LAMPIRAN A

PROGRAM M-FILE PADA MATLAB 6.5.1

Program Utama

```
clc;
% program Mpeg-4 audio %

clear;
close all;
clc;

% parameter input %
Lfr = 160;                    %interval frame dalam sample
Lan = 256;                    %panjang frame yang dianalisa
over = 96;                    %overlap
file = 'maranata.wav';       %input wav file

% ambil tabel
[SE_Gain,SE_Shape1,SE_Shape2,SE_Shape3,SE_Shape4,SE_Shape5,SE_Shape6]=tabel
1;
[cbL0_g,cbL_s,cbL1_g,cbL1_s,lsf_tbl,pd_tbl,d_tbl,lsf_q_enh]=tabel2;

%======ENCODER======%
%inialisasi encoder%
[si_asli,fs,nbits]=wavread(file);               %membaca wav file
mfr=floor(length(si_asli)/Lfr);    %jumlah frame maksimum
dummy_si(1:over/2)=0;
si(1:length(si_asli)+over/2)=[dummy_si si_asli(:,1)'];

%window hamming:
for n=1:Lan
        window(n)=0.54-0.46*cos(2*pi*(n-1)/(Lan-1));
end;
[ipc_bsamp] = bsamp (window,Lan);
min_gap = 4/256;
rasio_predict = 0.7;
mineub = 1000000000;
for i=1:10
        lsf_prev(i)=(pi*(i+1)/11);
end

% Inisialisasi pitch :
% Pitch =Prm(1) pitch dengan proses tracking
% Prob =Prm(2) puncak pertama dibagi puncak kedua dari autokorelasi
% R0r   =Prm(3) puncak pertama autokorelasi - termodifikasi
% rawR0r=Prm(4) puncak pertama autokorelasi - belum dimodifikasi
% rawPch=Prm(5) pitch tanpa tracking

currPrm (1:5) = 0;
```

```
prevPrm(1:5) = 0;
prevPrm (1) = (Lan/2) + 20;
prevlev =0;
prevGp = 0;
prevVUV = 0;
erl = 0;
erh = 0;

% Koefisien Filter
a = 1.1*[1 -1.99806642 1];
b = [1 -1.96282243 0.96849918];
c = [1 -1.99963331 0.99999999];
d = [1 -1.85809791 0.86545998];
e = 1.226*[1 0.551543 0.152];
f = [1 0.89 0.198025];
q = 0.768*[1 -2 1];
h = [1 1.4635 0.6084];

% Konversi Dimensi VQ Harmonik
for j=1:64
    if j==32
        coef (32)=1;
    else
        coef(j)=sin(pi*(j-32)/8)*(0.5-0.5*cos(2*pi*j/64))/(pi*(j-32)/8);
    end;
end;
Gain1(1:32)=SE_Gain';
for j=1:16
    SE_temp(1:44) = SE_Shape1((j-1)+1:j,:);
    Shape1(j,1:44)=SE_temp(1:44);
end;

for j=1:16
    SE_temp(1:44) = SE_Shape2((j-1)+1:j,:);
  Shape2(j,1:44)=SE_temp(1:44);
end;

for j=1:16
    x2(j,:) = Gain1(j)*(Shape1 (j,:)+Shape2(j,:)) ;
    x2(j+16,:) = Gain1(j+16)*(Shape1(j,:)+Shape2(j,:)) ;
end;

% 4.2 Analisis sinyal suara
% 4.2.1  Proses awal
for i=1:mfr  over));

    % Normalisasi
    % 4.2.2 windowing
    sif_temp(i,1:256) = sif ((i-1)*Lfr+1:i*Lfr+over);
    sw((i-1)*Lfr+1:i*Lfr+over)=window.*sif_temp;

    % 4.2.3 Perhitungan Autokorelasi
    sw_temp = sw((i-1)*Lfr+1:i*Lfr+over);
```

```
r = autocorr (sw_temp,10);

% 4.2.4 Perhitungan Koefisien Prediksi Linier
[lpc,E,rc] = levinson(r);

% 4.2.5 Perhitungan Koefisien LSP dan LSF
lsf = poly2lsf(lpc);
lsf_curr = lsf;
lsp = cos (lsf);

% 4.2.6 Kuantisasi Koefisien LSP
