LAMPIRAN A
LISTING PROGRAM

```
clc;
clear;
close all;
clc;

N_1=15; %
N_2=0; %
N_3=15;
sigma_0_2=0.2022; % light = 0.4497
kappa_0=4.4e-11;  % light = 5.9e-8;
f_max=91;
sigma_3=0.1175; % light =0.0101;
m_3=0.4906; % light =0.0875;
rho=0.1118;  % light =0.9856;
f_rho=f_max*0.6366;
theta_rho=0;
K_c=1.735;
f_c=f_max./K_c;
T_s=1.8e-4;
T_sim=3;
PLOT=1;

% [f1,c1,th1]=parameter_Jakes('es_j',N_1,sigma_0_2,f_max,'rand',0);
[f1,c1,th1]=parameter_Jakes('es_j',N_1,sigma_0_2,f_max,'rand',0);
c1=c1/sqrt(2);

f_i_n=f1;
c_i_n=c1;
theta_i_n=th1;
N_i=7;
K=K_c;

figure(1);
subplot(1,2,1);
stem([-f_i_n(N_i:-1:1);f_i_n],1/4*[c_i_n(N_i:-1:1);c_i_n].^2);
title('Estimasi untuk rapat spektral Jakes');
grid;
xlabel('f(Hz)');
ylabel('Rapat spektral daya');
tau_max=N_i/(K*f_max);
tau=linspace(0,tau_max,500);
% r_mm=sigma_0^2*besselj(0,2*pi*f_max*tau);
r_mm=sigma_0_2^2*besselj(0,2*pi*f_max*tau);
r_mm_tilde=acf_mue(f_i_n,c_i_n,tau);
subplot(1,2,2);
plot(tau,r_mm,'r-',tau,r_mm_tilde,'b--');
title('Estimasi untuk autokorelasi dengan Jakes');
```

```
legend('asli','estimasi');
grid;
xlabel('tau (s)');
ylabel('Fungsi autokorelasi');
% batas tambahan saja

N_2_s=ceil(N_2/(2/pi*asin(kappa_0)));
[f2,c2,th2]=parameter_Jakes('es_j',N_2_s,sigma_0_2,f_max,'rand',0);
f2 =f2(1:N_2);
c2 =c2(1:N_2)/sqrt(2);
th2=th2(1:N_2);

% tambahan aja
f_i_n=f2;
c_i_n=c2;
theta_i_n=th2;
N_i=N_2_s;
K=K_c;

figure(2);
subplot(1,2,1);
stem([-f_i_n(N_i:-1:1);f_i_n],1/4*[c_i_n(N_i:-1:1);c_i_n].^2);
title('Estimasi untuk rapat spektral Jakes');
grid;
xlabel('f(Hz)');
ylabel('Rapat spektral daya');
tau_max=N_i/(K*f_max);
tau=linspace(0,tau_max,500);
% r_mm=sigma_0^2*besselj(0,2*pi*f_max*tau);
r_mm=sigma_0_2^2*besselj(0,2*pi*f_max*tau);
r_mm_tilde=acf_mue(f_i_n,c_i_n,tau);
subplot(1,2,2);
plot(tau,r_mm,'r-',tau,r_mm_tilde,'b--');
title('Estimasi untuk autokorelasi dengan Jakes');
legend('asli','estimasi');
grid;
xlabel('tau (s)');
ylabel('Fungsi autokorelasi');
% batas tambahan saja

% [f3,c3,th3]=parameter_Gauss('es_g',N_3,1,f_max,f_c,'rand',0);
[f3,c3,th3]=parameter_Gauss('es_g',N_3,1,f_max,f_c,'rand',0);gaMma=(2*pi*f_c/
sqrt(2*log(2)))^2;
f3(N_3)=sqrt(gaMma*N_3/(2*pi)^2-sum(f3(1:N_3-1).^2));

f_i_n=f3;
c_i_n=c3;
```

```
theta_i_n=th3;
N_i=N_3;
K=K_c;

figure(3);
subplot(1,2,1);
stem([-f_i_n(N_i:-1:1);f_i_n],1/4*[c_i_n(N_i:-1:1);c_i_n].^2);
title('Estimasi untuk rapat spektral Gaussian');
grid;
xlabel('f (Hz)');
ylabel('Rapat spektral daya');
% tau_max=N_i/(K*kappa_c*f_c);
tau_max=N_i/(K*K_c*f_c);
tau=linspace(0,tau_max,500);
r_mm=sigma_0_2*exp(-(pi*f_c/sqrt(log(2))*tau).^2);
r_mm_tilde=acf_mue(f_i_n,c_i_n,tau);
subplot(1,2,2);
plot(tau,r_mm,'r-',tau,r_mm_tilde,'b--');
title('Estimasi untuk autokorelasi dengan Gaussian');
legend('asli','estimasi');
grid;
xlabel('tau (s)');
ylabel('Fungsi autokorelasi');

% batas tambahan saja

N=ceil(T_sim/T_s);
t=(0:N-1)*T_s;

arg=2*pi*f_rho*t+theta_rho;

xi_t=abs(Mu_i_t(c1,f1,th1,T_s,T_sim)+...
