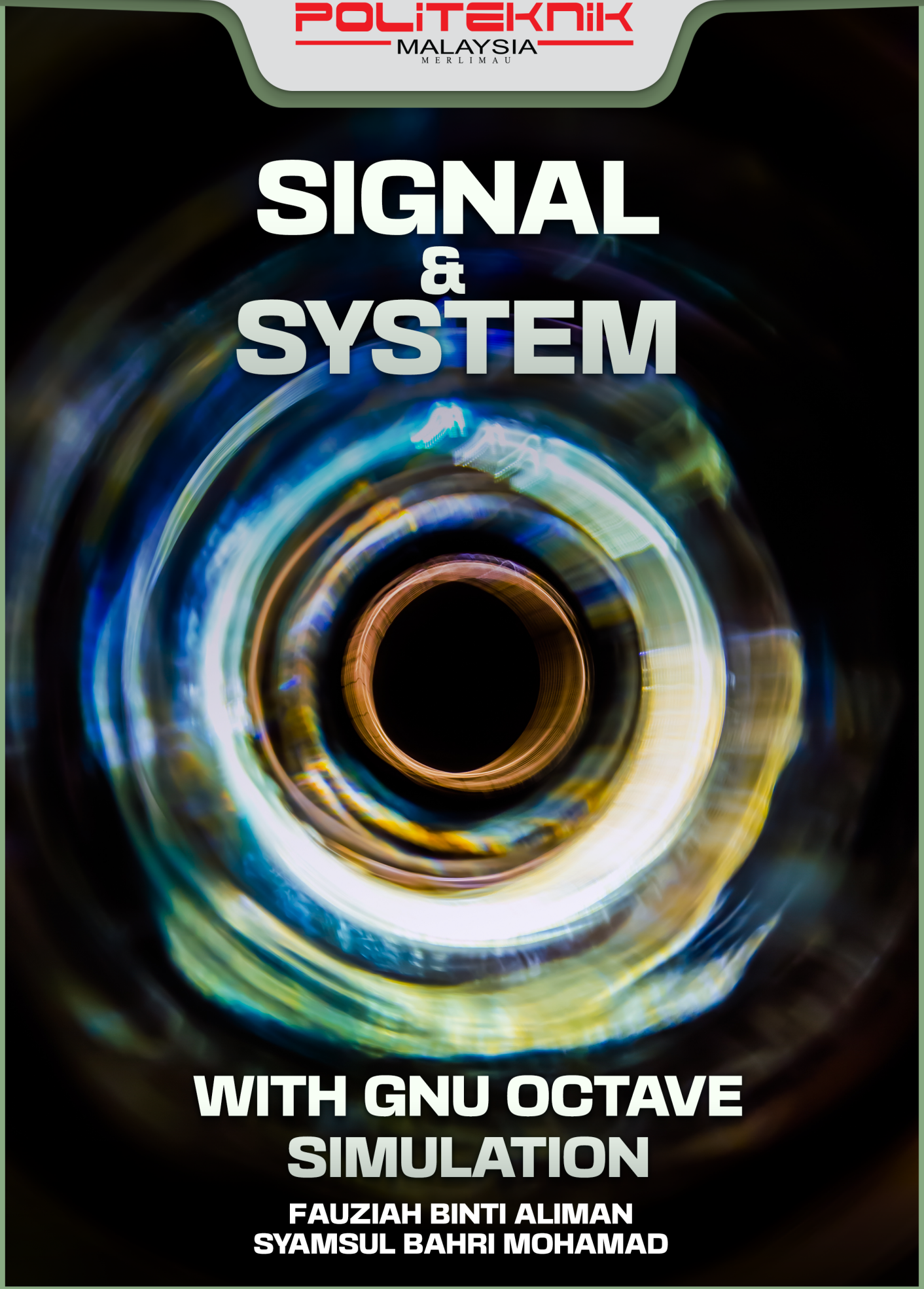
POLITEKNIK MALAYSIA
MERLIMAU

SIGNAL & SYSTEM



WITH GNU OCTAVE SIMULATION

FAUZIAH BINTI ALIMAN
SYAMSUL BAHRI MOHAMAD

SIGNAL AND SYSTEM

WITH GNU OCTAVE SIMULATION

FAUZIAH BINTI ALIMAN
SYAMSUL BAHRI MOHAMAD

Writer
Fauziah Binti Aliman
Syamsul Bahri Mohamad

Published in 2022

Perpustakaan Negara Malaysia Cataloguing-in-Publication Data

Fauziah Aliman
SIGNAL AND SYSTEM: WITH GNU OCTAVE SIMULATION / FAUZIAH BINTI ALIMAN, SYAMSUL BAHRI MOHAMAD.
Mode of Access: Internet
eISBN 978-967-2762-14-0
1. Signal processing.
2. Signal theory (Telecommunication).
3. System analysis.
4. Government publications--Malaysia.
5. Electronic books.
I. Syamsul Bahri Mohamad.
II. Title.
621.3822

Published by:
Politeknik Merlimau, Melaka
KB1031 Pej Pos Merlimau,
77300 Merlimau Melaka

Acknowledgement

In the name of Allah, the Most Gracious and the Most Merciful.

My humblest gratitude to the Holy Prophet Muhammad (Peace be upon him) whose way of life has been continuous guidance for us. Thank you to everyone who has been supporting us in preparing this e-book. Never stop learning !

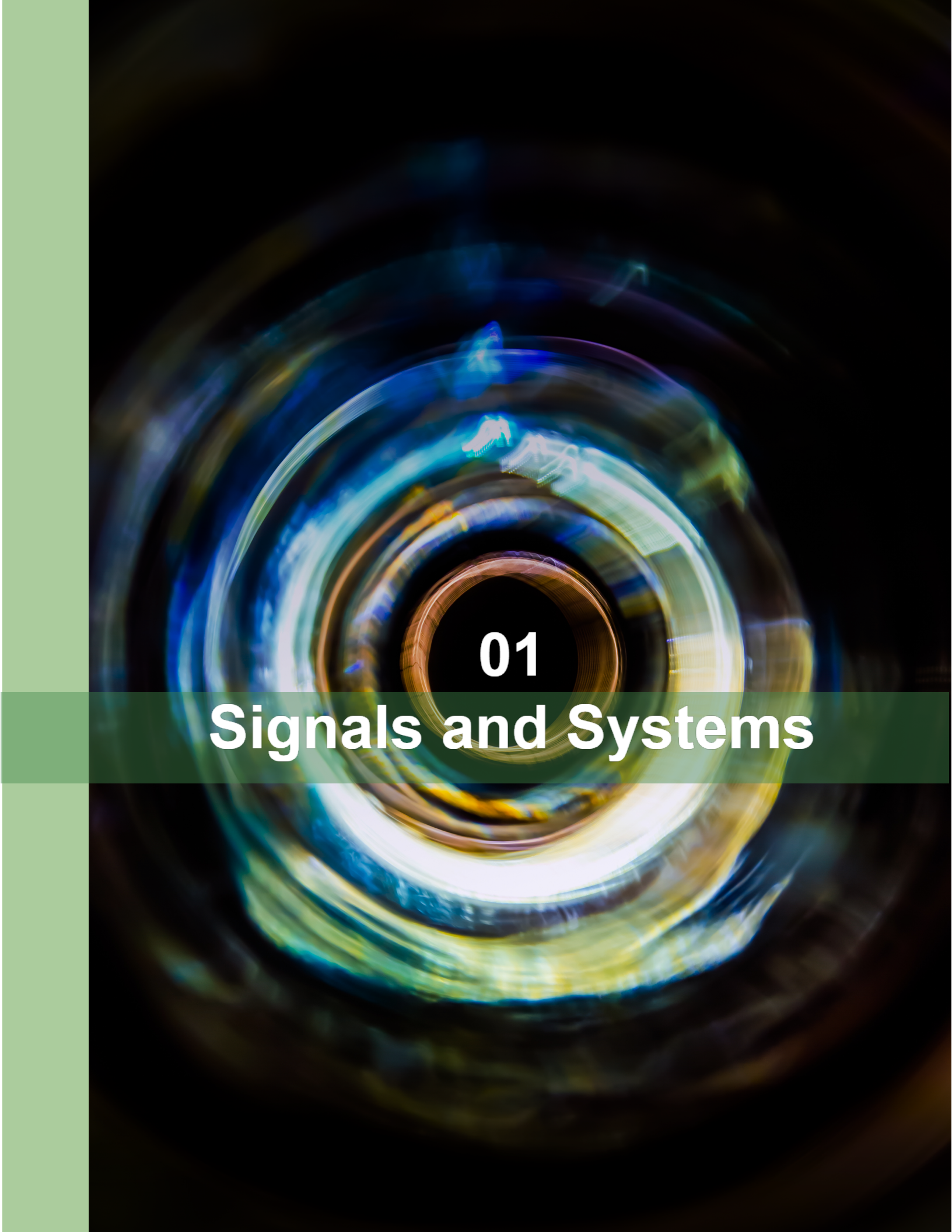
Preface

This book includes a thorough study of continuous and discrete time signals and systems, as well as several GNU OCTAVE examples. It is intended for junior and seniorelectronic engineering students, as well as working professionals who wish to study independently. A fundamental course in differential and integral calculus, as well as basic electric circuit theory, are required.

This book is suitable for one semester course. The author has taught the subject material at Polytechnic Merlimau in Melaka, Malaysia, for many years and has covered all of the contents in 14 weeks, with two lecture hours weekly. GNU Octave is open- source software that can be freely distributed. Students may redistribute and/or modifyit under the provisions of the Free Software Foundation's GNU General Public License (GPL).

In Chapter 1, the fundamental signals are covered, supported with various examples. This chapter's goal is to teach the reader how to express any continuous-time or discrete- time waveform in terms of the unit step function, unit impulse function, and subsequent operation. The convolution integral for continuous-time signals and the convolution sum for discrete-time signals are discussed in Chapters 2. The Laplace Transform and continuous-time signals are introduced in Chapter 3. The discrete-timesignals and the Z transform are discussed in Chapter 4. The Fourier Transform and FastFourier Transform (FFT) are covered in Chapters 5 and 6, with the most basic explanations available.

Table of Contents



01
Signals and Systems

Signals and Systems

Signals and systems theory and idea are required in practically all electrical engineering specialties, as well as many other technical and scientific disciplines. The mathematical description and representation of signals and systems, as well as their classifications, are covered in this chapter along with a few key fundamental signals that are critical to the studies.

Discrete time and continuous time are two different frameworks for modelling variables that change over time in mathematical dynamics.

Time is considered as a discrete variable in discrete time, which regards values of variables as occurring at distinct, separate "points in time," or equivalently as remaining unchanging across each non-zero region of time.

As time passes from one time period to the next, a non-time variable jumps from one value to the next. This perception of time is analogous to a digital clock that displays a fixed reading, for example, of 10:37 for a period of time before jumping to a new fixed reading of 10:38, and so on. In most cases, measurements are taken at sequential integer values of the variable time.

Source: pixabay

Classification of Signal

A signal is a function that represents a physical quantity or variable and usually incorporates information about the phenomenon's behavior or nature. In an resistor-capacitor (RC) circuit, the signal could represent the voltage across the capacitor or the current flowing through the resistor, for example. A signal is mathematically expressed as a function of the independent variable t. t usually stands for time. Thus, a signal is denoted by x(t).

Continuous-time Signal

Continuous time considers variables to have a specific value for an infinitesimally short period of time. There are an endless number of points in time between any two points in time.

The variable "time" can span the full real number line or a subset of it, such as the non-negative reals, depending on the context. As a result, time is regarded as a continuous variable.

A continuous signal, also known as a continuous-time signal, is a variable quantity (a signal) with a continuous domain, which is generally time (e.g., a connected interval of the reals). That is, the domain of the function is an uncountable set. The functions themselves do not have to be continuous.

In short, a signal x(t) is a continuous-time signal if t is a continuous variable.

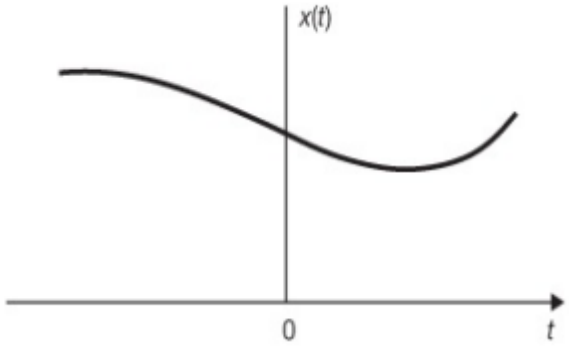


Fig. 1.1: Graphical representation of continuous-time signal

Basic Continuous-time Signal

A. Unit Step Function

The unit step function u(t) , also known as the *Heaviside unit function* , is defined as

$$u(t)=\begin{cases}1 & t>0\\ 0 & t<0\end{cases}$$

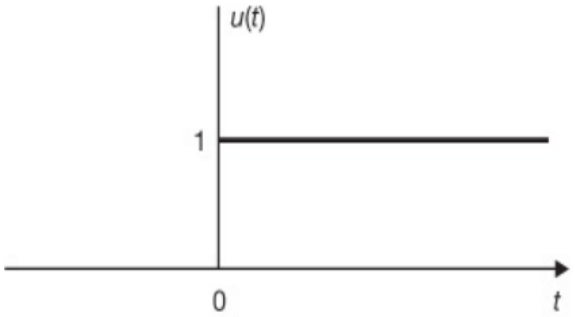
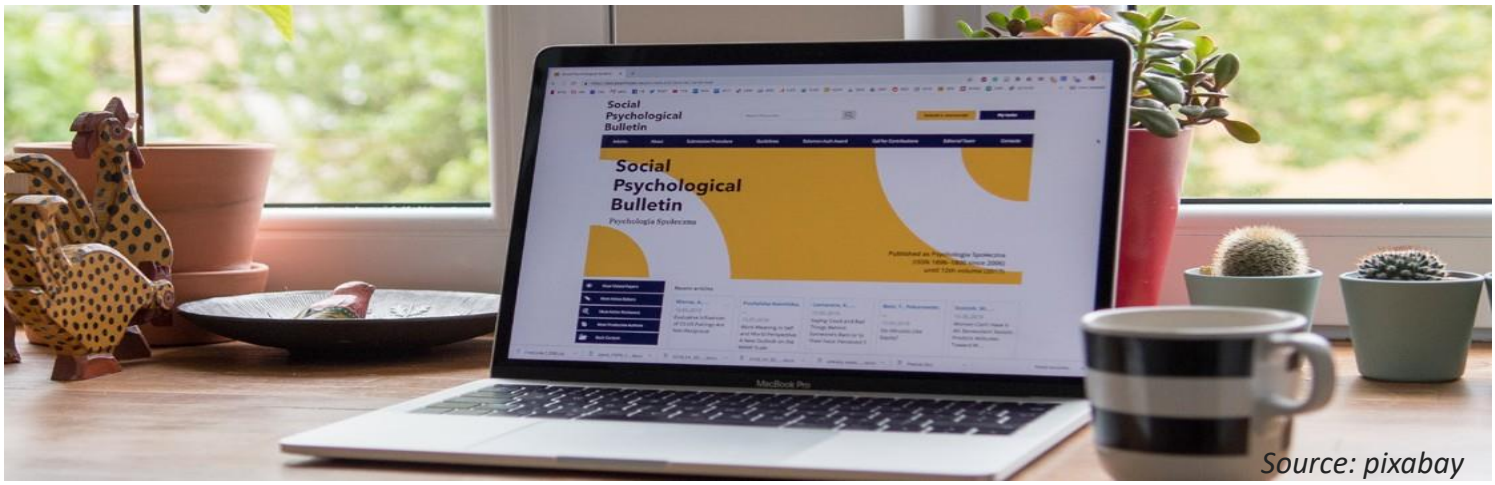


Fig. 1.2: Unit Step Function



Source: pixabay

Similarly, the shifted unit step function $u(t - t_0)$ is defined as

$$u(t-t_0)=\begin{cases}1 & t>t_0\\ 0 & t<t_0\end{cases}$$

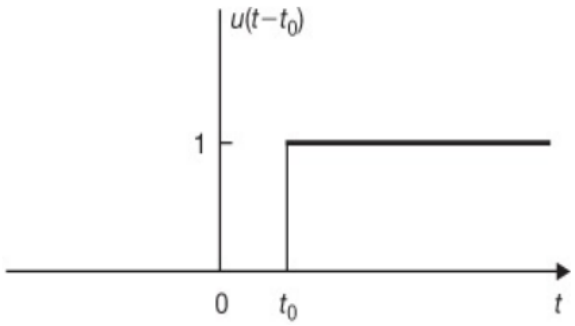


Fig. 1.3: Shifted Unit Step Function

B. Unit Impulse Function

The unit impulse function $\delta(t)$, also known as the Dirac delta function, plays a central role in system analysis.

$$\delta(t) = \begin{cases}1 & ; t = 0\\ 0 & ; t \neq 0\end{cases}$$

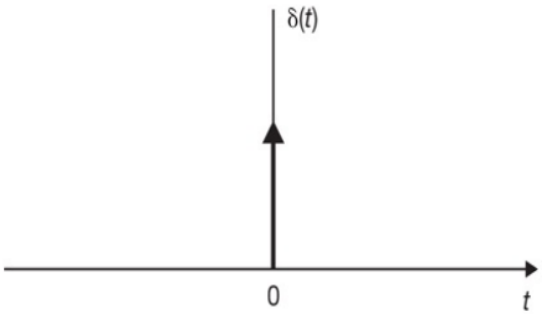


Fig. 1.4: Unit impulse function

Similarly, the delayed delta function $\delta(t - t0)$ is shown in Fig.1.5

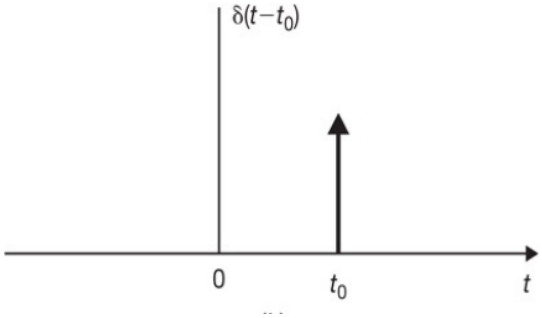


Fig. 1.5: Shifted Unit impulse function

C. Sinusoidal Signals

A continuous-time sinusoidal signal can be expressed as

$$x(t) = A \cos(\omega_0 t + \theta)$$

where

A is the amplitude (real)
ω_0 is the radian frequency in radians per second
θ is the phase angle in radians.

