UNIVERSIDAD SERGIO ARBOLEDA

Teoría de sistema lineales

Laboratorio 04: Convolución de señales en MATLAB

Escuela de Ciencias exactas e Ingeniería

Profesor: Marco Teran

Name: _____ **Deadline:** 10 de octubre de 2023

Resumen

Durante el desarrollo del presente laboratorio se pondrán a prueba las técnicas de MATLAB para evaluar las convoluciones. Se busca comprender el concepto de convolución e implementar varios filtros y comprender su efecto en los archivos de audio e imágenes digitales. Se aplican las propiedades de la convolución utilizando MATLAB: la linealidad, la invariancia del tiempo y la asociación.

1. Desarrollo de la práctica de laboratorio

Una vez entendido la naturaleza de una secuencia discreta, es importante conocer el efecto que tiene esta sobre un sistema específico. Un sistema recibe como entrada una señal discreta, su salida es una versión transformada de la entrada. Sea un sistema denominado $T\{x[n]\}$, que representa la transformación que hace el sistema sobre la secuencia discreta $x[n]$. Definimos un sistema LTI de la siguiente manera:

$$T\left\{\alpha x_1[n] + \beta x_2[n]\right\} = T\left\{\alpha x_1[n]\right\} + T\left\{\beta x_2[n]\right\}$$

donde α y β son constantes. Esta propiedad se denomina linealidad.
La invarianza en el tiempo se puede definir de la siguiente forma:

$$T\left\{x[n]\right\} = y[n]$$
$$T\left\{x[n-k]\right\} = y[n-k]$$

Esto quiere decir que si el sistema genera una salida $y[n]$ en un instaste de tiempo dado, el sistema deberá genera la misma salida pero desplazada en el mismo instaste de tiempo después en que fue desplazada su entrada. Es decir, el sistema no cambia con el tiempo.

1. **Responda brevemente en la sección de Marco Teórico de su plantilla de laboratorio las siguientes preguntas:**
   (a) Definir qué es la convolución y sus principales propiedades.
   (b) Investigue acerca de la generación de señales aleatorias y las características estocásticas del ruido blanco gaussiano aditivo (AWGN, *ing.* Aditive White Gaussian Noise).
   (c) Investigue sobre la generación en MATLAB para una relación señal ruido (SNR, *ing.* Signal to noise ratio) predeterminada utilizando la función `out = awgn(in,snr,signalpower)`. Determine cada uno de sus argumentos. Que ocurre cuando el argumento `signalpower` es igual a `'measured'`.
   (d) ¿Que es la desconvolución y como se implementa en MATLAB?

2. Operación de convolución utilizando MATLAB

La forma de determinar la salida del sistema a partir de una entrada, asumiendo que se conoce la respuesta al impulso $h[n]$ que caracteriza al sistema, es mediante la suma de convolución. La convolución vista como operación matemática, combina dos señales y da como resultado una tercera señal. Suponiendo que tenemos dos funciones, $x[n]$ y $h[n]$, la convolución es una suma que expresa la cantidad de solapamiento de una función h al desplazarse sobre la función x.

La convolución se expresa de la forma

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

La suma de convolución es una operación que posee las propiedades conmutativa y distributiva. Dependiendo de la aplicación de la convolución, las funciones pueden ser sustituidas por señales, imágenes u otros tipos de datos.
La convolución y sus aplicaciones pueden implementarse de varias maneras en MATLAB. En el paquete matemático MATLAB se puede realizar la suma de convolución de dos secuencias discretas mediante la función `conv`.

1. El siguiente código permite encontrar la suma de convolución de dos señales $x[n]$ y $h[n]$:

```
x=[0.2, 1.4, 2.6, 5.1, 3.4, 8.4];
h=[1.0, 4.2, 3.7, 0.8, 3.9];
y=conv(x,h);
```

Realice de forma teórica esta convolución y compare los resultados obtenidos.

