LAMPIRAN A

Listing Program Utama

```
clear;
close all;
clc;

% Inisialisasi masukan
n=1;    % ukuran matriks Hadamard (2^n*2^n) --> n minimal = 1
N=(2.^n).^2;     % N = panjang_kode_max
panjang_data = 10;


% Proses pembangkitan data berupa integer (0 atau 1)
data_awal=randint(1,panjang_data)


% Proses pembentukan data bipolar (0 jadi -1, 1 jadi +1)
for k=1:panjang_data
    if (data_awal(k)==0)
        data_bipolar(k)=-1;
    else
        data_bipolar(k)=1;
    end;
end;
data_bipolar
data_ulang=ulang(N,data_bipolar)


% Pembentukan complementary code
masuk=1;
for k=1:n
    comp_code=bentuk_comp_code(masuk,2.^k);
    masuk=comp_code;
end;
comp_code;
```

```
% Pembentukan EXTENDED complementary code
ext_comp_code=bentuk_ext_com(comp_code,n);


% Proses modulasi dengan spreading sequence dan / atau Proses
interferensi

kode_used1=ext_comp_code(1,:);
kode_used2=ext_comp_code(2,:);

for k=1:length(data_bipolar)
    data_termod_kali1((k-1)*N+1:k*N)=kali(data_ulang((k-
1)*N+1:k*N),kode_used1);
    data_termod_kali2((k-1)*N+1:k*N)=kali(data_ulang((k-
1)*N+1:k*N),kode_used2);
end;


% Plot sinyal termodulasi
for k=1:length(data_bipolar)
    sinyal_termod1((k-1).*20+1:k*20)=sinyal(data_termod_kali1(k));
    sinyal_termod2((k-1).*20+1:k*20)=sinyal(data_termod_kali2(k));
end;


% sinyal_termod(2,:)=sinyal(data_termod_kali2);

figure;plot(sinyal_termod1);grid;title('Sinyal Termodulasi : User -
1');xlabel('Time Chip');ylabel('Amplituda');
figure;plot(sinyal_termod2);grid;title('Sinyal Termodulasi : User -
2');xlabel('Time Chip');ylabel('Amplituda');


% Proses Dekoding Sinyal dan Menghitung BER


% (BER sebagai fungsi jumlah user dalam sistem), kanal AWGN, single
path

% Inisialisasi kanal
% SNR = 3;
% T = 50000;
% alpha = 1;
% SNR_dec = 10.^(SNR / 10);
% ku = 1;
% k=1;
% p=N;
% kode=ext_comp_code;
% kode_multi_prima=ext_comp_code;
% Matcodes=[kode;kode_multi_prima]
% % Matcodes=[kode;kode_multi_prima(n)]
```

```
%
% % Mulai perhitungan dengan MUD : matched filter
% for K=1:1:size(Matcodes,1)  %K jumlah user
%     A=floor((0:(K-1))*5/K)+1;
%     Aavg = sum(A.^2,2)/T;
%     C = Matcodes(K,:);
%     Eb = C(:,ku)' * C(:,ku)*Aavg(ku);
%     mf = C(:,ku);
%     sigma = Eb/SNR_dec(k)/2;
%     data=randuni(100);
%     Sentbit(K,:)=ulang(p,data);
%     kode_used=Matcodes(K,:);
%     kode_used=circshift(kode_used,[0 K-1]);
%     for m=1:length(data)
%         Y(K,(m-1)*p.^2+1:m*p.^2)=kali(Sentbit(K,(m-
1)*p.^2+1:m*p.^2),kode_used)...
%             +0.01.*randn(1,p.^2);
%     end;
%     for m=1:length(data)
%         Receivedbit1(K,(m-1)*p.^2+1:m*p.^2)=sign(mf.*Y(K,(m-
1)*p.^2+1:m*p.^2));
%     end;
%     Perr1(K) = sum(abs(Receivedbit1(K,:)-Sentbit(K,:)))/2/T;
%
%     % Perhitungan deteksi (MUD) : decorrelator
%     for pp=1:length(C)
%         if C(pp)==0
%             C(pp)=C(pp)+100*eps;
%         end;
%     end;
%     Receivedbit2 = sign(inv(C'*C)*C'*Y(K,:));
%     Perr2(K) = sum(abs(Receivedbit2(K,:)-Sentbit(K,:)))/2/T;
% end;
%
figure;semilogy(1:1:size(Matcodes,1),Perr2(1:1:size(Matcodes,1)),'ro
-');
% hold on;grid;
% xlabel('Jumlah user');ylabel('BER');title('MUD dengan
decorrelator');


ext_comp_code;
% % Pemeriksaan apakah memenuhi sifat ortogonal cross correlation
% hitung=0;
% for m=1:size(mat_H,1)
%     cek=cek_ortogonal(mat_H(1,:),mat_H(m,:));
%     if (cek==1)
%         hitung=hitung+1;
%     end;
% end;
% %hitung
%
```

