8pts

Sistemas LTI

Linealidad: debe cumplir con el ppio de superposición.
Invarianza en el tiempo: cualquier corrimiento en tiempo en la entrada produce los mismos resultados en la salida, es decir, la salida solo depende de que se ingresa a la entrada y no depende del momento en el que se aplicó la misma.

- y(t) = x(t) ∗ cos (t)

  Linealidad: a*x1(t)+b*x2(t) → ¿L? → a*y1(t)+b*y2(t)

  x(t) * cos(t) → (a*x1(t)+b*x2(t))* cos(t) → a*x1(t)*cos(t) + b*x2(t)*cos(t)

  → a*y1(t)+b*y2(t) **Es lineal** ✓

  Invarianza en el tiempo: x(t-to) → ¿TI? → y(t-to)

  x(t) * cos(t) → x(t-to) * cos**(t)** es distinto a y(t-to)=x(t-to)* cos**(t-to)**

  No es invariante en el tiempo → No es LTI ✓

y(t) = cos(x(t))

Linealidad: a*x1(t)+b*x2(t) → ¿L? → a*y1(t)+b*y2(t)

cos(a*x1(t)+b*x2(t)) es distinto de a*cos(x1(t))+b*cos(x2(t))

por lo tanto **no es lineal → no es LIT** ✓

y(t) = e(x(t))

Linealidad: a*x1(t)+b*x2(t) → ¿L? → a*y1(t)+b*y2(t)

e(a*x1(t)+b*x2(t)) es distinto de a*e(x1(t))+b*e(x2(t))

por lo tanto **no es lineal → no es LIT** ✓

y(t) = ½ x(t)

Linealidad: a*x1(t)+b*x2(t) → ¿L? → a*y1(t)+b*y2(t)
½ * (a*x1(t)+b*x2(t)) → ½ * a*x1(t) + ½ *b*x2(t) = a*y1(t)+b*y2(t)

Es lineal ✓

Invarianza en el tiempo: x(t-to) → ¿TI? → y(t-to)

½ * x(t-to) = y(t-to)
por lo tanto **es invariante en el tiempo → es LIT**

✓

Ruido de cuantización

1. Calcule la relación señal a ruido de cuantización teórica máxima de un sistema con un ADC de:

- 24 bits
  SNR = 1,76 + 6,02 * N → 1,76 + 6,02 * 24
  → **146.24dB > (S/R = 100.000.000.000.000**)

- 16 bits
  SNR = 1,76 + 6,02 * N → 1,76 + 6,02 * 16
  → **98.08dB > (S/R = 1.000.000.000)**

- 8 bits
  SNR = 1,76 + 6,02 * N → 1,76 + 6,02 * 8
  → **49.92dB aprox 50dB → (S/R = 100.000)**

- 2 bits
  SNR = 1,76 + 6,02 * N → 1,76 + 6,02 * 2
  → **13,8dB → (S/R = 24)**

2. Dado un sistema con un ADC de 10 bits, que técnica le permitiría aumentar la SNR? En qué consiste?

Teniendo en cuenta las condiciones indicadas en la teoría como modelo estadístico del sistema se procede de la siguiente manera.
Como la potencia de ruido de cuantización es constante para una determinada cantidad de bits del conversor.
La densidad espectral de potencia (DEP) determina còmo se distribuye la potencia de ruido a lo largo del ancho de banda. Al sobremuestrear la señal de entrada se amplía el ancho de banda de entrada, esto provoca que la potencia de ruido se distribuya uniformemente a lo largo del nuevo ancho de banda, reduciendo el valor máximo de la DEP.
Por lo cual al duplicar la frecuencia de muestreo se duplica el ancho de banda produciendo que la magnitud del ruido se reduzca a la mitad (-3dB).

