```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
%                          mimo_main.m
% Program ini digunakan untuk mengestimasi parameter MIMO dalam domain
% frekuensi
%
% Penggabungan matriks diagonalisasi untuk mendapatkan hasil yang lebih
% baik
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

Program_START = clock;           % Set timer pada program

Over_Estimate_R_and_C=1          % estimasi korelasi dan kumulan

Modify_Hest_Mag_by_Order_Constrain=0

Process_hest_CUM_his=1              % untuk tujuan statistik, estimasi
kanal impuls dibandingkan dengan yang sebenarnya.

Equalizer_Length=15;

MAX_CORRELATION_LENGTH=100;

L = 5                            % Panjang kanal termasuk h(0).

Le=L+4                           % Perpanjangan panjang kanal.

if Le > L+1
   MAX_Minus_ORDER_ifft=floor((Le-L)/3);
else
   MAX_Minus_ORDER_ifft=0;
end

MAX_Minus_ORDER=1;
MAX_Plus_ORDER=Le-MAX_Minus_ORDER-1;

sum_w1_w2=0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% setting pada sistem MIMO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m = 2                           %%% Jumlah dari sinyal keluaran
n = 2                           %%% Jumlah dari sinyal masukan


SISTEM_REAL=1

L_extend=L
%L_extend=Le-L
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% setting pada simulasi Monte-Carlo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N=1024*4                        %%% Panjang sinyal masukan(keluaran)
seg_length = 128*4;              %%% Panjang segmen menggunakan estimasi
pada penyilangan kumulan
seg_num=N/seg_length;           %%% Jumlah segmen

NF = 128                        %%% Estimasi menggunakan panjang FFT dan
juga meniadakan resolusi frekwensi

MAX_CORRELATION_LENGTH_ifft=NF;       %%% Perhitungan hest_ifft langsung
dari ifft
RUN_TIMES=3                     %%% Jumlah pengujian Monte-Carlo

ADD_NOISE=1                        %%% Observasi gangguan dilakukan bila
diset 1


SNR=10                          %%% SNR pada observasi sinyal

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Setting pada statistik orde kedua
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if Over_Estimate_R_and_C
   R_LENGTH=2*(Le-1)
else
   R_LENGTH=L-1
end

ADD_WINDOW=1
ADD_WINDOW=ADD_WINDOW & Over_Estimate_R_and_C

S_x_symmetry=1
V_diagonal_real=1
Rx_diagonal_real=1
Select_Freq_by_Rx_cond=1

Rx_cond_selection_percentage=80 %%% Berapa banyak frekwensi yang dipilih
berdasarkan jumlah kondisi pada penyilangan spectrum daya P_x(w), dalam
persentase.

interpolate_V=1                      %%% V(w) adalah matriks white yang
merupakan inverse akar dari penyilangan matriks korelasi P_x(w).
CHOOSE_EIG=1;                      %%% Estimasi sistem hanya menggunakan
bagian frekwensi

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Setting pada orde statistik yang lebih tinggi
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Over_Estimate_R_and_C
   C_LENGTH=2*(Le-1)
else
```

```
   C_LENGTH=L-1
end

ADD_CUM_WINDOW=1
win_width=C_LENGTH;

Using_Bispectrum=1;             %%% Menggunaka orde ketiga kumulan bila diset 1 dan bila diset 0 maka digunakan orde keempat kumulan untuk estimasi

CUMULANTS_ARE_REAL = 0;          %%% Bila diset 1 maka estimasi kumulan adalah sebenarnya

Calculate_B_x_Using_FFT2=1;       %%% metoda ini lebih cepat dibanding metoda langsung

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Parameters pada kontruksi matriks polispektra
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Select_Rx1_index=1;
w2=1
w3_equal_0=1                       %%% Hanya pada orde keempat kumulan, bila di set 1 maka w3=0

w2_equal_minus_w3_plus_arfa=0     %%% Hanya pada orde keempat kumulan, bila di set 1 maka w2+w3=arfa

Select_Freq_by_SVD=0;

Smat_std_plus_coeff=2
Smat_std_minus_coeff=2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Setting pada penggabunggan matriks diagonalisasi
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Using_Joint_Diag=1;                %%% Menggunakan gabungan matriks dekomposisi untuk mengestimasi matriks orthogonal w(w), ada 2 jenis penggabungan matriks dekomposisi, pertama adalah penggabungan matriks diagonalisasi Cardoso, kedua adalah penggabungan SVD Pesquet

Freq_Select_Ratio=0.16;
Ref_Frequencies=0:1:(NF/2-1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Setting pada Estimasi Fasa
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

k_arfa=1
i_phase_index_1=1
i_phase_index_2=2

Using_Pesquet_phase=0
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Setting pada rekontruksi masukan
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

RECONSTRUCT_INPUT=1;                %%% Rekontruksi masukan menggunakan
filter Wiener bila diset 1
Minimum_Phase_Sistem=0;

Modify_Hest_by_Phase_est=1

Limit_Delay_time=1


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Beberapa parameter pada pengujian algoritma
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Roots_Amplitude=1.50;

Diagonal_amplitude=1;           %%% Parameter ini merubah amplituda dari
elemen-elemen diagonal pada matriks respon impulse MIMO sistem.

PLOT_PHASE=0;                   %%% Plot gambar fasa atau tidak.


