UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Control 3

Integrante:	Franco Tapia Cabañas
RUT:	19.234.165-5
Curso:	Redes de computadores
Profesor:	Carlos González
Ayudante:	Fernanda Muñoz

7 de Agosto de 2020

Tabla de contenidos

1. C3-P1: Teorema del Muestreo

1. Verifique empíricamente que si se cumple el teorema del muestreo entonces es posible recuperar la señal original a partir de la señal muestreada. Para ello:

   a) Cree un programa en python que sea capaz de generar la siguiente señal de ejemplo: $f(t) = cos(2\pi \cdot 3t) + sin(2\pi \cdot 2t)$ para t en el intervalo [0, 1]

   Solución:

   Se define la función signal, quien se encarga de retornar la función original presentada en el enunciado. Posteriormente, se grafica la función obtenida con un rango de tiempo de 0 a 1 segundo, utilizando un intervalo de $\frac{1}{500}$. A continuación se muestra el código elaborado.

```
1        #a. Generar se al de ejemplo f(t):
2        def signal(t):
3            return cos(2pi3t)+sin(2pi2t)
4        # Graficando funcion f(t)
5        t = np.arange(0, 1, 1/500)
6        f = signal(t)
7        plt.plot(t, f)
8        plt.grid(True)
9        plt.title("f(t) = cos(2pi3t)+sin(2pi2t)")
10       plt.xlabel("tiempo (s)")
11       plt.show()
```

   En base al segmento de código presentado, se obtiene la siguiente función.

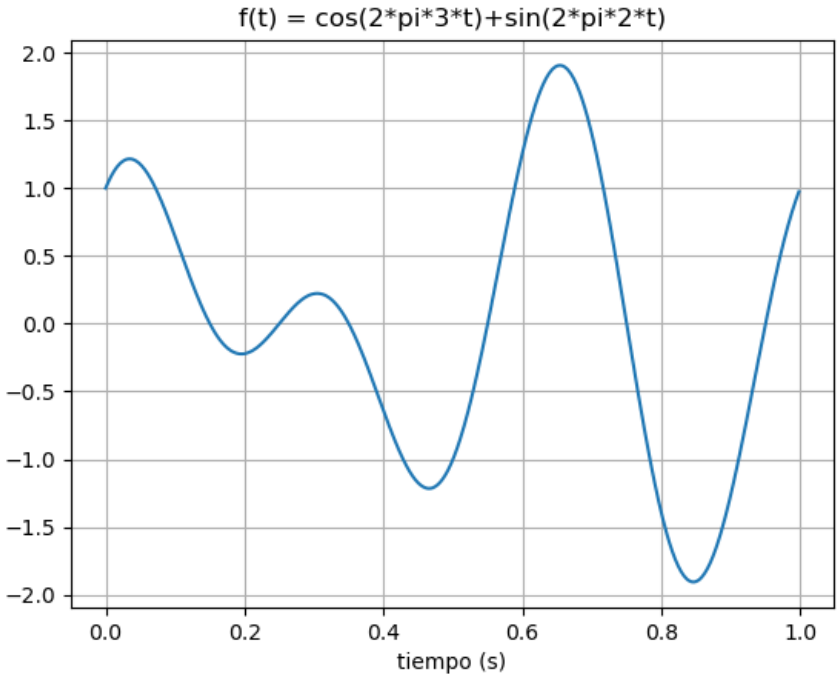


Figura 1: Señal original representada en Python.

b) Determine la frecuencia de muestreo correcta y genere una señal muestreada $f_s[n]$

Solución: La frecuencia de muestreo se puede obtener directamente de la función original. En este caso corresponde a 6 (Hz) o más, debido a que la frecuencia máxima de la función es 3 (valor que acompaña a 2*pi*t en coseno). En el siguiente segmento de código, es posible ver como se elabora la señal muestreada y graficada, posteriormente.

```
1         #b. Frecuencia de muestreo correcta y se al muestreada fs
2         # Funcion de muestreo
3         def sinc(t, compress=1):
4             return sin(compresspit)/(compresspit)
5
6         # Graficando funcion de muestreo
7         Fs = 6      #Frecuencia de muestreo = 6
8                     #(para error < 10%, Fs = 7)
9         tn = np.arange(0, 1.1, 1/Fs)
10        fs = signal(tn)
```

```
11          plt.figure()
12          plt.plot(t, f, '--')
13          plt.stem(tn, fs, 'r', markerfmt='C3o', use_line_collection
                =True)
14          plt.grid(True)
15          plt.title("fs(n) = cos(2pi3n)+sin(2pi2n)")
16          plt.xlabel("tiempo (s)")
17          plt.show()
```

