Department of Electrical and Electronics

Lab Assignment 6

Devansh Tanna
Roll: *2019A3PS0158P*
Email: *f20190158@pilani.bits-pilani.ac.in*

Course: *Communication System*
Submission date: *September 30, 2021*

Contents

Python Task 1

— Generate three message signals as $m_1(t) = \cos(2\pi Nt)$, $m_2(t) = 2N\text{sinc}(2N\pi t)$, and raised cosine pulse$m_3(t) = 200\frac{\cos(\pi 200t)}{1-4000t^2}\text{sinc}(\pi 200t)$ each with duration one second. Randomly select one of the message signals and use DSB-SC to modulate the carrier of frequency 1 KHz and amplitude 2 Volts. Transmit the DSB-SC signal over a band-limited channel of appropriate bandwidth. Add AWGN $n(t) \sim (0, 0.01)$. Demodulate the signal using synchronous detector. Synchronous detector is implemented by multiplying the received signal with carrier signal followed by a low pass filter. Plot the modulated and demodulated signal both in time and frequency domain using the real time code for 30 seconds. Take N as the sum of the last three digits of your BITS ID.

Code

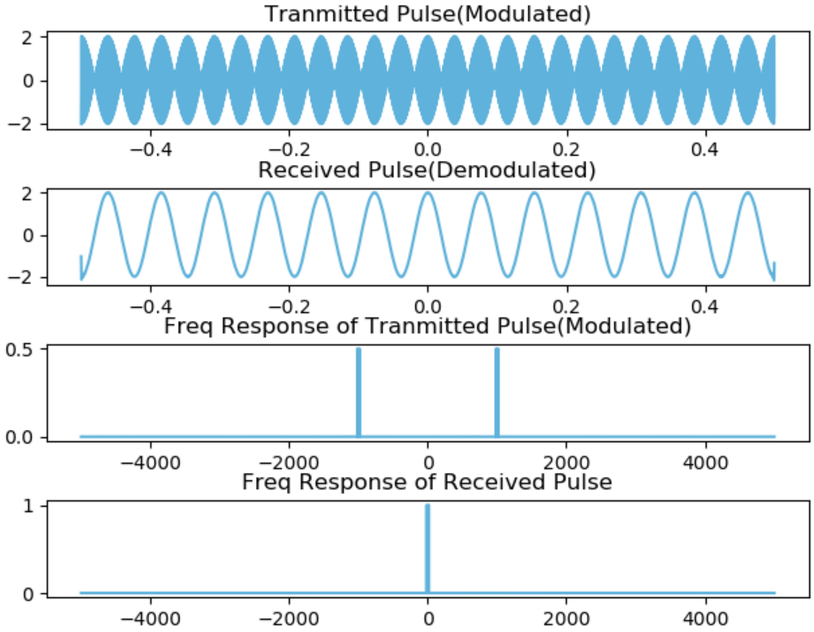
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl

mpl.style.use('default')
# N = 2
N = 13


def m1(t):
    return np.cos(2*np.pi*N*t)

def m2(t):
    return 2*N*np.sinc(2*N*t)

def m3(t):
    beta = 1
    Rb = 200
    return Rb*np.cos(np.pi*beta*Rb*t)/(1-np.square(2*beta*Rb*t))*np.sinc
   (Rb*t)

# fs = 1e2
fs = 1e4
# fc = 10
fc = 1e3
B = fc
t_car = np.arange(-0.5, 0.5, 1/fs)
carrier = 2*np.cos(2*np.pi*fc*t_car)
sig_t = np.array([])
sig_rx = np.array([])
h_filter = 2*B*np.sinc(2*B*t_car)
ft = np.random.choice([m1, m2, m3])

# Packet
# Sig_t
# Sig_f

fig, ax = plt.subplots(4, 1)
fig.tight_layout()
```

```
for j in range(30):
    mtx = ft(t_car)*carrier + 0.01*np.random.randn(int(fs))
    Ns = mtx.size
    # mtx = m_t*carrier

    sig_t = np.append(sig_t, mtx)
    mrx = np.convolve(mtx*carrier, h_filter, 'same')/Ns
    sig_rx = np.append(sig_rx, mrx)


    # plt.plot(sig_t)
    # plt.xlim([0, 29*fs])
    # with plt.xkcd():
    freq = np.linspace(-fs / 2, fs / 2, Ns)
    # testx = ft(t_car)
    sig_tx_f = np.fft.fftshift(np.abs(np.fft.fft(mtx)) / Ns)
    sig_rx_f = np.fft.fftshift(np.abs(np.fft.fft(mrx)) / Ns)
    # fig.clear()
    ax[0].clear()
    ax[1].clear()
    ax[2].clear()
    ax[3].clear()
    ax[2].plot(freq, sig_tx_f, color='#5eb2db')
    ax[2].set_title("Freq Response of Tranmitted Pulse(Modulated)")
    # ax[2].
    ax[0].plot(t_car, mtx, color='#5eb2db')
    ax[0].set_title("Tranmitted Pulse(Modulated)")
    ax[1].plot(t_car, mrx, color='#5eb2db')
    ax[1].set_title("Received Pulse(Demodulated)")
    # ax[1].set_xlim([0, 5*fs])
    ax[3].plot(freq, sig_rx_f, color='#5eb2db')
    ax[3].set_title("Freq Response of Received Pulse")
    plt.pause(0.5)

