencial de tiempo discreto. Seguidamente utilice la función para representar gráficamente la exponencial $x(n) = (0.9)^n$, en el intervalo $n = 0, 1, 2, 3, \cdots, 20$.

```
function y=genexp(c,n0,L)
% GENEXP genera una señal exponencial: c ^ n
%        uso: y = genexp(c,n0,L)
%        c:  entrada escalar que da la razón entre términos
%        n0: instante de comienzo (entero)
%        L:  longitud de la señal generada
%        y: señal de salida Y(1:L)
```

```
if (L <= 0)
  error('GENEXP: longitud no positiva')
end
nn = n0 + [0:L-1].';     % vector de índices
y = c .^ nn;
```

Observe que los comentarios que aparecen después de la definición de la función, se corresponden con el texto que se obtiene al hacer un `help` de ésta.

2. En muchas ocasiones hay que sumar los valores de la secuencia exponencial $\alpha^n u(n)$. Para un intervalo finito esta suma tiene una expresión compacta conocida:

$$\sum_{n=0}^{L-1} \alpha^n = \frac{1-\alpha^L}{1-\alpha} \qquad \text{para } \alpha \neq 1 \tag{4}$$

Utilice la función del apartado anterior para generar una exponencial y después sume sus valores (función `sum`). Compare el resultado con el que se obtiene al aplicar la ecuación (4).

3. Una razón por la que la secuencia exponencial aparece con tanta frecuencia en tratamiento digital de señal es que el desplazamiento en el tiempo no cambia el carácter de esta señal. Demuestre que las exponenciales de longitud finita satisfacen la siguiente relación:

$$y(n) \quad = \quad \alpha y(n-1) \text{ en el intervalo } 0 \leq n \leq L-1 \tag{5}$$

donde $y(n) = \alpha^n$.

Para ello, compare los vectores $y(2:L)$ y $\alpha y(1:L-1)$ (para un valor de α arbitrario). ¿Qué representan los vectores $y(2:L)$ y $y(1:L-1)$? Cuando desplace señales de longitud finita en MATLAB, deberá tener cuidado con los instantes finales porque no se añadirán ceros automáticamente (*zero-padding*).

4. Otra forma de generar una secuencia exponencial es aplicando una fórmula recurrente dada por una ecuación en diferencias. Cuando la entrada $x(n)$ es un impulso la señal $y(n) = \alpha^n u(n)$ es la solución de la siguiente ecuación en diferencias:

$$y(n) \quad = \quad x(n) + \alpha y(n-1) \text{ condición inicial: } y(-1) = 0 \tag{6}$$

Puesto que se ha tomado una ecuación en diferencias recurrente en forma causal[1] se necesita la condición para $n = -1$. En MATLAB la función `filter` implementa una ecuación en diferencias como la de (6). Utilice `filter` para generar la misma señal que en el primer apartado de esta subsección, es decir, $\alpha = 0.9$.

3 Señales Complejas

Aunque las señales por naturaleza tienen amplitudes reales, es muy frecuente que para generarlas, procesarlas e interpretarlas sean transformadas en señales de amplitudes complejas. Esto se hace combinando señales en parejas. Como se hace con las partes real e imaginaria se puede procesar con otras señales de valores complejos usando las leyes de aritmética compleja. En muchos sistemas de tratamiento de señal es importante el uso de estas parejas de señales, especialmente aquellos que implican modulación.

En MATLAB, las funciones `real` e `imag` extraen las partes real e imaginaria de un número complejo. Cuando representamos gráficamente un vector complejo, las representaciones por defecto de `plot` y `stem` pueden llevar a confusiones. Si z es un complejo, entonces `plot(z)` será

[1] Es decir: sólo definida para valores positivos de n.

la representación gráfica de la parte imaginaria en función de la parte real y `plot(n,z)` será la representación de la parte real de z en función de n. Sin embargo, `stem(z)` dibujará únicamente la parte real. Si desea ver simultáneamente las representaciones de las partes real e imaginaria, los comandos `subplot` antepuestos a cada comando `stem`, forzarán a que las representaciones se realicen en la misma ventana, pero en gráficas distintas. Observe el resultado del siguiente código:

```
%  Secuencia
 nn = 0:25;
 xx = exp(j*nn / 3);

% Representación
 subplot(211)
 stem(nn,real(xx))
 title('Parte real')
 xlabel('Indice (n)')
 subplot(212)
 stem(nn,imag(xx))
 title('Parte imaginaria')
 xlabel('Indice (n)')
```

Otra forma de representar los números complejos es mediante su módulo y su fase. Para esto, MATLAB dispone de las instrucciones `abs` y `angle`. Haciendo uso de instrucciones `subplot` represente, en una misma ventana pero en cuatro gráficas distintas, la parte real e imaginaria y su módulo y su fase de la secuencia generada en el código anterior.

IMPORTANTE: Tenga en cuenta que matlab utiliza i y/o j para representar el número complejo $\sqrt{-1}$. Si define variables con estos nombres, la gestión de números complejos de MATLAB fallará sin dar ningún tipo de error.

3.1 Exponenciales Complejas

La notación de la exponencial real puede extenderse a las señales exponenciales de valores complejos, las cuales engloban a las señales seno y coseno. Estas señales forman la base de las transformadas de Fourier.

