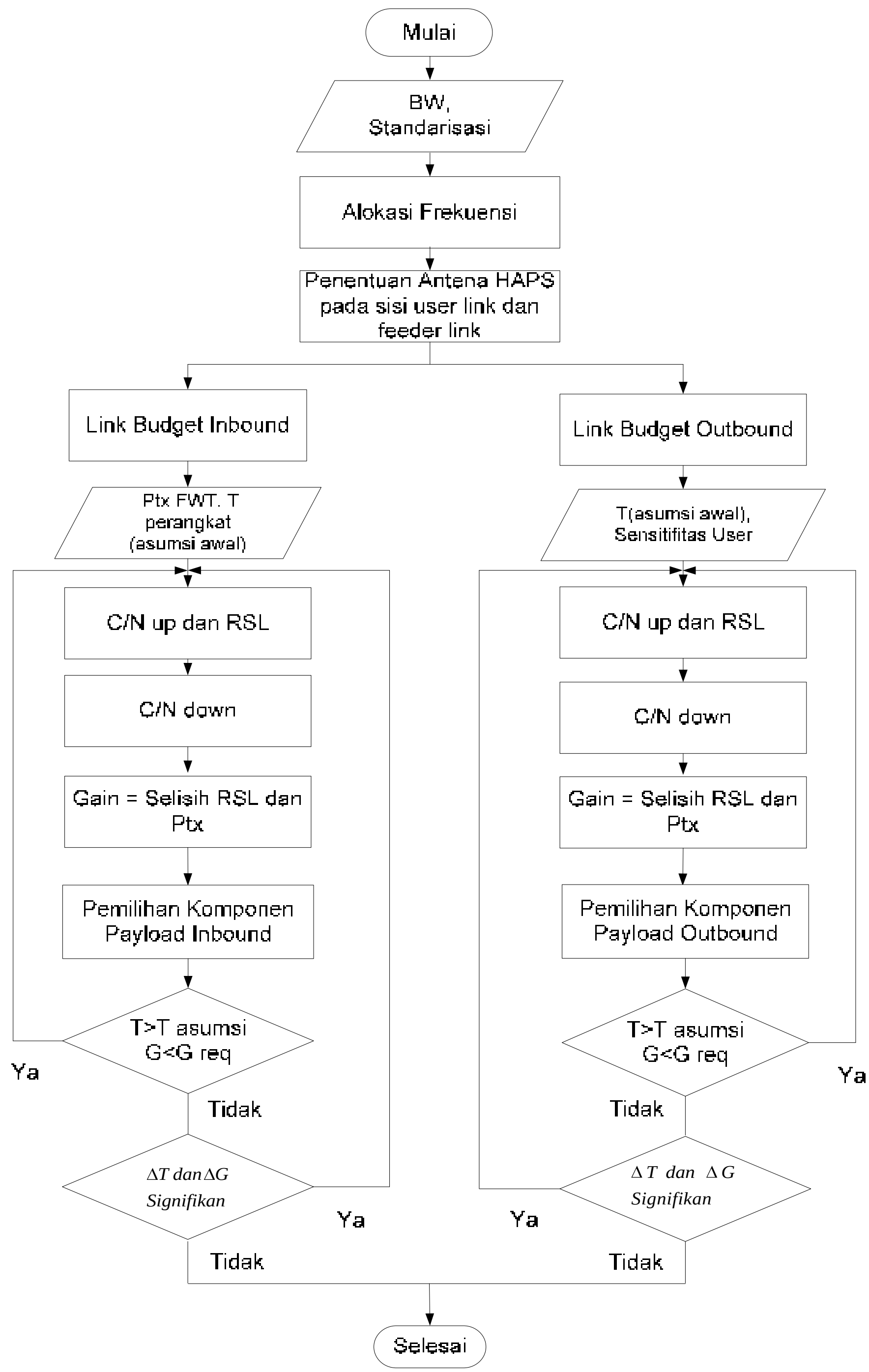
LAMPIRAN A

FLOWCHART
PERANCANGAN PAYLOAD


Mulai
BW,
Standarisasi
Alokasi Frekuensi
Penentuan Antena HAPS pada sisi user link dan feeder link
Link Budget Inbound
Link Budget Outbound
Ptx FWT. T perangkat (asumsi awal)
T(asumsi awal), Sensitifitas User
C/N up dan RSL
C/N down
Gain = Selisih RSL dan Ptx
Pemilihan Komponen Payload Inbound
Pemilihan Komponen Payload Outbound
T>T asumsi G<G req
Ya
Tidak
ΔT dan ΔG Signifikan
Ya
Tidak
Selesai

LAMPIRAN B

PENURUNAN FILTER SPEKTRAL

Penurunan Filter Spektral

Dengan memperhitungkan total sinyal yang diterima antena *Rx*, dapat diasumsikan bahwa antena menerima sinyal tak hingga banyak. Sehingga untuk sudut datang yang sangat kecil $(d\alpha)$, daya yang diterima antena pada sudut tersebut adalah $p(\alpha)d(\alpha)$. Dengan asumsi gain azimuth antena pada sudut datang tersebut $G(\alpha)$, maka total daya yang diterima antena adalah :

$$P_r = \int_0^{2\pi} AG(\alpha)p(\alpha)d(\alpha) \quad \text{(B.1)}$$

$AG(\alpha)p(\alpha)d(\alpha)$ merupakan variasi daya terima terhadap sudut datang. Sehingga daya terima antena *Rx* merupakan penjumlahan variasi daya terima untuk tiap sudut pada *doppler spread* yang dinyatakan dengan :

$$f(\alpha) = \frac{v}{\lambda}\cos(\alpha) + f_c \quad \text{(B.2)}$$

dengan $f(\alpha) = f(-\alpha)$

Jika *S(f)* merupakan spektral daya terima, maka variasi diferensialnya dinyatakan dengan :

$$S(f)|d(f)| \quad \text{(B.3)}$$

Sehingga,

$$S(f)\,|\,df\,| = A[\,p(\alpha)G(\alpha) + p(-\alpha)G(-\alpha)]\,|\,d\alpha\,| \quad \text{(B.4)}$$

dengan mendeferensiasikan persamaan (B.4) :

