**PROCESAMIENTO DIGITAL DE SEÑALES**

***SESIÓN DE LABORATORIO 3 - CURSO ACADÉMICO 2023/2024***

El concepto de muestreo, que es la base de muchos sistemas de procesado de señal y comunicaciones modernos, consiste en obtener una secuencia discreta tomando muestras equi-espaciadas de una señal continua. El teorema de muestreo garantiza que, si la señal continua es suficientemente suave y las muestras se toman lo suficientemente rápido, la señal continua se puede recuperar de manera única y exacta a partir de la secuencia discreta de sus muestras. Este hecho se puede aprovechar para implementar un sistema continuo mediante su equivalente discreto siempre que la señal de entrada sea de banda limitada. En concreto, los objetivos perseguidos con esta práctica son:

* Comprender el concepto de muestreo y las limitaciones impuestas por el teorema de muestreo para poder recuperar la señal continua a partir de sus muestras.
* Analizar el proceso de muestreo y la reconstrucción de la señal, tanto en el dominio del tiempo como en el de la frecuencia, prestando especial atención a las limitaciones introducidas al simularlo en Matlab.
* Observar qué ocurre cuando no se cumplen las condiciones del teorema de muestreo y aparece solapamiento espectral (“aliasing”).

1. **Muestreo de una señal cosenoidal**

El teorema de muestreo garantiza que una señal continua, , limitada en banda (es decir, que para ) se puede reconstruir de manera única y exacta a partir de la secuencia de muestras, , siempre que . En caso contrario aparecerá solapamiento espectral (“aliasing”) y la señal recuperada será diferente de la original.

Diagrama

Descripción generada automáticamente

En este apartado se trata de corroborar estos resultados utilizando señales cosenoidales del tipo:

Como ya se vio en las prácticas anteriores, no es posible trabajar directamente con señales continuas en Matlab, de modo que se simularán utilizando un número suficientemente elevado de muestras (esto es, utilizando una frecuencia de muestreo mucho mayor que la mínima dada por el teorema de muestreo). Nótese además que, dada la relación inversa tiempo-frecuencia, para que una señal continua sea de banda limitada su duración temporal debe de ser infinita (tal y como ocurre con la señal anterior). Obviamente no es posible simular esto en Matlab, de modo que la señal con la que se trabajará verdaderamente será:

siendo un pulso rectangular cuya duración dependerá de la porción de señal simulada. En consecuencia, a la hora de trabajar en el dominio de la frecuencia la transformada de Fourier obtenida será:

Este hecho deberá tenerse en cuenta a la hora de interpretar las transformadas de Fourier obtenidas, tanto en el caso continuo como en el discreto.

Una vez construida la señal continua simulada, se muestreará utilizando el siguiente conversor continuo-discreto (función conv\_cd). Las entradas de la función son el vector xc, que contiene las muestras de la señal continua, el vector t con los instantes de tiempo correspondientes a dichas muestras y el escalar Ts con el periodo de muestreo. Las salidas son otros dos vectores: xd, que contiene la secuencia muestreada, y tsamp, que contiene los instantes de muestreo.

function [xd, tsamp] = conv\_cd(xc, t, Ts)

% "xc": señal continua de entrada (vector de muestras de la señal)

% "t": vector de muestras de tiempo correspondientes a las muestras de la señal "xc"

% "Ts": período de muestreo sobre "xc" para obtener la señal de salida "xd" que a su % vez tendrá como vector de muestras de tiempo "tsamp"

% Cálculo del intervalo de tiempo entre primera y la última muestra de "xc":

T = t(end) - t(1);

% Cálculo del número de muestras si muestreo "xc" cada "Ts" (redondeo al inferior + 1):

Ns = floor(T/Ts) + 1;

% Preparo el vector "xd" con la primera muestra "xc" y el resto de momento ceros:

xd = [xc(1) zeros(1, Ns-1)];

% Preparo el vector de muestras de tiempo de "xd" con la primera muestra de "t" y el

% resto de momento ceros:

tsamp = [t(1) zeros(1, Ns-1)];

% Con este bucle “for” calculo el resto de valores de "xd" y de "tsamp":

for n = 1:Ns-1

% Voy restando a los valores de "t" el valor de "t(1)+nTs", y calculo el valor absoluto

% de la diferencia y el mínimo valor del vector resultante, para acercarme lo más posible

% al valor de la señal en "nTs" (carga la posición del valor más cercano en “ind” y el

% error de la aproximación en “delta”):

[delta, ind] = min(abs(t-t(1)-n\*Ts));

xd(n+1) = xc(ind);

tsamp(n+1) = t(ind);

end

A continuación, se pide que simule y analice el proceso de muestreo, tanto en el dominio del tiempo como de la frecuencia.

