LAMPIRAN

```
function [A, W] = fpica(X, whiteningMatrix, dewhiteningMatrix, approach, ...
                        numOfIC, g, finetune, a1, a2, myy, stabilization, ...
                        epsilon, maxNumIterations, maxFinetune, initState, ...
                        guess, sampleSize, displayMode, displayInterval, ...
                        s_verbose);
%FPICA - Fixed point ICA. Algoritma utama FASTICA.
%
% [A, W] = fpica(whitesig, whiteningMatrix, dewhiteningMatrix, approach,
%       numOfIC, g, finetune, a1, a2, mu, stabilization, epsilon,
%       maxNumIterations, maxFinetune, initState, guess, sampleSize,
%       displayMode, displayInterval, verbose);
%
% Menampilkan independent component analysis menggunakan algoritma Hyvarinen's
fixed point
%. Hasil adalah matriks A dan inversnya W.
%
% whitesig                      :data whitening
% whiteningMatrix                   :dapat didapat dari function whitenv
% dewhiteningMatrix                 :didapat dari function whitenv
% approach      [ 'symm' | 'defl' ]     :pendekatan yang digunakan (deflation or symmetric)
% numOfIC       [ 0 - Dim of whitesig ] :penjumlahan independent component
% g             [ 'pow3' | 'tanh' |     :nonlinearity
%               'gaus' | 'skew' ]
% finetune      [same as g + 'off']     :nonlinearity digunakan pada finetuning.
% a1                            :parameter 'tanh'
% a2                            :parameter 'gaus'
% mu                            :step size
% stabilization [ 'on' | 'off' ]        :if mu < 1 then automatically on
% epsilon                       :stopping criterion
% maxNumIterations                  :jumlah maksimal iterasi
% maxFinetune                       :jumlah maksimal iterasi untuk finetuning
% initState     [ 'rand' | 'guess' ]    : initial state.
%                               :initial untuk A. diabaikan bila initState = 'rand'
% sampleSize    [ 0 - 1 ]               :presentasi sample yang digunakan untuk iterasi
% displayMode   [ 'signals' | 'basis' | :plot running
%               'filters' | 'off' ]
% displayInterval                   :jumlah iterasi yang di plot
% verbose       [ 'on' | 'off' ]        :menampilkan hasil
%
% CONTOH
%       [E, D] = pcamat(vectors);
%       [nv, wm, dwm] = whitenv(vectors, E, D);
%       [A, W] = fpica(nv, wm, dwm);
%
%
% Fungsi ini dibutuhkan oleh FASTICA dan FASTICAG
```

```
%
%   Lihat juga FASTICA, FASTICAG, WHITENV, PCAMAT

% @(#)$Id: fpica.m,v 1.7 2005/06/16 12:52:55 jarmo Exp $

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Global variable untuk menghentikan perhitungan  ICA dari GUI
global g_FastICA_interrupt;
if isempty(g_FastICA_interrupt)
  clear global g_FastICA_interrupt;
  interruptible = 0;
else
  interruptible = 1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Default values

if nargin < 3, error('Not enough arguments!'); end
[vectorSize, numSamples] = size(X);
if nargin < 20, s_verbose = 'on'; end
if nargin < 19, displayInterval = 1; end
if nargin < 18, displayMode = 'on'; end
if nargin < 17, sampleSize = 1; end
if nargin < 16, guess = 1; end
if nargin < 15, initState = 'rand'; end
if nargin < 14, maxFinetune = 100; end
if nargin < 13, maxNumIterations = 1000; end
if nargin < 12, epsilon = 0.0001; end
if nargin < 11, stabilization = 'on'; end
if nargin < 10, myy = 1; end
if nargin < 9, a2 = 1; end
if nargin < 8, a1 = 1; end
if nargin < 7, finetune = 'off'; end
if nargin < 6, g = 'pow3'; end
if nargin < 5, numOfIC = vectorSize; end     % vectorSize = Dim
if nargin < 4, approach = 'defl'; end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Memeriksa data

if ~isreal(X)
  error('Input memiliki bagian imajiner.');
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Memeriksa nilai verbose

switch lower(s_verbose)
 case 'on'
  b_verbose = 1;
 case 'off'
  b_verbose = 0;
 otherwise
  error(sprintf('Illegal value [ %s ] for parameter: "verbose"\n', s_verbose));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Memeriksa pendeketan

switch lower(approach)
 case 'symm'
  approachMode = 1;
 case 'defl'
  approachMode = 2;
 otherwise
  error(sprintf('Illegal value [ %s ] for parameter: "approach"\n', approach));
end
if b_verbose, fprintf('Used approach [ %s ].\n', approach); end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Memeriksa nilai numOfIC

if vectorSize < numOfIC
  error('Must have numOfIC <= Dimension!');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Memeriksa sampleSize
if sampleSize > 1
  sampleSize = 1;
  if b_verbose
    fprintf('Warning: Setting "sampleSize" to 1.\n');
  end
elseif sampleSize < 1
```

```
  if (sampleSize * numSamples) < 1000
    sampleSize = min(1000/numSamples, 1);
    if b_verbose
      fprintf('Warning: Setting "sampleSize" to %0.3f (%d samples).\n', ...
