Spread Spectrum Communication and Jamming

Term Project

By

Prasoon Bopche (12EE10060)

Gaurav Jain(12MT10013)

TABLE OF CONTENTS

INTRODUCTION

This report was done as a part of the class project in the course Spread Spectrum Communication and Jamming. The project work involves developing a spread spectrum communication system using Kasami Codes and evaluating the Bit Error Rate vs. Signal to Noise Ratio for the channel and normal white Gaussian noise.

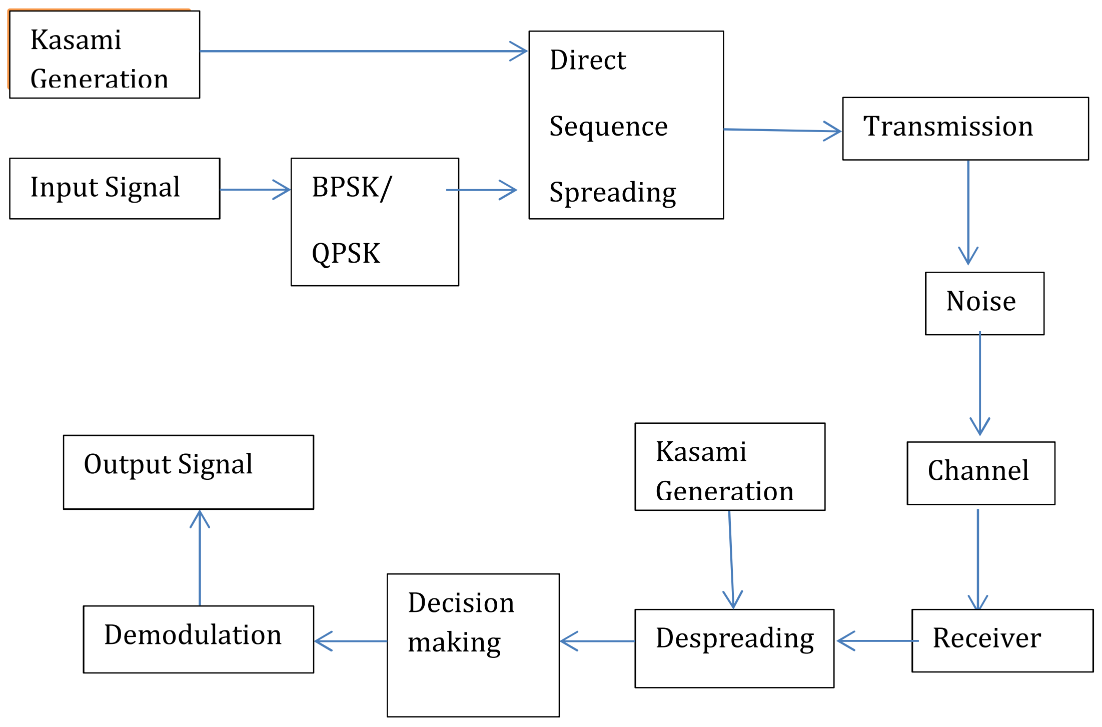
Kasami Codes are binary code sequences that are used for spread spectrum communication in order to provide extra prevention against the Jammer. These are created by cyclic decimation sampling and are known to provide good auto correlation. In this project, we have spread is using direct sequence spread spectrum communication. The channel is a 60GHz channel, whose channel coefficients were provided to us beforehand.

MOTIVATION

The motivation behind this project is apply the knowledge learned in the course and be able to create a simple communication system as a system designer does. The project was done as a group work and was guided by Prof. Debarati Sen throughout the process.

BLOCK DIAGRAM OF SYSTEM DESIGNED

The block diagram of the system designed is as given below:



THE KASAMI CODE GENERATION

We have generated the Kasami code using an initial M-sequence that we generated using a Linear shift register with initial state 1 0 1 1. The M sequence is then sampled at into a set of 3 taking the 5th value of the M sequence. This is then repeated over 5 times to get a 15 bit sequence, which is used in a XOR operation with the initial M-sequence to get the Kasami Code. The figure below explains the values and the results observed:

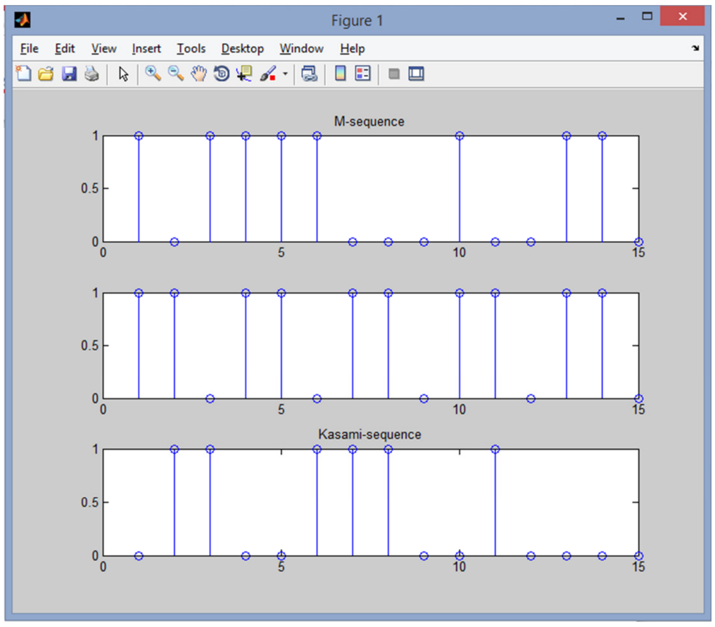


Figure: Generation of Kasami Code

AUTO CORRELATION AND CROSS CORRELATION

For Auto and Cross Correlation, we converted the Kasami Code generated in the previous process into bipolar representation. For auto-correlation we correlate it with itself and for cross-correlation we shift the sequence by one bit to generate a new sequence and do the correlation with the newly generated sequence.

For calculating for noise, an awgn noise of SNR 3db was added and the resulting sequence was correlated with the initial sequence.

The figure below shows the Auto and Cross Correlation results for Kasami Code with and without noise:

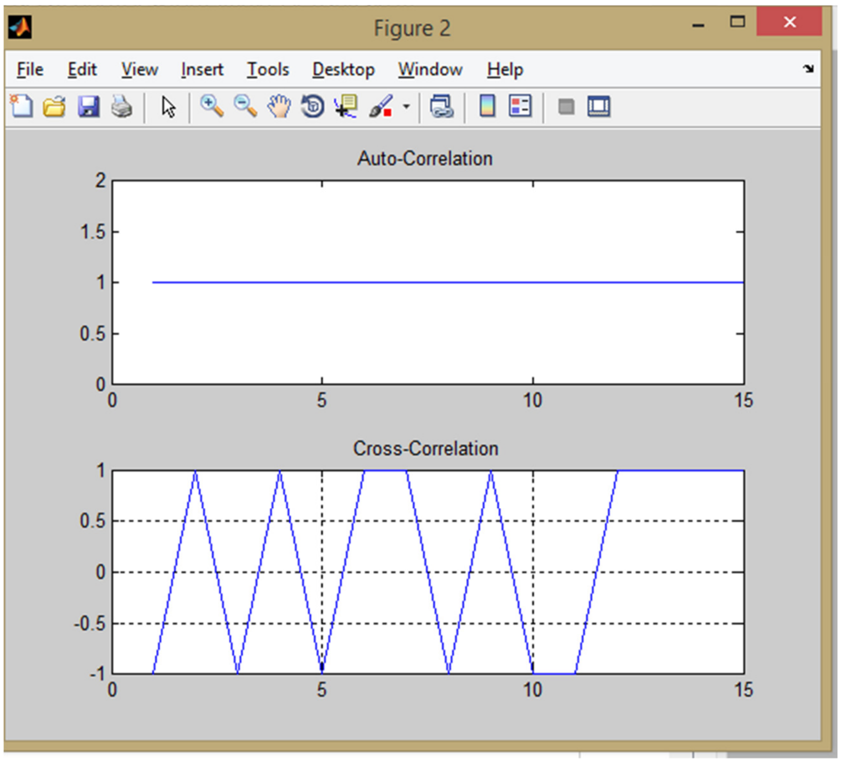


Figure: Auto and cross correlation without noise

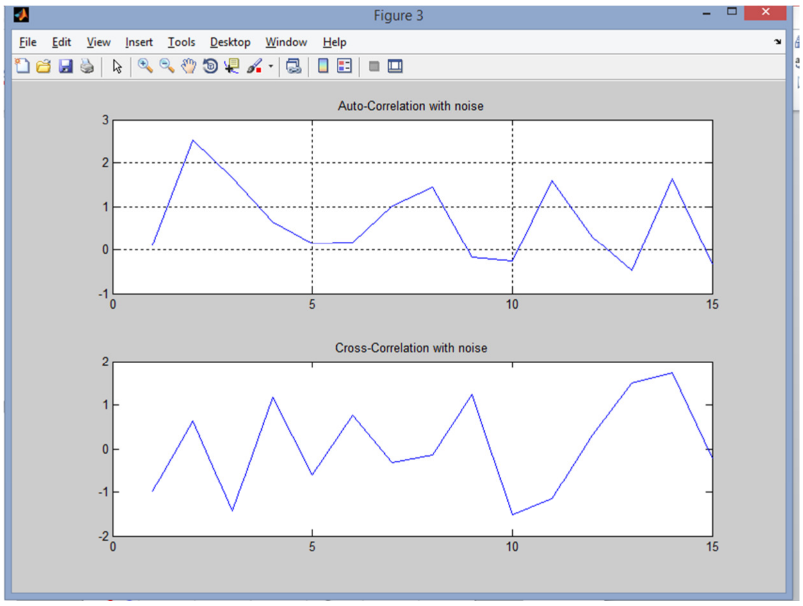


Figure: Auto and Cross Correlation with Noise

BER PLOT

The Bit error rate plot was generated by spreading the BPSK and QPSK modulation of the signal using direct sequence spread spectrum coding and then using noise and channel filter, and then dispreading the received signal. The received signal is then matched with the transmitted signal to get the bit error rate. This is done for different SNR of noise and thus gives us the BER vs SNR plot.

The red and the blue lines are for AWGN and the black and the pink is for the channel with Equalizer