**Ejercicio 1**

Simule una señal continua cosenoidal con frecuencia Hz y duración desde hasta segundos, generando un vector con 1001 muestras. Utilice la función conv\_cd para realizar la conversión continua-discreta de dicha señal con un periodo de muestreo segundos. Represente en la misma figura la señal continua en azul (usando el comando plot) y sus muestras discretas en rojo (con el comando stem). Compruebe que la señal está muestreada correctamente, es decir, que las amplitudes de las muestras coinciden con el valor de la señal en los instantes de muestreo.

%% EJERCICIO 1

% Definir parámetros de la señal

f0 = 2; % Frecuencia en Hz

t\_inicio = -1; % Tiempo inicial en segundos

t\_final = 1; % Tiempo final en segundos

num\_muestras = 1001; % Número de muestras

% Crear vector de tiempo y señal continua sinusoidal

t = linspace(t\_inicio, t\_final, num\_muestras);

xc = cos(2\*pi\*f0\*t);

% Parámetros para la conversión continua-discreta

Ts = 0.1; % Periodo de muestreo en segundos

% Realizar la conversión continua-discreta utilizando la función conv\_cd

[xd, tsamp] = conv\_cd(xc, t, Ts);

% Representar la señal continua y sus muestras discretas en la misma figura

figure;

plot(t, xc, 'b', 'LineWidth', 2); % Señal continua en azul

hold on;

stem(tsamp, xd, 'r', 'LineWidth', 1.5); % Muestras discretas en rojo

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Continua y Muestras Discretas');

legend('Señal Continua', 'Muestras Discretas');

grid on;

hold off;

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Como se puede observar en la gráfica, los valores rojos coinciden con la gráfica azul a su paso por el tiempo correspondiente y por tanto podemos decir que se ha muestreado correctamente.

**Ejercicio 2**

Calcule la transformada de Fourier de la señal continua de entrada, xc, y de la secuencia discreta, xd, utilizando las instrucciones vistas en la práctica anterior. Para ello, genere de nuevo la señal , ahora con una duración desde hasta segundos y suficientes muestras (e.g. con ), y evalúe numéricamente entre y usando 1001 puntos. A continuación, muestree la nueva señal con segundos, obteniendo , y evalúe (de acuerdo con lo visto en la Sección 2 de la práctica anterior) entre y usando 1001 puntos. Utilizando el comando subplot dibuje ambas transformadas en la misma figura, pero en sub-figuras diferentes. ¿Coinciden las transformadas obtenidas con los resultados teóricos esperados? ¿A qué cree que se deben las diferencias? Recalcule la transformada de Fourier de la secuencia discreta usando . ¿Se parece la transformada obtenida más al resultado teórico?

%% EJERCICIO 2

delta = 0.001;

t=-2:delta:2;

n=1001;

Ts=0.1;

xc=cos(2\*pi\*f0\*t);

w=linspace(-6\*pi,6\*pi,n);

for k=1:length(w)

XC(k)=sum(xc.\*exp(-1j\*w(k)\*t)\*delta);

end

[xd, tsamp] = conv\_cd(xc, t, Ts);

w1=linspace(-6\*pi,6\*pi,n);

for k=1:length(w1)

XD(k)=sum(xd.\*exp(-j\*w1(k)\*tsamp));

end

figure;

% Subgráfico 1: Transformada de Fourier de xc(t)

subplot(2, 1, 1);

plot(w, XC, 'b', 'LineWidth', 2);

title('Transformada de Fourier de xc(t)');

xlabel('Frecuencia (\omega)');

ylabel('Magnitud');

grid on;

