LAMPIRAN A:

M-FILE MATLAB

```
% gambar4_1.m

%rand('state',state0); randn('state',state1);
state0 = rand('state'); state1 = randn('state');

nu = 16;                % panjang CP
tLh_mid = 32;           % delay spread dari remote scatterers
tLh_local = 6;          % delay spread dari tiap set local
scatterers
decay = 2;              % delay rate exponensial kanal

hmid = exp(-[1:tLh_mid].'/(tLh_mid/decay)) .* ...
   (randn(tLh_mid,1) + j*randn(tLh_mid,1));
hlocal1 = randn(tLh_local,1) + j*randn(tLh_local,1);
hlocal2 = randn(tLh_local,1) + j*randn(tLh_local,1);
h = conv(conv(hlocal1,hmid),hlocal2);
h = h/norm(h);
Lh = length(h)-1;

d = MaxWindow(h,nu+1);

figure(1); clf;
set(0,'DefaultAxesFontSize',16);
stem([0:Lh],abs(h));
hold on;
stem([d:d+nu],abs(h(d+1:d+nu+1)),'filled');
hold off;
vv = axis;
axis([0 Lh 0 vv(4)]);
xlabel('tap index','FontSize',16);
ylabel('tap magnitude','FontSize',16);
grid on;
```

```
% gambar4_2_4_3.m
% perbandingan BER desain-desain equalizer

%rand('state',state0); randn('state',state1);
state0 = rand('state'); state1 = randn('state');

% untuk Rayleigh fading channels (gambar4_2):
%    tLw=48, nu=16, Nchans=100, and rawiter=100
% untuk Rayleigh fading channels (gambar4_3):
%    tLw=8, nu=8, Nchans=100, and rawiter=100
%    (setelah loop 8, mulai lagi pada loop 1
%    dengan set 100 symbol baru)

Nchans = 100;           % jumlah kanal untuk disimulasikan
%Nchans = 28000;         % jumlah kanal untuk disimulasikan
chantype = 1;           % 1 = Rayleigh fading, 2 = ARMA, 3 = CSA
tLw = 48;               % panjang TEQ untuk Rayleigh fading
channels
%tLw = 8;                % panjang TEQ untuk CSA loops
Lw = tLw-1;             % orde TEQ
nu = 16;                % panjang TEQ untuk Rayleigh fading
channels
%nu = 8;                 % panjang TEQ untuk CSA loops
N = 64;                 % ukuran FFT
M = N+nu;               % total ukuran simbol
SNRdB = [0:5:60]';      % SNR dalam dB
Lsnr = length(SNRdB);   % jumlah nilai SNR
tLh_mid = 32;           % delay spread dari remote scatterers
tLh_local = 6;          % delay spread dari tiap set local
scatterers
decay = 2;              % delay rate eksponensial kanal
rawiter = 100;          % jumlah block data untuk dikirim
Kest = 100;             % jumlah block untuk estimasi TEQ
ssx = 1;                % daya sinyal
sx = sqrt(ssx);         % standar deviasi sinyal
alpha = 1/2;            % weighting sinyal

%-----------------------------------------
% membuat data yang ditransmisi dan noise

x = zeros(M*rawiter,1);
Xtrue = sign(randn(2*N*rawiter,1));
Xdd = zeros(size(Xtrue));
Xinit = (1+j)*ones(N,1);
for k=1:rawiter
   X = Xtrue((k-1)*(2*N)+[1:2:2*N]) + j*Xtrue((k-
1)*(2*N)+[2:2:2*N]);
   x([nu+1:M]'+M*(k-1)) = sqrt(N)*ifft(X,N);
   x([1:nu]'+M*(k-1)) = x([N+1:N+nu]'+M*(k-1));  % tambahkan CP
   % decoding differensial :
   temp = (angle(X) - angle(Xinit))/pi;
   Xdd((k-1)*(2*N)+[1:2:2*N]) = sign(mod(temp+0.25,2)-1);
   Xdd((k-1)*(2*N)+[2:2:2*N]) = sign(mod(temp-0.25,2)-1);
   Xinit = X;
end % untuk k=1:rawiter
x = x*(sx/sqrt(2));
```

```
n = (1/sqrt(2)) * (randn(M*rawiter,1)+j*randn(M*rawiter,1));

%-----------------------------------------
hh = waitbar(0,'Iterasi melalui kanal dan SNR...');
BER = zeros(Lsnr,6);
for channum=1:Nchans

% membuat kanal

if chantype==1 % Rayleigh fading channel
   hmid = exp(-[1:tLh_mid].'/(tLh_mid/decay)) .* ...
