Text

Description automatically generated with medium confidenceDigital Communication Systems

**Laboratory Report**

Fall 2021

|  |  |
| --- | --- |
| Laboratory Number: | **05** |
| Laboratory Title: | **Passband Transmission and Detection Pt. 1** |
| Full Name: | **Robert Louis Bara** |
| TUID: | **915614617** |

**Description:**

This lab examines Passband Transmission and Detection techniques and serves as part 1 to a multi-week lab. Passband modulation extends off of baseband modulation, however in passband modulation, the wave spectrum uses a carrier frequency to be transmitted at a higher frequency represented by,

From this, the signal itself can be encoded using either Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), or Phase Shift Keying (PSK). Detection and decoding are simply the inverse of the modulation by finding the indices of the transmitted symbols and detecting if the message has been transmitted. An LTI implementation known as the matched filter can be used to find the coefficients of the threshold for decoding and filter out the excess,

Diagram

Description automatically generated

Figure . Detection of coefficients where the received signal is r(t), base encoded signal is s(t) and the coefficients are a(t)

This week’s lab will focus on using a 4-bit ASK system to detect a signal and carrier frequency based upon my TUID. 1000 random samples will then be generated and modulated with white gaussian noise, and the signal to noise ratio will be computed for comparison in next week’s lab.

**Images:**

Below is the results after finding the thresholds for signal detection of the 4-ASK.

Graphical user interface

Description automatically generated with low confidence

Figure 2. Command Line output after finding threshold levels

**Numerical Tables:**

Table 1. 4-ASK table with variable SNR vs Mean Absolute Error

|  |  |
| --- | --- |
| 4-ASK | |
| SNR | Mean Absolute Error |
| 20 | 0 |
| 16 | 0 |
| 8 | 0 |
| 4 | 0 |
| 2 | 0 |
| 0 | 0 |
| -2 | 0 |
| -4 | 0 |
| -8 | 0 |
| -16 | 0.003 |
| -20 | 0.039 |

**Code:**

### Section 01

The initial parameters are usually defined at the beginning of the program.

clc; clear;

%tuid 915614617

A = 8; % Signal amplitude

rb = 20; % Fundamental frequency of signal

Tb = 1 / rb; % Period of signal

fc = (20+6) \* rb;

Tc = 1 / fc;

fs = 100 \* fc; % Sampling frequency

Ts = 1 / fs; % Sampling period

### Section 05

#### ASK Correlator Decoder and Detection

Since only a single sinusoid has used for encoding, a single correlator will be enough for detection.

Shape

Description automatically generated with medium confidence

% decoding

rt = demodulated;

Ns = round(fc / rb);

dec\_signal = zeros([1, floor(length(rt) \* rb / fc)]);

dcounter = 1;

for n = Ns:Ns:length(rt)

r\_hat\_t = rt(n-Ns+1:n) .\* sin(2\*pi\* (fc/rb) \* (0:Ns-1)/Ns \* Tc/Tb);

dec\_signal(dcounter) = sum(r\_hat\_t)\*(Tc/Tb)\*2;

dcounter = dcounter + 1;

end

figure()

stem(dec\_signal); xlabel('symbols index'); ylabel('symbols amplitude'); title('detected symbols amplitude');

% detection

% finding the unique levels

levels = [-inf, unique(round(dec\_signal, 2)), inf]

% levels = [-inf, -0.25, -0.22, -0.19, -0.16, -0.13, -0.09, -0.06, -0.03, 0.03, 0.06, 0.09, 0.13, 0.16, 0.19, 0.22, 0.25, inf];

% finding the thresholds

thresholds = (levels(2:end) + levels(1:end-1))/2;

detected\_indices = zeros([1, length(dec\_signal)]);

for n = 1:length(dec\_signal)

if length(thresholds(thresholds == dec\_signal(n))) == 0

lvl = sort([thresholds, dec\_signal(n)]);

end

indices = 1:length(lvl);

detected\_indices(n) = indices(lvl == dec\_signal(n))-2;

end

figure()

stem(detected\_indices); xlabel('symbols index in the message stream'); ylabel('symbols index'); title('detected symbols indices');

% computing error rates

err = mean(msg ~= detected\_indices)

### Section 06

#### Noise

Noise is the impact of channel that signal transfer through.

The most common way to find the noise impact is Additive White Gaussian Noise (AWGN) model. In this model, the noise will be generted by a Gaussian distribution and added to the transmitted signal.

% generation of a random message with fixed length

rng(0);

Nm = 1000;

msg = randi(M-1, [1, Nm]); % message: a set of symbols

figure(); stem(msg);

msg\_amplitude = symbol\_amplitude(symbol\_set(msg+1)+1); % mepping the symbol to the amplitude

amplitudes = rate\_transition(msg\_amplitude, rb, fs); % translate rate 1 to rate rb

t = (0:length(amplitudes)-1)\*Ts; % make the parameter time

sin\_signal = 1 \* sin(2 \* pi \* rb \* t + 0); % simple sinusoid wave

signal = amplitudes .\* sin\_signal; % message multiply by sinusoid

figure(); plot(signal);

carrier = 1 \* cos(2 \* pi \* fc \* t + 0);

modulated = signal .\* carrier;

figure()

subplot(2, 1, 1); plot(t, carrier); xlabel('time'); ylabel('amplitude'); ylim([-A-0.1, +A+0.1]); title('carrier');

subplot(2, 1, 2); plot(t, modulated); xlabel('time'); ylabel('amplitude'); ylim([-A-0.1, +A+0.1]); title('Amplitude Shift Keying (ASK)');

snr = -20;

noisy\_signal = awgn(modulated, snr, 'measured');

figure(); plot(noisy\_signal);

% demodulation

rt = noisy\_signal;

Ns = round(fs / fc);

demodulated = zeros([1, floor(length(rt) \* fc/fs)]);

dcounter = 1;

for n = Ns:Ns:length(rt)

r\_hat\_t = rt(n-Ns+1:n) .\* cos(2\*pi\*fc\*(0:Ns-1)\*Ts);

demodulated(dcounter) = sum(r\_hat\_t)\*Ts\*2/Tc;

dcounter = dcounter + 1;

end

figure()

plot(demodulated); xlabel('time'); ylabel('amplitude'); title('demodulated signal');

% decoding

rt = demodulated;

Ns = round(fc / rb);

dec\_signal = zeros([1, floor(length(rt) \* rb / fc)]);

dcounter = 1;

for n = Ns:Ns:length(rt)

r\_hat\_t = rt(n-Ns+1:n) .\* sin(2\*pi\* (fc/rb) \* (0:Ns-1)/Ns \* Tc/Tb);

dec\_signal(dcounter) = sum(r\_hat\_t)\*(Tc/Tb)\*2;

dcounter = dcounter + 1;

end

figure()

stem(dec\_signal); xlabel('symbols index'); ylabel('symbols amplitude'); title('detected symbols amplitude');

% detection

detected\_indices = zeros([1, length(dec\_signal)]);

for n = 1:length(dec\_signal)

if length(thresholds(thresholds == dec\_signal(n))) == 0

lvl = sort([thresholds, dec\_signal(n)]);

end

indices = 1:length(lvl);

detected\_indices(n) = indices(lvl == dec\_signal(n))-2;

end

figure()

stem(detected\_indices); xlabel('symbols index in the message stream'); ylabel('symbols index'); title('detected symbols indices');

% computing error rates

err = mean(msg ~= detected\_indices)