2. Dada las secuencias $x1$, $x2$ y h, verificar la propiedad lineal de la convolución mediante la aditividad:

```
x1=[0.6, 2, 6, 3, 2, 0.5, 7];
x2=[0.3, 2, 5, 6, 4, 0.3, 8];
h=[1 1 2 2 3 4 5 5];

y1=conv(x1,h)+conv(x2,h);;
y2=conv(x1+x2,h);
```

Comparar los resultados de $y1$ y $y2$. De manera similar, valide las propiedad de homogeneidad.

3. La respuesta al impulso de un filtro discreto LTI esta dada por:

$$h[n] = \{1, 1, 1, 1, 1, -1, -1, -1, -1, -1\}.$$

Utilice MATLAB para determinar la salida si la entrada es la señal es:

$$x[n] = \cos\left(\frac{\pi n}{2}\right)$$

Represente gráficamente la señal de entrada, su respuesta al impulso y su salida.

4. **Convolución de señales discretas:** Utilice un vector de tiempo discreto descrito por `n=-50:50` para resolver el siguiente problema. Un sistema LTI discreto presenta la siguiente respuesta al impulso:

$$h[n] = (0{,}6)^n u[n].$$

Con la ayuda de MATLAB calcule la respuesta del sistema para una entrada:

$$x[n] = u[n].$$

Dibuje para cada caso, las tres señales que interfieren en el proceso ($h[n]$, $x[n]$ y la salida $y[n]$).

```
ts=.01;
gox=0; stopx=1/2;
goh=0; stoph=2;

tx=gox:ts:stopx;
```

```
x=exp(-tx);

th=goh:ts:stoph;
h=exp(-th);

ty=gox+goh:ts:stopx+stoph;
y=ts*conv(x,h);

figure('Convolución')
subplot(311)
plot(tx,x)
xlabel('t'), ylabel('x(t)')
subplot(312)
plot(th,h)
xlabel('t'), ylabel('h(t)')
subplot(313)
plot(ty,y)
xlabel('t'), ylabel('y(t)=x(t)*h(t)')
```

- Comente cada una de las líneas de código
- Agregue al informe los resultados arrojados por el código.
- ¿Qué significado tiene `ts` y por qué se multiplica por el resultado de la convolución?

3. Convolución en el procesamiento de señales

La convolución se utiliza en el procesamiento digital de señales para estudiar y diseñar sistemas lineales invariantes en el tiempo (LTI), como lo son los filtros digitales. La señal de salida, $y[n]$, en los sistemas LTI es la convolución de la señal de entrada, $x[n]$ y la respuesta al impulso $h[n]$ del sistema.
En la práctica, se utiliza el teorema de la convolución para diseñar filtros en el dominio de la frecuencia. El teorema de la convolución establece que la convolución en el dominio del tiempo es igual a la multiplicación en el dominio de la frecuencia. Las funciones de MATLAB como `conv` y `filter` permiten realizar la convolución y construir filtros desde cero.

3.1. Convolución en procesamiento de señales de audio

Los grupos musicales eventualmente requieren que sus grabaciones de estudio suenen como si se tocaran en algún lugar en vivo. Este efecto se conoce como reverberación *hall*. Un método para obtener este efecto es convolucionando la grabación de audio con la *respuesta de impulso* tomada en una sala de concierto u otro lugar con una acústica similar (*reverberizada*). La convolución también puede ser utilizada para alterar la voz.

Respuesta al impulso de una señal de audio: se han estudiado respuestas de impulso expresadas como funciones matemáticas, por ejemplo de la forma $h[n] = 2^{-n}u[n]$. Tales funciones con las que calculamos las convoluciones pueden modelar respuestas de impulso reales como el *eco* que se desvanece después de aplaudir en una sala de conciertos o el sonido de un platillo después de ser golpeado.

Para obtener una buena grabación de una respuesta al impulso real de audio para modelar por ejemplo una sala de concierto, los datos de audio deben incluir el sonido que se genera justo después de un impulso, hasta que este haya desaparecido lo suficiente. La distorsión en una salida mediante este método puede ser resultado de la convolución con una respuesta de impulso incorrectamente grabada o de un inicio en silencio antes de la grabación de la señal de audio de entrada.