              sampleSize, floor(sampleSize * numSamples));
    end
  end
end
if b_verbose
  if  b_verbose & (sampleSize < 1)
    fprintf('Menggunakan %0.0f%% dari sample random
 step.\n',sampleSize*100);
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Memeriksa nilai nonlinearity.

switch lower(g)
 case 'pow3'
  gOrig = 10;
 case 'tanh'
  gOrig = 20;
 case {'gaus', 'gauss'}
  gOrig = 30;
 case 'skew'
  gOrig = 40;
 otherwise
  error(sprintf('Illegal value [ %s ] for parameter: "g"\n', g));
end
if sampleSize ~= 1
  gOrig = gOrig + 2;
end
if myy ~= 1
  gOrig = gOrig + 1;
end

if b_verbose,
  fprintf('Used nonlinearity [ %s ].\n', g);
end

finetuningEnabled = 1;
switch lower(finetune)
 case 'pow3'
  gFine = 10 + 1;
```

```
 case 'tanh'
  gFine = 20 + 1;
 case {'gaus', 'gauss'}
  gFine = 30 + 1;
 case 'skew'
  gFine = 40 + 1;
 case 'off'
  if myy ~= 1
    gFine = gOrig;
  else
    gFine = gOrig + 1;
  end
  finetuningEnabled = 0;
 otherwise
  error(sprintf('Illegal value [ %s ] for parameter: "finetune"\n', ...
                finetune));
end

if b_verbose & finetuningEnabled
  fprintf('Finetuning enabled (nonlinearity: [ %s ]).\n', finetune);
end

switch lower(stabilization)
 case 'on'
  stabilizationEnabled = 1;
 case 'off'
  if myy ~= 1
    stabilizationEnabled = 1;
  else
    stabilizationEnabled = 0;
  end
 otherwise
  error(sprintf('Illegal value [ %s ] for parameter: "stabilization"\n', ...
                stabilization));
end

if b_verbose & stabilizationEnabled
  fprintf('Using stabilized algorithm.\n');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parameter lainnya
myyOrig = myy;
% Ketika memulai fine-tuning set myy = myyK * myy
myyK = 0.01;
```

```
% Beberapa langkah untuk menkonvergen.
failureLimit = 5;


usedNlinearity = gOrig;
stroke = 0;
notFine = 1;
long = 0;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Memeriksa nilai initial state.

switch lower(initState)
 case 'rand'
  initialStateMode = 0;
 case 'guess'
  if size(guess,1) ~= size(whiteningMatrix,2)
    initialStateMode = 0;
    if b_verbose
      fprintf('Warning: ukuran initial state salah. Masukan initial secara random.\n');
    end
  else
    initialStateMode = 1;
    if size(guess,2) < numOfIC
      if b_verbose
            fprintf('Warning: Initial state hanya %d komponen. Menggunakan initial random
yang lain.\n', size(guess,2));
      end
      guess(:, size(guess, 2) + 1:numOfIC) = ...