lsf_temp(1)=0;
lsf_temp(2:10) = lsf(1:9)*0.5;
panLsf (1:10) = 0;
for j=1:9
    panLsf (j) = lsf_temp(j+1)*2;
end;

dLsf(1) = panLsf(1);
for j=1:9
    dLsf(j+1)=panLsf(j+1)-panLsf(j);
end;

dLsf(11)=1-panLsf(10);
for j=1:11
    if dLsf(j) < min_gap
        dLsf(j) = min_gap*2;
    end;
    dLsf(j)= 1/dLsf(j);
end;

lsfWeight(1:10) = dLsf(1:10) + dLsf(2:11);
w_fact =1;

for j=1:4
    lsfWeight (j) = lsfWeight (j)*w_fact;
end;

for j=5:8
    w_fact = w_fact*0.694;
    lsfWeigt(j) = lsfWeight(j)*w_fact;
 end;

for j=9:10
    w_fact = w_fact*0.510;
    lsfWeight(j) = lsfWeight(j)*w_fact;
end;

% Kuantisasi Vektor LSF Stage Pertama
err1(1:32)=0;
for m =1:32
    for j=1:10
        err1(m)=err1(m)+((lsf(j)-lsf_tbl(m,j))^2)*lsfWeight(j);
```

```
   end;
end;

for m = 1:32
   if err1(m) == min(err1)
      nVq1 (1,i) = m;
      lsf_first(1:10) = lsf_tbl(m,:);
   end;
end;
```

% Kuantisasi Vektor Stage Kedua
% a. Kuantisasi Vektor tanpa Prediksi Interframe

```
err201(1:80)=0;
err202(1:80)=0;
err211(1:80)=0;
err212(1:80)=0;

for m=1:80
   for j=1:5
      lsf_res(6-j)=lsf(6-j)-lsf_first(6-j) ;
      err201(m)=err201(m)+((lsf_res(6-j)+d_tbl(m,j)^2)*lsfWeight(6-j));
   end;
   for j=6:10
      lsf_res(11-j)=lsf(11-j)-lsf_first(11-j);
      err202(m)=err202(m)+((lsf_res(11-j)+d_tbl(m,j-5)^2)*lsfWeight(11-j));
   end;
   for j=1:5
      lsf_res(6-j)=lsf(6-j)-lsf_first(6-j);
      err211(m)=err211(m)+((lsf_res(6-j)-d_tbl(m,j))^2)*lsfWeight(6-j);
   end;
   for j=6:10
      lsf_res(11-j)=lsf(11-j)-lsf_first(11-j);
      err212(m)=err212(m)+((lsf_res(11-j)-d_tbl(m,j-5))^2)*lsfWeight(11-j);
   end;

   err2_total11 = sum(err201) - sum(err211);
   err2_total12 = sum(err202) - sum(err212);
   err2_total1 = err2_total11 - err2_total12;
```

   % b. Kuantisasi Vektor dengan Prediksi Interframe

```
   err221(1:80)=0; err222(1:80)=0;
   err231(1:80)=0; err232(1:80)=0;

   for m=1:80
      for j=1:5
         lsf_pres(6-j)=lsf (6-j)-((1-0.7)*lsf_first(6-j)-0.7*lsf_prev(6-j));
         err221(m)=err221(m)+((lsf_res(6-j)+pd_tbl(m,j))^2)*lsfWeight(6-j);
      end;
      for j=6:10
         lsf_pres(11-j)=lsf(11-j)-((1-0.7)*lsf_first(11-j)-0.7*lsf_prev(11-j));
         err222(m)=err222(m)+((lsf_res(11-j)+pd_tbl(m,j-5))^2)*lsfWeight(11-j);
      end;
      for j=1:5
```

```
        lsf_pres(6-j)=lsf(6-j)-((1-0.7)*lsf_first(6-j)-0.7*lsf_prev(6-j));
        err231(m)=err231(m)+((lsf_res(6-j)-pd_tbl(m,j))^2)*lsfWeight(6-j);
    end;
    for j=6:10
        lsf_pres(11-j)=lsf(11-j)-((1-0.7)*lsf_first(11-j)-0.7*lsf_prev(11-j));
        err232(m)=err232(m)+((lsf_res(11-j)-pd_tbl(m,j-5))^2)*lsfWeight(11-j);
    end;
end;

err2_total21 = sum(err221) - sum(err231);
err2_total22 = sum(err222) - sum(err232);