        Mu_i_t(c2,f2,th2,T_s,T_sim)+rho*cos(arg)+...
        j*(Mu_i_t(c1,f1,th1-pi/2,T_s,T_sim)-...
        Mu_i_t(c2,f2,th2-pi/2,T_s,T_sim)+rho*sin(arg)));
lambda_t=exp(Mu_i_t(c3,f3,th3,T_s,T_sim)*sigma_3+m_3);

eta_t=xi_t.*lambda_t;

figure(4);
if PLOT==1,
   plot(t,20*log10(eta_t),'b-');
   grid;
   xlabel('t (s)');
   ylabel('20 log eta(t)');
   title('Estimasi Proses Suzuki tipe I');
   legend('heavy shadowing');
```

```
 %    legend('light shadowing');
end



function [f_i_n,c_i_n,theta_i_n]=parameter_Jakes_ku(METHOD,N_i,...
                        sigma_0_2,f_max,PHASE,PLOT)

%-------------------------------------------------------------------
% parameter_Jakes_ku.m ----------------------------------------------
%
% Program untuk menghitung frekuensi-frekuensi Doppler diskrit,
% koefisien Doppler, dan fasa Doppler dengan rapat spektral daya Jakes.
%
% Program m-file yang digunakan : LPNM_opt_Jakes.m, fun_Jakes.m, grad_Jakes.m,
% dan acf_mue.m

%-------------------------------------------------------------------
% [f_i_n,c_i_n,theta_i_n]=parameter_Jakes(METHOD,N_i,sigma_0_2,...
%                                 f_max,PHASE,PLOT)
%-------------------------------------------------------------------
% Penjelasan dari parameter-parameter input :
%
% METHOD:
% |-----------------------------------------|-----------------|
% | Metode untuk menghitung frekuensi |      Input        |
% | Doppler diskrit dan koefisien Doppler |                   |
% |-----------------------------------------|-----------------|
% |-----------------------------------------|-----------------|
% | Method of equal distances (MED)    |     'ed_j'          |
% |-----------------------------------------|-----------------|
% | Mean square error method  (MSEM) |     'ms_j'        |
% |-----------------------------------------|-----------------|
% | Method of equal areas (MEA)          |     'ea_j'          |
% |-----------------------------------------|-----------------|
% | Monte Carlo method (MCM)             |     'mc_j'        |
% |-----------------------------------------|-----------------|
% | Lp-norm method (LPNM)                |     'lp_j'          |
% |-----------------------------------------|-----------------|
% | Method of exact Doppler spread (MEDS)   |     'es_j'       |
% |-----------------------------------------|-----------------|
% | Jakes method (JM)                        |     'jm_j'        |
% |-----------------------------------------|-----------------|
%
% N_i: number of harmonic functions
% sigma_0_2: average power of the real deterministic Gaussian
```

```
%           process mu_i(t)
% f_max: maximum Doppler frequency
%
% PHASE:
% |-----------------------------------------|-----------------|
% | Methods for the computation of the |      Input          |
% | Dopler phases                          |                     |
