Singular Spectrum Analysis

CEE 629. System Identification

Duke University, Fall 2017

Filtering of data via rank-reduced Hankel matrices goes by the terms "Structured Total Least Squares" and "Singular Spectrum Analysis" and finds application to a very wide range of problems [4]. For noise-filtering applications, a discrete-time signal, $y(i)$, $i = 1, \cdots N$, is broken up into k time-shifted segments and the segments are arranged as rows in a Hankel matrix, $\boldsymbol{Y} \in \mathbb{R}^{k \times j}$, $k < j$, $Y_{pq} = y_{p+q-1}$.

$$\boldsymbol{Y} = \begin{bmatrix} y(1) & y(2) & \cdots & \cdots & y(j-1) & y(j) \\ y(2) & y(3) & \cdots & \cdots & y(j) & y(j+1) \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ y(k-1) & y(k) & \cdots & \cdots & y(k+j-3) & y(k+j-2) \\ y(k) & y(k+1) & \cdots & \cdots & y(k+j-2) & y(k+j-1) \end{bmatrix}_{(k \times j)}$$

This is a Hankel matrix because values along the anti-diagonals are all equal to one another.

The SVD of $\boldsymbol{Y}$ is $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}}$, and a reduced-rank version of $\boldsymbol{Y}$ can be reconstructed from the first n dyads of the SVD.

$$\boldsymbol{Y}_n = \boldsymbol{U}_n \boldsymbol{\Sigma}_n \boldsymbol{V}_n^{\mathsf{T}}$$

where $\boldsymbol{U}_n \in \mathbb{R}^{k \times n}$, $\boldsymbol{\Sigma}_n \in \mathbb{R}^{n \times n}$, and $\boldsymbol{V}_n \in \mathbb{R}^{j \times n}$. are the first n singular vectors of $\boldsymbol{U}$ and $\boldsymbol{V}$ and the largest n singular values. This reduced-rank matrix $\boldsymbol{Y}_n$ is the rank-n matrix that is closest to $\boldsymbol{Y}$ in the sense of minimizing the Frobeneous norm of their difference, $||\boldsymbol{Y} - \boldsymbol{Y}_n||_F^2$. In general $\boldsymbol{Y}_n$ will not have the same Hankel structure as $\boldsymbol{Y}$, but a matrix with a Hankel structure, $\bar{\boldsymbol{Y}}_n$, can be obtained from $\boldsymbol{Y}$ in a number of ways.

- Singular Spectrum Analysis [2].
  In SSA, $\boldsymbol{Y}_n$ is computed as above, and elements along each anti-diagonal are replaced by the average of the anti-diagonal. The resulting matrix $\bar{\boldsymbol{Y}}_n$ will not have rank-n and will not be the closest matrix to $\boldsymbol{Y}$ in the sense of Frobeneous, but it will have a Hankel structure.

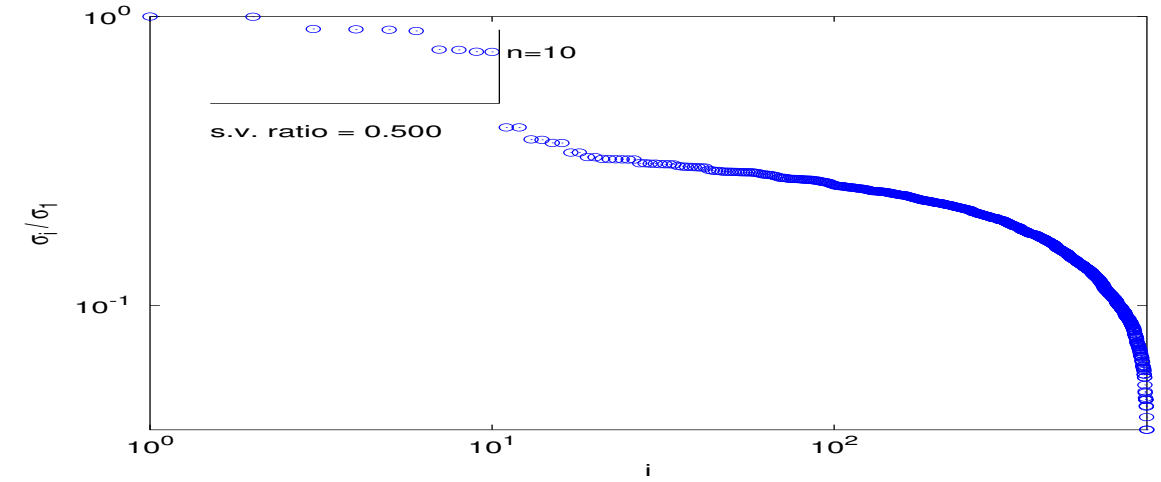
- Cadzow's algorithm [1].
  In Cadzow's algorithm, the SSA averaging is repeatedly applied. After each anti-diagonal averaging step, the matrix grows in rank, so a new SVD can be computed, a new rank-n matrix can be constructed, and the anti-diagonals of the new reduced rank matrix can be averaged.