5. **Experimento de efecto de voz:** se realizará el siguiente procedimiento para crear efectos de voz utilizando la convolución de forma experimental utilizando MATLAB. Se buscará que una señal de voz suene como si se hablara dentro de un gran vaso plástico, mediante la convolución de la voz normal con la respuesta al impulso del gran vaso plástico. La siguiente función permite grabar X segundos de audio:

```
function [ ] = recordToFile( inFileName, Fs, nbits, seconds )
    recordObj = audiorecorder(Fs, nbits, 1);
    record(recordObj);
    pause(seconds);
    stop(recordObj);
    audiowrite(inFileName, getaudiodata(recordObj), Fs);
end
```

Esta función permite grabar el audio en un archivo con un nombre y tiempo especificado mediante la frecuencia de muestreo y el tamaño de bits (8, 16 o 24).
A continuación, la siguiente función permite reproducir un archivo de audio soportado por MATLAB. `playAudioNormal` reproduce un archivo de audio a la frecuencia de muestreo con la que fue muestreado. El parámetro `inFileName` es el nombre tipo *string* (`' '`) del archivo de audio a reproducir.

```
function [ ] = playAudioNormal(inFileName)
    [y,fs] = audioread(inFileName);
    f = audioread(inFileName);
    sound(f,fs)
end
```

La siguiente función realiza la convolución entre un archivo de audio y la grabación de una respuesta al impulso.

```
function [] = convolveAudio(inputAudioFileName, impulseResponseFileName, ...
    outputFileName, Fs)
    x = audioread(inputAudioFileName);
    h = audioread(impulseResponseFileName);
    y = conv(x,h);
    audiowrite(outputFileName, y, Fs);
end
```

La función `colvolveAudio` calcula la convolución de las señales de audio almacenadas en dos archivos y crea un nuevo archivo de audio que contiene la señal de audio de salida. El parámetro `Fs` es la frecuencia a la que se muestrearon las señales de audio de entrada. `outputFileName` es el nombre del archivo de audio de salida. El parámetro `impulseResponseFileName` representa el nombre del archivo de audio de respuesta al impulso recortado. `inputAudioFileName` es el nombre del archivo de audio de entrada al sistema.
Procedimiento detallado:

- En la ventana de comandos de MATLAB, ejecute la función `recordToFile` y grabe su voz durante el tiempo que desee, mencione los integrantes de su grupo.
- Descargue la respuesta al impulso del gran vaso de plástico aquí
- Finalmente, llame la función `convolveAudio`, grabe y escuche el resultado.
- Realice el mismo procedimiento con las siguientes respuestas al impulso: impres1 y impres2.

Describa los resultados obtenidos en los audios de respuesta. ¿Qué se puede inferir de la duración de los audios de salida? ¿Qué efectos pudo apreciar? Descríbalos.

6. **Ejemplo de reverberación por convolución utilizando MATLAB:** Vaya a la página de Impulse Responses libres. Descargue la grabación mono ommnidireccional de la respuesta al impulso de *Terrys*

Factory Warehouse. Lea el audio con el comando `audioread` y almacénelo en una variable llamada `ir`. Almacene también su frecuencia de muestreo en una variable llamada `fs`. Escúchelo utilizando el comando `soundsc`. Descargue y lea también en MATLAB el archivo `singing.wav`, de la carpeta */laboratory/convolution/* del repositorio. Almacénelo en una variable llamada `x`. Convolucione ambas señales, escúchelas y grafíque el resultado utilizando el siguiente código:

```
y = conv(x,ir);
soundsc(y,fs)

figure();
plot(y,'b');
hold on
plot(x,'black');
t = 0 : 1/fs : length(ir)/fs;
title('Respuesta al impulso Señal reverberada');
xlabel('Tiempo'),ylabel('Amplitud')

figure();
plot(ir);
title('Respuesta al impulso');
```

Realice este mismo procedimiento con su propio archivo de sonido, puede ser un fragmento de una canción. Describa los resultados.

