UNIVERSIDAD SERGIO ARBOLEDA

Laboratorio 01: Introducción a LaTeXy MATLAB: procesamiento de señales

CODwork: SA2017IIa_LAB01
Deadline: 07 de septiembre

Resumen
En el presente laboratorio se aplicarán las bases para desarrollar los posteriores laboratorios y talleres. Se presentarán las características de presentación de los laboratorios. Se realizará código de MATLAB que permita la generación, tratamiento y almacenamiento de imágenes. Se compilarán códigos en LaTeXy se generarán archivos *PDF* de acuerdo al formato de presentación. En el presente laboratorio se implementarán algunas herramientas necesarias para el procesamiento de archivos *multimedia* en MATLAB. Se realizarán códigos en MATLAB para la generación, transformación y almacenamiento de archivos de audio e imágenes. Se abrirán en MATLAB archivos de video y se obtendrá información de estos.

Palabras clave
Latex — Matlab — Imágenes — WAV — Video — DIP — DSP

[1] *Escuela de Ciencias exactas e Ingeniería, Universidad Sergio Arboleda, Bogotá, Colombia*
***Contacto**: marco.teran@usa.edu.co

Índice

Objetivos

General:

- Introducir al estudiante en el procesamiento digital de señales implementando al paquete de software matemático MATLAB, para abrir, modificar y guardar archivos multimedia, de imágenes, audio y video, implementando al paquete de software matemático.

Específicos:

- Generar y realizar gráficas de señales continuas y discretas en la herramienta de software matemático MATLAB
- Guardar las variables en archivos de sistema y exportar los gráficos obtenidos en la simulación.
- Generar y compilar archivos en lenguaje LaTeX
- Abrir, procesar y guardar imágenes en MATLAB de acuerdo a las características de codificación de colores de estas.
- Abrir archivos de video en MATLAB y extraer información de estos.
- Generar archivos de audio, procesarlos y guardar los resultados.

Recursos utilizados

- **Software:**
  1. Paquete matemático *Matlab*.
  2. IDE de preferencia para generar archivos *PDF* utilizando código LaTeX.
- **Equipos:**
  1. Computador

1. Marco Teórico

Leer el Formato de Tareas (descargar), tanto en la versión compilada *PDF* como la versión de código *.tex*.
Definir en el *marco teórico*, en una sección llamada **Lenguajes de trabajo: LaTeXy MATLAB**:

- ¿Que son en LaTeX los *packages*? ¿Que es una distribución TeX/LaTeX?
- Definir los siguientes comandos de MATLAB y describir sus funciones: *print, clear, close, clc, round, sin, cos, 'LineWidth', 'Color', hold, xlabel, ylabel, legend, xlim, ylim, grid*.
- Investigar como abrir y guardar variables del espacio de memoria de Matlab en archivos tipo *.dat*.
- Investigar como guardar las imágenes del plot en formato *.png* y alta calidad, para exportarlas a LaTeX.

En el marco teórico es necesario definir las características, tipo de codificación y propiedades de un archivo de audio con extensión *.wav*. ¿Que diferencia hay entre archivos de audio *mono* y *estereo* al ser leídos y guardados en MATLAB?
Para leer una imagen en MATLAB es necesario conocer el tipo de formato que maneja la imagen y su codificación. Describa los formatos *.PNG*, *.BMP* y *.JPG*, diferencie, preferiblemente con una tabla. ¿Que son los modelos de color RGB y CMYK?
Investigue acerca de la generación de señales aleatorias y las características estocásticas del ruido blanco Gaussiano aditivo (AWGN, *ing.* Aditive White Gaussian Noise). ¿Como se genera en MATLAB una señal contaminada con ruido especificando una relación señal ruido (SNR, *ing.* Signal to noise ratio) predeterminada?
Determinar que son las codificaciones $Q15$ y $Q31$. ¿Como convertir cualquier valor a las codificación implementando MATLAB? ¿Definir que es el espectro de una señal, por que su importancia? ¿Que entidad en Colombia es la encargada de regular el espectro de radio? ¿Que función en MATLAB permite encontrar el espectro? Describa su modo de implementación de forma general como caja negra (entradas de la función, transformacion y salidas).