err2_total2 = err2_total21 - err2_total22;
```

% pencarian vektor kuantisasi

```
if err2_total1 < err2_total2            %gunakan tanpa prediksi interframe (d_tbl)
    nVq1(3,i) = 1;
    if err2_total11 < err2_total12      %untuk koefisien 1-5
        if sum(err201) < sum(err211)    %gunakan kombinasi penambahan
            nVq1(4,i) = 1;
            for m=1:80
                if err201(m)==min(err201)
                    nVq1(2,i) = m;
                    lsf_curr(1:5) = lsf_first(1:5)+d_tbl(m,:);
                end;
            end;
        else                            %gunakan kombinasi pengurangan
            nVq1(4,i) = 2;

            for m=1:80
                if err211(m)==min(err211)
                    nVq1(2,i) = m;
                    lsf_curr(1:5) = lsf_first(1:5)-d_tbl(m,:);
                end;
            end;
        end;

    else %untuk koefisien 6-10
        if sum(err2O2) < sum(err212) %gunakan kombinasi penambahan
            nVq1(4,i)= 1;

            for m=1:80
                if err202(m)==min(err202)
                    nVq1(2,i) = m;
                    lsf_curr(6:10) = lsf_first(6:10)+d_tbl(m,:);
                end;
            end;

        else                            %gunakan kombinasi pengurangan

            nVq1 (4,i) = 2;

            for m =l:80
                if err212(m)==min (err212)
                nVq1(2,i) = m;
                lsf_curr(6:10) = lsf_first(6:10)-d_tbl(m,:);
            end;
        end;
    end;
end;

else                                  %gunakan kuantisasi dengan prediski interframe (pd_tbl)

    nVq1(3,i) = 2;

    for j=1:10
        lsf_predict(j)=(1-rasio_predict)*lsf_first(j)+...
            rasio_predict*lsf_prev(j);
    end;

    if err2_total21 < err2_total22              %untuk koefisien 1-5
        if sum(err221)< sum(err231)             %gunakan kombinasi penambahan
            nVq1(4,i)=1;
            for m=1:80
                if err221(m)==min(err221)
                    nVq1(2,i) = m;
                    lsf_curr(1:5) = lsf_predict(1:5)+pd_tbl(m,:);
                end;
            end;

        else                                    %gunakan kombinasi pengurangan
            nVq1(4,i) = 2;

            for m =1:80
                if err231(m) ==min(err231)
                    nVq1(2,i)= m;
                    lsf_curr(1:5) = lsf_predict(1:5)-pd_tbl(m,:);
                end;
            end;
        end;

    else %untuk koefisien 6-10

        if sum(err222) < sum(err232) %gunakan kombinasi penambahan
            nVq1(4,i) = 1;

            for m=1:80
                if err222(m)==min(err222)
                    nVq1(2,i) = m;