$$\frac{df(\alpha)}{d(\alpha)} = f_d \cos\alpha + f_c \quad \text{(B.5)}$$

$$df = \left(f_d \cos\alpha + f_c\right)d\alpha \quad \text{(B.6)}$$

$$df = -f_d \sin\alpha \, d\alpha$$
$$= |d\alpha| |-\sin\alpha| f_d \qquad \text{(B.7)}$$

dari persamaan (B.4) dapat diperoleh α yaitu :

$$\frac{f - f_c}{f_d} = \cos\alpha \qquad \text{(B.8)}$$

$$\alpha = \cos^{-1}\left(\frac{f - f_c}{f_d}\right) \qquad \text{(B.9)}$$

dengan menggunakan sifat trigonometri $sin^2 a + cos^2 a = 1$, maka :

$$\sin\alpha = \sqrt{1 - \left(\frac{f - f_c}{f_d}\right)^2} \qquad \text{(B.10)}$$

dengan substitusi diperoleh spektral daya :

$$S(f) = \frac{A[p(\alpha)G(\alpha) + p(-\alpha)G(-\alpha)]}{f_d \sqrt{1 - \left(\frac{f - f_c}{f_m}\right)^2}} \qquad \text{(B.11)}$$

$$S(f) = \frac{7{,}94}{\pi f_m \sqrt{1 - \left(\frac{f - f_c}{f_m}\right)^2}} \qquad \text{(B.12)}$$

LAMPIRAN C

LISTING PROGRAM

C.1 Link Budget

C.1.1 Program Utama Link Budget

```
%function output      = budget(T, bw, eta, frek, diameter, alpha, Gtx, ht, hr, pol, Llain, Tsb, frek28, diameter28, alpha28, ht28, pol28, rsl, T31, frek31, pol31, Psb, Tfwt, frek2, pol2, rsl_fwt, Ptx, T_bumi)


clear all

clc


% Power

Ptx         = input('Ptx user (dBw)                          = ');

rsl         = input('Sensitivitas Gateway (dBw)              = ');

Psb         = input('Power Transmit SB (dBw)                 = ');

rsl_fwt     = input('Sensitivitas fwt (dBw)                  = ');

Gtx         = input('Gain Antena FWT (dBi)                   = ');


% Suhu

T           = input('Suhu sistem HAPS pada Transponder S-Band (K) = ');

Tfwt        = input('Suhu sistem FWT (K)                     = ');

Tsb         = input('Suhu sistem  Gateway (K)                = ');

T31         = input('Suhu sistem HAPS Transponder Ka-Band (K)  = ');

T_bumi      = input('Suhu Permukaan Bumi (C)                 = ');


% Komponen Noise

bw          = input('Bandwidth (MHz)                         = ');

eta         = input('efisiensi antena HAPS S-Band (persen)   = ');
```

```
% Frekuensi (GHz)
frek          = input('f reverse user link (GHz)                  = ');
frek28        = input('f forward feeder link(GHz)                 = ');
frek31        = input('f reverse feeder link (GHz)                = ');
frek2         = input('f forward userlink (GHz)                   = ');

% Diameter Antena
diameter      = input('Diameter antena HAPS S-Band (m)            = ');
diameter28_sb = input('Diameter antena Gateway Ka-Band (m)        = ');
diameter28_haps = input('Diameter antena HAPS Ka-Band (m)         = ');

% Sudut Elevasi
alpha         = input('elevasi user (derajat)                     = ');
alpha28       =input('elevasi  Gateway (derajat)                  = ');

% Ketinggian Antena
ht            = input('Tinggi antena FWT (m)                      = ');
hr            = input('Ketinggian HAPS (m)                        = ');
ht28          = input('Tinggi antena Gateway (m)                  = ');

% Polarisasi yang digunakan
pol           = input('polarisasi user link horizontal(1) vertikal(2)  = ');
pol28         = input('polarisasi downfeederlink horizontal(1) vertikal(2)
= ');
pol31         = input('polarisasi upfeederlink horizontal(1) vertikal(2)= ');
pol2          = input('polarisasi downuserlink horizontal(1) vertikal(2)
= ');
```

```
% Redaman Konektor
Llain          = input('Redaman konektor (dB)                    = ');

ht31 = ht28
hr28 = hr
hr31 = hr28
alpha31 = alpha28
diameter2 = diameter
alpha2 = alpha
ht2 = ht;
hr2 = hr
hr28 = hr

T1     = 300+T
Tsb1   = Tant(2,2,frek28,diameter28_sb,ht28,alpha28,pol28,T_bumi)+Tsb
T311   = 300+T31
Tfwt   = Tfwt+298

kdb    = -(10*log10(1.38e-23))
Bdb    = 10*log10(bw*1e6)
Tdb1   = 10*log10(T1);
Tdb2   = 10*log10(Tsb1);
Tdb3   = 10*log10(Tfwt);
Tdb4   = 10*log10(T311);
```

```
%   Perhitungan (C/N)up reverse user link
Grx = gain(eta,frek,diameter)

    %(1)
    alpha1= deg2rad(alpha);
    if alpha= = 90
       d1=1e3
    else
       d1=1e3./(tan(alpha1))
    end

PL_2ray1      = 40*log10(d1)-(Gtx+Grx+(20*log10(ht))+(20*log10(hr)))
Ltotal1       =
hujan(ht,frek,alpha,pol)+freespace(ht,alpha,frek)+Llain+PL_2ray1
rsl1          = Ptx+Gtx+Grx-Ltotal1
CtoN1         = rsl1+kdb-Tdb1-Bdb

%   Perhitungan (C/N)down forward feeder link
Grx2   = gain(eta,frek28,diameter28_sb)
Gtx2   = gain(eta,frek28,diameter28_haps)
lamda28= 3e8/(frek28*1e9)

    %(2)
    alpha2=deg2rad(alpha28);
    if alpha28==90
       d2=1e3
       %d2=((4*ht28*hr28)/lamda28)
    else
```

```
        d2=1e3./(tan(alpha2))
    end

    %PL_2ray2=40*log10(d2)-(Gtx2+Grx2+(20*log10(ht28))+(20*log10(hr28)))
Ltotal2=hujan(ht28,frek28,alpha28,pol28)+freespace(ht28,alpha28,frek28)+Llain
Ptx_HAPS=rsl-Gtx2-Grx2+Ltotal2
CtoN2=rsl+kdb-Tdb2-Bdb

%   Menghitung Gain Payload HAPS
G_payload_inbound=Ptx_HAPS-rsl1

%   Perhitungan (C/N)up reverse feeder link
Grx3=gain(eta,frek31,diameter28_haps)
Gtx3=gain(eta,frek31,diameter28_sb)
lamda31=3e8/(frek31*1e9)

    %(3)
    alpha3=deg2rad(alpha31);
    if alpha31==90
        d3=1e3
        %d3=((4*ht31*hr31)/lamda31)
    else
        d3=1e3./(tan(alpha3))
    end

Ltotal3=hujan(ht31,frek31,alpha31,pol31)+freespace(ht31,alpha31,frek31)+Llain
```

```
rsl_HAPS=Psb+Gtx3+Grx3-Ltotal3

CtoN3=rsl_HAPS+kdb-Tdb4-Bdb


%   Perhitungan (C/N)down forward user link

Gtx4=gain(eta,frek2,diameter2)

Grx4=Gtx


    d4=d1;


PL_2ray4=40*log10(d1)-(Gtx+Grx+(20*log10(ht))+(20*log10(hr)))

Ltotal4=hujan(ht2,frek2,alpha2,pol2)+freespace(ht,alpha,frek2)+Llain+PL_2ray4

Ptx_HAPS2=rsl_fwt-Gtx4-Grx4+Ltotal4

CtoN4=rsl_fwt+kdb-Tdb3-Bdb


%   Menghitung Gain Payload HAPS

G_payload_outbound=Ptx_HAPS2-rsl_HAPS


CtoN1_lin=10^(CtoN1/10);

CtoN2_lin=10^(CtoN2/10);

CtoN3_lin=10^(CtoN3/10);

CtoN4_lin=10^(CtoN4/10);


CtoN_total=10*log10 (1/((1/CtoN1_lin) + (1/CtoN2_lin) + (1/CtoN3_lin) + (1/CtoN4_lin)))
```

