LAMPIRAN

% MAIN.m

```
clear,close all,clc;

%Variabel Modifikasi
sigma=0.12;
tipe='S';    % 'S' untuk Softthreshold dan 'H' untuk Hardthreshold
lev=1;
qmf=makeonfilter('Symmlet',5);
init=1603198200;

%Load Data 16265 Sebanyak 4096
load data1;
t1=data65(1:4096,1).';
sig065=data65(1:4096,2).';
sig165=data65(1:4096,3).';

%Load Data 16272 Sebanyak 4096
load data2;
t2=data72(1:4096,1).';
sig072=data72(1:4096,2).';
sig172=data72(1:4096,3).';

%Load Data 16273 Sebanyak 4096
load data3;
t3=data73(1:4096,1).';
sig073=data73(1:4096,2).';
sig173=data73(1:4096,3).';
```

```
%Load Data 16420 Sebanyak 4096
load data4;

t4=data20(1:4096,1).';
sig020=data20(1:4096,2).';
sig120=data20(1:4096,3).';


%Tambah Noise pada Sinyal Data
nsig065=addgwn(sig065,sigma,init);
nsig165=addgwn(sig165,sigma,init);
nsig072=addgwn(sig072,sigma,init);
nsig172=addgwn(sig172,sigma,init);
nsig073=addgwn(sig073,sigma,init);
nsig173=addgwn(sig173,sigma,init);
nsig020=addgwn(sig020,sigma,init);
nsig120=addgwn(sig120,sigma,init);

%Hitung SNR Sebelum Denoising
s065=snr(sig065,nsig065);
s165=snr(sig165,nsig165);
s072=snr(sig072,nsig072);
s172=snr(sig172,nsig172);
s073=snr(sig073,nsig073);
s173=snr(sig173,nsig173);
s020=snr(sig020,nsig020);
s120=snr(sig120,nsig120);

%Proses Denoising
warning off MATLAB:fmins:ObsoleteFunction;
d065=reccv(nsig065,tipe,lev, qmf);
d165=reccv(nsig165,tipe,lev, qmf);
```

```
d072=reccv(nsig072,tipe,lev, qmf);
d172=reccv(nsig172,tipe,lev, qmf);
d073=reccv(nsig073,tipe,lev, qmf);
d173=reccv(nsig173,tipe,lev, qmf);
d020=reccv(nsig020,tipe,lev, qmf);
d120=reccv(nsig120,tipe,lev, qmf);

%Hitung SNR Sesudah Denoising
r065=snr(sig065,d065);
r165=snr(sig165,d165);
r072=snr(sig072,d072);
r172=snr(sig172,d172);
r073=snr(sig073,d073);
r173=snr(sig173,d173);
r020=snr(sig020,d020);
r120=snr(sig120,d120);

%Hitung RSE Sesudah Denoising
rs065=rse(sig065,d065);
rs165=rse(sig165,d165);
rs072=rse(sig072,d072);
rs172=rse(sig172,d172);
rs073=rse(sig073,d073);
rs173=rse(sig173,d173);
rs020=rse(sig020,d020);
rs120=rse(sig120,d120);

figure,subplot(4,1,1),plot(t1,nsig065),axis([0 4 min(nsig065) max(nsig065)]),title(['sinyal ternoise 065, SNR= ',num2str(s065)]),...
   subplot(4,1,2),plot(t1,d065),axis([0 4 min(d065) max(d065)]),title(['sinyal denoising 065, SNR= ',num2str(r065),' RSE= ',num2str(rs065)]),...
```

```
   subplot(4,1,3),plot(t1,nsig165),axis([0 4 min(nsig165) max(nsig165)]),title(['sinyal ternoise 165, SNR= ',num2str(s165)]),...
   subplot(4,1,4),plot(t1,d165),axis([0 4 min(d165) max(d165)]),title(['sinyal denoising 165, SNR= ',num2str(r165),' RSE= ',num2str(rs165)]);

figure,subplot(4,1,1),plot(t1,nsig072),axis([0 4 min(nsig072) max(nsig072)]),title(['sinyal ternoise 072, SNR= ',num2str(s072)]),...
   subplot(4,1,2),plot(t1,d072),axis([0 4 min(d072) max(d072)]),title(['sinyal denoising 072, SNR= ',num2str(r072),' RSE= ',num2str(rs072)]),...
   subplot(4,1,3),plot(t1,nsig172),axis([0 4 min(nsig172) max(nsig172)]),title(['sinyal ternoise 172, SNR= ',num2str(s172)]),...
   subplot(4,1,4),plot(t1,d172),axis([0 4 min(d172) max(d172)]),title(['sinyal denoising 172, SNR= ',num2str(r172),' RSE= ',num2str(rs172)]);

figure,subplot(4,1,1),plot(t1,nsig073),axis([0 4 min(nsig073) max(nsig073)]),title(['sinyal ternoise 073, SNR= ',num2str(s073)]),...
   subplot(4,1,2),plot(t1,d073),axis([0 4 min(d073) max(d073)]),title(['sinyal denoising 073, SNR= ',num2str(r073),' RSE= ',num2str(rs073)]),...
   subplot(4,1,3),plot(t1,nsig173),axis([0 4 min(nsig173) max(nsig173)]),title(['sinyal ternoise 173, SNR= ',num2str(s173)]),...
   subplot(4,1,4),plot(t1,d173),axis([0 4 min(d173) max(d173)]),title(['sinyal denoising 173, SNR= ',num2str(r173),' RSE= ',num2str(rs173)]);

figure,subplot(4,1,1),plot(t1,nsig020),axis([0 4 min(nsig020) max(nsig020)]),title(['sinyal ternoise 020, SNR= ',num2str(s020)]),...
   subplot(4,1,2),plot(t1,d020),axis([0 4 min(d020) max(d020)]),title(['sinyal denoising 020, SNR= ',num2str(r020),' RSE= ',num2str(rs020)]),...
   subplot(4,1,3),plot(t1,nsig120),axis([0 4 min(nsig120) max(nsig120)]),title(['sinyal ternoise 120, SNR= ',num2str(s120)]),...
   subplot(4,1,4),plot(t1,d120),axis([0 4 min(d120) max(d120)]),title(['sinyal denoising 120, SNR= ',num2str(r120),' RSE= ',num2str(rs120)]);
```

