De-convolution of mixed signals

Harikrishnan Murali

Yifei Guan

University of Washington

March 2016

Table of Contents:

1. INTRODUCTION AND OVERVIEW 3
2. THEORETICAL BACKGROUND 3
3. ALGORITHM IMPLEMENTATION AND DEVELOPMENT 4
   3.1 *PREPROCESSING THE DATA* 4
   3.2 *FINDING THE MATRIX W* 5
4. RESULT AND VISUALIZATION 5
5. DE-CONVOLUTION BY PRINCIPAL COMPONENT ANALYSIS 7
6. SUMMARY AND CONCLUSION 9

APPENDIX A. SPECIAL MATLAB FUNCTION USED AND EXPLANATION 10

APPENDIX B1. MATLAB CODE FOR PERFORMING ICA 10

APPENDIX B2. MATLAB CODE FOR SVD AND PCA 12

1. Introduction and Overview

Independent Component Analysis is a technique that recovers a set of independent signals from a set of measured signals. It is assumed that each measured signal is a linear combination of each of the independent signals, and that there are an equal number of measured signals and independent signals.

Imagine that you are in a room where two people are speaking simultaneously. You have two microphones, which you hold in different locations. The microphones give you two recorded time signals, which we could denote by $x_1(t)$ and $x_2(t)$, with x_1 and x_2 the amplitudes, and t the time index. Each of these recorded signals is a weighted sum of the speech signals emitted by the two speakers, which we denote by $s_1(t)$ and $s_2(t)$. We could express this as a linear equation:

$$x_1(t) = a_{11}s_1 + a_{12}s_2$$
$$x_2(t) = a_{21}s_1 + a_{22}s_2$$

where a_{11}, a_{12}, a_{21}, and a_{22} are some parameters that depend on the distances of the microphones from the speakers. This is called the cocktail-party problem.

We can express the entire system of n measured signals as:

$$X = AS$$

Now we need to find W such that,

$$W = A^{-1}$$

$$S = W\tilde{X}$$

$\tilde{X}$ is the whitened signal

2. Theoretical Background

Kurtosis

Kurtosis is the classical method of measuring nongaussianity. When data is preprocessed to have unit variance, kurtosis is equal to the fourth moment of the data. In an intuitive sense, kurtosis measured how "spikiness" of a distribution or the size of the tails. Kurtosis is extremely simple to calculate, however, it is very sensitive to outliers in the data set. It values may be based on only a few values in the tails which means that its statistical significance is poor. Kurtosis is not robust enough for ICA.

Negentropy

The entropy of a discrete signal is equal to the sum of the products of probability of each event and the log of those probabilities. A value called differential entropy can be found for continuous function that uses the integral of the function times the log of the function. Negentropy is simply the differential entropy of a signal y, minus the differential entropy of a Gaussian signal with the same covariance of y. Negentropy is always positive and is zero only if the signal is a pure Gaussian signal. It is stable but difficult to calculate.

3. Algorithm Implementation and Development

3.1 *Preprocessing the Data*

It is easier to estimate W if the measured signals have a mean of zero, a variance of one and zero correlation. The data is preprocessed to meet these goals before W is estimated.[1]

Centering is achieved simply by subtracting the mean of signal from each reading of that signal.

A covariance matrix can be formed by taking the covariance between every pair of signals and forming a matrix.

$$C = cov(X)$$
$$C = E(XX^T)$$

The covariance matrix will be square and symmetric. We can perform an eigenvalue decomposition on the covariance matrix and then transform the data so the covariance matrix of the transformed data is equal to the identity. This procedure is also called *sphereing* since it normalizes the eigenvalues of the covariance matrix.

$$C = V\Lambda V^T$$

$$\tilde{X} = V\,\Lambda^{-\frac{1}{2}} V^T X$$

Now we have a whitened signal $\tilde{X}$ such that

$$E\left(\tilde{X}\tilde{X}^T\right) = I$$

Which can be shown by :

$$C = cov(X)$$

$$\tilde{X} = C^{-\frac{1}{2}} X$$

$$E\left(\tilde{X}\tilde{X}^T\right) = E\left(C^{-\frac{1}{2}} X X^T C^{-\frac{1}{2}}\right)$$

$$E\left(\tilde{X}\tilde{X}^T\right) = C^{-\frac{1}{2}} E(XX^T) C^{-\frac{1}{2}}$$

$$E\left(\tilde{X}\tilde{X}^T\right) = C^{-\frac{1}{2}} C C^{-\frac{1}{2}}$$

$$E\left(\tilde{X}\tilde{X}^T\right) = I$$

3.2 Finding the Matrix W

The Central Limit Theorem states that the sum of several independent random variables, such as those in S, tends towards a Gaussian distribution.

