MIMO Wiener Filtering Examples

CEE 629. System Identification

Duke University, Fall 2017

1 De-noise a signal containing uncorrelated noise

1.1 System

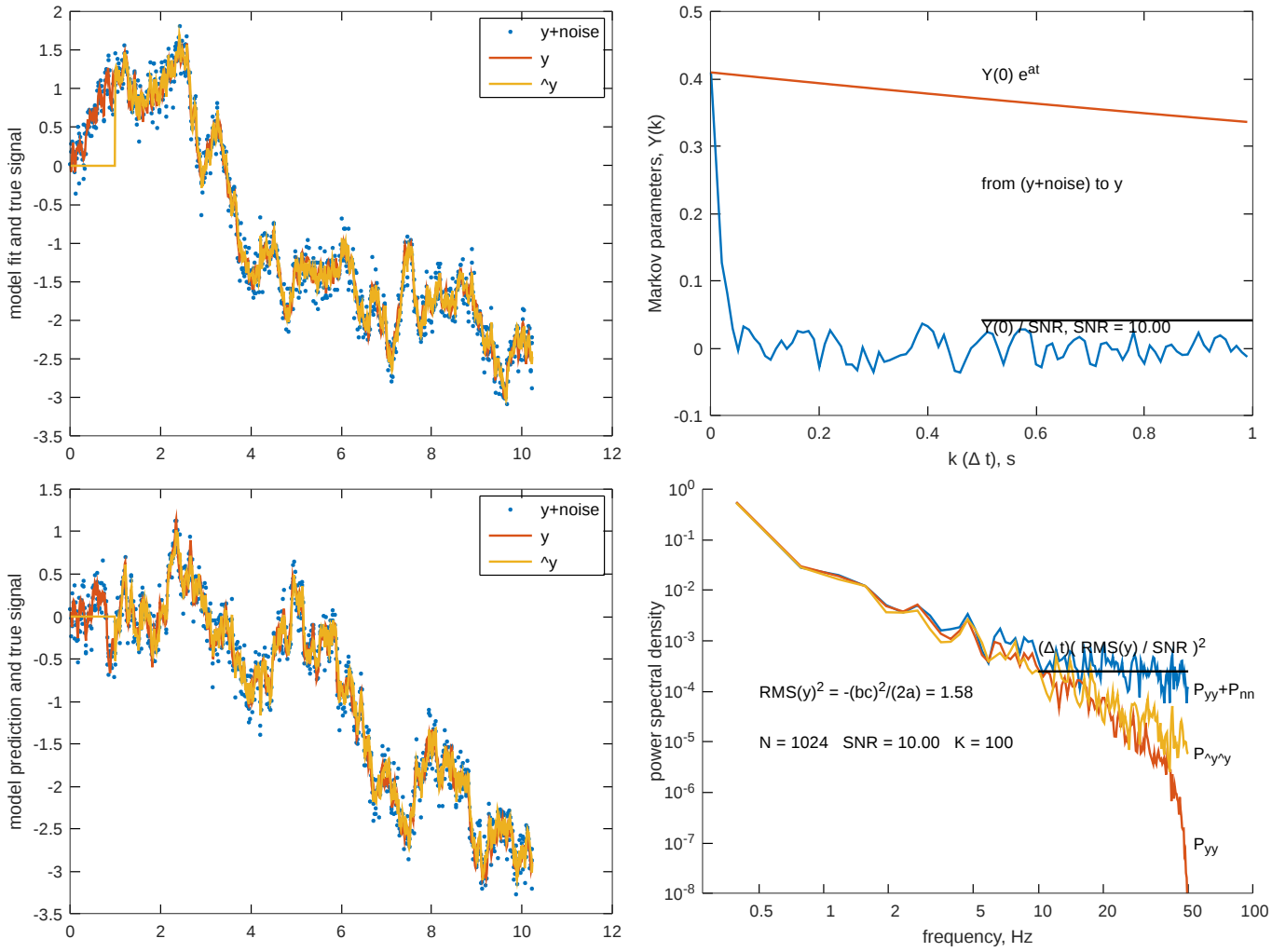
noise-driven linear dynamics:	$\dot{x} = ax + bu$	u is Gaussian unit white noise
true output:	$y = cx$	
noisy measurements:	$\tilde{y} = y + dv$	v is Gaussian unit white noise
parameter values:	$a = -0.2;\ b = 1;\ c = 1;$	
	$d = \sqrt{-(bc)^2/(2a)}\ /\ \text{SNR}$	SNR : signal-to-noise ratio
Number of coefficients :	$K = \log(\text{SNR})/\max(-\text{Re}(\lambda)(\Delta t))$	