                    lsf_curr(6:10) = lsf_predict(6:10)+pd_tbl(m,:);
                end;
            end;

        else %gunakan kombinasi pengurangan

            nVq1(4,i) = 2;
        for m=1:80
            if err232(m)==min(err232)
                nVq1(2,i) = m;
                lsf_curr(6:10) = lsf_predict(6:10)-pd_tbl(m,:);
            end;
        end;
    end;
end;

lsf_second = stabil_lsf(lsf_curr);

% Kuantisasi Vektor LSF Stage Ketiga
resVQ = lsf-lsf_second;
err3(1:256)=0;

for m=1:256
    for j=1:10
        err3(m)=err3(m)+((resVQ(j)-lsf_q_enh(m,j))^2)*lsfWeight(j);
    end;
end;

for m=1:256
    if err3(m) == min(err3)
        nVq2(1,i) = m;
        lsf_third(1:10) = lsf_q_enh(m,:);
    end;
end;

qlsf = stabil_qlsf(lsf_temp);
```

% 4.2.7 Perhitungan Koefisien Prediksi Linier terkuantisasi

```
qlpc = lsf2poly (qlsf);
si_res((i-1)*Lfr+1:i*Lfr+over)=filter([1-qlpc(2:11)],1,sif_temp);

% 4.2.8 Estimasi Pitch
sir_temp = si_res((i-1)*Lfr+1:i*Lfr+over);
X(1:Lan)=fft(sir_temp,Lan);
[rms,ang,re,im,lev]=freq_balance(X,Lan);
ZeroXP=CntZeroXP (sir_temp);

[currPrm,global_pitch,peakPos1]=getpitch(currPrm,prevPrm,sir_temp,lev,prevGp);
currPrm(5)=currPrm(1);
[currPrm(1),mineub]= pitch_akhir(currPrm,rms,ipc_bsamp,global_pitch,mineub);
currPrm(1)=TrackingPitch(currPrm,prevPrm,global_pitch,prevVUV,peakPos1);
Prm(:,i)=currPrm'; % Parameter-parameter Pitch
```

% 4.2.9 Ekstraksi Magnituda Harmonik

```
E(1:Lan)=fft(window,Lan);
Anum=0;
Adenum=0;
em=0;

for j=1:Lan
```

```
    Anum = Anum + abs(X(j))*abs(E(j));
    Adenum = Adenum + abs(E(j))^2;
end;

Am(i) = Anum/Adenum;

for j=1:Lan
    em = em + (abs(X(j))-abs(Am(i))*abs(E(j)))^2;
 end;

err_SE(i) = em;
nh(i) = floor(currPrm(1)/2); %jumlah harmonik
sbeban=1;

X1 = fft((i-1)*Lfr+1:i*Lfr+over,256);
PX1 = (X1).*conj(X1)/Lan;

for j=1:88
    x1(i,j)=max(PX1(ceil((j-1)*Lan/88)+1:ceil(j*Lan/88)));
end;

% 4.2.10 Perhitungan Sinyal Pembobotan Perceptual

for j=1:11
    wnum(j)=lpc(j) *0.4^(j-1);
    wden(j)=lpc(j)*0.9.^(j-1);
end;
sbeban((i-1)*Lfr+1:i*Lfr+over) = filter(wnum,wden,sw_temp);

% 4.2.11 Keputusan VUV
[idVUV] = VUVDecision(lev,prevlev,ZeroXP,currPrm,prevPrm,prevVUV);

% 4.2.12 Enkoder VQ Harmonik
% Menghitung matriks diagonal W(z) H(z)
[frek_si,w]=freqz(wnum,wden,Lan);
mag_freqW = abs(frek_si);
mag_freqH = abs(frek_si);
for j=1:Lan
    matriksWH(j)=mag_freqW(j)*mag_freqH(j);
end;
    Dist(1:32)=0;

    for k=1:32
        x3 = x1(i,1:44)-x2(k,:);
        x3 = x3;
        for j=1:44
            Dist(k) = Dist(k) + (matriksWH(j)*x3(j))^2;
        end;
    end;
    for k=1:32
        if Dist(k)==min(Dist)
            idGain = k;
            SE_Gain_kirim(i) = Gain1(k);
            if k<=16
                idSE_shape1 = k;