% |-----------------------------------------|-----------------|
% |-----------------------------------------|-----------------|
% | Random Doppler phases                  |     'rand'          |
% |-----------------------------------------|-----------------|
% | Permuted Doppler phases                |     'perm'          |
% |-----------------------------------------|-----------------|
%
% PLOT: plot of the ACF and the PSD of mu_i(t), if PLOT==1



if nargin<6,
  error('Not enough input parameters')
end

sigma_0=sqrt(sigma_0_2);

% Method of equal distances (MED)
if    METHOD=='ed_j',
      n=(1:N_i)';
      f_i_n=f_max/(2*N_i)*(2*n-1);
      c_i_n=2*sigma_0/sqrt(pi)*(asin(n/N_i)-asin((n-1)/N_i)).^0.5;
      K=1;

% Mean square error method (MSEM)
elseif METHOD=='ms_j',
      n=(1:N_i)';
      f_i_n=f_max/(2*N_i)*(2*n-1);
      Tp=1/(2*f_max/N_i);
      t=linspace(0,Tp,5E3);
      Jo=besselj(0,2*pi*f_max*t);
      c_i_n=zeros(size(f_i_n));
      for k=1:length(f_i_n),
         c_i_n(k)=2*sigma_0*...
               sqrt(1/Tp*( trapz( t,Jo.*...
               cos(2*pi*f_i_n(k)*t )) ));
      end
      K=1;

% Method of equal areas (MEA)
```

```
elseif METHOD=='ea_j'
      n=(1:N_i)';
      f_i_n=f_max*sin(pi*n/(2*N_i));
      c_i_n=sigma_0*sqrt(2/N_i)*ones(size(n));
      K=1;

% Monte Carlo method (MCM)
elseif METHOD=='mc_j'
      n=rand(N_i,1);
      f_i_n=f_max*sin(pi*n/2);
      c_i_n=sigma_0*sqrt(2/N_i)*ones(size(n));

      K=1;

% Lp-norm method (LPNM)
elseif METHOD=='lp_j',
      if   exist('fminu')~=2
          disp([' =====> This method requires ',...
               'the Optimization Toolbox !!'])
          return
      else
          N=1E3;
          p=2;   % Norm
          s_o=1;
          [f_i_n,c_i_n]=LPNM_opt_Jakes(N,f_max,sigma_0,p,N_i,s_o);
          K=1;
      end

% Method of exact Doppler spread (MEDS)
elseif METHOD=='es_j',
      n=(1:N_i)';
      f_i_n=f_max*sin(pi/(2*N_i)*(n-1/2));
      c_i_n=sigma_0*sqrt(2/(N_i))*ones(size(f_i_n));
      K=1;

% Jakes method (JM)
elseif METHOD=='jm_j',
      n=1:N_i-1;
      f_i_n=f_max*[[cos(pi*n/(2*(N_i-1/2))),1]',...
                [cos(pi*n/(2*(N_i-1/2))),1]'];
      c_i_n=2*sigma_0/sqrt(N_i-1/2)*[[sin(pi*n/(N_i-1)),1/2]',...
                               [cos(pi*n/(N_i-1)),1/2]'];
      K=1;
      theta_i_n=zeros(size(f_i_n));
      PHASE='none';

else
```

```
    error('Method is unknown')
end

% Computation of the Doppler phases:
if   PHASE=='rand',
    theta_i_n=rand(N_i,1)*2*pi;

elseif PHASE=='perm',
    n=(1:N_i)';
    Z=rand(size(n));
    [dummy,I]=sort(Z);
    theta_i_n=2*pi*n(I)/(N_i+1);
end;

if PLOT==1,
  if  METHOD=='jm_j'
    subplot(2,3,1)
    stem([-f_i_n(N_i:-1:1,1);f_i_n(:,1)],...
        1/4*[c_i_n(N_i:-1:1,1);c_i_n(:,1)].^2)
    title('i=1')
    xlabel('f (Hz)')
    ylabel('PSD')
    subplot(2,3,2)
    stem([-f_i_n(N_i:-1:1,2);f_i_n(:,2)],...
