564 Digital Signal Processing

Computing Assignment

Module 1

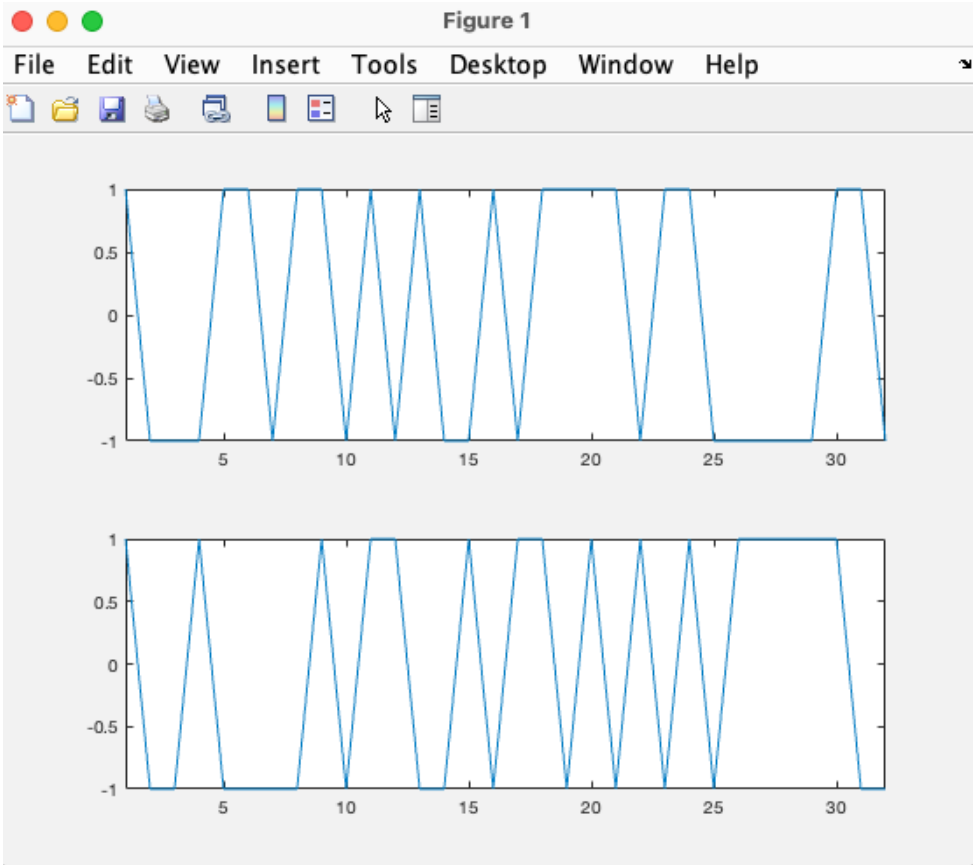
Hao Che Tsai

934412754

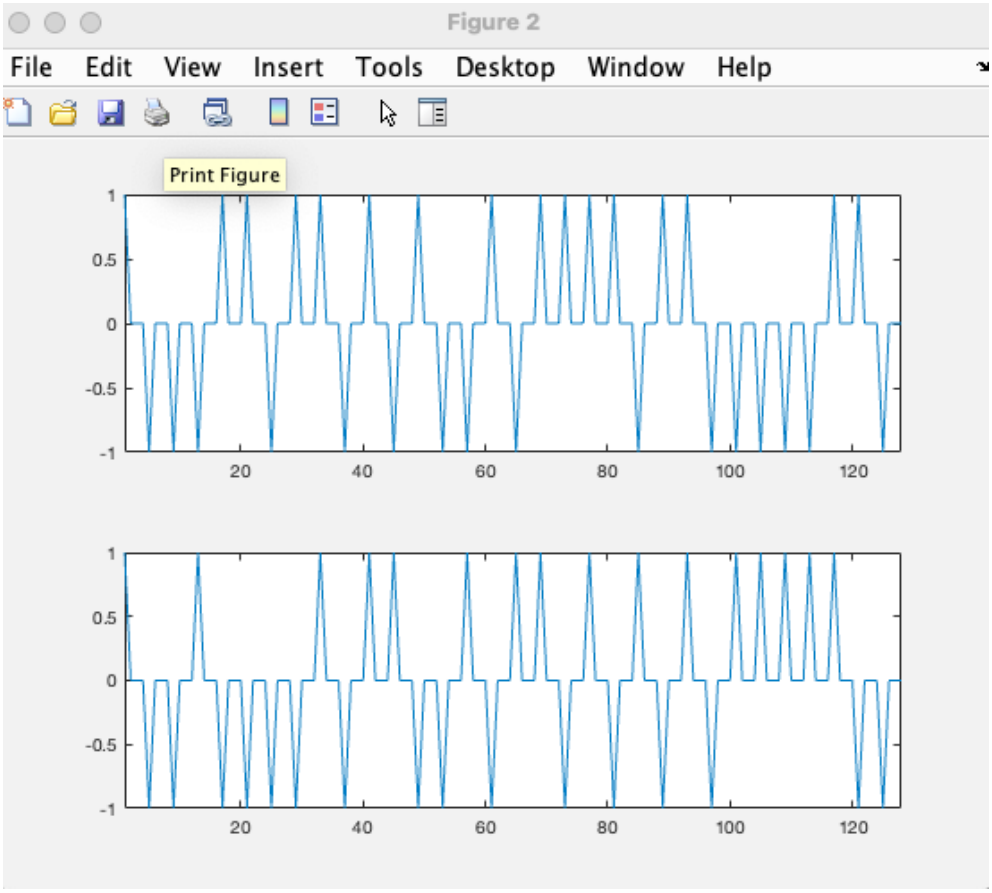
Introduction & Goal

In this computing assignment, the whole program I'm going to do is understanding how a QPSK transmitter system works and using the MATLAB to implement this module. First, I need to generate two sequences of random value between 1 and -1, and next make these two sequences into an up-sampling progress which by the instruction I need to insert 3 zeros between