LAMPIRAN A
LISTING PROGRAM

Program simulasi untuk melihat diperlukan channel coding.

```
%Program memplot Distribusi level tegangan kanal bit pada kanal AWGN.
%Untuk nilai Eb/N0 = 20dB
clear;
clc;
clf;
N = 1e6;                    %Jumlah bit yang akan dikirimkan
EbpN0 = 20;                 %Nilai Eb/N0

%Data random yang dikirim berupa level tegangan 1 atau -1
signal = sign(randn(1,N));

%N-point data random sebagai AWGN dengan daya noise tertentu
noise = randn(1,N)./10^(EbpN0/20)./sqrt(2);

received = signal + noise;      %Data yang diterima
hist(received,500);             %Memplot diagram

xlabel('Distribusi tegangan untuk bit "1" = -1V, dan bit "0" = +1V');
ylabel('Jumlah channel bits');
grid on

%Program memplot Distribusi level tegangan kanal bit melalui kanal AWGN.
%Untuk nilai Eb/N0 = 6dB
clear;
clc;
clf;
N = 1e6;                    %Jumlah bit yang akan dikirimkan
EbpN0 = 6;                  %Nilai Eb/N0

%Data random yang dikirim berupa level tegangan 1 atau -1
signal = sign(randn(1,N));

%N-point data random sebagai AWGN dengan daya noise tertentu
noise = randn(1,N)./10^(EbpN0/20)./sqrt(2);

received = signal + noise;      %Data yang diterima
hist(received,500);             %Memplot diagram

xlabel('Distribusi tegangan untuk bit "1" = -1V, dan bit "0" = +1V');
ylabel('Jumlah channel bits');
grid on;
```

Program simulasi forward error control menggunakan viterbi decoding

```
%Program Simulasi Convolutional Coding Dengan Viterbi Decoding

% Pembangkitan sinyal biner acak, dengan parameter tertentu
Fd=1;
Fs=4;                       %Frekuensi sampling
N=Fs/Fd;
M=4;
k=log2(M);
numSymb=100;            %Jumlah simbol
numPlot=20;                 %Jumlah yang diplot
codeRate=1/2;               %Code rate
constlen=7;                  %Constraint length
SNRpBitDemo=3;
SNR=SNRpBitDemo*k;
seed=[654321 123456];
rand('state',seed(1));
randn('state',seed(2));
msg_orig=randsrc(numSymb,1,[0:1]);
figure(1);
stem([0:numPlot-1],msg_orig(1:numPlot),'bx');
axis([0 numPlot -0.2 1.2]);
xlabel('Waktu');
ylabel('Amplituda');
title('Simbol Biner Sebelum Convolutional coding');



% Program untuk melakukan Convolutional Encoding terhadap  sinyal biner

constlen=[3];                       %Constraint length
codegen=[7 5];                  %Generator polinomial
tblen=32;                           %Trace back length
codeRate=1/2;                       %Code rate
trel=poly2trellis(constlen,codegen);          %Trellis description
[msg_enc_bi]=convenc(msg_orig,trel);
numEncPlot=numPlot./codeRate;
tEnc=[0:numEncPlot-1]*codeRate;
figure(2);
stem(tEnc,msg_enc_bi(1:length(tEnc)),'rx');
axis([min(tEnc) max(tEnc) -0.2 1.2]);
xlabel('Waktu');ylabel('Amplituda');
```

```
title('Simbol Biner Setelah Convolutional Encoding');

%Modulasi terhadap simbol hasil Encoding
msg_enc=bi2de(reshape(msg_enc_bi,size(msg_enc_bi,2)*k,size(msg_enc_bi,1)/k)');
grayencod=bitxor([0:M-1],floor([0:M-1]/2));
msg_gr_enc=grayencod(msg_enc+1);
msg_tx=dmodce(msg_gr_enc,Fd,[Fs,pi/4],'psk',M);
msg_rx=awgn(msg_tx,SNR-10*log10(1/codeRate)-10*log10(N));
numModPlot=numEncPlot*Fs./k;
tMod=[0:numModPlot-1]./Fs.*k;
figure(3);
plot(tMod,real(msg_tx(1:length(tMod))),'c-',tMod,imag(msg_tx(1:length(tMod))),'m-
');
axis([min(tMod) max(tMod) -1.5 1.5]);
xlabel('sample/Waktu');ylabel('Amplituda');
title('Simbol Hasil Encoding Setelah Dimodulasi Baseband QPSK');
legend('In-phase','Quadrature',0);

%Decoding dengan Viterbi decoder
