# Part II

# Signals Circuits and Systems

# Workshop 4: Signals in Time Domain

**Objective**: To analyze continuous-time and discrete-time signals in time domain.

**Outcome**: After successfully completion of this session, the student would be able to

1. Plot basic continuous-time and discrete-time signals in time domain
2. Creating a simple audio effect

**Equipment Required**:

1. A personal computer.
2. Python and NumPy.
3. A headphone set (to be brought by the student)

**Components Required**: None.

## 4.1 Real CT Sinusoid

Consider the following general sinusoidal signal:

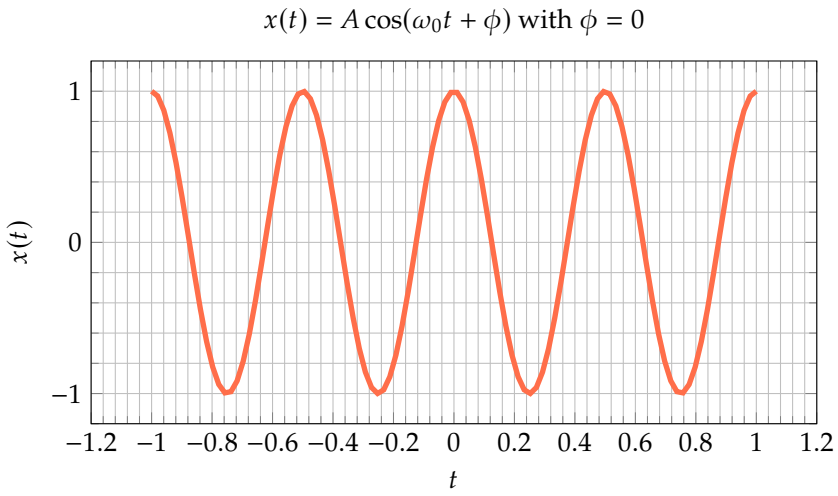
$$x(t) = A\cos(\omega_0 t + \phi) \tag{4.1}$$

where $A$ is the amplitude, $\omega_0 = 2\pi f_0$ is the angular frequency and $\phi$ is the phase. We can use the following Python code to plot such a signal: Let's import required packages first

```
import numpy as np              # import numpy to handle numerics
import matplotlib.pyplot as plt     # import matplotlib to hadle figures

%matplotlib inline
```

**Example 1.** *Plot of* $x(t) = A\cos(\omega_0 t + \phi)$ *with* $\phi = 0$.

```
fs = 44100 # 44,100—Hz sampling frequency
ts = 1/fs
t = np.arange(—1., 1., ts) # A linearly—spaced array with step ts
A = 1.
f = 2 # 2 Hz
w0 = 2*np.pi*f
phi = 0.
xt = A*np.cos(w0*t + phi)
fig, ax = plt.subplots()
ax.plot(t,xt)
plt.show()
```




**Task 28.** *Use the following code snippet to generate plots for*

1. $A = 1$, $\omega_0 = 2\pi$, $\phi = 0$, *i.e.*, $x_1(t) = \cos(2\pi t)$.
2. $A = 0.5$, $\omega_0 = 2\pi$, $\phi = 0$, *i.e.*, $x_2(t) = 0.5\cos(2\pi t)$.
3. $A = 1$, $\omega_0 = 2\pi$, $\phi = \pi/2$, *i.e.*, $x_3(t) = -\sin(2\pi t)$.
4. $A = 1$, $\omega_0 = 4\pi$, $\phi = 0$, *i.e.*, $x_4(t) = \cos(4\pi t)$.

```
fs = 44100 # 44,100—Hz sampling frequency
ts = 1/fs
t = np.arange(—1., 1., ts) # A linearly—spaced array with step ts
f = 2 # 2 Hz
fig, axes  = plt.subplots(2,2, sharex='all', sharey='all', figsize=(18,18))
# Your code goes here
xt1 =
xt2 =
xt3 =
xt4 =

axes[0, 0].plot(t,xt1)
axes[0, 0].set_xlabel('$t$')
axes[0, 0].set_ylabel('$xt1$')
axes[0, 0].grid(True)
axes[0, 0].title.set_text('$\cos(2\pi t)$')

# Your code goes here


plt.show()
```

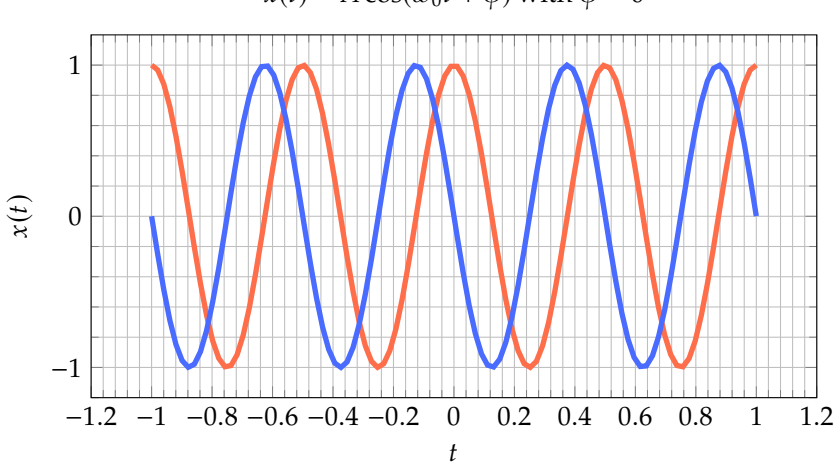
Among $x_1(t)$, $x_2(t)$, $x_3(t)$, and $x_4(t)$, identify the odd and even signals.

**Example 2.** *The following example illustrates plotting two waves in the same plot.*

```
fs = 4.4e4 # sampling frequency
ts = 1/fs
t = np.arange(—1., 1., ts)
f = 2 # 2 Hz
```

```
w0 = 2*np.pi*f
phi = 0.
x1t = np.cos(w0*t + phi)
phi = np.pi/2
x2t = np.cos(w0*t + phi)
fig, ax = plt.subplots()
ax.plot(t,x1t, label='$\cos(4\pi t)$')
ax.plot(t,x2t, label='$\cos(4\pi t + \pi/2)$')
ax.set_xlabel('$t$')
ax.set_ylabel('$x(t)$')
plt.legend(loc='lower right')
ax.grid(True)

plt.show()
```

Listing 4.1: Plot of $x(t) = A\cos(\omega_0 t + \phi)$ with $\phi = 0$ and $\phi = \pi/2$




## 4.2 Real CT Exponential(Graded)

Write a Python code to plot the signal $x(t) = Ce^{\alpha t}$ with $C = 1.0$ and $\alpha = 1.2$. Hint: np.exp gives the element-wise exponent of a NumPy array.

```
fs = 4.4e4 # sampling frequency
ts = 1/fs
t = np.arange(−1., 1., ts)

# Your code goes here
```

Comment on the effect due to the sign of $\alpha$ on the signal.