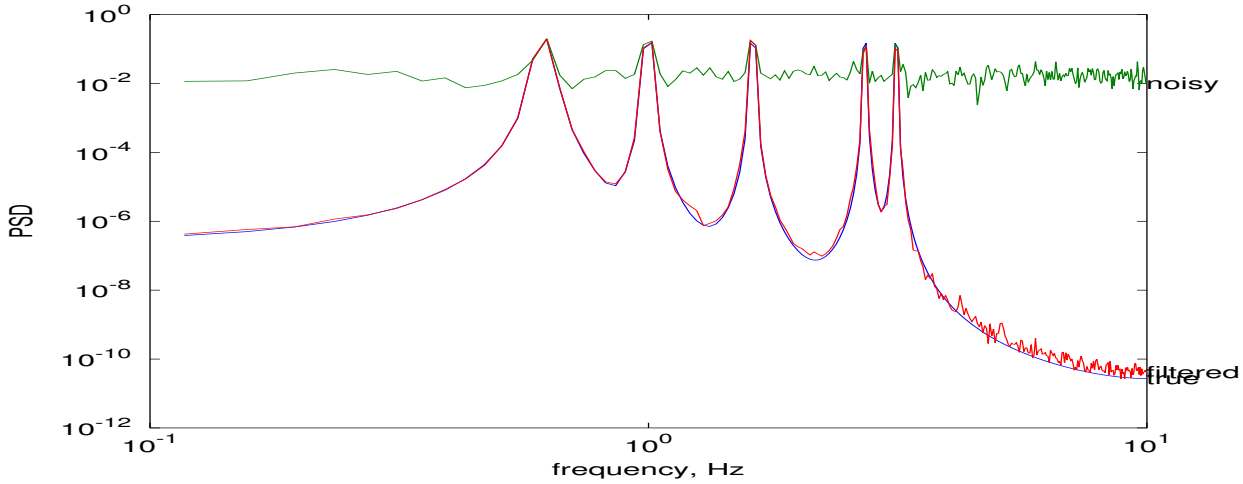
- Structured low-rank approximation [3].
  These methods solve the constrained optimization problem: $\min ||\boldsymbol{Y} - \bar{\boldsymbol{Y}}_n||_F^2$ such that $\mathsf{rank}(\bar{\boldsymbol{Y}}_n) = n$ and $\bar{\boldsymbol{Y}}_n$ has a desired structure. These methods are iterative, but apply more rigorous methods to determining $\bar{\boldsymbol{Y}}_n$.

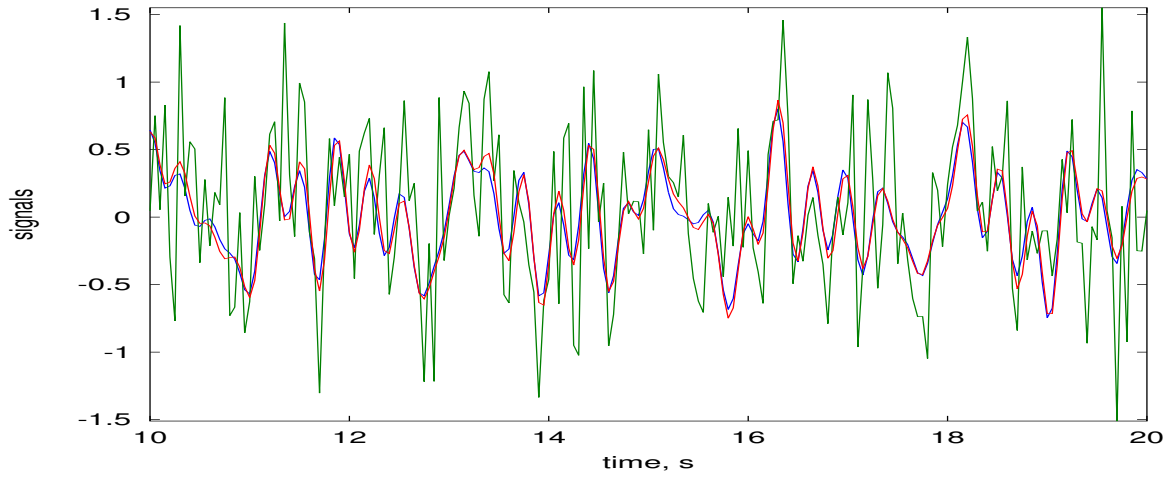
The low-rank (filtered signal) can be recovered from the first row and last column of the "Hankelized" $\bar{\boldsymbol{Y}}$. The examples below apply SSA to recoving signals from noisy data. An apparent "cliff" in the plot of the singular values of $\boldsymbol{Y}$ indicates that a signal can be represented by a few components of the SVD of $\boldsymbol{Y}$. This is the case in the first example (a good application of SSA in which the signal-to-noise ratio can even be less than 1), but not in the second. SSA is a type of Principal Component Analysis (PCA).