3.2. Convolución en procesamiento de imágenes

En el procesamiento de imágenes, el filtrado convolucional puede utilizarse para implementar algoritmos como la detección de bordes, la nitidez de la imagen y el desenfoque de la misma. Esto se hace seleccionando el *kernel* (núcleo) adecuado (matriz de convolución) y realizando la convolución de este con la imagen. Un *kernel* de imagen es una pequeña matriz de convolución o máscara que se utiliza para aplicar efectos como el desenfoque, la nitidez, el contorno o el relieve. Dependiendo de los valores de los elementos de la matriz, un *kernel* puede producir un amplio rango de efectos. Para mayor información revisar en la página oficial de MATLAB. Los *kernels* también se utilizan en el *Deep Learning* para la .ᵉˣtracción de características", una técnica para determinar las partes más importantes de una imagen. La siguiente ecuación representa la fórmula de convolución 2D discreta que se implementará para el laboratorio.

$$y[m,n] = x[m,n] * h[m,n] = \sum_{j=-\omega}^{\omega} \sum_{i=-\omega}^{\omega} x[i,j]h[m-i,n-j]$$

Image Processing Toolbox cuenta con funciones como `fspecial` e `imfilterv` para diseñar filtros que enfaticen ciertas características o eliminen otras en las imágenes.

1. **Máscara de mejora LOG para la mejora de la imagen mediante conv2:** El LOG (laplaciano de Gauss, *ing.* Laplacian of Gaussian) es uno de los más comunes detectores de regiones y se basa en el operador laplaciano de Gauss. El LOG representa una medida isotrópica bidimensional de la segunda derivada espacial de una imagen. El laplaciano de una imagen resalta las regiones de rápido cambio de intensidad y, por lo tanto, se utiliza a menudo para la detección de bordes.
   Este puede calcularse mediante un filtro de convolución. Dado que la imagen de entrada se representa como un conjunto de píxeles discretos, tenemos que encontrar un kernel de convolución discreto que pueda aproximar las segundas derivadas en la definición del laplaciano (más información). A continuación se muestran dos núcleos pequeños comúnmente utilizados,

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

El siguiente código permite realizar el laplaciano sobre una imagen utilizando la función `conv2`. En primer lugar se obtienen las dimensiones de la imagen. La variable `numberOfColorChannels` tiene el valor 1 para una imagen en escala de grises, y de 3 para una imagen a color RGB. Si la imagen es a color se convierte a escala de grises utilizando el comando `rgb2gray`.

```
grayImage = imread('peppers.png');
imshow(grayImage);
[rows, columns, numberOfColorChannels] = size(grayImage);

if numberOfColorChannels > 1
     grayImage = rgb2gray(grayImage);
end

subplot(2, 1, 1);
imshow(grayImage, []);
title('Imagen original en escala de grises', 'FontSize', fontSize, '...
    Interpreter', 'None');

h = [-1 -1 -1; -1 9 -1; -1 -1 -1];
filteredImage = conv2(double(grayImage), h, 'same');

subplot(2, 1, 2);
imshow(filteredImage, [0, 255]);
title('Imagen filtrada', 'FontSize', fontSize, 'Interpreter', 'None');
```

Realice este mismo procedimiento con una imagen propia. Describa los resultados obtenidos.

2. **Gaussian blur mediante la convolución de una imagen y el coeficiente gaussiano** El desenfoque gaussiano (*Gaussian Smoothing*) es un efecto de suavizado para imagenes digitiales. En esencia, el efecto mezcla ligeramente los píxeles vecinos, lo que provoca que la imagen pierda algunos detalles minúsculos y, de esta forma, hace que la imagen se vea más suave, es decir los bordes presentes en la imagen se ven afectados. El desenfoque gaussiano es un tipo de filtro de desenfoque de la imagen que utiliza una función gaussiana para calcular la transformación que debe aplicarse a cada píxel de la imagen. La fórmula de una función gaussiana en dos dimensiones se puede representar de la manera,