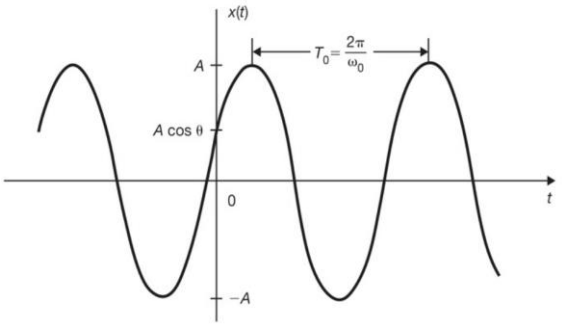


Fig. 1.6 : Continuous-time sinusoidal signal.

Discrete-time Signal

A discrete signal, also known as a discrete-time signal, is a time series made up of a succession of values. A discrete-time signal, unlike a continuous-time signal, does not have a continuous argument; nonetheless, it can be created by sampling from a continuous-time signal.

A sample rate is assigned to a discrete-time signal acquired by sampling a sequence at evenly spaced periods.

Discrete-time signals can come from a variety of places, but they usually fall into one of two categories:

- By obtaining analogue signal values at a fixed or variable pace. This is referred to as sampling.
- By looking at a process that is intrinsically discrete in time, such as the weekly peak value of a specific economic indicator.

In short, discrete-time signal x[n] may be obtained by sampling a continuous-time signal x(t). Since a discrete-time signal is defined at discrete times, a discrete-time signal is often identified as a sequence of number denoted by x[n], where n is integer.

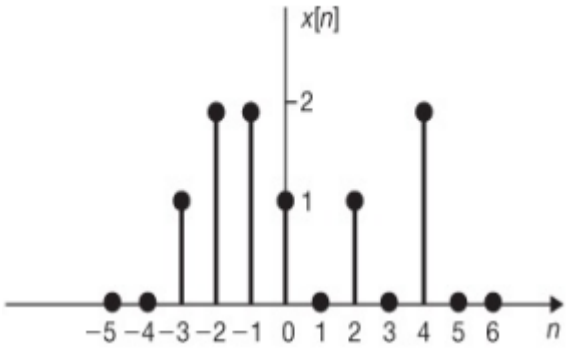
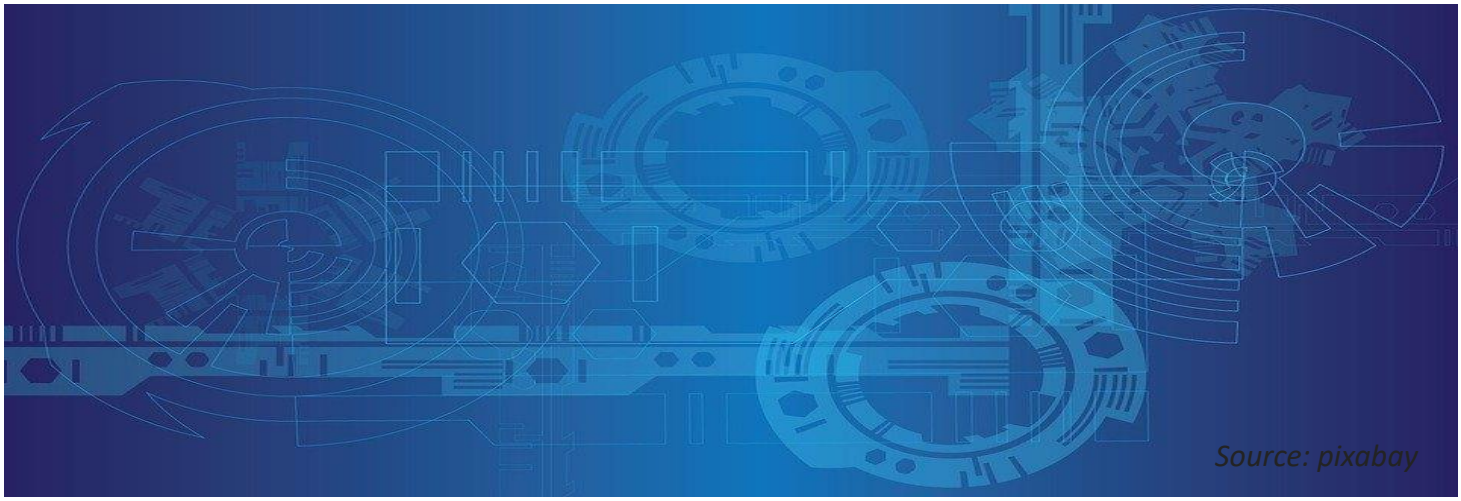


Fig. 1.7: Graphical representation of discrete-time signal



Basic Discrete-time Signal

A. Unit Step Sequence

The unit step sequence u[n] is defined as

$$u[n]=\begin{cases}1 & n\geq 0\\ 0 & n<0\end{cases}$$

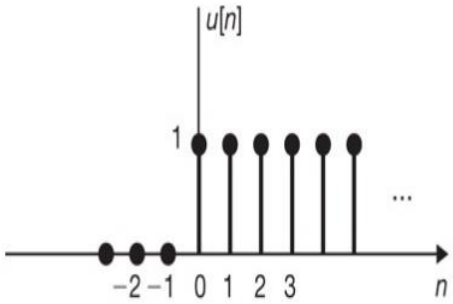


Fig. 1.8: Unit step sequence

Similarly, the shifted unit step sequence u[n – k] is defined as

$$u[n-k]=\begin{cases}1 & n\geq k\\ 0 & n<k\end{cases}$$

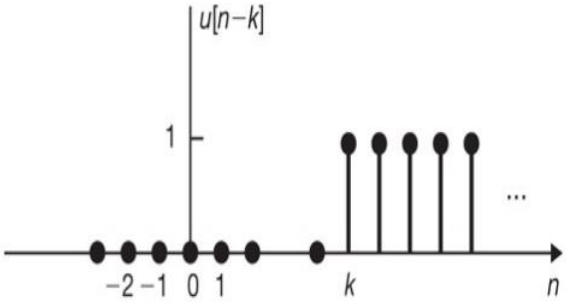


Fig. 1.9: Shifted unit step sequence

In short, a signal x(t) is a continuous-time signal if t is a continuous variable.

B. Unit Impulse Sequence

The unit impulse (or unit sample) sequence δ[n] is defined as

$$\delta[n]=\begin{cases}1 & n=0\\ 0 & n\neq 0\end{cases}$$

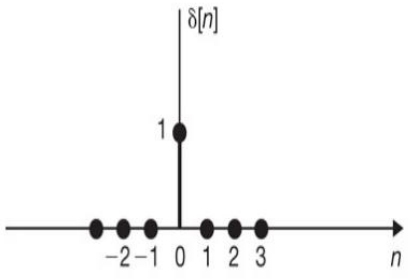


Fig. 1.10 : Unit impulse sequence

Similarly, the shifted unit impulse (or sample) sequence δ[n – k] is defined as

$$\delta[n-k]=\begin{cases}1 & n=k\\ 0 & n\neq k\end{cases}$$

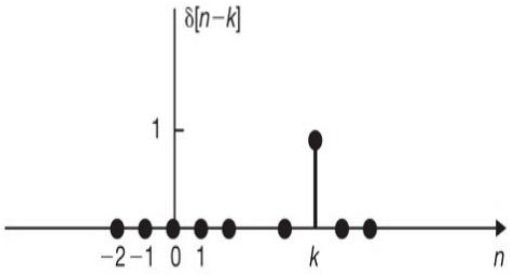


Fig. 1.11 : Shifted unit impulse sequence

C. Sinusoidal Sequence

A sinusoidal sequence can be expressed as

$$x[n] = A\cos(\Omega_o n + \theta)$$

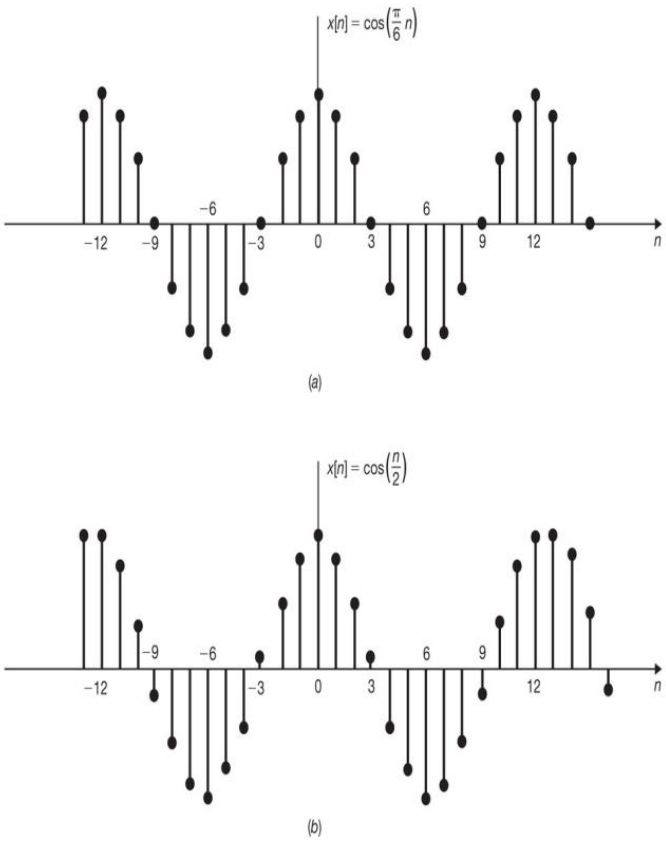


Fig. 1.12: Example of sinusoidal sequence

Simulation

Students can learn how to utilize the octave-online software in order to create numerous waveforms that are commonly used for continuous time signal processing and discrete time signal processing. GNU Octave, an open-source alternative to MATLAB, has a web interface called Octave Online.

Basic Continuous-time Signal

A. Unit Step Function

The unit steps are already incorporated into GNU Octave. The names of the mathematicians who utilized them in their work are used to identify them. *Heaviside(t)* is the name of the unit step function. The example below shows how they can be used.

```
t=-20:0.001:20
y=heaviside(t)          % Create unit step function
plot(t,y)               % Plot unit step function
axis([-10 10 -2 2])     % Define axis margin
```

GNU Octave displays the unit step function waveform as shown in Fig. 1.13 where it satisfies the definition of unit step function as

$$u(t)=\begin{cases}1 & t>0\\ 0 & t<0\end{cases}$$

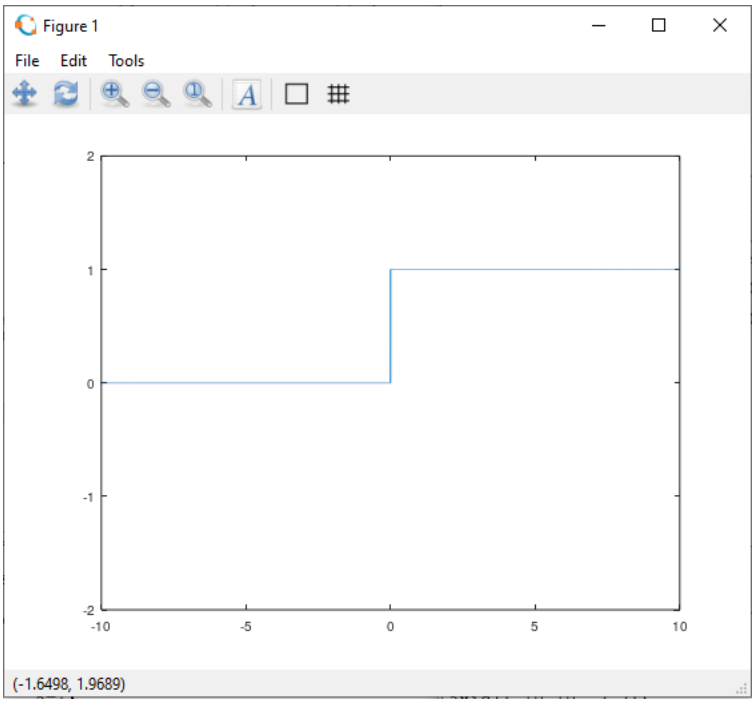


Fig. 1.13: Unit step function waveform

B. Unit Impulse Function

Similarly , the unit impulse function waveform is illustrated using the below example :

```
t=-10:10          % Set column
figure(1)         % Create a new figure window for plotting
x=[t==0]          % Set impulse at t=0
plot(t,x)         % Create simple x-y plots
```

GNU Octave display unit impulse function waveform as shown in Fig. 1.14 where it satisfies the definition of unit impulse function as

$$\delta[n]=\begin{cases}1 & n=0\\ 0 & n\neq 0\end{cases}$$

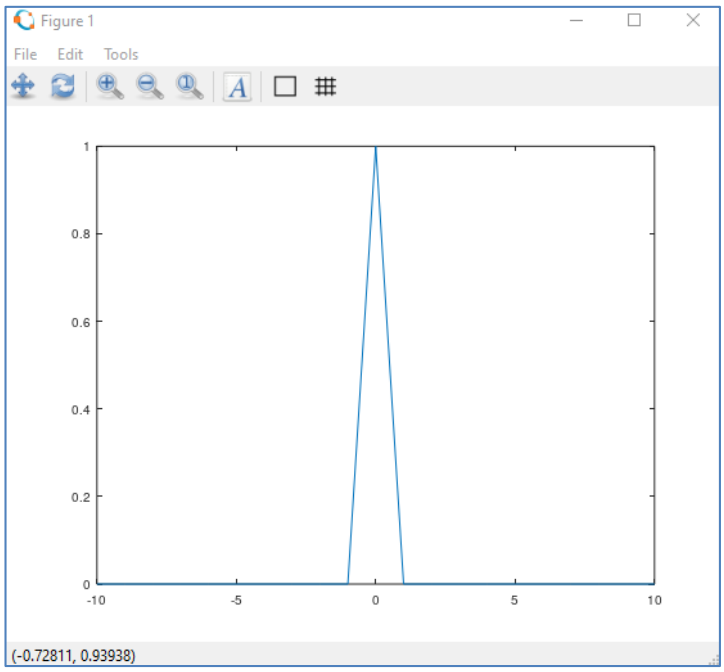


Fig. 1.14: Unit step function waveform

C. Sinusoidal Signals

The continuous-time sinusoidal waveform signals are illustrated using the following array commands

```
t=0:0.01:2;                          % set axis
f=2;                                 % frequency
a=1;                                 % amplitude
y=a*sin(2*pi*f*t);                   % define y as sinusoidal function
subplot(2,1,1);                      % define subplot
plot(t,y)                            % plot waveform
xlabel('t');                         % x axis title
ylabel('y(t)');                      % y axis title
title('Sinusoidal C-T Signal')       % plot title
```

GNU Octave displays continuous-time sinusoidal signal waveform as shown in Fig. 1.15.