SNR = Psignal / Pnoise

Al duplicar la frecuencia de muestreo se reduce a la mitad la potencia de ruido

SNR = Psignal / (Pnoise/2) → SNR = 2 (Psignal / Pnoise)

→ En dB → SNR = SNRsinSobremuestreo + 3dB

SNR10bits = 61.96dB (S/R = 1.570.000)

Sobremuestreo x2 → SNR10bitsX2 = **64.96dB** → **(S/R = 3.000.000)**
Sobremuestreo x4 → SNR10bitsX4 = **67.96dB** → **(S/R = 6.250.000)**
Sobremuestreo x8 → SNR10bitsX8 = **70.96dB** → **(S/R = 12.400.000)**
Sobremuestreo x16 → SNR10bitsX16 = **73.96dB** → **(S/R = 24.800.000)**

Filtro antialias y reconstrucción

1. Calcular el filtro antialias que utilizará para su práctica y/o trabajo final y justifique su decisión.

Si se desea procesar señales de audio comúnmente se utiliza como frecuencia de corte 20KHz. Por lo cual para diseñar un filtro antialiasing de primer orden debemos obtener los valores R C mediante una aproximación de prueba y error para ajustar los valores de estos componentes en valores fáciles de conseguir.

fcorte = 20KHz y suponemos R = 2.2Kohm

C = 1 / (20KHz * 2kohm * 2 * pi) = 3.61 nF

los valores comerciales de capacitores próximos al resultado obtenido son 3.3nF o 4.7nF por lo cual nuestra frecuencia de corte sería para cada caso:

fcorte@3.3nF = 21.92KHz
fcorte@4.7nF = 15.39KHz

Como el valor de 4.7nF reduce notablemente el ancho de banda deseable de 20KHz para dejar pasar las frecuencias de audio. Se utiliza el capacitor de 3.3nF junto con una resistencia de 2200ohms para poder obtener una frecuencia de corte de 21.92KHz.
Como la frecuencia de corte es de 21.92KHz es preferible que la frecuencia de muestreo sea de del orden de los 50KHz o mayor ya que el rechazo de las altas frecuencias con un filtro de primer orden no es abrupto y es preferible tener un rango de seguridad.

Generación y simulación

1. Genere un módulo o paquete con al menos las siguientes funciones
   - Senoidal (fs[Hz], f0[Hz], amp[0 a 1], muestras),fase [radianes]
   - Cuadrada (fs[Hz], f0[Hz], amp[0 a 1], muestras)
   - Triangular(fs[Hz], f0[Hz], amp[0 a 1], muestras)

   Link con la sinusoidal cuadrada y diente de sierra.
2. Realice los siguientes experimentos
   - fs = 1000
   - N = 1000
   - fase = 0
   - amp = 1

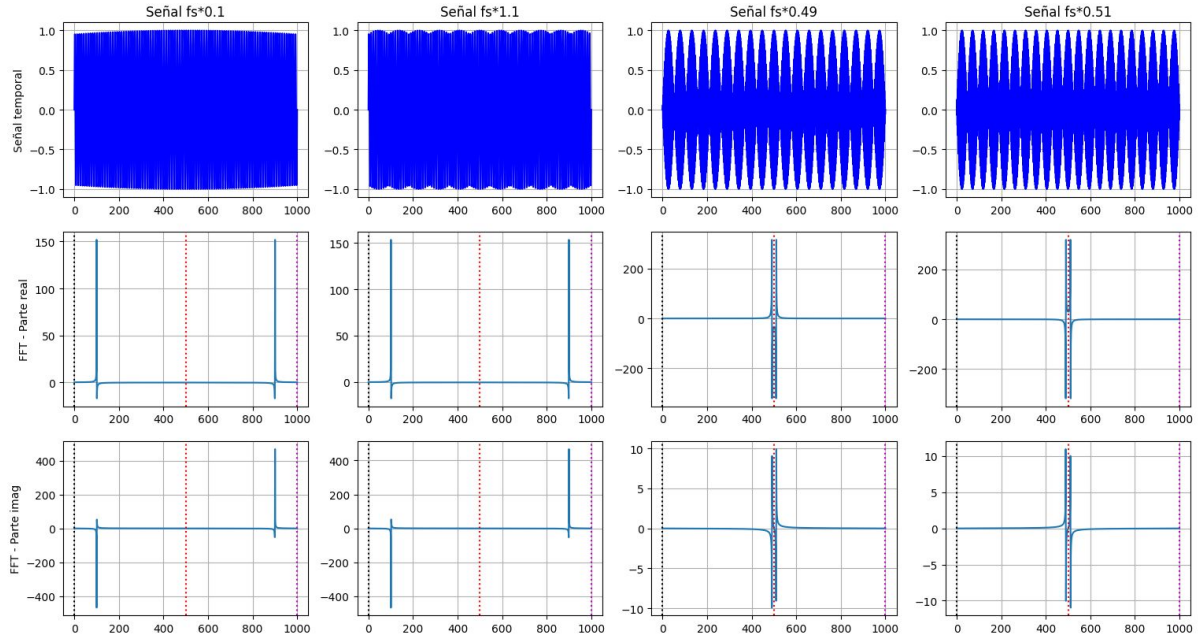
2.1 f0 = 0.1*fs y 1.1*fs Cómo podría diferenciar las senoidales?

