LAMPIRAN A

LISTING PROGRAM

1. Penghitung korelasi dan rate of distrotion dari PCM dan DPCM

```
%=========================================================
clear;
close all;
clc;
%=========================================================
function [R,H]=countDist(nama_file,D)
% Program penghitung korelasi dan rate of distortion
load data;
e=2.7283;
Dn=10.^(D./10);

% nama_file=['a' 'i' 'u' 'e' 'o']
nama_file=['p'];
[y,fs,nbits]=wavread(nama_file);
if fs>=8000
   inc=round(fs/8000);
fs=8000;

% Sub sampling menjadi 8000
y=y(1:inc:length(y));
% wavplay(yin,8000)
else
   error('fs is lower than 8 Khz, Please use Fs=>8 Khz');
end
y=y.*8192;

% source untuk DPCM
u=[y;0];
v=[0;y];
d=u-v;
```

```
% Hitung korelasi
ryy=xcorr(y,y);
rdd=xcorr(d,d);
x1=(length(ryy)-1)/2;
delay1=[-x1:1:x1];
x2=(length(rdd)-1)/2;
delay2=[-x2:1:x2];

% koefisien korelasi
rhoyy=ryy/max(ryy); % Normalisasi
rhodd=rdd/max(rdd); % Normalisasi

% hitung probabilitas
proby=hist(y,256)/length(y);
probd=hist(d,256)/length(d);
proby=proby(find(proby~=0));
probd=probd(find(probd~=0));

% hitung entropy sumber
Hy=sum(proby.*(log10(proby)/log10(2)));
Hd=sum(probd.*(log10(probd)/log10(2)));
H=[Hy;Hd];

% hitung lower bound untuk rate of distortion dari Shannon
RDy=Hy-(0.5.*log10(2*pi.*e.*Dn));
RDd=Hd-(0.5.*log10(2*pi.*e.*Dn));
R=[RDy;RDd];
RFS=R.*fs;

% ploting autocorellation
subplot(211);plot(delay1,rhoyy);
```

```
xlabel('shift');ylabel('Coefficient of Correlation');
title('Autocorrellation of source signal for PCM');
grid on;
subplot(212);plot(delay2,rhodd);
xlabel('shift');ylabel('Coefficient of Correlation');
title('Autocorrellation of source signal for DPCM');
grid on;
figure;subplot(211);plot(Dlog,RDy,'-*',Dlog,RDd,':ro');
ylabel('R(D)(bit/syimbol)');
xlabel('distortion D (db)');
title('Shanon Lower Bound for Rate of Distortion (R(D))Function');
legend('PCM source','DPCM source');
grid on;
subplot(212);plot(Dlog,RFS(1,:),'-*',Dlog,RFS(2,:),':ro');
ylabel('R(D)(bit/second)');
xlabel('distortion D (db)');
fs=fs/1000;
t=sprintf('Sampling Rate=%.2f Khz',fs);
title(t);
legend('PCM source','DPCM source');
grid on;
```

2. Penghitung distorsi untuk ADPCM

```
function [Dlog,D,e,e2]=Distortion(nama_file,N)
% program penghitung distorsi untuk ADPCM
% baca file
[yin,fs]=wavread(nama_file);
if fs >=8000
   inc=round(fs/8000);
   fs=8000;

 % sub sampling menjadi 8000
   yin=yin(1:inc:length(yin));
%wavplay(yin,8000);
else
   error('Fs is lower than 8 Khz, Please use Fs => 8 Khz');
end;
y14=yin.*8192;

% masukan ke enkoder ADPCM
[I,sr,dout,dqout,yout,aiout,biout.tdout,trout,alout]=adpcm(y14,N);
yout=sr./8192;
% wavplay(yout,8000);
% hitung distorsi
e=yin-yout;
e2=e.^2;
D=mean(e2);
Dlog=10*log10(D);
```

3. Program ADPCM dengan standar G. 726

```
function [I,sr,dout,dqout,yout,aiout,biout,tdout,trout,alout]=adpcm(sl,N,format);

%   ADPCM  Adaptive Differential Pulse Code Modulation
%   [I,SR]=ADPCM(SL,N)proses enkoding sinyal input PCM
%   dengan kode sinyal SL
%   dengan N kbit/s adaptive differntial pulse code modulation (ADPCM)
%   N dipilih dari 40, 32, 24, 16 kbit/s.