                idSE_ShaPe2 = k;
                SE_Shape1_kirim(i,1:44) = ShaPe1(k,1:44);
                SE_Shape2_kirim(i,1:44) = Shape2(k,1:44);
            else
                idSE_Shape1 = k-16;
                idSE_Shape2 = k-16;
                SE_Shape1_kirim(i,1:44) = Shape1(k-16,1:44);
                SE_Shape2_kirim(i,1:44) = Shape2(k-16,1:44);
            end;
        end;
    end;

    % 4.2.13 Enkoder Time Domain
    sif_temp1 = sif((i-1)*Lfr+1+(over/2):(i-1)*Lfr+(over/2)+(Lfr/2));
    sif_temp2 = sif((i-1)*Lfr+(over/2)+(Lfr/2)+1:i*Lfr+(over/2));
    sub_temp1 = filter(wnum,wden,sif_temp1);
    sub_temp2 = filter(wnum,wden,sif_temp2) ;
end;

% Koefisien-koefisien untuk frame sebelumnya :
lsf_prev = lsf_curr;
prevlev = lev;
prevVUV = idVUV;
prevPrm = currPrm;

% Paket bit-bit yang dikirim ke dekoder
lsfpack1=dec2bin(nVq1(1,i)-1,5);
lsfpack2=dec2bin(nVq1(2,i)-1,7);
lsfpack3=dec2bin(nVq1(3,i)-1,1);
lsfpack4=dec2bin(nVq1(4,i)-1,1);
lsfpackS=dec2bin(nVq2(1,i)-1,8);
vuvpack =dec2bin(idVUV,2);
pitchpack=dec2bin(floor(currPrm(1))-20,7);
shapepack1=dec2bin(idSE_Shape1-1,4);
shapepack2=dec2bin(idSE_Shape2-1,4);
gainpack1=dec2bin(idGain-1,5);

% Pengiriman Paket ke dekoder
              v                      UV
LSF1=[lsfpack1 lsfpack2 lsfpack3 lsfpack4];         % 10bit/20ms    10bit/20ms
LSF2 =[lsfpack5];                                   % 8bit/20ms     8bit/20ms
VUV =[vuvpack];                                     % 2bit/20ms     2bit/20ms
pitch=[pitchpack];                                  % 7bit/20ms
shape_harm1=[shapepack1 shapepack2];                %  8bit/20ms
gain_harm1=[gainpack1];                             %  5bit/20ms
shape_VXC1 =[kcb0spack1 kcb0spack2];                %               12bit/20ms
gain_VXC1=[kcbOgpack1 kcbOgpack2];                  %                8bit/20ms
 if idVUV > 0                                       % ------------  +   ------------  +
 BIT=[VUV LSF1 LSF2 Pitch shape_harm1 gain_harm1];
                                                    %  40bit/20ms   40bit/20ms
 else                                               % =2 kbps       = 2 kbps
```

```
   BIT=[VUV LSF1 LSF2 shape_VXC1 gain_VXC1];
end;



 %====DECODER ========%
 % Membuka paket bit-bit yang dikirim

% Dekode Koefisien LSF
lsf_first(1:10) = lsf_tbl(lsfunpack1,:);
if lsfunpack3==1
   if lsfunPack4==1
      lsf_second(1:5)=lsf_first(1:5)+d_tbl(lsfunpack2,:);
      lsf_second(6:10)=lsf_first(6:10)+d_tbl(lsfunpack2,:);
      % hal 95
   else
      lsf_second(1:5) = lsf_first(1:5)-d_tbl(lsfunpack2,:);
      lsf_second(6:10)= lsr_first(6:10)-d_tbl(lsfunPack2,:);
   end;
   %else

   for j=1:10
      lsf_predict(j) = (1-rasio_predict)*lsf_first(j)+...