2. Desarrollo de la practica

2.1 Compilación del archivo LaTeX *formatWorks.tex*

Descargue el Formato de tareas y cámbiele el nombre a

SA2017IIa_LAB01_CODgroup

El archivo principal *formatWorks.tex* renombrelo de la misma manera. Compile con anticipación el archivo principal del *Formato de Tareas* antes de llegar a la clase de laboratorio [1]. No olvide actualizar las librerías antes de realizar la primera compilación (no funcionará y le saldrán muchos errores). En este documento encontrará ayuda para la instalación, actualización y compilación inicial (descargar). Es posible recurrir y utilizar editores en línea, como:

- ShareLaTeX
- Overleaf

2.2 Generar una gráfica en MATLAB y guardarla

MATLAB es un paquete de análisis matemático que trabaja solo con funciones vectoriales y matriciales, **discretas**.
Procedemos a abrir la herramienta de software matemático MATLAB.
Procedemos a crea un nuevo archivo *script* (CTRL+N) y lo guardamos con el nombre

SA2017IIa_LAB01_CODgroup

Escribimos el siguiente código para borrar memoria, cerrar todas las ventanas abiertas y limpiar el *Command Window*

```
clear all; close all; clc;
```

Luego *seteamos* los parámetros por defecto de las gráficas

```
set(0,'DefaultTextInterpreter', 'latex');
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize',16);
```

MATLAB no trabaja variables continuas en su dominio, por tanto se hace necesario realizar aproximaciones de señales discretas en el tiempo a continuas, es decir, que si se toma un intervalo de muestreo que tienda a cero (o una frecuencia de muestreo que tienda a infinito), a *simple vista* la señal parecerá *como* de tiempo continuo, es decir aparentemente *aproximadamente continua*. Esta frecuencia se llamará **frecuencia aparente de MATLAB** f_s.

```
settings.fs=1e4;
settings.ts=1/settings.fs;      % Step in time
```

Se aconseja tomar la costumbre de guardar parámetros característicos en forma de estructuras (orientadas a objetos).
A continuación se generará el vector de tiempos. Este se puede generar:

- Con un tiempo inicial t_i y un tiempo final dado t_f:

```
signal.ti=0;      % intitial time
signal.tf=0.1;    % final time
t=signal.ti:settings.ts:signal.tf;
```

[1] Más ayuda en la instalación de *MikTex*, *TeXstudio* y *ShareLaTeX*:

- Link1
- Link2
- Link3
- Ayuda ShareLaTeX

- Con una duración determinada t_m:

```
signal.tm=1e-1;             % signal duration
L=round(signal.tm/settings.ts); % Samples
t=(0:L-1)*settings.ts;
```

Se utilizará la primera forma en el laboratorio. Procedemos a generar dos señales sinusoidales, una con frecuencia $f_1 = 2{,}42kHz$ y la otra con el doble de esa frecuencia $f_2 = 2f_1$

```
f_1 = 1.5e3;  %[Hz]
f_2 = 2*f_1;  %[Hz]
s1=cos(2*pi*f_1*t); % Carrier based on time ...
    vector;
s2=cos(2*pi*f_2*t); % Carrier based on time ...
    vector;
```

Se procede a la generación de gráficas en una primera figura 1.

```
figure(1)
plot(t/1e-3,s1,'-','LineWidth',2,'Color',[0 ...
    0.4470 0.7410]);
```

El set de colores (RGB) permitido en las gráficas de MATLAB son los especificados en la tabla 1.

R	G	B	hex	name
0	0.4470	0.7410	'#0072bd'	blue
0.8500	0.3250	0.0980	'#d95319'	orange
0.9290	0.6940	0.1250	'#edb120'	yellow
0.4940	0.1840	0.5560	'#7e2f8e'	purple
0.4660	0.6740	0.1880	'#77ac30'	green
0.3010	0.7450	0.9330	'#4dbeee'	light-blue
0.6350	0.0780	0.1840	'#a2142f'	red

Tabla 1 – Set de colores MATLAB

Para dibujar las dos gráficas superpuestas se utilizan los comandos

```
hold on
plot(t/1e-3,s2,'-','LineWidth',2,'Color',[0.8500 ...
    0.3250 0.0980]);
hold off
```