1.2 Computation

```
  a = -0.2; b = 1; c = 1;   m = 1; r = 1;          % model parameters
  RMSy = sqrt(-(b*c)^2/a/2)                        % RMS output to unit noise
  dt = 0.01;                                       % time-step increment
% K = log(SNR)/(-real(a)*dt)                       % number of Markov param's
  N = 2^floor(log2(20*K))                          % number of data points
  t  = [0:N-1]*dt;                                 % time axis
  y = lsym(a,b,c,0, randn(r,N)/sqrt(dt),t,0);      % clean fit data
  d = RMSy/SNR;                                    % msmnt noise amplitude
  y_noisy = y +  d * randn(m,N);                   % noisy fit data
  Y     = WienerFilter( y_noisy, y, K);            % fit Wiener-Hopf model
  y_hat = WienerFilter( y_noisy, zeros(m,N), Y);   % evaluate fit model
  FitError = norm( y(K:N) - y_hat(K:N), 'fro' ) / norm( y(K:N), 'fro');

  y = lsym(a,b,c,0, randn(r,N)/sqrt(dt),t,0);      % clean validation data
  y_noisy = y + d * randn(m,N);                    % noisy validation data
% use the estimated filter to predict the clean data from the noisy data
  y_hat = WienerFilter( y_noisy, zeros(m,N), Y);   % evaluate w/ validation data
  PredictionError = norm( y(K:N) - y_hat(K:N), 'fro' ) / norm( y(K:N), 'fro');
```

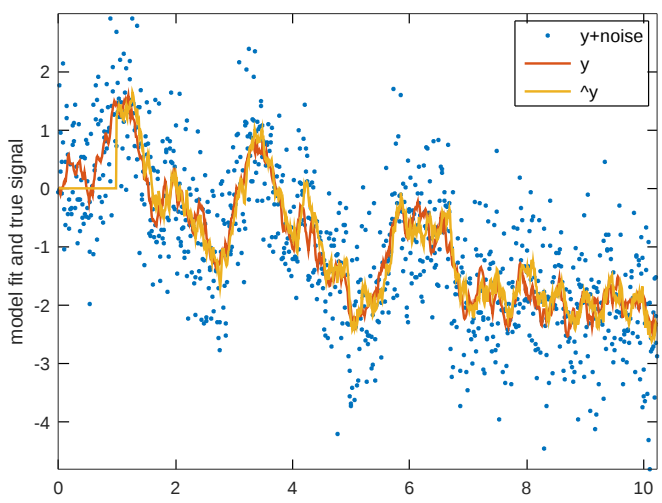
- SISO, single state. For unit white input, $\lim_{N\to\infty}(y^{\mathsf{T}}y)/N = (bc)^2/(2a)$. Why?
- Unit white noise u drives the system, $\dot{x} = ax + bu$,
- Unit white noise v is added to the output, $y = cx$, $\tilde{y} = y + dv$
- The noise u and the noise v are uncorrelated.
- The *noisy output* ($\tilde{y}$) is the *input* to the Wiener filter. (!)
- The *clean output* (y) is the *output* of the Wiener filter.
- All the noise is in the *independent* data, the filter input (!)
- The noisy signal $\tilde{y}$ is used to compute an estimate $\hat{y}$ of the true signal y.
  This is "model-based de-noising."
- The input u driving y is not involved in this application of the Wiener filter. In this example, Wiener filtering is *not* Kalman filtering.
- Wiener filtering is *not* band-pass filtering.

