Grado en Ingeniería en Inteligencia Artificial

SEÑALES Y SISTEMAS

Práctica nº 3

Antonio Valle Sánchez

DFESTS

UA | UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE

PRÁCTICA 3
Conversión A/D y D/A

1.- Conversión analógica-digital.

1.1.- Estudio del **muestreo** en el dominio del tiempo.

1.2.- **Cuantificación**.

2.- **Conversión** digital-analógica.

2.1.- Reconstrucción ideal.

2.2.- Reconstrucción práctica.

2.3.- Alteración de la frecuencia de muestreo en reconstrucción.

1.- CONVERSIÓN ANALÓGICA-DIGITAL

En primer lugar, veamos cada una de las limitaciones que nos encontramos en la conversión A/D.

Antes de empezar hay que tener cuidado con una cuestión fundamental: ¡MATLAB trabaja con datos discretos! Puede parecer un poco ilógico usar esta herramienta para explicar una práctica de muestreo (¡los datos ya están muestreados!). La solución adoptada es considerar intervalos de tiempo muy pequeños frente al periodo de muestreo de tal forma que tendremos una señal "quasi-continua". Esto ya se tuvo en cuenta en prácticas anteriores, cuando se utilizó Matlab para representar señales en tiempo continuo.

Para diferenciar entre señales continuas y discretas, emplearemos distintos modos de representación según las señales sean de un tipo u otro. Para representar señales continuas emplearemos la función plot (y similares), y para representar señales discretas emplearemos la función stem. Para comprobar cómo representa las secuencias esta última función, ejecuta las siguientes órdenes en Matlab:

```
>> n=0:19;
>> x=cos(0.125*pi*n);
>> stem(n,x)
```

1.1.- Estudio del muestreo en el dominio del tiempo

A continuación, crea un fichero de extensión M para el siguiente programa que proporciona una señal continua de frecuencia dada y su versión muestreada:

```
f0=input('¿Frecuencia de la sinusoide continua (en Hz)?');
T0=1/f0;
t = [0:2*T0/10000:2*T0];
xa = cos(2*pi*f0*t);
subplot(2,1,1);
plot(t, xa);
grid;
xlabel('Tiempo (s)','Fontsize',8);
ylabel('Amplitud','Fontsize',8);
title('Señal continua x_{a}(t)');
axis([0 2*T0 -1.5 1.5]);
Ts = 0.0001;
nTs = [0:Ts:2*T0];
xs = cos(2*pi*f0*nTs);
n = 0:length(nTs)-1; subplot(2,1,2);
stem(n, xs);
grid;
xlabel('Índice de muestreo, n. T_s=0.1 ms','Fontsize',8);
ylabel('Amplitud','Fontsize',8);
title('Señal discreta x[n]'); axis([0 2*T0/Ts -1.5 1.5]);
```

Como se puede comprobar el programa es bastante incómodo a la hora de cambiar el valor de los parámetros. Haz las debidas modificaciones para convertir el fichero tipo M en una función donde la frecuencia de la señal analógica f0 y el periodo de muestreo Ts sean variables de entrada, y la señal discreta resultante xs sea el vector de salida.

Vamos a utilizar ahora la función para evaluar el Teorema de Muestreo de Nyquist. Varía la frecuencia de la señal continua y el periodo de muestro y escoge un caso donde se cumpla el Teorema de Muestreo, y otro caso donde no se cumpla y se esté produciendo aliasing.

¿Qué ocurre justo cuando la frecuencia de muestreo es igual al doble la frecuencia de la sinusoide analógica? ¿Cuántas muestras se capturan en un periodo de la señal continua?

A continuación modifica en el programa la línea donde aparece:

```
xa = cos(2*pi*f0*t);
```

por

```
xa = cos(2*pi*f0*t-pi/2);
```

y donde aparece

```
xs = cos(2*pi*f0*nTs);
```

por

```
xs = cos(2*pi*f0*nTs-pi/2);
```

Ejecuta otra vez la función tras este cambio y observa qué ocurre ahora en el caso de que la frecuencia de muestreo sea justo la frecuencia de Nyquist de la señal analógica. ¿Será posible reconstruir la señal original a partir de sus muestras?