RECCV

```
function f = reccv(signal,type,lev, h)

% recgcv:    Smoothing dipergunakan secara umum dalam tujuan cross validation.
%           Dikenal dengan prosedur CV.
%            f = reccv(signal,type,lev,h)
% Masukan
%   signal        1-d Noisy signal, length(signal)= 2^J
%   type            'S' untuk soft thresholding, 'H' untuk hard thresholding
%   h              Quadrature Mirror Filter untuk transformasi wavelet.
%                       Optional, default = Symmlet 8
% Keluaran
%   f               Estimasi, berisi nilai thresholding yang digunakan dalam
wavelet
%if nargin < 3,
        %h = MakeONFilter('Symmlet',8);
   %end

%inisialisasi
n=length(signal);
%lev=floor(log2(log(n)))+1;
J=log2(n);

%Normalisation of the noise level to 1
[signalnorm,coef] = NormNoise(signal,h);

%Determining the cross-validated threshold
thr=fmins('cv',sqrt(2*log(n)),[],[],signalnorm,lev,h,type);
thr=thr/sqrt((1-log(2)/log(n)));

%Extraction of the wavelet coefficients
wcoef=FWT_PO(signalnorm,lev,h);
```

```
if strcmp(type,'H'),
  wcoef(2^lev+1:2^J) = HardThresh(wcoef(2^lev+1:2^J),thr);
else
  wcoef(2^lev+1:2^J) = SoftThresh(wcoef(2^lev+1:2^J),thr);
end
reconstruct=IWT_PO(wcoef,lev,h);
f = (1/coef)*reconstruct;
```

RSE

```
function value=RSE(sig1,sig2)
%  MSE - Reconstruction Square Error
%  Pemakaian
%    value=RSE(sig1,sig2)
%  Masukan
%    sig1        Signal Referensi.
%    sig2        Signal Hasil Restorasi, Estimasi atau Denoising.
%  Keluaran
%    value       Nilai Reconstruction Square Error.
n=length(sig1);
value=sum((sig1-sig2).^2);
```