        1/4*[c_i_n(N_i:-1:1,2);c_i_n(:,2)].^2)
    title('i=2')
    xlabel('f (Hz)')
    ylabel('PSD')
    tau_max=N_i/(K*f_max);
    tau=linspace(0,tau_max,500);
    r_mm=sigma_0^2*besselj(0,2*pi*f_max*tau);

    r_mm_tilde1=acf_mue(f_i_n(:,1),c_i_n(:,1),tau);
    subplot(2,3,4)
    plot(tau,r_mm,'r-',tau,r_mm_tilde1,'g--')
    title('i=1')
    xlabel('tau (s)')
    ylabel('ACF')
    r_mm_tilde2=acf_mue(f_i_n(:,2),c_i_n(:,2),tau);
    subplot(2,3,5)
    plot(tau,r_mm,'r-',tau,r_mm_tilde2,'g--')
    title('i=2')
    xlabel('tau (s)')
    ylabel('ACF')
    subplot(2,3,3)
    stem([-f_i_n(N_i:-1:1,1);f_i_n(:,1)],...
        1/4*[c_i_n(N_i:-1:1,1);c_i_n(:,1)].^2+...
```

```
        1/4*[c_i_n(N_i:-1:1,2);c_i_n(:,2)].^2)
    title('i=1,2')
    xlabel('f (Hz)')
    ylabel('PSD')
    subplot(2,3,6)
    plot(tau,2*r_mm,'r-',tau,r_mm_tilde1+r_mm_tilde2,'g--')
    title('i=1,2')
    xlabel('tau (s)')
    ylabel('ACF')
  else
    subplot(1,2,1)
    stem([-f_i_n(N_i:-1:1);f_i_n],...
        1/4*[c_i_n(N_i:-1:1);c_i_n].^2)
    xlabel('f/Hz')
    ylabel('LDS')
    tau_max=N_i/(K*f_max);
    tau=linspace(0,tau_max,500);
    r_mm=sigma_0^2*besselj(0,2*pi*f_max*tau);
    r_mm_tilde=acf_mue(f_i_n,c_i_n,tau);
    subplot(1,2,2)
    plot(tau,r_mm,'r-',tau,r_mm_tilde,'g--')
    xlabel('tau (s)')
    ylabel('ACF')
  end
end


%-------------------------------------------------------------------
% parameter_Gauss.m ------------------------------------------------
%
% Program for the computation of the discrete Doppler frequencies,
% Doppler coefficients, and Doppler phases by using the Gaussian
% power spectral density.
%
% Used m-files: LPNM_opt_Gauss.m, fun_Gauss.m,
%                    grad_Gauss.m, acf_mue.m
%-------------------------------------------------------------------
% [f_i_n,c_i_n,theta_i_n]=parameter_Gauss(METHOD,N_i,sigma_0_2,...
%                                f_max,f_c,PHASE,PLOT)
%-------------------------------------------------------------------
% Explanation of the input parameters:
%
% METHOD:
% |-----------------------------------------|-----------------|
% | Methods for the computation of the discrete  |     Input       |
% | Doppler frequencies and Doppler coefficients |                 |
% |-----------------------------------------|-----------------|
```

```
% |-----------------------------------------|-----------------|
% | Method of equal distances (MED)    |     'ed_g'         |
% |-----------------------------------------|-----------------|
% | Mean square error method  (MSEM) |     'ms_g'        |
% |-----------------------------------------|-----------------|
% | Method of equal areas (MEA)          |     'ea_g'         |
% |-----------------------------------------|-----------------|
% | Monte Carlo method (MCM)             |     'mc_g'        |
% |-----------------------------------------|-----------------|
% | Lp-norm method (LPNM)                |     'lp_g'         |
% |-----------------------------------------|-----------------|
% | Method of exact Doppler spread(MEDS) |     'es_g'       |
% |-----------------------------------------|-----------------|
%
% N_i: number of harmonic functions
% sigma_0_2: average power of the real deterministic Gaussian
%           process mu_i(t)
% f_max: maximum Doppler frequency
% f_c: 3-dB-cutoff frequency
%
% PHASE:
% |-----------------------------------------|-----------------|
% | Methods for the computation of the Doppler   |      Input       |
% | phases                                       |                  |
% |-----------------------------------------|-----------------|
% |-----------------------------------------|-----------------|
% | Random Doppler phases                |     'rand'         |
% |-----------------------------------------|-----------------|
% | Permuted Doppler phases              |     'perm'         |
% |-----------------------------------------|-----------------|
%
% PLOT: plot of the ACF and the PSD of mu_i(t), if PLOT==1

function [f_i_n,c_i_n,theta_i_n]=parameter_Gauss(METHOD,N_i,...