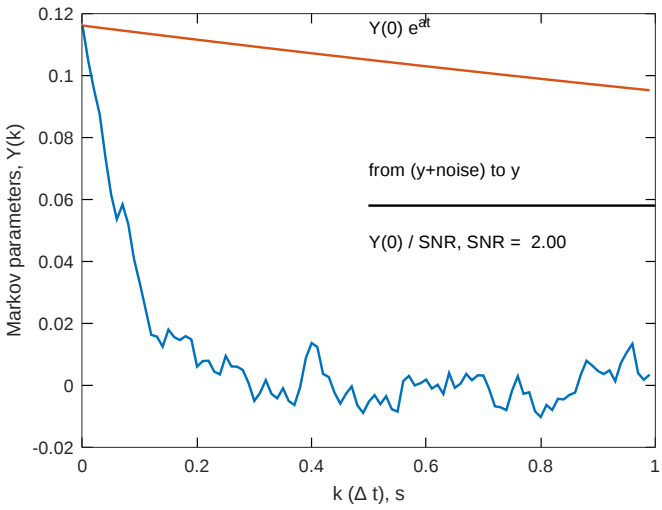
K=100, SNR = 10

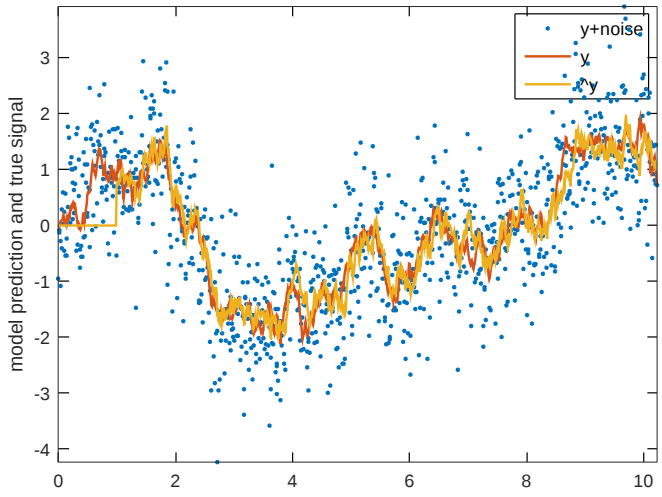


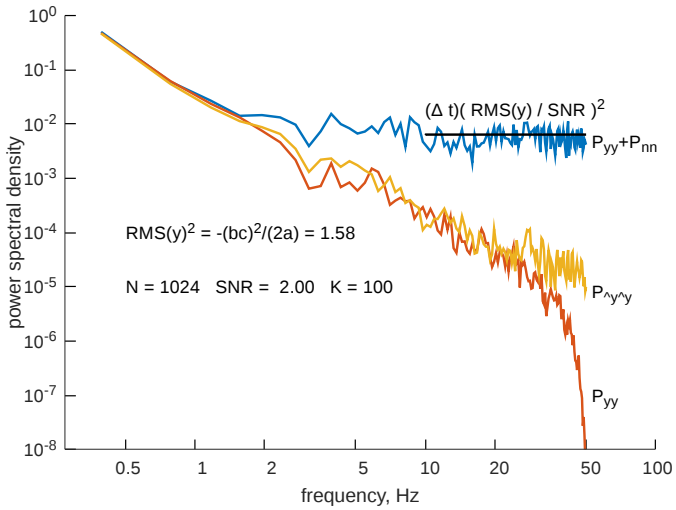
- In this example, the filter Markov parameters are *not* the Markov parameters of the dynamic system (a, b, c, d).
- The PSD of y drops to zero at $f = 1/(2\Delta t)$ but the PSD of $\hat{y}$, computed from the Wiener filter does not. Why?
- Compare the performance of Wiener filtering in this example to the performance of the SSA filter in a similar example.