                                                            rand(vectorSize,numOfIC-size(guess,2))-.5;
    elseif size(guess,2)>numOfIC
      guess=guess(:,1:numOfIC);
      fprintf('Warning: Initial state terlalu besar. Colom dibuang.\n');
    end
    if b_verbose, fprintf(‘Mengunakan initial state.\n'); end
  end
 otherwise
  error(sprintf('Illegal value [ %s ] untuk parameter: "initState"\n', initState));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Memeriksa nilai display mode.
```

```
switch lower(displayMode)
 case {'off', 'none'}
  usedDisplay = 0;
 case {'on', 'signals'}
  usedDisplay = 1;
  if (b_verbose & (numSamples > 10000))
    fprintf('Warning: Data vectors terlalu panjang. Plotting membutuhkan waktu yang
lama.\n');
  end
  if (b_verbose & (numOfIC > 25))
    fprintf('Warning: Terlalu banyak sinyal yang diplot. Plot akan terlihat jelek.\n');
  end
 case 'basis'
  usedDisplay = 2;
  if (b_verbose & (numOfIC > 25))
    fprintf('Warning: Terlalu banyak sinyal yang diplot. Plot akan terlihat jelek.\n');
  end
 case 'filters'
  usedDisplay = 3;
  if (b_verbose & (vectorSize > 25))
    fprintf('Warning: Terlalu banyak sinyal yang diplot. Plot akan terlihat jelek.\n');
  end
 otherwise
  error(sprintf('Illegal value [ %s ] untuk parameter: "displayMode"\n', displayMode));
end

% The displayInterval tidak boleh kurang dari 1...
if displayInterval < 1
  displayInterval = 1;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if b_verbose, fprintf('Starting ICA calculation...\n'); end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SYMMETRIC APPROACH
if approachMode == 1,

  % set beberapa parameter yang lain...
  usedNlinearity = gOrig;
  stroke = 0;
  notFine = 1;
  long = 0;
```

```
A = zeros(vectorSize, numOfIC);  % Dewhitened basis vectors.
if initialStateMode == 0
  % ambil orthonormal initial vectors secara random.
  B = orth (randn (vectorSize, numOfIC));
elseif initialStateMode == 1
  % Gunakan initial vector sebagai initial state
  B = whiteningMatrix * guess;
end

BOld = zeros(size(B));
BOld2 = zeros(size(B));

% actual fixed-point iteration loop.
for round = 1:maxNumIterations + 1,
  if round == maxNumIterations + 1,
    fprintf('No convergence after %d steps\n', maxNumIterations);
    fprintf(Plot kemungkinan salah.\n');
    if ~isempty(B)
        % Symmetric orthogonalization.
        B = B * real(inv(B' * B)^(1/2));

        W = B' * whiteningMatrix;
        A = dewhiteningMatrix * B;
    else
        W = [];
        A = [];
    end
    return;
  end

  if (interruptible & g_FastICA_interrupt)
    if b_verbose
      fprintf('\n\nCalculation interrupted by the user\n');
    end
    if ~isempty(B)
        W = B' * whiteningMatrix;
        A = dewhiteningMatrix * B;
    else
        W = [];
        A = [];
    end
    return;
  end
```

```
% Symmetric orthogonalization.
B = B * real(inv(B' * B)^(1/2));

% Test untuk kondisi termination
minAbsCos = min(abs(diag(B' * BOld)));
minAbsCos2 = min(abs(diag(B' * BOld2)));

if (1 - minAbsCos < epsilon)
  if finetuningEnabled & notFine
    if b_verbose, fprintf('Initial convergence, fine-tuning: \n'); end;
    notFine = 0;
    usedNlinearity = gFine;
    myy = myyK * myyOrig;
    BOld = zeros(size(B));
    BOld2 = zeros(size(B));

  else
    if b_verbose, fprintf('Convergence after %d steps\n', round); end

    % Calculate the de-whitened vectors.