         rasio_predict*lsf_prev(j);
   end;
   if lsfunPack4==1
      lsf_second(1:5) = lsf_predict(1:5)+pd_tbl(lsfunpack2,:);
      lsf_second(6:10) = lsf_predict(6:10)+pd_tbl(lsfunpack2,:);
   else
      lsf_second(1:5) = lsf_predict(1:5)-pd_tbl(lsfunpack2,:);
      lsf_second(6:10) = lsf_predict(6:10)-pd_tbl(lsfunpack2,:);
   end;
end;

lsf_second = stabil_lsf(lsf_second);
lsf_third = lsf_q_enh(lsfunpack5,:);
lsf_temp = lsf_second+lsf_third;
qlsf = stabil_qlsf(lsf_temp);
qlpc = lsf2poly(qlsf);

% Keputusan VUV
idVUV = vuvunpack;

% Nilai Pitch
pitch = Pitchunpack + 20;

% 4.2.14 Sinyal Sintesis
if idVUV > 0
[spektral_envelope]=d_quantV(gainunpack1,shapeunpack1,shapeunpack2,tabel1);
   so_temp=d_quantHarmonik(pitch,idVUV,spektral_envelope,Lan);
else
```

```
[vxc]=d_quantUV(shapevxclunpack,shapevxc2unpack,gainvxc1unpack,gainvxc2unPack,
tabel2);
                so_temp1=d_quantEksitasi(vxc,Lan);
                so_temp(1:Lan)=window.*so_temp1;
            end;
            so((i-1)*Lfr+1:i*Lfr+over) = filter(1,qlpc,sir_temp);
            so1((i-1)*Lfr+1:i*Lfr+over)= filter(a,b,so((i-1)*Lfr+1:i*Lfr+over));
            so2((i-1)*Lfr+1:i*Lfr+over)= filter(c,d,so1((i-1)*Lfr+1:i*Lfr+over));
            so3((i-1)*Lfr+1:i*Lfr+over)= filter(e,f,so2((i-1)*Lfr+1:i*Lfr+over));
            sout((i-1)*Lfr+1:i*Lfr+over)= filter(g,h,so3((i-1)*Lfr+1:i*Lfr+over));

            % 4.3 Pengujian Kualitas
            numSNR(1:160)=0;
            denSNR(1:160)=0;

            for j=1:Lfr
                numSNR(j)=si-((i-1)*Lfr+1+(over/2)+j)^2+numSNR(j);
                denSNR(j)=(si((i-1)*Lfr+1+(over/2)+j)-sout((i-
1)*Lfr+1+(over/2)+j))^2+denSNR(j);
            end;

            SNR(i)= sum(numSNR/denSNR);
        end;

        SNR_total = sum(SNR);
        RataSNR = 10*log1O(SNR_total/mfr);

        disp('selesai');
        wavwrite(sout,8000,'maranatha.wav');
    end;
end;

% ====================================================
%   Fungsi untuk menghitung magnituda window di kawasan frekuensi
% ====================================================
Function [ipc_bsamp] = bsamp(window,Lan)
w(1:Lan) = window;
w(Lan+1:Lan*8) = 0;
w = real(fft(w,Lan*8));
im = imag (fft(w,Lan*8));

for i=1:(Lan*8)
    mag(i)=(sqrt(w(i)*w(i)+im(i)*im(i)));
    if w(i) == 0
        w(i)= 0.0001;
    end;
end;
ipc_bsamp(1:Lan*4)=mag(Lan*4+1:Lan*8);
ipc_bsamp(Lan*4+1:Lan*8)=mag(1:Lan*4);

function [lsf_stabil] = stabil_lsf(lsf_curr)
function [rms,ang,re,im,lev] = freq_balance(arys,SAMPLE)

% ==================================================
% Fungsi untuk menghitung level sinyal dalam kawasan frekuensi
% ==================================================
rrr=0;
lbl= 0;
hbl= 0;
re= real(fft(arys,SAMPLE));
im = imag(fft(arys,SAMPLE));
for i=1:SAMPLE
    rrr = rrr + re(i)*re(i);
    rms(i)= sqrt (rrr);
    if re(i) == 0
        re(i) = 0.0001;
    end;
    ang(i)= atan2(im(i),re(i));
end;

for i=1:SAMPLE/4
    lbl = lbl+rms(i)^2;
end;

for i=(SAMPLE/4)+1:SAMPLE/2
    hbl = hbl+rms(i)^2;
end;
lev = ((lbl+hbl)/ (SAMPLE/2));

% ========================================
% Fungsi untuk stabilisasi koefisien lsf pada tahap
% ========================================
min_gap = 4/256;
for j=1:10
    if lsf_curr(j) < min_gap
        lsf_curr(j) = min_gap;
    end;
end;

for j=1:9
    if (lsf_curr (j+1)-lsf_curr(j)) < min_gap
        lsf_curr (j+1) = lsf_curr(j)+min_gap;