SNR

```
function value=SNR(sig1,sig2)
%  SNR- Signal/Noise ratio
%  Pemakaian
%    value=SNR(sig1,sig2)
%  Masukan
%    sig1        Sinyal Sebernarnya
%    sig2        Sinyal Noise
%  Keluaran
%    value       Signal/Noise ratio.
value=20*log10(norm(sig1)/norm(sig1-sig2));
```

SoftThresh

```
function x = SoftThresh(y,t)
% SoftThresh – Soft Threshold pada koefisien wavelet
%  Pemakaian
```

```
%    x = SoftThresh(y,t)
%  Masukan
%    y     Noisy Data
%    t     Threshold
%  Keluaran
%    x     sign(y)(|y|-t)_+
%
          res = (abs(y) - t);
          res = (res + abs(res))/2;
          x   = sign(y).*res;
```

HardThresh

```
function x = HardThresh(y,t)
% HardThresh -- Apply Hard Threshold
%  Pemakaian
%    x = HardThresh(y,t)
%  Masukan
%    y     Noisy Data
%    t     Threshold
%  Keluaran
%    x     y 1_{|y|>t}
%
          x   = y .* (abs(y) > t);
```

RSE

```
function value=RSE(sig1,sig2)
%  RSE – Reconstruction Square Error
%  Pemakaian
%    value=RSE(sig1,sig2)
%  Masukan
%    sig1          Signal Referensi.
%    sig2          Signal Hasil Restorasi, Estimasi atau Denoising.
%  Keluaran
%    value         Nilai Reconstruction Square Error.

n=length(sig1);
value=sum((sig1-sig2).^2);
```

NormNoise

```
function [y,coef] = NormNoise(x,qmf)
% NormNoise – Estimasi level noise, Normalize signal to noise level 1
%  Pemakaian
%    [y,coef] = NormNoise(x,qmf)
%  Masukan
%    x     1-d signal
%    qmf   quadrature mirror filter
```

```
%  Keluaran
%    y     1-d signal, scaled so wavelet coefficients
%          at finest level have median absolute deviation 1.
%    coef  estimation of 1/sigma
%
%  Keterangan
%    This is required pre-processing to use any of the DeNoising
%    tools on naturally-occurring data.
%  Perhatikan
%    WaveShrink, CPDeNoise,WPDeNoise
%
        u = DownDyadHi(x,qmf);
        s = median(abs(u));
        if s ~= 0
                y =  .6745 .* x ./s;
           coef = .6745 /s;
        else
                y = x;
           coef = 1;
        end
```

DownDyadHi

```
function d = DownDyadHi(x,qmf)
% DownDyadHi -- Hi-Pass Downsampling operator (periodized)
%  Pemakain
%    d = DownDyadHi(x,f)
%  Masukan
%    x    1-d signal at fine scale
%    f    filter
%  Keluaran
%    y    1-d signal at coarse scale
%
%  Perhatikan juga
%    DownDyadLo, UpDyadHi, UpDyadLo, FWT_PO, iconv
%
        d = iconv( MirrorFilt(qmf),lshift(x));
        n = length(d);
        d = d(1:2:(n-1));
```

DownDyadLo

```
function d = DownDyadLo(x,qmf)
% DownDyadLo -- Lo-Pass Downsampling operator (periodized)
%  Pemakaian
%    d = DownDyadLo(x,f)
%  Masukan
%    x    1-d signal at fine scale
%    f    filter
```

```
%  Keluaran
%    y    1-d signal at coarse scale
%
%  Perhatikan juga
%    DownDyadHi, UpDyadHi, UpDyadLo, FWT_PO, aconv
%
          d = aconv(qmf,x);
          n = length(d);
          d = d(1:2:(n-1));
```

Dyad.m

```
function i = dyad(j)
% dyad -- Index entire j-th dyad of 1-d wavelet xform
%  Pemakaian
%    ix = dyad(j);
% Masukan
%    j     integer
%  Keluaran
%    ix    list of all indices of wavelet coeffts at j-th level
%
   i = (2^(j)+1):(2^(j+1)) ;
```