C.1.2 Sub Program Perhitungan Free Space Loss

```
function Lfs = freespace(ht,alpha,frek)

ht1=ht/1000;
R1=6370;            %dalam km
alpha1=deg2rad(alpha)

if alpha1 >= 0.0873
   Leff=(1-ht1)/(sin(alpha1)) % dalam km
else
   q1=(sin(alpha1)).^2;
   q2=2*(1-ht1)/R1;
   q3=sqrt(q1+q2);
   q4=sin(alpha1);
   q5=q3+q4;%sisi bawah
   q6=2*ht1;
   Leff=q5+q6;
end

Lfs=92.4+(20*log10(Leff))+(20*log10(frek))
```

C.1.3 Sub Program Perhitungan Redaman pada Two Ray Model

```
function PL_2ray=tworay(frek,ht,hr,alpha)

Gtx=9;
Grx=28;
```

```
%Perhitungan pathloss 2 ray model

%************************************************************
******

f_bmb=frek*1e9;

c=3e8;

lamda=c/f_bmb

alpha1=deg2rad(alpha);

if alpha==90

    d=1e3;

else

    d=1e3./(tan(alpha1))

end

PL_2ray=(40*log10(d))-(Gtx+Grx+(20*log10(ht))+(20*log10(hr)));

%PL_2ray=40*log10(d)-
(gain(eta,frek,diameter)+gain(eta,frek,diameter)+(20*log10(ht))+(20*log1
0(hr)));

figure;

plot(alpha,PL_2ray)
```