qcode=quantiz(msg_rx,[0.001,.1,.3,.5,.7,.9,.999]);
msg_dec=vitdec(msg_demod_bi,trel,tblen,'cont','soft',3);    %soft decision
figure(5);
stem([0:numPlot-1],msg_orig(1:numPlot),'rx');
hold on;
stem([0:numPlot-1],msg_dec(1+tblen:numPlot+tblen),'bo');
hold off;
axis([0 numPlot -0.2 1.2]);
xlabel('Waktu');ylabel('Amplituda');
title('Simbol Hasil Decoding');
```

Program menggunakan Hard decision pada viterbi decoding

```
%Program Conv-coding vit-decoding dengan K = 3

clear;
clc;
clf;
N=1e6                                   %Jumlah bit yang akan disimulasikan
EbpN0=linspace(3,7,9);                  %Nilai Eb/N0 yang ingin dihitung BER-nya

for i=1: length(EbpN0)
   BER(i)=0;
   for j=1:5
      b=sign(randn(1,N));               %Bit data random yang dipancarkan
      orig_bit = b;
      for k=1:length(b)
         if b(k) == 1;
            b(k) = 0;
         else
            b(k) = 1;
         end
      end


      %Constraint length K = 3, code generator = [7 5]
      trellis=poly2trellis(3,[7 5]);
      code=convenc(b,trellis);

      for l = 1:length(code)
         if code(l) == 0
            code(l) = 1;
         else
            code(l) = -1;
         end
      end

      %N-point data random sebagai AWGN dengan daya noise tertentu
      n=randn(1,2*N)./10^(EbpN0(i)/20)./sqrt(2);
      r=code+n;                         %Data diterima oleh receiver
      r=sign(r);                        %Data hasil recovery oleh decision circuit

      for k=1:length(r)
         if r(k) == 1;
```

```
            r(k) = 0;
        else
            r(k) = 1;
        end
    end

    tb = 2;                                 % Traceback length for decoding
    decoded = vitdec(r,trellis,tb,'trunc','hard');    % Viterbi Decoding

    for k=1:length(decoded)
        if decoded(k) == 0;
            decoded(k) = 1;
        else
            decoded(k) = -1;
        end
    end
    %Menghitung jumlah bit error
    BER(i) = BER(i) + sum(abs(decoded-orig_bit))/2;
  end
  BER(i) = BER(i)/5/N;        %BER=Jml error rata2 / jml bit yg dikirim
end
semilogy(EbpN0,BER, '*');
hold on;

%Routine untuk memplot P(e) sebagai fungsi dari Eb/N0 secara teoritis
EbpN0=linspace(0,10,11);            %Nilai Eb/N0 yang ingin dihitung BER-nya

t=linspace(0,max(EbpN0),500);

Pe=0.5.*erfc(sqrt(10.^(t/10)));   semilogy(t,Pe,'r');
grid on;
title ('BER terhadap Eb/N0 untuk Kanal AWGN Dengan K = 3');
xlabel('Eb/N0 (dB)');
ylabel('BER');
disp ('Kurva menunjukkan nilai teoritis P(e)');
disp('Tanda bintang menunjukkan hasil simulasi BER');
legend('Hasil simulasi BER','Nilai teoritis P(e)',0);
```

```
%Program Conv-coding vit-decoding dengan K = 5

clear;
clc;
clf;
N=1e6                                   %Jumlah bit yang akan disimulasikan
EbpN0=linspace(3,7,9);                  %Nilai Eb/N0 yang ingin dihitung BER-nya

for i=1: length(EbpN0)
   BER(i)=0;
   for j=1:4
      b=sign(randn(1,N));               %Bit data random yang dipancarkan
      orig_bit = b;
      for k=1:length(b)
         if b(k) == 1;
            b(k) = 0;
         else
            b(k) = 1;
         end
      end


      %Constraint length K = 5, code generator = [35 23]
      trellis=poly2trellis(5,[35 23]);
      code=convenc(b,trellis);

      for l = 1:length(code)
         if code(l) == 0
            code(l) = 1;
         else
            code(l) = -1;
         end
      end

      %N-point data random sebagai AWGN dengan daya noise tertentu
      n=randn(1,2*N)./10^(EbpN0(i)/20)./sqrt(2);