    end;
end;

for j=1:10
             if lsf_curr(j) > (pi-min_gap)
        lsf_curr(j) = pi-min_gap;
    end;
end;

for j=10:2
    if (lsf_curr(j)-lsf_curr(j-1)) < min_gap
        lsf_curr(j-1) = lsf_curr(j)-min_gap;
    end;
end;
lsf_stabil = lsf_curr;

function [lsf_stabil] = stabil_lsf(lsf_curr)


% ========================
```
% fungsi untuk mencari nilai pitch
```
% ========================
ub_r=1;
Lan = 256;
FRAKSI = 0.25;
w0= Lan*8/global_pitch;
w0_r= Lan/global_pitch;
send= global_pitch/2;
erl=0;
erh=0;

for k=1:send
    if k==1
        lb_r=1;
    else
        lb_r=ub_r+1;
    end;

    ub_r = floor(k*w0_r + (w0_r/2)+0.5);

    if ub_r >= Lan/2
        ub_r=Lan/2;
    end;

    deno = 0;
    nume2= 0;

    for j=lb_r:ub_r
        id = floor((Lan*4)+(4*j)-(k*w0)+0.5);
        if id<=0
            id=1;
        end;
        deno = deno+ipc_bsamp(id+1)^2;
        nume2= nume2+(rms(j)*ipc_bsamp(id+1));
    end;

    if deno <= 0
        am(k) = 0;
    else
        am(k) = nume2/deno;
    end;

    for j = lb_r:ub_r
        id = floor((Lan*4)+(8*j)-(k*w0)+0.5);
        if id <= 0
```

```
            id=1;
        end;
        re = rms(j)-am(k)*ipc_bsamp(id+1);
    end;

    if j >= 50
        erl = erl+ re*re;
    else
        erh = erh + re*re;
    end;
end;

initpch = currPrm(1);

if initpch < 20
    initpch = 20;
end;

if initpch <= 148
    pitch = initpch - 1;
    finalpitch = pitch;
    for k=1:3
        [erl,erh] = Am_reduce(pitch,w0,w0_r,rms,erl,erh,ipc_bsamp);
        eubfine = (erl + erh);
        if (eubfine < mineub) | k==1
            %hal 102
            mineub = eubfine;
            min_erh = erh;
            min_erl = erl;
            finalpitch = pitch;
        end;
        pitch = pitch+1;
    end;
    pitch = finalpitch - 3*FRAKSI;
    for k=1:7
        if k == 3
            eubfine = min_erh;
        else
            [erl,erh] = Am_reduce(pitch,w0,w0_r, rms,erl,erh,ipc_bsamp);
            eubfine = erh;
        end;
        if (eubfine <= mineub)
            mineub = eubfine;
            finalpitch = pitch;
        end;
        pitch = pitch+FRAKSI;
    end;
else
    finalpitch = initpch;
end;
finalpitch < 20
finalpitch = 20;
function [erl,erh] = Am_reduce(pitch,w0,w0_r,rms,erl,erh,ipc_bsamp)
```

```
% ===================================
% Fungsi untuk pengurangan amplituda sinyal
% ===================================
SAMPLE = 256; R = 8;
ub_r = 1;
send = floor(pitch/2); %jumlah harmonik
for i=1:send-1
    if i==1
        lb_r=1;
    else
        lb_r=ub_r+1;
    end;

    ub_r = ceil((i*w0_r)+w0_r/2+0.5);

    if ub_r >= SAMPLE/2
        ub_r=ceil(SAMPLE/2 -1);
    end;
        deno = 0; nume2= 0;

    if ub_r>floor((SAMPLE*R/2)+ R*ub_r - i* w0 )
        break;

        for j=lb_r:ub_r;
            id = floor((SAMPLE*R/2)+ R*j - i*w0);
            if id<=0
                id=1;
            end;
            deno = deno+(ipc_bsamp(abs(id))^2);
            nume2= nume2+rms (j)*ipc_bsamp(id);
        end;
    end;

    if deno <= 0
        am(i) = 0;
    else
        am(i) = nume2/deno;
    end;

    for j=lb_r:ub_r
      id=floor((SAMPLE*R/2)+R*j - i*w0 +0.5);
      if id<=0
          id=1;
      end;
      re=rms(j)-am(i)*ipc_bsamp(id);
      if j >= 50
          erl = erl+re*re;
      else
          erh = erh+re*re;
      end;
   end;
end;
```