## 4.3 Growing Sinusoidal Signal (Graded)

Write a Python code to plot the signal with $x(t) = Ce^{rt}\cos(\omega_0 t + \theta)$

1. $C = 0.15$, $r = 2$, and $\theta = 0$.
2. $C = 0.15$, $r = -2$, and $\theta = \pi/2$.

```
fs = 4.4e4 # sampling frequency
ts = 1/fs
t = np.arange(-1., 1., ts)
f = 5 # 5 Hz

# Your code goes here
```

## 4.4 Real DT Exponential (Graded)

Use the following code snippet plots the discrete-time signal $x[n] = C\alpha^n$ for

1. $C = 1$, $\alpha = 1.1$
2. $C = 1$, $\alpha = 0.92$
3. $C = 1$, $\alpha = -1.1$
4. $C = 1$, $\alpha = -0.92$

```
n = np.arange(-10,10,1)
fig, axes  = plt.subplots(2,2, sharex='all', sharey='all', figsize=(18,18))
C = 1.0
alpha = 1.1
xn1 = C*alpha**(n)

# Your code goes here


axes[0, 0].stem(n,xn1)
axes[0, 0].set_xlabel('$n$')
axes[0, 0].set_ylabel('$1.1^{n}$')
axes[0, 0].grid(True)

# Your code goes here

plt.show()
```

## 4.5 DT Sinusoids (Graded)

Use the following code snippet plots the discrete-time signal $x[n] = C\alpha^n$ for

1. $x_1[n] = A\cos(\omega_0 n)$ with $A = 1$ and $\omega_0 = 2\pi/12$.
2. $x_2[n] = A\cos(\omega_0 n)$ with $A = 1$ and $\omega_0 = 2\pi/6$.
3. $x_3[n] = A\cos(\omega_0 n)$ with $A = 1$ and $\omega_0 = 8\pi/31$.
4. $x_4[n] = A\cos(\omega_0 n)$ with $A = 1$ and $\omega_0 = 1/1.5$.

```
n = np.arange(−32,32,1)
fig, axes  = plt.subplots(4,1, sharey='all', figsize=(18,18))

# Your code goes here
```

Which of the signals are periodic? State the period for the periodic signals. (Graded)

1. $x_1[n]$
2. $x_2[n]$
3. $x_3[n]$
4. $x_4[n]$

## 4.6 General Complex Exponential Signals (Graded)

Plot the real part of $x[n] = C\alpha^n$ with $C = |C|e^{j\theta}$, $\alpha = |\alpha|e^{j\omega_0}$ for following $C$, $\alpha$, and $\omega_0$ values. Hint: $\text{Re}\{x[n]\} = |C||\alpha|^n \cos(\omega_0 n + \theta)$

1. $C = 1, \alpha = 1.1, \omega_0 = 2\pi/12$
2. $C = 1, \alpha = 0.92, \omega_0 = 2\pi/12$

```
n = np.arange(−12,12,1)
fig, axes  = plt.subplots(2,1, sharex='all', sharey='all', figsize=(18,18))
theta = 0.0

# Your code goes here
```

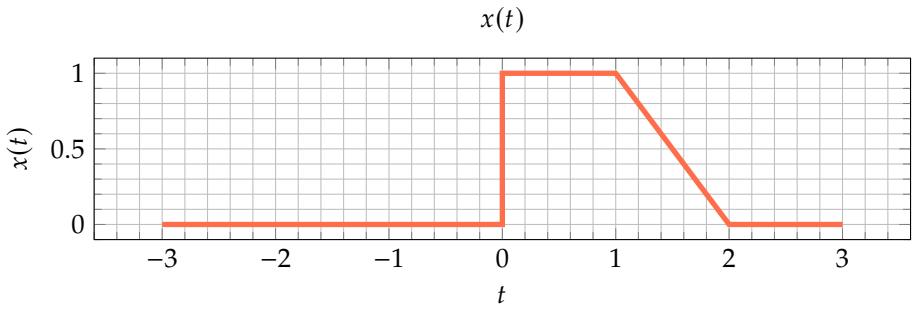
What will happen if $\alpha$ is negative? (Graded)

## 4.7 Transformation of the Independent Variable (Graded)

The following code plots a signal $x(t)$.

```
def x(t):
    if (t < 0.):
        return 0.
    elif (t < 1.):
        return 1.
    elif (t < 2.):
        return 2. − t
    else:
        return 0.

fs = 4.4e4 # 44,000−Hz sampling frequency
ts = 1/fs
t = np.arange(−3.5, 3.5, ts) # A linearly−spaced array with step ts
fig, axes  = plt.subplots(7,1, sharey='all', figsize=(10,15))

# x(t)
axes[0].plot(t, [x(t_) for t_ in t])
axes[0].set_xlabel('$t$')
```

```
axes[0].set_ylabel('$x(t)$')
axes[0].grid(True)

# Your code goes here

plt.show()
```




Update the above code to plot following functions:

1. $x(t-1)$
2. $x(t+1)$
3. $x(-t)$
4. $x(-t+1)$
5. $x(3t/2)$
6. $x(3t/2+1)$

## 4.8 Observing a Signal in Frequency Domain

The following code plots a signal and its frequency domain representation. Note that the non-zero values are actually delta functions in frequency, which we will learn later.

```
fs = 100# 100—Hz sampling frequency
ts = 1/fs
t = np.arange(—1., 1., ts) # A linearly—spaced array with step ts
fig, axes  = plt.subplots(2,1, figsize=(10,10))


f = 2 # 2 Hz
omega0 = 2*np.pi*f


xt = 0.75 + 2.*np.cos(omega0*t) + 1.*np.cos(3*omega0*t)

Xf = np.fft.fft(xt)
freq = np.fft.fftfreq(t.shape[—1], d=ts)

axes[0].plot(t,xt)
axes[0].set_xlabel('$t$ in s')
axes[0].set_ylabel('$x(t)$')
valsubrange = np.concatenate((np.arange(0,21,1), np.arange(—1,—21,—1)))
freqsubrage = np.concatenate((np.arange(0,21,1), np.arange(—1,—21,—1)))
axes[1].stem(freq[freqsubrage], Xf.real[valsubrange]/len(t))
```

```
axes[1].set_xlabel('$f$ in Hz')
axes[1].set_ylabel('$\mathfrak{Re}(X(j\omega))$')
plt.xticks(np.arange(-10,11))
plt.show()
```