Recreamos los parámetros del gráfico de tipo tamaño de la caja y las rejillas, etiquetas y leyendas

```
xmin=0; xmax=0.5; ymin=-1.1; ymax=1.1; % Box size
Box=[xmin xmax ymin ymax];
labels = {'t, [ms]', 's_n(t)'}; %labelx, labely
legends = {'$s_1(t)$','$s_2(t)=-\int_0^\pi {s_1(t...
    ) dt}$'};
h=legend(legends);
set(h, 'Interpreter','latex', 'Location', '...
    NorthEast','FontSize',16,'FontWeight','bold',...
    'Orientation','vertical');
xlabel(labels(1),'fontsize',16,'FontAngle','...
    Italic');
ylabel(labels(2),'fontsize',16);
xlim([Box(1) Box(2)]); ylim([Box(3) Box(4)]);
grid(gca,'minor');
```

Para guardar la imagen, es preferible guardarla con los formatos *.fig* y *.png* automáticamente para futuras modificaciones. Se implementan los siguientes comandos

```
set(gcf, 'Position', get(0,'Screensize')); % ...
    Maximize figure.
saveas(gca, 'example1','png');
print('example2', '-dpng', '-r600'); % Save as PNG...
     with 600 DPI
saveas(gca, 'example1.fig');
```

A continuación se generará una imagen aleatoria. Vale recordar que cada vez que se compile este *script*, la linea de generación de ruido se volverá a interpretar y a generar un nuevo ruido. Muchas veces es necesario generar una sola iteración de una señal aleatoria, y debe ser utiliza más adelante sin perder esos datos, por eso guardamos los valores de la variable en un archivo tipo *.DAT*

```
noise=0.1*randn(1,length(s1));
save('noisesignal.mat','noise');
```

Para cargar la variable con el archivo guardado con anterioridad se utiliza el código

```
load('noisesignal.mat','noise');
```

Procedemos a compilar el siguiente resto de código, y describimos cada uno de los comandos utilizados.

```
subplot(2,2,1)
plot(t/1e-3,noise,'-','LineWidth',1,'Color'...
    ,[0.3010 0.7450 0.9330]);
labels = {'t, [ms]', 'n(t)'}; %labelx, labely
xmin=0; xmax=0.5; ymin=min(noise)-0.2; ymax=max(...
    noise)+0.2; % Box size
Box=[xmin xmax ymin ymax];
xlim([Box(1) Box(2)]); ylim([Box(3) Box(4)]);
grid(gca,'minor');
title('Ruido aleatorio')
subplot(2,2,2)
h = histogram(noise);
h.Normalization = 'probability';
h.BinWidth = 0.01;
h.FaceColor = [0.3010 0.7450 0.9330];
h.EdgeColor = 'k';
title('Histograma del ruido aleatorio')
subplot(2,2,[3 4])
plot(t/1e-3,s1+noise,'-','LineWidth',1,'Color'...
    ,[0.3010 0.7450 0.9330]);
hold on
plot(t/1e-3,s1,'-','LineWidth',2,'Color',[0 0.4470...
     0.7410]);
hold off
xmin=0; xmax=0.5; ymin=min(s1+noise)-0.3; ymax=max...
    (s1+noise)+0.3;
Box=[xmin xmax ymin ymax];
labels = {'t, [ms]', 's_n(t)'}; %labelx, labely
legends = {'$s_1(t)+n(t)$','$s_1(t)$'};
h=legend(legends);
set(h,'Interpreter','latex', 'Location', '...
    NorthEast','FontSize',16,'FontWeight','bold',...
    'Orientation','vertical');
xlabel(labels(1),'fontsize',16,'FontAngle','Italic...
    ');
ylabel(labels(2),'fontsize',16);
xlim([Box(1) Box(2)]); ylim([Box(3) Box(4)]);
grid(gca,'minor');
title('Senal sinusoidal contaminada con ruido')
```

Todas las gráficas generadas se deben guardar e incluir

en el informe. Las lineas de código relevantes deben ser comentadas. Debe existir una correcta *identación*.

2.3 Generar señales continuas y discretas en Matlab

Los siguientes comandos aparecen en versión de estándar, pero usted deberá aplicar el estilo y guardar las gráficas tal cual como se mostró en la sección 2.2. Recuerde comentar cada linea de código.