C.1.4 Sub Program Perhitungan Redaman Hujan

```
function Redaman_hujan=hujan(ht,frek,alpha,pol)

%posisi lintang receiver tidak diperhitungkan karena ketinggian HAPS =
1km

%maka heff tidak =hR tetapi 1 km

%HAPS pada daerah hujan
```

```
%Asumsi tower antena di atas permukaan laut

%*******************************************************
******

%

%perhitungan redaman hujan

%

%*******************************************************
******


ht1=ht/1000;

R1=6370;                    %dalam km

alpha1=deg2rad(alpha);


if alpha1>=0.0873

   Leff=(1-ht1)./(sin(alpha1)) ;% dalam km

else

   q1=(sin(alpha1)).^2;

   q2=2*(1-ht1)/R1;

   q3=sqrt(q1+q2);

   q4=sin(alpha1);

   q5=q3+q4;%sisi bawah

   q6=2*ht1;

   Leff=q5+q6;

end


%R1 merupakan jari-jari bumi

%ht1 dalam km

%Leff dalam km
```

```
%Lg dalam km

Lg=Leff.*cos(alpha);

%menghitung reduksi outage hujan
%R merupakan curah hujan dalam mm/hr
%pada derah P nilai R = 145

R=145;
penyebut=(0.045*Lg)+1;
gamma=1./penyebut;

%Penentuan  nilai a dan b pada frekuensi 2 dan 31 GHz
%**********************************************************
******
%ketik 1 untuk polarisasi horizontal ----------->    pol<=3
%ketik 2 untuk polarisasi vertikal ------------->    pol<=5
%**********************************************************
******

if frek==2
    ah2=0.000154;
    bh2=0.936;
    av2=0.000138;
    bv2=0.923;

    ah4=0.00065;
    bh4=1.121;
```

```
    av4=0.000591;
    bv4=1.075;
    if pol==1
        a=0.000154;
        b=0.936;
        %disp('Polarisasi = horizontal')
    elseif pol==2
        a=0.000138;
        b=0.923;
        %b=0.923;
        %disp('Polarisasi = vertikal')
    else
        a=NaN;
        b=NaN;
        disp('polarisasi tidak dikenali, Hasil tidak akurat')
        disp('masukkan 1 untuk polarisasi horizontal       ')
        disp('masukkan 2 untuk polarisasi vertikal         ')
disp('*******************************************************
*')
    end

elseif frek = = 1.9
    ah1=0.0000387;
    bh1=0.912;
    av1=0.0000352;
    bv=0.88;
```

```
    ah2=0.000154;
    bh2=0.936;
    av2=0.000138;
    bv2=0.923;
    if pol = = 1
        a=((1.9-1)/(2-1))*(ah2-ah1)+ah1;
        b=((1.9-1)/(2-1))*(bh2-bh1)+bh1;
        %disp('Polarisasi = horizontal')
        %disp('Polarisasi = horizontal')
    elseif pol==2
        a=((1.9-1)/(2-1))*(av2-av1)+av1;
        %a=0.2022;
        %b=1.0126;
        b=((1.9-1)/(2-1))*(bv2-bv1)+bv1;
        %disp('Polarisasi = horizontal')
    end

elseif frek==27.5
    ah30=0.187;
    bh30=1.021;
    av30=0.167;
    bv30=1;

    ah25=0.124;
    bh25=1.061;
    av25=0.113;
    bv25=1.03;
    if pol==1
```

```
        a=((27.5-25)/(30-25))*(ah30-ah25)+ah25;
        b=((27.5-25)/(30-25))*(bh30-bh25)+bh25;
        %disp('Polarisasi = horizontal')
        %disp('Polarisasi = horizontal')
    elseif pol==2
        a=((27.5-25)/(30-25))*(av30-av25)+av25;
        b=((27.5-25)/(30-25))*(bv30-bv25)+bv25;
        %disp('Polarisasi = horizontal')
    end

elseif frek==28
    ah30=0.187;
    bh30=1.021;
    av30=0.167;
    bv30=1;

    ah25=0.124;
    bh25=1.061;
    av25=0.113;
    bv25=1.03;
    if pol==1
        a=((28-25)/(30-25))*(ah30-ah25)+ah25;
        b=((28-25)/(30-25))*(bh30-bh25)+bh25;
        %disp('Polarisasi = horizontal')
        %disp('Polarisasi = horizontal')
    elseif pol==2
        a=((28-25)/(30-25))*(av30-av25)+av25;
        b=((28-25)/(30-25))*(bv30-bv25)+bv25;
```

```
        %disp('Polarisasi = horizontal')
    else
        a=NaN;
        b=NaN;
        disp('polarisasi tidak dikenali, Hasil tidak akurat')
        disp('masukkan 1 untuk polarisasi horizontal       ')
        disp('masukkan 2 untuk polarisasi vertikal         ')

disp('*****************************************************
*')
    end


elseif frek==31
    ah30=0.187;
    bh30=1.021;
    av30=0.167;
    bv30=1;


    ah35=0.263;
    bh35=0.979;
    av35=0.233;
    bv35=0.963;


    if pol = = 1
        a=((31-30)/(35-30))*(ah35-ah30)+ah30;
        b=((31-30)/(35-30))*(bh35-bh30)+bh30;
        %disp('Polarisasi = horizontal')
    elseif pol = = 2
```

```
        a=((31-30)/(35-30))*(av35-av30)+av30 ;
        b=((31-30)/(35-30))*(bv35-bv30)+bv30;
        %disp('Polarisasi = vertikal')
    else
        a=NaN;
        b=NaN;
        disp('polarisasi tidak dikenali, Hasil tidak akurat')
        disp('masukkan 1 untuk polarisasi horizontal       ')
        disp('masukkan 2 untuk polarisasi vertikal         ')

disp('*****************************************************
*')
    end
elseif frek = =1.8
    ah1=0.0000384;
    bh1=0.912;
    av1=0.0000352;
    bv1=0.88;

    ah2=0.000154;
    bh2=0.936;
    av2=0.000138;
    bv2=0.923;

    if pol = =1
        a=((1.8-1)/(2-1))*(ah2-ah1)+ah1;
        %a=0.2022;
        %b=1.0126;
```

```
        b=((1.8-1)/(2-1))*(bh2-bh1)+bh1;

        %disp('Polarisasi = horizontal')

    elseif pol = =2

        a=((1.8-1)/(2-1))*(av2-av1)+av1;

        b=((1.8-1)/(2-1))*(bv2-bv1)+bv1;

        %disp('Polarisasi = vertikal')

    else

        a=NaN;

        b=NaN;

        disp('polarisasi tidak dikenali, Hasil tidak akurat')

        disp('masukkan 1 untuk polarisasi horizontal       ')

        disp('masukkan 2 untuk polarisasi vertikal         ')

disp('*******************************************************
*’)

    end

else

    disp('false')

end


%disp('******************************************************
****’);

%disp('Besar nilai redaman dalam dB yaitu : ');

%disp('******************************************************
****’);


er=a*(R^b);


Redaman_hujan=er.*gamma.*Leff;
```

C.1.5 Sub Program Perhitungan Gain Antena

```
function Gain_parabola=gain(eta,frek,diameter)

% eta = efisiensi antena parabola (persen)
% frek= frekuensi resonansi antena parabola (GHz)
% diamter = diamter antena (m)

eta_num=eta/100;
Gain_parabola=20.4+10*log10(eta_num)+20*log10(frek)+20*log10(diam
eter);
```

C.1.6 Sub Program Perhitungan Thermal Noise Matahari

```
function Tmatahari=matahari(cuaca,frek,diameter);

%frek dalam GHz
%diamter dalam m
%HPBW dalam derajat
%1=cerah
%2=berawan

if cuaca = = 1
   if frek = = 1.8
      PFD = -178;
   elseif frek = = 1.9
      PFD=-179;
   elseif frek==27.5
```

```
        PFD=-183;
    elseif frek==31
        PFD=-183;
    else
        PFD=NaN
    end
else cuaca==2
    if frek==1.8
        PFD=-179;
    elseif frek==1.9
        PFD=-180;
    elseif frek==27.5
        PFD=-188;
    elseif frek==31
        PFD=-188;
    else
        PFD=NaN
    end
end


HPBW=22/(frek*diameter)
Tmatahari=((1-exp(-
0.48/(1.2*HPBW))^2))/((frek^2)*(0.48^2)))*(10^(0.1*(PFD+250)));
```

C.1.6 Sub Program Perhitungan Thermal Noise Hujan

```
function suhu_sky=sky(T_bumi,ht,frek,alpha,pol)
% Menghitung Sky Noise Temperature
```

```
T_bumi_K=T_bumi+273;
Tm=1.12*(T_bumi_K)-50;
suhu_sky=Tm*(1-10^(-0.1*hujan(ht,frek,alpha,pol)));
```