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2 2}}$$

El siguiente código permite realizar el Gaussian blur sobre una imagen utilizando la función `conv2`.

```
img = imread('peppers.png');
img = rgb2gray(img);
figure, imshow(img);
img = im2double(img);
sigma = 2;
G = zeros(5,5);
for i = 1:5
     for j = 1:5
          x = [-2,1,0,-1,2];
          y = [-2,1,0,-1,2];
          G(i,j) = exp(-(x(i)^2+y(j)^2)/(2*(sigma^2)));
     end
end
G_ = G/sum(sum(G,1),2);
A = conv2(img, G_);
figure, imshow(A);
```

Realice este mismo procedimiento con una imagen propia. Describa los resultados obtenidos.

3. **Experimente con diferentes filtros:** usando los códigos anteriores, implemente los siguientes filtros en las siguientes imágenes de prueba.

   - Filtros de suavizado: a) Box Filter (Filtro de caja) b) Weighted Box Filter (Filtro de caja ponderado) c) Filtro horizontal d) Filtro vertical e) Filtro de forma circular f) Filtro de caja de 5×5 g) Filtro de forma circular de 5×5. (descargar).

$$a)\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\quad b)\ \frac{1}{16}\begin{bmatrix}1&2&1\\2&1&2\\1&2&1\end{bmatrix}\quad c)\ \frac{1}{3}\begin{bmatrix}0&0&0\\1&1&1\\0&0&0\end{bmatrix}\quad d)\ \frac{1}{9}\begin{bmatrix}0&1&0\\0&1&0\\0&1&0\end{bmatrix}$$

$$e)\ \frac{1}{5}\begin{bmatrix}0&1&0\\1&1&1\\0&1&0\end{bmatrix}\quad f)\ \frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}\quad g)\ \frac{1}{25}\begin{bmatrix}0&1&1&1&0\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\0&1&1&1&0\end{bmatrix}$$

   Realice algunos comentarios generales sobre cada uno de los filtros y lo que observa. ¿Qué filtro funciona mejor en cada imagen? ¿Cuál es la diferencia entre el filtro de caja y el filtro de caja ponderado?

4. **Filtro de imágenes mediante la función imfilterv** se realizará el siguiente procedimiento para para filtrar una imagen bidimensional en escala de grises con un filtro de *5 por 5* de pesos iguales (filtro de promedio) utilizando `imfilterv` para realizar la convolución.

```
I = imread("coins.png");
figure
imshow(I)
title("Original Image")
h = ones(5,5)/25;
I2 = imfilter(I,h);
figure
imshow(I2)
title("Filtered Image")
```

   A continuación se muestra cómo filtrar una imagen a color (RGB) con el mismo filtro. Filtrar una imagen RGB con un filtro 2D es equivalente a filtrar cada plano de la imagen individualmente con el mismo filtro 2D.

```
rgb = imread("peppers.png");
imshow(rgb);
title("Original Image")
h = ones(5,5)/25;
I2 = imfilter(rgb,h);
figure
imshow(I2)
title("Filtered Image")
```

Realice este mismo procedimiento con una imagen propia. Describa los resultados obtenidos.

Referencias

[1] A.V. Oppenheim, A.S. Willsky, and S.H. Nawab. *Signals and Systems*. Prentice-Hall signal processing series. Prentice Hall, 1997.

[2] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, CA, USA, 1997.

[3] Bassem R. Mahafza. *Radar Systems Analysis and Design Using MATLAB*. CRC Press, Inc., Boca Raton, FL, USA, 2000.

[4] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[5] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[6] B.P. Lathi and R.A. Green. *Essentials of Digital Signal Processing*. Cambridge University Press, 2014.

[7] J.W. Leis. *Digital Signal Processing Using MATLAB for Students and Researchers*. Wiley, 2011.

[8] Monson H. Hayes. *Schaum's Outline of Digital Signal Processing*. McGraw-Hill, Inc., New York, NY, USA, 1st edition, 1998.

[9] L.W. Couch. *Digital and Analog Communication Systems*. Pearson internationl edition. Pearson/Prentice Hall, 2007.