So $x_i = a_{1i}s_1 + a_{2i}s_2$ is more Gaussian than either s_1 or s_2. For example, the sum of a pair of dice approximates a Gaussian distribution with a mean of seven.

The Central Limit Theorem implies that if we can find a combination of the measured signals in X with minimal Gaussian properties, then that signal will be one of the independent signals. Once W is determined it is a simple matter to invert it to find A.

In order to find this signal, some way to measure the non-Gaussianity of $w^T X$ is needed. Where w is a random vector we use to try to find the direction along which the independent components in $\tilde{X}$ are the most non-gaussian. In order to do this we first choose a random vector and apply it to a formula to calculate negentropy given in (the name of the paper) and run it through a while loop to try to determine the perfect random vector that produces the most non-Gaussian feature. This is done for the matrix X for each timestep and then the w vector obtained for each of the mixed signals is stored in a matrix W which is the approximation for the inverse of the mixing matrix A.

The final step is to multiply this approximate W with the whitened signal matrix $\tilde{X}$ to get back the initial source signals. [2]

4. Result and visualization

The computational result from the Independent Component Analysis are shown from **Figure 1** to **Figure 3**. It has been shown that the recovered signal matches pretty well to the original signals.

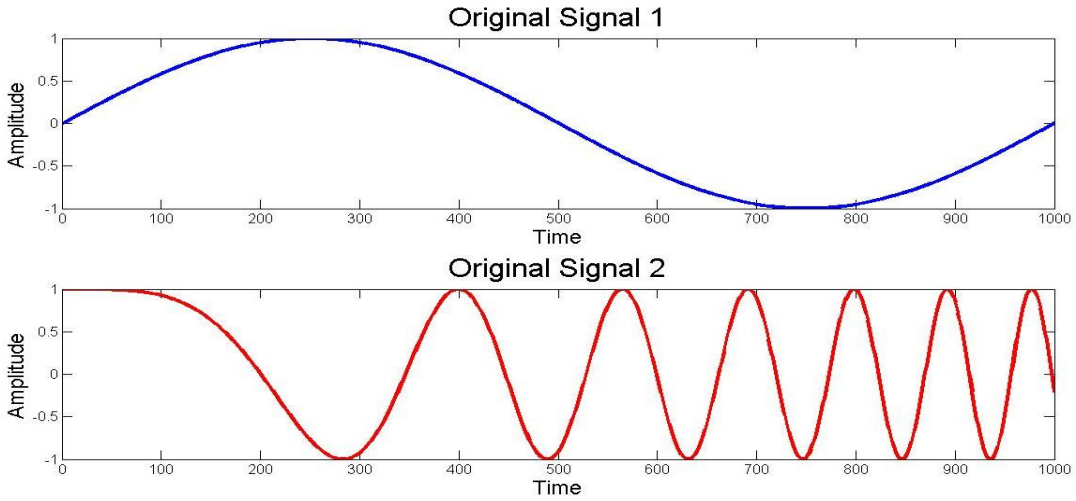

Figure 1. Original 2 signals generated for mixing

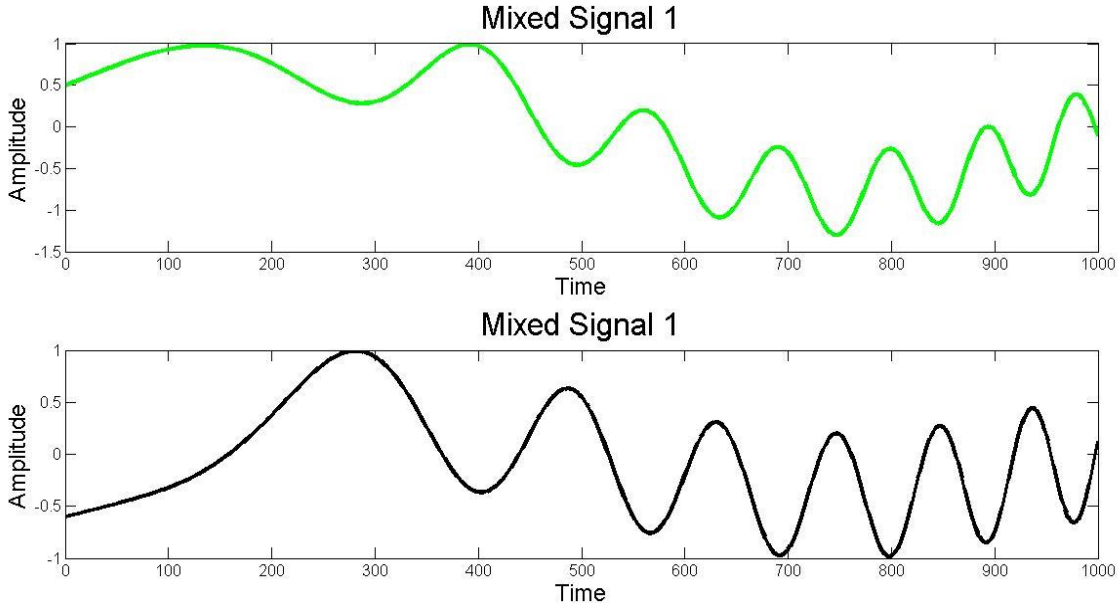

Figure 2. Original 2 signals generated for mixing

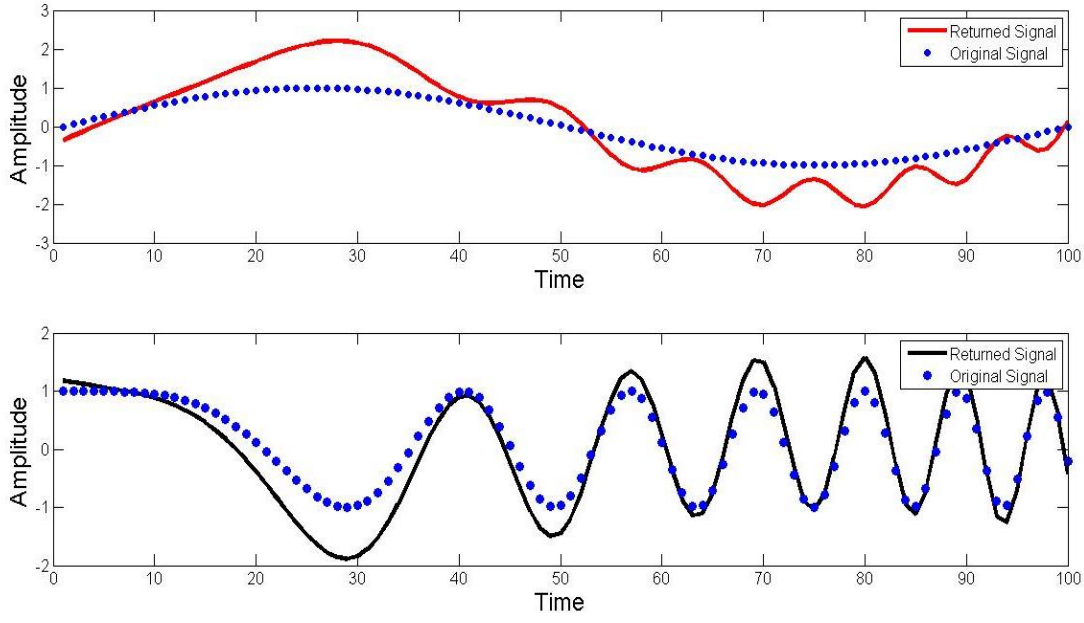

Figure 3. Comparison of recovered signals to original signals

5. De-convolution by Principal Component Analysis

Two mixed signals are stacked together into a 2-by-n matrix. This matrix now becomes our input and the original signals need to be recovered with the mixing matrix unknown. After the singular value decomposition (SVD) is done on the input matrix, we noticed that the first two modes are mostly dominant. [3] When we plot the proper orthogonal projection for the first two modes in **Figure 5**, we can see that the projections have similar trends as the original signal, even though there scales are off. If we scale the recover signal by the magnitude of the original signal (which is illegal since we assumed the original signal unknown), as shown in **Figure 6**, we could get a close approximation of the original signal. This proves that the recovered signals behave somewhat similar to the original ones. The recovery of the scale is not possible without knowing the peak of the original signals.