%   jika N tidak ditentukan  maka bit rate diset N = 32 Kbit/s
%   output pertama I,  sinyal output dari enkoder yang merepresentasi
%   bit stream over the channel.
%   sampel I didapat dari -2^(N/8-1),...,+2^(N/8-1).
%   output kedua (opsional):
%   output adalah reconstructed speech signal SR
%   (terskala antara -8191 and +8191) yang menjadi keluaran
%   dari koresponding ADPCM dekoder untuk error-free transmission dari I.
%
%      [I,SR,D,DQ,Y,AI,BI,TD,TR,AL]=ADPCM(SL,N);
%
%      with the following interpretation:
%      D   prediction difference signal
%      DQ  quantized prediction difference signal
%      Y   scale factor for adaptive quantizer
%      AI  recursive predictor coefficients (2-column matrix)
%      BI  non-recursive predictor coefficients (6-column matrix)
%      TD  tone detector signal
%      TR  transient detctor signal
%      AL  speed control for scale factor adaptation
%   fungsi yang juga dipergunakan adalah APUPDATE, AIUPDATE,
    BIUPDATE
```

```
% input argument checks, default bit rate and format setting
if(nargin<1)|(nargin>3)
   disp('ADPCM:1to3 input variables required!')
   help adpcm,return
end
if nargin==1
   N=32;
   disp('*****N=32 kbit/s*****');
end
a=N==[40 32 24 16];
if(sum(a)~=1)
   N=32;
   disp('*****N=32 kbit/s*****');
end;
if nargin <=2
   format='lin';
end;
if isstr(format)~=1
   error('ADPCM:Input format must be string variable!')
elseif strcmp(format,'mu')
   sl=8191*mu2lin(sl);
elseif~strcmp(format,'lin')
   error('ADPCM:wrong input format')
end
if min(size(sl))~=1
   error('ADPCM :input signal must be row or coloumn vector only');
end;
sl=sl(:);
if max(abs(sl))>8191
   disp('ADPCM warning: there can be clipping');
   disp(['you should use the amplitude range of','-8191<= SL <=8191 for the input
signal SL!'])
```

```
end;
if max(sl)<=1
   disp(('ADPCM warning: there can be underflow'));
   disp(['you should use the amplitude range of','-8191<= SL <=8191 for the input
signal SL!'])
end;
I=zeros(size(sl));  % mengalokasikan memori keluaran (channel simbol untuk I)
sr=zeros(size(sl)); % mengalokasikan memori keluaran (reconstructed speech)
if (nargout>2)      % mengalokasikan memori untuk output diagnostik
   dout=zeros(size(sl));
   dqout=zeros(size(sl));
   yout=zeros(size(sl));
   aiout=zeros(length(sl),2);
   biout=zeros(length(sl),6);
   tdout=zeros(size(sl));
   trout=zeros(size(sl));
   alout=zeros(size(sl));
end

% mendefinisikan tabel kuantisasi dan tabel adaptasi faktor skala
if N==40
   h_fi=fliplr([6 6 5 4 3 2 1 1 1 1 1 0 0 0 0 0]);
   h_wi=fliplr([43.50 33.06 27.50 22.38 17.50 13.69 11.19 8.81 6.25 3.63 2.56
2.50 2.44 1.50 0.88 0.88]);
   h_iadq=fliplr([4.42 4.21 4.02 3.81 3.59 3.35 3.09 2.80 2.48 2.14 1.75 1.32 0.81
0.22 -0.52 -inf]);
   h_qan=fliplr([4.31 4.12 3.91 3.70 3.47 3.22 2.95 2.64 2.32 1.95 1.54 1.08 0.52 -
0.13 -0.96 -inf]);