Dyadlength

```
function [n,J] = dyadlength(x)
% dyadlength -- Find length and dyadic length of array
%  Pemakaian
%    [n,J] = dyadlength(x)
% Masukan
%    x    array of length n = 2^J (hopefully)
%  Keluaran
%    n    length(x)
%    J    least power of two greater than n
%
%  Efek Samping
%    A warning is issued if n is not a power of 2.
%
%  Perhatikan juga
%    quadlength, dyad, dyad2ix
%
 n = length(x) ;
 J = ceil(log(n)/log(2));
 if 2^J ~= n ,
    disp('Warning in dyadlength: n != 2^J')
 end
```

FWT_PO

```
function wcoef = FWT_PO(x,L,qmf)
```

```
% FWT_PO -- Forward Wavelet Transform (periodized, orthogonal)
%  Pemakaian
%    wc = FWT_PO(x,L,qmf)
%  Masukan
%    x    1-d signal; length(x) = 2^J
%    L    Coarsest Level of V_0;  L << J
%    qmf  quadrature mirror filter (orthonormal)
%  Keluaran
%    wc    1-d wavelet transform of x.
%
%  Keterangan
%    1. qmf filter may be obtained from MakeONFilter
%    2. usually, length(qmf) < 2^(L+1)
%    3. To reconstruct use IWT_PO
%
%  Perhatikan juga
%    IWT_PO, MakeONFilter
%
 [n,J] = dyadlength(x) ;
 wcoef = zeros(1,n) ;
 beta = ShapeAsRow(x);  %take samples at finest scale as beta-coeffts
 for j=J-1:-1:L
     alfa = DownDyadHi(beta,qmf);
     wcoef(dyad(j)) = alfa;
     beta = DownDyadLo(beta,qmf) ;
 end
 wcoef(1:(2^L)) = beta;
 wcoef = ShapeLike(wcoef,x);
```

AddGWN

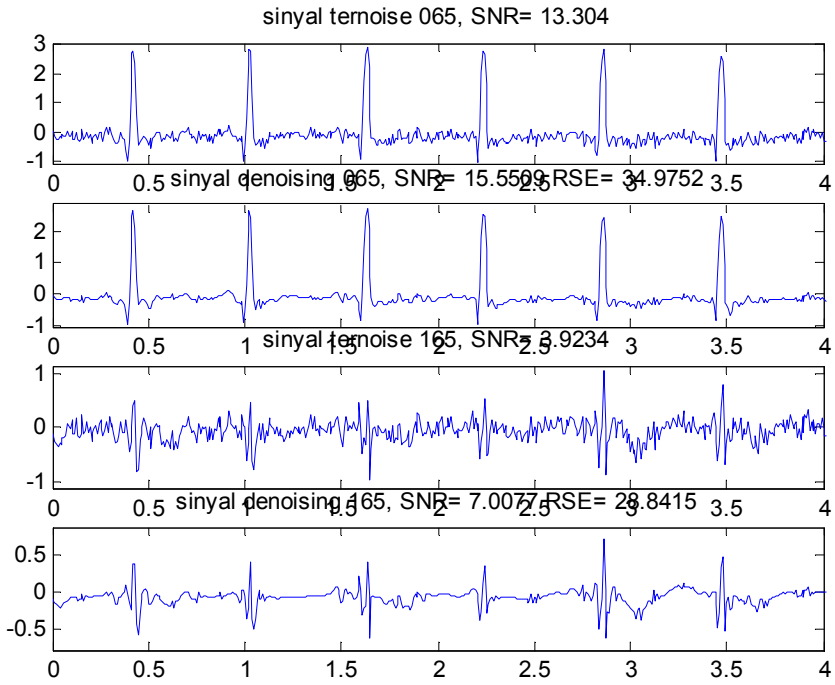
```
function [Noisysig] = AddGWN(data,sigma,init,snr)
%  AddGWN - Penambahan White Gaussian Noise pada Sinyal Input
%  Pemakaian
%    Noisysig = AddGWN(sig,sigma,init)
%  Masukan
%    sig        Sinyal Input
%    sigma      Standar Deviasi dari Additive White Gaussian Noise
%    init       Generator Seed untuk AWGN
%  Keluaran
%    Noisysig   Sinyal 1-d yang Telah Memiliki Noise
%

n=length(data);
if nargin >2
   randn('seed',init);
```