# fig.savefig("Task1.png")
```

Results

Here we have transmitted sine pulse which is modulated by carrier of frequency f_c. So for demodulation we will multiply again by carrier signal which shifts signal at $f = 0$ and there will be 2 components at $f = 2f_c$ which can remove by Low Pass Filter of bandwidth f_c or B(Bandwidth of $m(t)$).

By running code repeatedly we can check for different pulses.



Python Task 2

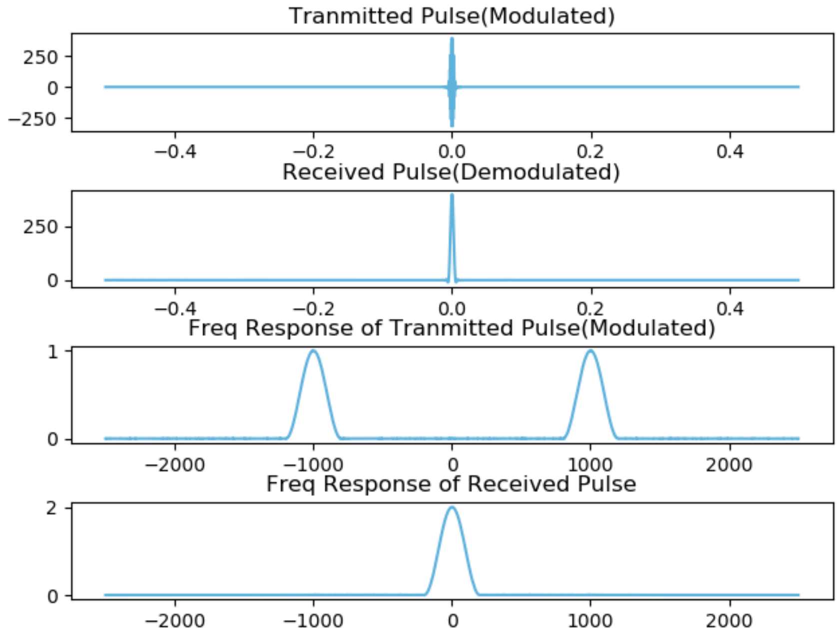
Repeat the task 1 if the demodulation is done using envelope detector. The envelope detector is implemented using the MATLAB function hilbert. Say, the received modulated signal is x_t, then use $y_t = \text{hilbert}(x_t) \cdot e^{-j2\pi f_c t}$. The exponential signal is multiplied to shift the modulated signal from bandpass to low pass.

Code

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from scipy.signal import hilbert

mpl.style.use('default')
# N = 2
N = 13


def m1(t):
    return np.cos(2*np.pi*N*t)

def m2(t):
    return 2*N*np.sinc(2*N*t)

def m3(t):
    beta = 1
    Rb = 200
    return Rb*np.cos(np.pi*beta*Rb*t)/(1-np.square(2*beta*Rb*t))*np.sinc
   (Rb*t)

fs = 0.5e4
fc = 1e3
```

```
B = fc
t_car = np.arange(-0.5, 0.5, 1/fs)
carrier = 2*np.cos(2*np.pi*fc*t_car)
sig_t = np.array([])
sig_rx = np.array([])
h_filter = 2*B*np.sinc(2*B*t_car)
ft = np.random.choice([m1, m2, m3])

# Packet
# Sig_t
# Sig_f

fig, ax = plt.subplots(4, 1)
fig.tight_layout()
# plt.subplots_adjust(wspace=0.3, hspace=1.2)
for j in range(30):
    mtx = ft(t_car)*carrier + 0.1*np.random.randn(int(fs))
    Ns = mtx.size
    # mtx = m_t*carrier

    sig_t = np.append(sig_t, mtx)
    mrx = np.real(hilbert(mtx)*np.exp(-1j*2*np.pi*fc*t_car))

    sig_rx = np.append(sig_rx, mrx)

    # plt.plot(sig_t)
    # plt.xlim([0, 29*fs])
    # with plt.xkcd():
    freq = np.linspace(-fs/2, fs/2, Ns)
    # testx = ft(t_car)
    sig_tx_f = np.fft.fftshift(np.abs(np.fft.fft(mtx))/Ns)
    sig_rx_f = np.fft.fftshift(np.abs(np.fft.fft(mrx))/Ns)
    # fig.clear()
    ax[0].clear()
    ax[1].clear()
    ax[2].clear()
    ax[3].clear()
    ax[2].plot(freq, sig_tx_f, color='#5eb2db')
    ax[2].set_title("Freq Response of Tranmitted Pulse(Modulated)")
    # ax[2].
    ax[0].plot(t_car, mtx, color='#5eb2db')
    ax[0].set_title("Tranmitted Pulse(Modulated)")
    ax[1].plot(t_car, mrx, color='#5eb2db')
    ax[1].set_title("Received Pulse(Demodulated)")
    # ax[1].set_xlim([0, 5*fs])
    ax[3].plot(freq, sig_rx_f, color='#5eb2db')
    ax[3].set_title("Freq Response of Received Pulse")
    fig.show()

    plt.pause(0.5)

fig.savefig("Task2.png")
```

Results



As we can see, first Raised Cosine Pulse has been tranmitted with carrier frequency 1 kHz. For receiving, we used hilbert transform function. which gives analytical signal as output, i.e. $H(x_t) = m(t) \cdot e^{j2\pi f_c t}$ after which we can use frequency translation to obtain the message signal $m(t)$. So , we used envelope detection for removing higher frequency component using Hilbert Transform.

By running code, different times we can obtain different transmitting pulses.