      (randn(tLh_mid,1) + j*randn(tLh_mid,1));
   hlocal1 = randn(tLh_local,1) + j*randn(tLh_local,1);
   hlocal2 = randn(tLh_local,1) + j*randn(tLh_local,1);
   h = conv(conv(hlocal1,hmid),hlocal2);
elseif chantype==2 % ARMA channel
   p1 = 0.85;
   p2 = 0.65+0.45*j;
   p3 = 0.65-0.45*j;
   MApart = randn(10,1) + j*randn(10,1);
   ARpart = poly([p1,p2,p3]);
   imp = zeros(42,1); imp(1)=1;
   h = filter(MApart,ARpart,imp);
elseif chantype==3 % CSA loops
   channum2 = mod(channum-1,8)+1;
   eval(['load csaloop',num2str(channum2),'.time']);
   eval(['h = csaloop',num2str(channum2),'(:,2);']);
   h = h(1:4:end/2);
end
h = h/norm(h);
tLh = length(h);  % panjang kanal
tLc = tLh+tLw-1;  % panjang kanal efektif
dmin = 0;         % delay minimum yang diijinkan
dmax = tLc-1-nu;  % delay maksimum yang diijinkan

for SNRi = 1:Lsnr %---------------

SNR = SNRdB(SNRi);    % SNR dalam dB
g = 10^(-SNR/10);     % 1/SNR, skala linier
ssn = g*ssx;          % daya noise
sn = sqrt(ssn);

% membuat data yang diterima
r = conv(h,x);
r = r(1:length(n)) + sn*n;

% menghitung beragam TEQ dan equalisasi

% 1: tanpa TEQ ----------------------------------

% hitung delay optimum kanal
d1 = MaxWindow(h,nu+1);

y = r;
```

```
Xest = zeros(size(Xtrue));
Ydd = zeros(size(Xtrue));
Yinit = (1+j)*ones(N,1);
for k=1:rawiter-1
   Y = fft(y((k-1)*M+[nu+1:M]+d1),N)*sqrt(2)/sqrt(N);
   Xest((k-1)*(2*N)+[1:2:2*N]) = sign(real(Y));
   Xest((k-1)*(2*N)+[2:2:2*N]) = sign(imag(Y));
   % differential decoding:
   temp = (angle(Y) - angle(Yinit))/pi;
   Ydd((k-1)*(2*N)+[1:2:2*N]) = sign(mod(temp+0.25,2)-1);
   Ydd((k-1)*(2*N)+[2:2:2*N]) = sign(mod(temp-0.25,2)-1);
   Yinit = Y;
end % untuk k=1:rawiter-1

BER(SNRi,1) = BER(SNRi,1) + sum(Ydd(2*N+1:2*(N-1)*rawiter) ...