if Using_Bispectrum
   GAMMA=2;
else
   %GAMMA=-1.2;
   %GAMMA=-1.2;
   %GAMMA=-2;
   GAMMA=-1;
   %GAMMA=6;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  Setting parameter berakhir
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if n < m
   Less_Input_than_Output=1
else
   Less_Input_than_Output=0
end


hest_CUM_his=zeros(MAX_Minus_ORDER+MAX_Plus_ORDER+1,m,n,RUN_TIMES);


format compact;
i=sqrt(-1);
```

```
CIRCLE=exp(i*(0:1000)*2*pi/1000);

r_window = kaiser(2*R_LENGTH+1,7);
%r_window=hamming(2*R_LENGTH+1);

%d_win=zeros(1,1+win_width);
%d_win=abs(sin((0:win_width)*pi/win_width))/pi          +          (1.0-
(0:win_width)/win_width).*cos((0:win_width)*pi/win_width);
d_win = kaiser(2*win_width+1,7);
d_win = d_win(win_width+1:2*win_width+1);
%% Parzen Window
%d_win(1:1+floor(win_width/2))=1-
6*((0:floor(win_width/2))/win_width).^2+6*((0:floor(win_width/2))/win_wi
dth).^3;
%d_win(2+floor(win_width/2):win_width+1)=2*(1-
(1+floor(win_width/2):win_width)/win_width).^3;

dd_win=zeros(1,2*win_width+1);
dd_win(win_width+1:2*win_width+1)=d_win;
dd_win(1:win_width)=d_win(win_width+1:-1:2);

cum_win=zeros(win_width*2+1);
for ii=-win_width:win_width
   for jj=-win_width:win_width
      if abs(ii-jj) <= win_width

cum_win(ii+win_width+1,jj+win_width+1)=d_win(abs(ii)+1)*d_win(abs(jj)+1)
*d_win(abs(ii-jj)+1);
      end
   end
end


eig_index_his=zeros(NF,RUN_TIMES);

h_amplitude=ones(m)+eye(m)*(Diagonal_amplitude-1);

h=zeros(L,m,m);

if 0              %% Desain matriks respon impulse sistem h(t,i,j)
   for ii=1:m
      for jj=1:m
         Zeros = roots(2*rand(1,L)-1);
         Zeros       =       Roots_Amplitude*tanh(abs(Zeros))       .*
exp(j*angle(Zeros));
         h(:,ii,jj)=real((poly(Zeros)));
      end
   end
end

if 1   %%% Fasa non minimum  m=2; L=5
h(:,1,1) = [ 1.0000   -1.5537   -0.0363    0.5847    0.5093];
h(:,1,2) = [ 1.0000    2.2149    1.0828   -1.1731   -0.8069];
h(:,2,1) = [ 1.0000    0.9295    0.2453   -0.7510    0.3717];
```

```
h(:,2,2) = [ 1.0000   -0.7137   -1.5079    1.6471   -1.2443];
end

if 0  %%% Fasa non minium  m=3; L=5
h(:,1,1) = [ 1.0000   -1.5537   -0.0363    0.5847    0.5093];
h(:,1,2) = [ 1.0000    2.2149    1.0828   -1.1731   -0.8069];
h(:,1,3) = [ 1.0000   -1.7325    0.4123    0.3471   -0.2796];
h(:,2,1) = [ 1.0000    0.9295    0.2453   -0.7510    0.3717];
h(:,2,2) = [ 1.0000   -0.7137   -1.5079    1.6471   -1.2443];
h(:,2,3) = [ 1.0000    2.1911    1.7313   -0.1818   -0.2214];
h(:,3,1) = [ 1.0000   -1.0191   -1.5532    1.5117   -0.7217];
h(:,3,2) = [ 1.0000    2.0637    0.8907   -0.3785   -0.3789];
h(:,3,3) = [ 1.0000   -0.6879   -0.8976   -0.6126   -0.1318];
end

%%% Peningkatan matriks respon impulse pada sistem MIMO
for ii=1:m
   for jj=1:m
      h(:,ii,jj)=h(:,ii,jj)*h_amplitude(ii,jj);
   end
end

%%% Domain respon frekwensi MIMO sistem, matriks fungsi transfer sistem
H = fft(h,NF,1);


if Select_Rx1_index
   if Less_Input_than_Output
      H_0=reshape(H(sum_w1_w2+1,:,1:n),m,n);
      for row=1:m
         H_0_dummy=sort(abs(H_0(row,:)));
         min_ratio(row)=min(H_0_dummy(n:-1:2)./H_0_dummy(n-1:-1:1));
      end
      [max_min_ratio Rx1_index]=max(min_ratio);
   else
      H_0=reshape(H(sum_w1_w2+1,:,:),m,m);
      for row=1:m
         H_0_dummy=sort(abs(H_0(row,:)));
         min_ratio(row)=min(H_0_dummy(m:-1:2)./H_0_dummy(m-1:-1:1));
      end
      [max_min_ratio Rx1_index]=max(min_ratio);
   end
else
   Rx1_index=1;
end


H_order=zeros(1,n);

HH=shiftdim(H,1);               %%% Fungsi Transfer sistem MIMO, m x n x
NF.
Phase_true=angle(HH);           %%% Fasa sebenarnya dari sistem MIMO


Ryn = zeros(m,m,Freq_number,NF);
```

```
Cyn = zeros(m,m,Freq_number,NF);    %%      Ryn'*Ryn,digunakan      pada
penggabungan matriks diagonalisasi Cardoso
Wn  = zeros(m,m,NF);               %%% Estimasi matriks orthogonal W(w)
berdasarkan pada penggabungan matriks diagonalisasi.

if w2_equal_minus_w3_plus_arfa & ~Using_Bispectrum
   arfa_matriks=ones(2*C_LENGTH+1,2*C_LENGTH+1,2*C_LENGTH+1);
   for ii=-C_LENGTH:C_LENGTH

arfa_matriks(:,:,ii+C_LENGTH+1)=arfa_matriks(:,:,ii+C_LENGTH+1)*exp(-
j*arfa_w2_plus_w3*ii/NF);
   end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  Penggujian metoda Monte-Carlo dimulai
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for iloop=1:RUN_TIMES

   N
   iloop
   rand('state',sum(100*clock));

   if Using_Bispectrum
      %s = -log(rand(m,N))-log(rand(m,N))*i;
      s = -log(rand(m,N));
   else  % Using Trispectrum
      s = -log(rand(m,N));
      %s = rand(m,N)+i*rand(m,N);

      for ii=1:m
         %s(ii,:)=qam(N,2,465*sum(100*clock),54.564*sum(100*clock));
      end

      %s = rand(m,N);
      %s = s-(mean(s'))'*ones(1,N);
      %s=sign(s);
   end

   s = s-(mean(s'))'*ones(1,N);
   s=diag(1./std(s,0,2))*s;       %% Sinyal masukan
   if Less_Input_than_Output
      s(n+1:m,:)=0;
   end



   x = zeros(size(s));
   for ii=1:m
      for jj=1:n
         x(ii,:) = x(ii,:) + filter(h(:,ii,jj),1,s(jj,:));
      end
   end
   x = x-(mean(x'))'*ones(1,N);
```

```
   std_x_1=std(x(1,:));
   std_x_2=std(x(2,:));

   if ADD_NOISE
      for ii=1:m
         rand('state',sum(100*clock)*rand);
         noise=randn(1,N);
         noise=noise-mean(noise);
         power_coefficient = (10^(SNR/20))*std(noise)/std(x(ii,:));
         x(ii,:) = x(ii,:)+noise/power_coefficient;
      end
   end

   x(1,:) = std_x_1*x(1,:)/std(x(1,:));
   x(2,:) = std_x_2*x(2,:)/std(x(2,:));


   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Proses dimulai
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%% Estimasi pada penyilangan korelasi
   rx = zeros(2*R_LENGTH+1,m,m);
   for ii=1:m
      for jj=1:m
         for seg=0:seg_num-1
            rx(:,ii,jj) = rx(:,ii,jj) + reshape(xcorr(...
               x(ii,seg_length*seg+1:seg_length*(seg+1)),...

x(jj,seg_length*seg+1:seg_length*(seg+1)),R_LENGTH,'unbiased'),2*R_LENGT
H+1,1);
         end

         if MATLAB_VERSION_NUMBER(1) >= '6'
             rx(:,ii,jj)=flipud(rx(:,ii,jj));
         end

      end
   end
   rx=rx/seg_num;



   if ADD_WINDOW
      for ii=1:m
         for jj=1:m
            rx(:,ii,jj)=rx(:,ii,jj).*r_window;
         end
      end
   end



   %% Estimasi pada penyilangan spectrum daya
```

```
   S_x=zeros(NF,m,m);
   for ii=1:m
      for jj=1:m
         rr=zeros(NF,1);
         rr(NF/2-
R_LENGTH+1:NF/2+R_LENGTH+1)=reshape(rx(:,ii,jj),2*R_LENGTH+1,1);
         S_x(:,ii,jj)=fft(fftshift(rr),NF);
      end
   end

   if S_x_symmetry
      for w=1:NF
         S_x_dummy=reshape((S_x(w,:,:)+S_x(w,:,:))/2,m,m);
         S_x(w,:,:)=(S_x_dummy+S_x_dummy')/2;
      end
   end



   %% Estimasi pada penyilangan kumulan
   TSTART = clock;
   if w2_equal_minus_w3_plus_arfa & ~Using_Bispectrum
      cx = zeros(2*C_LENGTH+1,4*C_LENGTH+1,m,m,m);
   else % w3=0
      cx = zeros(2*C_LENGTH+1,2*C_LENGTH+1,m,m,m);
   end

   if ~Using_Bispectrum
      cx_phase = zeros(2*C_LENGTH+1,2*C_LENGTH+1,m,m,m);
   end

   if ADD_CUM_WINDOW
      for kk=1:2*C_LENGTH+1

cum_4_window_4(:,:,kk)=ones(2*C_LENGTH+1,2*C_LENGTH+1)*dd_win(kk);
      end
   end

   for ii=1:m
      for jj=1:m
         for kk=1:m
            for seg=0:seg_num-1
               if Using_Bispectrum
                  cx(:,:,ii,jj,kk)=cx(:,:,ii,jj,kk)+g_cc3_matriks(...