each two random value I generate. After that I need to make these two signals multiply with cosine wave and sine wave respectively. Last, combine the signal I got and that would be the transmitter output signal for this module. At this point, I want to make sure that the waveform figure generate from the two signal is reasonable.

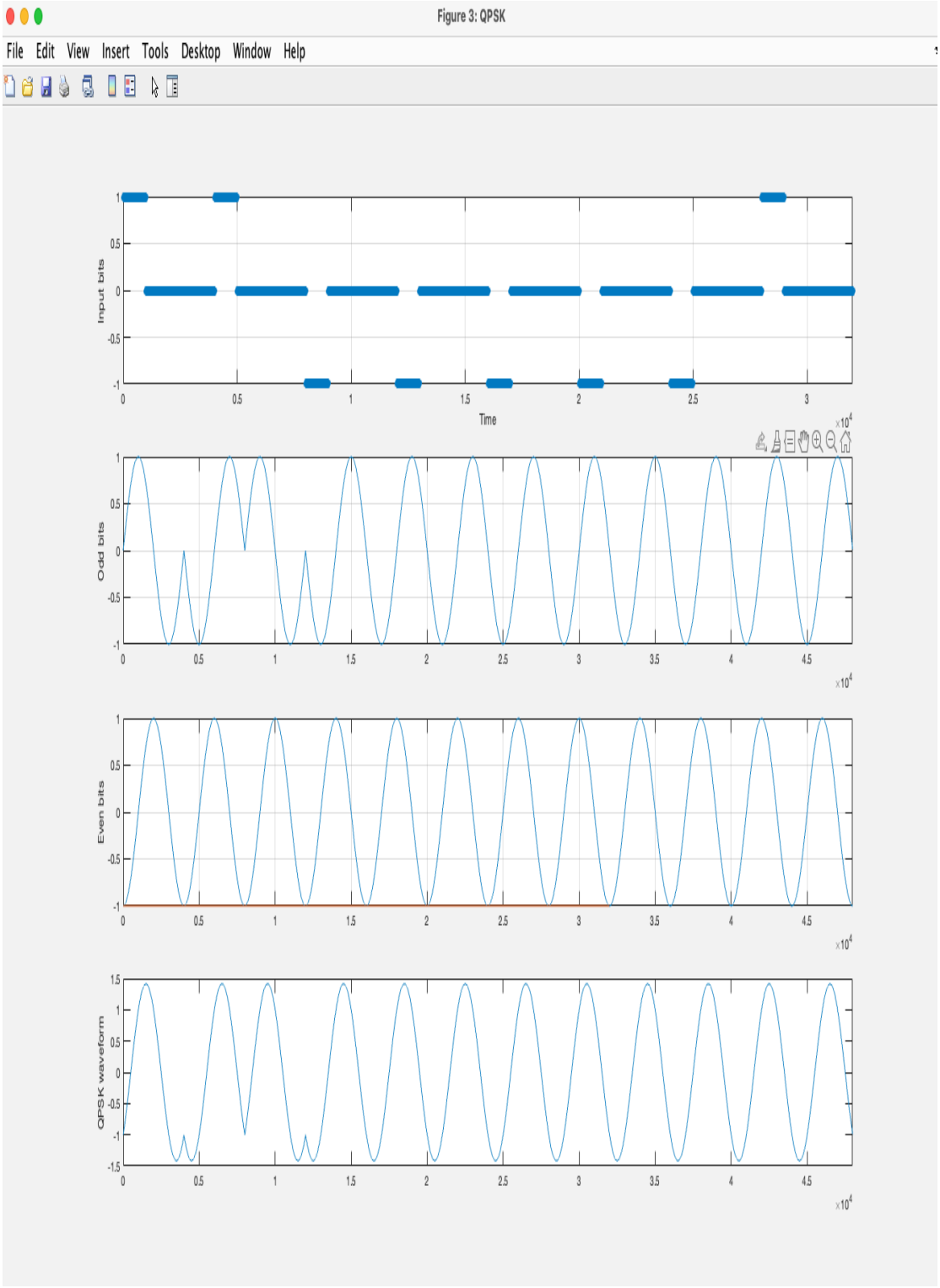
Result



The first figure I generate from this implementation is that I would like to see how the 32 samples long random sequences are like before they turn into the waveform.



