Communication System Lab 6 Part 2

By: Saurabh Kumar (SC22B146)

3 Basic passband demodulation - BPSK

Redoing previous part (Section 1)

```
rng('default');

N = 1000;
T = 1;
fs = 1000;
n = T*fs;
% Bits and Signal generation
source = rand(1, N) > 0.5;

signal = zeros(1,n*N);
for i=1:(length(source))
 signal((i-1)*n+1:(i)*n) = repelem(source(i),n);
end
signal(signal==0) = -1;

% Passband channel
channel = firpm(200, [0, 18, 20, 30, 32, fs/2]/(fs/2),[0, 0, 1, 1, 0, 0]);

fc = 25;
t = (0:length(signal)-1)/fs;
% Modulated signal
mod_signal = cos(2*pi*fc*t).*signal;

receive_signal = conv(mod_signal,channel,'same');


% Noise
noise_var = 0.1;
noise_stddev = sqrt(noise_var);
noise = noise_stddev * randn(1,length(receive_signal));
% Received Signal
receive_signal_noisy = receive_signal + noise;

% Bandpass receive filter
receiver_filter = firpm(200, [0, 18, 20, 30, 32, fs/2]/(fs/2),[0, 0, 1, 1,
0, 0]);

% Filtered signal
filtered_signal = conv(receive_signal_noisy,receiver_filter,'same');
```

1. Implement a coherent demodulator for the received BPSK signal that you have obtained at the output of the bandpass receive filter in Task 10 of Section 1. Please show to the TAs how you have made sure that the demodulator's (more precisely the local oscillator signal's) phase matches with that of the received carrier.

2. Implement a sampler for sampling the output of the demodulator. Then implement a decision making device - a thresholder that will convert the samples to 0 or 1, and obtain the bit estimates at the output of the decision device.

3. Find out the bit error rate for N = 1000 for σ2 = 0.1.

BPSK Demodulator

```
samples_per_symbol_time = n;
samples_per_second = fs;
fc = 25;
signal_points = [1 0; -1 0];

decoded_symbols = correlation_receiver(filtered_signal,
samples_per_symbol_time, samples_per_second, fc, signal_points);
% correlation_receiver function does the complete demodulation -
% sampling, decision making

% 1 -> bit 1, 2 -> bit 0
decoded_bits = decoded_symbols;
decoded_bits(decoded_symbols==2) = 0;
BER = sum(decoded_bits ~= source)
```

```
BER =
0
```

Inference: We get zero BER because of very small variance value.

Demodulator function for BPSK:

```
function decoded_symbols = correlation_receiver(signal,
samples_per_symbol_time, samples_per_second, fc, signal_points)
    frequency_offset = 0;
    phase_offset = 0;

    num_symbols = floor(length(signal)/samples_per_symbol_time);
    signal = signal(1:num_symbols * samples_per_symbol_time);
    signal = reshape(signal, samples_per_symbol_time, num_symbols)';

    constellation = [];
    basis_cos = cos(phase_offset + 2 * pi * (fc + frequency_offset) *
[0:samples_per_symbol_time - 1]/samples_per_second);
    basis_sin = sin(phase_offset + 2 * pi * (fc + frequency_offset) *
[0:samples_per_symbol_time - 1]/samples_per_second);
    for i = 1:num_symbols
        signal_cos_component = sum(signal(i, :) .* basis_cos) /
samples_per_symbol_time;
```

```
        signal_sin_component = sum(signal(i, :) .* basis_sin) /
samples_per_symbol_time;
        constellation = [constellation; [signal_cos_component,
signal_sin_component]];
    end

    number_of_symbols = size(constellation, 1);
    decoded_symbols = zeros(1, number_of_symbols);
    signal_constellation_size = size(signal_points, 1);

    for i = 1:number_of_symbols
        t = repmat([constellation(i, 1), constellation(i, 2)],
signal_constellation_size, 1);
        t = (t - signal_points).^2;
        t = sum(t, 2);
        [~, decoded_symbols(i)] = min(t);
    end
end
```