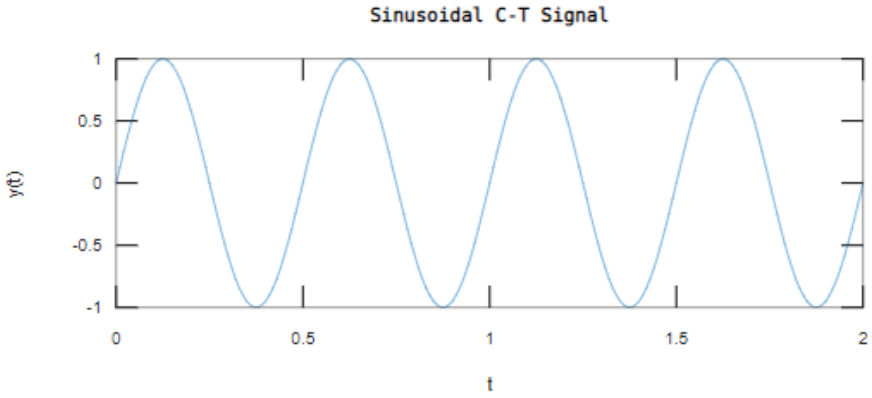


Fig. 1.15: Sinusoidal signal

Basic Discrete-time Signal

A. Unit Step Sequence

The unit step sequence waveform signals are illustrated using the following array commands

```
clc                                 % Clear the terminal screen and move the cursor to
                                      the upper left corner.
clear all                           % Clear all local and global user-defined variables
                                      and all functions from the symbol table.
n = -5 : 5;                         % set n values
y = heaviside(n);                   % unit step
stem(n, y)                          % plot unit step
axis([-5 5 -1 2])                   % set axis
xlabel('n');                        % set x axis title
ylabel('u[n]');                     % set y axis title

title({' Discrete Unit Step Function'})          % set waveform title
```

GNU Octave displays the unit step sequence waveform as shown in Fig. 1.16 where it satisfies the definition of unit step sequence as

$$u[n]=\begin{cases}1 & n\geq 0\\ 0 & n<0\end{cases}$$

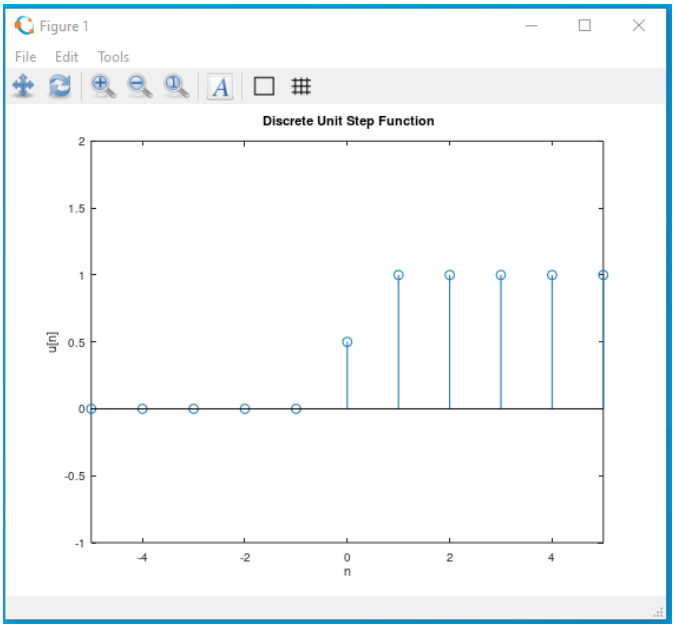


Fig. 1.16: Unit step sequence waveform

B. Unit Impulse Sequence

Similarly , the unit impulse sequence waveform is illustrated using below example :

```
clc                          % clear the terminal screen and move the cursor to
                               the upper left corner.
n=-4:4;                      % set n values
delta_n=[0,0,0,0,1,0,0,0,0]  % set unit impulse sequence
stem(n,delta_n)              % plot
```

GNU Octave displays the unit impulse sequence waveform as shown in Fig. 1.17 where it satisfies the definition of unit impulse sequence as

$$\delta[n]=\begin{cases}1 & n=0\\ 0 & n\neq 0\end{cases}$$

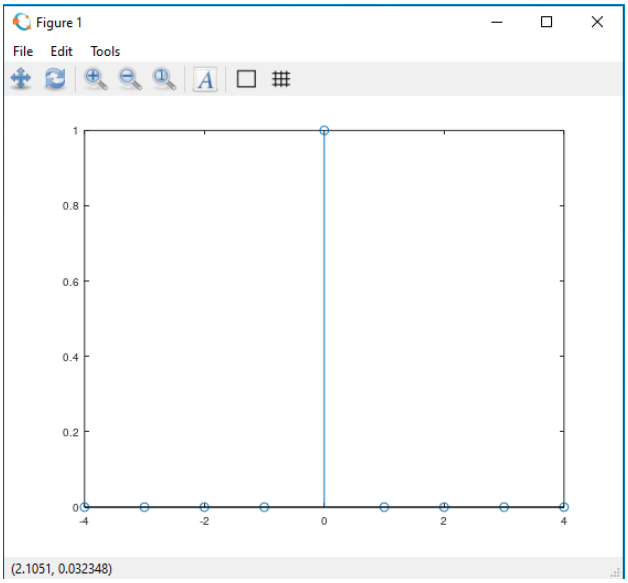


Fig. 1.17 : Unit impulse sequence waveform

C. Sinusoidal Sequence

The discrete-time sinusoidal waveform is illustrated using the following array commands

```
clc                                   % clear the terminal screen
t=0:0.05:2;                           % set axis
f=2;                                  % frequency
a=1;                                  % amplitude
y=a*sin(2*pi*f*t);                    % define y as sinusoidal sequence
subplot(2,1,1);                       % define subplot
stem(t,y)                             % plot waveform
xlabel('n');                          % x axis title
ylabel('y[n]');                       % y axis title
title({' Discrete-Time Sinusoidal waveform signal '})        % plot title
```

GNU Octave displays the discrete-time sinusoidal waveform as shown in Fig. 1.18.

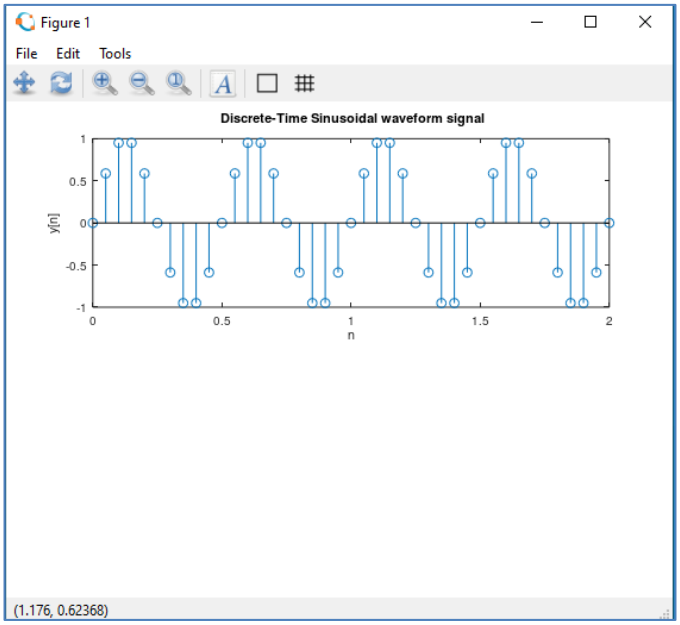


Fig. 1.18 : Discrete-time sinusoidal waveform

Source: pixabay

Exercises

1. Write the discrete-time cosine array commands . Use subplot (2,1,2). Label for each axis and name the title of the graph.

2. Write the array commands for Impulse Shifted Function when δ (n- 2). Label for each axis and name the title of the graph.

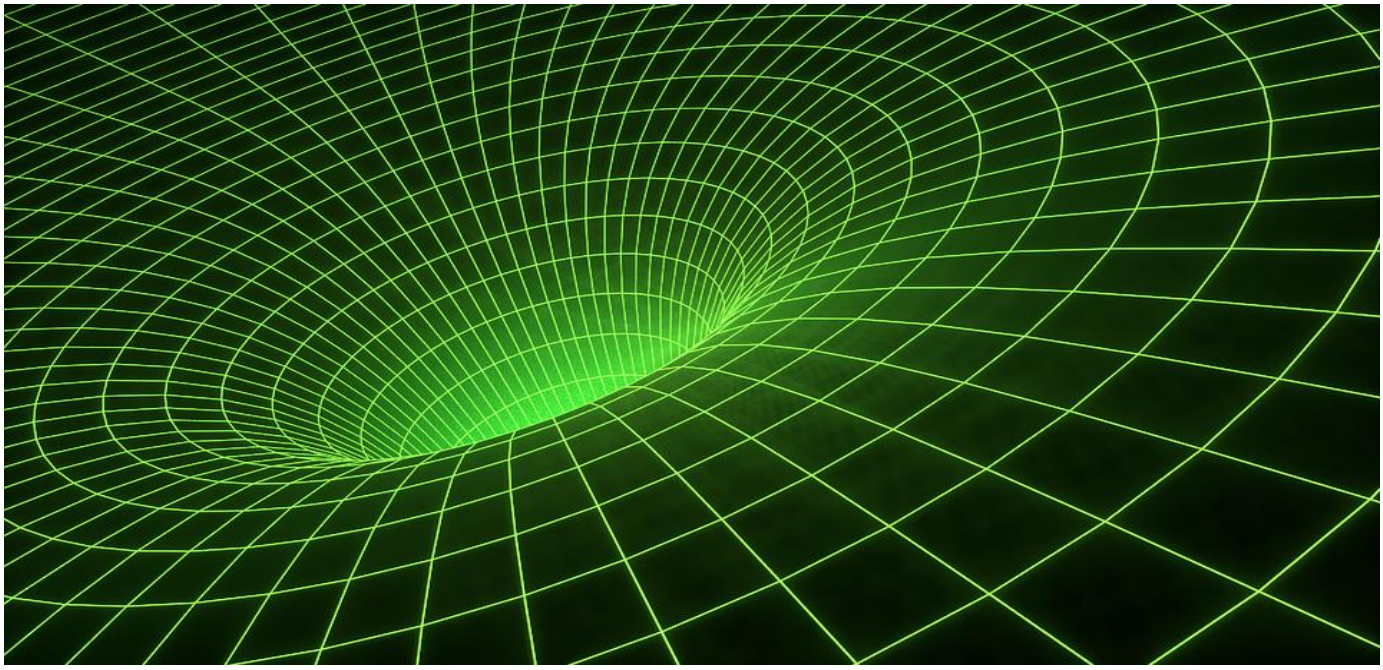
3. Write the waveform command for Step Shifted Function when u (n + 2). Label for each axis and name the title of the graph.

02
Linear Time-invariant (LTI) Systems

Linear Time-invariant (LTI) Systems

Linearity and time-invariance are two of the most significant characteristics of systems. The fundamental input-output relationship for systems with these features is developed in this chapter. The input-output relationship for LTI systems will be shown as to be defined by a convolution operation.

The significance of the convolution operation in LTI systems arises from the fact that understanding an LTI system's reaction to the unit impulse input helps us to identify its output to any input signals.



Source: pixabay

Continuous-Time LTI System and Convolution Integral

Convolution of two continuous-time signals x(t) and h(t) is denoted by

$$y(t) = x(t)*h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

It is commonly called the convolution integral. The convolution integral operation involves the following four steps:

a. The impulse response h(τ) is time-reversed (that is, reflected about the origin) to obtain h (–τ) and then shifted by t to form h(t –τ) = h[–(τ – t)], which is a function of τ with parameter t.
b. The signal x(τ) and h(t – τ) are multiplied together for all values of τ with t fixed at some values.
c. The product x(τ)h(t – τ) is integrated over all τ to produce a single output value y(t).
d. Steps 1 to 3 are repeated as t varies over – ∞ to ∞ to produce the entire output y(t).

Example:

Evaluate y(t) = x(t) * h(t), where x(t) and h(t) are shown in Fig. 2.1, by graphical method.

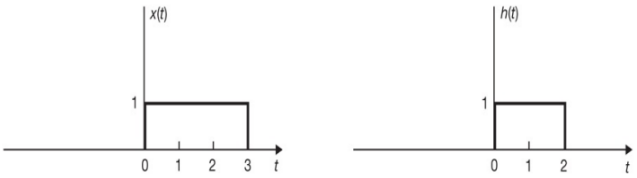
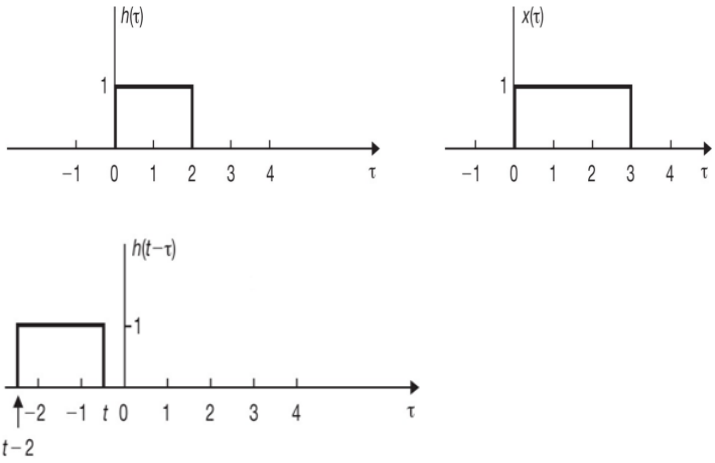


Fig. 2.1

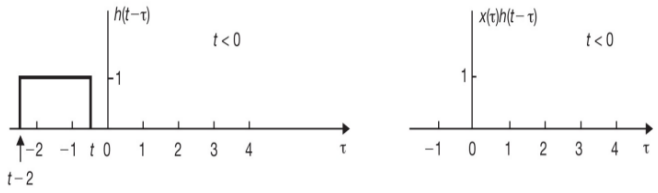
Solution :

Step 1



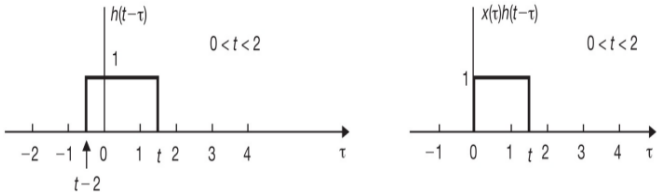
Step 2

For t < 0



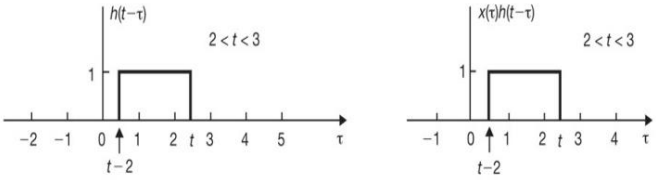
$$y(t) = 0$$

For 0 < t < 2



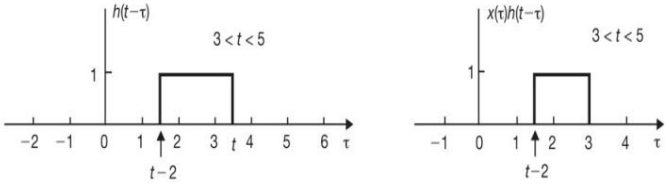
$$y(t) = \int_0^t d\tau = t$$

For 2 < t < 3



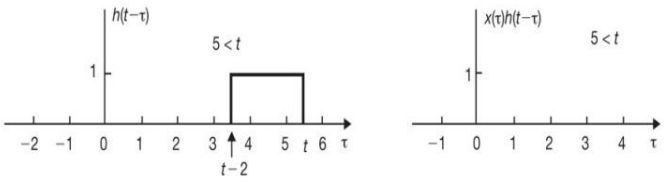
$$y(t) = \int_{t-2}^t d\tau = 2$$

For 3 < t < 5



$$y(t) = \int_{t-2}^3 d\tau = 5 - t$$

For 5 < t

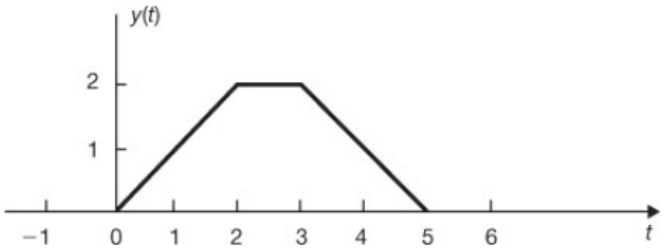


$$y(t) = 0$$

Step 3

Thus, we obtain

$$f(x) = \begin{cases} 0, & t < 0 \\ t, & 0 < t \leq 2 \\ 2, & 2 < t \leq 3 \\ 5 - t, & 3 < t \leq 5 \\ 0, & 5 < t \end{cases}$$



Discrete-Time LTI System and Convolution Sum

Convolution sum is the convolution of two sequences x[n] and h[n] which is denoted by

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

The convolution sum operation involves the following four steps:

a. The impulse response h[k] is time-reversed (that is, reflected about the origin) to obtain h[–k] and then shifted by n to form h[n – k] = h[– (k – n)], which is a function of k with parameter n.
b. Two sequences x[k] and h[n – k] are multiplied together for all values of k with n fixed at some values.
c. The product x[k]h[n – k] is summed over all k to produce a single output sample y[n].
d. Steps 1 to 3 are repeated as n varies over – ∞ to ∞ to produce the entire output y[n].

Example:

Evaluate y[n] = x[n] * h[n], where x[n] and h[n] are shown in Fig.2.2 by a graphical method.