% Subgráfico 2: Transformada de Fourier de xd(t)

subplot(2, 1, 2);

plot(w1, XD, 'r', 'LineWidth', 2);

title('Transformada de Fourier de xd(t)');

xlabel('Frecuencia (\omega)');

ylabel('Magnitud');

grid on;

delta = 0.001;

t=-2:delta:2;

n=1001;

Ts=0.01;

xc=cos(2\*pi\*f0\*t);

w=linspace(-6\*pi,6\*pi,n);

for k=1:length(w)

XC(k)=sum(xc.\*exp(-1j\*w(k)\*t)\*delta);

end

[xd, tsamp] = conv\_cd(xc, t, Ts);

w1=linspace(-6\*pi,6\*pi,n);

for k=1:length(w1)

XD(k)=sum(xd.\*exp(-j\*w1(k)\*tsamp));

end

figure;

% Subgráfico 1: Transformada de Fourier de xc(t)

subplot(2, 1, 1);

plot(w, XC, 'b', 'LineWidth', 2);

title('Transformada de Fourier de xc(t)');

xlabel('Frecuencia (\omega)');

ylabel('Magnitud');

grid on;

% Subgráfico 2: Transformada de Fourier de xd(t)

subplot(2, 1, 2);

plot(w1, XD, 'r', 'LineWidth', 2);

title('Transformada de Fourier de xd(t)');

xlabel('Frecuencia (\omega)');

ylabel('Magnitud');

grid on;

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Ambas graficas casi coinciden, pero la falta de datos en el muestreo provoca discrepancias notables en los extremos del intervalo y en entornos del origen.

Gráfico, Gráfico de líneas, Histograma

Descripción generada automáticamente

Ahora las gráficas son prácticamente idénticas, pues hay más datos, se puede asegurar entonces que este muestreo es mucho mejor que el primero.

1. **Reconstrucción de una señal cosenoidal a partir de sus muestras**

La recuperación de la señal continua, simulada a partir de la secuencia discreta de sus muestras, se lleva a cabo simplemente mediante la aplicación la fórmula de interpolación vista en teoría:

Para conseguir una reconstrucción perfecta sería necesario utilizar el **filtro de reconstrucción ideal**, cuya respuesta al impulso es:

El valor de esta respuesta al impulso para un conjunto de instantes de tiempo se puede obtener fácilmente mediante la siguiente función de Matlab, en la que se tiene en cuenta el caso especial de , en el que .

function hi = filtro\_ideal(t, Ts)

% FILTRO DE RECONSTRUCCIÓN IDEAL:

% "t": vector de muestras de tiempo.

% "Ts": período de muestreo de la señal original.

% "hi": respuesta al impulso del filtro de reconstrucción ideal.

hi = Ts\*sin(pi\*t/Ts)./(pi\*t);

ind = find(t==0);

hi(ind) = 1;

Desafortunadamente, el filtro ideal es irrealizable en la práctica de manera exacta debido a su no causalidad y a la duración infinita de su respuesta al impulso. En consecuencia, en la práctica a menudo se utilizan filtros reconstructores más sencillos, tales como el **retenedor de orden cero**, cuya respuesta al impulso es

y cuya implementación en Matlab se muestra a continuación.

function h0 = filtro\_orden0(t, Ts)

% RETENEDOR DE ORDEN CERO:

% "t": vector de muestras de tiempo.

% "Ts": período de muestreo de la señal original.

% "h0": respuesta al impulso del filtro retenedor de orden cero.

h0 = zeros(size(t));

epsilon = 1e-10;

ind = find((t>=0) & (t<(Ts-epsilon)));

h0(ind) = 1;

Otra alternativa es utilizar un **interpolador lineal**, cuya respuesta al impulso es

Su implementación en Matlab se muestra a continuación.

function h1 = filtro\_lineal(t, Ts)

% INTERPOLADOR LINEAL:

% "t": vector de muestras de tiempo.

% "Ts": perÌodo de muestreo de la seÒal original.

% "h1": respuesta al impulso del interpolador lineal.

h1 = zeros(size(t));

ind = find(abs(t)<Ts);

h1(ind) = (Ts-abs(t(ind)))/Ts;

En estos casos, en general la señal reconstruida no va a coincidir exactamente con la original, pero el error de reconstrucción será pequeño si la frecuencia de muestreo es suficientemente grande en comparación con la máxima frecuencia de la señal.

