**EE-5283-002**

**Spread Spectrum Communication**

**&**

**GPS (Global Positioning System)**

**HomeWork #2**

**Submitted By**

**Name:**  Nimish Joshi

**Banner Id:**  @01482152

* **Homework Topic #**
* **BER curves for M-Code:**

In CDMA systems, for code generation we are using feedback shift register method. Shift register contains number of cells, and those cells are storage unit. While in m-sequence I generated by single Linear Shift Register (LSR) and the sequence is generated with n-stage binary shift register with linear feedback. The function of sequence is programmed and its BER curves for users 1,4 & 7 are given below.

The only change in the given program what we have done is taking sequence no:1- as a M-sequence & using it for different user 1, 4 and & respectively.

**1). Program: DS-CDMA**

% Program 5-6

%

% Simulation program to realize DS-CDMA system

%

% dscdma.m

%

% Programmed by M.Okita and H.Harada

%

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Preparation part \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

sr = 256000.0; % symbol rate

ml = 2; % number of modulation levels

br = sr \* ml; % bit rate

nd = 100; % number of symbol

ebn0 = 0; % Eb/No

while (ebn0 <= 30)

ebn0 = ebn0+1;

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Filter initialization \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

irfn = 21; % number of filter taps

IPOINT = 8; % number of oversample

alfs = 0.5; % roll off factor

[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1); % T FILTER FUNCTION

[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0); % R FILTER FUNCTION

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Spreading code initialization \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

user = 7; % number of users

seq = 1; % 1:M-sequence 2:Gold 3:Orthogonal Gold

stage = 3; % number of stages

ptap1 = [1 3]; % position of taps for 1st

ptap2 = [2 3]; % position of taps for 2nd

regi1 = [1 1 1]; % initial value of register for 1st

regi2 = [1 1 1]; % initial value of register for 2nd

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Generation of the spreading code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

switch seq

case 1 % M-sequence

code = mseq(stage,ptap1,regi1,user);

case 2 % Gold sequence

m1 = mseq(stage,ptap1,regi1);

m2 = mseq(stage,ptap2,regi2);

code = goldseq(m1,m2,user);

case 3 % Orthogonal Gold sequence

m1 = mseq(stage,ptap1,regi1);

m2 = mseq(stage,ptap2,regi2);

code = [goldseq(m1,m2,user),zeros(user,1)];

end

code = code \* 2 - 1;

clen = length(code);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Fading initialization \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

rfade = 0; % Rayleigh fading 0:nothing 1:consider

itau = [0,8]; % delay time

dlvl1 = [0.0,40.0]; % attenuation level

n0 = [6,7]; % number of waves to generate fading

th1 = [0.0,0.0]; % initial Phase of delayed wave

itnd1 = [3001,4004]; % set fading counter

now1 = 2; % number of direct wave + delayed wave

tstp = 1 / sr / IPOINT / clen; % time resolution

fd = 160; % doppler frequency [Hz

flat = 1; % flat Rayleigh environment

itndel = nd \* IPOINT \* clen \* 30; % number of fading counter to skip

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* START CALCULATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

nloop = 1000; % simulation number of times

noe = 0;

nod = 0;

for ii=1:nloop

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Transmitter \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

data = rand(user,nd\*ml) > 0.5;

[ich, qch] = qpskmod(data,user,nd,ml); % QPSK modulation

[ich1,qch1] = spread(ich,qch,code); % spreading

[ich2,qch2] = compoversamp2(ich1,qch1,IPOINT); % over sampling

[ich3,qch3] = compconv2(ich2,qch2,xh); % filter

if user == 1 % transmission

ich4 = ich3;

qch4 = qch3;

else

ich4 = sum(ich3);

qch4 = sum(qch3);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Fading channel \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if rfade == 0

ich5 = ich4;

qch5 = qch4;

else

[ich5,qch5] = sefade(ich4,qch4,itau,dlvl1,th1,n0,itnd1, ... % fading channel

now1,length(ich4),tstp,fd,flat);

itnd1 = itnd1 + itndel;

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Receiver \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

spow = sum(rot90(ich3.^2 + qch3.^2)) / nd; % attenuation Calculation

attn = sqrt(0.5 \* spow \* sr / br \* 10^(-ebn0/10));

[ich6,qch6] = comb2(ich5,qch5,attn); % Add White Gaussian Noise (AWGN)