C.1.7 Sub Program Perhitungan Suhu Antena

```
function suhuantena=Tant(langit,cuaca,frek,diameter,ht,alpha,pol,T_bumi)

%kondisi langit
%1=cerah
%2=hujan

%kondisi cuaca
%1=cerah
%2=berawan
A_hujan=hujan(ht,frek,alpha,pol);

Tground=273+T_bumi;
if langit = = 1
%    suhuantena=matahari(cuaca,frek,diameter)+Tground
    suhuantena=matahari(cuaca,frek,diameter);
elseif langit = = 2

suhuantena=(matahari(cuaca,frek,diameter)./10.^(0.1*A_hujan))+sky(T_b
umi,ht,frek,alpha,pol)+Tground;
end

%suhuantena = ((matahari (cuaca, frek, diameter) )./ (10^ (0.1* hujan(ht,
frek, alpha, pol)))) +s ky(T_bumi,ht,frek,alpha,pol)+Tground
```

```
%(matahari(cuaca,frek,diameter)/10^(0.1*hujan(ht,frek,alpha,pol)))+sky(
T_bumi,ht,frek,alpha,pol)+Tground
```

C.2 Performansi Kanal

C.2.1 Program Utama Simulasi Performansi Kanal Ricean dengan Spreader Code Walsh 4 Chips

```
clear all
qpsk4 = zeros(1,40);

L=10176;
data=gen_data(L);

odata=orth_mod4(data);% 1x108544
kirim=PhaseMod(odata,2); %1x54272
SNR=linspace(1,40,40);
a=length(kirim);
for i=1:40
    kanal=ricean(50,2*a,a)';
    sig=kirim(:).*kanal(:);
    received_signal = AddNoise(normalize(sig),SNR(i),2);
    demodulated_signal = PhaseDemod(normalize(received_signal),2);
    dedata=de_ort4(demodulated_signal) ;
    [e,qpsk4(i)]=biterr(data,dedata)
end
```

```
save hasil_simulasi4 qpsk4

figure;
plot(10*log10(SNR),qpsk4)
title('BER QPSK pada Kanal Ricean')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;

figure;
semilogy(10*log10(SNR),qpsk4,'b^-')
title('BER QPSK pada Kanal Ricean')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;
```

C.2.2 Program Utama Simulasi Performansi Kanal Ricean dengan Spreader Code Walsh 8 Chips

```
clear all

qpsk8 = zeros(1,40);

L=10176;
data=gen_data(L);

odata=orth_mod8(data);% 1x108544
kirim=PhaseMod(odata,2); %1x54272
```

```
SNR=linspace(1,40,40);
a=length(kirim);
for i=1:40
    kanal=ricean(50,2*a,a)';
    sig=kirim(:).*kanal(:);
    received_signal = AddNoise(normalize(sig),SNR(i),2);
    demodulated_signal = PhaseDemod(normalize(received_signal),2);
    dedata=de_ort8(demodulated_signal) ;
    [e,qpsk8(i)]=biterr(data,dedata)
end

save hasil_simulasi8 qpsk8

figure;
plot(10*log10(SNR),qpsk8)
title('BER QPSK pada Kanal Ricean 8')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;

figure;
semilogy(10*log10(SNR),qpsk8,'bo-')
title('BER QPSK pada Kanal Ricean')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;
```

C.2.3 Program Utama Simulasi Performansi Kanal Ricean dengan Spreader Code Walsh 16 Chips

```
clear all

qpsk16 = zeros(1,40);

L=10176;
data=gen_data(L);

odata=orth_mod16(data);% 1x108544
kirim=PhaseMod(odata,2); %1x54272
SNR=linspace(1,40,40);
a=length(kirim);
for i=1:40
    kanal=ricean(50,2*a,a)';
    sig=kirim(:).*kanal(:);
    received_signal = AddNoise(normalize(sig),SNR(i),2);
    demodulated_signal = PhaseDemod(normalize(received_signal),2);
    dedata=de_ort16(demodulated_signal) ;
    [e,qpsk16(i)]=biterr(data,dedata)
end

save hasil_simulasi16 qpsk16

figure;
plot(10*log10(SNR),qpsk16)
title('BER QPSK pada Kanal Ricean 16')
```

```
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;

figure;
semilogy(10*log10(SNR),qpsk16,'bx-')
title('BER QPSK pada Kanal Ricean 16')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;
```

C.2.4 Program Utama Simulasi Performansi Kanal Ricean dengan Spreader Code Walsh 32 Chips

```
clear all

qpsk32 = zeros(1,40);

L=10175;
data=gen_data(L);

odata=orth_mod32(data);% 1x108544
kirim=PhaseMod(odata,2); %1x54272
SNR=linspace(1,40,40);
a=length(kirim);
for i=1:40
    kanal=ricean(50,2*a,a)';
```

```
    sig=kirim(:).*kanal(:);
    received_signal = AddNoise(normalize(sig),SNR(i),2);
    demodulated_signal = PhaseDemod(normalize(received_signal),2);
    dedata=de_ort32(demodulated_signal) ;
    [e,qpsk32(i)]=biterr(data,dedata)
end

save hasil_simulasi32 qpsk32

figure;
plot(10*log10(SNR),qpsk32)
title('BER QPSK pada Kanal Ricean 32')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;

figure;
semilogy(10*log10(SNR),qpsk32,'bx-')
title('BER QPSK pada Kanal Ricean 32')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;
```

C.2.5 Program Utama Simulasi Performansi Kanal Ricean dengan Spreader Code Walsh 64 Chips

```
clear all
```

```
qpsk64 = zeros(1,20);

L=10176;
data=gen_data(L);

odata=orth_mod(data);% 1x108544
kirim=PhaseMod(odata,2); %1x54272
SNR=linspace(1,20,20);
a=length(kirim);
for i=1:20
    kanal=ricean(50,2*a,a)';
    sig=kirim(:).*kanal(:);
    received_signal = AddNoise(normalize(sig),SNR(i),2);
    demodulated_signal = PhaseDemod(normalize(received_signal),2);
    dedata=de_ort(demodulated_signal) ;
    [e,qpsk64(i)]=biterr(data,dedata)
end

save hasil_simulasi64 qpsk64

figure;
plot(10*log10(SNR),qpsk64)
title('BER QPSK pada Kanal Ricean 64')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;

figure;
```

```
semilogy(10*log10(SNR),qpsk32,'bv-')
title('BER QPSK pada Kanal Ricean 64')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;
```