En este apartado se trata de analizar estas tres alternativas de reconstrucción de la señal continua cuando esta se encuentra correctamente muestreada.

**Ejercicio 3**

Utilice el filtro reconstructor ideal para recuperar la señal continúa generada en el anterior ejercicio (i.e. señal cosenoidal con Hz, duración desde hasta segundos, segundos) a partir de las dos secuencias muestreadas con y segundos. Represente en una misma figura, pero en sub-figuras diferentes, para ambos casos: la señal continua original en azul (con el comando plot), la secuencia muestreada en rojo (con el comando stem) y la señal reconstruida en verde (con el comando plot). Analice el error de reconstrucción obtenido en ambos casos (puede hacer un zoom de las figuras para apreciarlo mejor) y extraiga conclusiones.

%% EJERCICIO 3

t=-2:0.001:2;

Ts=0.1;

for k=1:length(t)

X(k)=sum(xd.\*filtro\_ideal(t(k)-tsamp,Ts));

end

figure;

plot(t,X);

hold on

plot(t,xc);

hold on

stem(tsamp,xd);

• En el primer caso (intervalo de muestreo de 10 ms), la reconstrucción con el filtro reconstructor ideal es bastante precisa, y el error de reconstrucción es bajo. Esto demuestra que el filtro ideal funciona bien cuando el muestreo es suficientemente fino.

• En el segundo caso (intervalo de muestreo de 100 ms), el filtro reconstructor ideal no puede recuperar la señal con precisión, y el error de reconstrucción es más alto. Esto es esperado, ya que el intervalo de muestreo es más grande en relación con la frecuencia de la señal.

• En general, la elección del filtro de reconstrucción debe adaptarse al intervalo de muestreo y las características de la señal para obtener una reconstrucción precisa. El filtro reconstructor ideal es efectivo cuando el muestreo es adecuadamente fino, pero en la práctica, es posible que se prefieran filtros más simples y prácticos cuando el muestreo es más espaciado.

Gráfico, Histograma

Descripción generada automáticamente

**Ejercicio 4**

Repita el ejercicio anterior utilizando el reconstructor de orden cero y el interpolador lineal. Compare los resultados obtenidos con los del ejercicio anterior y extraiga conclusiones respecto a los casos en los que resulta apropiado usar filtros reconstructores más sencillos que el ideal.

%% EJERCICIO 4

for k=1:length(t)

X1(k)=sum(xd.\*filtro\_orden0(t(k)-tsamp,Ts));

end

figure;

plot(t,X1);

hold on

plot(t,xc);

hold on

stem(tsamp,xd);

for k=1:length(t)

X2(k)=sum(xd.\*filtro\_lineal(t(k)-tsamp,Ts));

end

figure;

plot(t,X2);

hold on

plot(t,xc);

hold on

stem(tsamp,xd);

Imagen que contiene Gráfico

Descripción generada automáticamente

Gráfico, Histograma

Descripción generada automáticamente

4 Conclusiones:

• Reconstructor de Orden Cero vs. Filtro Reconstructor Ideal:

• El reconstructor de orden cero es una aproximación más simple que el filtro reconstructor ideal.

• Para el intervalo de muestreo de 10 ms, el MSE del reconstructor de orden cero es mayor que el del filtro ideal, lo que indica una reconstrucción menos precisa.

• Para el intervalo de muestreo de 100 ms, el MSE del reconstructor de orden cero es aún mayor, lo que demuestra que esta aproximación es menos efectiva cuando el muestreo es más espaciado.

• Interpolador Lineal vs. Filtro Reconstructor Ideal:

• El interpolador lineal también es una aproximación más simple que el filtro reconstructor ideal.

• Para el intervalo de muestreo de 10 ms, el MSE del interpolador lineal es mayor que el del filtro ideal, mostrando una reconstrucción menos precisa.

• Para el intervalo de muestreo de 100 ms, el MSE del interpolador lineal es aún mayor que en el caso de 10 ms, lo que indica un rendimiento deficiente en la reconstrucción.

