EE 4323 – Industrial Control Systems
Module 5: Model Identification via the Least Squares Method

James H. Taylor
Department of Electrical & Computer Engineering
University of New Brunswick
Fredericton, New Brunswick, Canada E3B 5A3
e-mail: jtaylor@unb.ca
web site: www.ee.unb.ca/jtaylor/

17 February 2017

Model Identification – Overview

- Motivation
- Least squares methods (model identification) – basic concepts and procedure
  - Experiment design: generating information for model identification
  - Choosing a model type/structure (static nonlinearity, linear dynamic)
- Identifying a static nonlinearity; under- and over-fitting
- Identifying a linear dynamic model; effect of excessive noise
- Application to a chemical vapor deposition process
- Application to a two-stage crude oil separator

References:

- K. Åström and B. Wittenmark, *Computer Controlled Systems – Theory and Design*, Prentice Hall, 1984 (chapter 13)
- L. Ljung, *System Identification: Theory for the User*, Prentice Hall, 1987

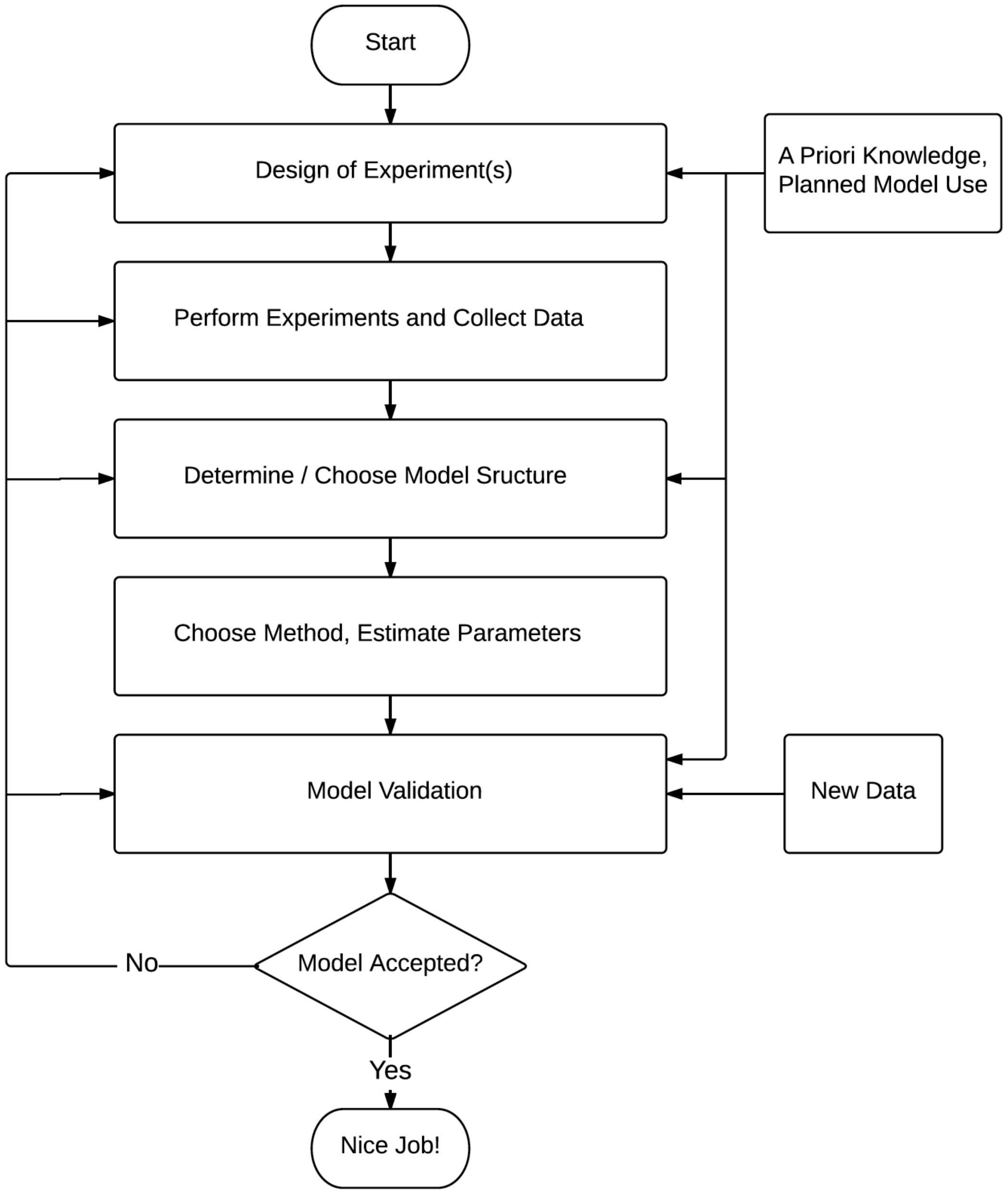
Model Identification – Motivation

- Models of processes, sensors and actuators are essential for system understanding and design
- Two modelling approaches are dominant: **physical modelling** and **model identification**
- Physical modelling may be impractical or very difficult for some (sub)systems
- Modelling a *real* system or process for a *real* industrial control implementation should produce multiple instances:
  - A realistic (usually nonlinear and complex) model for process understanding and design validation
  - One or more linearized models for controls design and analysis
- Effective model identification approaches and algorithms are well developed for producing models of **linear dynamics** and **static nonlinearities**
- **Experience** and **“common sense”** are needed to obtain good results!

Model Identification – Overview

Model identification involves four important **iterative** activities:

- Experiment planning
- Selection of model structure
- Parameter estimation
- Model validation


Start
Design of Experiment(s)
A Priori Knowledge, Planned Model Use
Perform Experiments and Collect Data
Determine / Choose Model Sructure
Choose Method, Estimate Parameters
Model Validation
New Data
Model Accepted?
No
Yes
Nice Job!

Model ID Experiment Planning

Experiment planning must be carefully considered:

- Gathering data for model ID is usually costly in terms of time and money – you should "do it right the first time" (or try to)
- Model ID of an operational process (e.g., to design a new controller) may be especially expensive due to lost production
- Points to consider for static nonlinearity model ID include:
  - Sufficient number of input values to cover the operational range
  - Check for hysteresis (recall earlier discussion)
  - Check for repeatability (also, recall earlier discussion)
- Points to consider for linear dynamic model ID include:
  - Operating points needed to cover the operational range (you can identify multiple "SSL" models at these points)
  - Amplitudes of input signals – large enough to get good data, small enough to yield approximately linear response, small and large to detect nonlinearity
  - Frequency content of input signals

Many of these issues will be discussed as we proceed.

Model Identification via Least Squares – Basic Formulation

As mentioned, we consider two problems, curve fitting experimental data to characterize the **static behaviour** of a nonlinear element, and determining the linear **dynamic models** that best approximate the dynamic response of a system with known input and output data sequences. We can use the same formulation for both problems.

Each of these problems can be set up exactly as follows: Given the input and output data sets $\{ x_i \}$, $\{ y_i \}$, $i = 1, 2, \ldots N$ containing the **experimental data** and the functions $\phi_j(x)$, j = 1, 2, $\ldots n$ which define the **candidate model structure**, then the **parameters** θ_j are to be determined (**identified**) to provide the **best fit**, i.e., to minimize the squared modelling errors. In essence, our experimental data provide N **instances** of this model relationship; generally $n << N$.

$$\hat{y} = \theta_1\phi_1(x) + \theta_2\phi_2(x) + \ldots + \theta_n\phi_n(x) \triangleq \Phi\,\theta$$

The **least squares solution** is:

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T y \triangleq \Phi^\dagger y$$

where $\Phi^\dagger$ is called the **matrix pseudoinverse** of Φ. Note that the square array $\Phi^T\Phi$ only has an inverse if the experimental data is “rich enough”; for example, if your data corresponds to an exact straight line and you try to fit a cubic candidate model then $(\Phi^T\Phi)^{-1}$ will not exist.