                     x(ii,seg_length*seg+1:seg_length*(seg+1)),...
                     x(jj,seg_length*seg+1:seg_length*(seg+1)),...

x(kk,seg_length*seg+1:seg_length*(seg+1)),C_LENGTH)/seg_num;
               else  % Using Trispectrum

                  %TSTART_matriks = clock;
                  cx_3D = g_cc4_3D(...
                     x(ii,seg_length*seg+1:seg_length*(seg+1)),...
                     conj(x(jj,seg_length*seg+1:seg_length*(seg+1))),...
                     x(kk,seg_length*seg+1:seg_length*(seg+1)),...
```

```
conj(x(kk,seg_length*seg+1:seg_length*(seg+1))),C_LENGTH);
                  %TIME_matriks = etime(clock,TSTART_matriks)

                  %TSTART_mex = clock;
                  %cx_3D = g_ccum4_3D_mex(...  %% Using the mex function
for fast speed
                  %x(ii,seg_length*seg+1:seg_length*(seg+1)).',...
                  %conj(x(jj,seg_length*seg+1:seg_length*(seg+1))).',...
                  %x(kk,seg_length*seg+1:seg_length*(seg+1)).',...

%conj(x(kk,seg_length*seg+1:seg_length*(seg+1))).',C_LENGTH);
                  %TIME_mex = etime(clock,TSTART_mex)



                  cx_phase_3D = g_cc4_3D(...
                     x(ii,seg_length*seg+1:seg_length*(seg+1)),...
                     conj(x(jj,seg_length*seg+1:seg_length*(seg+1))),...
                     x(kk,seg_length*seg+1:seg_length*(seg+1)),...

conj(x(ii,seg_length*seg+1:seg_length*(seg+1))),C_LENGTH);

                  if w2_equal_minus_w3_plus_arfa
                     if ADD_CUM_WINDOW
                        cx_3D=cx_3D.*cum_4_window_4;
                     end

                     cx_3D=cx_3D.*arfa_matriks;

                     cx_dummy=zeros(2*C_LENGTH+1,4*C_LENGTH+1);
                     for dd=-2*C_LENGTH:2*C_LENGTH
                        for t1=1:2*C_LENGTH+1

cx_dummy(t1,dd+2*C_LENGTH+1)=sum(diag(shiftdim(cx_3D(t1,:,:),1),-dd));
                        end
                     end
                     cx(:,:,ii,jj,kk)      =      cx(:,:,ii,jj,kk)      +
cx_dummy/seg_num;
                  else  % w3_equal_0
                     if ADD_CUM_WINDOW
                        cx(:,:,ii,jj,kk)     =     cx(:,:,ii,jj,kk)     +
sum(cx_3D.*cum_4_window_4,3)/seg_num;
                     else
                        cx(:,:,ii,jj,kk)     =     cx(:,:,ii,jj,kk)     +
sum(cx_3D,3)/seg_num;
                     end
                  end

                  if ADD_CUM_WINDOW
                     cx_phase(:,:,ii,jj,kk)  =  cx_phase(:,:,ii,jj,kk)  +
sum(cx_phase_3D.*cum_4_window_4,3)/seg_num;
                  else
                     cx_phase(:,:,ii,jj,kk)  =  cx_phase(:,:,ii,jj,kk)  +
sum(cx_phase_3D,3)/seg_num;
                  end
```

```
            end
         end
      end
   end
end

TIME_cx = etime(clock,TSTART)

if CUMULANTS_ARE_REAL
   cx=real(cx);

   if ~Using_Bispectrum
      cx_phase=real(cx_phase);
   end
end



if ADD_CUM_WINDOW
   for ii=1:m
      for jj=1:m
         for kk=1:m
            cx(:,:,ii,jj,kk) = cx(:,:,ii,jj,kk).*cum_win;
         end
      end
   end
end



%% Estimasi pada penyilangan polispektra pada estimasi magnitude
TSTART = clock;
B_x=zeros(NF,1,m,m,m);
B_xn=zeros(NF,Freq_number,m,m,m);
if w2_equal_minus_w3_plus_arfa & ~Using_Bispectrum
   for ii=1:m
      for jj=1:m
         for kk=1:m
            cx_dummy=zeros(NF);
            cx_dummy((NF/2+1-C_LENGTH) : (NF/2+1+C_LENGTH) , (NF/2+1-
2*C_LENGTH) : (NF/2+1+2*C_LENGTH))=cx(:,:,ii,jj,kk);
            cx_dummy=fftshift(cx_dummy);
            B_x_dummy=(fft2(cx_dummy,NF,NF));
            B_x(:,1,ii,jj,kk)=B_x_dummy(:,w2);

B_xn(:,1:Freq_number,ii,jj,kk)=B_x_dummy(:,Ref_Frequencies);
         end
      end
   end
else % gunakan bispectrum atau w3=0
   for ii=1:m
      for jj=1:m
         for kk=1:m
            %%% Merupakan irisan diagonal pada (w, \beta-w)
            cx_dummy=zeros(NF);
```

```
               cx_dummy((NF/2+1-C_LENGTH) : (NF/2+1+C_LENGTH) , (NF/2+1-
C_LENGTH) : (NF/2+1+C_LENGTH))=cx(:,:,ii,jj,kk);
               cx_dummy=fftshift(cx_dummy);
               B_x_dummy=(fft2(cx_dummy,NF,NF));
               all_w=1:NF;
               B_x(:,1,ii,jj,kk)=diag(B_x_dummy(all_w,mod(sum_w1_w2+1-
all_w, NF)+1));

               for Freq_Index=1:Freq_number

B_xn(:,Freq_Index,ii,jj,kk)=diag(B_x_dummy(all_w,mod(Ref_Frequencies(Fre
q_Index)+1-all_w,NF)+1));
               end
            end
         end
      end
   end

   TIME_B_x = etime(clock,TSTART)


   %% Estimasi pada penyilangan polispektra pada fasa retrieval
   if Using_Bispectrum
      B_x_phase=zeros(NF,m,m,m);
      for ii=1:m
         for jj=1:m
            for kk=1:m
               cx_dummy=zeros(NF);
               cx_dummy((NF/2+1-C_LENGTH) : (NF/2+1+C_LENGTH) , (NF/2+1-
C_LENGTH) : (NF/2+1+C_LENGTH))=cx(:,:,ii,jj,kk);
               cx_dummy=fftshift(cx_dummy);
               B_x_dummy=(fft2(cx_dummy,NF,NF));

               B_x_phase(NF,ii,jj,kk)=B_x_dummy(2,mod(k_arfa-1, NF)+1);
               B_x_dummy(2:NF-1,:)=B_x_dummy(NF:-1:3,:);
               B_x_phase(1:NF-k_arfa,ii,jj,kk)=diag(B_x_dummy(1:NF-
k_arfa, mod(k_arfa,NF)+1:NF));
               if k_arfa > 1