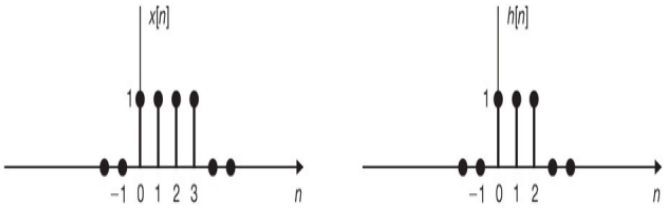
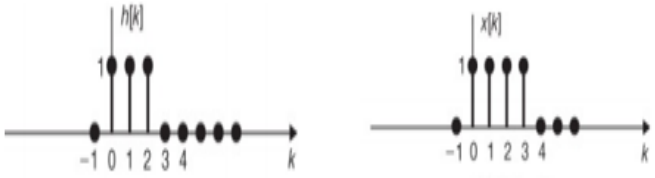


Fig. 2.2

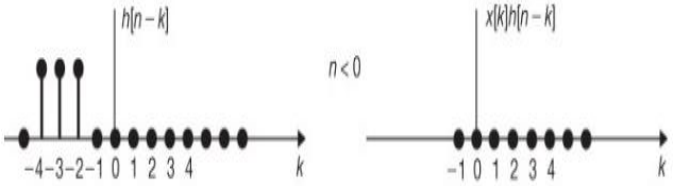
Solution :

Step 1

Sketch the sequences h[k], x[k] and h[n – k], x[k] h[n – k] for different values of n



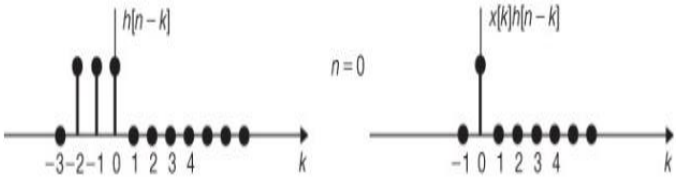
For n < 0



$$y[<0] = 0$$

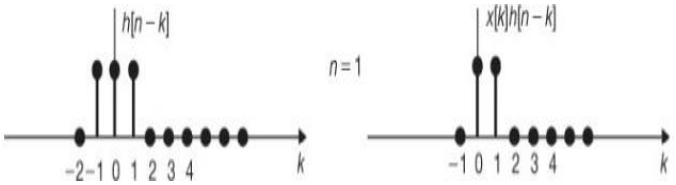
Discrete-Time LTI System and Convolution Sum

For n = 0



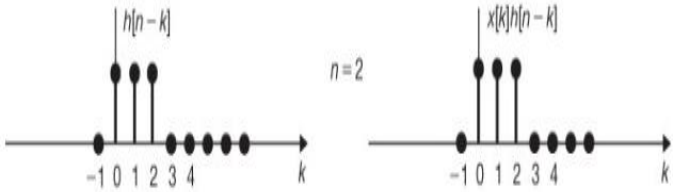
y[0] = 1

For n = 1



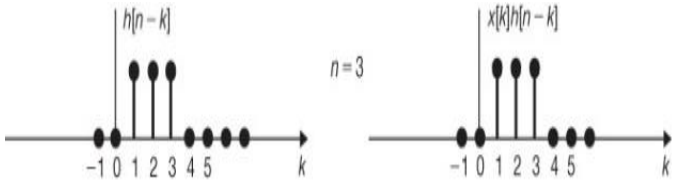
y[1] = 1 + 1 = 2

For n = 2



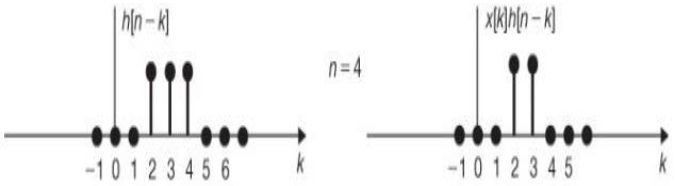
y[2] = 1 + 1 + 1 = 3

For n = 3



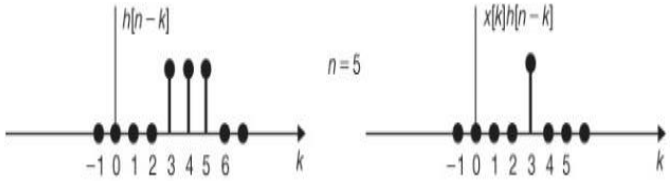
y[3] = 1 + 1 + 1 = 3

For n = 4



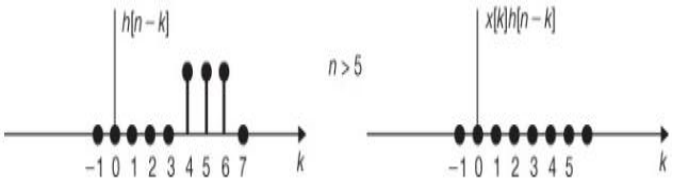
y[4] = 1 + 1 = 2

For n = 5



y[5] = 1

For n > 5



y[>5] = 0

Discrete-Time LTI System and Convolution Sum

Step 2

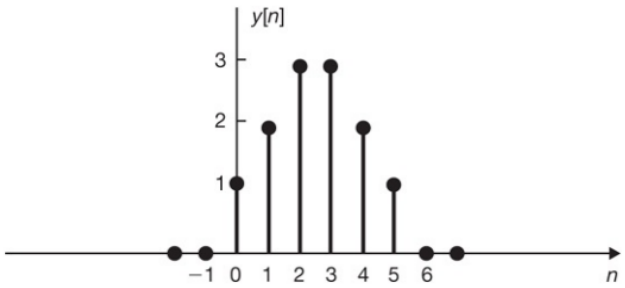
Thus, summing x[k] h[n – k] for 0 ≤ n ≤ 5, we obtain

y[0] = 1 y[1] = 2 y[2] = 3

y[3] = 3 y[4] = 2 y[5] = 1

or

y[n] = {1,2,3,3,2,1}



Simulation

Students can learn how to utilize octave-online software to write and execute convolution integral signals problem as well as convolution sum signals problem.

Convolution Integral using Heaviside Commands

We are utilizing the *heaviside(t)* command in GNU Octave to perform convolution integral simulation. To simulate y(t) = x(t) * h(t), two unit step function are used as shown in Fig. 2.3. The example of command array given shows how the convolution integral is done .

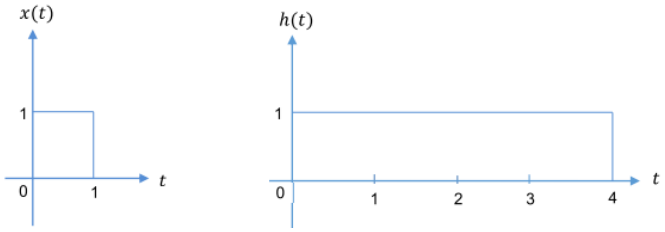


Fig. 2.3 : Input x(t) and impulse response h(t)

```
% set x(t)
t=-5:0.01:5;
x=heaviside(t)-heaviside(t-1);
subplot(3,1,1);
plot (t,x);
axis([-2 5 -1 1.5]);
xlabel('t'); ylabel('x(t)=u(t)-u(t-1)');
title({'Convolution integral y(t)=x(t)*h(t)'})
% set h(t)
h=heaviside(t)- heaviside(t-4);
subplot(3,1,2);
plot (t,h);
axis([-2 5 -1 1.5]);
xlabel('t'); ylabel('h(t)=u(t)-u(t-4)');
% perform convolution of x(t)*h(t)
m=conv(x,h); subplot(3,1,3); plot (m);
axis([0 2000 -100 150]); xlabel('t'); ylabel('y(t)');
```

GNU Octave displays the continuous-time input x(t), impulse response h(t) and convolution output y(t) as shown in Fig. 2.4.

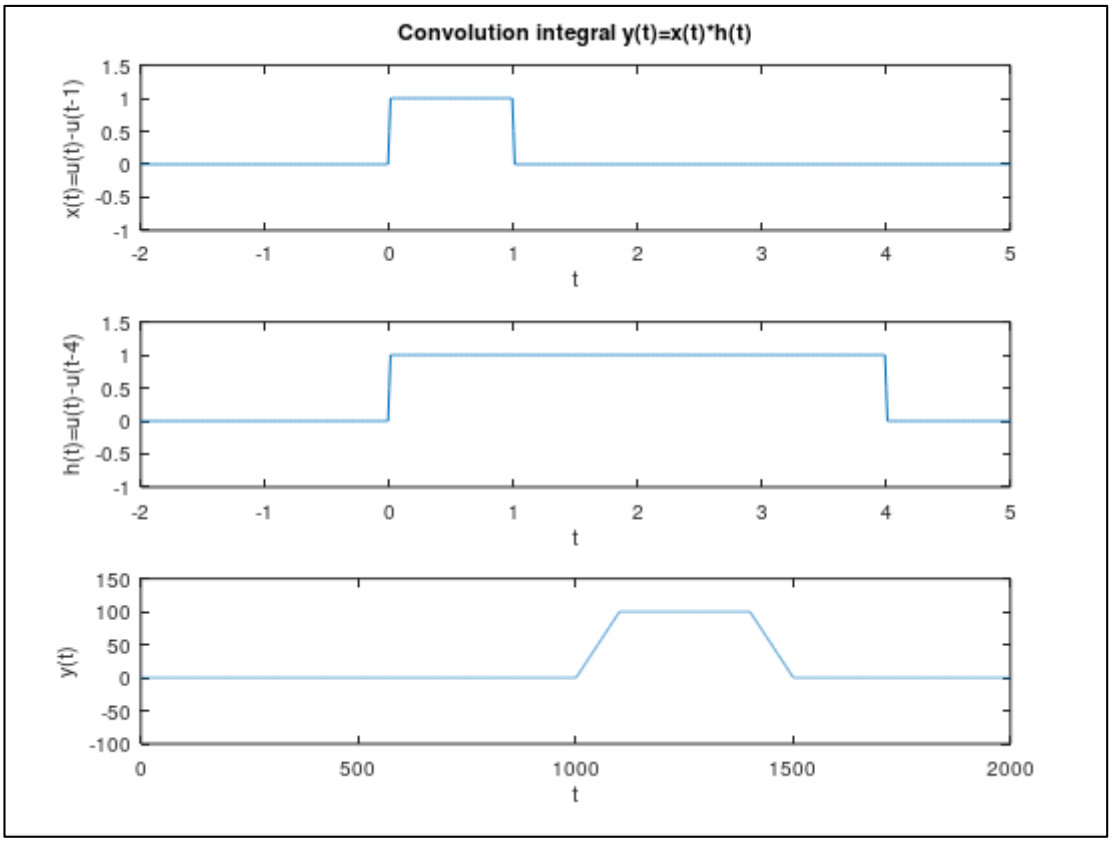


Fig. 2.4: Continuous-time input x(t), impulse response h(t) and convolution output y(t)

Convolution Sum

The convolution operation in GNU Octave is performed by using *conv* command. Similar to convolution integral, convolution sum is perform on the input x[n] and impulse response h[n]. The output is denoted by y[n]. The simulation on convolution sum of x[n] and h[n] can be achieved by using the following command array example. Note that the bold sequence is referring to n=0.

x[n] = $\left\{0, \frac{1}{3}, \frac{2}{3}, \mathbf{1}, \frac{4}{3}, \frac{5}{3}, 2\right\}$

h[n] = {1, 1, **1** ,1 , 1}

```
clear all
%input signal
n=0:6;
xn=0:1/3:2;
figure, stem(n,xn)
grid
xlabel('n')
ylabel('x[n]')
title('Input signal')

%impulse response
h1=-2;xh1=1;
h2=-1;xh2=1;
h3=0;xh3=1;
h4=1;xh4=1;
h5=2;xh5=1;
h=[h1,h2,h3,h4,h5];
hn=[xh1,xh2,xh3,xh4,xh5];
figure, stem(h,hn)
grid
xlabel('n')
ylabel('h[n]')
title('Impulse Response')
```

```
%output signal
yn=-2:8;
y=conv(hn,xn);
figure, stem(yn,y)
grid
xlabel('n')
ylabel('y[n]')
title('Output 1 of Convolution Sum')
```

GNU Octave displays the discrete-time input x[n], impulse response h[n] and convolution output y[n] as shown in Fig. 2.5 to Fig. 2.7.

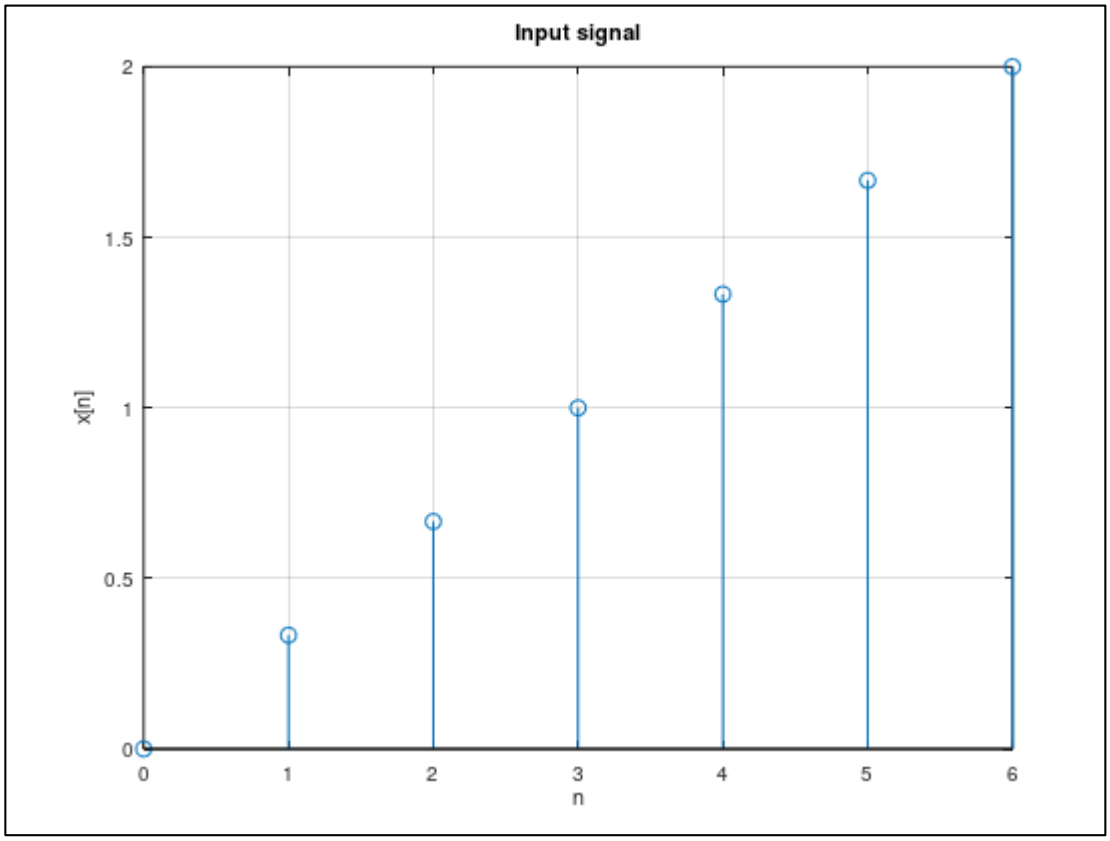


Fig. 2.5: Discrete-time input x[n]

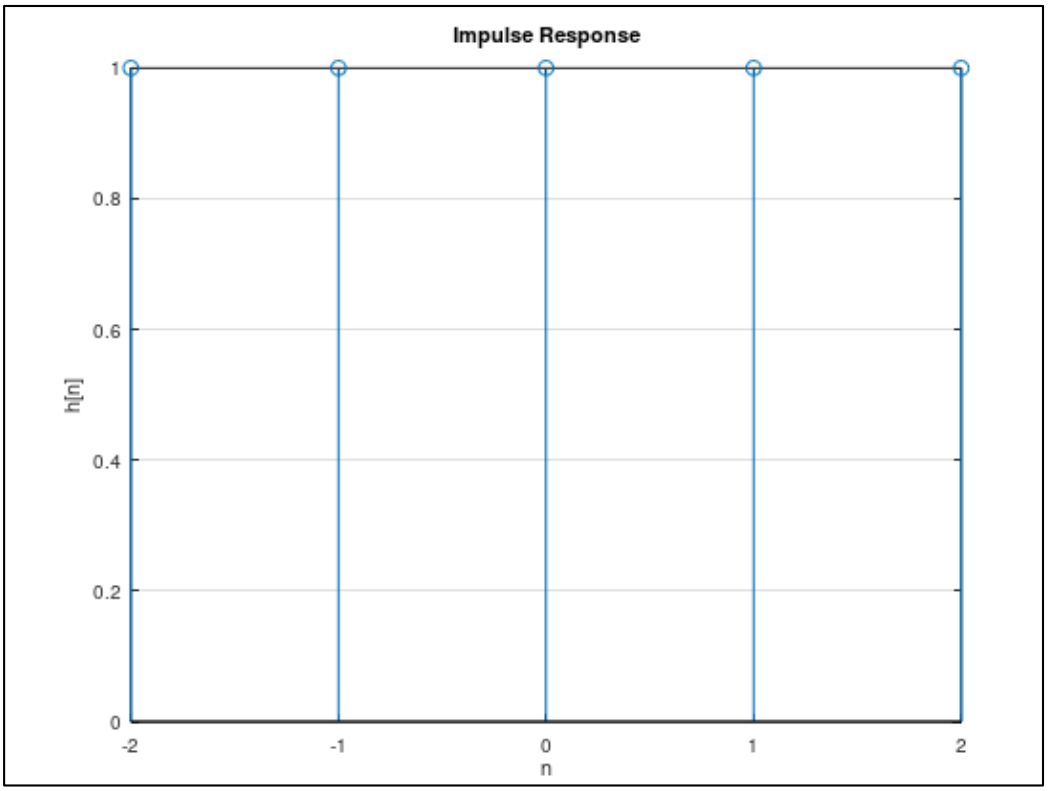


Fig. 2.6: Discrete-time impulse response h[n]