Con respecto a este código, se muestran los puntos representativos a la frecuencia de muestreo y además, una función punteada que representa a la función original.

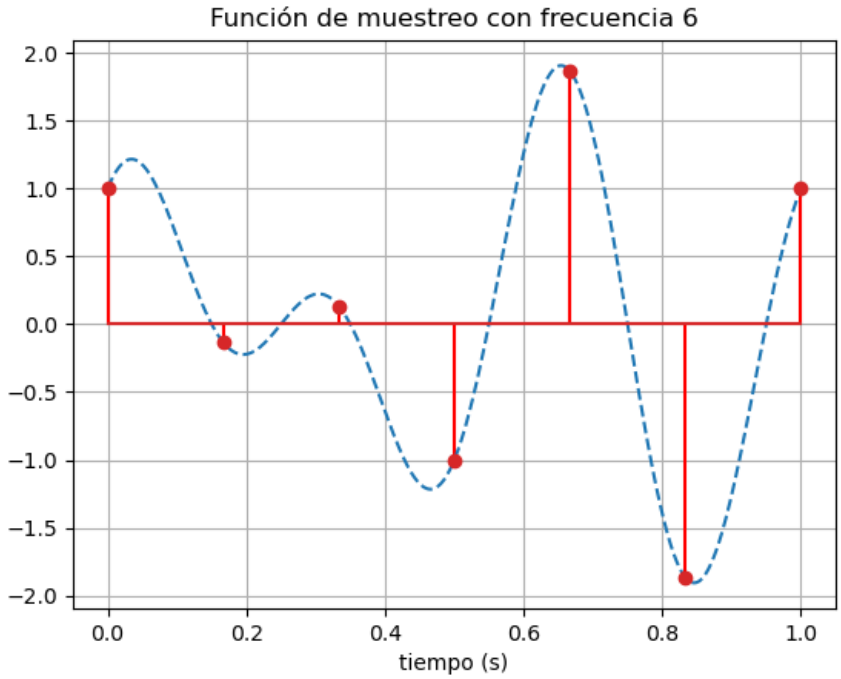


Figura 2: Señal muestreada con una frecuencia de 6.

c) Reconstruya la señal f(t) a partir de las muestras fs[n] usando la función de muestreo sinc(t)

Solución:

Para reconstruir la función original, se elabora el siguiente código implementado en Python y se grafica gracias a la librería matplotlib.

```
        #c. Reconstruyendo f(t) a partir de fs
        tf = np.arange(-5, 5, 1/500)
        sinc_n = -1sinc((tf-2))

        plt.figure()
        for i,s in zip(tn,fs):
          sinc_s = ssinc((tf-i),Fs)

        plt.plot(tf,sinc_n)
        plt.stem(tn,fs, 'r', markerfmt='C3o', use_line_collection=
            True)
        plt.xlim([-1, 2])
        plt.grid(True)
        plt.title("fisinc(t-ti)")
        plt.xlabel("time (s)")
        plt.show()
```

En el gráfico mostrado a continuación, se representa con puntos rojos la función reconstruida.