                  B_x_phase(NF-k_arfa+1:NF-
1,ii,jj,kk)=diag(B_x_dummy(NF-k_arfa+1:NF-1, 1:k_arfa-1));
               end % if k_arfa > 1
            end % kk
         end % jj
      end % ii
   else % Using Trispectrum
      B_x_phase=zeros(NF,m,m,m);
      for ii=1:m
         for jj=1:m
            for kk=1:m
               cx_dummy=zeros(NF);
               cx_dummy((NF/2+1-C_LENGTH) : (NF/2+1+C_LENGTH) , (NF/2+1-
C_LENGTH) : (NF/2+1+C_LENGTH))=cx_phase(:,:,ii,jj,kk);
               cx_dummy=fftshift(cx_dummy);
               B_x_dummy=(fft2(cx_dummy,NF,NF));

               B_x_phase(NF,ii,jj,kk)=B_x_dummy(2,mod(k_arfa-1, NF)+1);
```

```
               B_x_dummy(2:NF-1,:)=B_x_dummy(NF:-1:3,:);
               B_x_phase(1:NF-k_arfa,ii,jj,kk)=diag(B_x_dummy(1:NF-
k_arfa, mod(k_arfa,NF)+1:NF));
               if k_arfa > 1
                  B_x_phase(NF-k_arfa+1:NF-
1,ii,jj,kk)=diag(B_x_dummy(NF-k_arfa+1:NF-1, 1:k_arfa-1));
               end
            end
         end
      end
   end

   Rx=reshape(shiftdim(S_x(:,:,:),1),m,m,NF);       %% The cross power
spectrum matriks
   Rx1=reshape(shiftdim(B_x(:,1,Rx1_index,:,:),2),m,m,NF);  %% The cross
polyspectra matriks


   for Freq_Index=1:Freq_number  %% Rxn is a "m x m x NF x Freq_number"
matriks, the last m is the index ii

Rxn(:,:,:,Freq_Index)=reshape(shiftdim(B_xn(:,Freq_Index,Rx1_index,:,:),
3),m,m,NF);  %% The cross polyspectra matriks
   end

   jj=0;
   for ii=1:m
      if ii ~= Rx1_index
         jj=jj+1;
         for Freq_Index=1:Freq_number

Rxn(:,:,:,Freq_Index+jj*Freq_number)=reshape(shiftdim(B_xn(:,Freq_Index,
ii,:,:),3),m,m,NF);
         end
      end
   end



   if Rx_diagonal_real
      for w=1:NF
         Rx(:,:,w) = Rx(:,:,w)-i*imag(diag(diag(Rx(:,:,w))));
      end
   end

   %%% Perhitungan jumlah kondisi dari penyilangan matriks spectrum daya
   for w=1:NF
      Rx_eig(:,w)=sort(abs(real(svd(Rx(:,:,w)))));
      Rx_cond(w,1)=real(Rx_eig(m,w)/Rx_eig(1+m-n,w));
   end

   [Rx_cond_sort Rx_cond_sort_index]=sort(Rx_cond);


Rx_cond_max=Rx_cond(Rx_cond_sort_index(floor(NF*Rx_cond_selection_percen
tage/100)));
```

```
   Rx_cond_Index=[]; %% Select freqs by the conditiona number of Rx(w)
   for w=1:NF
      if Rx_cond(w,1) < Rx_cond_max
         Rx_cond_Index=[Rx_cond_Index w];
      end
   end

   Rx_INDEX=zeros(1,NF);
   Rx_INDEX(Rx_cond_Index)=1;

   %%% Perhitungan V(w) dan V^{-1}(w)
   if Less_Input_than_Output
      for w=1:NF
         [Urx Drx Vrx]=svd(Rx(:,:,w));
         Dv=diag(Drx);
         Dv(1+n:m)=0.000001;
         Dv_1=sqrt(Dv);
         Dv(1:n)=1./sqrt(Dv(1:n));
         V(:,:,w) = Urx*diag(Dv)*Vrx';
         V_1(:,:,w) = Urx*diag(Dv_1)*Vrx';
      end
   else
      for w=1:NF
         V_1(:,:,w) = sqrtm(Rx(:,:,w));
         if Rx_cond(w,1) < Rx_cond_max |1
            V(:,:,w) = inv(V_1(:,:,w));
         else
            [Urx Drx Vrx]=svd(Rx(:,:,w));
            Dv=diag(Drx);
            Dv(1:m-1)=1./sqrt(Dv(1:m-1));
            Dv(m)=0.000001;
            V(:,:,w) = Urx*diag(Dv)*Vrx';
         end
      end
   end

   if V_diagonal_real
      for w=1:NF
         V(:,:,w) = V(:,:,w)-i*imag(diag(diag(V(:,:,w))));
      end
   end


   %%% Interpolasi V dan V^{-1} pada frekwensi menggunakan jumlah
kondisi tertinggi
   if interpolate_V
      for ii=1:m
         for jj=1:m

V_inter_real(:,ii,jj)=interpolate_func(reshape(real(V(ii,jj,:)),NF,1),Rx
_cond_Index);

V_inter_imag(:,ii,jj)=interpolate_func(reshape(imag(V(ii,jj,:)),NF,1),Rx
_cond_Index);
         end
```

```
      end

      V_inter=shiftdim(V_inter_real+i*V_inter_imag,1);

      V=V_inter;
   end %% End of if interpolate_V


   %%% Get Ry1(w)
   if w2_equal_minus_w3_plus_arfa & ~Using_Bispectrum
      for w=1:NF
         Ry1(:,:,w)            =            V(:,:,mod(w+arfa_w2_plus_w3-
1,NF)+1)*Rx1(:,:,w)*V(:,:,w);
      end
   else
      for w=1:NF
         Ry1(:,:,w)                  =                 V(:,:,mod(NF+1-w,
NF)+1)*Rx1(:,:,w)*V(:,:,mod(sum_w1_w2+NF+1-w, NF)+1)';
      end
   end


   %%% perhitungan Ryn(w) dan Cyn, Cyn=Ryn'*Ryn.
   if w2_equal_minus_w3_plus_arfa & ~Using_Bispectrum
      for w=1:NF  %% Cyn is a "m x m x Freq_number x NF" matriks, the
last m is the index ii
         for Freq_Index=1:Freq_number*m %% Cyn=Ryn^{H} x Ryn
            Ryn_dummy           =           V(:,:,mod(w+arfa_w2_plus_w3-
1,NF)+1)*Rxn(:,:,w,Freq_Index)*V(:,:,w);
            Ryn(:,:,Freq_Index,w) = Ryn_dummy;
            Cyn(:,:,Freq_Index,w) = Ryn_dummy'*Ryn_dummy;
         end
      end
   else
      for w=1:NF  %% Cyn is a "m x m x Freq_number x NF" matriks, the
last m is the index ii
         for Freq_Index=1:Freq_number %% Cyn=Ryn^{H} x Ryn
            for ii=0:m-1
               Ryn_dummy                =                V(:,:,mod(NF+1-w,
NF)+1)*Rxn(:,:,w,Freq_Index+Freq_number*ii)*V(:,:,mod(Ref_Frequencies(Fr
eq_Index)+1-w, NF)+1)';
               Ryn(:,:,Freq_Index+Freq_number*ii,w) = Ryn_dummy;
               Cyn(:,:,Freq_Index+Freq_number*ii,w)                    =
Ryn_dummy*Ryn_dummy';
            end
         end
      end
   end


   %%% Perhitungan the W(w) menggunakan SVD pada satu matriks Ry1.