Interpret the frequency domain representation. [Graded]

## 4.9 Simple Audio Effects

First, let's install "pyaudio". Please run following commands:

```
!apt install libasound2-dev portaudio19-dev libportaudio2 libportaudiocpp0 ffmpeg
```

```
!pip install pyaudio
```

The following code will play the audio file with the extension ".wav":

```
from IPython.display import Audio # playing the echo wav file
Audio('audio_file.wav')
```

The following code reads and plays a wave file. Create a simple audio effect of your choice. This will need pyaudio to be installed and ffmpeg in the path.

```
import numpy as np
import pyaudio
import wave
from IPython.display import Audio

# import utility

CHUNK = 8820 # 44100 = 1 s

wf = wave.open("power_of_love.wav", 'r')

p = pyaudio.PyAudio()
nchannels=wf.getnchannels()


stream = np.array(np.zeros(nchannels), dtype=np.int16) # init stream
data = wf.readframes(CHUNK)

dtype = '<i2' # little-endian two-byte (int16) signed integers

sig = np.frombuffer(data, dtype=dtype).reshape(-1, nchannels)
signal_chunk = np.asarray(sig)

delayed = np.zeros(signal_chunk.shape, dtype=dtype)

i=0
alpha = 1.0
while data != '' and signal_chunk.shape[0] == CHUNK and i<120:
    i+=1
```

```
    modified_signal_chunk = alpha*signal_chunk + (1. - alpha)*delayed
    modified_signal_chunk_int16 = modified_signal_chunk.astype(np.int16)
    stream = np.vstack((stream, modified_signal_chunk_int16)) # append modified to stream
    # byte_chunk = modifed_signal_chunk_int16.tobytes()
    # stream.write(byte_chunk)
    delayed = signal_chunk
    data = wf.readframes(CHUNK)
    sig = np.frombuffer(data, dtype=dtype).reshape(-1, nchannels)
    signal_chunk = np.asarray(sig)


stream = stream[1:] # pop stream init
byte_stream = stream.tobytes() # np array to bytes
p.terminate()
wf.close()

wfo = wave.open("power_of_love_eco.wav", 'wb') # writing the bot stream to a output wav file
wfo.setnchannels(nchannels)
wfo.setsampwidth(wf.getsampwidth())
wfo.setframerate(wf.getframerate())
wfo.writeframes(byte_stream)
wfo.close()

Audio('power_of_love_eco.wav')
```

Write down the changes that you made to the code. [Graded]

# Part III

# Electronics

# Workshop 9: Building a Simple Audio System using its Building Blocks

**Objective**: To design and implement a simple audio amplifier using common electronic components and verify its operation

**Outcome**: After successful completion of this session, the student will be able to

1. Design voltage amplifiers with controlled gain using operational amplifiers
2. Implement active low pass and active high pass filters using operational amplifiers
3. Explain the relationship between tones and frequency components of an audio signal

**Equipment Required**:

1. Audio Source (PC, Smartphone etc.)
2. Digital Oscilloscope
3. Signal Generator
4. DC power supply
5. Digital multi-meter
6. Breadboard and wires

**Components Required**:

1. Electronic Module 1: Pre-amplifier and Tone-Controller
2. Electronic Module 2: Power Amplifier
3. 3.5mm Audio Jack
4. Resistors - 1kΩ (1 Nos.), 10kΩ(1 Nos), 22kΩ (1 Nos.)
5. 4Ω 5W Speaker

## 9.1 Introduction

An amplifier is an active electronic device designed to amplify the voltage/current/power of a signal. In this experiment, experiment you will build a simple audio amplifier and in the process, you will learn about voltage amplifiers, active filters and power amplifiers. As shown in figure 9.1, the block diagram of the audio system, the original audio signal from the audio source is first fed into a *pre-amplifier*. Audio sources such as microphones typically produce a weak AC signal. The purpose of the pre-amplifier is to amplify this weak AC signal to a more noise-tolerant output signal which is strong enough for further processing. After the pre-amplifier, the signal is passed through a *tone controller*, which allows one to increase or diminish the effects of different tones, such as bass and treble, in the audio signal. Then, as the final stage of the audio amplifier, we have the *power amplifier*. The power amplifier amplifies the power of the processed input audio signal to a level that is high enough to drive speakers or headphones.


Figure 9.1: Audio Amplifier System - Block Diagram

In this experiment, you will find two electronic modules among your set of components.

- Electronic Module 1: Pre-amplifier and Tone-Controller - implemented using a NE5532 dual low-noise operational amplifier IC