    A = dewhiteningMatrix * B;
    break;
  end
elseif stabilizationEnabled
  if (~stroke) & (1 - minAbsCos2 < epsilon)
        if b_verbose, fprintf('Stroke!\n'); end;
        stroke = myy;
        myy = .5*myy;
        if mod(usedNlinearity,2) == 0
          usedNlinearity = usedNlinearity + 1;
        end
  elseif stroke
        myy = stroke;
        stroke = 0;
        if (myy == 1) & (mod(usedNlinearity,2) ~= 0)
          usedNlinearity = usedNlinearity - 1;
        end
  elseif (~long) & (round>maxNumIterations/2)
        if b_verbose, fprintf('Taking long (reducing step size)\n'); end;
        long = 1;
        myy = .5*myy;
        if mod(usedNlinearity,2) == 0
          usedNlinearity = usedNlinearity + 1;
        end
  end
end
```

```
BOld2 = BOld;
BOld = B;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Menampilkan proses
if b_verbose
  if round == 1
    fprintf('Step no. %d\n', round);
  else
    fprintf('Step no. %d, change in value of estimate: %.3g \n', round, 1 - minAbsCos);
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot keadaan yang terjadi
switch usedDisplay
 case 1
  if rem(round, displayInterval) == 0,
        % There was and may still be other displaymodes...
        % 1D signals
        icaplot('dispsig',(X'*B)');
        drawnow;
  end
 case 2
  if rem(round, displayInterval) == 0,
        % ... and now there are :-)
        % 1D basis
        A = dewhiteningMatrix * B;
        icaplot('dispsig',A');
        drawnow;
  end
 case 3
  if rem(round, displayInterval) == 0,
        % ... and now there are :-)
        % 1D filters
        W = B' * whiteningMatrix;
        icaplot('dispsig',W);
        drawnow;
  end
 otherwise
end

switch usedNlinearity
  % pow3
 case 10
```

```
 B = (X * (( X' * B) .^ 3)) / numSamples - 3 * B;
case 11

 Y = X' * B;
 Gpow3 = Y .^ 3;
 Beta = sum(Y .* Gpow3);
 D = diag(1 ./ (Beta - 3 * numSamples));
 B = B + myy * B * (Y' * Gpow3 - diag(Beta)) * D;
case 12
 Xsub=X(:, getSamples(numSamples, sampleSize));
 B = (Xsub * (( Xsub' * B) .^ 3)) / size(Xsub,2) - 3 * B;
case 13
 % Optimoitu
 Ysub=X(:, getSamples(numSamples, sampleSize))' * B;
 Gpow3 = Ysub .^ 3;
 Beta = sum(Ysub .* Gpow3);
 D = diag(1 ./ (Beta - 3 * size(Ysub', 2)));
 B = B + myy * B * (Ysub' * Gpow3 - diag(Beta)) * D;

 % tanh
case 20
 hypTan = tanh(a1 * X' * B);
 B = X * hypTan / numSamples - ...
        ones(size(B,1),1) * sum(1 - hypTan .^ 2) .* B / numSamples * ...
        a1;
case 21
 % optimoitu - epsilonin kokoisia
 Y = X' * B;
 hypTan = tanh(a1 * Y);
 Beta = sum(Y .* hypTan);
 D = diag(1 ./ (Beta - a1 * sum(1 - hypTan .^ 2)));
 B = B + myy * B * (Y' * hypTan - diag(Beta)) * D;
case 22
 Xsub=X(:, getSamples(numSamples, sampleSize));
 hypTan = tanh(a1 * Xsub' * B);
 B = Xsub * hypTan / size(Xsub, 2) - ...
        ones(size(B,1),1) * sum(1 - hypTan .^ 2) .* B / size(Xsub, 2) * a1;
case 23
 % Optimoitu
 Y = X(:, getSamples(numSamples, sampleSize))' * B;
 hypTan = tanh(a1 * Y);
 Beta = sum(Y .* hypTan);
 D = diag(1 ./ (Beta - a1 * sum(1 - hypTan .^ 2)));
 B = B + myy * B * (Y' * hypTan - diag(Beta)) * D;

 % gauss
```

```
case 30
 U = X' * B;
 Usquared=U .^ 2;
 ex = exp(-a2 * Usquared / 2);
 gauss =  U .* ex;
 dGauss = (1 - a2 * Usquared) .*ex;
 B = X * gauss / numSamples - ...
        ones(size(B,1),1) * sum(dGauss)...