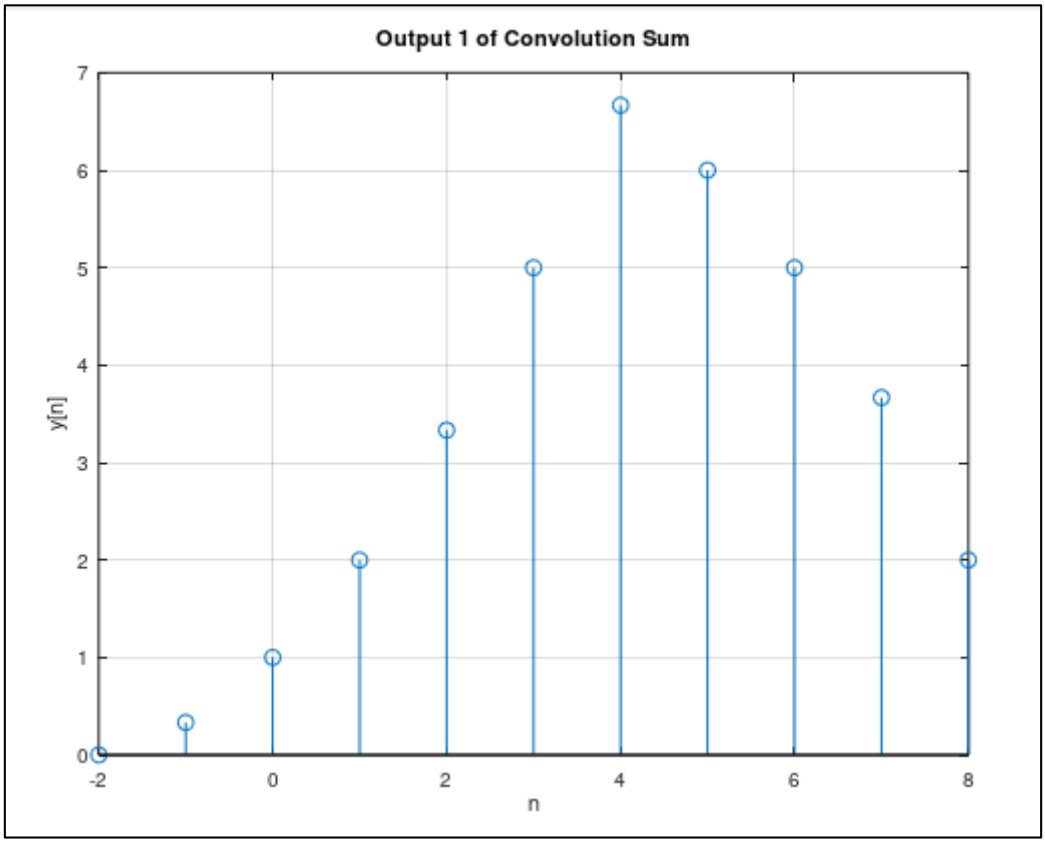
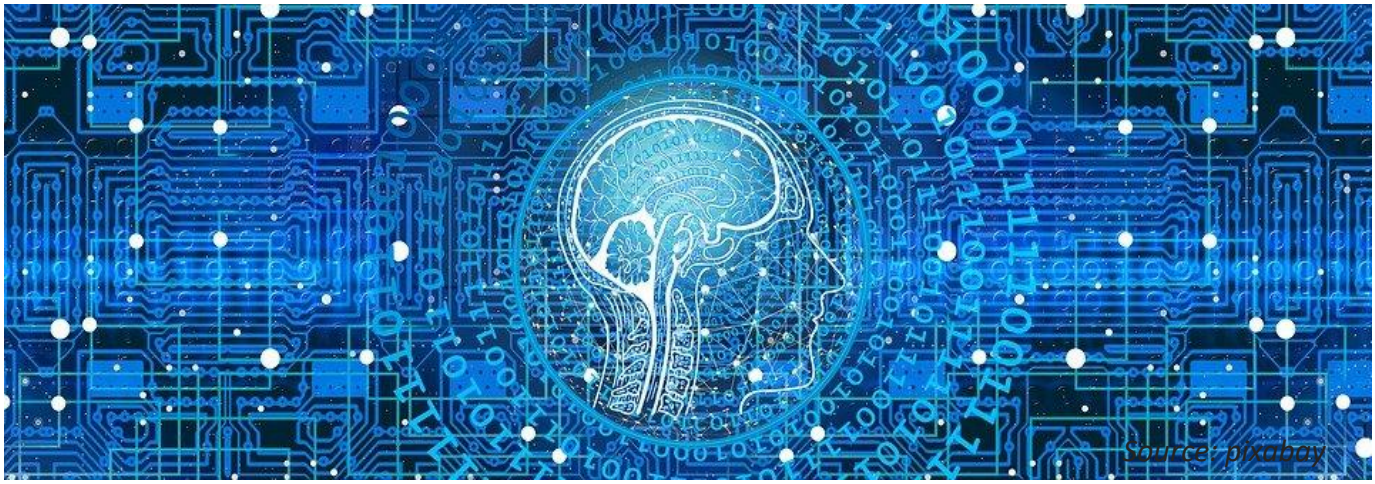


Fig. 2.7: Convolution Sum output y[n]

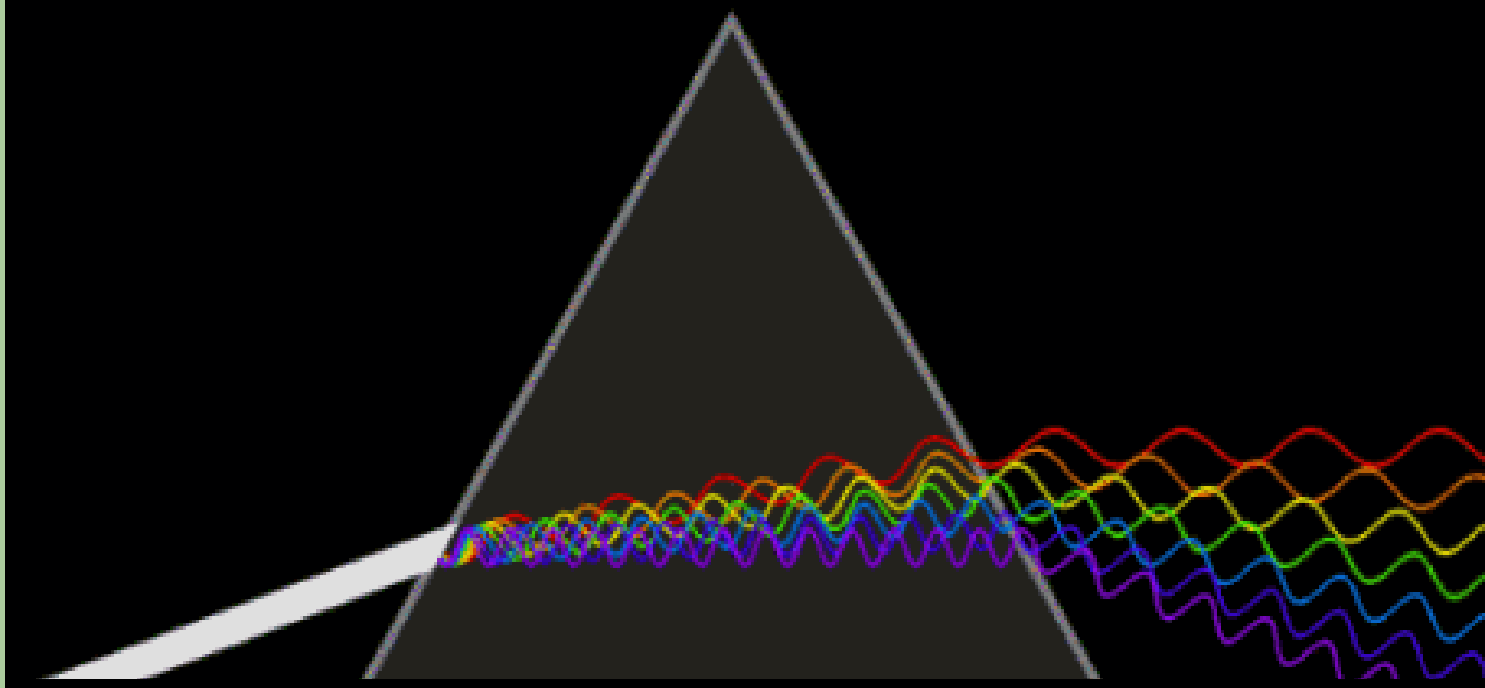


Exercises

1. Display the output graph of y(t) = x(t)*h(t) using convolution integral where x(t) = 3u(t) – 3u(t - 1) and h(t) = 2u(t) – 3u(t - 4) . Write and execute the command. Choose a suitable axis for all graphs x(t), h(t) and y(t).

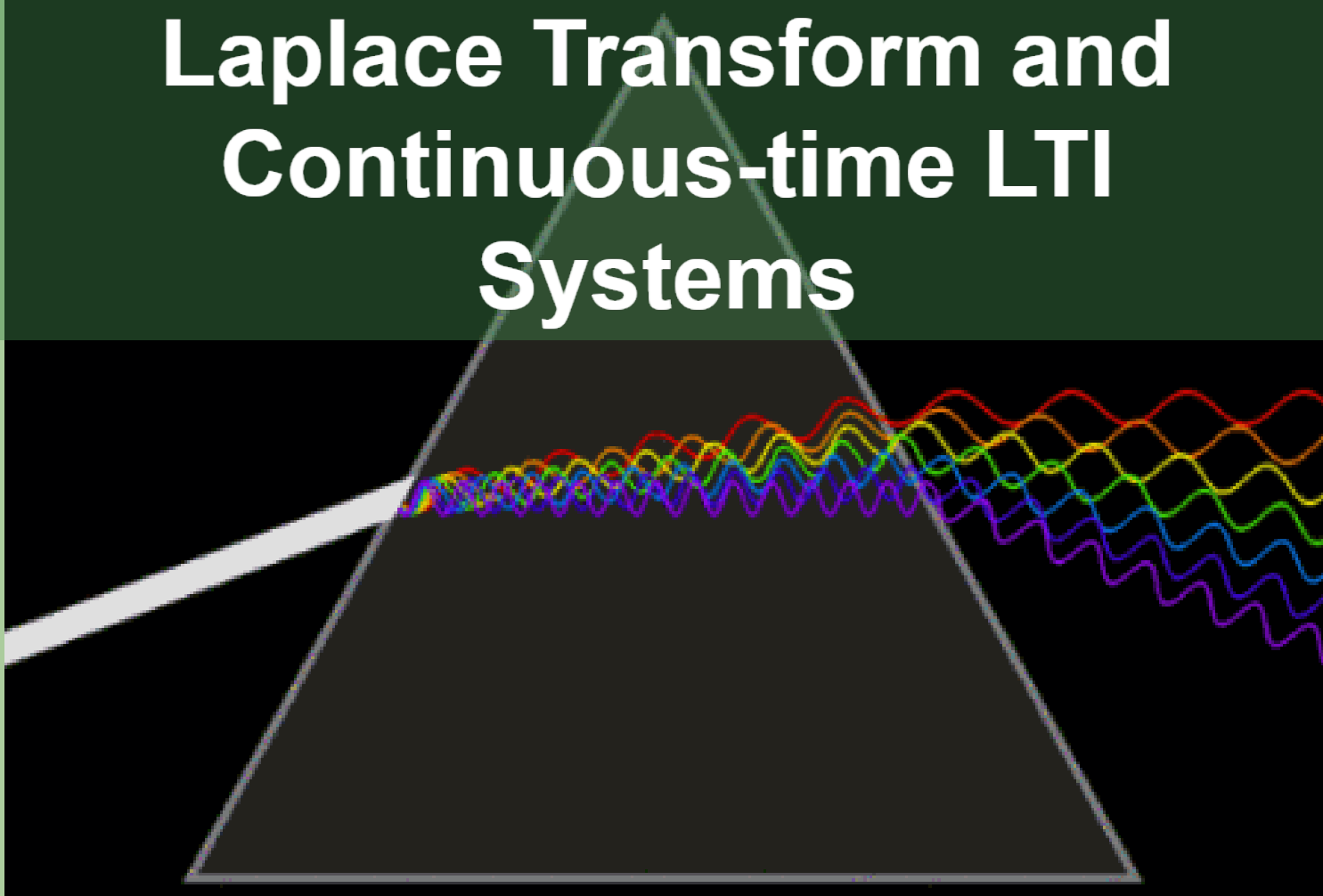
2. Write and compute the output of LTI systems for the following signal below for y[n]=x[n]*h[n] using octave online. Use the function *conv* or *[y=conv(h,x)]* to compute the problem in convolution sum.

   x[n] = {1,-1,2,**-1**,1}
   h[n] = {1,1,**1**,1}



03

Laplace Transform and Continuous-time LTI Systems



Laplace Transform and Continuous-time LTI Systems

Laplace Transform is to convert time-domain signals into complex s-domain representations in order to analyze and process the continuous-time signals and system.

The Laplace transform of a signal x(t) is represented by

$$X(s) = \int_{-\infty}^{+\infty} x(t)e^{-st}$$

A valuable tool for analyzing linear time-invariant systems is the Laplace transform. The Laplace transform can be written as a ratio of polynomials for a large class of signals.

These rational transforms appear as system functions for LTI systems that meet linear constant coefficient differential equations. The roots of polynomials N(s) and D(s), also known as zeros and poles, entirely determine rational transforms up to a scaling factor. As these roots are so crucial in studying LTI systems, it is indeed easier to visualize them on a pole-zero diagram.

$$X(s) = \frac{N(s)}{D(s)}$$

Source: pixabay

The Laplace transform, named after its inventor Pierre-Simon Laplace, is an integral transform in mathematics. It converts a real-valued function *t* (typically time) into a complex-valued function *s* (complex frequency).

In science and engineering, the transform has numerous uses. The Fourier transform and the Laplace transform are comparable. A Fourier transform is a complex function of a real variable (frequency), whereas a Laplace transform is a complex function of a complex variable.

In engineering and physics, the Laplace transform is used to calculate the output of a linear time-invariant system by convolving its unit impulse response with the input signal. When this calculation is done in Laplace space, the convolution becomes a multiplication, which is easier to solve due to its algebraic nature.

The Laplace transform, which is widely utilised in mechanical and electrical engineering, can also be used to solve differential equations. The Laplace transform converts a linear differential equation into an algebraic equation that can be solved using algebraic techniques. The inverse Laplace transform can then be used to solve the original differential equation.

Oliver Heaviside, an English electrical engineer, suggested a comparable technique without employing the Laplace transform and this results in an operational calculus is known as the Heaviside calculus.

Example:

Find the Laplace Transform and the ROC of X(t) = $e^{-at}\ u(t) + e^{at}\ u(-t)$

Solution :

From the Laplace Transform pairs table

x(t)	X(s)	ROC
$e^{-at}\ u(t)$	$\frac{1}{s+a}$	Re{s} > -a
$-e^{at}\ u(-t)$	$\frac{1}{s+a}$	Re{s} < -a

$L\{e^{-at}\ u(t)\} = \frac{1}{s+a}$; Re{s} > -a

$L\{e^{at}\ u(-t)\} = -\frac{1}{s-a}$; Re{s} < a

Therefore,

$X(s) = \frac{1}{s+a} - \frac{1}{s-a}$ $\quad -a < Re\{s\} < a$

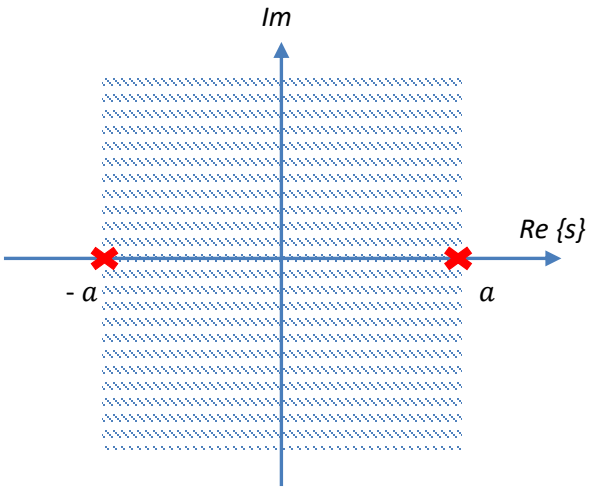
$= \frac{(s-a) - (s+a)}{(s+a)(s-a)}$

$= \frac{-2a}{(s+a)(s-a)}$

Thus,

Zeros : none
Poles : s = -a , s=a

The pole-zero diagram and ROC :



Example:

Find the inverse Laplace transform of the following X(s)

a) $X(s) = \frac{1}{s+1}$, Re{s} > -1

b) $X(s) = \frac{1}{s+1}$, Re{s} < -1

c) $X(s) = \frac{1}{s^2+4}$

d) $X(s) = \frac{1}{s+2} - \frac{1}{s+5} - \frac{2}{s+5}$

e) $X(s) = \frac{6}{s^2+36}$

Solution:

By referring to the Laplace Transform pairs table

a) $x(t) = L^{-1}\left\{\frac{1}{s+1}\right\} = \boldsymbol{e^{-t}\ u(t)}$

b) $x(t) = L^{-1}\left\{\frac{1}{s+1}\right\} = \boldsymbol{-e^{-t}\ u(-t)}$

c) $x(t) = L^{-1}\left\{\frac{1}{s^2+4}\right\}$

$= L^{-1}\left\{\frac{1}{s^2+2^2}\right\}$

= cos 2t u(t)

d) $x(t) = L^{-1}\left\{\frac{1}{s+2}\right\} - L^{-1}\left\{\frac{1}{s+5}\right\} - L^{-1}\left\{\frac{2}{s+5}\right\}$

$= \boldsymbol{e^{-2t}u(t) - e^{-5t}u(t) - 2e^{-5t}u(t)}$

e) $x(t) = L^{-1}\left\{\frac{6}{s^2+36}\right\}$

$= L^{-1}\left\{\frac{6}{s^2+6^2}\right\}$

= sin 6t u(t)

Simulation

GNU Octave is a high-level programming language that is mostly used for numerical calculations. Octave is a numerical solver for linear and nonlinear problems, as well as a language for executing various numerical experiments that is mainly compatible with MATLAB. Other characteristics of this software are as follows:

- Free software that runs on GNU/Linux, macOS, BSD, and Microsoft Windows with a powerful mathematics-oriented syntax and built-in 2D/3D charting and visualisation features.
- Compatible with a wide range of Matlab scripts.