1 Recover a reduced basis from a very noisy measurement

signal is sum of sines:	$y(i) = \sum_j \sin(2\pi f_j t_i)$	
noisy measurements:	$\tilde{y}(i) = y(i) + dv$	v is a unit Gaussian white noise process
	$d = \sigma_y \sqrt{\Delta t}$ / SNR	SNR : signal-to-noise ratio

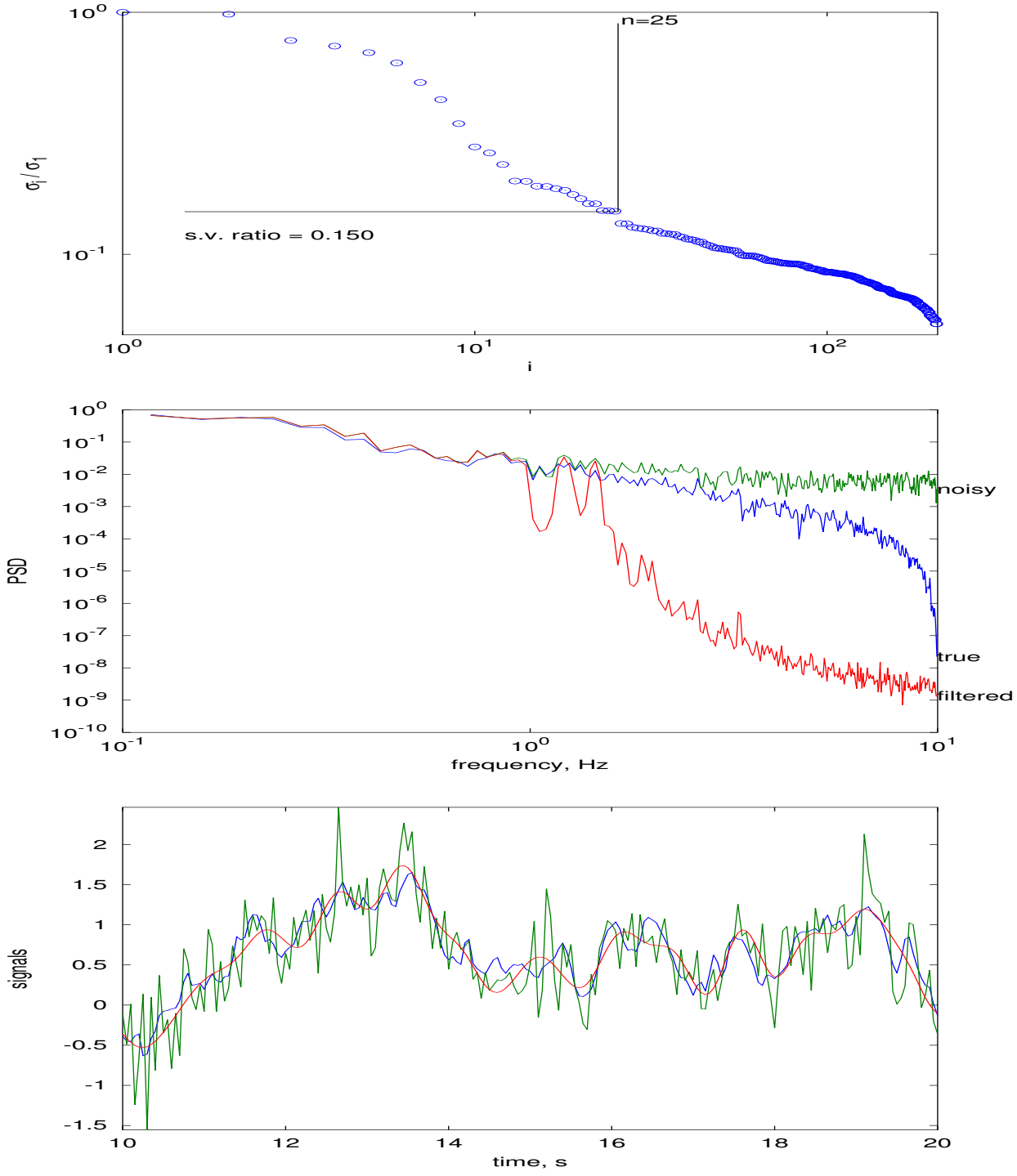






2 Filter a noisy measurement of a broad-band response

noise-driven linear dynamics:	$\dot{x} = ax + bw$	w is a unit Gaussian white noise process
true output:	$y = cx$	
noisy measurements:	$\tilde{y} = y + dv$	v is a unit Gaussian white noise process
parameter values:	$a = -0.5;\ b = 1;\ c = 1;$	
	$d = \sigma_y\sqrt{\Delta t}$ / SNR	SNR : signal-to-noise ratio



SSA_filter.m

```
function [ y , y_components ] = SSA_filter ( y, k, sv_ratio )
% [ y , y_components ] = SSA_filter ( y, k, sv_ratio )
% Singular Spectrum Analysis to filter out insignificant (noisy) components
%
%     INPUTS              DESCRIPTION
%   ==========          ==============
%     y                   signal to be filtered,                    1 by N
%     k                   rows in the Hankel matrix of y, Y         k < N/2+1
%   sv_ratio              remove components of SVD of Y with s_i < sv_ratio*s_1
%
%     OUTPUT            DESCRIPTION
%   ==========          ==============
%     y                 re-construction of y from low-rank approximation of Y
%     y_components      principle components of y from low-rank approximation of Y

 [l,N] = size(y);

% (1) put signal into a Hankel Matrix. of dimension k x j;  k < j; k+j = N+1