4 General passband demodulation

1. Fix a sequence of 40 bits.

2. For this fixed sequence of bits generate QPSK, 16-QAM, 8-PSK, and 4-FSK passband waveforms.

3. Use the correlation receiver that you have implemented to receive and convert the above waveforms into bits.

4. Show the TA that the received sequence of bits is the same as the transmitted sequence of bits (since we are not modelling channel noise).

```
rng('default');

% Parameters
fs = 1000;
T = 1;
fc = 25;
samples_per_symbol = fs * T;
t = (0:samples_per_symbol-1)/fs;

% Fixed 40-bit sequence
fixed_bits = randi([0 1], 1, 40) % 40 bits sequence
```

```
fixed_bits = 1×40
     1     1     0     1     1     0     0     1     1     1     0     1     1···
```

QPSK

```
qpsk_map = [ 1  1;
            -1  1;
            -1 -1;
             1 -1] / sqrt(2);
```

```
% Reshape bits
bits_reshaped = reshape(fixed_bits, 2, []).';
indices_qpsk = bi2de(bits_reshaped, 'left-msb'); % 0 to 3
symbols_qpsk = qpsk_map(indices_qpsk+1, :);

% QPSK passband waveform
tx_waveform_qpsk = [];
for i = 1:size(symbols_qpsk, 1)
    i_comp = symbols_qpsk(i,1) * cos(2*pi*fc*t);
    q_comp = symbols_qpsk(i,2) * sin(2*pi*fc*t);
    symbol_wave = i_comp + q_comp;
    tx_waveform_qpsk = [tx_waveform_qpsk, symbol_wave];
end

% Decode QPSK using correlation receiver (pass in qpsk_map scaled to full
amplitude)
decoded_sym_qpsk = correlation_receiver_new(tx_waveform_qpsk,
samples_per_symbol, fs, fc, qpsk_map);
% Convert decoded symbol indices into bits (2 bits per symbol)
decoded_bits_qpsk = de2bi(decoded_sym_qpsk - 1, 2, 'left-msb');
decoded_bits_qpsk = reshape(decoded_bits_qpsk.', 1, []);  % serialize rowwise

fprintf('QPSK: BER = %.4f\n', sum(decoded_bits_qpsk ~= fixed_bits)/
length(fixed_bits));
```

```
QPSK: BER = 0.0000
```

```
assert(isequal(fixed_bits, decoded_bits_qpsk), 'QPSK decoding failed!');

disp('Transmitted bits:');
```

```
Transmitted bits:
```

```
disp(reshape(fixed_bits, 2, []));
```

```
     1     0     1     0     1     0     1     1     0     1     1     1     1     1     1     1     0
     1     1     0     1     1     1     0     0     1     1     0     1     1     0     0     0     0
```

```
disp('Received bits:');
```

```
Received bits:
```

```
disp(reshape(decoded_bits_qpsk, 2, []));
```

```
     1     0     1     0     1     0     1     1     0     1     1     1     1     1     1     1     0
     1     1     0     1     1     1     0     0     1     1     0     1     1     0     0     0     0
```