GNU Octave is used to compute Laplace transform using the *laplace* function. The following example shows the program arrays. It is important for the user to download and install the recent symbolic bundle package from https://github.com/cbm755/octsympy/releases

Laplace Transform

For each of the given continuous-time signal, Laplace transform are computed using the following command array

$$x(t) = 4\sin(100t)u(t)$$

Program :

```
syms t s
x=4*sin(100*t)*heaviside(t)
X=laplace(x,t,s)
```

Output :

```
X = (sym)

    400
 ----------
 s^2 + 10000
```

Alternatively, a simpler programing is also applicable to perform Laplace transform as shown in below example.

$$x(t) = 3t^3 + 4e^{5t}$$

Program :

```
syms s t
laplace ((3*t^3)+(4*e^(5*t)))
```

Output :

```
     4          18
--------- + --
  /s     \     4
5*|- - 1|    s
  \5     /
```

Inverse Laplace Transforms

GNU Octave can also compute the inverse Laplace Transforms using function *ilaplace* as shown in the following example.

$$X(s) = \frac{10(s+1)}{s^2 + 4s + 3}$$

Program :

```
syms t s
x=10*(s+1)/(s^2+4*s+3)
ilaplace(x)
simplify(ans)
pretty(ans)
```

Output :

```
      -3*t
10*e
```

Poles and Zeros

The graphical representation of X(s) through its poles and zeros in the s-plane is referred to as the pole-zero plot of X(s). GNU Octave can compute the poles and zeros of any given continuous-time function and map the pole-zero plot as illustrated in the following example.

$$X(s) = \frac{10(s+1)}{s^2 + 4s + 3}$$

Program :

```
num=[10 10]
den=[1 4 3]
sys=tf(num, den)
pzmap(sys)
p,z]=pzmap(sys)
```

The GNU Octave mapped the output as illustrated in Fig. 3.1 with zeros at s=0 and two poles at s=-1 and s=-3

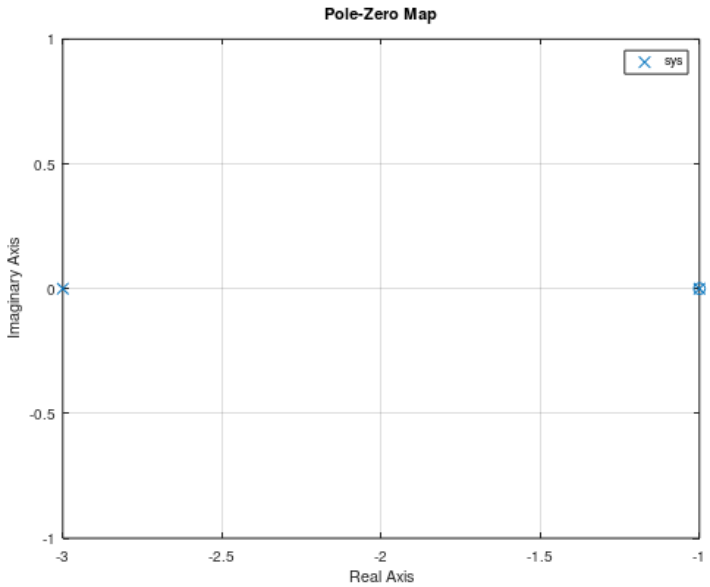


Fig. 3.1: pole-zero plot

Exercises

1. Compute the Laplace transform of the following functions. Write the program in GNU Octave using *laplace* command.

   a) $x(t) = e^{(-1/4)\,t}$

   b) $x(t) = 8 \cos(10t) \cdot 5u(2t)$

   c) $x(t) = 2 \cos 2t + 3 \sin 2t$

   d) $x(t) = e^{-t} \sin 5t + e^{-4t} \cos 3t$

2. Using GNU Octave, compute the inverse Laplace Transforms of the following functions.

   a) $X(s) = \frac{10(s+1)}{s^2+4s+8}$

   b) $X(s) = \frac{6}{s} + \frac{2s}{s^2+16} - \frac{4}{s^2}$

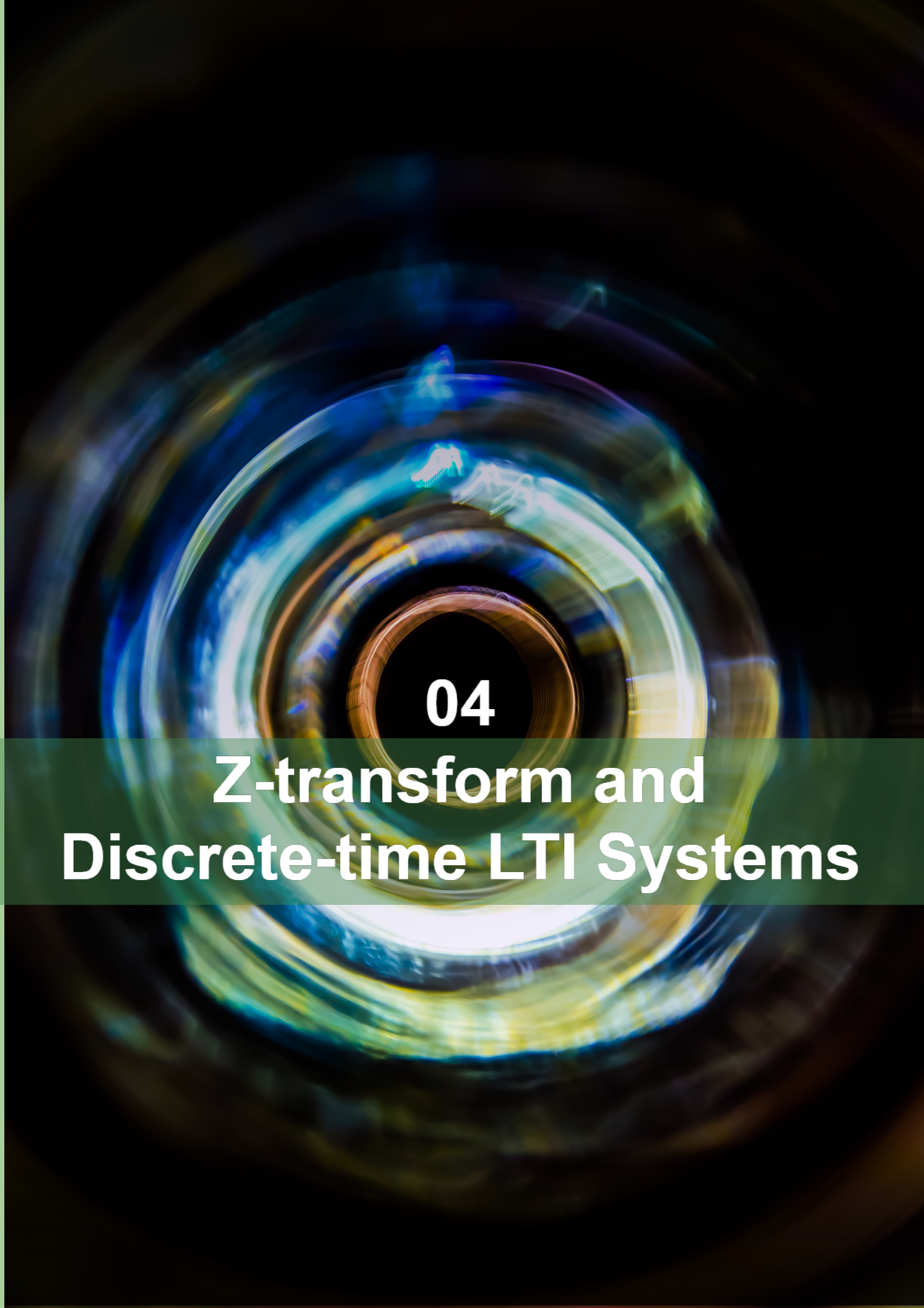
   c) $x(t) = 2 \cos 2t + 3 \sin 2t$

   d) $x(t) = e^{-t} \sin 5t + e^{-4t} \cos 3t$

3. Using GNU Octave, compute the poles and zeros of the following function.

a) $X(s) = \frac{10(s+1)}{s^2+4s+8}$

b) $X(s) = \frac{2s+100}{(s+1)(s+8)(s+10)}$



04
Z-transform and Discrete-time LTI Systems

Z-transform and Discrete-time LTI Systems

The Z-transform translates a discrete-time signal, which is a sequence of real or complex values, into a complex frequency-domain representation in mathematics and signal processing.

It is possible to specify the Z-transform as a one-sided or two-sided transform. The formal power series X[z] defines the bilateral or two-sided Z-transform of a discrete-time signal x[n].

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n}$$

The inverse Z-transform is the reverse of the Z-transform.

$$x[n] = \mathcal{Z}^{-1}\{X(z)\} = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz$$

where C is a counterclockwise closed path encircling the origin and entirely in the region of convergence (ROC). In the case where the ROC is causal, this means the path C must encircle all of the poles of X[z].

Source: pixabay

The graphical representation of X(z) through its poles and zeros in the z-plane is referred to as the pole-zero plot of X(z)

Example:

Find the Z-Transform and the ROC of x[n] = $(\frac{1}{2})^n u[n] + (-\frac{1}{3})^n u[n]$

Solution :

From the Z-Transform pairs table

x[n]	X(z)	ROC
$(\frac{1}{2})^n u[n$	$\frac{z}{z-\frac{1}{2}}$	$\|z\| > \|\frac{1}{2}\|$
$(-\frac{1}{3})^n u[n]$	$\frac{z}{z+\frac{1}{3}}$	$\|z\| > \|\frac{1}{3}\|$

$$Z\{(\tfrac{1}{2})^n u[n]\} = \tfrac{z}{z-\frac{1}{2}} \quad ; |z| > |\tfrac{1}{2}|$$

$$Z\{(-\tfrac{1}{3})^n u[n]\} = \tfrac{z}{z+\frac{1}{3}} \quad ; |z| > |\tfrac{1}{3}|$$

Therefore,

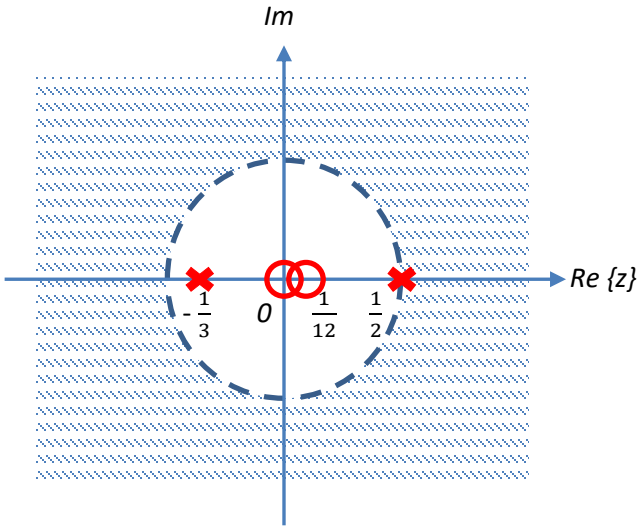
$$X(z) = \frac{z}{z-\frac{1}{2}} + \frac{z}{z+\frac{1}{3}}$$

$$= \frac{2z(z-\frac{1}{12})}{(z-\frac{1}{2})(z+\frac{1}{3})}$$

Zero : z = 0 , z= $\frac{1}{12}$

Poles : z= $\frac{1}{2}$, z= - $\frac{1}{3}$

The pole-zero plot of X(z) :



Simulation

WolframAlpha is a Wolfram Research computational knowledge engine and answer engine. It directly responds to factual enquiries by computing the answer using data from external sources.

The simulation for z-transform and inverse z-transform utilizes WolframAlpha online and GNU Octave . The WolframAlpha computes z-transform . From the result, the GNU Octave is used to compute the poles and zeros and map the pole and zeros plot.

Z-Transform

The WolframAlpha online widget for z-transform can be accessed online at the WolframAlpha website as shown in Fig. 4.1.

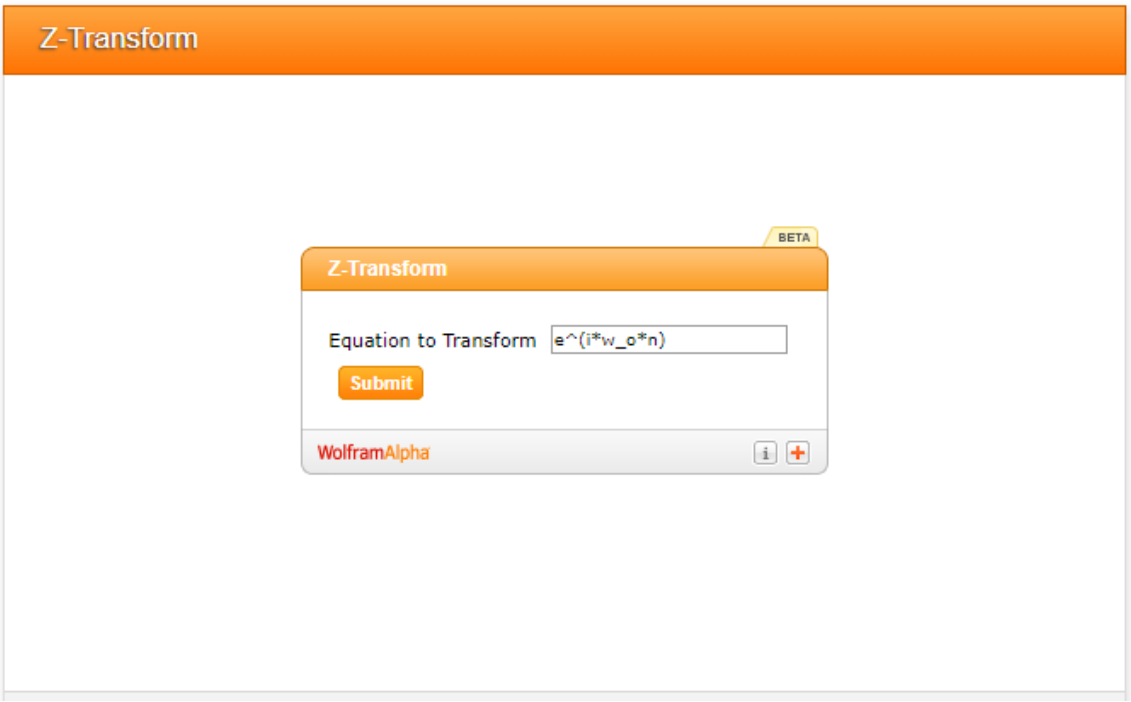


Fig. 4.1 WolframAlpha online widget for z-transform

In this simulation, we use the sequence

$$x[n] = (\frac{1}{2})^n u[n] + (\frac{1}{3})^n u[n]$$

Key in the given sequence in the widget by using the *heaviside* command as follows

(((1/2)^n)*heaviside(n)) + (((1/3)^n)*heaviside(n))

Fig. 4.2 shows the z-transform of the given sequence,

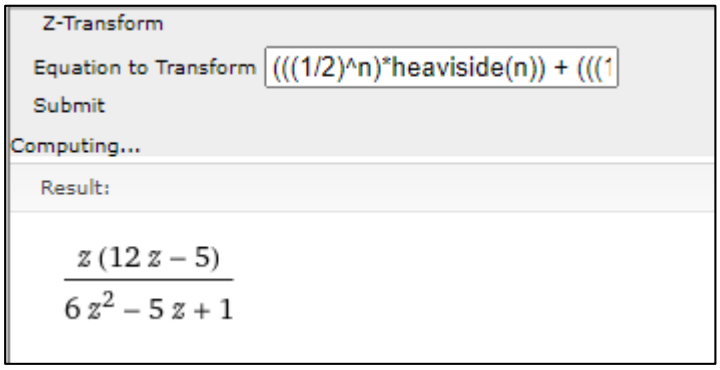


Fig. 4.2 The z-transform output

The GNU Octave code commands for the pole and zeros plot are as follows. Although there are two poles dan two zeros for the X(z), note that due to the programming contains, user can only plot one pole and zero at a time. The pole and zeros plot are shown in Fig. 4.3. User may use the same code command to plot the other pole and zero.

Program :

```
% zero and pole
b=[0]                %b=zeros
a=[1/3]              %a=poles
zplane(b,a)
title ("Zero pole plot");
```

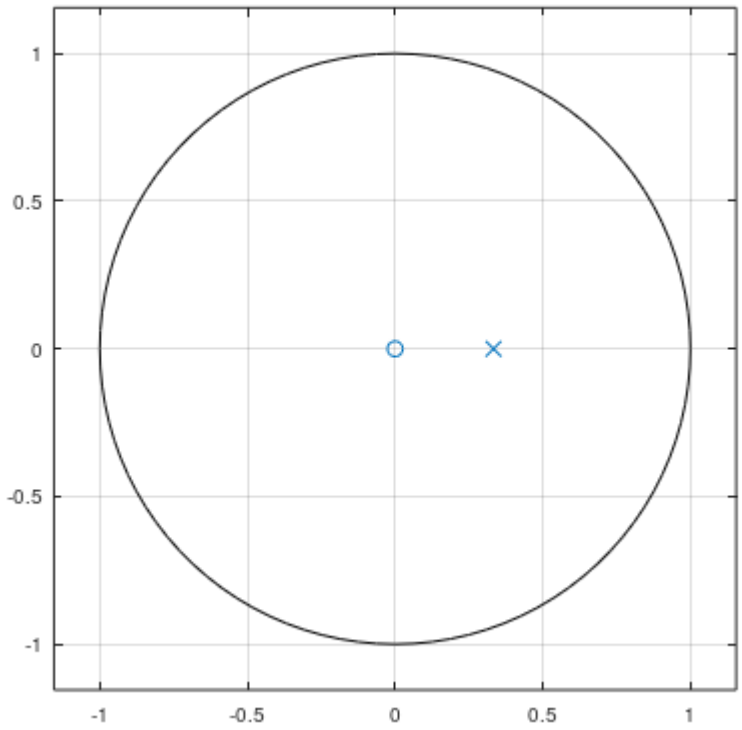


Fig.4.3 The pole and zero plot

Inverse z-Transform

The WolframAlpha online widget for inverse z-transform can be accessed online at WolframAlpha website as shown in Fig. 4.4.