end;
if N==32
   h_fi=fliplr([7 3 1 1 1 0 0 0]);
   h_wi=fliplr([70.13 22.19 12.38 7.00 4.00 2.56 1.13 -0.75]);
   h_iadq=fliplr([3.32 2.91 2.52 2.13 1.66 1.05 0.03 -inf]);
   h_qan=fliplr([3.12 2.72 2.34 1.91 1.38 0.62 -0.98 -inf]);
end;
if N==24
   h_fi=fliplr([7 2 1 0]);
   h_wi=fliplr([36.38 8.56 1.88 -0.25]);
   h_iadq=fliplr([2.91 2.13 1.05 -inf]);
   h_qan=fliplr([2.58 1.70 0.06 -inf]);
end;
if N==16
   h_fi=fliplr([7 0]);
   h_wi=fliplr([27.44 -1.38]);
   h_iadq=fliplr([2.85 0.91]);
   h_qan=fliplr([2.04 -inf]);
end;

% Numerical constants
c1=0.125;       % 2^-3
c2=0.9375;      % 1-2^-4
c3=0.03125;     % 2^-5
c4=0.96875;     % 1-2^-5
c5=0.015625;    % 2^-6
c6=0.984375;    % 1-2^-5
c7=0.0078125;   % 2^-7
c8=0.9921875;   % 1-2^-7
c9=0.01171875;  % 3*2^-8
c10=0.99609375; % 1-2^-8
c11=0.998046875;%1-2^-9

% inisialisasi variabel internal
yu=1.06; y1=0; dms=0; dml=0; ap=0;
dq=zeros(1,7);     % difference signal vector [dq(k), dq(k-1),...,dq(k-6)]
```

```
ai=zeros(1,2);      % recursive predictor coefficient [a1(k), a2(k)]
bi=zeros(1,6);      % non-recursive predictor coeficients
                    % [b1(k), b2(k),...,b6(k)]
p=zeros(1,3);       % surrogate vector for recursive predictor
                    % adaptation [p(k), p(k-1), p(k-2)]
srv=zeros(1,2);     % reconstructed speech vector [sr(k), sr(k-1)]

% proses looping dari semua sampel sinyal
for k=1:length(sl)
   %kuantisasi dan komputasi dari defferent signal
   sez=bi*dq(1:6)';        % prediksi sinyal non rekursif
   se=ai*srv'+sez;         % prediksi sinyal rekursif
   d=sl(k)-se;             % komputasi different signal
   al=min(ap,1);           % parameter speed control
   y=al*yu+(1-al)*y1;      % update faktor skala kuantisasi
   x=log2(abs(d))-y;       % skala different signal pada domain logaritmik
   I(k)=sign(d)*(sum(h_qan<=x)-1);     % proses kuantisasi
   % invers quantization dan signal reconstruction
   dqh=sign(I(k))*2^(h_iadq(abs(I(k))+1)+y);      %invers quantization/ log
   dq=[dqh dq(1:6)];           % shift different signal vector
   p=[dq(1)+sez,p(1:2)];       % shift surrogate signal vector
   sr(k)=se+dq(1);             % reconstruct speech sample
   srv=[sr(k),srv(1)];         % shift reconstructed speech vector

   % adaptasi koefisien predictor
   ai=aiupdate(ai,p,c9,c10,c7,c8,c2);
   bi=biupdate(bi,dq,N,c7,c10,c11);

   % adaptasi quantizer scale factor
   WI=h_wi(abs(I(k))+1);
   yu=c4*y+c3*WI;
   yu=max(min(yu,10.00),1.06);
```

```
    y1=c6*y1+c5*yu;

    % tone and transion detection
    td=ai(2)<-0.71875;          % deteksi sinyal patrial
    tr=td&(abs(dq(1))>24*2^y1);
    if tr==1
        ai=zeros(1,2);
        bi=zeros(1,6);
    end;

    % quantizer adaptation speed control
    FI=h_fi(abs(I(k))+1);
    dml=c8*dml+c7*FI;
    dms=c4*dms+c3*FI;
    ap=apupdate(ap,dms,dml,y,td,tr,c1,c2);  % parameter speed control