      r=code+n;                         %Data diterima oleh receiver
      r=sign(r);                        %Data hasil recovery oleh decision circuit

      for k=1:length(r)
         if r(k) == 1;
            r(k) = 0;
         else
```

```
            r(k) = 1;
        end
    end

    tb = 2;                                 % Traceback length for decoding
    decoded = vitdec(r,trellis,tb,'trunc','hard');    % Viterbi Decoding

    for k=1:length(decoded)
        if decoded(k) == 0;
            decoded(k) = 1;
        else
            decoded(k) = -1;
        end
    end
    %Menghitung jumlah bit error
    BER(i) = BER(i) + sum(abs(decoded-orig_bit))/2;
  end
  BER(i) = BER(i)/5/N;        %BER=Jml error rata2 / jml bit yg dikirim
end
semilogy(EbpN0,BER, '*');
hold on;

%Routine untuk memplot P(e) sebagai fungsi dari Eb/N0 secara teoritis
EbpN0=linspace(0,10,11);            %Nilai Eb/N0 yang ingin dihitung BER-nya

t=linspace(0,max(EbpN0),500);

Pe=0.5.*erfc(sqrt(10.^(t/10)));   semilogy(t,Pe,'r');
grid on;
title ('BER terhadap Eb/N0 untuk Kanal AWGN Dengan K = 5');
xlabel('Eb/N0 (dB)');
ylabel('BER');
disp ('Kurva menunjukkan nilai teoritis P(e)');
disp('Tanda bintang menunjukkan hasil simulasi BER');
legend('Hasil simulasi BER','Nilai teoritis P(e)',0);
```

```
%Program Conv-coding vit-decoding dengan K = 7

clear;
clc;
clf;
N=1e6                                   %Jumlah bit yang akan disimulasikan
EbpN0=linspace(3,7,9);                  %Nilai Eb/N0 yang ingin dihitung BER-nya

for i=1: length(EbpN0)
   BER(i)=0;
   for j=1:4
      b=sign(randn(1,N));               %Bit data random yang dipancarkan
      orig_bit = b;
      for k=1:length(b)
         if b(k) == 1;
            b(k) = 0;
         else
            b(k) = 1;
         end
      end


      %Constraint length K = 7, code generator = [171 133]
      trellis=poly2trellis(7,[171 133]);
      code=convenc(b,trellis);

      for l = 1:length(code)
         if code(l) == 0
            code(l) = 1;
         else
            code(l) = -1;
         end
      end

      %N-point data random sebagai AWGN dengan daya noise tertentu
      n=randn(1,2*N)./10^(EbpN0(i)/20)./sqrt(2);
      r=code+n;                         %Data diterima oleh receiver
      r=sign(r);                        %Data hasil recovery oleh decision circuit

      for k=1:length(r)
         if r(k) == 1;
            r(k) = 0;
         else
```

```
            r(k) = 1;
        end
    end

    tb = 2;                                 % Traceback length for decoding
    decoded = vitdec(r,trellis,tb,'trunc','hard');    % Viterbi Decoding

    for k=1:length(decoded)
        if decoded(k) == 0;
            decoded(k) = 1;
        else
            decoded(k) = -1;
        end
    end
    %Menghitung jumlah bit error
    BER(i) = BER(i) + sum(abs(decoded-orig_bit))/2;
  end
  BER(i) = BER(i)/5/N;        %BER=Jml error rata2 / jml bit yg dikirim
end
semilogy(EbpN0,BER, '*');
hold on;

%Routine untuk memplot P(e) sebagai fungsi dari Eb/N0 secara teoritis
EbpN0=linspace(0,10,11);            %Nilai Eb/N0 yang ingin dihitung BER-nya

t=linspace(0,max(EbpN0),500);

Pe=0.5.*erfc(sqrt(10.^(t/10)));   semilogy(t,Pe,'r');
grid on;
title ('BER terhadap Eb/N0 untuk Kanal AWGN Dengan K = 7');
xlabel('Eb/N0 (dB)');
ylabel('BER');
disp ('Kurva menunjukkan nilai teoritis P(e)');
disp('Tanda bintang menunjukkan hasil simulasi BER');
legend('Hasil simulasi BER','Nilai teoritis P(e)',0);