Next, this figure is what I want to know how the up-sampling work for the two 32 samples long random sequences after I insert 3 zeros into each of them. By the result shown in the figure, I believe the up-sampling progress worked well and it turn the original signal from 32 samples long to 128 samples long.



The figure in the next page is the outcome waveform of the transmitter I built. For the first chart, it shows what I input to this system exactly. The second and third charts are which both signal multiply by cosine and sine function respectively. The fourth chart is the QPSK signal we would get from the output of the transmitter whenwe add those two waveforms after their multiplication.

Codes:

```
clc;
clear all;

n = 32; %Set the size of the random sequence
x1 = randn([n 1]); % Generating random sequence x1
x2 = randn([n 1]); % Generating random sequence x2

for i = 1:n %Generating b1
    if(x1(i) >= 0) %If x1>=0 then b1=1;
        b1(i) = 1;
    elseif(x1(i) < 0)%If x1<0 then b1=-1;
        b1(i) = -1;
    end;
end;

for i = 1:n %Generating b2
    if(x2(i) >= 0) %If x2>=0 then b2=1;
        b2(i) = 1;
    elseif(x2(i) < 0)%If x2<0 then b2=-1;
        b2(i) = -1;
    end;
end;

b = [b1;b2]; % Make b1 & b2 into an array

figure
subplot(2,1,1);
plot(b1); %Plot outcome of b1
xlim([1 32]);
ylim([-1 1]);

subplot(2,1,2);
plot(b2);%Plot outcome of b2
xlim([1 32]);
ylim([-1 1]);

%Upsmapling
```

```
b1u = upsample(b1,4);
b2u = upsample(b2,4);
bu = upsample(b,4);

figure
subplot(2,1,1);
plot(b1u); %Plot outcome of b1u
xlim([1 128]);
ylim([-1 1]);

subplot(2,1,2);
plot(b2u);%Plot outcome of b2u
xlim([1 128]);
ylim([-1 1]);

%Initialize of the parameters
bit_seq1 = b1u;
bit_seq2 = b2u;
bit_seq = bu;
n1 = length(b1u);
n2 = length(b2u);
n = length(bu);
fc = 0.25;
t = 0:0.001:4;
bb = [];
b_o = [];
b_e = [];
bit_e = [];
bit_o = [];
qpsk = [];
bec = [];
bes = [];

%Creating input waveform on bit sequence
for i = 1:n
    bx = bit_seq(i)*ones(1,1000);
    bb = [bb bx];
end
```

```
%Seperating Even and Odd bits
for i = 1:n
    if bit_seq(i) == 0
        bit_seq(i) =-1;
    end
    if mod(i,2) == 0
        e_bit = bit_seq(i);
        b_e = [b_e e_bit];
    else
        o_bit = bit_seq(i);
        b_o = [b_o o_bit];
    end
end

%Calculating QPSK signal
for i = 1:n/2
    be_c = (b_e(i)*cos(2*pi*fc*t));
    bo_s = (b_o(i)*sin(2*pi*fc*t));
    q = be_c+bo_s;
    even = b_e(i)*ones(1,2000);
    odd = b_o(i)*ones(1,2000);
    bit_e = [bit_e even];
    bir_o = [bit_o odd];
    qpsk = [qpsk q];
    bec = [bec be_c];
    bes = [bes bo_s];
end
figure('Name','QPSK')

%Plot waveform of input bit sequence
subplot(4,1,1);
plot(bb,'o');
grid on;
axis([0 (n*1000) -1 1]);
xlabel('Time');
ylabel('Input bits');
```

```
%Plot odd bits waveform
subplot(4,1,2);
plot(bes);
hold on;
plot(bit_o);
grid on;
axis([0 (n*1500) -1 1]);
ylabel('Odd bits');

%Plot even bits waveform
subplot(4,1,3);
plot(bec);
hold on;
plot(bit_e);
grid on;
axis([0 (n*1500) -1 1]);
ylabel('Even bits');

%Plot QPSk signal waveform
subplot(4,1,4);
plot(qpsk);
axis([0 (n*1500) -1.5 1.5]);
ylabel('QPSK waveform');
```