Para terminar, tomando como base el programa anterior, desarrolla un fichero que te permita representar en la misma gráfica dos señales sinusoidales continuas, de diferentes frecuencias, y sus correspondientes muestreadas. Haz que las frecuencias analógicas cumplan la relación:

$$f_{02} = f_{01} + k \cdot f_S$$

siendo f_{01} y y f_{02} las frecuencias de las sinusoides continuas, k un número entero positivo y f_S la frecuencia de muestreo. ¿Qué ocurre cuando se muestrean estas sinusoides a la misma frecuencia de muestreo f_S?

1.2.- Cuantificación

Una vez que se ha visto el muestreo, pasemos a la siguiente etapa en la conversión A/D: la cuantificación. Comprobarás lo que ocurre cuando el margen dinámico del conversor no se adecua a la excursión de la señal.

Sea $x_q[n]$ la señal obtenida al cuantificar la secuencia de muestras:

$$x[n] = \text{sen}(2\pi f_o n)$$

Para obtener esta secuencia deberemos emplear la función característica del cuantificador uniforme siguiente:

$$X_q = Q(x) = \begin{cases} \left(E\left[\frac{|x|}{\Delta}\right] + \frac{1}{2}\right) \cdot \Delta \cdot sign(x), & |x| < x_{max} \\ \frac{L-1}{2} \cdot \Delta \cdot sign(x), & |x| \geq x_{max} \end{cases}$$

donde **E[ ]** es la función parte entera (fix() en Matlab), Δ es el tamaño del escalón de cuantificación y **sign()** es la función signo.

La potencia de error de cuantificación **Pq** de una secuencia de **N** muestras se puede estimar calculando:

$$P_q = \frac{1}{N}\sum_{n=0}^{N-1} e_q^2[n] = \frac{1}{N}\sum_{n=0}^{N-1} \left(x_q[n] - x[n]\right)^2$$

La calidad de una señal cuantificada se mide mediante la relación señal a ruido de cuantificación,

$$\left(\frac{S}{N}\right) q = 10\, log_{10} \left(\frac{Px}{Pq}\right) dB$$

donde **Px** es la potencia de la señal sin cuantificar:

$$P_x = \frac{1}{N}\sum_{n=0}^{N-1} x^2[n]$$

Para $f_O = 1/50$, N = 200 muestras, y $x[n] = \text{sen}(2\pi f_o n)$

1.- Programa una función MATLAB que realiza la cuantificación uniforme de una señal, devuelve la señal cuantificada y la relación señal a ruido de cuantificación. La llamada a la función tiene que ser:

function [Xq, SNq]=cuantificacion(X, Xmax, Xmin, b)

donde X es la señal a cuantificar, Xmax y xmin definen el margen margen dinámico del cuantificador (2Xm=Xmax - Xmin), y b es el número de bits del cuantificador.

2.- Representa en la misma gráfica, en 3 subventanas, la señal de entrada X, la señal cuantificada Xq y la función error de cuantificación eq=(Xq - X).

3.- Estudia el funcionamiento de dicho cuantificador observando el aspecto de la sinusoide cuantificada con diferente número de bits: 2, 4, 6, 8, 16. Como varía el error al aumentar los bits? Y la relación señal a ruido de cuantificación?

4.- Comprueba lo que ocurre cuando el margen dinámico del conversor es menor que la excursión de la señal. Observa el valor máximo del error de cuantificación y el valor estimado de la relación señal a ruido.

5.- Comprueba lo que ocurre cuando el margen dinámico del conversor es mucho mayor que la excursión de la señal. Observa el valor máximo del error de cuantificación y el valor estimado de la relación señal a ruido.

Hasta aquí hemos cubierto la primera parte de una conversión A/D normal; no hemos implementado modificaciones de la frecuencia de muestreo y hemos considerado una cuantificación uniforme (ya sabes que existen otros esquemas de cuantificación no uniforme). Pasemos ahora a la reconstrucción.