   ~= Xdd(2*N+1:2*(N-1)*rawiter)) / (2*(N-2)*rawiter);

% 2: MMSE, asumsi sinyal dan noise putih ----

% hitung delay optimal dan TEQ MMSE
[w2,d2] = mmse(h,nu+1,tLw,dmin,dmax,ssx,ssn);

y = conv(r,w2);

Xest = zeros(size(Xtrue));
Ydd = zeros(size(Xtrue));
Yinit = (1+j)*ones(N,1);
for k=1:rawiter-1
   Y = fft(y((k-1)*M+[nu+1:M]+d2),N)*sqrt(2)/sqrt(N);
   Xest((k-1)*(2*N)+[1:2:2*N]) = sign(real(Y));
   Xest((k-1)*(2*N)+[2:2:2*N]) = sign(imag(Y));
   % decoding diferensial:
   temp = (angle(Y) - angle(Yinit))/pi;
   Ydd((k-1)*(2*N)+[1:2:2*N]) = sign(mod(temp+0.25,2)-1);
   Ydd((k-1)*(2*N)+[2:2:2*N]) = sign(mod(temp-0.25,2)-1);
   Yinit = Y;
end % untuk k=1:rawiter-1

BER(SNRi,2) = BER(SNRi,2) + sum(Ydd(2*N+1:2*(N-1)*rawiter) ...
   ~= Xdd(2*N+1:2*(N-1)*rawiter)) / (2*(N-2)*rawiter);

% 3: MSSNR ----------------------------------

% hitung delay optimum dan TEQ MSSNR
[w3,d3] = mssnr(h,nu+1,tLw,dmin,dmax);

y = conv(r,w3);

Xest = zeros(size(Xtrue));
Ydd = zeros(size(Xtrue));
Yinit = (1+j)*ones(N,1);
for k=1:rawiter-1
   Y = fft(y((k-1)*M+[nu+1:M]+d3),N)*sqrt(2)/sqrt(N);
   Xest((k-1)*(2*N)+[1:2:2*N]) = sign(real(Y));
```

```
   Xest((k-1)*(2*N)+[2:2:2*N]) = sign(imag(Y));
   % decoding diferensial:
   temp = (angle(Y) - angle(Yinit))/pi;
   Ydd((k-1)*(2*N)+[1:2:2*N]) = sign(mod(temp+0.25,2)-1);
   Ydd((k-1)*(2*N)+[2:2:2*N]) = sign(mod(temp-0.25,2)-1);
   Yinit = Y;
end % untuk k=1:rawiter-1

BER(SNRi,3) = BER(SNRi,3) + sum(Ydd(2*N+1:2*(N-1)*rawiter) ...
   ~= Xdd(2*N+1:2*(N-1)*rawiter)) / (2*(N-2)*rawiter);

% 4: MDS-Q --------------------------------

% hitung delay optimum dan TEQ MDS-Q
nmin = 0;      % lokasi centroid terkecil
nmax = tLc-1;  % lokasi centroid terbesar
LorQ = 1;      % weights quadratic
[w4,d4,dummy] = mdsteq(h,tLw,nmin,nmax,LorQ,alpha,ssx,ssn);

y = conv(r,w4);

Xest = zeros(size(Xtrue));
Ydd = zeros(size(Xtrue));
Yinit = (1+j)*ones(N,1);
for k=1:rawiter-1
   Y = fft(y((k-1)*M+[nu+1:M]+d4),N)*sqrt(2)/sqrt(N);
   Xest((k-1)*(2*N)+[1:2:2*N]) = sign(real(Y));
   Xest((k-1)*(2*N)+[2:2:2*N]) = sign(imag(Y));
   % decoding diferensial:
   temp = (angle(Y) - angle(Yinit))/pi;
   Ydd((k-1)*(2*N)+[1:2:2*N]) = sign(mod(temp+0.25,2)-1);
   Ydd((k-1)*(2*N)+[2:2:2*N]) = sign(mod(temp-0.25,2)-1);
   Yinit = Y;
end % untuk k=1:rawiter-1

BER(SNRi,4) = BER(SNRi,4) + sum(Ydd(2*N+1:2*(N-1)*rawiter) ...