Es imposible determinar una señal de la otra porque se superponen recíprocamente.
0.1*fs genera un pulso replicado en 1.1*fs
por otro lado,
1.1*fs genera un pulso replicado en 0.1*fs

2.2 f0 = 0.49*fs y 0.51*fs Como es la frecuencia y la fase entre ambas?

Al producirse el efecto de aliasing las señales son representadas en las mismas frecuencias. Pero en este caso graficando la parte real e imaginaria por separado se obtiene que, la parte real de la señal de 0.49fs es la misma señal que la de 0.51fs pero espejada con respecto al eje de las absisas. Lo mismo sucede con respecto a la representación en frecuencia de la parte imaginaria.

Link al programa



Esta muy bien, pero podrias mejorar los graficos mostrando soloa algunos ciclos de las seniales para que se aprecie mejor. Incluso superponer las dos

Adquisición y reconstrucción con la CIAA

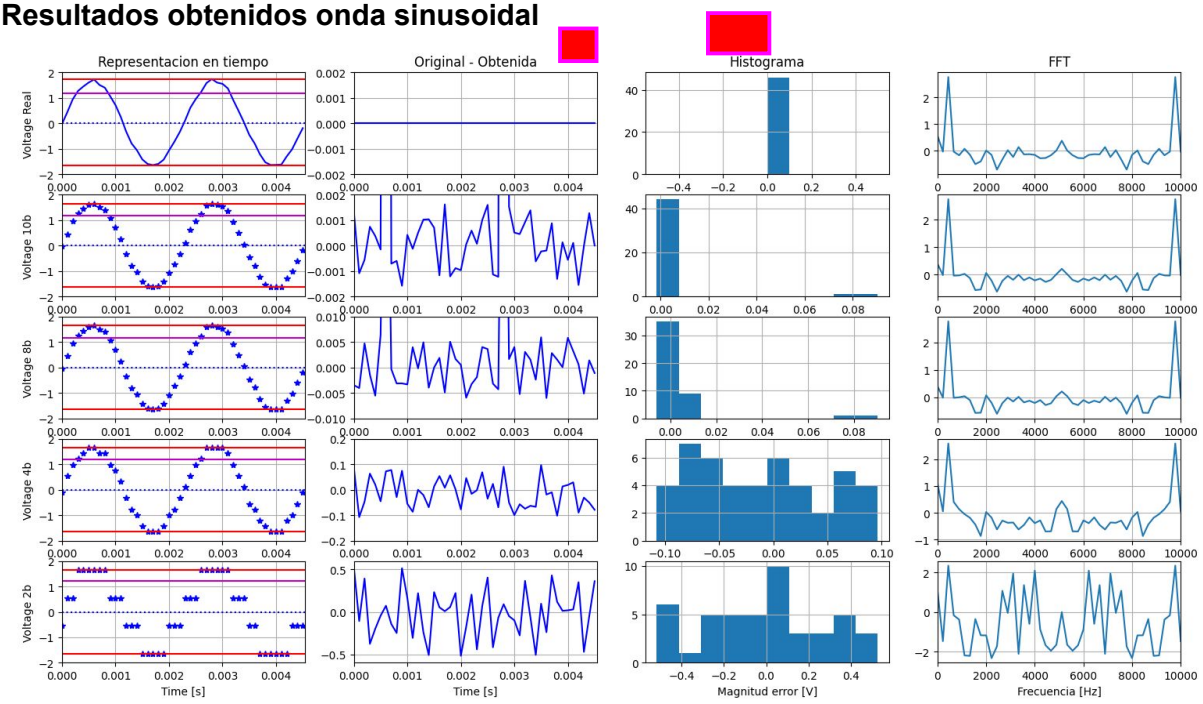
1. Genere con un tono de LA-440. Digitalice con 10, 8, 4 y 2 bits con el ADC, envíe los datos a la PC, grafique y comente los resultados