        .* B / numSamples ;
case 31
 % optimoitu
 Y = X' * B;
 ex = exp(-a2 * (Y .^ 2) / 2);
 gauss = Y .* ex;
 Beta = sum(Y .* gauss);
 D = diag(1 ./ (Beta - sum((1 - a2 * (Y .^ 2)) .* ex)));
 B = B + myy * B * (Y' * gauss - diag(Beta)) * D;
case 32
 Xsub=X(:, getSamples(numSamples, sampleSize));
 U = Xsub' * B;
 Usquared=U .^ 2;
 ex = exp(-a2 * Usquared / 2);
 gauss =  U .* ex;
 dGauss = (1 - a2 * Usquared) .*ex;
 B = Xsub * gauss / size(Xsub,2) - ...
        ones(size(B,1),1) * sum(dGauss)...
        .* B / size(Xsub,2) ;
case 33
 % Optimoitu
 Y = X(:, getSamples(numSamples, sampleSize))' * B;
 ex = exp(-a2 * (Y .^ 2) / 2);
 gauss = Y .* ex;
 Beta = sum(Y .* gauss);
 D = diag(1 ./ (Beta - sum((1 - a2 * (Y .^ 2)) .* ex)));
 B = B + myy * B * (Y' * gauss - diag(Beta)) * D;

 % skew
case 40
 B = (X * ((X' * B) .^ 2)) / numSamples;
case 41
 % Optimoitu
 Y = X' * B;
 Gskew = Y .^ 2;
 Beta = sum(Y .* Gskew);
 D = diag(1 ./ (Beta));
 B = B + myy * B * (Y' * Gskew - diag(Beta)) * D;
```

```
    case 42
     Xsub=X(:, getSamples(numSamples, sampleSize));
     B = (Xsub * ((Xsub' * B) .^ 2)) / size(Xsub,2);
    case 43
     % Uusi optimoitu
     Y = X(:, getSamples(numSamples, sampleSize))' * B;
     Gskew = Y .^ 2;
     Beta = sum(Y .* Gskew);
     D = diag(1 ./ (Beta));
     B = B + myy * B * (Y' * Gskew - diag(Beta)) * D;

    otherwise
     error('Code for desired nonlinearity not found!');
   end
  end


  % Menghitung ICA filter
  W = B' * whiteningMatrix;

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Plot yang terakhir
  switch usedDisplay
   case 1

    % 1D signals
    icaplot('dispsig',(X'*B)');
    drawnow;
   case 2

    % 1D basis
    icaplot('dispsig',A');
    drawnow;
   case 3

    % 1D filters
    icaplot('dispsig',W);
    drawnow;
   otherwise
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pendekata
if approachMode == 2
```

```
B = zeros(vectorSize);



numFailures = 0;

while round <= numOfIC,
  myy = myyOrig;
  usedNlinearity = gOrig;
  stroke = 0;
  notFine = 1;
  long = 0;
  endFinetuning = 0;

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Menunjukan proses
  if b_verbose, fprintf('IC %d ', round); end

  % Ambil intial sembarang kemudian ortogonalkan
  % dengan mengikuti aturan
  if initialStateMode == 0
    w = randn (vectorSize, 1);
  elseif initialStateMode == 1
    w=whiteningMatrix*guess(:,round);
  end
  w = w - B * B' * w;
  w = w / norm(w);

  wOld = zeros(size(w));
  wOld2 = zeros(size(w));

  % Fixed Point
  %    for i = 1 : maxNumIterations + 1
  i = 1;
  gabba = 1;
  while i <= maxNumIterations + gabba
    if (usedDisplay > 0)
        drawnow;
    end
    if (interruptible & g_FastICA_interrupt)
      if b_verbose
        fprintf('\n\nCalculation interrupted by the user\n');
      end
      return;
    end
```

```
    w = w - B * B' * w;
    w = w / norm(w);

    if notFine
        if i == maxNumIterations + 1
          if b_verbose
            fprintf('\nComponent number %d did not converge in %d iterations.\n', round,
maxNumIterations);
          end
          round = round - 1;
          numFailures = numFailures + 1;
          if numFailures > failureLimit
            if b_verbose
              fprintf('Too many failures to converge (%d). Giving up.\n', numFailures);
            end
            if round == 0
              A=[];
              W=[];
            end
            return;
          end
          % numFailures > failurelimit
          break;
        end
        % i == maxNumIterations + 1
    else
        % if notFine
        if i >= endFinetuning
          wOld = w; % So the algorithm will stop on the next test...