2.- CONVERSIÓN DIGITAL-ANALÓGICA

2.1.- Reconstrucción ideal

Una vez que se tiene la señal muestreada hay que reconstruirla para lo que se utiliza la función sinc de longitud infinita. Lógicamente no podemos usar algo infinito por lo que nos contentaremos con una versión truncada de esta función. Recordemos que la reconstrucción a aplicar es la siguiente:

$$x(t) = \sum_{n=-\infty}^{\infty} x(n)\, sinc\left(\frac{1}{Ts}\ (t - nTs)\right)$$

El siguiente programa realiza esta reconstrucción:

```
Ts = 0.1;
f0 = 9;
ta = [0:0.0001:1];
xa = cos(2*pi*f0*ta);
nTs = (0:Ts:1).';
xd = cos(2*pi*f0*nTs);
xr = zeros(size(ta));
for m=1:length(nTs)
     xr = xr + xd(m).*sinc((ta-nTs(m))/Ts);
end
plot(ta,xa,'.-g',nTs,xd,'or',ta,xr,'b');
grid;
legend('Original','Muestras','Reconstruida');
xlabel('Tiempo (s)');
ylabel('Amplitud');
axis([0 1 -1.9 1.9]);
```

Usa este programa para ver la reconstrucción de una señal sinusoidal muestreada correctamente e incorrectamente, modificando el valor de Ts con respecto a f0.

Éste es el reconstructor ideal. Una de sus limitaciones es que, de acuerdo con la expresión de interpolación, es necesario conocer TODAS las muestras para ir determinando la señal continua. Dicho de otro modo, para reconstruir la señal entre, por ejemplo, sólo dos muestras, es necesario conocer el valor de todas las muestras.

2.2.- Reconstrucción práctica

¿Qué se puede hacer entonces cuando se trabaja en tiempo real y no se conocen a priori todas las muestras de la señal? El siguiente programa simula el funcionamiento de un sistema de interpolación muy sencillo, se trata de un reconstructor de primer orden con retardo. Razona cómo funciona este sistema a partir del código siguiente:

```
function rec(fs, f) Ts=1/fs; tc=0.001*Ts;
xc=cos(2*pi*f*(0:tc:1));
xd=cos(2*pi*f*(0:Ts:1));
stem(0:Ts:1,xd);
hold on;
xrec=[0 xd];
plot(0:tc:1,xc,'g',Ts:Ts:1,xrec(2:end-1),'k');
grid;
hold off;
```

Prueba para empezar con este ejemplo:

```
>> rec(25,5)
```

Fíjate cómo mejora la calidad de la reconstrucción de este sistema conforme se aumenta fs.

2.3.- Alteración de la frecuencia de muestreo en reconstrucción

Considera 8 señales sinusoidales reales continuas de amplitud 1 y fase inicial de valor 0 rad, cuyas frecuencias son:

f01 = 550 Hz,

f02 = f01·(9/8) Hz,

f03 = f02·(10/9) Hz,

f04 = f03·(16/15) Hz,

f05 = f04·(9/8) Hz,

f06 = f05·(10/9) Hz,

f07 = f06·(9/8) Hz,

f08 = f07·(16/15) Hz

1.- Crea un fichero tipo M para calcular en Matlab 4096 muestras de cada sinusoide suponiendo que se emplea una frecuencia de muestreo fs = 11025 Hz.

2.- Concatena las ocho sinusoides en un único vector x, en orden creciente de pulsación. Escucha la secuencia generada, con 8 y 16 bits de precisión. Para ello utiliza la función sound de Matlab (o bien guarda la secuencia en un fichero WAV con wavwrite):

```
>> sound(x,11025,8)

>> sound(x,11025,16)
```

3.- ¿Notas alguna diferencia de calidad entre la secuencia cuantificada con 8 bits y la cuantificada con 16 bits? ¿Cuál es la razón?

4.- Para apreciar con mayor claridad el efecto del ruido de cuantificación sobre la señal, aplica la función cuantifica.m con Xmax=1, Xmin=-1, y con b=4 bits sobre el vector X,y después escucha el resultado con sound(X,11025).

5.- Por último vamos a observar qué ocurre si modificamos la frecuencia de muestreo en reconstrucción. Para ello, escucha de nuevo la secuencia a 16 bits, pero empleando las frecuencias de muestreo 13, 15 y 18 KHz. ¿Cuál es el efecto observado? ¿A qué se debe?

6.- ¿Qué sucede cuando se emplea una frecuencia en reconstrucción más baja que la frecuencia de muestreo nominal? Compruébalo escuchando la secuencia con 16 bits y una frecuencia de 8 KHz.