16-QAM

```
qam_map = [ -3 -3;
            -3 -1;
            -3  1;
            -3  3;
            -1 -3;
```

```
            -1 -1;
            -1  1;
            -1  3;
             1 -3;
             1 -1;
             1  1;
             1  3;
             3 -3;
             3 -1;
             3  1;
             3  3] / sqrt(10);

% Reshape bits for 16-QAM
bits_reshaped_qam = reshape(fixed_bits, 4, []).';
indices_qam = bi2de(bits_reshaped_qam, 'left-msb');  % 0 to 15
symbols_qam = qam_map(indices_qam+1, :);

% 16-QAM waveform
tx_waveform_qam = [];
for i = 1:size(symbols_qam, 1)
    i_comp = symbols_qam(i,1) * cos(2*pi*fc*t);
    q_comp = symbols_qam(i,2) * sin(2*pi*fc*t);
    symbol_wave = i_comp + q_comp;
    tx_waveform_qam = [tx_waveform_qam, symbol_wave];
end

decoded_sym_qam = correlation_receiver_new(tx_waveform_qam,
samples_per_symbol, fs, fc, qam_map);
decoded_bits_qam = de2bi(decoded_sym_qam - 1, 4, 'left-msb');
decoded_bits_qam = reshape(decoded_bits_qam.', 1, []);

fprintf('16-QAM: BER = %.4f\n', sum(decoded_bits_qam ~= fixed_bits)/
length(fixed_bits));
```

```
16-QAM: BER = 0.0000
```

```
assert(isequal(fixed_bits, decoded_bits_qam), '16-QAM decoding failed!');
disp('Transmitted bits:');
```

```
Transmitted bits:
```

```
disp(reshape(fixed_bits, 2, []));
```

```
     1     0     1     0     1     0     1     1     0     1     1     1     1     1     1     0
     1     1     0     1     1     1     0     0     1     1     0     1     1     0     0     0
```

```
disp('Received bits:');
```

```
Received bits:
```

```
disp(reshape(decoded_bits_qam, 2, []));
```

```
     1     0     1     0     1     0     1     1     0     1     1     1     1     1     1     1     0
     1     1     0     1     1     1     0     0     1     1     0     1     1     0     0     0     0
```

8-PSK

```
num_bits = length(fixed_bits);
r = mod(num_bits, 3);
if r ~= 0
    pad_length = 3 - r;
    fixed_bits_psk = [fixed_bits, zeros(1, pad_length)];
else
    fixed_bits_psk = fixed_bits;
end
bits_reshaped_psk = reshape(fixed_bits_psk, 3, []).';
indices_psk = bi2de(bits_reshaped_psk, 'left-msb'); % 0 to 7

angles = 2*pi*(0:7)/8;
psk_map = [cos(angles)', sin(angles)'];
symbols_psk = psk_map(indices_psk+1, :);

% Waveform
tx_waveform_psk = [];
for i = 1:size(symbols_psk, 1)
    i_comp = symbols_psk(i,1) * cos(2*pi*fc*t);
    q_comp = symbols_psk(i,2) * sin(2*pi*fc*t);
    symbol_wave = i_comp + q_comp;
    tx_waveform_psk = [tx_waveform_psk, symbol_wave];
end

decoded_sym_psk = correlation_receiver_new(tx_waveform_psk,
samples_per_symbol, fs, fc, psk_map);
decoded_bits_psk = de2bi(decoded_sym_psk - 1, 3, 'left-msb');
decoded_bits_psk = reshape(decoded_bits_psk.', 1, []);

decoded_bits_psk = decoded_bits_psk(1:num_bits);
fprintf('8-PSK: BER = %.4f\n', sum(decoded_bits_psk ~= fixed_bits)/num_bits);
```

```
8-PSK: BER = 0.0000
```

```
disp('Transmitted bits:');
```

```
Transmitted bits:
```

```
disp(reshape(fixed_bits, [], 8));
```

```
     1     0     0     0     1     1     1     1
     1     0     1     0     0     1     0     1
     0     1     1     1     1     0     0     0
     1     1     0     1     1     1     0     1
     1     1     1     1     1     0     0     0
```

```
disp('Received bits:');
```

Received bits:

```
disp(reshape(decoded_bits_psk, [], 8));
```

```
     1     0     0     0     1     1     1     1
     1     0     1     0     0     1     0     1
     0     1     1     1     1     0     0     0
     1     1     0     1     1     1     0     1
     1     1     1     1     1     0     0     0
```