[ich7,qch7] = compconv2(ich6,qch6,xh2); % filter

sampl = irfn \* IPOINT + 1;

ich8 = ich7(:,sampl:IPOINT:IPOINT\*nd\*clen+sampl-1);

qch8 = qch7(:,sampl:IPOINT:IPOINT\*nd\*clen+sampl-1);

[ich9 qch9] = despread(ich8,qch8,code); % despreading

demodata = qpskdemod(ich9,qch9,user,nd,ml); % QPSK demodulation

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Bit Error Rate (BER) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

noe2 = sum(sum(abs(data-demodata)));

nod2 = user \* nd \* ml;

noe = noe + noe2;

nod = nod + nod2;

fprintf('%d\t%e\n',ii,noe2/nod2);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Data file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ber(ebn0) = noe / nod;

end

semilogy(ber,'m-\*')

axis([1,30,10^-4,10^-1])

xlabel('Eb/n0(dB)-time');

ylabel('BER');

title('BER PErformance of DS-CDMA with m-sequence')

legend('Number of User = 7')

whitebg('w')

grid on;

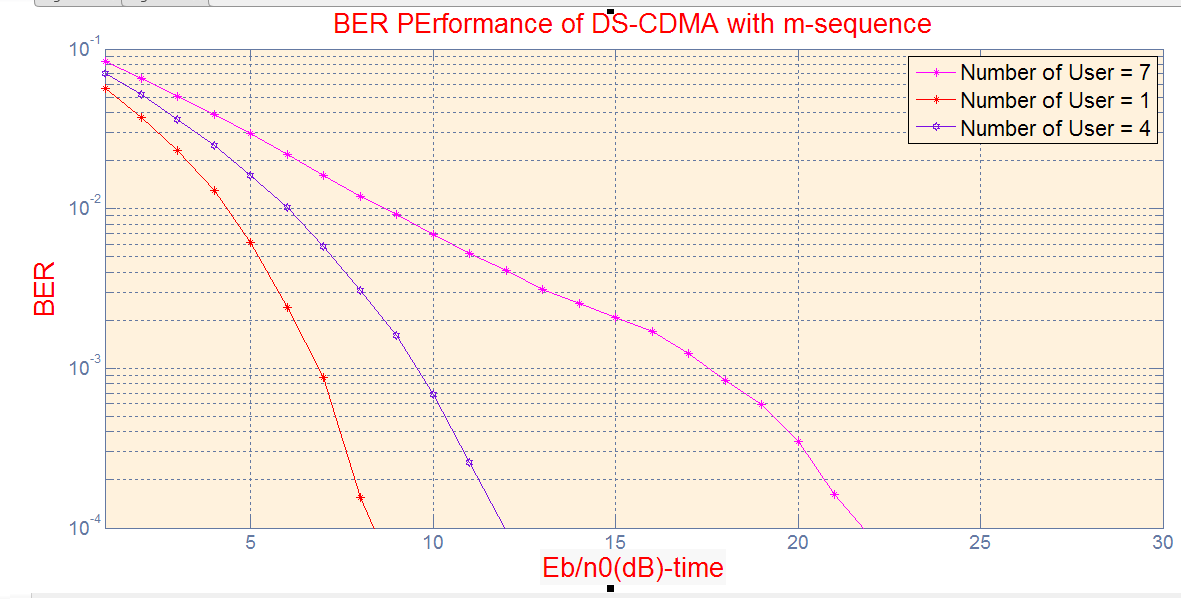
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);% fprintf: built in function

fid = fopen('BER.dat','a');

fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod); % fprintf: built in function

fclose(fid);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* end of file\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



**Figure: BER Performance of ds-CDMA for m-sequence.**

* **BER curves for Gold-Code:**

The limitation of m-sequence is its function which is very small. In the CDMA system where many users communicate with one another , wen need sequences with many different codes which may have some correlation values. We consider Gold sequence with these concerns.