C.2.6 Program Utama Simulasi Performansi BER pada Kanal Ricean Tanpa Spreader Walsh

```
clear all

tanpaspreading = zeros(1,40);

L=10176;
data=gen_data(L);

kirim=PhaseMod(data,2); %1x54272
SNR=linspace(1,40,40);
a=length(kirim);
for i=1:40
    kanal=ricean(50,2*a,a)';
    sig=kirim(:).*kanal(:);
    received_signal = AddNoise(normalize(sig),SNR(i),2);
    demodulated_signal = PhaseDemod(normalize(received_signal),2);
    [e,tanpaspreading(i)]=biterr(data,demodulated_signal)
end

save kanalnospreading tanpaspreading
```

```
figure;
plot(10*log10(SNR),tanpaspreading)
title('BER QPSK pada Kanal Ricean')
ylabel('Pe')
xlabel('Eb/No (dB)')
grid on;

figure;
semilogy(10*log10(SNR),tanpaspreading,'bv-')
title('BER QPSK pada Kanal Ricean')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;
```

C.2.7 Program Utama Simulasi Performansi BER AWGN Tanpa Spreader

```
clear all

qpskawgn = zeros(1,40);

L=10176;
data=gen_data(L);

kirim=PhaseMod(data,2);
SNR=linspace(1,40,40);
for i=1:40
   received_signal = AddNoise(normalize(kirim),SNR(i),2);
```

```
    demodulated_signal = PhaseDemod(normalize(received_signal),2);
    [e,qpskawgn(i)]=biterr(data,demodulated_signal)
end

save awgnsaja qpskawgn


figure;
plot(10*log10(SNR),qpskawgn)
title('BER QPSK pada Kanal AWGN')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;


figure;
semilogy(10*log10(SNR),qpskawgn,'b^-')
title('BER QPSK pada Kanal AWGN')
ylabel('BER')
xlabel('Eb/No (dB)')
grid on;
```

C.2.8 Program Utama Perhitungan BER QPSK pada Kanal Ricean Teori

```
function [Perr]=PendekatanTeori(gamma,K)


%   gamma adalah EB/No
%   K adalah faktor ricean
```

```
a=(1+K)./(2*(gamma+1+K));
b=exp(-K*gamma./(gamma+1+K));
Perr=a(:).*b(:);
Perr1=Perr';
save ricean_teori Perr1

figure;
semilogy(gamma,Perr,'b*-');
grid on;
xlabel('Eb/No (dB)');
ylabel('BER');
```

C.2.9 Sub Program Spreader Menggunakan Code Walsh 4 Chips

```
function orthd=orth_mod4(intd)

walsh=walsh(4);
temp=[0];
for i=1:2:length(intd)
   wn=intd(i)+2*intd(i+1)+1;
   temp=[temp,walsh(wn,:)];
end
temp(:,1)=[];
orthd=temp;
```

C.2.10 Sub Program Spreader Menggunakan Code Walsh 8 Chips

```
function orthd=orth_mod8(intd)
```

```
walsh=walsh(8);
temp=[0];

for i=1:3:length(intd)
    wn=intd(i)+2*intd(i+1)+4*intd(i+2)+1;
    temp=[temp,walsh(wn,:)];
end

temp(:,1)=[];
orthd=temp;
```

C.2.11 Sub Program Spreader Menggunakan Code Walsh 16 Chips

```
function orthd=orth_mod16(intd)

walsh=walsh(16);
temp=[0];

for i=1:4:length(intd)
    wn=intd(i)+2*intd(i+1)+4*intd(i+2)+8*intd(i+3)+1;
    temp=[temp,walsh(wn,:)];
end

temp(:,1)=[];
orthd=temp;
```

C.2.12 Sub Program Spreader Menggunakan Code Walsh 32 Chips

```
function orthd=orth_mod32(intd)


walsh=walsh(32);

temp=[0];


for i=1:5:length(intd)

    wn=intd(i)+2*intd(i+1)+4*intd(i+2)+8*intd(i+3)+16*intd(i+4)+1;

    temp=[temp,walsh(wn,:)];

end

temp(:,1)=[];

orthd=temp;
```

C.2.13 Sub Program Spreader Menggunakan Code Walsh 64 Chips

```
function orthd=orth_mod(intd)


walsh=walsh(64);

temp=[0];


for i=1:6:length(intd)


wn=intd(i)+2*intd(i+1)+4*intd(i+2)+8*intd(i+3)+16*intd(i+4)+32*intd(i
+5)+1;

    temp=[temp,walsh(wn,:)];

end
```

```
temp(:,1)=[];
orthd=temp;
```

C.2.14 Sub Program Despreader Menggunakan Code Walsh 8 Chips

```
function dorth=de_ort8(d)

%demodulasi  64 ary modulasi orthogonal

walsh=walsh(8);
wrn=0;
temp=zeros(1,8);
dum=[0];

for i=1:8:length(d)
    temp=d(i:i+7);
    c=zeros(1,8);
    for rn=1:8
        for k=1:8
        if walsh(rn,k)==temp(k)
        c(rn)=c(rn)+1;
        end
    end
end

max=0;
```

```
for n=1:8
    if c(n)>max
    max=c(n);
    index=n;
    end
end

wrn=index-1;



if (wrn>=4)&(wrn<8)
    c2=1;
    wrn=wrn-4;
else c2=0;
end

if (wrn>=2)&(wrn<4)
    c1=1;
    wrn=wrn-2;
else c1=0;
end

c0=wrn;
dum=[dum c0 c1 c2];
end
dum(:,1)=[];
dorth=dum;
```

C.2.15 Sub Program Despreader Menggunakan Code Walsh 4 Chips

```
function dorth=de_ort4(d)


walsh=walsh(4);
wrn=0;
temp=zeros(1,4);
dum=[0];


for i=1:4:length(d)
    temp=d(i:i+3);
    c=zeros(1,4);
    for rn=1:4
        for k=1:4
        if walsh(rn,k)==temp(k)
        c(rn)=c(rn)+1;
        end
    end
end


max=0;


for n=1:4
    if c(n)>max
    max=c(n);
    index=n;
    end
```

```
end

wrn=index-1;

if (wrn>=2)&(wrn<4)
    c1=1;
    wrn=wrn-2;
else c1=0;
end

c0=wrn;
dum=[dum c0 c1];
end
dum(:,1)=[];
dorth=dum;
```