4-FSK

```
bits_reshaped_fsk = reshape(fixed_bits, 2, []).';
indices_fsk = bi2de(bits_reshaped_fsk, 'left-msb');  % 0 to 3

fsk_offsets = [-15, -5, 5, 15];
fsk_freqs = fc + fsk_offsets;
fsk_points = zeros(4, samples_per_symbol);
for i = 1:4
    fsk_points(i,:) = cos(2*pi*fsk_freqs(i)*t);
end

% Waveform
tx_waveform_fsk = [];
for i = 1:length(indices_fsk)
    f = fsk_freqs(indices_fsk(i)+1);
    symbol_wave = cos(2*pi*f*t);
    tx_waveform_fsk = [tx_waveform_fsk, symbol_wave];
end

% Decode FSK
decoded_fsk = zeros(1, length(indices_fsk));
num_sym_fsk = length(indices_fsk);
rx_syms_fsk = reshape(tx_waveform_fsk, samples_per_symbol, num_sym_fsk)';

for symIdx = 1:num_sym_fsk
    energies = zeros(1,4);
    for tone = 1:4
        energies(tone) = sum(rx_syms_fsk(symIdx,:) .* fsk_points(tone,:));
    end
    [~, decoded_fsk(symIdx)] = max(energies);
end
decoded_bits_fsk = de2bi(decoded_fsk - 1, 2, 'left-msb');
decoded_bits_fsk = reshape(decoded_bits_fsk.', 1, []);

fprintf('4-FSK: BER = %.4f\n', sum(decoded_bits_fsk ~= fixed_bits)/
length(fixed_bits));
```

```
4-FSK: BER = 0.0000
```

```
assert(isequal(fixed_bits, decoded_bits_fsk), '4-FSK decoding failed!');
```

```
fprintf('All modulations successfully decoded with BER = 0.\n');
```

```
All modulations successfully decoded with BER = 0.
```

```
disp('Transmitted bits:');
```

```
Transmitted bits:
```

```
disp(reshape(fixed_bits, 2, []));
```

```
     1     0     1     0     1     0     1     1     0     1     1     1     1     1     1     0
     1     1     0     1     1     1     0     0     1     1     0     1     1     0     0     0
```

```
disp('Received bits:');
```

```
Received bits:
```

```
disp(reshape(decoded_bits_fsk, 2, []));
```

```
     1     0     1     0     1     0     1     1     0     1     1     1     1     1     1     0
     1     1     0     1     1     1     0     0     1     1     0     1     1     0     0     0
```

Inferences: We get zero error in all these schemes because there was no noise considered and the less number of bits transmiited.

Demodulator function (used for other schemes):

```
function decoded_symbols = correlation_receiver_new(signal,
samples_per_symbol_time, samples_per_second, fc, signal_points)
    frequency_offset = 0;
    phase_offset = 0;

    num_symbols = floor(length(signal) / samples_per_symbol_time);
    signal = signal(1:num_symbols * samples_per_symbol_time);
    signal = reshape(signal, samples_per_symbol_time, num_symbols)';

    basis_cos = cos(phase_offset + 2 * pi * (fc + frequency_offset) *
(0:samples_per_symbol_time - 1)/samples_per_second);
    basis_sin = sin(phase_offset + 2 * pi * (fc + frequency_offset) *
(0:samples_per_symbol_time - 1)/samples_per_second);

    constellation = zeros(num_symbols, 2);
    for i = 1:num_symbols
        signal_cos_component = sum(signal(i, :) .* basis_cos) /
samples_per_symbol_time;
        signal_sin_component = sum(signal(i, :) .* basis_sin) /
samples_per_symbol_time;
        constellation(i, :) = [signal_cos_component, signal_sin_component];
    end

    constellation = 2 * constellation;

    number_of_symbols = size(constellation, 1);
```

```
    decoded_symbols = zeros(1, number_of_symbols);
    signal_constellation_size = size(signal_points, 1);

    for i = 1:number_of_symbols
        diff = repmat(constellation(i, :), signal_constellation_size, 1) -
signal_points;
        dists = sum(diff.^2, 2);
        [~, decoded_symbols(i)] = min(dists);
    end
end
```