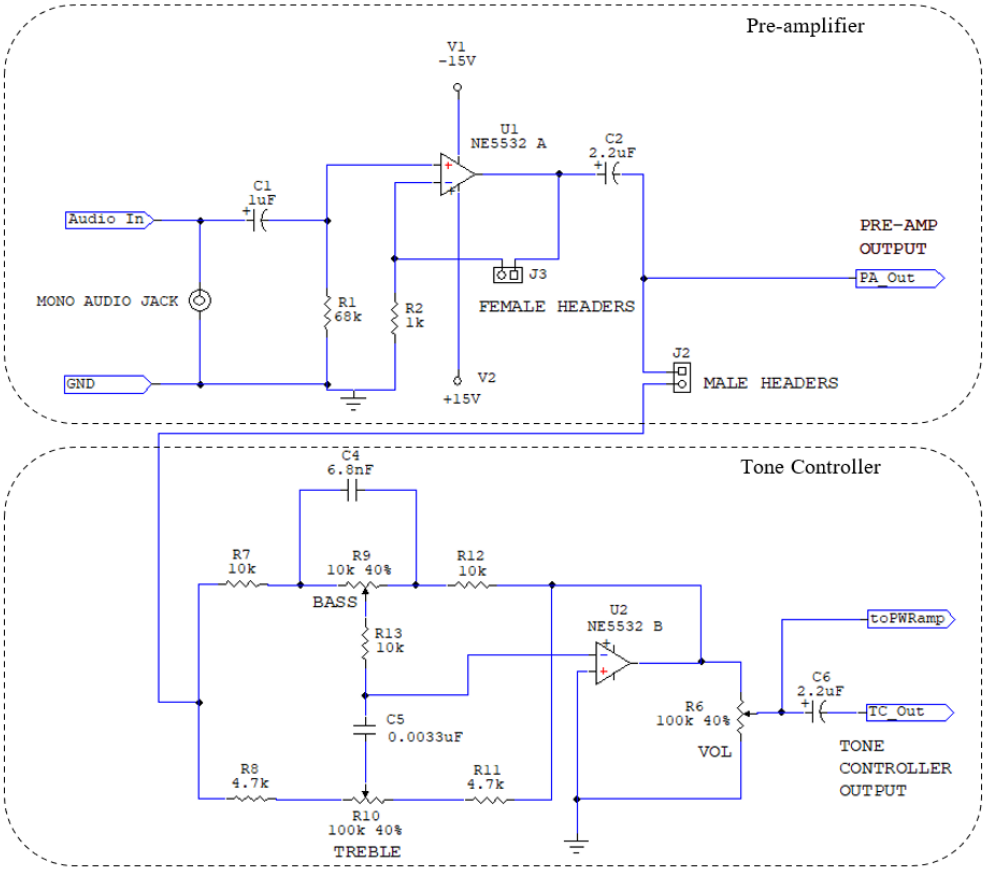



Figure 9.2: Schematic diagram of Electronic Module 1

- Electronic Module 2: Power Amplifier - implemented using a TDA2003 power amplifier IC

Go through your components and identify the two modules.

## 9.2 Pre-amplifier

The schematic diagram of the pre-amplifier is shown in figure 9.3. Compare the schematic of the pre-amplifier (figure 9.3) and Electronic Module 1 (figure 9.2) to identify the roles of J3 and J2 in the module.

**Task 89.** *Is the pre-amplifier designed as an inverting amplifier or a non-inverting amplifier?*

**Task 90.** *Derive an equation for the voltage gain of the pre-amplifier in terms of R3 and R2.*

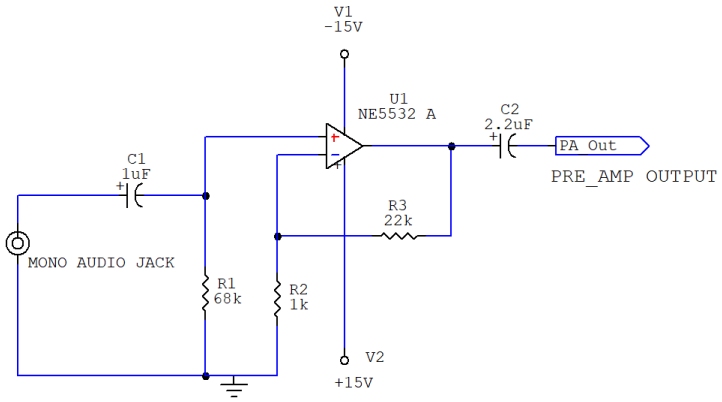



Figure 9.3: Schematic diagram of the pre-amplifier

**Task 91.** *Calculate the theoretical voltage gain of the pre-amplifier according to the values given in the schematic diagram (figure 9.3).*

**Task 92.** *Give power to the pre-amplifier using the DC power supply. Configure a 1kHz 1Vpp Sinosoidal waveform on the signal generator, give that waveform as the input signal to the powered-up pre-amplifier. Observe the output of the pre-amplifier using the oscilloscope and plot the observed output waveform.*

**Task 93.** *Calculate the practically observed voltage gain of the pre-amplifier (considering the input waveform and the output waveform observed in task 92).*

## 9.3 Tone Controller

The schematic diagram of the tone controller is shown in figure 9.4.

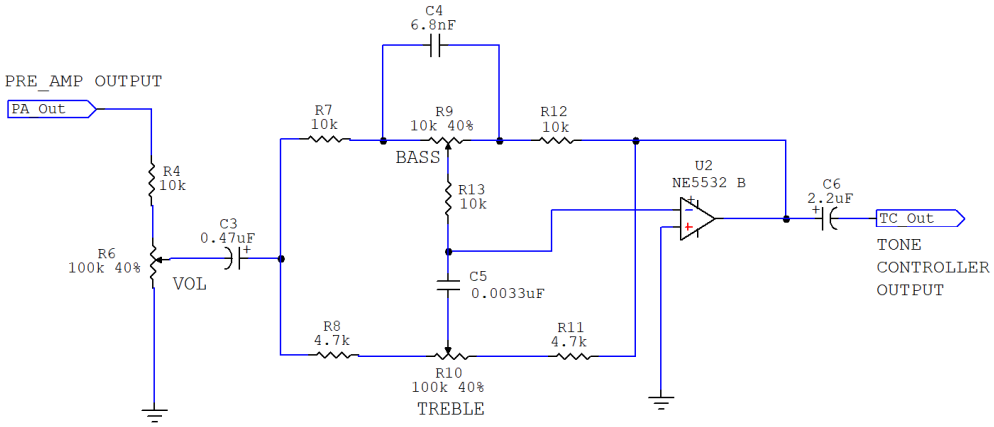



Figure 9.4: Schematic diagram of the tone controller

**Task 94.** *Set both the bass potentiometer and the treble potentiometer to their mid values.*

**Task 95.** *Configure a 100Hz 1Vpp Sinosoidal waveform on the signal generator, give that waveform as the input signal to the powered-up pre-amplifier. Observe the output of the tone controller using the oscilloscope and adjust the volume potentiometer so that the tone controller output is not clipped.*

**Task 96.** *Change the value of the bass potentiometer and comment on its effect on the tone controller output.*

**Task 97.** *Set the bass potentiometer to back to its mid value.*

**Task 98.** *Change the value of the treble potentiometer and comment on its effect on the tone controller output.*

**Task 99.** *Set the treble potentiometer to back to its mid value.*

**Task 100.** *Configure a 5kHz 1Vpp Sinosoidal waveform on the signal generator, give that waveform as the input signal to the powered-up pre-amplifier. Observe the output of the tone controller using the oscilloscope and adjust the volume potentiometer so that the tone controller output is not clipped.*

**Task 101.** *Change the value of the bass potentiometer and comment on its effect on the tone controller output.*

**Task 102.** *Set the bass potentiometer to back to its mid value.*

**Task 103.** *Change the value of the treble potentiometer and comment on its effect on the tone controller output.*

## 9.4 Power Amplifier

The schematic diagram of the power amplifier is shown in figure 9.5.