The basic logic behind Gold Sequence is, it generates signals by EXOR of two m-sequences. Additionally, to perform this Three stage shift registr is used

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Spreading code initialization \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

user = 4; % number of users

seq = 2; % 1:M-sequence 2:Gold 3:Orthogonal Gold

stage = 3; % number of stages

ptap1 = [1 3]; % position of taps for 1st

ptap2 = [2 3]; %position of taps for 2nd

regi1 = [1 1 1]; % initial value of register for 1st

regi2 = [1 1 1]; % initial value of register for 2nd

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Generation of the spreading code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

switch seq

case 1 % M-sequence

code = mseq(stage,ptap1,regi1,user);

case 2 % Gold sequence

m1 = mseq(stage,ptap1,regi1);

m2 = mseq(stage,ptap2,regi2);

code = goldseq(m1,m2,user);

case 3 % Orthogonal Gold sequence

m1 = mseq(stage,ptap1,regi1);

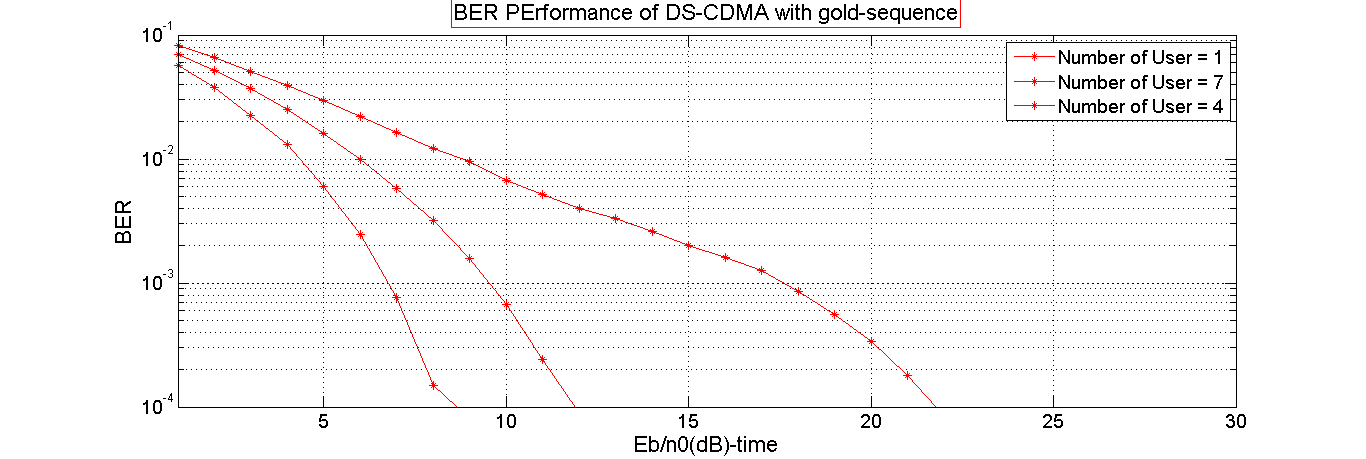
m2 = mseq(stage,ptap2,regi2);

code = [goldseq(m1,m2,user),zeros(user,1)];

end

code = code \* 2 - 1;

clen = length(code);



**Arbitary CODE:**

**Program:**

clc;

clear all, close all;

%% Create random data

nbits = 3;

ngarbage = 3\*nbits;

data = randi([0,1],1,nbits);

garbage = randi([0,1],1,ngarbage);

stream = horzcat(data,garbage);

%% Convert from Unipolar to Bipolar Encoding

stream\_b = 2\*stream - 1;

%% Define Parameters

%%% Variable Parameters

nsamples = 20\*nbits;

nseq = 5 %# Iterate stream nseq times

T = 1000; %# Number of periods

Ts = 1; %# Symbol Duration

Es = Ts/2; %# Energy per Symbol

fc = 1e9; %# Carrier frequency

%%% Dependent Parameters

A = sqrt(2\*Es/Ts); %# Amplitude of Carrier

omega = 2\*pi\*fc %# Frequency in radians

t = linspace(0,T,nsamples) %# Discrete time from 0 to T periods with nsamples samples

nspb = nsamples/length(stream) %# Number of samples per bit

%% Creating the BPSK Modulation

%# simple matrix manipulation.