En el siguiente *script* se generará una señal sinusoidal *aparentemente continua* (no se volverá a utilizar esta denominación, se debe sobreentender) con una frecuencia de $f_m = 10\,kHz$. Para que la señal discreta sea *aparentemente continua* se utilizará una frecuencia de muestreo de $f_s = 500\,kHz$. En el siguiente código representaremos el osciloscopio de la señal (comportamiento de la señal en el dominio del tiempo) y la magnitud de su espectro.

```
fs=500e3; % Frecuencia aparente de Matlab 500 kHz
f=10e3;   % Frecuencia de la señal sinusoide
ti=0;   % Tiempo incial de la señal [s]
nCyl=5;   % Numero de ciclos (periodos) a generar ...
    de la sinusoidal
t=ti:1/fs:nCyl*1/f; % Vector de tiempo conntinuo
x=cos(2*pi*f*t);  % Señal sinusoide
figure
plot(t,x)      % Dibujar la señal
title('Señal sinusoidal continua');
xlabel('Tiempo, [s]');
ylabel('Amplitud');
figure
[f,X]=spectrumGen(x,fs,2);     % Espectro ...
    bidireccional de la señal y vector de ...
    frecuencia
plot(f,X)      % Dibujar la señal
title('Espectro de la señal sinusoidal');
xlabel('Frecuencia, [Hz]');
ylabel('Amplitud');
```

Código 1 – Representación de una señal continua en Matlab

Para encontrar el espectro de la señal utilizamos la función `[freq,X]=spectrumGen(Signal,fs,side)`. La cual tiene como entradas la señal *Signal*, fs la frecuencia de muestreo aparente y *side* para establecer si el espectro es de un lado o dos lados ($side = 1$ o $side = 2$). Descargue `spectrumGen.m` en la carpeta *IntroductiontoLatexandMatlab/code/* del repositorio del repositorio de los laboratorios aquí.

Para trabajar con secuencias de tiempo discreto en MATLAB, en la practica es mucho más sencillo. Para gráficas de señales de tipo discreto, se utiliza el comando `stem` en lugar de `plot`, ambos son análogos.

A continuación, se representará en MATLAB la siguiente secuencia de tiempo discreto:

$$x[n] = \ldots,0,0{,}5,1,1{,}5,2,2{,}5,\underset{\uparrow}{3},2{,}5,2,1{,}5,1,0{,}5,0,\ldots$$

Para ello primero debemos definir el vector tiempo n. En este caso, para tener un panorama completo de la señal, lo ieal es crear un vector temporal entre $[-8,8]$, de la siguiente forma: `n=-8:1:8`, lo cual significa que generará un vector de -8 a 8, con un paso (*step*) de una unidad. El siguiente código seutilizara para representar de forma gráfica la señal, el espectro no presenta ningún tipo de interés.

```
n=-8:1:8; % Vector de tiempo discreo
x_n=[0,0,0,0.5,1,1.5,2,2.5,3,2.5,2,...
   ...1.5,1,0.5,0,0,0]; % señal x_n
stem(n,x_n);  % Dibujar la señal
title('Señal de tiempo discreto');
xlabel('n, [sample]');
ylabel('Amplitud');
```

Código 2 – Señal de tiempo discreto en Matlab

Dibuje de forma discreta, y para un itervalo adecuado donde se puedan apreciar todas las características, las siguientes funciones `sign`, `sawtooth`, `pulstran...`, `rectpuls`, `tripuls` y `gauspuls`.

2.4 Exportar señales de MATLAB a archivos *.wav*

Un archivo *.wav* es un estándar de audio para computadores[2].

A continuación se muestran los comandos utilizados para abrir, leer y guardar archivos *.wav* en MATLAB. La linea de comando

`[x,fs,bits] = wavread('filename')`

Lee un archivo llamado *'filename.wav'* y lo convierte en el vector x como variable de MATLAB; fs extrae la frecuencia de muestreo del archivo y *bits*, tal cual como su nombre lo indica, representa la resolución de bits de la señal muestrada. El comando

`x = wavread('filename',Nsamples)`

lee las primeros *Nsamples* muestras del archivo *.wav* en caso de que no se desee extraer el audio completo. El comando

`wavwrite(x,fs, 'filename')`

crea a partir del vector x un archivo de audio *.wav* con una frecuencia de muestreo fs y llamado *'filename.wav'*. La resolución estándar para archivos de audio es de $16\,bits$ o ± 32768 niveles. La amplitud se encuentra escalada y restringida a un intervalo de $[-1,1]$ en el vector x. El comando

`wavwrite(x,Fs,bit, 'filename')`

cambia la resolución de bits del archivo.