Static Nonlinearity Model Identification

If $\{ x_i \}$, $\{ y_i \}$ represent the static input and output measurements of a nonlinear element, then

$$\hat{y} = \begin{bmatrix} \hat{y_1} \\ \hat{y_2} \\ \vdots \\ \hat{y_N} \end{bmatrix} = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \ldots & \phi_n(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \ldots & \phi_n(x_2) \\ \vdots & \vdots & \ldots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \ldots & \phi_n(x_N) \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \triangleq \Phi\,\theta$$

As a specific case, if we are fitting an n^{th} order polynomial

$$y(x) = a_0 + a_1 x + a_2 x^2 \ldots + a_n x^n$$

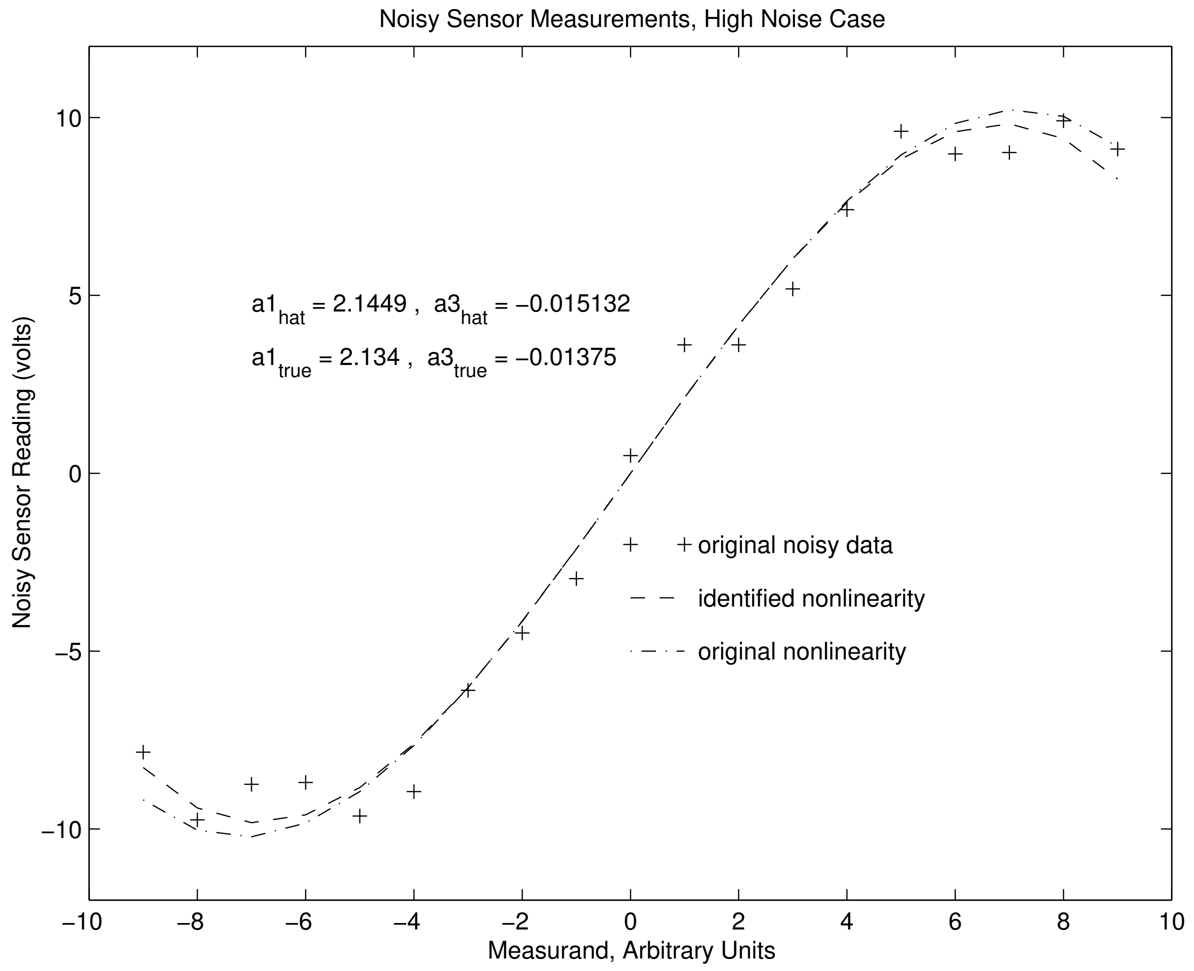
then $\phi_1(x) = 1$, $\phi_2(x) = x$, $\ldots \phi_n(x) = x^{n-1}$ and obviously $\theta_j = a_{j-1}$. We are by no means restricted to polynomial curve fitting; however, we **are** restricted to fitting models that are **linear in the parameters** θ_j.