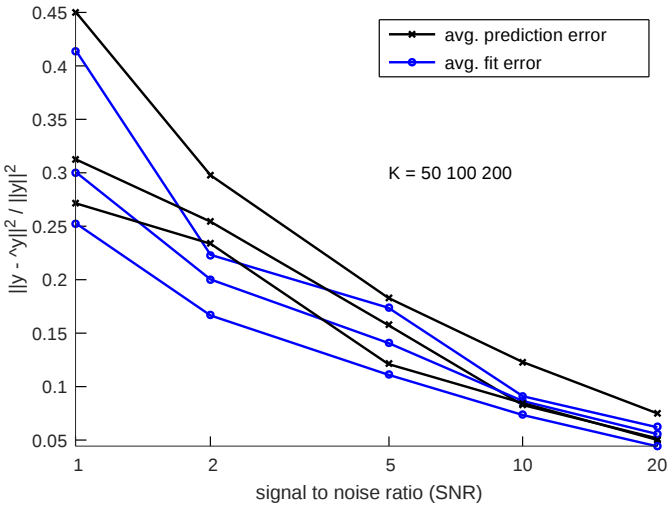
K=100, SNR = 2











- The accuracy of the de-noising Wiener filter improves monotonically with K, the number of filter coefficients, and SNR, the signal-to-noise ratio.

2 Identify impulse response of a MIMO system from noisy I/O data

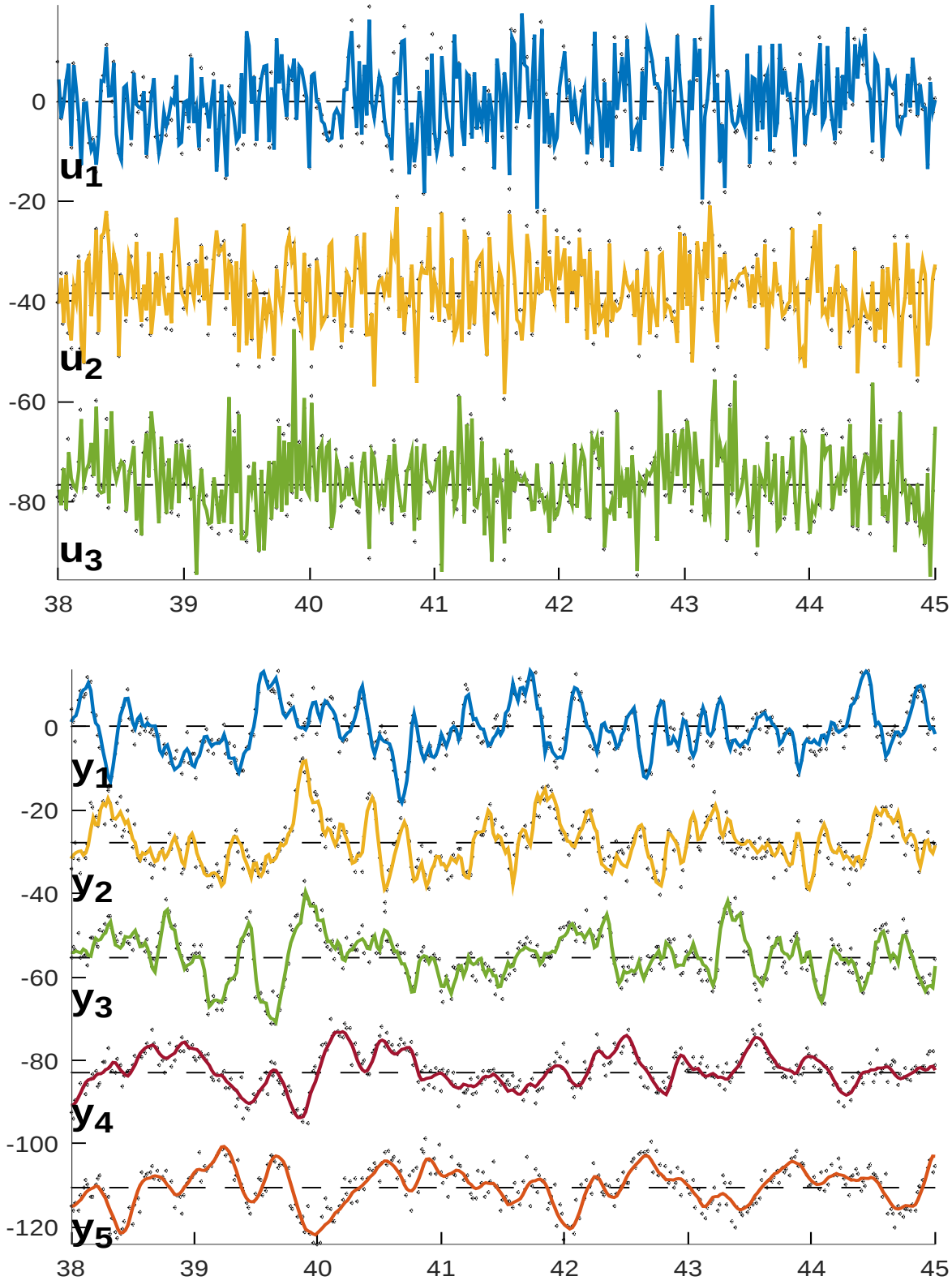
2.1 An LTI system in complex modal coordinates