                          sigma_0_2,f_max,f_c,PHASE,PLOT)

if nargin<7,
  error('Not enough input parameters')
end

sigma_0=sqrt(sigma_0_2);
kappa_c=f_max/f_c;

% Method of equal distances (MED)
if     METHOD=='ed_g',
       n=(1:N_i)';
       f_i_n=kappa_c*f_c/(2*N_i)*(2*n-1);
```

```
    c_i_n=sigma_0*sqrt(2)*sqrt(erf(n*kappa_c*...
        sqrt(log(2))/N_i)-erf((n-1)*kappa_c*...
        sqrt(log(2))/N_i) );
    K=1;

% Mean square error method (MSEM)
elseif METHOD=='ms_g',
    n=(1:N_i)';
    f_i_n=kappa_c*f_c/(2*N_i)*(2*n-1);
    tau_max=N_i/(2*kappa_c*f_c);
    N=1E3;
    tau=linspace(0,tau_max,N);
    f1=exp(-(pi*f_c*tau).^2/log(2));
    c_i_n=zeros(size(f_i_n));
    for k=1:length(c_i_n),
       c_i_n(k)=2*sigma_0*sqrt(trapz(tau,f1.*...
              cos(2*pi*f_i_n(k)*tau))/tau_max);
    end
    K=1;

% Method of equal areas (MEA)
elseif METHOD=='ea_g'
    n=(1:N_i)';
    c_i_n=sigma_0*sqrt(2/N_i)*ones(size(n));
    f_i_n=f_c/sqrt(log(2))*erfinv(n/N_i);
    f_i_n(N_i)=f_c/sqrt(log(2))*erfinv(0.9999999);
    K=1;

% Monte Carlo method (MCM)
elseif METHOD=='mc_g'
    n=rand(N_i,1);
    f_i_n=f_c/sqrt(log(2))*erfinv(n);
    c_i_n=sigma_0*sqrt(2/N_i)*ones(size(n));
    K=1;

% Lp-norm method (LPNM)
elseif METHOD=='lp_g',

    if   exist('fminu')~=2
        disp([' =====> This method requires ',...
            'the Optimization Toolbox !!'])
        return
    else
        N=1e3;
        p=2;
        [f_i_n,c_i_n]=LPNM_opt_Gauss(N,f_max,f_c,...
                sigma_0_2,p,N_i,PLOT);
```

```
        K=2;
    end

% Method of exact Doppler spread (MEDS)
elseif METHOD=='es_g',
    n=(1:N_i)';
    c_i_n=sigma_0*sqrt(2/N_i)*ones(size(n));
    f_i_n=f_c/sqrt(log(2))*erfinv((2*n-1)/(2*N_i));
    K=1;
else
    error([setstr(10),'Method is unknown'])
end

% Computation of the Doppler phases:
if   PHASE=='rand',
    theta_i_n=rand(N_i,1)*2*pi;
elseif PHASE=='perm',
    n=(1:N_i)';
    Z=rand(size(n));
    [dummy,I]=sort(Z);
    theta_i_n=2*pi*n(I)/(N_i+1);
end

if PLOT==1,
  subplot(1,2,1)
  stem([-f_i_n(N_i:-1:1);f_i_n],...