% Replicate each bit nspb/nseq times

repStream\_b = repmat(stream\_b',1,nspb/nseq);

% Tranpose and replicate nseq times to be able to fill to t

modSig\_proto = repmat(repStream\_b',1,nseq);

% Tranpose column by column, then rearrange into a row vector

modSig = modSig\_proto(:)';

%% The Carrier Wave

carrier = A\*cos(omega\*t);

%% Modulated Signal

sig = modSig.\*carrier;

* **PART\_2: Autocorrelation and Cross correlation part**

**Program Part:**

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Generation of the spreading code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

switch seq

case 1 % M-sequence

code = mseq(stage,ptap1,regi1,user);

% auto = code

case 2 % Gold sequence

m1 = mseq(stage,ptap1,regi1);

m2 = mseq(stage,ptap2,regi2);

code = goldseq(m1,m2,user);

auto = code;

% code2=goldseq(m1,m2,user);

case 3 % Orthogonal Gold sequence

m1 = mseq(stage,ptap1,regi1);

m2 = mseq(stage,ptap2,regi2);

code = [goldseq(m1,m2,user),zeros(user,1)];

end

code = code \* 2 - 1;

clen = length(code);

.

.

.

.

.

subplot(2, 1, 1)

plot(auto)

subplot (2, 1, 2)

plot(autocorr([auto]))

% subplot(3, 1, 1)

% plot(auto)

% subplot (3, 1, 2)

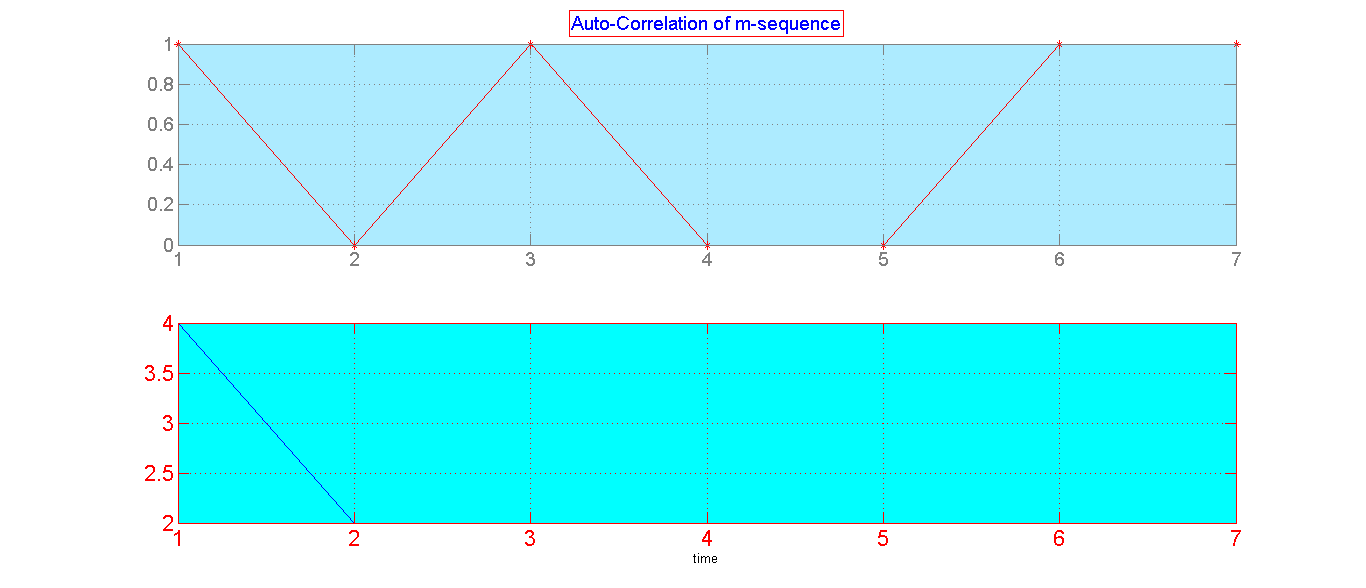
% plot(code2)

%(3,1,1)