- Señal original con su máximo, mínimo y RMS
- Señal adquirida con su máximo, mínimo y RMS
- Señal error = Original-Adquirida
- Histograma del error

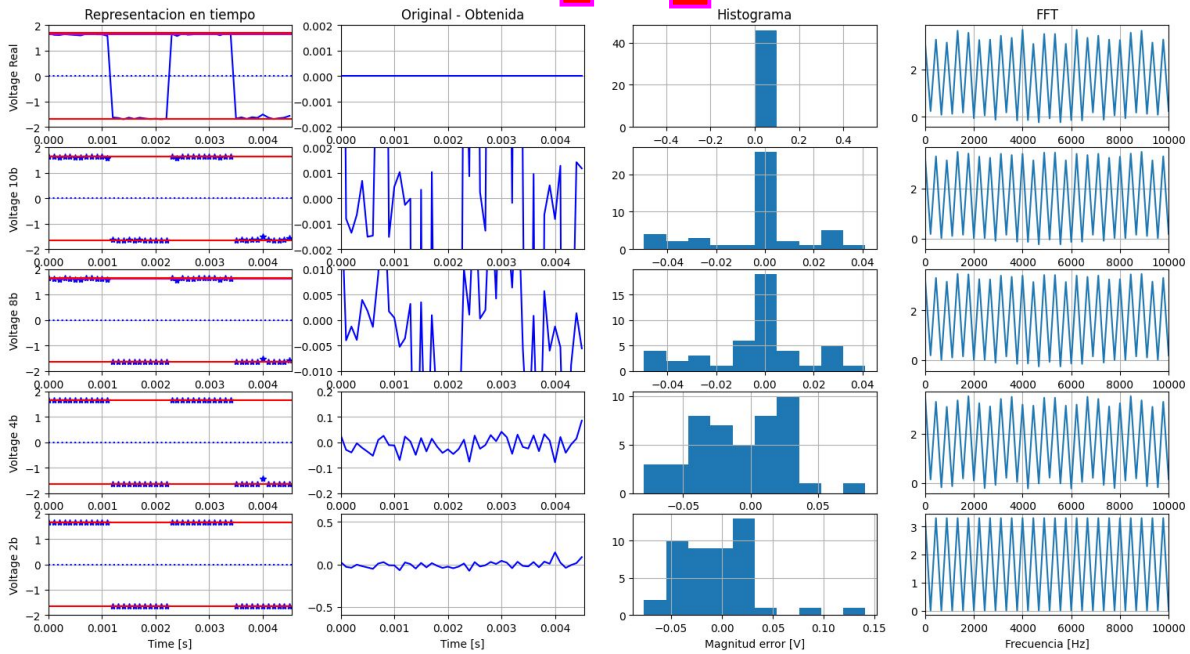
2. Realice el mismo experimento con una cuadrada y una triangular

[Link al programa.]