noise-driven linear dynamics:	$\dot{x} = Ax + Bu$	u is Gaussian unit white noise
true output:	$y = Cx$	
noisy input measurements:	$\tilde{u} = u + (1/\mathrm{SNR})w$	w is Gaussian unit white noise
noisy output measurements:	$\tilde{y} = y + (\sqrt{\Delta t}\ \sigma_Y/\mathrm{SNR})v$	v is Gaussian unit white noise
parameter values:	$A = \mathsf{diag}\left(\begin{bmatrix} p_i & \\ & p_i^* \end{bmatrix}\right)$	$A \in \mathbb{C}^{n\times n}$
	$B = \begin{bmatrix} \vdots & \\ - \ b_i \ - & \\ - \ b_i^* \ - & \\ \vdots & \end{bmatrix}$	$B \in \mathbb{C}^{n\times r}$
	$C = \begin{bmatrix} & \vert & \vert & \\ \cdots & c_i & c_i^* & \cdots \\ & \vert & \vert & \end{bmatrix}$	$C \in \mathbb{C}^{m\times n}$
	$D = 0$	$D \in \mathbb{C}^{m\times r}$

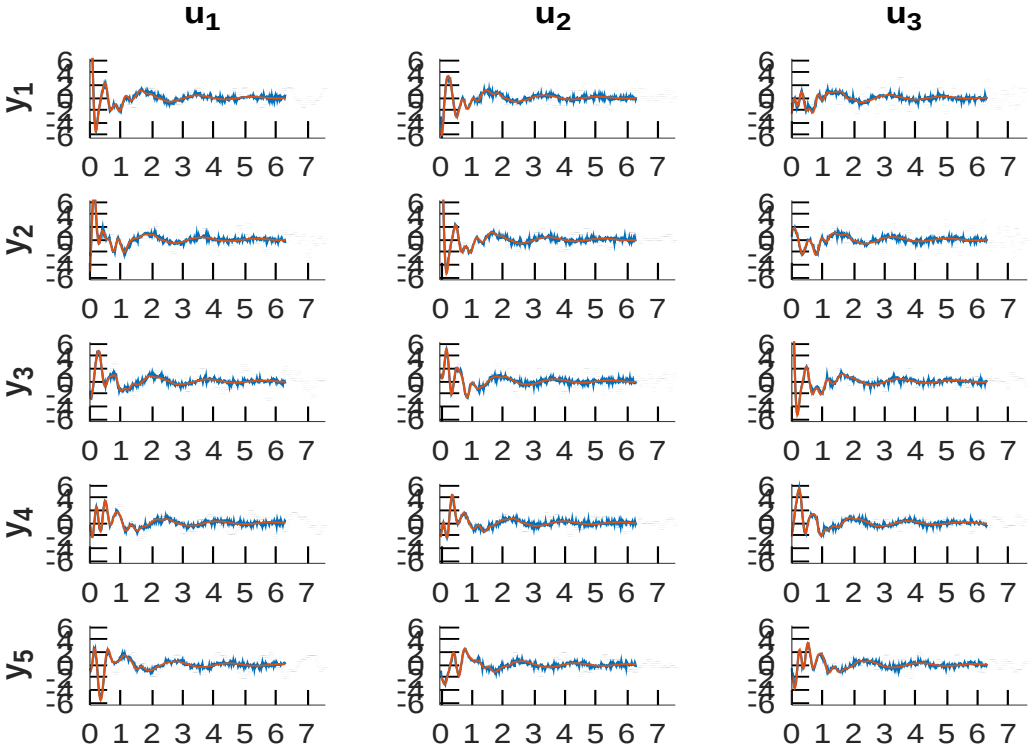
2.2 Computation

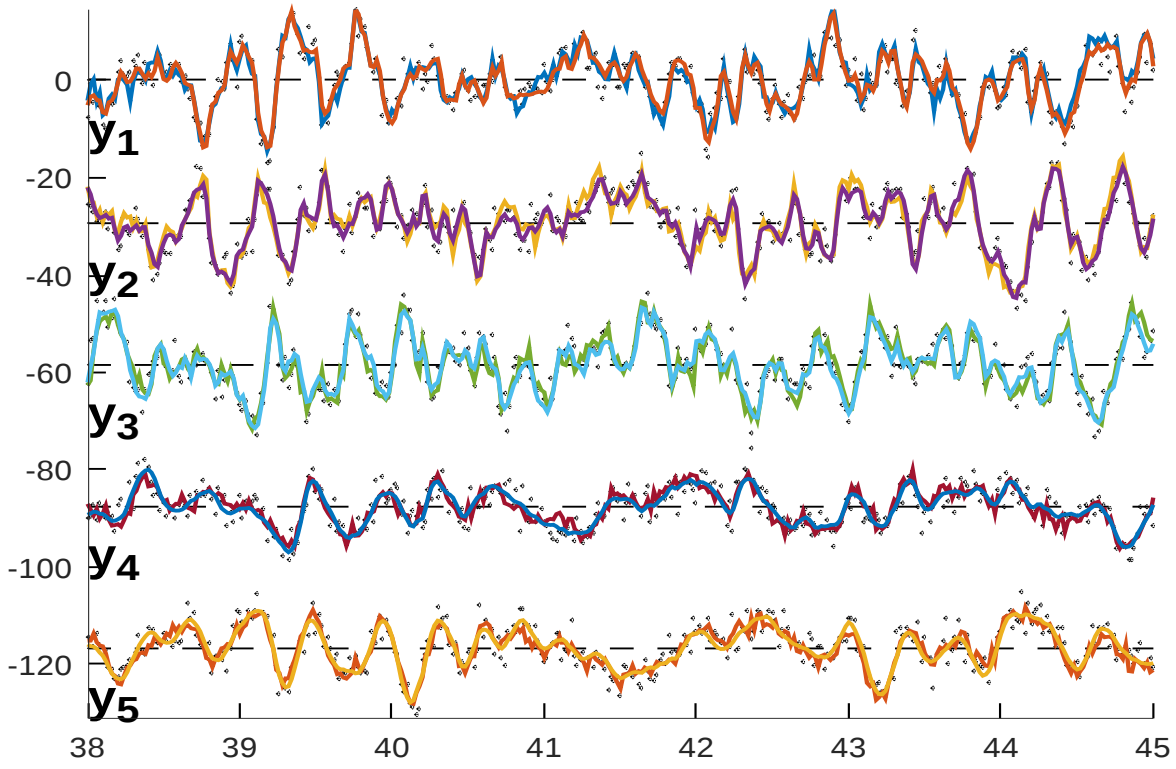
```
% Specify an LTI system used to simulate some input-output data ...
 n = 18;         % number of internal states (even) ... complex-conjugate poles
 r = 3;          % number of inputs
 m = 5;          % number of outputs
 SNR = 5;        % signal-to-noise ratio

 f_min = 0.50;  f_max = 5.0;            % range of natural frequencies, Hz
 z_min = 0.05;  z_max = 0.20;           % range of damping ratios
 wn = 2*pi*(f_min + (f_max-f_min)*rand(1,n/2));
 z  = z_min+(z_max-z_min)*rand(1,n/2);
 [A, B, C, D] = modalLTI(wn,z,m,r);     % generate a continuous time LTI