Conclusiones Generales:

• Los filtros reconstructores más simples, como el reconstructor de orden cero y el interpolador lineal, son menos precisos que el filtro reconstructor ideal cuando el muestreo es insuficiente.

• El filtro reconstructor ideal es más apropiado cuando el intervalo de muestreo es pequeño en relación con la frecuencia de la señal.

• Para intervalos de muestreo más grandes, es probable que los filtros más simples sean aceptables, pero es importante tener en cuenta que la calidad de la reconstrucción será menor y el error será mayor.

1. **Efectos del muestreo incorrecto: “aliasing”**

Cuando la señal continua no está bien muestreada (es decir, se muestrea a una frecuencia inferior a la mínima dada por el teorema de muestreo) se producirá solapamiento espectral (“aliasing”) y la señal reconstruida no coincidirá con la señal original. En este apartado se exploran los efectos del “aliasing” trabajando con señales cosenoidales.

**Ejercicio 5**

Genere una señal cosenoidal como la de los ejercicios anteriores (ahora con Hz, duración desde hasta segundos, segundos) y muestréela con segundos. Recupere la señal utilizando el filtro reconstructor ideal y dibuje en una misma figura la señal continua original, la secuencia muestreada y la señal reconstruida (tal y como hizo en el ejercicio 3). ¿Qué forma tiene la señal recuperada? ¿Cuál es su relación con la señal original? Para apreciar mejor esta última parte, calcular la transformada de Fourier de la señal original y de la señal reconstruida, representando ambas en la misma figura.

%% EJERCICIO 5

% Definir parámetros de la señal

f0 = 5; % Frecuencia en Hz

t\_inicio = -2; % Tiempo inicial en segundos

t\_final = 2; % Tiempo final en segundos

num\_muestras = 1001; % Número de muestras

% Crear vector de tiempo y señal continua sinusoidal

t = linspace(t\_inicio, t\_final, num\_muestras);

xc = cos(2\*pi\*f0\*t);

% Parámetros para la conversión continua-discreta

Ts = 0.25; % Periodo de muestreo en segundos

% Realizar la conversión continua-discreta utilizando la función conv\_cd

[xd, tsamp] = conv\_cd(xc, t, Ts);

for k=1:length(t)

X3(k)=sum(xd.\*filtro\_ideal(t(k)-tsamp,Ts));

end

figure;

plot(t,X3);

hold on

plot(t,xc);

hold on

stem(tsamp,xd);

xc=cos(2\*pi\*f0\*t);

[xd, tsamp] = conv\_cd(xc, t, Ts);

w1=linspace(-6\*pi,6\*pi,1001);

for k=1:length(w1)

XD(k)=sum(xd.\*exp(-j\*w1(k)\*tsamp));

end

for k=1:length(w)

XD3(k)=sum(xc.\*exp(-1j\*w(k)\*t)\*delta);

end

figure;

% Subgráfico 1: Transformada de Fourier de xc(t)

subplot(2, 1, 1);

plot(w1, XD, 'b', 'LineWidth', 2);

title('Transformada original');

xlabel('Frecuencia (\omega)');

ylabel('Magnitud');

grid on;

% Subgráfico 2: Transformada de Fourier de xd(t)

subplot(2, 1, 2);

plot(w1, XD3, 'r', 'LineWidth', 2);

title('Transformada reconstruida');

xlabel('Frecuencia (\omega)');

ylabel('Magnitud');

grid on;

Gráfico

Descripción generada automáticamente Gráfico, Histograma

Descripción generada automáticamente

• La forma de la señal reconstruida será una serie de impulsos. Esta es una característica del filtro reconstructor ideal, que introduce impulsos en la señal reconstruida en los instantes de tiempo de muestreo.

• La señal reconstruida tiene una relación directa con la señal original, ya que intenta recuperar la forma original de la señal cosenoidal a partir de las muestras discretas.

• Al calcular y representar las transformadas de Fourier, podrás observar cómo la señal original tiene un pico en 5 Hz (frecuencia de la señal cosenoidal), y la señal reconstruida mostrará un conjunto de picos en las frecuencias múltiples de 5 Hz debido a la sincronización de los impulsos del filtro ideal.