   for w=1:NF

      [Udummy,Sdummy,Vdummy] = svd(Ry1(:,:,w));
```

```
      Smat(:,w)=real(diag(Sdummy));

      [Smat_sort(:,w),Smat_Index]=sort(abs(Smat(:,w)));

      W(:,:,mod(1-w, NF)+1)=Udummy(:,Smat_Index);     %% The orthogonal
matriks W(w)
   end

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %%% Penggabungan matriks diagonalisasi Cardoso
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %%%
   TSTART = clock;

   Selected_Freq_number=floor(Freq_number*m*Freq_Select_Ratio);
   D_closeness=zeros(Freq_number*m,1);
   for w=1:1  %% Joint Diagonalization of Cyn(:,:,:,w)

      [Udummy,Ddummy]                                                  =
joint_diag(reshape(Cyn(:,:,:,w),m,m*m*Freq_number),0.00000001);

      Ddummy=reshape(Ddummy,m,m,Freq_number*m);

      for ii=1:Freq_number*m
         D_diag(:,ii)=sort(diag(abs(Ddummy(:,:,ii))));

         D_closeness(ii)=(D_diag(m,ii)-D_diag(1,ii))/D_diag(m,ii);
      end

      for w=1:Freq_number
         D_diag(:,w)=sort((abs(HH(1,:,w)))).';
         D_diag(:,w+Freq_number)=sort((abs(HH(2,:,w)))).';
      end

      [D_close_dummy Selected_w2]=sort(-D_closeness);

      Selected_w2=Selected_w2(1:Selected_Freq_number);


      if iloop==1

         Dominant_freq=Selected_w2(1);

         %%% Hitung H_order
         Rx1_index_d=ceil(Dominant_freq/Freq_number);
         if Less_Input_than_Output
            [H_dummy        H_order_i]=sort(reshape(abs(H(Dominant_freq-
(Rx1_index_d-1)*NF/2,Rx1_index_d,1:n)),n,1));

            H_order(H_order_i)=m-n+1:m;
            H_order(n+1:m)=1:m-n;
         else  %% m=n, INPUT=OUTPUT
```

```
            [H_dummy        H_order_i]=sort(reshape(abs(H(Dominant_freq-
(Rx1_index_d-1)*NF/2,Rx1_index_d,1:n)),n,1));

            H_order(H_order_i)=1:n;
            H_order(n+1:m)=n+1:m;
         end

      else
         if ~length(find(Selected_w2 == Dominant_freq))
            Selected_w2=[Dominant_freq Selected_w2.'].';
         end
      end

      Dominant_freq_position=find(Selected_w2 == Dominant_freq);
      if Dominant_freq_position ~= 1

Selected_w2(2:Dominant_freq_position)=Selected_w2(1:Dominant_freq_positi
on-1);
         Selected_w2(1)=Dominant_freq;
      end


   end


   for w=1:NF  %% Joint Diagonalization of Cyn(:,:,:,w)

      [Udummy,Ddummy]                                                 =
joint_diag(reshape(Cyn(:,:,Selected_w2,w),m,m*length(Selected_w2)),0.000
00001);

      Ddummy=reshape(Ddummy,m,m,length(Selected_w2));

      D1(:,w)=abs(diag(Ddummy(:,:,1)));

      [D1_sort(:,w),D1_Index]=sort(abs(D1(:,w)));

      Wn(:,:,mod(1-w,  NF)+1)=Udummy(:,D1_Index);        %%  The  joint
orthogonal matriks W(w)

   end

   TIME_Joint_Daig = etime(clock,TSTART)



   H_abs=reshape(abs(H(w2,Rx1_index,:)),m,1)*abs(GAMMA)
   Smat_mean=mean(Smat_sort,2)
   Smat_std=std(Smat_sort,0,2)
   Smat_mean_array=diag(Smat_mean)*ones(m,NF);

Smat_1_std_array=diag(Smat_mean+Smat_std_plus_coeff*Smat_std)*ones(m,NF)
;
```

```
   Smat__1_std_array=diag(Smat_mean-
Smat_std_minus_coeff*Smat_std)*ones(m,NF);

   if Select_Freq_by_SVD
      EIG_INDEX=find(
(~(Smat_sort(m,:)<Smat__1_std_array(m,:)|Smat_sort(1,:)>Smat_1_std_array
(1,:))) );
   else
      EIG_INDEX=1:NF;
   end


   EIG_ONE=zeros(1,NF);EIG_ONE(EIG_INDEX)=1;
   if Select_Freq_by_Rx_cond
      EIG_INDEX=find(EIG_ONE(:) & Rx_INDEX(:))';
   end

   %%% Plot nilai singular pada semua frekwensi.
   figure(1);clf;hold on;grid;
   plot(Smat_sort');
   plot(Smat_mean_array');
   plot(Smat__1_std_array','.');
   plot(EIG_INDEX,Smat_mean(m),'r*');
   title('Singular values at all frequencies');
   %axis([1 NF 0 10])


   eig_index_his(EIG_INDEX,iloop)=1;
   Smat_his(:,:,iloop)=Smat;
   Smat_mean_his(:,iloop)=Smat_mean;
   Smat_std_his(:,iloop)=Smat_std;

   if Using_Joint_Diag
      W=Wn;
   end

   Hest = zeros(m,m,NF);
   for w=1:NF
      M = V_1(:,:,w) * W(:,:,w);
      Hest(:,:,w) = conj(M);    %%% Estimasi pada sistem fungsi transfer
   end


   if SISTEM_REAL
      Hest(:,:,NF/2+2:NF)=conj(Hest(:,:,NF/2:-1:2));
   end

   Phase_hat=angle(Hest);       %%% Estimasi fasa dengan kejamakkan fasa

   TSTART = clock;
   hest=zeros(size(h));
   if CHOOSE_EIG
      for ii=1:m
         for jj=1:m  %% Rekontruksi fasa minimum respon impulse
```

```
            hest_dummy                                                    =
real(rec_frommag_complex(abs(reshape(Hest(ii,H_order(jj),:),1,NF)),EIG_I
NDEX,L+L_extend));
            hest(1:L+L_extend,ii,H_order(jj)) = hest_dummy;
         end
      end
   else
      for ii=1:m
         for jj=1:m
            hest_dummy                                                    =
real(rec_frommag_complex(abs(reshape(Hest(ii,H_order(jj),:),1,NF)),1:NF,
L+L_extend));
            hest(1:L+L_extend,ii,H_order(jj)) = hest_dummy;
         end
      end
   end
   TIME_Mag_Reconstruction = etime(clock,TSTART)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Fasa retrieval menggunakan matriks special A
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   Phase_matriks_1=zeros(m,m,NF);
   Phase_matriks_2=zeros(m,m,NF);
   for w=1:NF

Phase_matriks_1(:,:,w)=inv(conj(Hest(:,:,w)))*reshape(B_x_phase(w,i_phas
e_index_1,:,:),m,m)...