 if (k > N/2) error(' SSA_filter: k should be less than N/2'); end
 j = N+1-k              % number of columns of the Hankel matrix

 Y = zeros(k,j);
 for i=1:k
   Y ( i , : ) = y ( i : i+j-1 );
 end

% (2) compute the SVD of the Hankel matrix
 [U,S,V] = svd ( Y , 'economy' );

 if ( sv_ratio < 1 )              % find the most significant components
   n = max(find(diag(S)/S(1,1) > sv_ratio ))
 else
   n = sv_ratio
   Snn =     S(n,n)
   Sn1 =     S(n+1,n+1)
   sv_ratio = (S(n,n)+S(n+1,n+1))/2 / S(1,1);
 end

 figure(557)
   loglog(diag(S)/S(1,1),'o', [1 n]+.5,[1 1]*sv_ratio,'k', [n n]+.5,[sv_ratio 0.9],'k')
     ylabel('\sigma_i / \sigma_1')
     xlabel('singular value number, i')
     text(1.5,0.8*sv_ratio,sprintf('s.v. ratio = %5.3f',sv_ratio))
     text(1.2*n,0.5+0.5*sv_ratio,sprintf('n = %3.0f',n))
     axis('tight')


% (3) build a new rank-n matrix from the first n dyads of the SVD
 Y = U(:,1:n)*diag(diag(S)(1:n))*V(:,1:n)';


% (4) average anti-diagonal components to make the lower-rank matrix a Hankel matrix.
% extract the filtered signal from the first column and last row of the
% lower-rank Hankel matrix.
 y = zeros(1,N);
 y(1) = Y(1,1);

 for i=2:j      % anti-diagonals of first row of Hankel matrix
   min_ik = min(i,k);   % length of anti-diagonal
   y(i) = sum(diag(Y( min_ik:-1:1 , i-min_ik+1:i ))) / min_ik;
 end

 for i=2:k      % anti-diagonals of last column of Hankel matrix
   y(j+i-1) = sum(diag(Y( k:-1:i , j-k+i:j ))) / (k-i+1);
 end