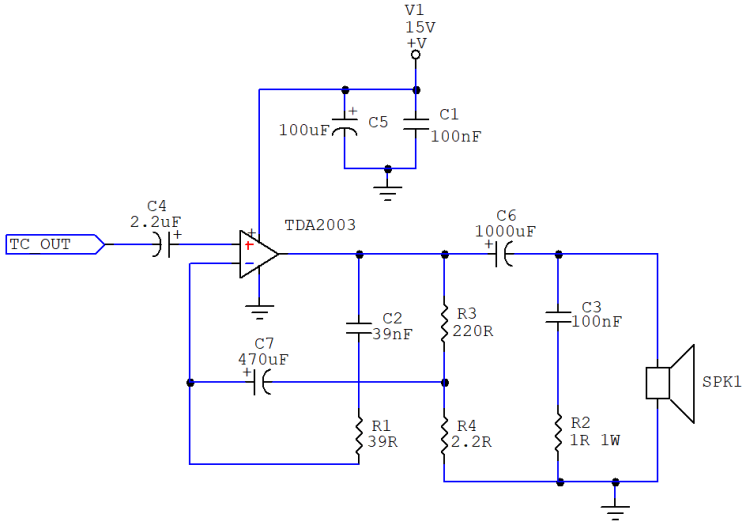



Figure 9.5: Schematic diagram of the power amplifier

**Task 104.** *Set the tone controller potentiometers to their mid values and connect the power amplifier module to the output of the tone controller.*

**Task 105.** *Disconnect the signal generator from the pre-amplifier and connect the 3.5mm mono audio jack to the pre-amplifier input.*

**Task 106.** *Connect the 3.5mm mono audio jack to your smartphone or PC and play a music track with noticeable bass and treble tones.*

**Task 107.** *Power up the audio amplifier using the DC power supply.*

**Task 108.** *Tune the volume control potentiometer so that the output from the speaker is minimally distorted. (A certain amount of distortion is unavoidable due to the wiring, heating of components etc.)*

**Task 109.** *Change the values of the tone controller potentiometers and verify the proper operation of the audio amplifier.*

♣ The End ♣

# Part IV

# Telecommunication

# Workshop 15: Communication Channel Characteristics & Effects of Noise

**Objective**: To Observe Communication Channel Characteristics & Effects of Noise

**Outcome**: After successfully completion of this session, the student would be able to

1. Identify the cut-off frequencies and bandwidth associated with channel characteristics
2. Estimate the cut-off frequencies and bandwidth by measurement
3. Identify the effects of bandwidth limitation on signals carried by a channel
4. Examine the effect S/N ratio in analog signal transmission
5. Estimate the Bit Error Rate (BER) in digital signal transmission as a function of S/N

**Equipment Required**:

1. Oscilloscope
2. Signal Generator
3. Proto board
4. A Personal Computer
5. MATLAB software or MATLAB online

**Components Required**:

1. Resistors 270Ω (1), 330Ω, (1), 1 kΩ (1)
2. Capacitors 1µF (1), 0.33µF (1)

## 15.1 Communication Channel

Figure 15.1 shows the block diagram of a basic communication system (analog/digital).

The **source** is the device or the person which generates the data to be transmitted. The **transmitter** processes and transmits the data. **Channel** is the medium through which the data is transmitted. **Receiver** receives and processes the transmitted data. The **Destination** is the final device or person to which the data is intended.

In this practical we will be focusing on the effects induced by the channel. The channels typically introduce three major effects to the transmitted data.

- Attenuation
- Noise Addition
- Bandwidth limitation

Let the transmitted signal be $X(t)$. Attenuation is simply the reduction of the amplitude of the signal. The signal after attenuation is $AX(t)$, where $0 < A < 1$ is the attenuation factor.

Noise is the random fluctuation of the signal around it's actual value. The signal after attenuation and noise can be written as $AX(t) + N(t)$, where $N(t)$ represents a random signal.

The channels typically limit the bandwidth of the transmitted signal. In other words certain frequencies of the transmitted signal are filtered. Hence, the signal after all three effects can be written as bandlimit$(AX(t) + N(t))$ where, the bandlimit function represents the bandwidth limitation.

In this practical we will be focusing on understanding these effects in detail.

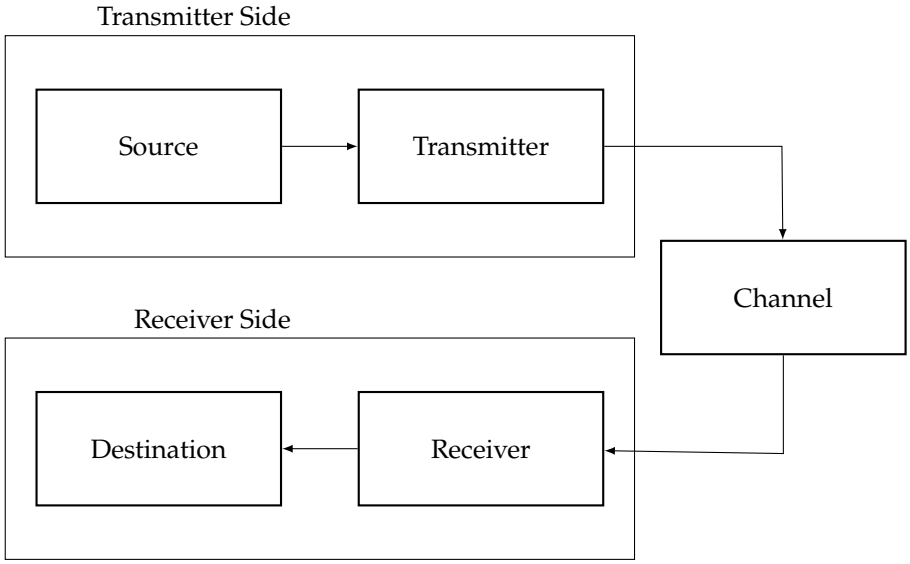



Figure 15.1: The block diagram of a communication system

## 15.2 Pre-Lab

Prior to the lab you will model a basic communication channel using Simulink. Channels for both analog and digital transmission can be modelled using the above effects.

### 15.2.1 Analog Channel

We will first focus on an analog channel. In an analog channel an analog signal is transmitted through the channel. In this case we model the noise $N(t)$ as a Gaussian noise. The bandlimiting filter is a low-pass filter. Build the following simulink model (Figure 15.2) in bring it to the lab.

Use the following parameters.