```

```
%Program Conv-coding vit-decoding dengan K = 9

clear;
clc;
clf;
N=1e6                                   %Jumlah bit yang akan disimulasikan
EbpN0=linspace(3,7,9);                  %Nilai Eb/N0 yang ingin dihitung BER-nya

for i=1: length(EbpN0)
   BER(i)=0;
   for j=1:4
      b=sign(randn(1,N));               %Bit data random yang dipancarkan
      orig_bit = b;
      for k=1:length(b)
         if b(k) == 1;
            b(k) = 0;
         else
            b(k) = 1;
         end
      end


      %Constraint length K = 9, code generator = [753 561]
      trellis=poly2trellis(9,[753 561]);
      code=convenc(b,trellis);

      for l = 1:length(code)
         if code(l) == 0
            code(l) = 1;
         else
            code(l) = -1;
         end
      end

      %N-point data random sebagai AWGN dengan daya noise tertentu
      n=randn(1,2*N)./10^(EbpN0(i)/20)./sqrt(2);
      r=code+n;                         %Data diterima oleh receiver
      r=sign(r);                        %Data hasil recovery oleh decision circuit

      for k=1:length(r)
         if r(k) == 1;
            r(k) = 0;
         else
```

```
            r(k) = 1;
        end
    end

    tb = 2;                                 % Traceback length for decoding
    decoded = vitdec(r,trellis,tb,'trunc','hard');    % Viterbi Decoding

    for k=1:length(decoded)
        if decoded(k) == 0;
            decoded(k) = 1;
        else
            decoded(k) = -1;
        end
    end
    %Menghitung jumlah bit error
    BER(i) = BER(i) + sum(abs(decoded-orig_bit))/2;
  end
  BER(i) = BER(i)/5/N;        %BER=Jml error rata2 / jml bit yg dikirim
end
semilogy(EbpN0,BER, '*');
hold on;

%Routine untuk memplot P(e) sebagai fungsi dari Eb/N0 secara teoritis
EbpN0=linspace(0,10,11);            %Nilai Eb/N0 yang ingin dihitung BER-nya

t=linspace(0,max(EbpN0),500);

Pe=0.5.*erfc(sqrt(10.^(t/10)));   semilogy(t,Pe,'r');
grid on;
title ('BER terhadap Eb/N0 untuk Kanal AWGN Dengan K = 9');
xlabel('Eb/N0 (dB)');
ylabel('BER');
disp ('Kurva menunjukkan nilai teoritis P(e)');
disp('Tanda bintang menunjukkan hasil simulasi BER');
legend('Hasil simulasi BER','Nilai teoritis P(e)',0);
```

Program menggunakan Soft decision pada viterbi decoding

```
%Program Conv-coding vit-decoding dengan K = 3

clear;
clc;
clf;
N=1e6;                                  %Jumlah bit yang akan disimulasikan
EbpN0=linspace(0,7,9);                  %Nilai Eb/N0 yang ingin dihitung BER-nya

for i=1: length(EbpN0)
   BER(i)=0;
   for j=1:4
      b=sign(randn(1,N));               %Bit data random yang dipancarkan
      orig_bit = b;
      for k=1:length(b)
         if b(k) == 1;
            b(k) = 0;
         else
            b(k) = 1;
         end
      end


      %Constraint length K = 3, code generator = [7 5]
      trellis=poly2trellis(3,[7 5]);
      code=convenc(b,trellis);

      for l = 1:length(code)
         if code(l) == 0
            code(l) = 1;
         else
            code(l) = -1;
         end
      end

      %N-point data random sebagai AWGN dengan daya noise tertentu
      n=randn(1,2*N)./10^(EbpN0(i)/20)./sqrt(2);



      tb = 24;                                  % Traceback length for decoding
      % To prepare for soft-decision decoding, map to decision values.
```

```
        [x,qcode] = quantiz(1-n,[-.75 -.5 -.25 0 .25 .5 .75 ],...
        [7 6 5 4 3 2 1 0]); % Values in qcode are between 0 and 2^3-1.