A continuación genere en MATLAB una señal sinusoide con una amplitud de $A = 0{,}1$, una frecuencia $f = 100\,Hz$, para $N = 100000$ muestras, y una frecuencia de muestreo de $f_s = 22050\,Hz$.

Escuche la señal, y luego guarde esta en un archivo *.wav*, cuyo nombre de archivo es el código asignado al laboratorio. Adjunte el archivo *.wav* en el proyecto *.zip* de su laboratorio.

El tiempo total de la señal: $N_{samples}/fs = 4{,}535\,s$.

[2] WAV, articulo *Wikipedia* https://en.wikipedia.org/wiki/WAV

```
N_samples = 100000; %Numero total de muestras
fs = 22050; %Frecuencia de muestreo
fc = 100; %Frecuencia de portadora
t = (0:N-1)/fs; %Vector temporal
x = 0.1*sin(2*pi*fc*t); %Sinusoidal
sound(x,fs) %Reproducir sinusiodal
wavwrite(x,fs,sinsound.wav); %Guardar sinusoidal
```

Código 3 – Exportar señales de Matlab a archivos *.wav*

Genere y guarde una versión ruidosa de esta señal, es decir adicione ruido blanco Gaussiano con una potencia igual a la mitad de la potencia de la señal de audio original. Llame esta versión del audio contaminado tal cual como la anterior, pero agregue sin espacios al final del nombre la palabra *NOISED*.

2.5 Tratamiento de imágenes en MATLAB

Para abrir la imagen `lena.png`, de tipo RGB que se encuentra en el repositorio, se debe utilizar la función:

```
RGB = imread('Lena.png');
```

En esta linea de código anterior, la imagen[3] se guarda en una variable-vector denominado RGB. Describa las dimensiones de la variable RGB y su significado. ¿Que tipo de valores toma el vector? Defina la estructura, y por ultimo genere las gráficas de los 3 histogramas correspondientes a los datos de las tres matrices. Utilice las funciones especializadas de MATLAB de DIP (ing. Digital Image Processing) si así lo desea.

Para mostrar los datos de la imagen RGB se utiliza la función `imshow(RGB)`.

Para convertir y mostrar una una imagen a escala de grises utilicemos las siguientes lineas de comando:

```
gray = rgb2gray(RGB);
imshow(gray)
```

Código 4 – Exportar señales de MATLAB a archivos *.wav*

Describa las dimensiones de la variable *gray*. ¿Que tipo de valores toma? Defina su estructura, y genere un histograma con sus valores.

El siguiente código extrae las componentes en rojo,verde y azul de la imagen `peppers.png` que se encuentra en la carpeta de *IntroductiontoLatexandMatlab/pictures* en el repositorio[4]:

```
close all; clear all; clc;
myimg=imread('peppers.png', 'PNG');
size(myimg)
nbcl=512; mcol=[0:nbcl-1]'/(nbcl-1);
mypal=zeros(nbcl,3,3); mypal(:,1,1)=mcol;
mypal(:,2,2)=mcol; mypal(:,3,3)=mcol;
for k=1:3
    figure(k), imagesc(myimg(:,:,k))
    colormap(mypal(:,:,k)); axis('image')
end
```

Código 5 – Extracción RGB de una imagen

[3]Descargar la imagen de prueba de la carpeta *IntroductiontoLatexandMatlab/pictures* del repositorio (Descargar).

[4]Descargar `peppers.png`

1. Escriba una función código *.m* que lea una imagen, sea de color o en escala de grises, y la convierta en una imagen en blanco y negro (solo 1 *bit*). **Nota:** No utilice la función `rgb2bw`.
2. (**Investigación**) A una imagen en escala de grises agregue ruido blanco, de acuerdo a las siguientes relaciones señal ruido (SNR, *ing.* Signal to noise rate): $-4dB, 0dB, 2dB$. Muestre en *subplots* la imagen original, y las tres versiones ruidosas de esta.

2.6 Archivos de video en MATLAB

EL programa matemático MATLAB puede leer distintos formatos de archivos de video como *.avi*, *.mpg* y *.wmv*. Para reproducir un video con MATLAB, basta con utilizar la linea de código

```
implay('rbsp_launch_720p.mp4');
```

Descargar el archivo de video aquí.