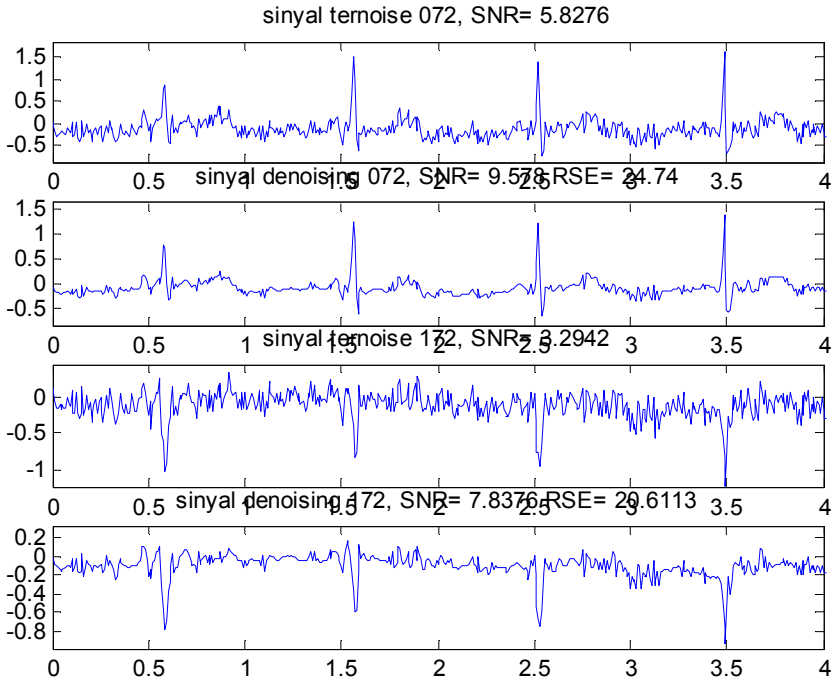
```
end

if nargin >3
   data=(data * (snr/std(data)));
end

Noise= sigma * randn(1,n);
Noisysig= Noise+data;
```