        end
    end
    % if notFine

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Menunjukan hasil
    if b_verbose, fprintf('.'); end;


    if norm(w - wOld) < epsilon | norm(w + wOld) < epsilon
      if finetuningEnabled & notFine
        if b_verbose, fprintf('Initial convergence, fine-tuning: '); end;
        notFine = 0;
          gabba = maxFinetune;
        wOld = zeros(size(w));
        wOld2 = zeros(size(w));
```

```
  usedNlinearity = gFine;
  myy = myyK * myyOrig;

     endFinetuning = maxFinetune + i;

else
  numFailures = 0;
  % Save the vector
  B(:, round) = w;

  % Menghitung dewhitenev
  A(:,round) = dewhiteningMatrix * w;
  % Calculate ICA filter.
  W(round,:) = w' * whiteningMatrix;

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Menunjukan hasil
  if b_verbose, fprintf('computed ( %d steps ) \n', i); end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Plot keadaan
  switch usedDisplay
      case 1
       if rem(round, displayInterval) == 0,

         % 1D signals
         temp = X'*B;
         icaplot('dispsig',temp(:,1:numOfIC)');
         drawnow;
       end
      case 2
       if rem(round, displayInterval) == 0,
         % ... and now there are :-)
         % 1D basis
         icaplot('dispsig',A');
         drawnow;
       end
      case 3
       if rem(round, displayInterval) == 0,
         % ... and now there are :-)
         % 1D filters
         icaplot('dispsig',W);
         drawnow;
       end
  end
     break; % IC ready - next...
```

```
  end
    %if finetuningEnabled & notFine
elseif stabilizationEnabled
    if (~stroke) & (norm(w - wOld2) < epsilon | norm(w + wOld2) < ...
                    epsilon)
      stroke = myy;
      if b_verbose, fprintf('Stroke!'); end;
      myy = .5*myy;
      if mod(usedNlinearity,2) == 0
        usedNlinearity = usedNlinearity + 1;
      end
    elseif stroke
      myy = stroke;
      stroke = 0;
      if (myy == 1) & (mod(usedNlinearity,2) ~= 0)
        usedNlinearity = usedNlinearity - 1;
      end
    elseif (notFine) & (~long) & (i > maxNumIterations / 2)
      if b_verbose, fprintf('Taking long (reducing step size) '); end;
      long = 1;
      myy = .5*myy;
      if mod(usedNlinearity,2) == 0
        usedNlinearity = usedNlinearity + 1;
      end
    end
end

wOld2 = wOld;
wOld = w;

switch usedNlinearity
    % pow3
 case 10
    w = (X * ((X' * w) .^ 3)) / numSamples - 3 * w;
 case 11
    EXGpow3 = (X * ((X' * w) .^ 3)) / numSamples;
    Beta = w' * EXGpow3;
    w = w - myy * (EXGpow3 - Beta * w) / (3 - Beta);
 case 12
    Xsub=X(:,getSamples(numSamples, sampleSize));
    w = (Xsub * ((Xsub' * w) .^ 3)) / size(Xsub, 2) - 3 * w;
 case 13
    Xsub=X(:,getSamples(numSamples, sampleSize));
    EXGpow3 = (Xsub * ((Xsub' * w) .^ 3)) / size(Xsub, 2);
    Beta = w' * EXGpow3;
    w = w - myy * (EXGpow3 - Beta * w) / (3 - Beta);
```

```
    % tanh
case 20
    hypTan = tanh(a1 * X' * w);
    w = (X * hypTan - a1 * sum(1 - hypTan .^ 2)' * w) / numSamples;
case 21
    hypTan = tanh(a1 * X' * w);
    Beta = w' * X * hypTan;
    w = w - myy * ((X * hypTan - Beta * w) / ...