Para obtener la información de un archivo de video se puede utilizar el siguiente código:

```
movieObj = VideoReader('grail_launch_720p.wmv'); %...
      Abre el archivo de video
get(movieObj) %Muestra toda la información
nFrames = movieObj.NumberOfFrames;
width = movieObj.Width;
height = movieObj.Height;
```

Código 6 – Extraer información de un archivo de video en MATLAB

Incluya en el informe la información del video.

Utilice el siguiente código para abrir, mirar la información de un archivo de video y mostrar de forma individual cada uno de sus *frames* (descargar archivo de video *oneCCC.wmv* que se encuentra en la carpeta *IntroductiontoLatexandMatlab/vid/* del repositorio del laboratorio enlace):

```
clear all
close all
movieObj = VideoReader('oneCCC.wmv');
get(movieObj)
nFrames = movieObj.NumberOfFrames;
%Para leer y mostrar cada uno de los frames
for iFrame=1:2:nFrames
I = read(movieObj,iFrame);
fprintf('Frame %d\n', iFrame);
imshow(I,[]); %Muestra la imagen
pause(0.1);
end
```

Código 7 – Exportar señales de MATLAB a archivos *.wav*

Comente cada una de las lineas del código anterior e incluyalas en el informe, muestre los resultados.

Para más ejemplos de código MATLAB, visitar `http://www.mathworks.com/help/matlab/ref/videoreader.read.html`

Recomendaciones

Los laboratorios se realizarán en horas de clases estipuladas con anticipación, a 5 minutos de empezar formalmente la sesión de clases todos los estudiantes deben

contar con **TODOS** los materiales y componentes necesarios y la guía de laboratorio **IMPRESA**.
El formato de informe de laboratorio, las condiciones, la forma de evaluación, presentación y sustentación estarán explicitas en el documento presentación de informes en LaTeX.
Los archivos del laboratorio, tanto los *scripts* de MATLAB y LaTeX, deben ser enviados al correo oficial del docente y presentado de forma impresa el día estipulado. Recuerde que el asunto y el nombre del archivo a enviar al correo deberán ser:
$SA2017II a_L AB01\_CODgroup$.
Si el estudiante se ausenta el día de desarrollo de la practica, no se aceptará su informe de laboratorio (a menos que la inasistencia sea resultado de una ausencia justificada oficialmente) sin excepción. Es importante entender que laboratorios presentados sin el formato correspondiente son los más rápidos de calificar, cero.
Todas las declaraciones y lineas de código relevantes deben estar comentadas. El script se debe entender por si solo, pero cualquier integrante del grupo debe estar en capacidad de sustentar cada linea.
Cada gráfica debe estar etiquetada (ejes, leyendas y título). Utilice de forma eficiente el documento, lo que escribe y la inserción de las gráficas, con el fin de obtener un informe claro, corto y conciso
Envíe el código fuente como archivos apartes en la carpeta **code** que se encuentra en el *usathemeformat* (no incluya códigos completos en el informe a menos que sea necesario para el desarrollo del informe).
No se aceptarán laboratorios fuera de la fecha estipulada (a menos que la demora sea resultado de una ausencia justificada oficialmente) sin excepción.

Resultados

En este apartado incluya algunas imágenes (legibles y con leyenda) del desarrollo de la práctica, describa los distintos resultados producidos y sus posibles causas.

Conclusiones

Es obligatorio que todos los trabajos tengan conclusiones. Esta debe contener una revisión de todos los temas claves del trabajo. Esta a su vez debe presentar el análisis de los resultados que se obtuvieron. Esta sección NO es un resumen.
Realice cuatro o más conclusiones de lo aprendido, demostrado y verificado en la práctica.
Recuerde que pésimas conclusiones le quitarán importancia a todo el esfuerzo y trabajo realizado durante el laboratorio.

Recursos

El estudiante deberá enumerar las referencias consultadas para el desarrollo de la práctica, por ejemplo de hojas de especificaciones, textos, Internet, etc. **Texto guía:** El texto guía a utilizar es el siguiente [1]. En la sección de referencias también se podrán encontrar todos los libros recomendados para un desarrollo exitoso del curso. Para manejo del diferente material de consulta o de trabajo, se dispone del Aula Virtual. Por favor matricule e inicie sesión en el curso virtual para todos los materiales relacionados con esta asignatura.

Anexos

Relación de documentación de soporte para la el desarrollo de la práctica, tales como: como hojas de especificaciones, planos de impresos, manuales, códigos implementados en MATLAB, etc.

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.