 damp(A)                          % check the nat'l frequencies and damping ratios

% --- generate records of clean input/output data ...
 dt = 0.02;                             % time-step increment
 K  = ceil(-log(0.05)/(min(z.*wn*dt)))  % number of Markov parameters
 N  = 5*K*(r+1);                        % number of data points
 t  = [0:N-1]*dt;                       % time data
 u  = randn(r,N) / sqrt(dt);            % clean input data = unit white noise
 y  = lsym(A,B,C,D,u,t);                % clean output data
 rmsY = sqrt(real(trace(C*liap(A,B)*C')))       % mean-square of all outputs
 d_u  = 1 / SNR;
 d_y  = sqrt(dt) * rmsY / SNR;
 u_noisy = u  +  d_u * randn(r,N)/sqrt(dt);     % noisy input  record
 y_noisy = y  +  d_y * randn(m,N)/sqrt(dt);     % noisy output record
 Y = WienerFilter( u_noisy, y_noisy, K );       % fit Wiener model
 y_hat = WienerFilter ( u_noisy, zeros(m,N), Y );
 ModelError = norm(y(:,K:N)-y_hat(:,K:N),'fro') / norm(y(:,K:N),'fro')
% validation step
 u = randn(r,N) / sqrt(dt);             % generate _new_ clean input data
 u_noisy = u  +  d_u * randn(r,N)/sqrt(dt);
% use the estimated filter to predict the new output from new input data w/noise
 y_hat = WienerFilter ( u_noisy, zeros(m,N), Y );
% simulate clean output output with the true model and the clean input
 y = lsym(A,B,C,D,u,t);
 PredictionError = norm(y(:,K:N)-y_hat(:,K:N),'fro')/norm(y(:,K:N),'fro')
 y_noisy = y  +  d_y * randn(m,N)/sqrt(dt);     % noisy output record
```