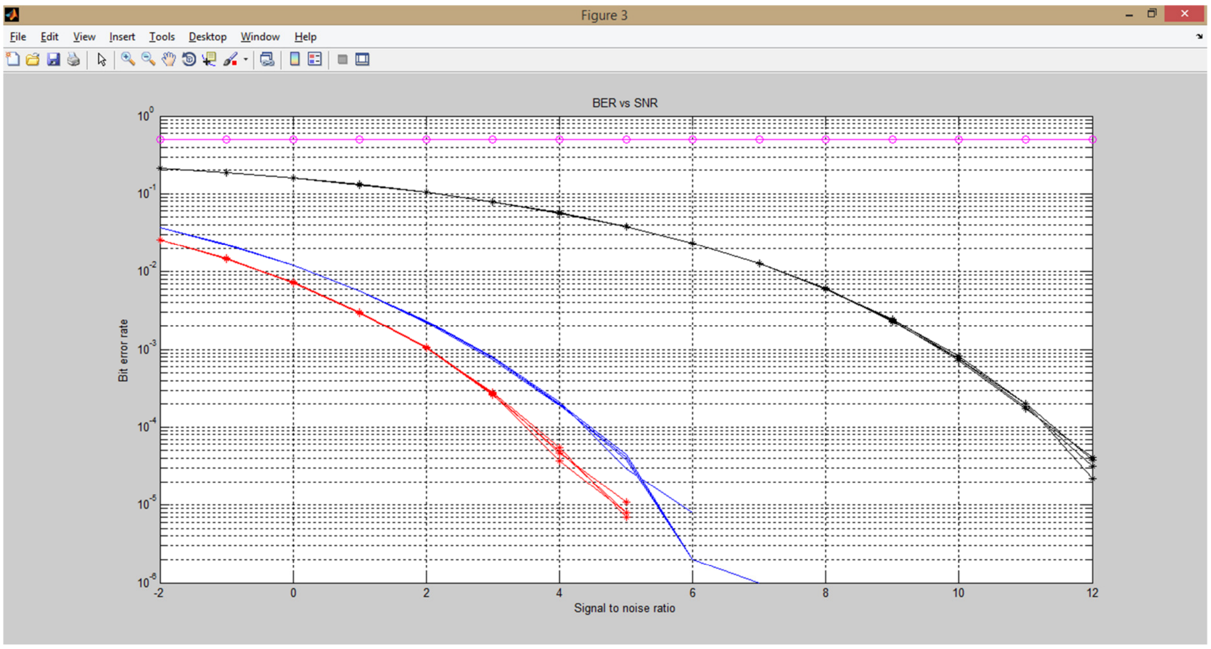


Figure: Bit Error rate vs SNR

Power Delay Profile

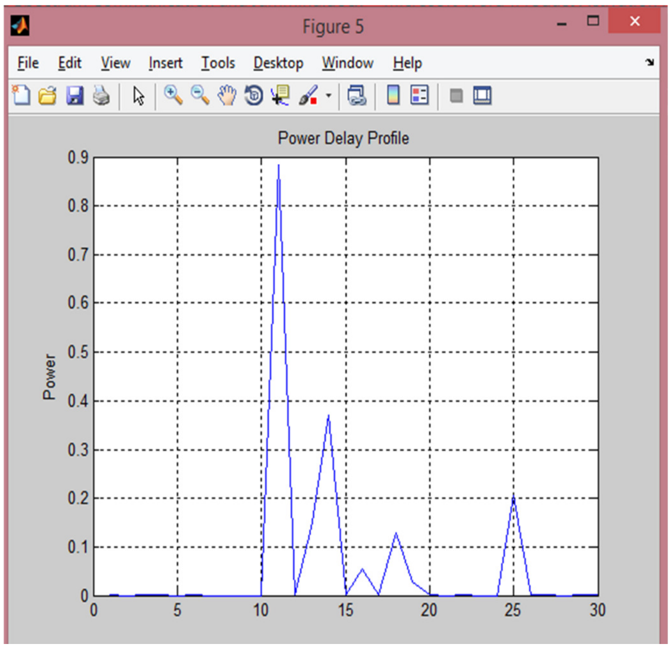


Figure: Power Delay Profile

Result and Discussion

As we can see from the figures above that the Bit Error profile is flat for channel, and there is a difference of processing gain between QPSK and BPSK. Also from power delay profile we can see that there is a peak near 11.

The results match with the expected outcomes, thus indicating that the system is working correctly. For the said channel, I would suggest that the optimum length of code should be higher than this, around 32 bits.