      1/4*[c_i_n(N_i:-1:1);c_i_n].^2)
  xlabel('f (Hz)')
  ylabel('PSD')
  tau_max=N_i/(K*kappa_c*f_c);
  tau=linspace(0,tau_max,500);
  r_mm=sigma_0_2*exp(-(pi*f_c/sqrt(log(2))*tau).^2);
  r_mm_tilde=acf_mue(f_i_n,c_i_n,tau);
  subplot(1,2,2)
  plot(tau,r_mm,'r-',tau,r_mm_tilde,'g--')
  xlabel('tau (s)')
  ylabel('ACF')
end
```

```
%-----------------------------------------------------------------
% Suzuki_Type_I.m -------------------------------------------------
%
% Program for the simulation of deterministic extended Suzuki
% processes of Type I (see Fig. 6.9).
%
% Used m-files: parameter_Jakes.m, parameter_Gauss.m, Mu_i_t.m
%-----------------------------------------------------------------
% eta_t=Suzuki_Type_I(N_1,N_2,N_3,sigma_0_2,kappa_0,f_max,sigma_3,...
%                  m_3,rho,f_rho,theta_rho,f_c,T_s,T_sim,PLOT)
%-----------------------------------------------------------------
% Explanation of the input parameters:
%
% N_1, N_2, N_3: number of harmonic functions of the real deter-
%              ministic Gaussian processes nu_1(t), nu_2(t),
%              and nu_3(t), respectively
% sigma_0_2: average power of the real deterministic Gaussian
%           processes mu_1(t) and mu_2(t)
% kappa_0: frequency ratio f_min/f_max (0<=kappa_0<=1)
% f_max: maximum Doppler frequency
% sigma_3: square root of the average power of the real deterministic
%          Gaussian process nu_3(t)
% m_3: average value of the third real deterministic Gaussian
%      process mu_3(t)
% rho: amplitude of the LOS component m(t)
% f_rho: Doppler frequency of the LOS component m(t)
% theta_rho: phase of the LOS component m(t)
% f_c: 3-dB-cut-off frequency
% T_s: sampling interval
% T_sim: duration of the simulation
% PLOT: plot of the deterministic extended Suzuki process eta(t) of
%       Type I, if PLOT==1

function eta_t=Suzuki_Type_I(N_1,N_2,N_3,sigma_0_2,kappa_0,f_max,...
            sigma_3,m_3,rho,f_rho,theta_rho,f_c,T_s,T_sim,PLOT)

if nargin==14,
  PLOT=0;
end

[f1,c1,th1]=parameter_Jakes('es_j',N_1,sigma_0_2,f_max,'rand',0);
c1=c1/sqrt(2);

N_2_s=ceil(N_2/(2/pi*asin(kappa_0)));
[f2,c2,th2]=parameter_Jakes('es_j',N_2_s,sigma_0_2,f_max,'rand',0);
f2 =f2(1:N_2);
```

```
c2 =c2(1:N_2)/sqrt(2);
th2=th2(1:N_2);

[f3,c3,th3]=parameter_Gauss('es_g',N_3,1,f_max,f_c,'rand',0);
gaMma=(2*pi*f_c/sqrt(2*log(2)))^2;
f3(N_3)=sqrt(gaMma*N_3/(2*pi)^2-sum(f3(1:N_3-1).^2));

N=ceil(T_sim/T_s);
t=(0:N-1)*T_s;

arg=2*pi*f_rho*t+theta_rho;

xi_t=abs(Mu_i_t(c1,f1,th1,T_s,T_sim)+...
        Mu_i_t(c2,f2,th2,T_s,T_sim)+rho*cos(arg)+...
        j*(Mu_i_t(c1,f1,th1-pi/2,T_s,T_sim)-...
        Mu_i_t(c2,f2,th2-pi/2,T_s,T_sim)+rho*sin(arg)));
lambda_t=exp(Mu_i_t(c3,f3,th3,T_s,T_sim)*sigma_3+m_3);

eta_t=xi_t.*lambda_t;

if PLOT==1,
   plot(t,20*log10(eta_t),'b-')
   xlabel('t (s)')
   ylabel('20 log eta(t)')
end
```