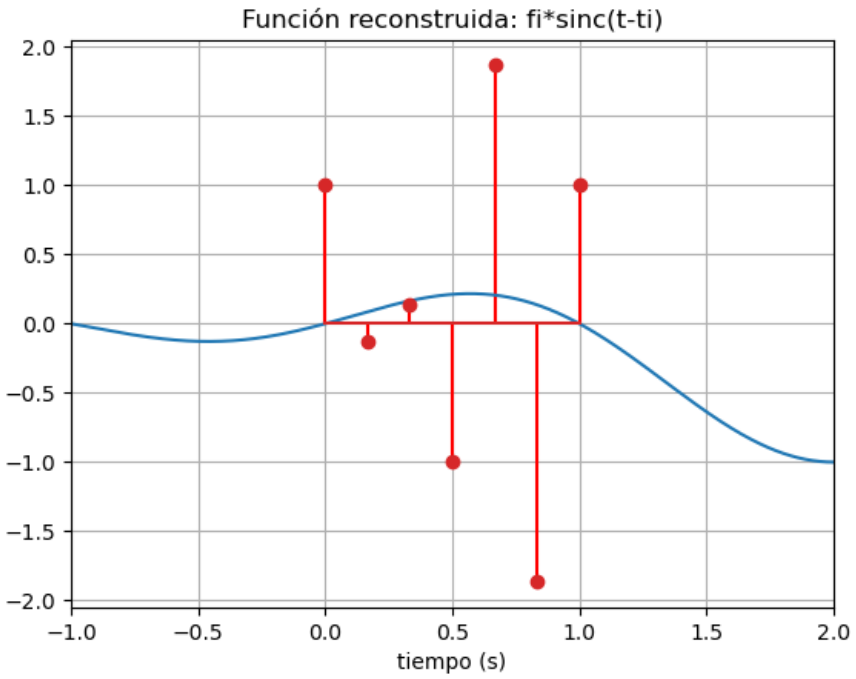


Figura 3: Función reconstruida con frecuencia 6

d) Calcule y compare el valor RMS de la señal original y la señal reconstruida. ¿Cuál es el porcentaje de error? Determine la frecuencia de muestreo tal que la diferencia entre los RMS sea menor al 10 %. ¿Cuáles podrían ser las fuentes de la diferencia observada?

Solución:

El error obtenido, con una frecuencia de muestreo igual a 6, es de 0.304. Es decir, existe un error del 30,4 %, lo que claramente es mayor a 10 %. Para poder disminuir el error, se incrementa el valor de la frecuencia de muestreo a 7, teniendo un error de un 9.6 %. Esto se debe a que la teoría no se cumple en la realidad, de hecho, en la práctica existen limitaciones en los circuitos que impiden obtener una correcta representación de la señal inicial. Es por ello, que se debe aumentar la frecuencia de muestro de 6 a 7 en este caso.

En el siguiente código, se muestra el segmento para calcular y graficar el error de la señal original respecto a la reconstruida. En este caso se obtiene el porcentaje de error punto a punto, según la señal original y la reconstruida, antes mencionada.

```
        # d. Comparando se al original con se al reconstruida
        sincs = []
        tf = np.arange(1e-10,1,1/500)
        plt.figure()

        for s,n in zip(fs,tn):
            sinc_n = ssinc(Fs*(tf-n))
            sincs.append(sinc_n)
        sincs = np.array(sincs)
        sincs = np.sum(sincs, axis=0)
        plt.plot(t, f)
        plt.plot(tf, sincs, 'r--')
        plt.xlim([0, 1])
        plt.grid(True)
        plt.title("SUM")
        plt.xlabel("time (s)")
        plt.show()

        # Obteniendo porcentaje de error
        rms = 0
        n = len(f)
        for i in range(n):
            rms = rms + (sincs[i] - f[i])2
        rms = (rms/n)(1/2)
        print(rms)
```

En la **Figura 4**, se puede ver una mayor diferencia entre la señal original (linea azul) y la señal reconstruida, con respecto a la presentada en la **Figura 5**. Esto, se debe que a en la primera figura mencionada se utiliza la frecuencia de muestre 6, mientras que en la segunda se utiliza una frecuencia de 7.