C.2.16 Sub Program Despreader Menggunakan Code Walsh 16 Chips

```
function dorth=de_ort16(d)

walsh=walsh(16);
wrn=0;
temp=zeros(1,16);
dum=[0];

for i=1:16:length(d)
    temp=d(i:i+15);
```

```
    c=zeros(1,16);
    for rn=1:16
        for k=1:16
        if walsh(rn,k)==temp(k)
        c(rn)=c(rn)+1;
        end
    end
end

max=0;

for n=1:16
    if c(n)>max
    max=c(n);
    index=n;
    end
end

wrn=index-1;

if (wrn>=8)&(wrn<16)
    c3=1;
    wrn=wrn-8;
else c3=0;
end

if (wrn>=4)&(wrn<8)
    c2=1;
```

```
    wrn=wrn-4;
else c2=0;
end

if (wrn>=2)&(wrn<4)
    c1=1;
    wrn=wrn-2;
else c1=0;
end

c0=wrn;
dum=[dum c0 c1 c2 c3];
end
dum(:,1)=[];
dorth=dum;
```

C.2.17 Sub Program Despreader Menggunakan Code Walsh 32 Chips

```
function dorth=de_ort32(d)

walsh=walsh(32);
wrn=0;
temp=zeros(1,32);
dum=[0];

for i=1:32:length(d)
    temp=d(i:i+31);
```

```
    c=zeros(1,32);
    for rn=1:32
        for k=1:32
        if walsh(rn,k)==temp(k)
        c(rn)=c(rn)+1;
        end
    end
end

max=0;

for n=1:32
    if c(n)>max
    max=c(n);
    index=n;
    end
end

wrn=index-1;

if (wrn>=16)&(wrn<32)
    c4=1;
    wrn=wrn-16;
else c4=0;
end

if (wrn>=8)&(wrn<16)
    c3=1;
```

```
    wrn=wrn-8;
else c3=0;
end

if (wrn>=4)&(wrn<8)
    c2=1;
    wrn=wrn-4;
else c2=0;
end

if (wrn>=2)&(wrn<4)
    c1=1;
    wrn=wrn-2;
else c1=0;
end

c0=wrn;
dum=[dum c0 c1 c2 c3 c4];
end
dum(:,1)=[];
dorth=dum;
```

C.2.18 Sub Program Despreader Menggunakan Code Walsh 64 Chips

```
function dorth=de_ort(d)

%demodulasi  64 ary modulasi orthogonal
```

```
walsh=walsh(64);
wrn=0;
temp=zeros(1,64);
dum=[0];

for i=1:64:length(d)
    temp=d(i:i+63);
    c=zeros(1,64);
    for rn=1:64
        for k=1:64
        if walsh(rn,k) = = temp(k)
        c(rn)=c(rn)+1;
        end
    end
end

max=0;

for n=1:64
    if c(n)>max
    max=c(n);
    index=n;
    end
end

wrn=index-1;
```

```
if (wrn>=32)&(wrn<=64)
    c5=1;
    wrn=wrn-32;
else c5=0;
end

if (wrn>=16)&(wrn<32)
    c4=1;
    wrn=wrn-16;
else c4=0;
end

if (wrn>=8)&(wrn<16)
    c3=1;
    wrn=wrn-8;
else c3=0;
end

if (wrn>=4)&(wrn<8)
    c2=1;
    wrn=wrn-4;
else c2=0;
end

if (wrn>=2)&(wrn<4)
    c1=1;
    wrn=wrn-2;
else c1=0;
```

```
end

c0=wrn;
dum=[dum c0 c1 c2 c3 c4 c5];
end
dum(:,1)=[];
dorth=dum;
```

C.2.19 Sub Program Pembangkitan Code Walsh

```
function Wn=walsh(N,option);
% Generator Walsh
M = ceil(log(N)/log(2));
if (nargin ~= 2),
    option = '++';
end
if (option= ='+-'),
    if 2^M == 1,
        Wn = [1];
    elseif 2^M = = 2,
        Wn = [1 1; 1 -1];
    else
        Wn =  [1 1 1 1; 1 -1 1 -1; 1 1 -1 -1; 1 -1 -1 1];
        for k = 1:M-2,
            Wn = [Wn Wn; Wn (-Wn)];
        end
    end
else
```

```
        if 2^M = = 1,
                Wn = [1];
        elseif 2^M = = 2,
                Wn = [1 1; 1 0];
        else
                Wn =  [1 1 1 1; 1 0 1 0; 1 1 0 0; 1 0 0 1];
                for k = 1:M-2,
                        Wn = [Wn Wn; Wn ~Wn];
                end
        end
end
```

C.2.20 Sub Program Pembangkitan Respon Kanal Ricean

```
function [row,a]= ricean(fd,fs,Ns)


%N = 4;
N = 10;
while(N)
  if (N < 2*fd*Ns/fs)
    N = 2*N;
  else
    break;
  end
end
N_inv = ceil(N*fs/(2*fd));
delta_f = 2*fd/N;
delta_T_inv = 1/fs;
```

```
I_input_time1 = randn(1,N);
Q_input_time1 = randn(1,N);

I_input_freq = fft(I_input_time1);
Q_input_freq = fft(Q_input_time1);

SEZ=zeros(1,N/2);
SEZ(1) = 1.5/(pi*fd);

for j=2:N/2
   f(j) = (j-1)*delta_f;
   SEZ(j) = 1.5/(pi*fd*sqrt(1-(f(j)/fd)^2));
   SEZ(N-j+2) = SEZ(j);
end


SEZabs = abs(SEZ);

%SEZ(10) = 100*1.5/(pi*fd*sqrt(1-(f(10)/fd)^2));
SEZ(10) = (100*1.5)/(pi*fd*sqrt(1-(f(10)/fd)^2));
SEZ(N-10) = SEZ(10);
%SEZ((N/2)+1) = 1000*SEZ(10);
SEZ((N/2)+1) = 1*SEZ(10);
SEZ=(SEZabs/abs(SEZ))*SEZ;

a=(100*1.5)/(pi*fd*sqrt(1-(f(10)/fd)^2));
```

```
I_output_freq = I_input_freq .* sqrt(SEZ);

Q_output_freq = Q_input_freq .* sqrt(SEZ);


I_temp = [I_output_freq(1:N/2) zeros(1,N_inv-N) I_output_freq(N/2+1:N)];

I_output_time = ifft(I_temp);


Q_temp = [Q_output_freq(1:N/2) zeros(1,N_inv-N) Q_output_freq(N/2+1:N)];

Q_output_time = ifft(Q_temp);


r=zeros(1,N_inv);

for j=1:N_inv

   r(j) = sqrt( (abs(I_output_time(j)))^2 + (abs(Q_output_time(j)))^2);

end


rms = sqrt(mean(r.*r));

row = r(1:Ns)/rms;
```