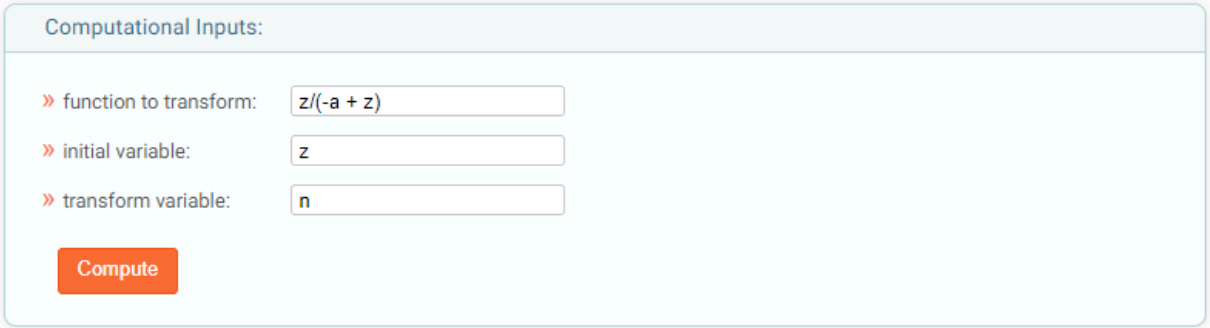


Fig. 4.4. WolframAlpha online widget for inverse z-transform

In this simulation, we use the sequence

$$X[z] = \frac{10z(z+5)}{(z-1)(z-2)(z+3)}$$

Key in the given sequence in the widget as follows

(10z(z+5))/((z-1)(z-2)(z+3))

The WolframAlpha displays the inverse z-transform as shown in Fig. 4.5.

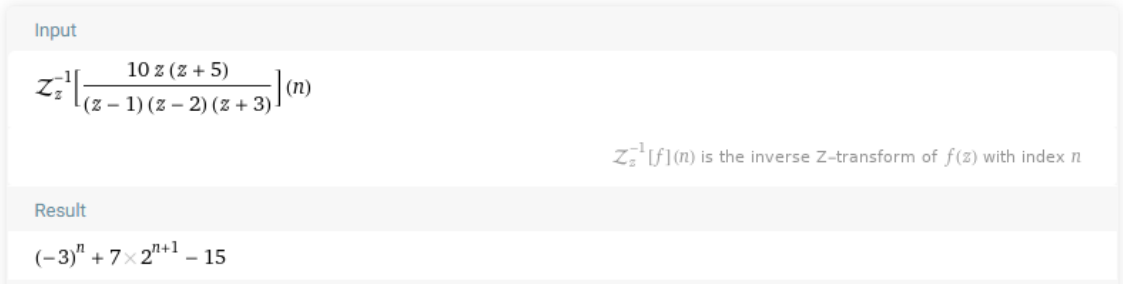
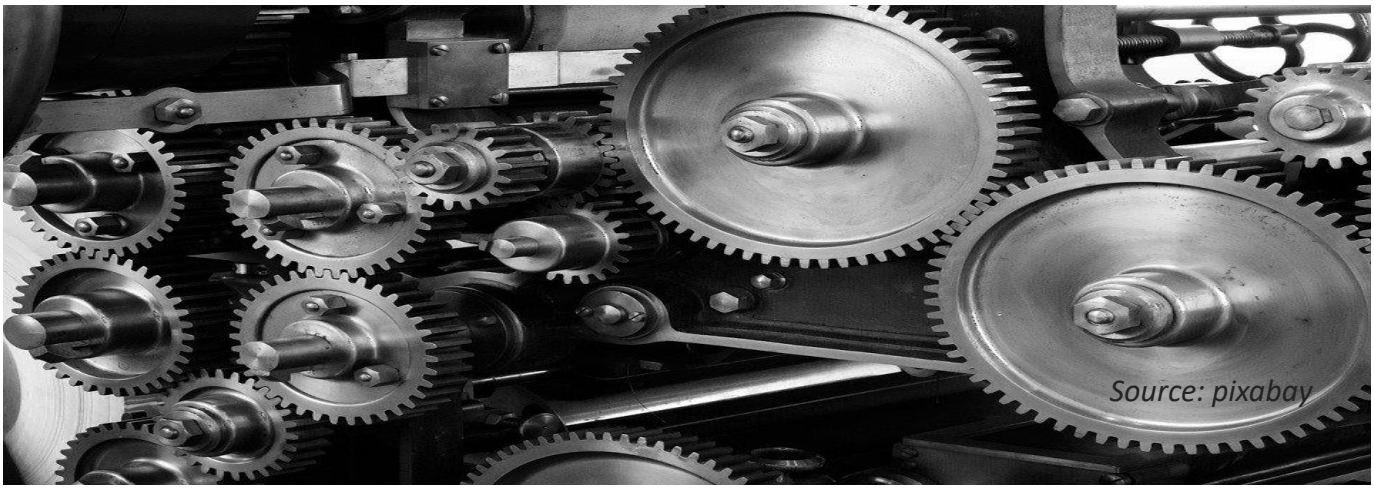


Fig. 4.5. The inverse z-transform output



Exercises

1. Using WolframAlpha Online, compute the z-transform of the following functions. Then compute the poles and zeros from the result using Octave Online. Write the program.

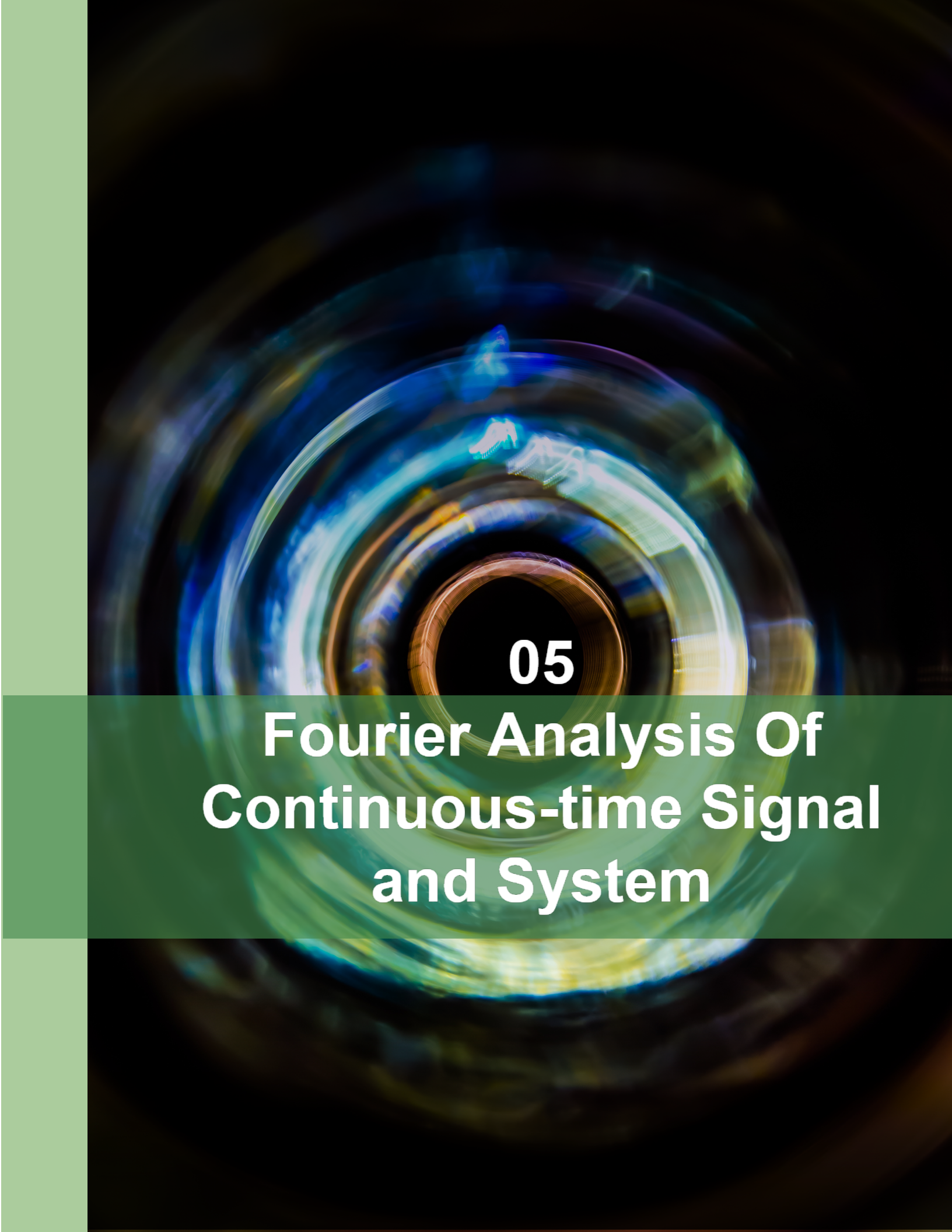
   a) $x[n] = (\frac{1}{3})^n u[n] + (\frac{1}{2})^n u[-n-1]$

   b) $x[n] = (\frac{1}{2})^n u[n] + (\frac{1}{3})^n u[-n-1]$

2. Using WolframAlpha Online, compute the inverse z-transform of the following functions.

   a) $X[z] = \frac{z^2}{(2z+1)(z-1)}$

   b) $X[z] = \frac{4(2z+1)}{(z+1)(z-3)}$



05

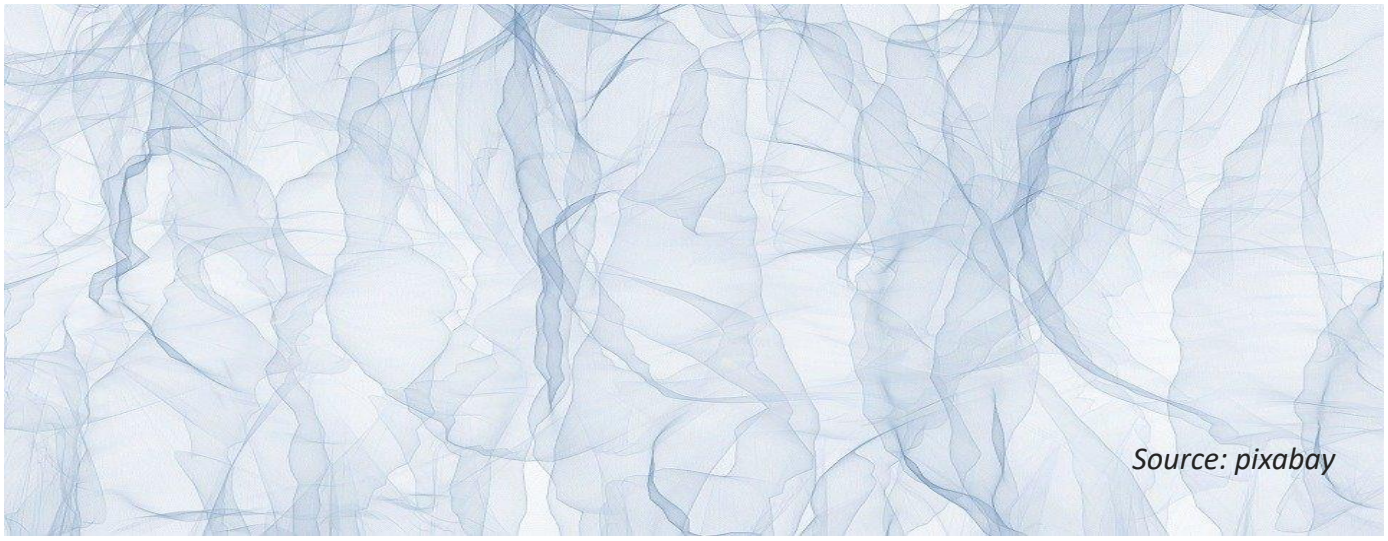
Fourier Analysis Of Continuous-time Signal and System

Fourier Analysis Of Continuous-time Signal and System

The Laplace transform and the z-transform have been introduced in the previous chapters as to convert time-domain signals into complex s-domain and z-domain. This representations are more straightforward to examine and process for various reasons. These modifications also reveal more information about the nature and behavior of various signals and systems.

We will introduce further transformations such as Fourier series and Fourier transform that convert time-domain signals into frequency domain (or spectral) representations in this and the subsequent chapters.

Fourier analysis is necessary for describing certain types of systems and their attributes in the frequency domain, in addition to generating spectral representations of signals. We will cover Fourier analysis in the context of continuous-time signals and systems in this chapter.



Source: pixabay

Trigonometric Fourier Series

The trigonometric Fourier series representation of a periodic signal x(t) with fundamental period T_0 is given by

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \{ak \cos k\,\omega_0 t + bk \sin k\,\omega_0 t\}$$

where a_k and b_k are the Fourier coefficients given by

$$a_k = \frac{2}{T_0} \int_{-T/2}^{T/2} f(x) \cos k\,\omega_0 t\, dt$$

$$b_k = \frac{2}{T_0} \int_{-T/2}^{T/2} f(x) \sin k\,\omega_0 t\, dt$$

If a periodic signal f(t) is even, then $b_k = 0$ and its Fourier series contains only cosine terms

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos k\,\omega_0 t$$

If f(t) is odd, then $a_k = 0$ and its Fourier series contains only sine terms

$$f(t) = \sum_{k=1}^{\infty} b_k \sin k\,\omega_0 t$$

Example:

Determine the trigonometric Fourier Series of the following signal.

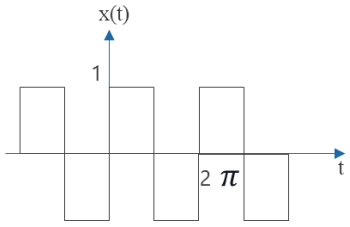


Fig. 5.1

Solution :

Step 1 : Identify type of given function either odd or even function

It is an odd function. Thus , $a_k = 0$

The Fourier series is

$$f(t) = \sum_{k=1}^{\infty} b_k \sin k\,\omega_0 t$$

Step 2 : Identify the fundamental period T_0 and ω_0

$$T_0 = 2\pi$$

$$\omega_0 = \frac{2\pi}{T_0} = \frac{2\pi}{2\pi} = 1$$

Step 3 : Find the Fourier coefficients

$$b_k = \frac{2}{T_0}\int_{-T/2}^{T/2} f(x)\sin k\,\omega_0 t\,dt$$

$$= \frac{2}{2\pi}\left[\int_0^{\pi}(1)\sin(kt)\,dt + \int_{\pi}^{2\pi}(-1)\sin(kt)\,dt\right]$$

$$= \frac{1}{\pi}\left[\left|\frac{(-\cos(kt))}{k}\right|_0^{\pi} + \left|\frac{\cos(kt)}{k}\right|_{\pi}^{2\pi}\right]$$

$$= \frac{1}{k\pi}[\{-cosk\pi - (-\cos(0))\} + \{\cos(k2\pi) - cosk\pi\}]$$

$$= \frac{1}{k\pi}[\{-cosk\pi - (-1)\} + \{1 - cosk\pi\}]$$

$$= \frac{1}{k\pi}[2 - 2\cos(n\pi)]$$

$$= \frac{2}{k\pi}(1 - (-1)^k)$$

Thus, trigonometric Fourier Series is

$$f(t) = \sum_{k=1}^{\infty}\left(\frac{2}{k\pi}1 - (-1)^k\right)\sin k\,\omega_0 t$$

Simulation

User can learn how to utilize octave-online software to execute Fourier Transform of Continuous-Time and find the Fourier coefficient

Trigonometric Fourier Series

We may utilize the *heaviside(t)* command in GNU Octave to plot a diagram of a periodic continuous-time signal as shown in Fig. 9. The Fourier coefficient are found with the following command array

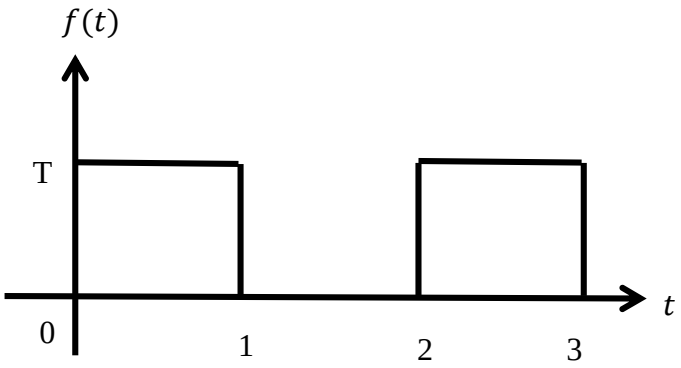


Fig. 9 : Periodic Signal

```
%Fourier Transform Of Continuous-Time
syms t
f=heaviside(t)-heaviside(t-1)

%Diagram of continuous-time
ezplot(f,[0,2])
T=2

%Fourier coefficients a0=c0
a0_sym=1/T*int(f,t,0,T)
double(a0_sym)
w0=2*pi/T
```

```
%Fourier coefficients
a1=2/T*int(f*cos(1*w0*t),t,0,T)
a2=2/T*int(f*cos(2*w0*t),t,0,T)
a3=2/T*int(f*cos(3*w0*t),t,0,T)

%Fourier coefficients
b1=2/T*int(f*sin(1*w0*t),t,0,T)
b2=2/T*int(f*sin(2*w0*t),t,0,T)
b3=2/T*int(f*sin(3*w0*t),t,0,T)
```