Se pide:

1. En MATLAB una señal compleja es una extensión natural de la notación de la sección anterior. Por tanto, para generar estas señales, el parámetro a puede tomarse como un número complejo. Recuerde la fórmula de Euler para la exponencial compleja:

$$x(n) \quad = \quad (z_0)^n = r^n e^{j\phi n} = \underbrace{r^n cos(\phi n)}_{parte\ real} + j \quad \underbrace{r^n sen(\phi n)}_{parte\ imaginaria} \tag{7}$$

   donde $z_0 = re^{j\phi}$. Use esta relación para generar una exponencial compleja con módulo, $r = 0.9$, y fase, $\phi = 45^o$. Represente las partes real e imaginaria de $x(n)$ en el intervalo $0 \leq n \leq 20$. Observe como el ángulo de ϕ controla la frecuencia de las sinusoides.

4 Señales Discretas Obtenidas Mediante el Muestreo de Señales Continuas

Esta sección muestra uno de los principios básicos del proceso de muestreo: el solapamiento (o aliasing). Este se produce debido a una mala elección de la frecuencia de muestreo. Para ver sus efectos, se estudiará el muestreo de señales sinusoidales.

Considere una señal sinusoidal de tiempo continuo cuya ecuación es

$$x(t) \quad = \quad sen(2\pi f_0 t + \phi) \tag{8}$$

podemos obtener una señal de tiempo discreto muestreando $x(t)$ a la frecuencia de muestreo $f_s = 1/T_s$:

$$x(n) \quad = \quad x(t)|_{t=nT_s} = x(t)|_{t=n/f_s} = sen(2\pi \frac{f_0}{f_s} n + \phi) \tag{9}$$

Observe que n representa la muestra correspondiente al instante de tiempo nT_s.

Si hacemos la representación de $x(n)$ para diferentes valores de f_0 y f_s se puede ver el efecto del solapamiento en caso de no cumplirse el teorema de muestreo. Recuerde que este teorema dice que $f_s \geq 2f_0$ para evitar el fenómeno de solapamiento. Para los siguientes apartados fijaremos la frecuencia de muestreo en $f_s = 8$Khz.

Se pide:

1. Haga la representación de una onda sinusoidal muestreada. La frecuencia de la onda sinusoidal será de 300Hz, tomándose las muestras correspondientes a 10mseg. ¿Qué longitud tendrá la señal discreta?. La fase ϕ puede ser cualquiera (por ejemplo $\phi = 0$). Represente, con la función `stem`, la señal de tiempo discreto que se obtiene, etiquetando el eje horizontal en unidades de tiempo. Analizando la gráfica obtenida, se apreciará con facilidad que la envolvente de la señal discreta es una sinusoide, debido a que su vista tiende a hacer una reconstrucción de la sinusoide.

   Si es necesario, haga la representación con `plot`. En este caso, los puntos se conectan con líneas rectas, resultando obvio el comportamiento sinusoidal. Conectar las muestras de la señal con líneas rectas es una forma de reconstrucción de señal, obteniéndose una señal de tiempo continuo a partir de las muestras de una señal de tiempo discreto. Tenga en cuenta que ésta no es la reconstrucción ideal de una señal a partir de sus muestras, pero en la mayoría de las ocasiones será suficiente.

2. Usando `stem` al igual que en el anterior apartado, haga una serie de representaciones con las frecuencias $f_0 = 100$Hz, $f_0 = 225$Hz, $f_0 = 350$Hz y $f_0 = 475$Hz. Observe que la frecuencia de la sinusoide se va incrementando, como era de esperar. Utilice un `subplot` para que aparezcan las cuatro representaciones en una misma ventana.

   Es interesante escuchar[2] la señal resultante para comprender mejor el significado de la frecuencia de la sinusoide. Para ello, puede usar la instrucción `sound` (ver `help sound`) o generar un fichero de audio `.wav` (`wavwrite`, ver `help wavwrite`) o `.au` (`auwrite`, ver `help auwrite`) y reproducirlo posteriormente.

3. De la misma forma que en el apartado anterior, realice una serie de representaciones variando la frecuencia de la sinusoide con $f_0 = 7525$Hz, $f_0 = 7650$Hz, $f_0 = 7775$Hz y $f_0 = 7900$Hz. ¿Qué pasa ahora con la frecuencia aparente de las sinusoides? En caso de haber cursado la asignatura medios de transmisión, ¿sabe explicar el motivo?

5 Documentación y Fecha de entrega

La fecha de depósito de la práctica será el día **18 de marzo**. Se deberán depositar los ficheros `*.m` de los programas de matlab en un nuevo directorio del servidor subversion (SVN), llamado "Practica 1", siguiendo las instrucciones de la WIKI de la Facultad

`https://wiki.fic.udc.es/cecafi%3Asvn%3Aindice`

[2] En el laboratorio conecte unos auriculares a la salida de la tarjeta de sonido de su ordenador.

CADA APARTADO DEBE REALIZARSE EN SUBDIRECTORIOS DIFERENTES CON EL NOMBRE DE ESTE. NO SE ACEPTAN FICHEROS COMPRIMIDOS.

Cada alumno ha de efectuar este depósito en su directorio correspondiente y conservar una copia de todos los programas para su evaluación el día del examen de prácticas.