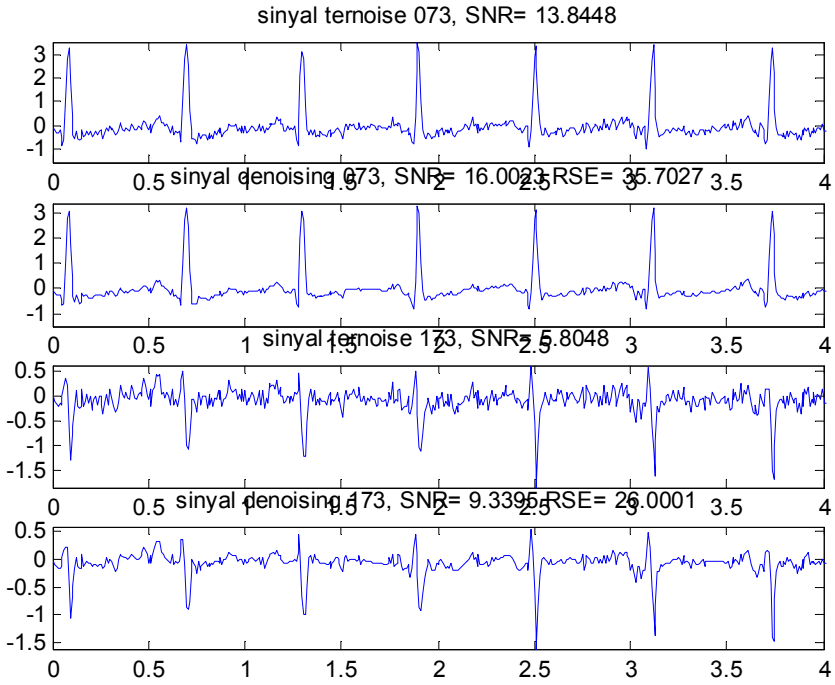
CV

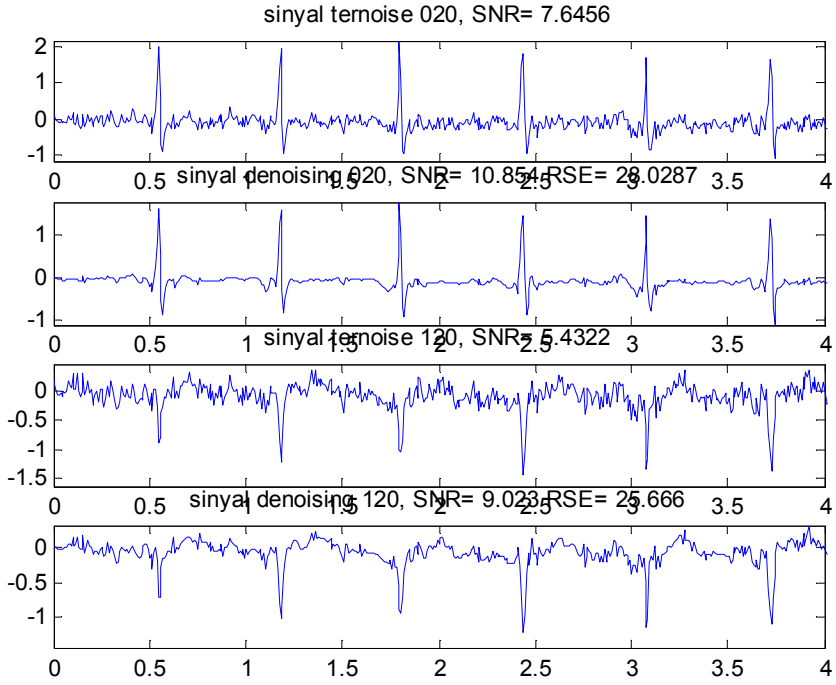
```
function M = cv(thr,x,L,h,type)

% gcv: Tujuan umum dari cross validation yang dibutuhkan dengan menggunakan
prosedur RECCV.
% Pemakaian
%        M = cv(thr,x,L,h,type)
% Masukan
%   thr   Threshold
%   x              1-d Noisy signal, length(signal)= 2^J
%   L              Low resolution level
%   h              Quadrature Miror Filter
%   type           'S' for soft thresholding, 'H' for hard thresholding
% Keluaran
%   M  Reconstruction Square Error.

if (L < 0)
    fprintf('Warning: primary level must be => 0\n');
    M = NaN;
    return;
end

% Inisialisasi
n=length(x);
n1=n/2;
J=log2(n);

% Pemilihan setengah dari nilai-nilai data
xodd=[];