                (a1 * sum((1-hypTan .^2)') - Beta));
case 22
    Xsub=X(:,getSamples(numSamples, sampleSize));
    hypTan = tanh(a1 * Xsub' * w);
    w = (Xsub * hypTan - a1 * sum(1 - hypTan .^ 2)' * w) / size(Xsub, 2);
case 23
    Xsub=X(:,getSamples(numSamples, sampleSize));
    hypTan = tanh(a1 * Xsub' * w);
    Beta = w' * Xsub * hypTan;
    w = w - myy * ((Xsub * hypTan - Beta * w) / ...
                (a1 * sum((1-hypTan .^2)') - Beta));
    % gauss
case 30
    % This has been split for performance reasons.
    u = X' * w;
    u2=u.^2;
    ex=exp(-a2 * u2/2);
    gauss =  u.*ex;
    dGauss = (1 - a2 * u2) .*ex;
    w = (X * gauss - sum(dGauss)' * w) / numSamples;
case 31
    u = X' * w;
    u2=u.^2;
    ex=exp(-a2 * u2/2);
    gauss =  u.*ex;
    dGauss = (1 - a2 * u2) .*ex;
    Beta = w' * X * gauss;
    w = w - myy * ((X * gauss - Beta * w) / ...
                (sum(dGauss)' - Beta));
case 32
    Xsub=X(:,getSamples(numSamples, sampleSize));
    u = Xsub' * w;
    u2=u.^2;
    ex=exp(-a2 * u2/2);
    gauss =  u.*ex;
    dGauss = (1 - a2 * u2) .*ex;
    w = (Xsub * gauss - sum(dGauss)' * w) / size(Xsub, 2);
case 33
```

```
        Xsub=X(:,getSamples(numSamples, sampleSize));
        u = Xsub' * w;
        u2=u.^2;
        ex=exp(-a2 * u2/2);
        gauss =  u.*ex;
        dGauss = (1 - a2 * u2) .*ex;
        Beta = w' * Xsub * gauss;
        w = w - myy * ((Xsub * gauss - Beta * w) / ...
                    (sum(dGauss)' - Beta));
        % skew
    case 40
        w = (X * ((X' * w) .^ 2)) / numSamples;
    case 41
        EXGskew = (X * ((X' * w) .^ 2)) / numSamples;
        Beta = w' * EXGskew;
        w = w - myy * (EXGskew - Beta*w)/(-Beta);
    case 42
        Xsub=X(:,getSamples(numSamples, sampleSize));
        w = (Xsub * ((Xsub' * w) .^ 2)) / size(Xsub, 2);
    case 43
        Xsub=X(:,getSamples(numSamples, sampleSize));
        EXGskew = (Xsub * ((Xsub' * w) .^ 2)) / size(Xsub, 2);
        Beta = w' * EXGskew;
        w = w - myy * (EXGskew - Beta*w)/(-Beta);

    otherwise
        error('Code for desired nonlinearity not found!');
   end

   % Normalisasi w
   w = w / norm(w);
   i = i + 1;
  end
  round = round + 1;
end
if b_verbose, fprintf('Done.\n'); end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot
if (usedDisplay > 0) & (rem(round-1, displayInterval) ~= 0)
  switch usedDisplay
   case 1
    % 1D signals
    temp = X'*B;
    icaplot('dispsig',temp(:,1:numOfIC)');
    drawnow;
```

```
    case 2
     % ... and now there are :-)
     % 1D basis
     icaplot('dispsig',A');
     drawnow;
    case 3
     % ... and now there are :-)
     % 1D filters
     icaplot('dispsig',W);
     drawnow;
    otherwise
   end
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
% Cek data untuk security
if ~isreal(A)
  if b_verbose, fprintf('Warning: Memindahkan data
  A = real(A);
  W = real(W);
end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Subfunction
% Menghitung tanh linier lebih cepat dari pada Matlab tanh.
function y=tanh(x)
y = 1 - 2 ./ (exp(2 * x) + 1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Samples = getSamples(max, percentage)
Samples = find(rand(1, max) < percentage);
```