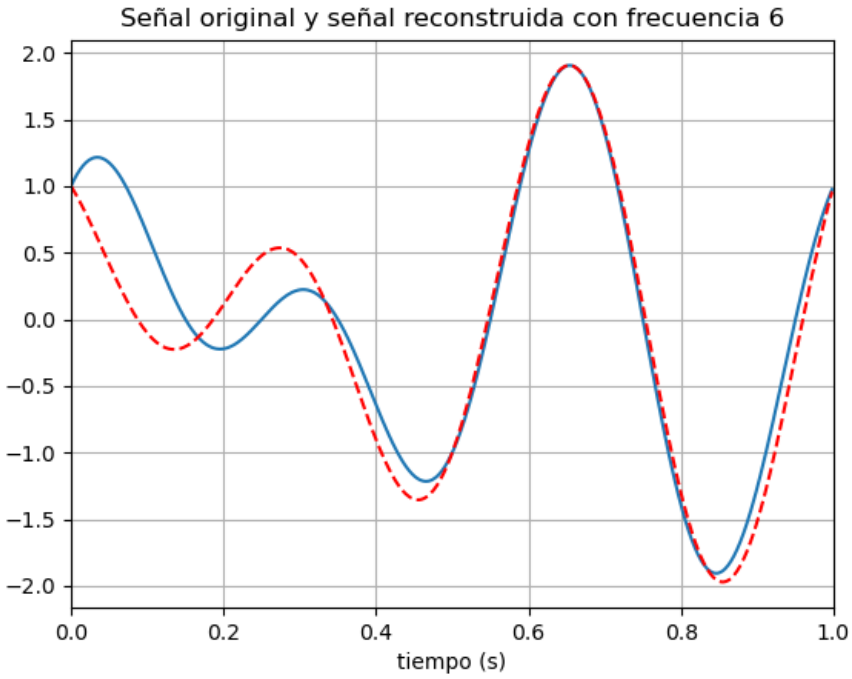


Figura 4: Señal original y Reconstruida con frecuencia de muestreo 6

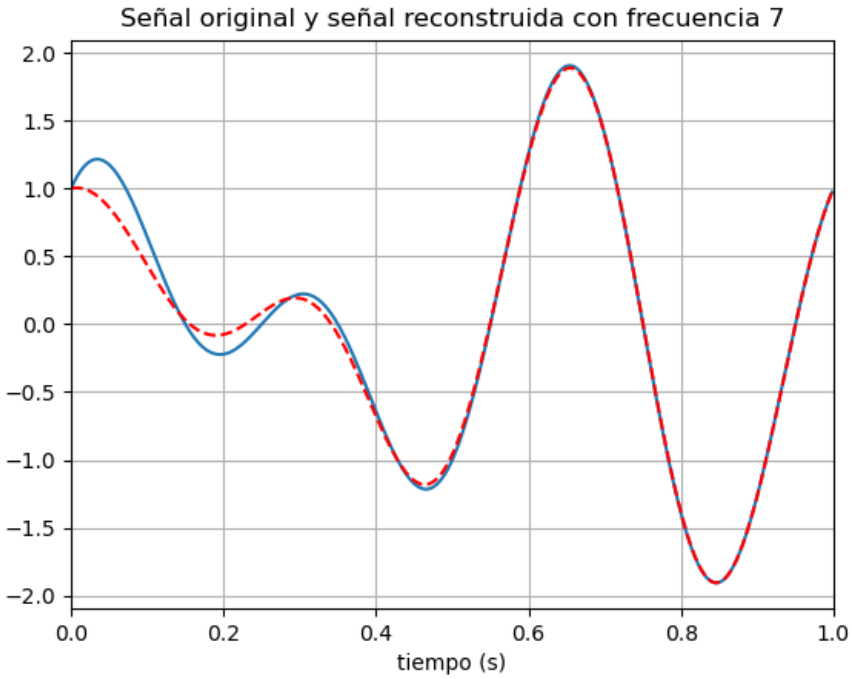


Figura 5: Señal original y Reconstruida con frecuencia de muestreo 7

2. Suponga que su empresa requiere construir un sistema de captura de audio digital de alta calidad que será comercializado masivamente. Como ingeniero(a) debe seleccionar el ADC de este sistema entre los siguientes componentes. ¿Cuál elegiría y por qué? Explique sus suposiciones.

   a) MAX11190, 12 bits, [0, 3.6] Volts, 3000 ksps (kilo muestras por segundo), USD$4.22 (desde mil unid.)

   b) MAX11156, 18 bits, [-5, 5] Volts, 500 ksps (kilo muestras por segundo), USD$17.75 (desde mil unid.)