Function untuk membuat complementary code

```
function [keluar] = bentuk_comp_code(masuk,n)
%
% Function ini untuk membentuk complementary code
% dengan menggunakan Matriks Hadamard
%
% Variabel masukan : complementary code yang mau dibentuk
%                                    (n) --> harus pangkat 2
%Variabel keluaran : keluara : complementary code yang dihasilkan
%
keluar(1:n/2,1:n/2)=masuk;
keluar(1:n/2,n/2+1:n)=masuk;
keluar(n/2+1:n,1:n/2)=masuk;
keluar(n/2+1:n,n/2+1:n)=-masuk;
```

Function untuk membentuk extended complementary codes

```
function [keluar]=bentuk_ext_com(masuk,n)
%
% Function ini untuk membentuk extended complementary code
%
% Variabel masukan : masuk = complementary code
   %                                    n = ukuran matriks masuk
   %
   % Variabel keluaran : keluar =extended complementary code yang
   % terbentuk
   %
  comp_code=masuk;
  if (n==1)
    ext_comp_code(1,:)=[comp_code(1,:),comp_code(2,:)];
    ext_comp_code(2,:)=[comp_code(1,:),-comp_code(2,:)];
end;
if (n==2)

ext_comp_code(1,:)=[comp_code(1,:),comp_code(2,:),comp_code(3,:),com
p_code(4,:)];
    ext_comp_code(2,:)=[comp_code(1,:),-
comp_code(2,:),comp_code(3,:),-comp_code(4,:)];
    ext_comp_code(3,:)=[comp_code(1,:),comp_code(2,:),-
comp_code(3,:),-comp_code(4,:)];
    ext_comp_code(4,:)=[comp_code(1,:),-comp_code(2,:),-
comp_code(3,:),comp_code(4,:)];
end;
keluar=ext_comp_code;
```

Function untuk mencek orthogonal complementary series

```
function [keluar]=cek_ortogonal(x,y)
%
% Function ini untuk mencek apakah dua
% buah complementary series ortogonal
% atau tidak
%
% Variabel masukan : x dan y adalah vektor
% kode yang mau diperiksa ortogonal atau tidak
%
% Variabel keluaran : keluar : 1 tidak, 0 ya
%
if (length (x) ~= length (y))
    error ('Kedua kode tidak sama panjang');
end;

cek=cumsum(x.*y);

if (cek(end)==0)
    keluar=0;
else
    keluar=1;
end;
```

Function untuk modulasi ASK

```
function keluar=sinyal(masuk)
%
% Function ini untuk modulasi ASK
%
% Variabel masukan : masuk
% Variabel keluaran : keluar
%
t=0.05:.05:1;
f=1;
A=1;

if masuk==1
    keluar=A.*sin(2.*pi.*f.*t);
else
    keluar=-A.*sin(2.*pi.*f.*t);
end;
```

Function untuk operasi perkalian

```
function keluar = kali(masuk1,masuk2)
%
% Function ini untuk operasi perkalian
% Variabel masukan : masuk1 dan masuk2
% Variabel keluaran : keluar
if length(masuk1) ~= length(masuk2)
    error('Dimensi kedua sinyal masukan berbeda');
end;

for k=1:length(masuk1)
    keluar(k)=masuk1(k).*masuk2(k);
end;
```

Function untuk mengulang data user sesuai dengan periode chip yang digunakan

```
function keluar=ulang(N,masuk)
%
% Function ini untuk mengulang data user
% sesuai dengan periode chip yang digunakan(N = n^2)
%
% Variabel masukan : N = panjang kode maks
%                                masuk = data user
%
% Variabel keluaran : keluar = data user yang
%                                sudah di-repetisi (diulang)
%                                sehingga panjangnya sama
%                                dengan banyak data * N^2
%

for m=1:length(masuk)
    keluar((m-1)*N+1:m*N)=masuk(m);
end;
```

Function untuk membangkitkan data biner unipolar secara random

```
function[p]=randuni(N)
%
% Function ini membangkitkan data biner
% unipolar secara random
%
% Variabel masukan : N = bilangan bulat (>=2) random
%
% Variabel keluaran : p = data random unipolar

for i=1:N
    temp=rand;
    if (temp<0.5)
        data(1,i)=0;
    else
        data(1,i)=1;
    end;
end;
p=data(1,:);
```