LAMPIRAN
KODE PROGRAM SISTEM MIMO ST-BICM MENGGUNAKAN MATLAB 7.0.1

```
% inisialisasi
nRx=4;

% ---------------------------------------------------------%
%Bagian Pengirim STBICM
%
% ---------------------------------------------------------%
% Membangkitkan deretan data keluaran 0 atau 1

% k=input('Masukkan banyaknya bit yang mau dikirimkan = ');
k=500;
msg_orig=randint(1,k);

% N=input('Masukkan level sinyal QPSK = ');
N=1;

% Dilewatkan ke Convolutional Encoder

numSymb=size(msg_orig,2);

numPlot=numSymb;


% Convolutional Encoding terhadap Sinyal Biner
codeRate = 1/2; % Code rate
constlen = [7]; % Constraint length
codegen = [171 133]; % Generator polinomial
trel=poly2trellis(constlen,codegen); % Trellis
[msg_enc_bi]=convenc(msg_orig,trel);
numEncPlot=numPlot./codeRate;
tEnc=[0:numEncPlot-1]*1;


% proses interleaving
out_conv=zeros(size(msg_enc_bi));
out_conv(1,1:size(msg_enc_bi,2)./2)=msg_enc_bi(1,size(msg_e
nc_bi,2)./2+1:size(msg_enc_bi,2));
out_conv(1,size(msg_enc_bi,2)./2+1:size(msg_enc_bi,2))=msg_
enc_bi(1,1:size(msg_enc_bi,2)./2);
```

```
% proses seri to parallel
ukur_out_conv=size(out_conv,2);
bnyk=ukur_out_conv./250;

for n=1:bnyk
    m=n:4:ukur_out_conv;
    para{n}=[out_conv(m)];
end

paralel_1(1,1:size(out_conv,2)./4)=para{1};
paralel_2(1,1:size(out_conv,2)./4)=para{2};
paralel_3(1,1:size(out_conv,2)./4)=para{3};
paralel_4(1,1:size(out_conv,2)./4)=para{4};

clear n;

% Proses Mapper (asumsi menggunakan QPSK)
ukur_out_mapper=size(paralel_1,2);
bnyk_kirim=ukur_out_mapper./2;

for b=1:4
    simpan(1,1:ukur_out_mapper)=para{b};
    temp=zeros(1,bnyk_kirim);
    for n=1:bnyk_kirim
        [uji]=[simpan(2*n-1) simpan(2*n)];
        if uji == [0 0]
            out_map=1.*N;
        elseif uji == [0 1]
            out_map=i.*N;
        elseif uji == [1 1]
            out_map=-1.*N;
        elseif uji == [1 0]
            out_map=-i.*N;
        end;
        temp(n)=out_map;
        clear uji
    end;
    out_mapper{b}=[temp];
end;

out_antena_1=out_mapper{1};
out_antena_2=out_mapper{2};
out_antena_3=out_mapper{3};
out_antena_4=out_mapper{4};

% ------------------------------------------------------%
```

```
% Batas bagian Pengirim STBICM
%
% ------------------------------------------------------%

% Proses di kanal %
% Matriks 4 x 4 : kondisi kanal diketahui
%
mat_kanal=sqrt(1/2).*(randn(numSymb/4,nRx) +
j*randn(numSymb/4,nRx));

modtype='psk';%tambahan
Mary=4;
dobiterr='no';
snr=4;
mc=10;
Set=[0:Mary-1]';
Smap=dmodce(Set,1,1,modtype,Mary);
Eav=Smap'*Smap/Mary;
NF=10^(snr/10);
S=sqrt(Eav/(2*NF));
noise=S*(randn(numSymb,1)+i*randn(numSymb,1));
%akhir tambahan
% size(noise)
noise=reshape(noise,125,4);
% noise';

for ja=1:nRx
    rx(:,ja)=mat_kanal(:,ja).*out_mapper{ja}' +
noise(1:125,ja);
end



H=H.';                                  %Start decoding
with perfect channel estimation

        for co_ii=1:num_X
            for co_tt=1:size(eta,2)
                if eta(co_ii,co_tt)~=0
                    if coj_mt(eta(co_ii,co_tt),co_ii)==0

r_til(eta(co_ii,co_tt),:,co_ii)=Y(eta(co_ii,co_tt),:);

a_til(eta(co_ii,co_tt),:,co_ii)=conj(H(epsilon(eta(co_ii,co
_tt),co_ii),:));
                    else
```

```
r_til(eta(co_ii,co_tt),:,co_ii)=conj(Y(eta(co_ii,co_tt),:))
;

a_til(eta(co_ii,co_tt),:,co_ii)=H(epsilon(eta(co_ii,co_tt),
co_ii),:);
                    end;
                end;
            end;
        end;

        RR=zeros(num_X,1);
        for ii=1:num_X                    %Generate
decision statistics for the transmitted signal "xi"
            for tt=1:size(eta,2)
                for jj=1:Nr
                    if eta(ii,tt)~=0

RR(ii,1)=RR(ii,1)+r_til(eta(ii,tt),jj,ii)*a_til(eta(ii,tt),
jj,ii)*delta(eta(ii,tt),ii);
                    end;
                end;
            end;
        end;

        re_met_sym=pskdemod(RR,M_psk,0,'gray');
% = ML decision for M-PSK
        re_met_bit=de2bi(re_met_sym);
        re_met_bit(1,num_bit_per_sym+1)=0;    %For correct
demension of "re_met_bit"

        for con_dec_ro=1:num_X
            con_dec_ro
            if re_met_sym(con_dec_ro,1)~=mat(con_dec_ro,1)
%             if
re_met_sym(con_dec_ro,1)~=de_data(con_dec_ro,1)
                n_err_sym=n_err_sym+1;
                for con_dec_co=1:num_bit_per_sym
                    if
re_met_bit(con_dec_ro,con_dec_co)~=bi_data(con_dec_ro,con_d
ec_co)
                        n_err_bit=n_err_bit+1;
                    end;
                end;
            end;
        end;
    end;
```

```
    Perr_sym=n_err_sym/(num_X*Nit);           %Count number
of error bits and symbols
    graph_inf_sym(SNR-snr_min+1,2)=Perr_sym;
    Perr_bit=n_err_bit/(num_X*Nit*num_bit_per_sym);
    graph_inf_bit(SNR-snr_min+1,2)=Perr_bit;
end;

x_sym=graph_inf_sym(:,1);                     %Generate
plot
y_sym=graph_inf_sym(:,2);
subplot(2,1,1);
semilogy(x_sym,y_sym,'k-v');
xlabel('SNR, [dB]');
ylabel('Symbol Error Probability');
grid on
x_bit=graph_inf_bit(:,1);
y_bit=graph_inf_bit(:,2);
subplot(2,1,2);
semilogy(x_bit,y_bit,'k-v');
xlabel('SNR, [dB]');
ylabel('Bit Error Probability');
grid on;
```