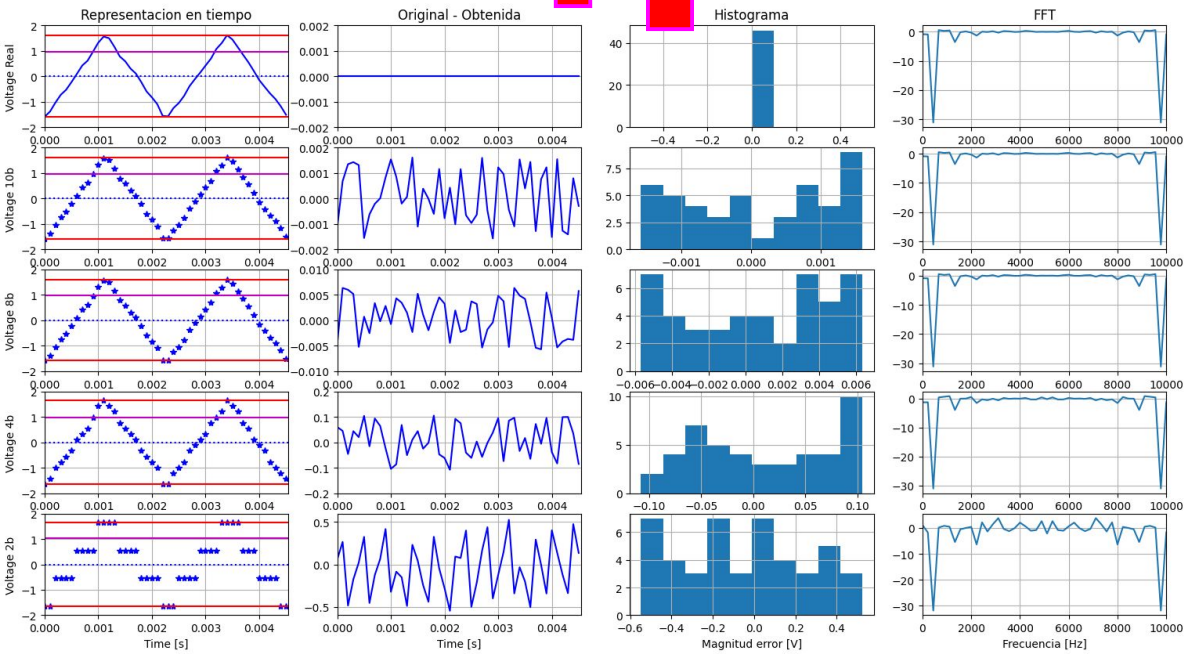
Resultados obtenidos onda sinusoidal



Resultados obtenidos onda cuadrada



Resultados obtenidos onda triangular



Mire un poco el codigo que usastes para generar estos graficos, y me hubiese gustado que lo intentaras hacer con la CIAA para darle mas realismo
Pero en cualquier caso tanto el error como el histograma del error te quedaron inconsistentes. Creo que trabajar con flotantes para este punto, aun en python genera confusion y no hace falta. Podrias haber usado directamente enteros como si fuera el ADC y enmascarar o shiftear.
Tambien es buena idea superponer la señal generada con la sampleada, en tu caso la simulada porque ahi se ve bien el error y salta cualquier inconsistencia.
Intenta comparar los resultados con los presentados en clase a ver si detectas las diferencias

Sistema de números

1. Explique brevemente algunas de las diferencias entre la representación flotante de simple precisión (32b) y el sistema de punto fijo Qn.m

Punto flotante de simple precisión:
Como su nombre lo indica el punto decimal no se encuentra fijo en la representación binaria, esto sucede porque el número es representado **por un bit de signo, 8 bits para el exponente y 23 bits para la mantisa.** Es denominado simple precisión ya que se utilizan 32 bits para su representación, existe también una representación de doble precisión que utiliza 64bits, comúnmente denominados float y double respectivamente.
La representación se basa en la IEEE754
La mantisa comienza por 1.0 que no es alojado y solo se aloja la parte fraccionaria.
El exponente es representado en una representación de complemento a 2.