```

```
% (5) break-out individual principal components of the signal ...
 y_components = zeros(n,N);
 for kk=1:n
   Y = U(:,kk)*S(kk,kk)*V(:,kk)';
   y_components(kk,1) = Y(1,1);
   for i=2:j            % first column of Hankel matrix
     min_ik = min(i,k);
     y_components(kk,i) = sum(diag(Y( min_ik:-1:1 , i-min_ik+1:i ))) / min_ik;
   end
   for i=2:k            % last row of Hankel matrix
     y_components(kk,j+i-1) = sum(diag(Y( k:-1:i , j-k+i:j ))) / (k-i+1);
   end
 end

% ------------------------------------------------- SSA_filter   HP Gavin
%  2013-09-10  2015-02-13
% System Identification, Duke University, Fall 2015
```

```
% SSA_filter_test --- test the use of SVD of signal Hankel matrix for filtering

% use SVD to remove noise
% k = number of rows in Hankel matrix ; k < N/2+1 ;
% larger k :: slower SVD :: less extraction
% larger k :: sharper SVD knee :: less noise in principal components

% HP Gavin, CEE 629, System Identification, Fall 2015

Example = 1;

randn('seed',2);          % initialize random number generator
N    = 2048;              % number of data points
dt   = 0.05;              % time step increment
t    = [1:N]*dt;          % time values

if (Example == 1)         % sum of harmonic signals

 freq = [ (sqrt(5)-1)/2 ; 1.0 ; 2/(sqrt(5)-1) ; e ; pi ]; % set of frequencies
 yt   = sum(sin(2*pi*freq*t));                           % true signal

 SNR = 0.5;               % works with very poor signal-to-noise ratio
 k = ceil(0.4*N + 1 )
 sv_ratio = 0.5;          % singular value ratio closer to 1 :: more filtering

end

if (Example == 2)         % dynamical system driven by unit white noise

 yt = lsym(-0.5,1,1,0,randn(1,N)/sqrt(dt),t,0);          % true signal

 SNR = 2.0;               % needs better signal-to-noise ratio
 k = ceil(0.1*N + 1 )
 sv_ratio = 0.15;         % smaller singular value ratio  :: less filtering

end

% add measurement noise
yn = yt + randn(1,N)/sqrt(dt) * ( sqrt(yt*yt'/N) * sqrt(dt) / SNR );

[yf, yf_components] = SSA_filter ( yn, k, sv_ratio );          % remove random components

yf_yt_err = norm(yf-yt)/norm(yt)                % compare filtered to true
yf_yn_err = norm(yf-yn)/norm(yt)                % compare filtered to noisy

nfft = 512;
[PSDyt,f]  =  psd(yt,1/dt,nfft);
[PSDyn,f]  =  psd(yn,1/dt,nfft);
[PSDyf,f]  =  psd(yf,1/dt,nfft);

[n,N] = size(yf_components);
J = length(freq);

% Mode Separation
% Least squares of principle components with cosine and sine at peak frequencies
% in order to separate modes ... this does seem to work ...
% The least squares calculation can make use of the SVD of Y ... faster.
a = nan(J,n);             % cosine coefficients
b = nan(J,n);             %   sine coefficients
for j=1:J
  a(j,:)  = cos(2*pi*freq(j)*t) / yf_components;
  b(j,:)  = sin(2*pi*freq(j)*t) / yf_components;
end

yf_components_a = a*yf_components;   % combine principle components into "cosine" modes
yf_components_b = b*yf_components;   % combine principle components into  "sine"  modes
yf_components_ab = (a+b)*yf_components;   % combine principle components into modes
```

References

[1] Cadzow, J.A., "Signal Enhancement: a composite property mapping algorithm," *IEEE Trans. Acoustics, Speech, and Signal Processing,* 36(2):49-82 (1988).

[2] Golyandina et. al., *Analysis of Time Series Structure: SSA and related techniques,* Champman-Hall, CRC 2001

[3] Lemmerling, Philippe, *Structured Total Least Squares: Analysis, Algorithms, and Applications* Ph.D. Dissertation, Leuven, 1999.

[4] Markovsky, Ivan "Structured low-rank approximation and its application," *Automatica* 44: 891-909 (2008).

[5] Markovsky, Ivan *Structured low-rank approximation and its application,* Springer-Verlag, 2012