- Transmitted sinusoidal signal [**Sine Wave** block in **Simulink** > **Sources**]
  - Amplitude: 10
  - Frequency: 200 rad/s
  - Sample time: 0.001 s
- Channel attenuation [**Gain** block in **Simulink** > **Commonly used blocks**]
  - Gain: 0.8
- Attenuated signal, Signal corrupted by noise, Signal output from channel, Information signal, Noise, SNR grpah, SNR average [**Scope** block in **Simulink** > **Commonly used blocks**]
- Add [**Add** block in **Simulink** > **Math Operations**]
- Channel noise [**Random Source** block in **DSP System Toolbox** > **Sources**]
  - Source type: Gaussian
  - Mean value: 0
  - Sample mode: Discrete
  - Variance: 1
  - Sample time: 0.001 s

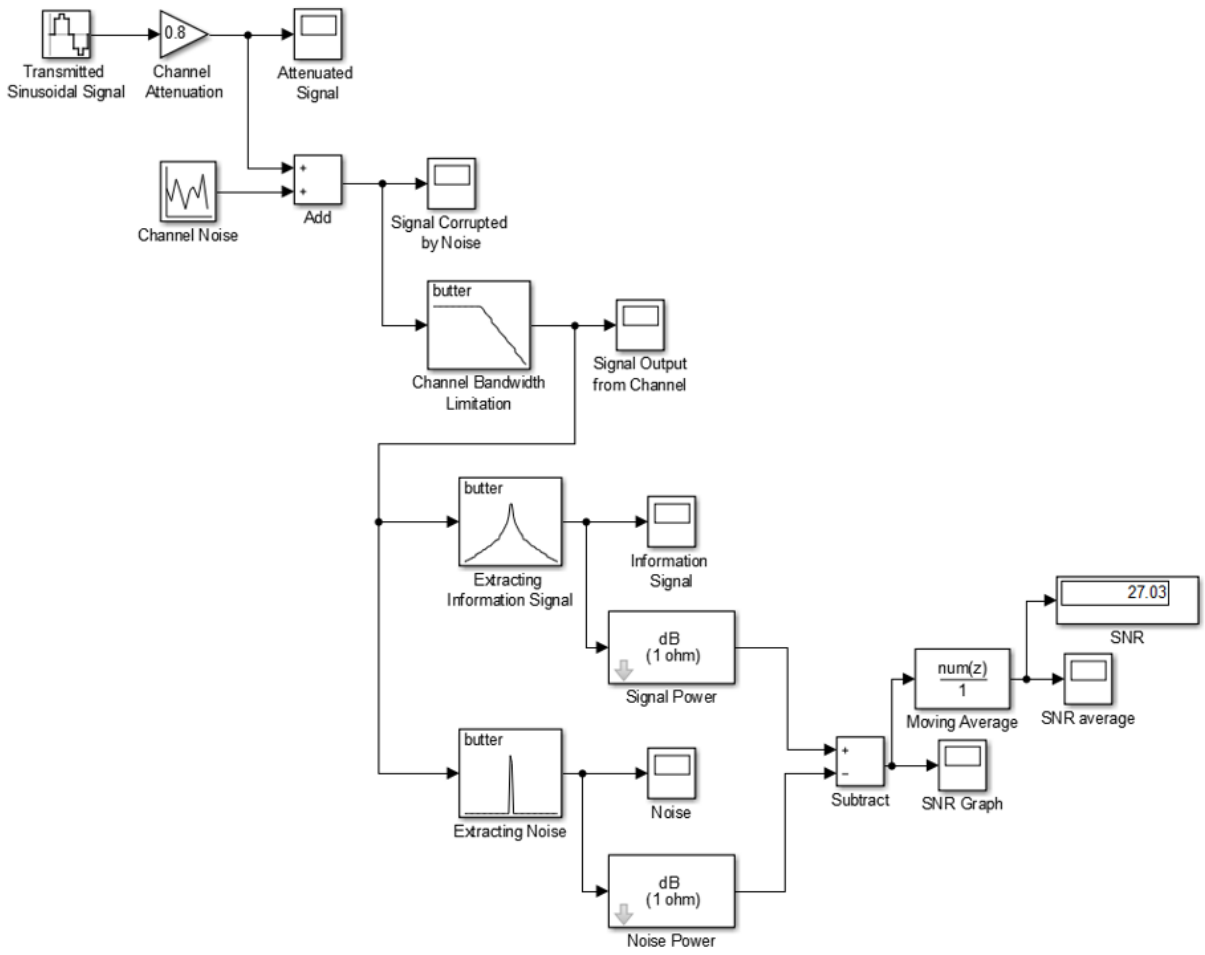



Figure 15.2: Simulink Model for Analog Channel.

- Channel bandwidth limitation [**Analog Filter Design** block in **DSP System Toolbox** > **Filtering** > **Filter Implementations**]
  - Design method: Butterworth
  - Filter type: Lowpass
  - Filter order: 10
  - Passband edge frequency: 1000 rad/s
- Extracting information signal [**Analog Filter Design** block in **DSP System Toolbox** > **Filtering** > **Filter Implementations**]
  - Design method: Butterworth
  - Filter type: Bandpass
  - Filter order: 10
  - Lower passband edge frequency: 195 rad/s
  - Upper passband edge frequency: 205 rad/s
- Extracting noise [**Analog Filter Design** block in **DSP System Toolbox** > **Filtering** > **Filter Implementations**]

  - Design method: Butterworth
  - Filter type: Bandstop
  - Filter order: 10
  - Lower passband edge frequency: 195 rad/s
  - Upper passband edge frequency: 205 rad/s
- Signal power, Noise power [**dB Conversion** block in **Communications System Toolbox** > **Utility Blocks**]
  - Convert to: dB
  - Input signal: Amplitude
  - Load resistance: 1 ohm
- Subtract [**Subtract** block in **Simulink** > **Math Operations**]
- SNR [**Display** block in **Simulink** > **Sinks**]
- Moving average [**Discrete FIR Filter** block in **Simulink** > **Discrete**]
  - Filter structure: Direct form
  - Sample time: 0.001
  - Coefficients: ones(1,1000)/1000

**Task 167.** *Set the simulation time to 20s and run the simulation. Observe the effects of attenuation, channel noise and channel bandwidth limitation. Explain how the SNR is calculated.*

### 15.2.2 Digital Channel

In a digital channel a bit stream is transmitted. The bit 1 is transmitted with a pulse of amplitude 4, and the bit 0 is transmitted with a pulse of amplitude -4. We model the noise as Gaussian noise similar to the analog channel. Build the following simulink model (Figure 15.3) in bring it to the lab.