xeven=[];
for k=1:n1,
  xodd=[xodd; x(2*k-1)];
  xeven=[xeven; x(2*k)];
end;

% Performa WT
wcodd=FWT_PO(xodd,L,h);
```

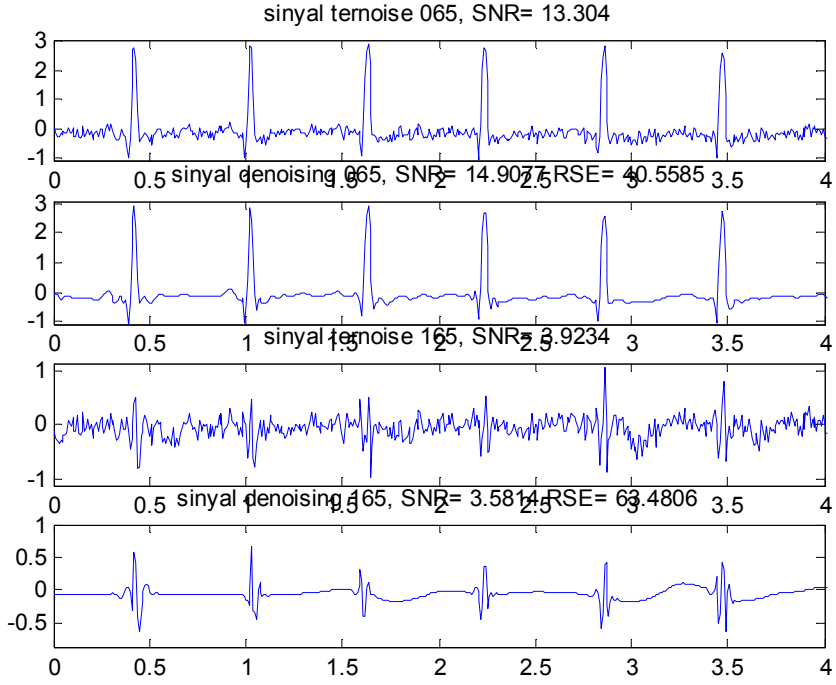
```
wceven=FWT_PO(xeven,L,h);

if strcmp(type,'H'),
  wcodd(2^L+1:n1) = HardThresh(wcodd(2^L+1:n1),thr);
  wceven(2^L+1:n1) = HardThresh(wceven(2^L+1:n1),thr);
else
  wcodd(2^L+1:n1) = SoftThresh(wcodd(2^L+1:n1),thr);
  wceven(2^L+1:n1) = SoftThresh(wceven(2^L+1:n1),thr);
end
hatxodd = IWT_PO(wcodd,L,h);
hatxeven = IWT_PO(wceven,L,h);

% Interpolating
barxodd = 0.5.*(hatxodd(1:n1-1) + hatxodd(2:n1));
barxodd(n1) = hatxodd(1);
barxeven = 0.5.*(hatxeven(1:n1-1) + hatxeven(2:n1));
barxeven(n1) = hatxeven(1);

M=norm(xodd-barxeven,'fro')^2+norm(xeven-barxodd,'fro')^2;
```