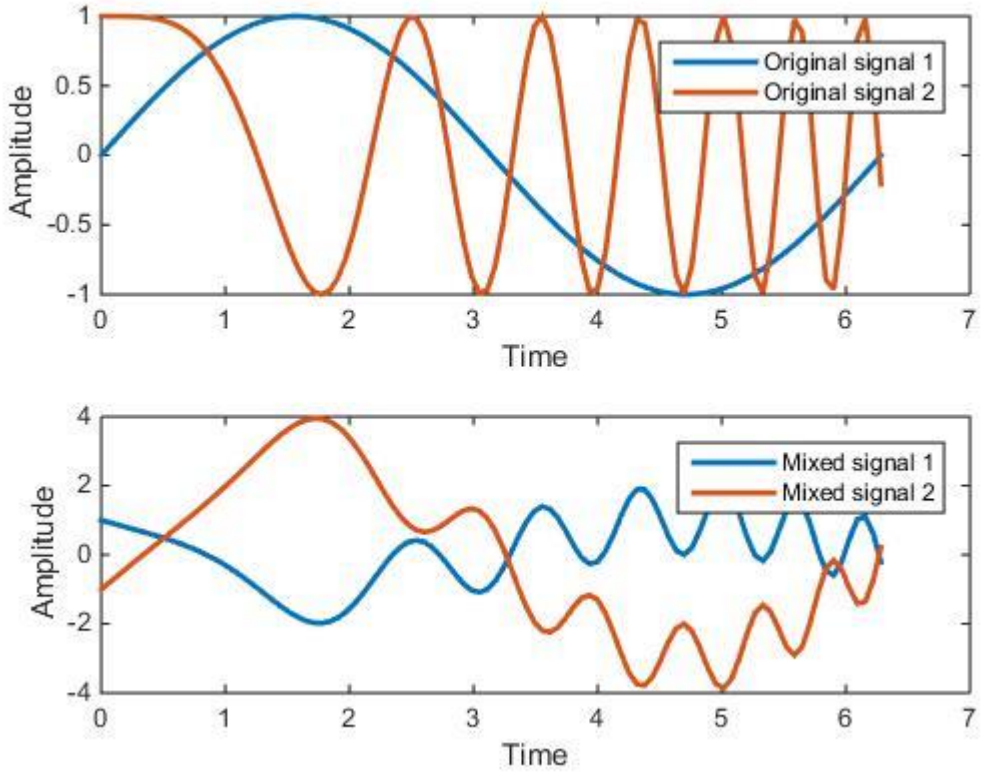

Figure 4. Original 2 signals (up) and mixed 2 signals (down), duplicated from **Figure 1** and **Figure 2**