Input-Output data used to estimate Markov parameters



Markov parameters and validation results





- In this application of the Wiener filter we estimate Markov parameters for a MIMO system from noisy input / output data.
- In discrete time, Gaussean unit white noise $u(k)$ is $\sim \mathcal{N}(0, 1/\sqrt{\Delta t})$. Why?
- For unit white input, $\sigma_Y^2 = \lim_{N\to\infty}(\text{trace}[YY^{\mathsf{T}}])/N = \text{trace}[CQC^{\mathsf{T}}]$. Why? What is Q?
- Unit white noise u drives the system, $\quad \dot{x} = Ax + Bu$,
- Scaled white noise w is added to the input data, $\tilde{u} = u + dw$
- Scaled white noise v is added to the output data, $\tilde{y} = y + dv$
- The noise sequences u, v, and w are uncorrelated.
- The *noisy input* ($\tilde{u}$) and *noisy output* ($\tilde{y}$) are used to estimate the Markov parameters.
- A coarse assessment of the system dynamics (damping ratio and natural frequency) and knowledge of the time step can be used to guide a suitable choice for the number of Markov parameters. Values of the last Markov parameters should be close to zero. If the system response decays with $e^{-\zeta\omega_n k(\Delta t)}$ the Markov parameters will decay by 95% in $K = -\ln(0.05)/(\zeta\omega_n(\Delta t))$ time steps.
- To validate the model, a new input signal u is used to simulate the true (state-space) model outputs y, the Wiener filter outputs $\hat{y}$ are computed using the noisy-version $\tilde{u}$ of the validation input.
- In this example, the filter Markov parameters *are* the Markov parameters, $CA^{k-1}B$, of the dynamic system (A, B, C, D).
- In this example, the Wiener filter has 310 Markov parameters, where each parameter is 5×3, (a total of 4650 values). The Markov parameters are computed from input data ($3 \times 6200 = 18600$ values) and output data ($5 \times 6200 = 31000$ values).
- The Markov parameters are the orthogonal projection of the cross-correlation of the input-output data $(m \times rK) = (5 \times 930)$ onto the auto-correlation of the output data $(rK \times rK) = (930 \times 930)$. This relationship is known as the Wiener-Hopf equations.
- The auto-correlation matrix is the inner product of block-Toeplitz matrices. It has full rank, even if the measurements are noise-free, and is generally well-conditioned (condition number ≈ 10).
- A state-space model for this system contains all the information of the sequence of Markov parameters, but involves only $(n + m)(n + r) = (18 + 5)(18 + 3) = 483$ values. State-space models are generally a more efficient model representation than the the sequence of Markov parameters, and are more useful for analyzing system dynamics and for synthesizing controllers.
- (comment on the use of ARMA models fitted via linear least squares)

WienerFilter.m

```
function Yy = WienerFilter( u, y, YK )
% Yy = WienerFilter( u, y, YK )
%
%     INPUTS              DESCRIPTION
%   ==========          ===============
%       u               sequence of input data                          (r x N)
%       y               sequence of outputs                             (m x N)
%      YK               if YK is a scalar it is interpeted as the number
%                       of Markov parameters, K, and the fitted Markov
%                       parameters, Y (m x rK),  are returned.
%                       otherwise,  YK is interpreted as a set of Markov
%                       parameters, Y (m x rK), and the model output, y (m x N),
%                       is returned
%
%    OUTPUTS            DESCRIPTION
%   ==========          ===============
%       Yy              YK is a scalar, Yy contains the Markov parameters (m x rK)
%                       otherwise, Yy contains the model output (m x N)

 [r,Nu] = size(u);       % number of  inputs, duration of  inputs
 [m,Ny] = size(y);       % number of outputs, duration of outputs

 N = min(Nu,Ny);

 if max(size(YK)) == 1           % --- fit the model to data sequences
        K = YK;
        if ( m*r*K > m*(N-K+1) )
                error(' WienerFilter: not enough data to fit');
        end
 else                            % --- test the model on data
        Y = YK;
        K = size(Y,2)/r;
 end

 U = zeros(r*K,N-K+1);
 for k=1:K
        U ( r*(k-1)+1:r*k , : ) = u ( : , K-k+1:N-k+1 );
 end

 if max(size(YK)) == 1           % --- fit the model to data

        Ruu = U*U';              %  auto-correlation of input data
        Ryu = y(:,K:N)*U';       % cross-correlation of input/output data

        Yy = Ryu / Ruu;          % Wiener-Hopf equations

 else                            % --- evaluate the model output

        Yy = [ zeros(m,K-1) Y*U ];

 end

 clear U

%{
 Su = svd(Ruu);
 figure(2)
  clf
  plot(Su/Su(1,1),'o')
   ylabel('\sigma_i/\sigma_1,   singular values of Ruu')
   xlabel('i')
%  axis([0 m*K+1 -1 1.1*Su(1,1)])
%}
% ------------------------------------------ WienerFilter   HP Gavin
% System Identification, Duke University, Fall 2013,
% updated:  2013-09-04, 2017-10-06
```