        decoded = vitdec(qcode,trellis,tb,'trunc','soft',3);    % Viterbi Decoding



        %Menghitung jumlah bit error
        BER(i) = BER(i) + sum(abs(decoded))/2;
    end
    BER(i) = BER(i)/4/N;        %BER=Jml error rata2 / jml bit yg dikirim
end
semilogy(EbpN0,BER, '*');
hold on;

%Routine untuk memplot P(e) sebagai fungsi dari Eb/N0 secara teoritis
EbpN0=linspace(0,10,11);            %Nilai Eb/N0 yang ingin dihitung BER-nya

t=linspace(0,max(EbpN0),500);

Pe=0.5.*erfc(sqrt(10.^(t/10)));   semilogy(t,Pe,'r');
grid on;
title ('BER terhadap Eb/N0 untuk Kanal AWGN Dengan K = 3');
xlabel('Eb/N0 (dB)');
ylabel('BER');
disp ('Kurva menunjukkan nilai teoritis P(e)');
disp('Tanda bintang menunjukkan hasil simulasi BER');
legend('Hasil simulasi BER','Nilai teoritis P(e)',0);
```

```
%Program Conv-coding vit-decoding dengan K = 5

clear;
clc;
clf;
N=1e6                                   %Jumlah bit yang akan disimulasikan
EbpN0=linspace(0,7,9);                  %Nilai Eb/N0 yang ingin dihitung BER-nya

for i=1: length(EbpN0)
   BER(i)=0;
   for j=1:4
      b=sign(randn(1,N));               %Bit data random yang dipancarkan
      orig_bit = b;
      for k=1:length(b)
         if b(k) == 1;
            b(k) = 0;
         else
            b(k) = 1;
         end
      end


      %Constraint length K = 5, code generator = [35 23]
      trellis=poly2trellis(5,[35 23]);
      code=convenc(b,trellis);

      for l = 1:length(code)
         if code(l) == 0
            code(l) = 1;
         else
            code(l) = -1;
         end
      end

      %N-point data random sebagai AWGN dengan daya noise tertentu
      n=randn(1,2*N)./10^(EbpN0(i)/20)./sqrt(2);



      tb = 2;                                % Traceback length for decoding
       % To prepare for soft-decision decoding, map to decision values.
      [x,qcode] = quantiz(1-n,[-.75 -.5 -.25 0 .25 .5 .75],...
```

```
      [7 6 5 4 3 2 1 0]); % Values in qcode are between 0 and 2^3-1.

      decoded = vitdec(qcode,trellis,tb,'trunc','soft',3);    % Viterbi Decoding



      %Menghitung jumlah bit error
      BER(i) = BER(i) + sum(abs(decoded))/2;
   end
   BER(i) = BER(i)/5/N;         %BER=Jml error rata2 / jml bit yg dikirim
end
semilogy(EbpN0,BER, '*');
hold on;

%Routine untuk memplot P(e) sebagai fungsi dari Eb/N0 secara teoritis
EbpN0=linspace(0,10,11);              %Nilai Eb/N0 yang ingin dihitung BER-nya

t=linspace(0,max(EbpN0),500);

Pe=0.5.*erfc(sqrt(10.^(t/10)));   semilogy(t,Pe,'r');
grid on;
title ('BER terhadap Eb/N0 untuk Kanal AWGN Dengan K = 5');
xlabel('Eb/N0 (dB)');
ylabel('BER');
disp ('Kurva menunjukkan nilai teoritis P(e)');
disp('Tanda bintang menunjukkan hasil simulasi BER');
legend('Hasil simulasi BER','Nilai teoritis P(e)',0);
```

```
%Program Conv-coding vit-decoding dengan K = 7

clear;
clc;
clf;
N=1e6                                   %Jumlah bit yang akan disimulasikan
EbpN0=linspace(0,7,9);                  %Nilai Eb/N0 yang ingin dihitung BER-nya

for i=1: length(EbpN0)
   BER(i)=0;
   for j=1:4
      b=sign(randn(1,N));               %Bit data random yang dipancarkan
      orig_bit = b;
      for k=1:length(b)
         if b(k) == 1;
            b(k) = 0;
         else
            b(k) = 1;
         end
      end


      %Constraint length K = 7, code generator = [171 133]
      trellis=poly2trellis(7,[171 133]);
      code=convenc(b,trellis);

      for l = 1:length(code)
         if code(l) == 0
            code(l) = 1;
         else
            code(l) = -1;
         end
      end

      %N-point data random sebagai AWGN dengan daya noise tertentu
      n=randn(1,2*N)./10^(EbpN0(i)/20)./sqrt(2);


      tb = 2;                                % Traceback length for decoding
      % To prepare for soft-decision decoding, map to decision values.