GNU Octave displays the Fourier coefficient a1,a2,a3,b1,b2 and b3 as in Table 1

Table 1 Fourier coefficient from GNU Octave

Fourier Coefficient	Output
a1	a1 = (sym) 0
a2	a2 = (sym) 0
a3	a3 = (sym) 0
b1	b1 = (sym) 2 -- pi
b2	b2 = (sym) 0
b3	b3 = (sym) 2 ---- 3*pi

Exercises

1. Execute the following integral function using GNU Octave as per example command given.

$$i)\ Q = \int_0^2 3t\,dt$$

$$ii)\ R = \int_0^5 t^2\,dt$$

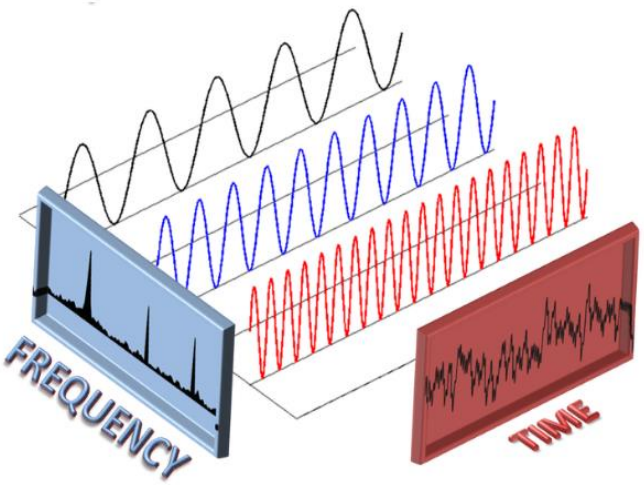
$$iii)\ S = \int_{-4}^4 (t+1)\,dt$$

```
Example :
%Calculate Integral Function Using 'Int' Command
syms t
Q=3*t
AQ= int(Q,t,0,2)
double(AQ)
```

06
Fast Fourier Transform

Fast Fourier Transform

The discrete Fourier transform (DFT) of a sequence, or its inverse, is computed using a fast Fourier transform (FFT) (IDFT). Fourier analysis transforms a signal from its original domain (typically time or space) to a frequency domain representation and back. Decomposing a sequence of values into components of different frequencies yields the DFT. This procedure is useful in a variety of domains, but computing it straight from a definition is typically too time consuming to be practical.



By factoring the DFT matrix into a product of sparse (mainly zero) elements, an FFT may quickly compute such modifications. The speed difference can be huge, especially for large data sets with N in the hundreds or millions. Many FFT techniques are substantially more accurate than directly or indirectly evaluating the DFT definition in the presence of round-off error. FFT algorithms are based on a variety of published theories, ranging from simple complex-number arithmetic to group theory and number theory.

By factoring the DFT matrix into a product of sparse (mainly zero) elements, an FFT may quickly compute such modifications. The speed difference can be huge, especially for large data sets with N in the hundreds or millions. Many FFT techniques are substantially more accurate than directly or indirectly evaluating the DFT definition in the presence of round-off error. FFT algorithms are based on a variety of published theories, ranging from simple complex-number arithmetic to group theory and number theory.

Simulation

Fast Fourier Transform

User can learn how to utilize GNU Octave to simulate Fourier analysis on discrete-time signal and system. The simulation is to covert a noise corrupted signal *y(t)* in time domain into frequency domain . First, a sound signal is created as shown in Fig. 6.1. Then, a random noise signal is created as shown in Fig. 6.2. This noise is added in the sound signal producing a corrupted signal as shown in Fig. 6.3. Next, fast Fourier Transform (FFT) is performed on the signal to calculate and plot the power spectrum as shown in Fig. 6.4. The number of sampling points is at 1000Hz for 5 seconds. This simulation is done using the following command array .

```
clc
clear all
% To convert a noise corrupted signal y in time domain into a frequency domain
%  number of sampling points at 1000Hz for 5 seconds
t2=0:0.001:0.5;
x2 = sin(2*pi*50*t2) + sin(2*pi*120*t2); %  a sound signal
figure(1)
plot(1000*t2(1:50),x2(1:50))
title('A sound signal x2');xlabel('Time(milliseconds)')
y = x2 + 2*randn(size(t2));  % create random noise to be added to the sound signal.
figure(2)
plot(1000*t2(1:50),y(1:50)) % plot the corrupted sound signal
title('Signal Corrupted with zero-mean Random noise');xlabel('time(milliseconds)')
% will perform 1 512-points fast Fourier transform(FFT), calculate and plot the
power spectrum
Y = fft(y,512);
Pyy = Y.*conj(Y)/512;
f = 1000*(0:256)/512;
figure(3)
plot(f,Pyy(1:257))
title('Frequency content of y');xlabel('frequency (Hz)')
```

GNU Octave displays discrete-time input x[n], impulse response h[n] and convolution output y[n] as shown in Fig. 6.1 to Fig. 6.4.

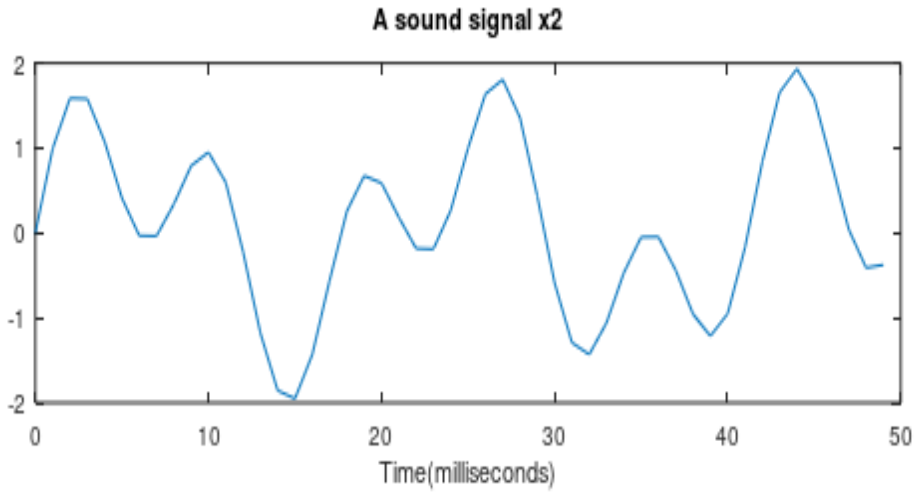


Fig. 6.1 : A sound signal

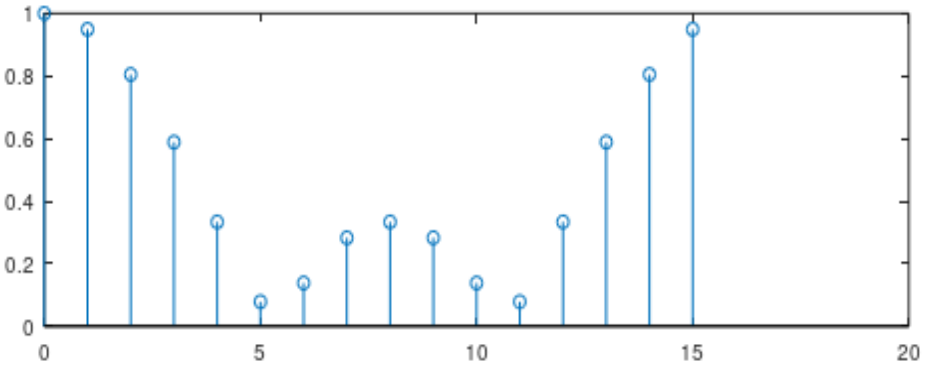


Fig. 6.2 : Random noise signal

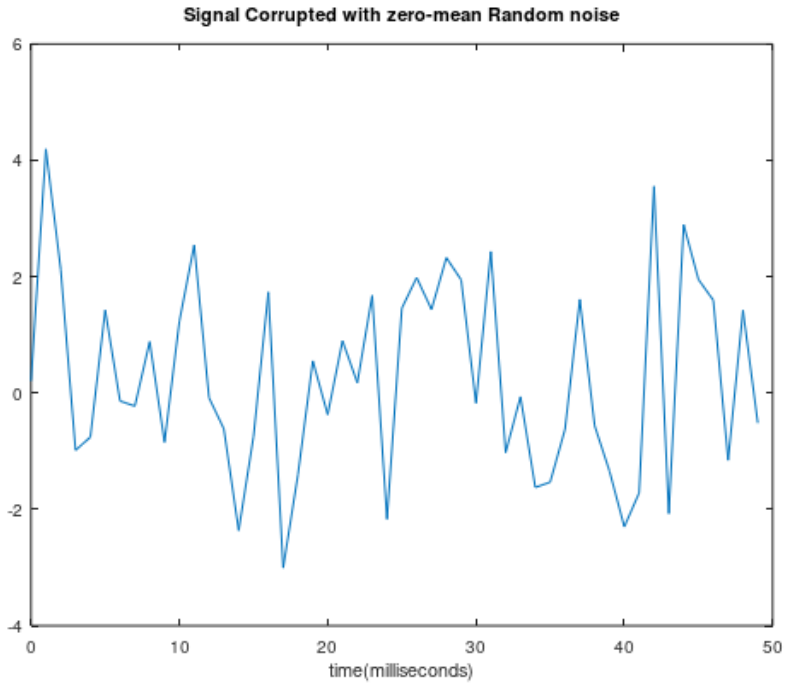


Fig. 6.3 : Signal corrupted with zero-mean random noise

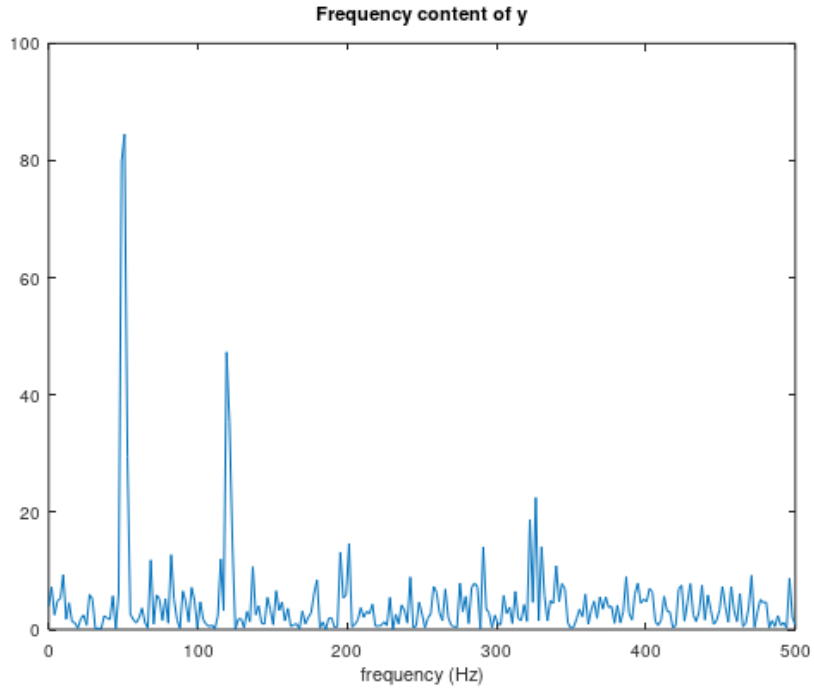
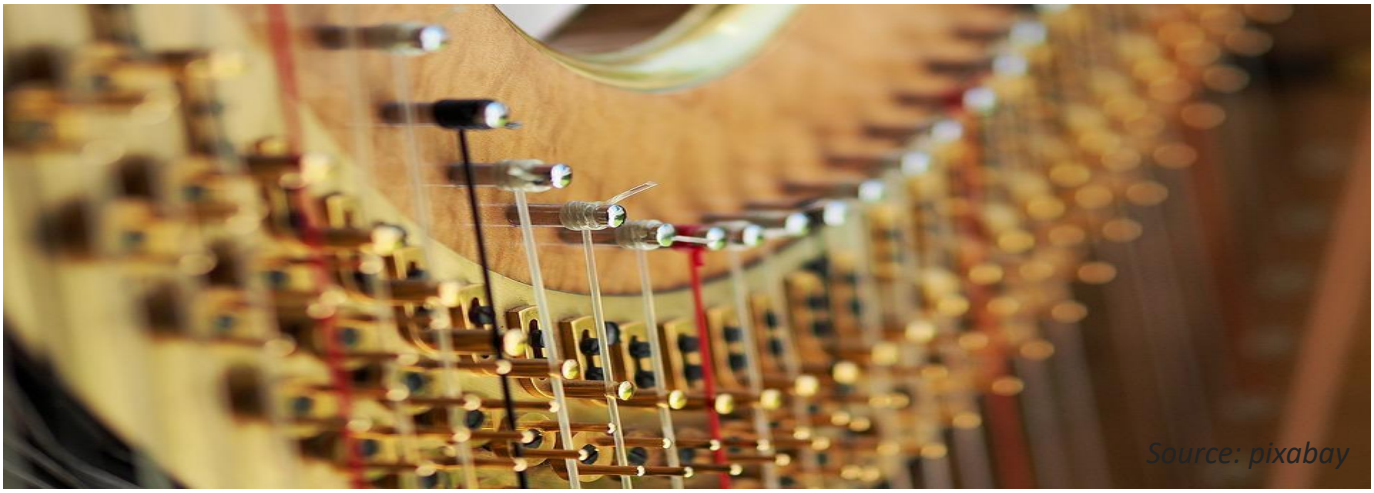


Fig. 6.4 : Frequency content of signal y(t)



Exercises

Consider the periodic square wave x(t) shown in Fig. 6.5. Find the Fourier Series for the function for which the graph is given below

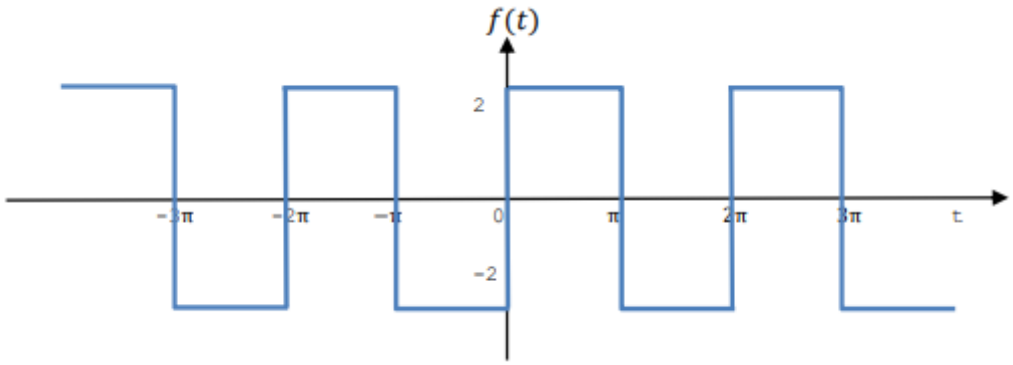


Fig. 6.5

REFERENCES

Hwei P.Hsu. (2014). *Schaums Outline of Digital Signal Processing* (4th ed)., United States: McGraw-Hill Education.

Steven T.Karris (2009). *Digital Signal Processing Using MATLAB & Wavelets. 2nd Edition*. Jones and Bartlett Publishers, Inc. Andreas Antoniou.

Sylvia Ong, Dyg Khayrunsalihaty. (2021). *Signal and System* : Augmented-Reality Experience. Politeknik Kuching Sarawak.

Icon
www.pixabay.com

Fauziah Aliman is the senior lecturer at Politeknik Merlimau, Melaka (PMM). She earned a Bachelor's Degree in Electronic Engineering at Multimedia University (MMU), Melaka and Master's Degree in Electronic Engineering (System Electronics) at Universiti Teknikal Malaysia Melaka (UTeM) . She is a registered graduate engineer by Board of Engineers Malaysia. She has over seven years of engineering experience in semiconductor industry. In addition, she has over fourteen years of teaching experience that she acquired at Technical and Vocational Education and Training (TVET) institutions. She is currently with Department of Electrical Engineering of Politeknik Merlimau.

Syamsul Bahri Bin Mohamad is a Lecturer in the Department of Electrical Engineering, Politeknik Merlimau, Melaka (PMM). He holds a Master's Degree (Communication and Computer) from Universiti Kebangsaan Malaysia and a Master's Degree (Technical and Vocational Education) as well as a Bachelor's Degree (Electrical Engineering) from Kolej Universiti Teknologi Tun Hussein Onn (KUiTTHO) which is now Universiti Tun Hussein Onn (UTHM). He is a lecturer for the Signal & System at Electrical Engineering Department, Politeknik Merlimau, which is one of the courses offered at the Malaysia polytechnic.

e ISBN 978-967-2762-14-0
9 7 8 9 6 7 2 7 6 2 1 4 0
SIGNAL AND SYSTEM