Use the following parameters.

- Binary Generator [**Bernoulli Binary Generator** block in **Communication System Toolbox** > **Comm sources** > **Random Data Sources**]
  - Probability of a zero: 0.5
  - Source of initial seed: Parameter
  - Initial seed: 61
  - Sample time: 0.1 s
- Gain [**Gain** block in **Simulink** > **Math Operations**]
  - Gain: 8
- Constant [**Constant** block in **Simulink** > **Sources**]
  - Constant value: -4
- Add [**Add** block in **Simulink** > **Math Operations**]
- Digital information signal, Signal corrupted by noise, Channel output, Output from sampler, Output from thresholder, Output from detector [**Scope** block in **Simulink** > **Commonly used blocks**]
- Channel attenuation [**Gain** block in **Simulink** > **Math Operations**]
  - Gain: 0.5

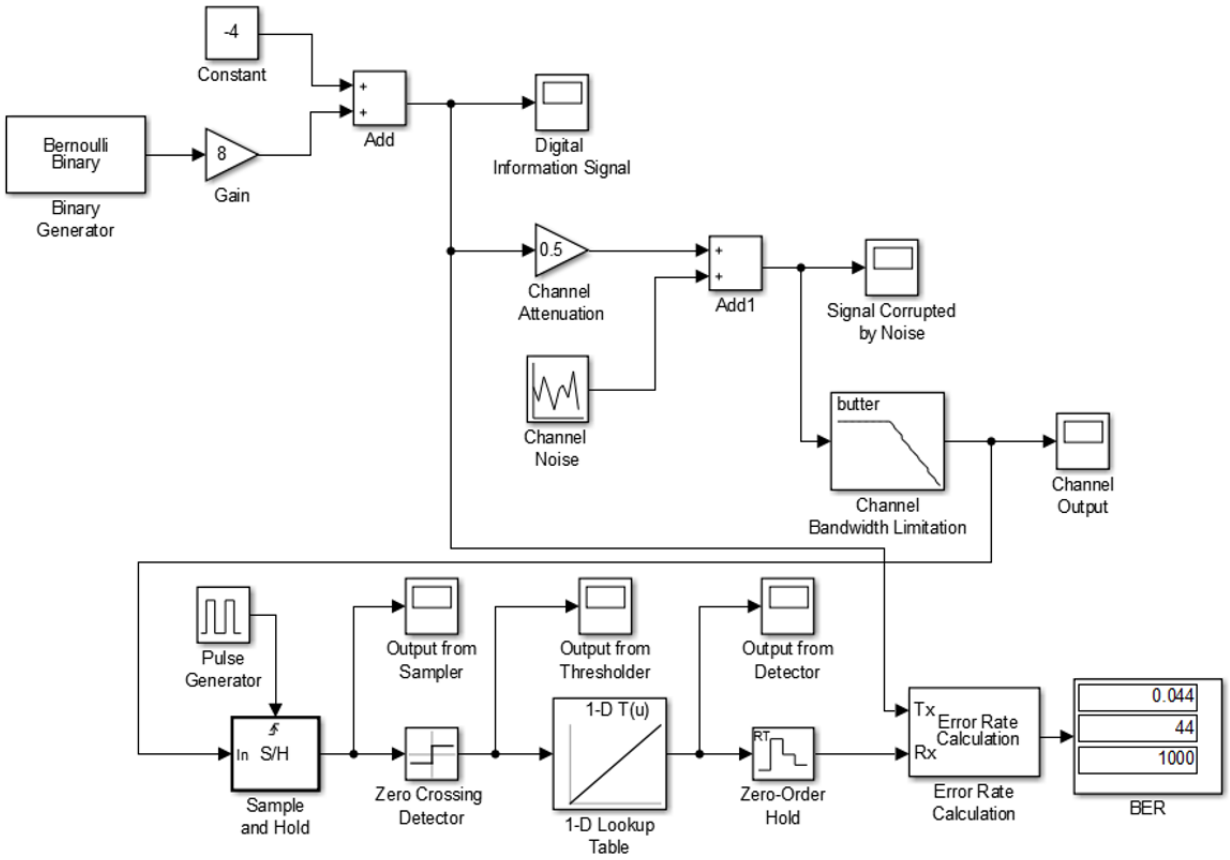



Figure 15.3: Simulink Model for Digital Channel.

- Channel noise [**Random Source** block in **DSP System Toolbox** > **Sources**]
  - Source type: Gaussian
  - Mean value: 0
  - Variance: 1
  - Sample time: 0.01 s
- Channel bandwidth limitation [**Analog Filter Design** block in **DSP System Toolbox** > **Filtering** > **Filter Implementations**]
  - Design method: Butterworth
  - Filter type: Lowpass
  - Filter order: 10
  - Passband edge frequency: 1000 rad/s
- Pulse Generator [**Pulse Generator** block in **Simulink** > **Sources**]
  - Pulse type: Time based
  - Time: Use simulation time
  - Amplitude: 1
  - Period: 0.1 s
  - Pulse width: 50
  - Phase delay: 0 s

- Sample and Hold [**Sample and Hold** block in **DSP System Toolbox** > **Signal Operations**]
  - Trigger type: Rising edge
  - Initial condition: 0
- Zero crossing detector [**Sign** block in **Simulink** > **Math Operations**]
  - Enable zero crossing detection
  - Sampling time: -1
- Lookup Table [**1-D Lookup Table** block in **Simulink** -> **Lookup Tables**]
  - Table Data: [-4,4]
  - Break Points: [-1,1]
  - Sample time: -1
- Zero-Order Hold [**Zero-Order Hold** block in **Simulink** > **Discrete**]
  - Sample time: 0.1 s
- Bit Error Rate Calculator [**Error Rate Calculation** block in **Communication Systems Toolbox** > **Comm Sinks**]
  - Receive delay: 1 s
  - Computation delay: 0 s
  - Output Data: Port
- BER [**Display** block in **Simulink** > **Sinks**]

**Task 168.** *Set the simulation time to 100s and run the simulation. Observe the signal at each scope output and identify the operation at each stage. Explain how the BER is calculated.*

## 15.3 Analyzing the Analog Channel

Answer the following questions using the simulink model developed in section 15.2.1.