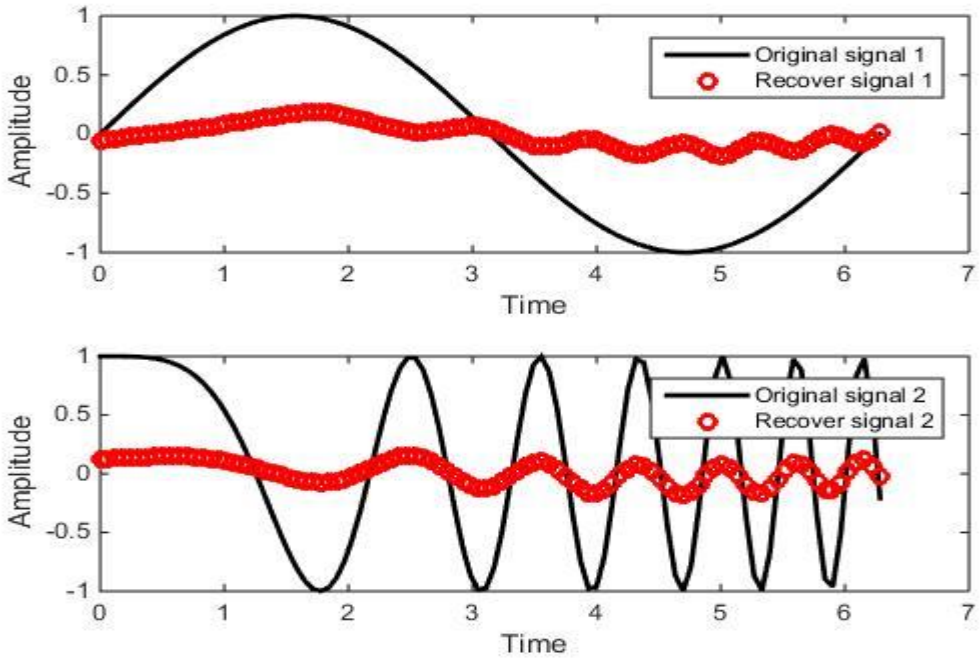

Figure 5. First two mode projections compared with two original signals

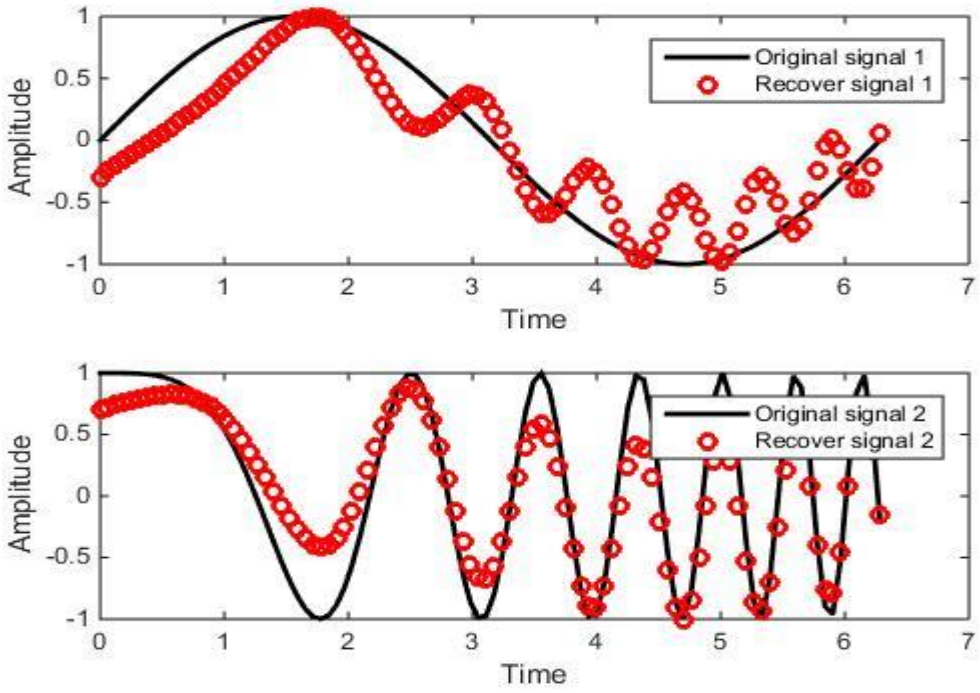

Figure 6. Scaled projections compared with original signals

6. Summary and conclusion

The Independent Component Analysis (ICA) method is proper in performing the de-convolution of mixed signals. Although the example shown in this report is simple and intuitive, this method can be implemented into much more complicated field such as medical imaging of brain with magnetoencephalography or separation of musical signals. There are other functions in the ICA algorithm which can be adjusted specifically for certain cases. Besides ICA, Principal Component Analysis is also proper for separating linearly mixed signals provided that some properties of the original signals are already known such as the peak value or concave value.

References:

1. *Estimating Independence*. http://www.oursland.net/tutorials/ica/ica-estimate.html.
2. Oja, A.H.a.E., *Independent Component Analysis: Algorithms and Applications.* 2000.
3. Kutz, J.N., *Data-Driven Modeling & Scientific Computation - Methods for Complex Systems & Big Data*. 2013, New York: Oxford University Press.

Appendix A. Special Matlab function used and explanation

Table 1. Special Matlab function used and explanation

Function	Implementation explanation
svd	Singular value decomposition on one matrix and stores the covariance in unitary matrix u, singular values in diagonal matrix s and projection in unitary matrix v.
eig	Eigenvalue decomposition
norm	Find a norm of a vector
cov	Calculate the covariance of two matrices or mean of one matrix
subplot	Plot several plots in one figure
linspace	Build a linear spaced vector within a certain domain with n grid points