   Solución:

   En este caso elegiría la opción b. Para fundamentar esta respuesta, se aborda en diferentes puntos cada variable a analizar.

   - **Resolución**: representa el número de bits de salida que el ADC puede generar. Este número afecta inversamente a la velocidad o tasa de muestra del dispositivo, es decir, a mayor resolución, menor será la velocidad de conversión.
     - De acuerdo a lo indicado, el ADC MAX11190 posee una resolución de 12 bits, mientras que el ADC MAX11156 tiene una resolución de 18 bits. **Por esto, se puede decir que el MAX11156 es mejor en cuanto a resolución.**
   - **Velocidad**: representación de cuantas muestras por segundo puede realizar el ADC.
     - El ADC MAX11190 posee mayor velocidad que el ADC de la alternativa b, en efecto, como se explicó en el punto anterior, la resolución del primer ADC mencionado es menor, por lo que su velocidad de representación será mayor. En este punto, **el mejor candidato, respecto a velocidad, es el ADC del enunciado a.**
   - **Voltaje**: la tensión de entrada indica el trabajo para mover una carga de prueba entre dos puntos. En un ADC, la tensión es útil para calcular el Error de Cuantificación, correspondiente al ruido o pérdida de información al pasar de una señal analógica a digital.

- Para poder entender el análisis de esta variable, se presenta el siguiente desarrollo matemático.
  - Inicialmente se procede a calcular la resolución Q, pero esta vez definida eléctricamente en términos de voltios, mediante la fórmula:

$$Q = \frac{E_{FSR}}{2^M} \tag{1}$$

E_{FSR} = Tensión superior - Tensión inferior

M = Resolución del ADC en bits.

Entonces se obtiene:

$Q = 8{,}789 \cdot 10^{-4}(V)$ para MAX11190 (opción a)

$Q = 3{,}815 \cdot 10^{-4}(V)$ para MAX11156 (opción b)

- A partir de los valores Q obtenidos, es posible calcular el error de cuantificación SQNR con la fórmula:

$$SQNR = 20 \cdot log_{10}(2^Q) \approx 6{,}02 \cdot QdB \tag{2}$$

Como se puede notar, el error de cuantificación o ruido es menor en la opción b, respecto a la opción a. Por lo que, se dice que **en términos de Voltaje es mucho mejor el ADC MAX11156.**

- **Precio:** indicador de costo por unidad en dólares.
  - e puede notar que el ADC de la opción a, es casi cuatro veces más costoso que el ADC de la opción b. Por esto, se puede decir que **MAX11190 es mejor en cuanto a precio.**
- **Conclusión:** como se puede notar, el ADC MAX11190 es mejor respecto a precio y velocidad, sin embargo, el ADC MAX11156 presenta una mejor resolución y un menor ruido. En vista de que se pide comercialización masiva y alta calidad, se selecciona la opción b: MAX11156. Esto se debe a que desde mi punto vista, existe un factor destacable al momento de comparar ambos ADC, el cual corresponde al precio y la calidad. La calidad de un producto debe estar por sobre el precio de este.