   ~= Xdd(2*N+1:2*(N-1)*rawiter)) / (2*(N-2)*rawiter);

% 5: MDS-L ---------------------------------

% hitung delay optimum dan TEQ MDS-L
nmin = 0;      % lokasi centroid terkecil
nmax = tLc-1;  % lokasi centroid terbesar
LorQ = 0;      % weights linier
[w5,d5,dummy] = mdsteq(h,tLw,nmin,nmax,LorQ,alpha,ssx,ssn);

y = conv(r,w5);

Xest = zeros(size(Xtrue));
Ydd = zeros(size(Xtrue));
Yinit = (1+j)*ones(N,1);
for k=1:rawiter-1
   Y = fft(y((k-1)*M+[nu+1:M]+d5),N)*sqrt(2)/sqrt(N);
   Xest((k-1)*(2*N)+[1:2:2*N]) = sign(real(Y));
   Xest((k-1)*(2*N)+[2:2:2*N]) = sign(imag(Y));
   % decoding diferensial:
```

```
   temp = (angle(Y) - angle(Yinit))/pi;
   Ydd((k-1)*(2*N)+[1:2:2*N]) = sign(mod(temp+0.25,2)-1);
   Ydd((k-1)*(2*N)+[2:2:2*N]) = sign(mod(temp-0.25,2)-1);
   Yinit = Y;
end % untuk k=1:rawiter-1

BER(SNRi,5) = BER(SNRi,5) + sum(Ydd(2*N+1:2*(N-1)*rawiter) ...
   ~= Xdd(2*N+1:2*(N-1)*rawiter)) / (2*(N-2)*rawiter);

% 6: Min-IBI ----------------------------------

% hitung delay optimum dan TEQ MDS-L
LorQ = 0;      % weights linier
[w6,d6] = minibi(h,tLw,nu,dmin,dmax,LorQ,alpha,ssx,ssn);

y = conv(r,w6);

Xest = zeros(size(Xtrue));
Ydd = zeros(size(Xtrue));
Yinit = (1+j)*ones(N,1);
for k=1:rawiter-1
   Y = fft(y((k-1)*M+[nu+1:M]+d6),N)*sqrt(2)/sqrt(N);
   Xest((k-1)*(2*N)+[1:2:2*N]) = sign(real(Y));
   Xest((k-1)*(2*N)+[2:2:2*N]) = sign(imag(Y));
   % decoding diferensial:
   temp = (angle(Y) - angle(Yinit))/pi;
   Ydd((k-1)*(2*N)+[1:2:2*N]) = sign(mod(temp+0.25,2)-1);
   Ydd((k-1)*(2*N)+[2:2:2*N]) = sign(mod(temp-0.25,2)-1);
   Yinit = Y;
end % untuk k=1:rawiter-1

BER(SNRi,6) = BER(SNRi,6) + sum(Ydd(2*N+1:2*(N-1)*rawiter) ...
   ~= Xdd(2*N+1:2*(N-1)*rawiter)) / (2*(N-2)*rawiter);

% ---------------------------------------------

end % untuk SNRi = 1:Lsnr
waitbar(channum/Nchans,hh);
end % untuk channum=1:Nchans
close(hh);
%----------------------------------------

BER = BER/Nchans;

figure(1); clf;
set(0,'DefaultAxesFontSize',16);
semilogy(SNRdB,BER(:,1),'bo-',...
     SNRdB,BER(:,2),'rs-',...
     SNRdB,BER(:,3),'gx-',...
     SNRdB,BER(:,4),'kd-',...
     SNRdB,BER(:,5),'kd--',...
     SNRdB,BER(:,6),'m+-.');
axis([SNRdB(1) SNRdB(end) 1e-4 0.5]);
%title('BER vs. SNR','FontSize',16);
xlabel('signal-to-noise ratio (SNR), in dB','FontSize',16);
ylabel('bit error rate (BER)','FontSize',16);
```

```
legend('No TEQ','MMSE','MSSNR','MDS-Q','MDS-L','Min-IBI',3);
grid on;
```