MATLAB CODE

The following code has to be run along with the equaliser and the channel coefficients in the same build path with file name as MMSE_eq and ch_coeff respectively.

```matlab
clear
clc
G=15;  % Code length
%Generation of first m-sequence using generator polynomial [45]
sdl =[1 0 1 1];              % Initial state of Shift register
PN1=[];                        % First m-sequence
for j=1:G
    PN1=[PN1 sd1(1)];
    if sd1(1)==sd1(2)
        temp1=0;
    else temp1=1;
    end
    sd1(1)=sd1(2);
    sd1(2)=sd1(3);
    sd1(3)=sd1(4);
    sd1(4)=temp1;
end
figure(1);
subplot(3,1,1)
stem(PN1)
title('M-sequence');
hold on;
for i=1:1:3
    kd1(i) = PN1(i*5);
end
KD1=[];
for i=1:G
    KD1=[KD1 kd1(1)];
    temp=kd1(1);
    kd1(1)=kd1(2);
    kd1(2)=kd1(3);
    kd1(3)=temp;
end
for i=1:G
    if KD1(i)==PN1(i)
        ks(i)=0;
    else
        ks(i)=1;
    end
end
figure(1);
subplot(3,1,2)
stem(KD1);
subplot(3,1,3)
```

```matlab
stem(ks)
title('Kasami-sequence');



for i=1:G
   if ks(i)==0
       ks1(i)= -1;
   else
       ks1(i) = 1;
   end
    r(i)=ks1(i)*ks1(i);
end
figure(2);
subplot(2,1,1)
plot(r)
title('Auto-Correlation')
hold on;
for i=1:G-1
    ks2(i)=ks1(i+1);
end
ks2(G) = ks1(1);
ks1_noise = awgn(ks,3);
ks2_noise = awgn(ks2,3);
for i=1:G
    cross_corr(i) = ks1(i)*ks2(i);
    r_noise(i) = ks1(i)*ks1_noise(i);
    cross_corr_noise(i) = ks1(i)*ks2_noise(i);
end
figure(2);
subplot(2,1,2)
plot(cross_corr)
title('Cross-Correlation')
hold on;
figure(3);
subplot(2,1,2)
plot(cross_corr_noise)
title('Cross-Correlation with noise')
hold on;
subplot(2,1,1)
plot(r_noise)
title('Auto-Correlation with noise')
grid on;



SNR = -2:1:12;
data_length = 1e6;
data_inr = 2*(rand(1,data_length/2)>0.5)-1;
data_ini = 2*(rand(1,data_length/2)>0.5)-1;
data_in = data_inr+1i*data_ini;

spread_data = kron(data_in,ks);
k=1;
for SNR =-2:1:12
    SNR_lin = 10^(SNR/10);
    sig_ener = 2;
    noise_pow =sig_ener/SNR_lin;
    noiser =sqrt(noise_pow/2)*randn(1,15*data_length/2);
    noisei =sqrt(noise_pow/2)*randn(1,15*data_length/2);
    noise = noiser+1i*noisei;
    trans_data = noise+spread_data;

load('ch_coeff.mat');

trans_chan = filter(ch_coeff,1,trans_data);
[qpsk_equal,equal_coeff]=MMSE_eq(trans_chan,ch_coeff,noise_pow/2);

qpsk_equal=qpsk_equal(1:length(trans_data));




des_data = reshape(qpsk_equal,G,data_length/2);
final_data = ks*des_data;
rec_datar =real(final_data);
rec_datai = imag(final_data);
decoder = 2*(rec_datar>0)-1;
    decodei = 2*(rec_datai>0)-1;
    errr = sum(decoder~=data_inr);
    erri = sum(decodei~=data_ini);
    err = errr+erri;
    prob1(k) = err/(data_length);


    des_data_ch = reshape(trans_chan,G,data_length/2);
final_data_ch = ks*des_data_ch;
rec_datar_ch =real(final_data_ch);
rec_datai_ch = imag(final_data_ch);
decoder_ch = 2*(rec_datar_ch>0)-1;
    decodei_ch = 2*(rec_datai_ch>0)-1;
    errr = sum(decoder_ch~=data_inr);
    erri = sum(decodei_ch~=data_ini);
    err = errr+erri;
    prob1_ch(k) = err/(data_length);

    des_data_awgn = reshape(trans_data,G,data_length/2);
final_data_awgn = ks*des_data_awgn;
rec_datar_awgn =real(final_data_awgn);
rec_datai_awgn = imag(final_data_awgn);
decoder_awgn = 2*(rec_datar_awgn>0)-1;
    decodei_awgn = 2*(rec_datai_awgn>0)-1;
    errr = sum(decoder_awgn~=data_inr);
    erri = sum(decodei_awgn~=data_ini);
    err = errr+erri;
    prob1_awgn(k) = err/(data_length);

    k=k+1;
end

no_bits = 1e6;
SNR = -2:1:12;
SNR_lin = 10.^(SNR/10);
in_datar = 2*(rand(1,no_bits/2)>0.5)-1;
in_datai = 2*(rand(1,no_bits/2)>0.5)-1;
in_data = in_datar+1i*in_datai;
Eb = 1;
Es = 2*Eb;
for ii=1:1:length(SNR)
    noise_pow = Es/SNR_lin(ii);
    noiser = sqrt(noise_pow/2)*randn(1,no_bits/2);
    noisei = sqrt(noise_pow/2)*randn(1,no_bits/2);
    rec_datar = in_datar + noiser;
    rec_datai = in_datai + noisei;
    decoder = 2*(rec_datar>0)-1;
    decodei = 2*(rec_datai>0)-1;
    errr = sum(decoder~=in_datar);
    erri = sum(decodei~=in_datai);
    err = errr+erri;
    prob1_wtspread(ii) = err/(no_bits);
end
SNR = -2:1:12;
figure(4);
semilogy(SNR,prob1);
grid on;
hold on
semilogy(SNR,prob1_ch,'-mo')
hold on
semilogy(SNR,prob1_wtspread,'-k*')
hold on
semilogy(SNR,prob1_awgn,'-r*')
title('BER vs SNR');
xlabel('Signal to noise ratio');
ylabel('Bit error rate');
grid on;

figure(5);
psd = abs(ch_coeff);
subplot(1,1,1);
plot(psd);
title('Power Delay Profile');
ylabel('Power');
grid on;
```