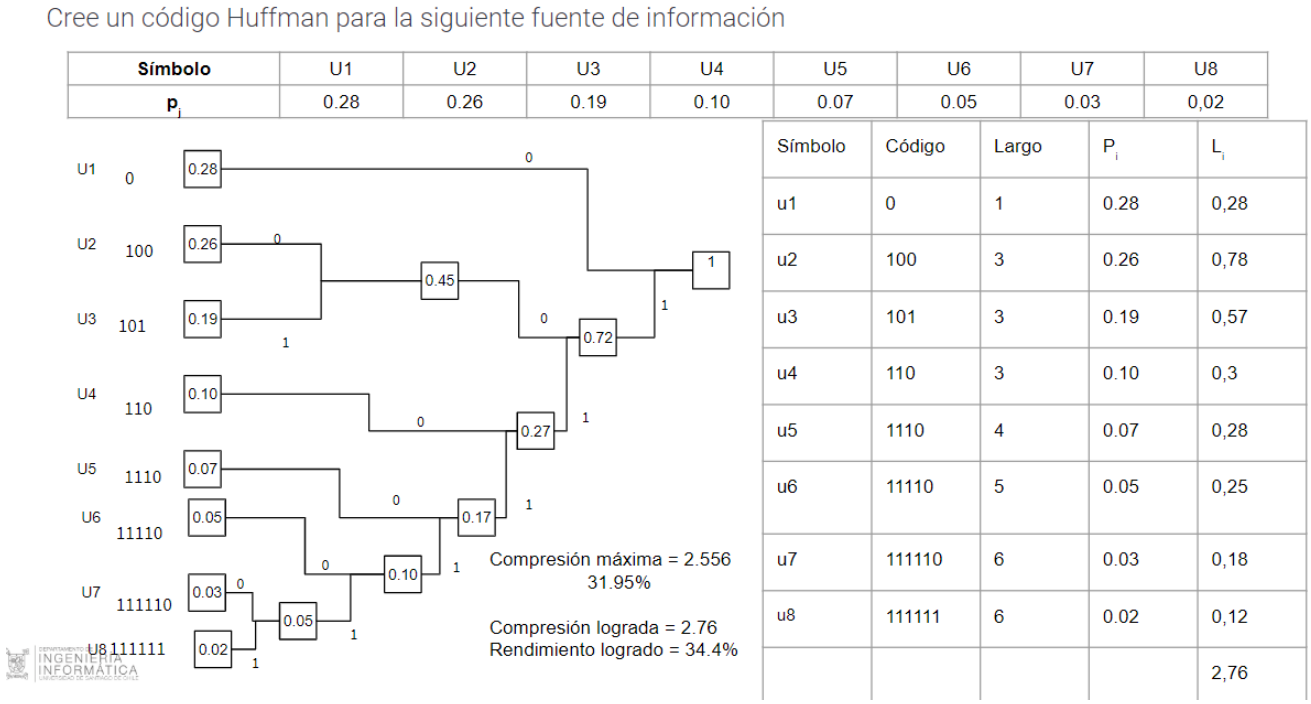
2. C3-P2: Códigos

1. Diseñe un código FANO y un código Huffman para la siguiente fuente de información

2. Calcule la tasa de codificación de cada uno y determine su rendimiento ¿Cuál es mejor?

Solución:

En la **Figura 6** se muestra el código Huffman elaborado en clases. De acuerdo a este, se obtiene un Rendimiento de 34.4 % con una compresión de 2.76.



Símbolo	Código	Largo	P_i	L_i
u1	0	1	0.28	0,28
u2	100	3	0.26	0,78
u3	101	3	0.19	0,57
u4	110	3	0.10	0,3
u5	1110	4	0.07	0,28
u6	11110	5	0.05	0,25
u7	111110	6	0.03	0,18
u8	111111	6	0.02	0,12
				2,76

Figura 6: Diseño de código Huffman

Por otro lado, en la **Figura 7**, se muestra el código Fano también creado en clases. De acuerdo a este, se obtiene un Rendimiento de 32.62 % con una compresión de 2.61. Por ello, se puede concluir que **el código Fano es mejor en cuanto a rendimiento.**

Cree un código Fano para la siguiente fuente de información

Símbolo	U1	U2	U3	U4	U5	U6	U7	U8
p_j	0.28	0.26	0.19	0.10	0.07	0.05	0.03	0,02

Símbolo	Pi	1	2	3	4	5
U1	0,28	0	0			
U2	0,26		1			
U3	0,19	1	0	0		
U4	0,1			1		
U5	0,07		1	0		
U6	0,05			1	0	
U7	0,03				1	0
U8	0,02					1

Código	Largo	Pi	Li	H
00	2	0.28	0.56	0.518
01	2	0.26	0.52	0.505
100	3	0.19	0.57	0.455
101	3	0.1	0.3	0.332
110	3	0.07	0.21	0.268
1110	4	0.05	0.20	0.216
11110	5	0.03	0.15	0.151
11111	5	0.02	0.1	0.112
			2.61	2.556

Compresión máxima = 2.556
31.95%

Compresión lograda = 2.61
Rendimiento logrado = 32.62%

El rendimiento logrado en el código Fano, es mejor que en el de Huffman (32.62% vs 34.4% respectivamente)

Figura 7: Diseño de código Fano