         *pinv(Hest(:,:,mod(w+k_arfa-1,NF)+1).');

Phase_matriks_2(:,:,w)=inv(conj(Hest(:,:,w)))*reshape(B_x_phase(w,i_phas
e_index_2,:,:),m,m)...
         *pinv(Hest(:,:,mod(w+k_arfa-1,NF)+1).');
   end

   Psi_1=zeros(m,NF);
   Psi_2=zeros(m,NF);
   for ii=1:m
      Psi_1(ii,:)=reshape(angle(Phase_matriks_1(ii,ii,:)),1,NF);
      Psi_2(ii,:)=reshape(angle(Phase_matriks_2(ii,ii,:)),1,NF);
   end

   Phase_1=reshape(Phase_true(i_phase_index_1,:,k_arfa+1),m,1);
   Phase_2=reshape(Phase_true(i_phase_index_2,:,k_arfa+1),m,1);

   Phase_1_sum=sum(Psi_1,2);
   Phase_2_sum=sum(Psi_2,2);

   Linear_phase_1=(Phase_1_sum(H_order)+Phase_1*NF)/pi
   Linear_phase_2=(Phase_2_sum(H_order)+Phase_2*NF)/pi

   Phase_1;
   Phase_1_est=-Phase_1_sum/NF;
   Phase_2;
```

```
   Phase_2_est=-Phase_2_sum/NF;

   Phi_1=zeros(m,NF);
   Phi_2=zeros(m,NF);

   A=hosmatriks(NF, k_arfa);
   A1=inv(A);
   for ii=1:m
      Phi_1(ii,2:NF)          =          (A1*reshape(Psi_1(ii,2:NF),NF-
1,1)+Phase_1_est(ii)*sum(A1,2)).';
      Phi_2(ii,2:NF)          =          (A1*reshape(Psi_2(ii,2:NF),NF-
1,1)+Phase_2_est(ii)*sum(A1,2)).';
   end

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %%%Metoda J.Pesquet pada estimasi fasa
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   Lamda_d=0.003;
   Lamda_a=0.003;

   PSI_1=fft(Psi_1,NF,2);
   PSI_2=fft(Psi_2,NF,2);

   Exp_k_arfa_term=ones(m,1)*(exp(-i*2*pi*k_arfa*(0:NF-1)/NF)-1);
   Exp_1_term=ones(m,1)*(exp(i*2*pi*(0:NF-1)/NF)-1);

   PHI_1  =  (Exp_k_arfa_term.*PSI_1)  ./  (abs(Exp_k_arfa_term).^2  +
Lamda_d*abs(Exp_1_term).^2 + Lamda_a);
   PHI_2  =  (Exp_k_arfa_term.*PSI_2)  ./  (abs(Exp_k_arfa_term).^2  +
Lamda_d*abs(Exp_1_term).^2 + Lamda_a);

   PHI_1(:,1)=zeros(m,1);
   PHI_2(:,1)=zeros(m,1);

   Phi_1_ifft=real(ifft(PHI_1,NF,2));
   Phi_2_ifft=real(ifft(PHI_2,NF,2));

   if Using_Pesquet_phase
      figure(201);clf
      for ii=1:m
         subplot(m,2,ii*2-1);
         plot(real(PHI_1(ii,:)));
         grid;
         title(sprintf('Real part of PHI_1(%d)',ii));

         subplot(m,2,ii*2);
         plot(imag(PHI_1(ii,:)));
         grid;
         title(sprintf('Imag part of PHI_1(%d)',ii));
      end

      figure(202);clf
      for ii=1:m
```

```
      subplot(m,2,ii*2-1);
      plot(real(PHI_2(ii,:)));
      grid;
      title(sprintf('Real part of PHI_2(%d)',ii));

      subplot(m,2,ii*2);
      plot(imag(PHI_2(ii,:)));
      grid;
      title(sprintf('Imag part of PHI_2(%d)',ii));
   end

   figure(203);clf
   for ii=1:m
      subplot(m,1,ii);hold on;
      plot(real(Phi_1_ifft(ii,:)),'b-');
      plot(real(Phi_1(ii,:)),'r:');
      grid;
      title(sprintf('Phi_1(%d)',ii));
   end

   figure(204);clf
   for ii=1:m
      subplot(m,1,ii);hold on;
      plot(real(Phi_2_ifft(ii,:)),'b-');
      plot(real(Phi_2(ii,:)),'r:');
      grid;
      title(sprintf('Phi_2(%d)',ii));
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Estimasi fasa J.C Pesquet berakhir
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if Using_Pesquet_phase
   Phi_1=Phi_1_ifft;
   Phi_2=Phi_2_ifft;
end

   for w=1:NF
   for ii=1:m
      Phase_est_1(:,ii,w)=Phase_hat(:,ii,w)+Phi_1(ii,w)*ones(m,1);
      Phase_est_2(:,ii,w)=Phase_hat(:,ii,w)+Phi_2(ii,w)*ones(m,1);
   end
end

%% Dapat menggunakan Phase_est_1 atau Phase_est_2
Phase_est=Phase_est_1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Modify_Hest_by_Phase_est
   Hest=abs(Hest).*exp(i*Phase_est);
end
```

```
   if Modify_Hest_Mag_by_Order_Constrain
      for (ii=1:m)
         for (jj=1:m)
            Hest(ii,jj,:)                                                    =
abs(fft(hest(:,ii,jj),NF,1)).*exp(i*reshape(angle(Hest(ii,jj,:)),NF,1));
         end
      end
   end

   if PLOT_PHASE
      figure(2); clf;   %%% Plot the phase
      for ii=1:m
         for jj=1:n
            subplot(m,n,(ii-1)*n+jj);
            title(sprintf('Phase Est',ii,jj));grid;hold on;
            plot(reshape((Phase_true(ii,jj,:)),NF,1)/pi,'r:');
            plot(reshape((Phase_est(ii,H_order(jj),:)),NF,1)/pi,'b-');
            ylabel('Phase in PI');
            Current_axis=axis;
            Current_axis(1)=1;
            Current_axis(2)=NF;
            axis(Current_axis);
         end
      end
   end  %%% End of if PLOT_PHASE

   for ii=1:m
      for jj=1:m
         Phase_est_unwrap(ii,jj,:)=unwrap(Phase_est(ii,jj,:));
         Phase_true_unwrap(ii,jj,:)=unwrap(Phase_true(ii,jj,:));
         Phase_est_1_unwrap(ii,jj,:)=unwrap(Phase_est_1(ii,jj,:));
         Phase_est_2_unwrap(ii,jj,:)=unwrap(Phase_est_2(ii,jj,:));
      end
   end

   Phase_est_unwrap_his(:,:,:,iloop)=Phase_est_unwrap;

   for ii=1:m
      for jj=1:m
         Hest_phase(:,ii,jj)                                                 =
abs(reshape(Hest(ii,jj,:),NF,1)).*exp(i*reshape((mod(Phase_est_unwrap(ii
,jj,:)+pi,2*pi)-pi),NF,1));
      end
   end

   if PLOT_PHASE
      figure(3); clf;   %%% Plot the unwraped phase
      for ii=1:m
         for jj=1:n
            subplot(m,n,(ii-1)*n+jj);
            title(sprintf('Phase Est and True Unwrap',ii,jj));grid;hold
on;

plot(reshape((Phase_est_unwrap(ii,H_order(jj),:)),NF,1)/pi,'b-');
            plot(reshape((Phase_true_unwrap(ii,jj,:)),NF,1)/pi,'r:');
            ylabel('Phase in PI');
```

```
            Current_axis=axis;
            Current_axis(1)=1;
            Current_axis(2)=NF;
            axis(Current_axis);
         end
      end

      figure(4); clf;   %%% Plot the unwraped phase