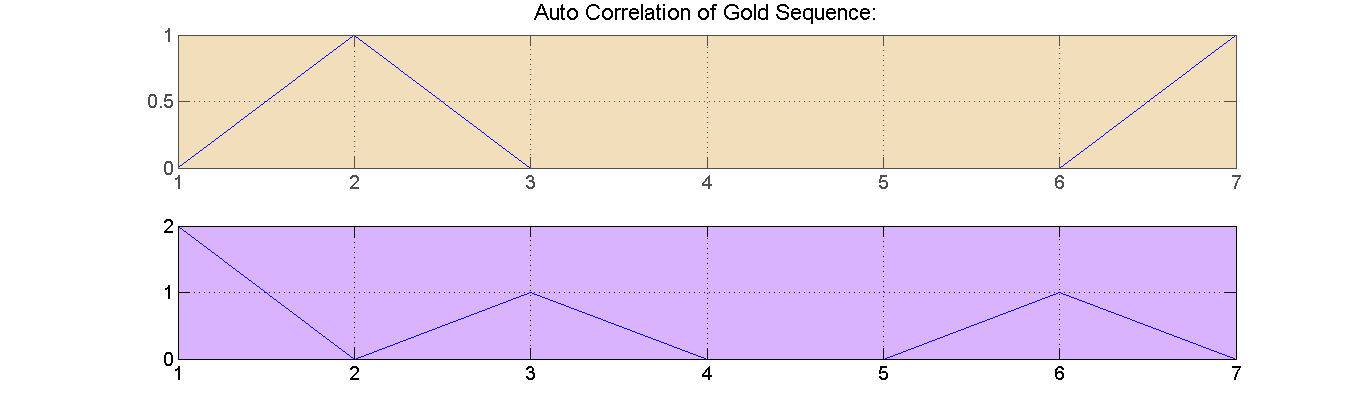
%plot(crosscorr(auto, code2)

-------------------------------------End------------------------------------

(1). Autocorrelation of m-sequence:



(2). Autocorrelation of Gold Sequence;



* **Crosscorrelation of Two Gold Sequences**

**CODING PART:**

switch seq

case 1 % M-sequence

code = mseq(stage,ptap1,regi1,user);

% auto = code

case 2 % Gold sequence

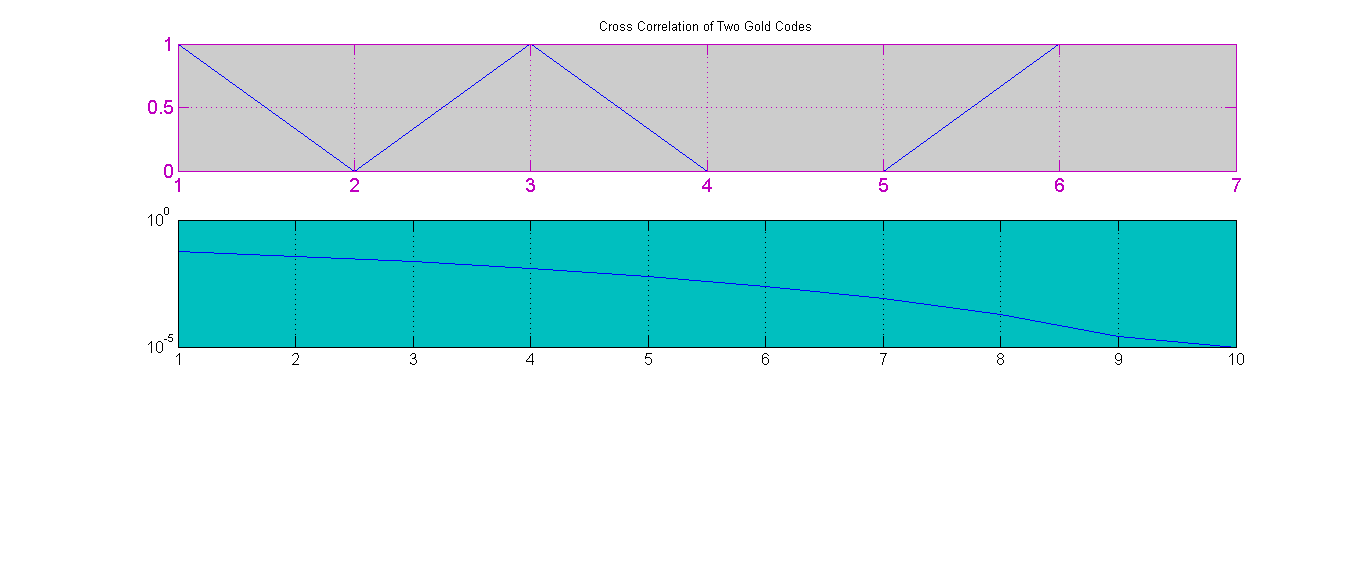
m1 = mseq(stage,ptap1,regi1);

m2 = mseq(stage,ptap2,regi2);

code = goldseq(m1,m2,user);

auto = code;

code2=goldseq(m1,m2,user);

****