      [x,qcode] = quantiz(1-n,[-.75 -.5 -.25 0 .25 .5 .75],...
      [7 6 5 4 3 2 1 0]); % Values in qcode are between 0 and 2^3-1.
```

```
      decoded = vitdec(qcode,trellis,tb,'trunc','soft',3);    % Viterbi Decoding



      %Menghitung jumlah bit error
      BER(i) = BER(i) + sum(abs(decoded))/2;
   end
   BER(i) = BER(i)/5/N;         %BER=Jml error rata2 / jml bit yg dikirim
end
semilogy(EbpN0,BER, '*');
hold on;

%Routine untuk memplot P(e) sebagai fungsi dari Eb/N0 secara teoritis
EbpN0=linspace(0,10,11);              %Nilai Eb/N0 yang ingin dihitung BER-nya

t=linspace(0,max(EbpN0),500);

Pe=0.5.*erfc(sqrt(10.^(t/10)));   semilogy(t,Pe,'r');
grid on;
title ('BER terhadap Eb/N0 untuk Kanal AWGN Dengan K = 7');
xlabel('Eb/N0 (dB)');
ylabel('BER');
disp ('Kurva menunjukkan nilai teoritis P(e)');
disp('Tanda bintang menunjukkan hasil simulasi BER');
legend('Hasil simulasi BER','Nilai teoritis P(e)',0);
```

```
%Program Conv-coding vit-decoding dengan K = 9

clear;
clc;
clf;
N=1e6                                   %Jumlah bit yang akan disimulasikan
EbpN0=linspace(0,7,9);                  %Nilai Eb/N0 yang ingin dihitung BER-nya

for i=1: length(EbpN0)
   BER(i)=0;
   for j=1:4
      b=sign(randn(1,N));               %Bit data random yang dipancarkan
      orig_bit = b;
      for k=1:length(b)
         if b(k) == 1;
            b(k) = 0;
         else
            b(k) = 1;
         end
      end


      %Constraint length K = 9, code generator = [753 561]
      trellis=poly2trellis(9,[753 561]);
      code=convenc(b,trellis);

      for l = 1:length(code)
         if code(l) == 0
            code(l) = 1;
         else
            code(l) = -1;
         end
      end

      %N-point data random sebagai AWGN dengan daya noise tertentu
      n=randn(1,2*N)./10^(EbpN0(i)/20)./sqrt(2);

      tb = 2;                                % Traceback length for decoding
      % To prepare for soft-decision decoding, map to decision values.
      [x,qcode] = quantiz(1-n,[-.75 -.5 -.25 0 .25 .5 .75],...
      [7 6 5 4 3 2 1 0]); % Values in qcode are between 0 and 2^3-1.

      decoded = vitdec(qcode,trellis,tb,'trunc','soft',3);    % Viterbi Decoding
```

```
      %Menghitung jumlah bit error
      BER(i) = BER(i) + sum(abs(decoded))/2;
   end
   BER(i) = BER(i)/5/N;        %BER=Jml error rata2 / jml bit yg dikirim
end
semilogy(EbpN0,BER, '*');
hold on;

%Routine untuk memplot P(e) sebagai fungsi dari Eb/N0 secara teoritis
EbpN0=linspace(0,10,11);            %Nilai Eb/N0 yang ingin dihitung BER-nya

t=linspace(0,max(EbpN0),500);

Pe=0.5.*erfc(sqrt(10.^(t/10)));   semilogy(t,Pe,'r');
grid on;
title ('BER terhadap Eb/N0 untuk Kanal AWGN Dengan K = 9');
xlabel('Eb/N0 (dB)');
ylabel('BER');
disp ('Kurva menunjukkan nilai teoritis P(e)');
disp('Tanda bintang menunjukkan hasil simulasi BER');
legend('Hasil simulasi BER','Nilai teoritis P(e)',0);
```