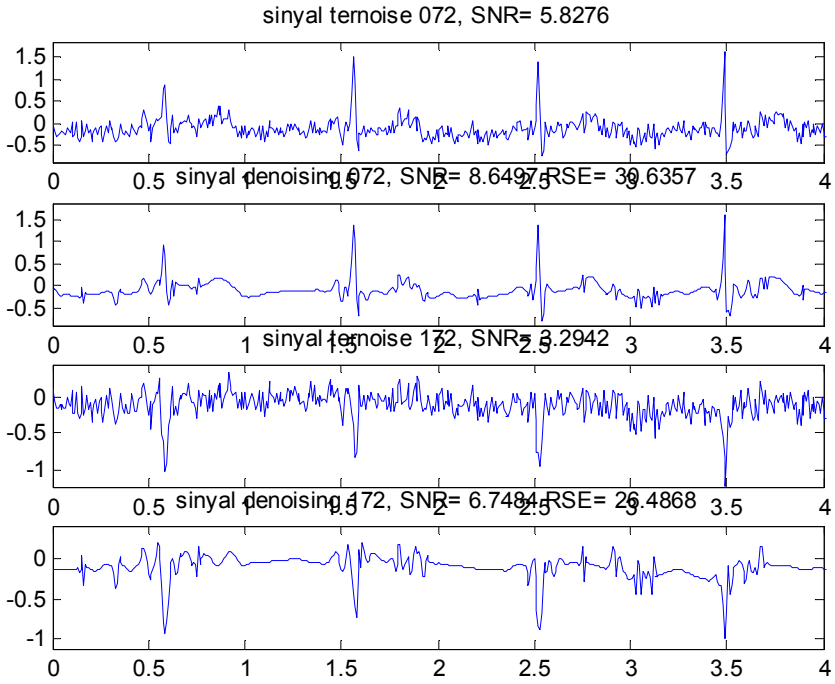


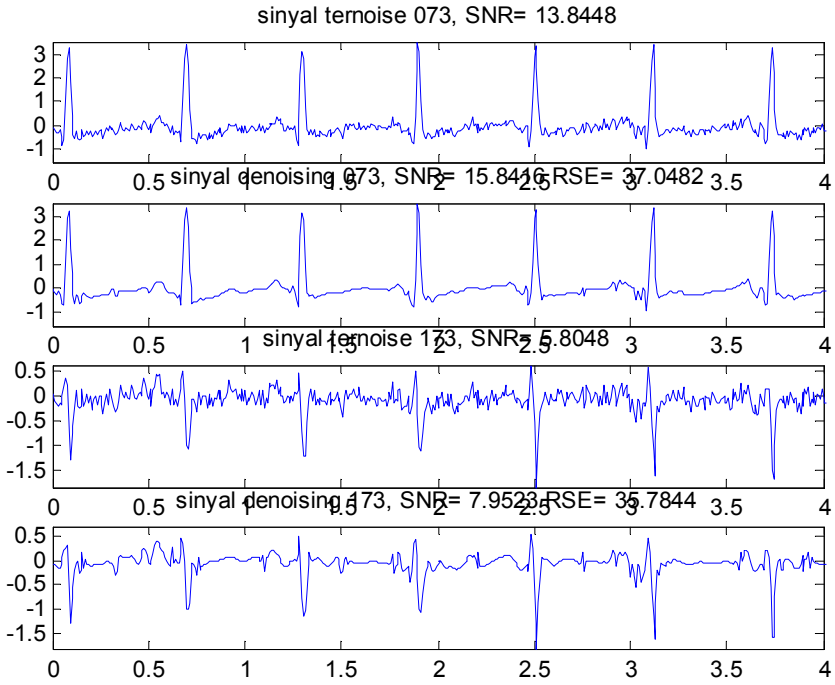

sinyal ternoise 072, SNR= 5.8276
sinyal denoising 072, SNR= 9.578, RSE= 24.74
sinyal ternoise 172, SNR= 3.2942
sinyal denoising 172, SNR= 7.8376, RSE= 29.6113

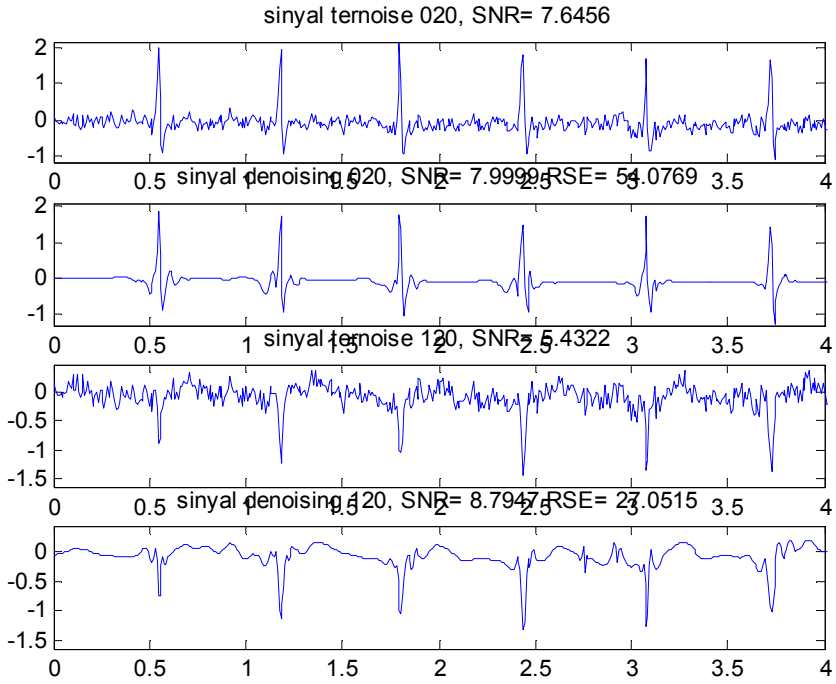

sinyal ternoise 073, SNR= 13.8448
sinyal denoising 073, SNR= 16.0023, RSE= 35.7027
sinyal ternoise 173, SNR= 5.8048
sinyal denoising 173, SNR= 9.3395, RSE= 26.0001


sinyal ternoise 020, SNR= 7.6456
sinyal denoising 020, SNR= 10.854, RSE= 28.0287
sinyal ternoise 120, SNR= 5.4322
sinyal denoising 120, SNR= 9.023, RSE= 25.666


sinyal ternoise 065, SNR= 13.304
sinyal denoising 065, SNR= 14.9077, RSE= 40.5585
sinyal ternoise 165, SNR= 3.9234
sinyal denoising 165, SNR= 3.5814, RSE= 63.4806


sinyal ternoise 072, SNR= 5.8276
sinyal denoising 072, SNR= 8.6497, RSE= 30.6357
sinyal ternoise 172, SNR= 3.2942
sinyal denoising 172, SNR= 6.7484, RSE= 26.4868


sinyal ternoise 073, SNR= 13.8448
sinyal denoising 073, SNR= 15.8416, RSE= 37.0482
sinyal ternoise 173, SNR= 5.8048
sinyal denoising 173, SNR= 7.9523, RSE= 35.7844


sinyal ternoise 020, SNR= 7.6456
sinyal denoising 020, SNR= 7.9999, RSE= 54.0769
sinyal ternoise 120, SNR= 5.4322
sinyal denoising 120, SNR= 8.7947, RSE= 27.0515