Appendix B1. Matlab code for performing ICA

```
clear all;
clc
%Initiating the Signals

theta=linspace(0,2*pi,100);
s1=sin(theta);
s2=cos(theta.^2);

figure;
subplot(2,1,1); plot(s1,'b','Linewidth',3);
xlabel('Time','fontsize',18); ylabel('Amplitude','fontsize',18); title('Original Signal
1','fontsize',24)
set(gca,'fontsize',12)                    % Plotting the Original Signals
subplot(2,1,2); plot(s2, 'r','Linewidth',3);
xlabel('Time','fontsize',18); ylabel('Amplitude','fontsize',18); title('Original Signal
2','fontsize',24)
set(gca,'fontsize',12)
%%
%Creating the Mixed Signals

S=[s1;s2];
A=[0.8 0.5;0.4 -0.6];              %Mixing Matrix
x=A*S;
x1=x(1,:)'; x2=x(2,:)';

figure;
subplot(2,1,1); plot(x1,'g','Linewidth',3);
xlabel('Time','fontsize',18); ylabel('Amplitude','fontsize',18); title('Mixed Signal
1','fontsize',24)
set(gca,'fontsize',12)                    %Plotting the Mixed Signals
subplot(2,1,2); plot(x2, 'k','Linewidth',3);
```

```
xlabel('Time','fontsize',18); ylabel('Amplitude','fontsize',18); title('Mixed Signal
1','fontsize',24)
set(gca,'fontsize',12)
%%
% Pre-Processing

%Centering the Data
y=bsxfun(@minus,x,mean(x,2));

%Getting the covariance matrix
C=cov(y');

%Whitening the Data
[V,D]=eig(C);
z=(V*(D^(-0.5))*V')*y;

%%
%Find the inverse of matrix A

W=[];
f=zeros(2,1);
g=zeros(size(z));
h=zeros(1,size(z,2));

for k=1:size(A,1)
    w=rand(2,1);
    w=w/norm(w);
    while (abs(dot(f,w)) ~= 1)
        a=1;
        f=w;
        for j=1:size(z,2)
            g(:,j)=z(:,j)*cos(a*w'*z(:,j));
            h(1,j)=-a*(sin(w'*z(:,j)));
        end
        l=mean(g,2);
        h=mean(h);
        w=l-h*w;
        w=w/norm(w);
    end
    W=[W,w];
end

%%
%Getting back the original signals

Snew=W*z;
S1=Snew(1,:); S2=Snew(2,:);
figure();
```

```
subplot(2,1,1);
plot(S1,'r','Linewidth',3);
hold on
plot(s1,'o','markersize',5,'markerfacecolor','b'); hold off
xlabel('Time','fontsize',18); ylabel('Amplitude','fontsize',18);
set(gca,'fontsize',12);
leg=legend('Returned Signal','Original Signal');
set(leg,'fontsize',12);

subplot(2,1,2);
plot(S2,'k','Linewidth',3);
hold on
plot(s2,'o','markersize',7,'markerfacecolor','b'); hold off
xlabel('Time','fontsize',18); ylabel('Amplitude','fontsize',18);
set(gca,'fontsize',12)
leg=legend('Returned Signal','Original Signal'); set(leg,'fontsize',12);

ANS=[max(S1-s1);max(S2-s2)];              %Check for magnitude of difference between the 2
signals
```

Appendix B2. Matlab code for SVD and PCA

```
% Principal component analysis
%% Signal generation
clear variables, close all, clc
x = linspace(-pi, pi, 1000);
y1 = exp(-x) .* sin(2.*x);
y2 = sin(x) .* cos(20 * x);

x=linspace(0,2*pi,100);
y1=sin(x);
y2=cos(x.^2);

figure(1);
subplot(2, 1, 1);
plot(x, y1, x, y2);
legend('Original signal 1', 'Original signal 2');



y = [y1; y2];

A = [-1, 1; 3, -1];

ymix = A * y;
subplot(2, 1, 2);
plot(x, ymix(1, :), x, ymix(2, :));
legend('Mixed signal 1', 'Mixed signal 2');
```

```
%% PCA
clc
ystack = [ ymix(1, :); ymix(2, :)];
[u, s, v] = svd(ystack, 'econ');
figure(2)
subplot(2, 1, 1)
%plot(x, y1,'k-', x, v(:, 1) / max(abs(v(:, 1))) * max(abs(y1)), 'ro');
plot(x, y1,'k-', x, v(:, 1), 'ro');
legend('Original signal 1', 'Recover signal 1');
subplot(2, 1, 2);
%plot(x, y2,'k-', x, v(:, 2) / max(abs(v(:, 2))) * max(abs(y2)), 'ro');
plot(x, y2,'k-', x, v(:, 2), 'ro');
legend('Original signal 2', 'Recover signal 2');
```