Punto fijo Qn.m:
Es una representación implícita de números binarios para poder operar números con parte fraccionaria en una plataforma que solo trabaja con número enteros comúnmente.
N determina la cantidad de bits de la parte entera (incluyendo el signo)
M determina la cantidad de bits de la parte fraccionaria.
Los números son representados en complemento a dos *(CA2)*.
El programador es el responsable de operar este formado de manera adecuada para evitar perder precisión o incluso la magnitud al efectuar por ejemplo operaciones aritméticas.

2. Escriba los bits de los siguientes números decimales (o el más cercano) en float, Q1.15, Q2.14

Número	Q1.15	Q2.14
0.5	1 >>1 → 0.1 **0 , 1000 0000 0000 000**	1 >>1 → 0.1 **00 , 1000 0000 0000 00**
-0.5	Complemento a 2 de 0.5: 0 , 1000 0000 0000 000 CA1 1 , 0111 1111 1111 111 Sumando 1 **1 , 1000 0000 0000 000**	Complemento a 2 de 0.5: 00 , 1000 0000 0000 00 CA1 11 , 0111 1111 1111 11 Sumando 1 **11 , 1000 0000 0000 00**
-1.25	1.25 → 5 >> 2 → 1,01 0 , 1111 1111 1111 111 **(no se puede representar el número 1 o mayores)** -1.25 CA2 de 1.25 1 , 0000 0000 0000 000 **1 , 0000 0000 0000 001**	1.25 → 5 >> 2 → 1,01 01 , 0100 0000 0000 00 -1.25 CA2 de 1.25 10 , 1011 1111 1111 11 **10 , 1100 0000 0000 00**

0.001	**1 >>10** **0 , 0000 0000 0100 000**	**1 >>10** **00 , 0000 0000 0100 00**
-2.001	**(no se puede representar el número 1 o mayores)** **1 , 0000 0000 0000 001**	**(no se puede representar el número 2 o mayores)** **10 , 0000 0000 0000 01**
204000000	**(no se puede representar el número 1 o mayores)** **0 , 1111 1111 1111 111**	**(no se puede representar el número 2 o mayores)** **01 , 1111 1111 1111 11**

Resumen de resultados

Número	float	Q1.15	Q2.14
0.5	0 01111110 00000000000000000000000	0 1000 0000 0000 000	00 1000 0000 0000 00
-0.5	1 01111110 00000000000000000000000	1 1000 0000 0000 000	11 1000 0000 0000 00
-1.25	1 01111111 01000000000000000000000	1 0000 0000 0000 001	10 1100 0000 0000 00
0.001	0 01110101 00000110001001001101111 **(no existe una representación exacta)**	0 0000 0000 0100 000 **(no existe una representación exacta)**	00 000 0000 0100 00 **(no existe una representación exacta)**
-2.001	1 10000000 00000000001000001100010 **(no existe una representación exacta)**	1 0000 0000 0000 001 **(satura)**	10 0000 0000 0000 01 **(satura)**
204000000	0 10011010 10000101000110010110000	0 , 1111 1111 1111 111 **(satura)**	01 1111 1111 1111 11 **(satura)**

seria 0 en realidad, indicando -1 en q15 o -2 en Q14 el numero mas negativo posible

Esta muy bien, solo presta atencion al 1 en la posicion LSB que le agregastes cuando no se podia representar numeros muy negativos. El numero mas negativo posible en CA2 es el MSB en 1 y todo el resto en cero. SI le agregas un 1 en el LSB lo haces 'menos' negativo.. aunque muy poquito
O sea en Q15 podes ir de aproc 0.999 a -1 exacto y en Q14 de aprox 1.999 a -2 exacto
Igual puedo apreciar que lo entendistes muy bien, solo para notar ese detalle.