C.2.21 Sub Program Penggambaran Distribusi Gaussian

```
x = (0:0.1:10)';

y2 = gaussmf(x, [1.3 5]);

y3 = gaussmf(x, [1.3 5]);

x1 = (5:0.1:15)';

figure;

plot(x,y2,'b-',x1,y3,'r-');
```

C.2.22 Sub Program Modulator M-PSK

```
function y = PhaseMod(x, bits)

l=bits;
M = 2^l;
if size(x,1)==1
   x = x.';
end
a = size(x,1);
x = num2str(x);
s = reshape(x,a/l,l);
ss = bin2dec(s);
y = exp(j*2*pi*ss/M);
```

C.2.23 Sub Program Demodulator M-PSK

```
function y=PhaseDemod(x, bits)

M = 2^bits;
if size(x,1) ==1
   x = x.';
end
ss = 0:M-1;
symbols = exp(j*2*pi*ss/M);
b = dec2bin(0:M-1);
for i=1:M
   e(i,:) = symbols(i)-x.';
```

```
end
[min,index] = min(abs(e));
y = b(index,:);
y = reshape(y, bits*size(y,1),1);
y = str2num(y).';
```

C.2.24 Sub Program Normalisasi Sinyal

```
function y=normalize(x)
y=x/sqrt(mean(abs(x).*abs(x)));
```

C.2.25 Sub Program Loading Data pada Kanal Tanpa Spreading Walsh

```
clear all
load kanalnospreading
load awgnsaja
load ricean_teori

Perr1x=[Perr1 0];
qpskawgny=[0.5030 qpskawgn];
tanpaspreadingz=[0.5036 tanpaspreading];

EbtoNo=0:40;

figure;
```

```
semilogy(EbtoNo, Perr1x, '*-', EbtoNo, tanpaspreadingz, 'o-', EbtoNo, qpskawgny, 's-');

legend('Kanal Ricean Teori', 'Kanal Ricean Simulasi', 'Kanal AWGN', 'K = 7,3291');

xlabel('Eb/No (dB)');

ylabel('BER');

grid on;
```

C.2.26 Sub Program Loading Data pada Kanal Dengan Variasi Panjang Chips Spreading Walsh

```
clear all


load hasil_simulasi4

load hasil_simulasi8

load hasil_simulasi16

load hasil_simulasi32

load hasil_simulasi64


qpsk4a=[0.4992 qpsk4];

qpsk8a=[0.5072 qpsk8];

qpsk16a1=[0.4943 qpsk16 zeros(1,20)];

qpsk32a=[0.5010 qpsk32];

qpsk64a1=[0.5036 qpsk64 zeros(1,20)];


EbtoNo=0:40;


figure;
```

```
semilogy(EbtoNo,qpsk4a,'o-',EbtoNo,qpsk8a,'s-',EbtoNo,qpsk16a1,'*-',EbtoNo,qpsk32a,'p-',EbtoNo,qpsk64a1,'h-');

legend('Walsh 4 Chips','Walsh 8 Chips','Walsh 16 Chips','Walsh 32 Chips','Walsh 64 Chips');

xlabel('Eb/No (dB)');

ylabel('BER');

%grid on;
```

C.2.27 Sub Program Simulasi Performansi BER pada Kanal Ricean Tanpa Spreader Walsh pada Eb/No = 0 dB

```
function [e,tanpaspreading] = kondisi_awal_nospreading(EbtoNo)


L=10176;

data=gen_data(L);


kirim=PhaseMod(data,2);

SNR=linspace(1,40,40);

a=length(kirim);

kanal=ricean(50,2*a,a)';

sig=kirim(:).*kanal(:);


EbtoNo=EbtoNo+(EbtoNo==0)*eps;


received_signal = AddNoise(normalize(sig),EbtoNo,2);

demodulated_signal = PhaseDemod(normalize(received_signal),2);

[e,tanpaspreading]=biterr(data,demodulated_signal)
```

C.2.28 Sub Program Simulasi Performansi BER pada Kanal AWGN Tanpa Spreader Walsh pada Eb/No = 0 dB

```
function [e,qpskawgn]=kondisi_awal_awgn(EbtoNo)

L=10176;
data=gen_data(L);

kirim=PhaseMod(data,2);
SNR=linspace(1,40,40);

EbtoNo=EbtoNo+(EbtoNo==0)*eps;

received_signal = AddNoise(normalize(kirim),EbtoNo,2);
demodulated_signal = PhaseDemod(normalize(received_signal),2);
[e,qpskawgn]=biterr(data,demodulated_signal)
```

C.2.29 Sub Program Generator Data Biner

```
function data=gen_data(L)

% pembangkitan data

d=randint(L-8,1,2,200);
dat=d';
data=[dat,0 0 0 0 0 0 0 0];
```

C.2.30 Sub Program Plotting Spektral Daya Doppler Spread

```
function hasil=doppler(f,fc,fd)

% f dalam range f-fc atau f+fc
% fc adalah frekuensi pembawa
% fd adalah frekuensi doppler maksimum

S= 7.94./(pi*fd*sqrt(1-((f-fc)./fd).^2))
figure;
plot(f,S)
title('Spektral Daya Frekuensi Doppler');
ylabel('Se(f)');
xlabel('frekuensi')
```

C.2.31 Sub Program Simulasi Performansi BER pada Kanal Ricean Eb/No=0 dB

```
function [e,q64] = db0(EbtoNo)

L=10176;
data=gen_data(L);

odata=orth_mod(data);

kirim=PhaseMod(odata,2);
a=length(kirim);
```

```
kanal=ricean(50,2*a,a)';

sig=kirim(:).*kanal(:);

EbtoNo=EbtoNo+(EbtoNo = = 0)*eps;

received_signal = AddNoise(normalize(sig),EbtoNo,2);

demodulated_signal = PhaseDemod(normalize(received_signal),2);

dedata=de_ort(demodulated_signal) ;
[e,q64]=biterr(data,dedata)
```

C.2.32 Sub Program Simulasi Penambahan Noise Total pada Sinyal

```
function y = AddNoise(x, SNRperBit, l)

% l=jumlah bit per simbol

sigma = sqrt(l/SNRperBit/2);

a = size(x,1);
b = size(x,2);

%y = x + sigma*randn(a,b) + j*sigma*randn(a,b);
y = x + sigma*randn(a,b);
```