**Task 169.** *Complete the Table in the Task Sheet by changing the channel noise variance as given.*

**Task 170.** *Complete the Table in the Task Sheet by changing the channel bandwidth as given. Keep the noise variance as 1 for all cases.*

**Task 171.** *Discuss the reasons for the observed variation of SNR with the channel bandwidth.*

**Task 172.** *Discuss the observed variation of SNR with the information signal amplitude and its relevance in signal transmission over communication channels in practice.*

**Task 173.** *It is often required to transmit composite analog signals through transmission channels. To evaluate the impact of noise and other factors on such applications, replace the previously transmitted analog signal to the following signal by using suitable blocks available in Simulink. Show your work to an instructor.*

$$x(t) = \sum_{k=0}^{4} (10 - 2k) \sin\left((200 + 20k)t\right). \tag{15.1}$$

**Task 174.** *Include the time domain behavior of the new composite signal.*

**Task 175.** *Adjust the passband of the Butterworth bandpass filter for information extraction and stopband of the Butterworth bandstop filter for noise extraction. What are the new cutoff frequencies for passband and stop band of the two filters for information extraction and noise extraction?*

**Task 176.** *How do you adjust the cutoff frequencies for passband and stop band of the two filters for information extraction and noise extraction? Elaborate your answer from the results obtainable from the simulations.*

## 15.4 Analyzing the Digital Channel

Answer the following questions using the simulink model developed in section 15.2.2.

**Task 177.** *Complete the Table in the Task Sheet by changing the channel noise variance as given.*

**Task 178.** *Discuss the reasons for the observed variation of BER with noise variance.*

**Task 179.** *Plot the BER with respect to signal to noise ratio (Eb/No).*

## 15.5 Modelling the Bandlimiting Effect of a Communication Channel Using Hardware

In this section we will be modelling the bandlimiting effect of a communication channel using hardware.

- Low-pass channel - filters the frequencies above a particular cutoff
- High-pass channel - filters the frequencies above a particular cutoff
- Band-pass channel - allows frequencies in a particular range to pass through while filtering the other frequencies
- Band-stop channel - filters frequencies in a particular range

### 15.5.1 Signal Transmission Through a Low-Pass Channel

We will first implement a low pass channel using hardware. Implement the circuit shown in Figure 15.4.

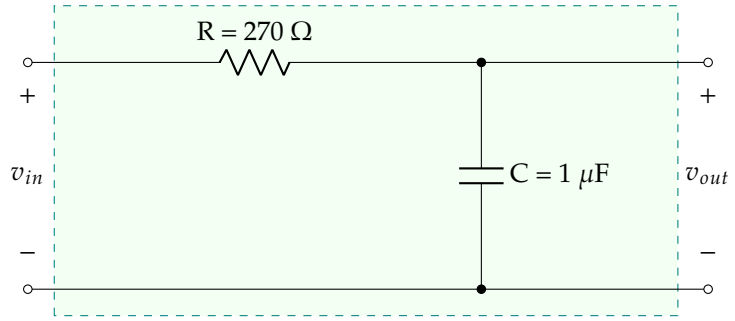



Figure 15.4: Low-Pass Channel

Connect the signal generator to $v_{\text{in}}$ and the two channels of the oscilloscope to $v_{\text{in}}$ and $v_{\text{out}}$.

**Task 180.** *Select the sinusoidal mode of the signal generator and apply 1V peak-to-peak signals having frequencies as shown in the Table in the Task Sheet. Complete the Table by taking measurements of $v_{out}$. Please note that you need to keep $v_{in}$ at 1V peak-to-peak in each measurement.*

**Task 181.** *Plot the output voltage vs. Log (frequency) on the following graph, complete the curve and write down the cut-off frequency obtained from graph.*

**Task 182.** *What is the obtained cut-off frequency?*

**Task 183.** *Find the theoretical cut-off frequency of this channel and compare with the value obtained by the above graph.*

### 15.5.2 Distortion in Signals When Transmitted Through Low-Pass Channels

The signals are subjected to distortion (change from the original signal) as a result of passing through channels. This is due to the partial or full loss of signal energy in certain frequencies due to bandwidth limitation. In this section we will try to observe the distortion.

Select the square wave mode of the signal generator. Set the peak-to-peak input voltage to 2V, and increase the frequency from 0 while looking at the output on the oscilloscope.

**Task 184.** *Complete the Table in the Task Sheet with sketches showing relative amplitudes correctly.*

**Task 185.** *What is the frequency at which you begin to see distortion in the shape of the output signal.*

### 15.5.3 Signal Transmission Through a Band-Pass Channel

In this section we will implement a band pass channel using hardware. Implement the circuit shown in Figure 15.5.

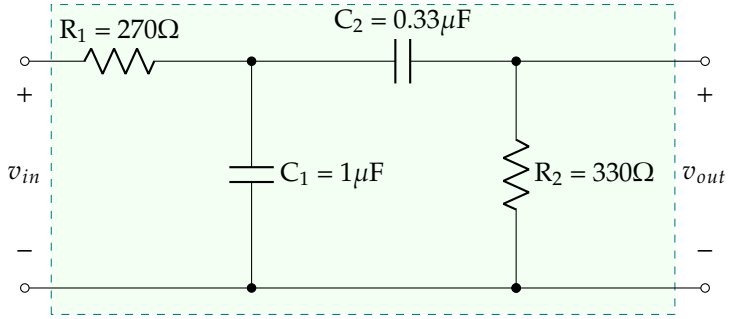



Figure 15.5: Band-Pass Channel

Connect the signal generator to the input and select the sinusoidal mode and apply 1V peak-to-peak signal.

**Task 186.** *Measure the output voltages $v_{out}$ at the frequencies shown in the Table in the Task Sheet. Please note that you need to keep $v_{in}$ at 1V peak-to-peak in each measurement.*

**Task 187.** *Plot the output voltage vs. Log (frequency) on the following graph, complete the curve and write down the lower cut-off frequency and upper cut-off frequency obtained from the graph.*

**Task 188.** *What is the obtained lower cut-off frequency?*

**Task 189.** *What is the obtained upper cut-off frequency?*

**Task 190.** *Find the theoretical Lower cut-off and Upper cut-off frequencies and bandwidth of the channel.*

**Task 191.** *What is the frequency of resonance?*

### 15.5.4 Distortion in Signals When Transmitted Through Band-Pass Channels

Select the square wave mode of the signal generator, set the input signal amplitude to 1V peak-to-peak, and vary the frequency in both directions from the frequency at which you got the resonance and observe the output.

**Task 192.** *Complete the Table in the Task Sheet with sketches showing relative amplitudes correctly.*

♣ The End ♣