    % write diagnostic output
    if(nargout>2)
        dout(k)=d;
        dqout(k)=dqh;
        yout(k)=y;
        aiout(k,:)=ai;
        biout(k,:)=bi;
        tdout(k)=td;
        trout(k)=tr;
        alout(k)=al;
    end
end;

function ai=aiupdate(aio,p,c1,c2,c3,c4,c5);
% ai=aiupdate(aio,p,c1,c2,c3,c4,c5)
```

```
%
% function :        update a1 and a2 coefficient of second order predictor
%
% input:aio         old coefficient of secon order predictor
%                   p   surrogate vector
%                   c1  =3*2^-8
%                   c2  =1-2^-8
%                   c3  =2^-7
%                   c4  =1-2^-7
%                   c5  =1-2^-4
%
% output:    :      ai  coefficient of second order recursive predictor

sgnp=sign(p);
sgnp(2:3)=sign(sgnp(2:3)+0.5);      %
ai(1)=c2*aio(1)+c1*sgnp(1)*sgnp(2);
if abs(aio(1))>0.5
   fa1=2*sign(aio(1));
else
   fa1=4*aio(1);
end;
ai(2)=c4*aio(2)+c3*(sgnp(1)*sgnp(3)-fa1*sgnp(1)*sgnp(2));
ai(2)=max(min(ai(2),0.75),-0.75);
b=c5-ai(2);
ai(1)=max(min(ai(1),b),-b);

function ap=apupdate(apo,dms,dml,y,td,tr,c1,c2);
% ap=apupdate(apo,dms,dml,y,td,tr,c1,c2);
% function :    update speed control parameter ap(k)
%               impementation of recomendation G. 726
%
% input:apo     old speed control parameter ap(k-1)
%              dml long term average of F[I(k)]
%              dms short term average of F[I(k)]
%              y   quantizer scale factor y(k)
%              td  tone detector
%              tr  transion detector
%              c1  =2^-3
%              c2  =1-2^-4
% output:      ap  Speed control parameter

if tr==1
   ap=1;
elseif(abs(dms-dml)>=c1*dml|y<3|td==1)
   ap=c2*apo+c1;
else
   ap=c2*apo;
end;

function bi=biupdate(bio,dq,N,c1,c2,c3);
% BIUPDATE      Update non recursive predictor coefficients
% bi=biupdate(bio,dq,N,c1,c2,c3);
%
% function:     update coeficient of sixth order predictor
%
% input:bio     old coefficient of sixth order predictor
%              dq  difference signal vector
%              c1  =2^-7
%              c2  =1-2^-8
%              c3  =1-2^-9
%
% output:      bi  coefficients of sixth order predictor
```

```
sgndq=sign(dq);
sgndq(2:6)=sign(sgndq(2:6)+0.5);
if N==40
   bi=c3*bio+c1*sgndq(1)*sgndq(2:7);
else
   bi=c2*bio+c1*sgndq(1)*sgndq(2:7);
end;
```

4. Program Eksekusi

```
%==========================================================
close all;
clear all;
clc;
%==========================================================
file=['a' 'i' 'u' 'e' 'o'];
% file=['p' 'h'];
pjg=length(file);
D(1:pjg,1:4)=[0];
N=[40 32 24 16];
for i=1:pjg
    clc;
    for j=1:4
        display=sprintf('File yang diproses adalah %s.wav dengan bitrate %d
kbps',file(i),N(j));
        disp(display);
        [Dlog(i,j),D(i,j),e,e2]=Distortion(file(i),N(j));
    end;
end;
meanD=mean(D);
meanD=10*log10(meanD)
save data
```