      for ii=1:m
         for jj=1:n
            subplot(m,n,(ii-1)*n+jj);
            title(sprintf('Phase       Est       1       and       True
Unwrap',ii,jj));grid;hold on;

plot(reshape((Phase_est_1_unwrap(ii,H_order(jj),:)),NF,1)/pi,'b-');
            plot(reshape((Phase_true_unwrap(ii,jj,:)),NF,1)/pi,'r:');
            ylabel('Phase in PI');
            Current_axis=axis;
            Current_axis(1)=1;
            Current_axis(2)=NF;
            axis(Current_axis);
         end
      end

      figure(5); clf;   %%% Plot the unwraped phase
      for ii=1:m
         for jj=1:m
            subplot(m,m,(ii-1)*m+jj);
            title(sprintf('Phase       Est       2       and       True
Unwrap',ii,jj));grid;hold on;

plot(reshape((Phase_est_2_unwrap(ii,H_order(jj),:)),NF,1)/pi,'b-');
            plot(reshape((Phase_true_unwrap(ii,jj,:)),NF,1)/pi,'r:');
            ylabel('Phase in PI');
            Current_axis=axis;
            Current_axis(1)=1;
            Current_axis(2)=NF;
            axis(Current_axis);
         end
      end
   end %%% End of if PLOT_PHASE

   Phase_unwrap_his(:,:,:,iloop)=Phase_est_unwrap;

   if SISTEM_REAL
      Phase_est(:,:,NF/2+2:NF)=-Phase_est(:,:,NF/2:-1:2);
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Estimasi fasa berakhir
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



   hest_L=hest(1:L,:,:);
```

```
   hest_L_his(:,:,:,iloop)=hest_L;

   figure(6);clf
   for ii=1:m
      for jj=1:n
         subplot(m,n,(ii-1)*n+jj);hold on;
         plot(reshape(abs(H(:,ii,jj)),NF,1),'rx');
         plot(reshape(abs(Hest(ii,H_order(jj),:)),NF,1),'b-');
         plot(EIG_INDEX',0,'r*');
         legend('True Mag','Est Mag','Selected Freq');
         title(sprintf('|H(%d,%d)|',ii,jj));
         Current_axis=axis;
         Current_axis(1)=1;
         Current_axis(2)=NF;
         axis(Current_axis);
         grid
      end
   end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
%%  Rekontruksi masukan dimulai
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
   if RECONSTRUCT_INPUT

      if Minimum_Phase_Sistem
         HREC = fft(hest_L(:,:,H_order(1:n)),NF,1);
      else
         HREC = shiftdim(Hest(:,H_order(1:n),:),2);
      end

      epsilon=0.0001;
      Hinv=zeros(NF,n,m);

      for w=1:NF
         H_dummy = shiftdim(HREC(w,:,:),1);
         H_dummy = inv(H_dummy'*H_dummy + epsilon*eye(n,n)) * H_dummy';

         Hinv(w,:,:) = H_dummy;
      end

      %hinv = ifft(Hinv,NF,1);
      hinv = real(ifft(Hinv,NF,1));

      for ii=1:n
         for jj=1:m
            hinv(:,ii,jj)=fftshift(hinv(:,ii,jj));
         end
      end

      srec=zeros(n,N);
      for ii=1:n
         for jj=1:m
```

```
            srec(ii,:) = srec(ii,:) + filter(hinv(:,ii,jj)',1,x(jj,:));
         end
      end

      ALL=zeros(n,n,NF);
      for w=1:NF
         ALL(:,:,w)=reshape(Hinv(w,:,:),n,m)*reshape(H(w,:,1:n),m,n);
      end

      h_all = real(fftshift(ifft(shiftdim(ALL,2),NF,1)));

      %   h_all=real(h_all);

      figure(7);clf
      for ii=1:n
         for jj=1:n
            subplot(2*n,n,(ii-1)*2*n+jj);hold on;grid;
            plot(real(h_all(:,ii,jj)));
            if(ii==1)
               title(sprintf('Real  part  of  the  whole  sistem  impulse
response \n(The whole sistem should be close to identity matriks)'));
            else
               title(sprintf('Real  part  of  the  whole  sistem  impulse
response'));
            end

            subplot(2*n,n,(ii-1)*2*n+n+jj);hold on;grid;
            plot(imag(h_all(:,ii,jj)));
            title(sprintf('Imag   part   of   All   the   sistem   impulse
response'));
         end
      end

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %%% Optimasi kode kumulan
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      TSTART = clock;

      figure(8);clf;
      for ii=1:n
         [f,    c,    s_eq(ii,:)]    =    cum3equalizer(real(srec(ii,:)),
Equalizer_Length);
         for jj=1:n

s_eq_Xcorr_s=xcorr(s_eq(ii,:),s(jj,:),MAX_CORRELATION_LENGTH);
            if MATLAB_VERSION_NUMBER(1) >= '6'
                s_eq_Xcorr_s=fliplr(s_eq_Xcorr_s);
            end

            max_e_Xcorr_s(ii,jj)=max(abs(s_eq_Xcorr_s));
            subplot(n,n,(ii-1)*n+jj);hold on;grid;
            plot(s_eq_Xcorr_s);
            title(sprintf('s_{eq}(%d) Xcorr s(%d)',ii,jj));
         end
```

```
         [max_e_Xcorr_s_dummy e_s_index(ii,1)]=max(max_e_Xcorr_s(ii,:));
      end
      correct(iloop)=~sum(abs(sort(e_s_index)-(1:n).'))

      if ~correct(iloop)
         e_s_index(2)=3-e_s_index(1);
      end

      hest_CUM=zeros(MAX_Minus_ORDER+MAX_Plus_ORDER+1,m,n);

      for ii=1:m
         for jj=1:n
            f=xcorr(s_eq(jj,:),x(ii,:),MAX_CORRELATION_LENGTH)/N;
            if MATLAB_VERSION_NUMBER(1) >= '6'
                f=fliplr(f);
            end


f_xcorr_h=xcorr(f,h(:,ii,e_s_index(jj)),MAX_CORRELATION_LENGTH);
            if MATLAB_VERSION_NUMBER(1) >= '6'
                f_xcorr_h=fliplr(f_xcorr_h);
            end

            [f_xcorr_h_max f_xcorr_h_max_index]=max(abs(f_xcorr_h));

            min_f_index=MAX_CORRELATION_LENGTH-f_xcorr_h_max_index-
MAX_Minus_ORDER+2;
            if min_f_index >= 1

hest_CUM(:,ii,e_s_index(jj))=f(min_f_index+(0:MAX_Minus_ORDER+MAX_Plus_O
RDER)).';
            else
               hest_CUM(1:(1-min_f_index),ii,e_s_index(jj))=0;
               hest_CUM((2-
min_f_index:MAX_Minus_ORDER+MAX_Plus_ORDER+1),ii,e_s_index(jj))=...
                  f(1:MAX_Minus_ORDER+MAX_Plus_ORDER+min_f_index).';
            end
         end
      end


      hest_CUM_his(:,:,:,iloop)=hest_CUM;

      TIME_Optimization = etime(clock,TSTART)


      figure(9);clf;
      for ii=1:m
         for jj=1:n
            subplot(m,n,(ii-1)*n+jj);hold on;

            [h_abs_max h_abs_max_index]=max(abs(h(:,ii,jj)));
            f_abs_max_index=h_abs_max_index+MAX_Minus_ORDER;
```

```
            fh_coeff=sign(h(h_abs_max_index,ii,jj))                    *
sign(hest_CUM(f_abs_max_index,ii,jj))                                  *
sqrt(sum(h(:,ii,jj).^2)/sum(hest_CUM(:,ii,jj).^2));
            plot(hest_CUM(:,ii,jj)*fh_coeff, 'b-*');
            stem(MAX_Minus_ORDER+(1:L),h(:,ii,jj),'r');
            legend('Estimated IR','True IR');
            title('Impulse Response');
            grid;
         end
      end

      drawnow;

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %%% Optimasi kode kumulan berakhir
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      h_all_his(:,:,:,iloop)=h_all;

   end  %% End of if RECONSTRUCT_INPUT

   Hest_his(:,:,:,iloop)=Hest;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
%%  Rekontruksi masukan berakhir
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

   drawnow

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
%%  Pengujian metoda Monte-Carlo berakhir
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
end   % End of Monte_carlo Test iloop


hest_L_mean=mean(hest_L_his,4);
hest_L_std=std(hest_L_his,0,4);

Hest_L_his=fft(hest_L_his,NF,1);
Hest_L_abs_mean=mean(abs(Hest_L_his),4);
Hest_L_abs_std=std(abs(Hest_L_his),0,4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Perhitungan nilai hest_ifft
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for iloop=1:RUN_TIMES
   for ii=1:m
      for jj=1:m
```

```
hest_ifft_his(:,ii,jj,iloop)=real(fftshift(ifft(shiftdim(Hest_his(ii,jj,
:,iloop),2))));
      end
   end
end

for iloop=1:RUN_TIMES
   for ii=1:m
      for jj=1:n

hest_h_corr=xcorr(hest_ifft_his(:,ii,H_order(jj),iloop),h(:,ii,jj),MAX_C
ORRELATION_LENGTH_ifft);

         if MATLAB_VERSION_NUMBER(1) >= '6'
             hest_h_corr=flipud(hest_h_corr);
         end

         [max_xcorr_abs max_xcorr_position]=max(abs(hest_h_corr));
         %%% hest_ifft_Le_his is not in the correct column order, so
H_order is needed.
         h_shift=MAX_CORRELATION_LENGTH_ifft+1-max_xcorr_position;
         hest_ifft_Le=hest_ifft_his(mod(h_shift-
MAX_Minus_ORDER_ifft+(1:Le)-1,NF)+1,ii,H_order(jj),iloop)*...
            sign(hest_h_corr(max_xcorr_position));

hest_ifft_Le=hest_ifft_Le*sqrt(sum(h(:,ii,jj).^2)/sum(hest_ifft_Le.^2));
         hest_ifft_Le_his(:,ii,H_order(jj),iloop)=hest_ifft_Le;
      end
   end
end

hest_ifft_Le_mean = mean(hest_ifft_Le_his,4);
hest_ifft_Le_std  = std(hest_ifft_Le_his,[],4);

figure(10);clf;
   for ii=1:m
      for jj=1:n
         subplot(m,n,(ii-1)*n+jj);grid;hold on;
         title(sprintf('hest ifft(%d,%d)',ii,jj));
         h_long=zeros(Le,1);
         h_long((MAX_Minus_ORDER_ifft+(1:L)))=h(:,ii,jj);

shadow_plot(h_long,hest_ifft_Le_mean(:,ii,H_order(jj)),hest_ifft_Le_std(
:,ii,H_order(jj)));
      end
   end

for ii=1:m
   for jj=1:n
      h_long=zeros(Le,1);
      h_long((MAX_Minus_ORDER_ifft+(1:L)))=h(:,ii,jj);
```

```
NMSE_ifft(ii,jj)=sum(sum((reshape(hest_ifft_Le_his(:,ii,H_order(jj),:),L
e,RUN_TIMES)-...
         h_long*ones(1,RUN_TIMES)).^2)) / (h_long'*h_long) / RUN_TIMES;
   end
end


NMSE_ifft

ONMSE_ifft=sum(sum(NMSE_ifft)) / (m*n)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%%% Perhitungan NMSE pada h_ifft berakhir
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%

if 1

hest_CUM_his_bak=hest_CUM_his;

if Process_hest_CUM_his
   for ii=1:m
      for jj=1:n
         [h_abs_max h_abs_max_index]=max(abs(h(:,ii,jj)));
         f_abs_max_index=h_abs_max_index+MAX_Minus_ORDER;
         for run=1:RUN_TIMES
            fh_coeff=sign(h(h_abs_max_index,ii,jj))                    *
sign(hest_CUM_his(f_abs_max_index,ii,jj,run))                          *
sqrt(sum(h(:,ii,jj).^2)/sum(hest_CUM_his(:,ii,jj,run).^2));
            %fh_coeff=sign(h(h_abs_max_index,ii,jj))                   *
sign(hest_CUM_his(f_abs_max_index,ii,jj,run));

hest_CUM_his(:,ii,jj,run)=hest_CUM_his(:,ii,jj,run)*fh_coeff;
         end
      end
   end
end

hest_CUM_mean=mean(hest_CUM_his,4);
hest_CUM_std=std(hest_CUM_his,0,4);

figure(11);clf;
for ii=1:m
   for jj=1:n
      subplot(m,n,(ii-1)*n+jj);hold on;

plot(reshape(hest_CUM_his(:,ii,jj,:),MAX_Minus_ORDER+MAX_Plus_ORDER+1,RU
N_TIMES));
      stem(MAX_Minus_ORDER+(1:L),h(:,ii,jj),'r');
      title('Impulse Response h_{CUM}');
      grid;
   end
end
```

```
for ii=1:m
   for jj=1:n
      h_long=zeros(MAX_Minus_ORDER+MAX_Plus_ORDER+1,1);
      h_long((MAX_Minus_ORDER+(1:L)))=h(:,ii,jj);


NMSE_CUM(ii,jj)=sum(sum((reshape(hest_CUM_his(:,ii,jj,:),MAX_Minus_ORDER
+MAX_Plus_ORDER+1,RUN_TIMES)-...
         h_long*ones(1,RUN_TIMES)).^2)) / (h_long'*h_long) / RUN_TIMES;
   end
end


NMSE_CUM

ONMSE_CUM=sum(sum(NMSE_CUM)) / (m*n)


end
% clear B_* cx* Cy* Ry* Rx* V* s x noise DING* Hest_L_his W* A A1 S_x*
cum* Ph*
Total_TIME = etime(clock,Program_START)
```