Static Nonlinearity Model ID Example

We inspect (plot) $\{ x_i \}$, $\{ y_i \}$ and we decide to fit these points with an **odd polynomial** (let's say that we know the curve should have odd symmetry based on the physics or prior knowledge), we might start with $y(x) = a_1 x + a_2 x^3$ – in this case

$$\Phi = \begin{bmatrix} x_1 & x_1^3 \\ x_2 & x_2^3 \\ \vdots & \vdots \\ x_N & x_N^3 \end{bmatrix}, \quad \theta = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

Result:



Static Nonlinearity Model ID – MATLAB Code

We perform the polynomial fitting in MATLAB as follows:

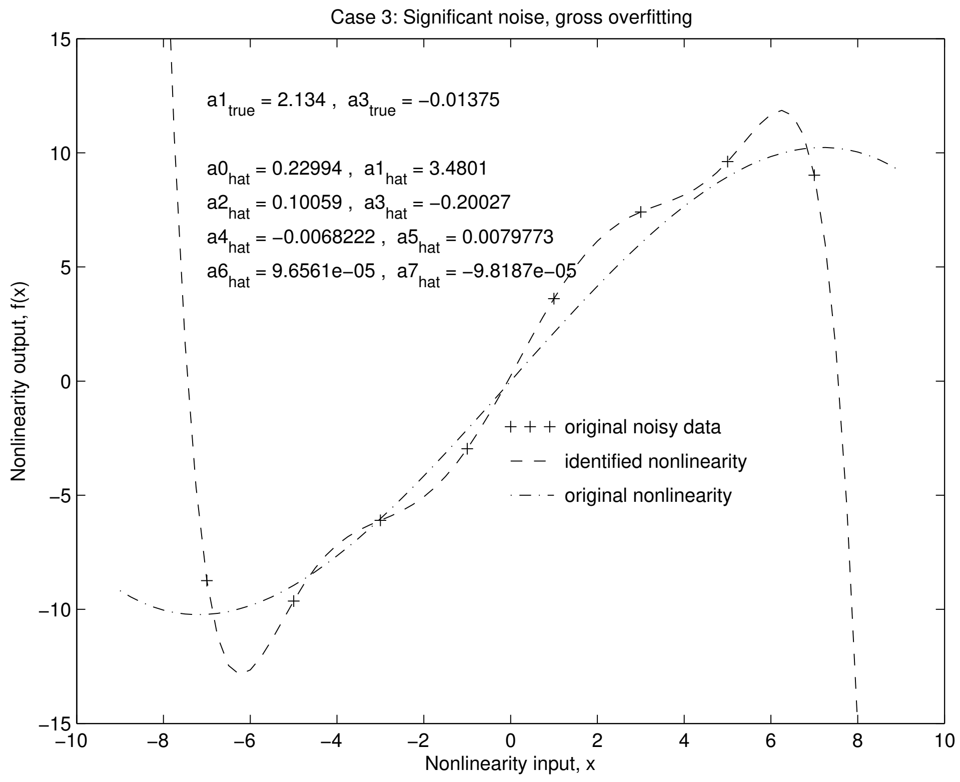
```
% matlab script to estimate a1 and a2 for fitting
% static nonlinearity data using the LS approach

% Assume the experimental x, y data are in the work space
x3 = x.*x.*x;               %% element-by-element cube
phi = [ x' x3' ];
PI = inv((phi'*phi))*phi'; %% manual implementation
theta = PI*y
```

The result is `theta` $= [\, a_1 \;\; a_2 \,]^T$ and the job is done[1].

[1]MATLAB has the pseudoinverse operator, but as you might observe via `help pinv` it is esoteric / beyond the scope of this presentation.

Static Nonlinearity Model ID (Cont'd)

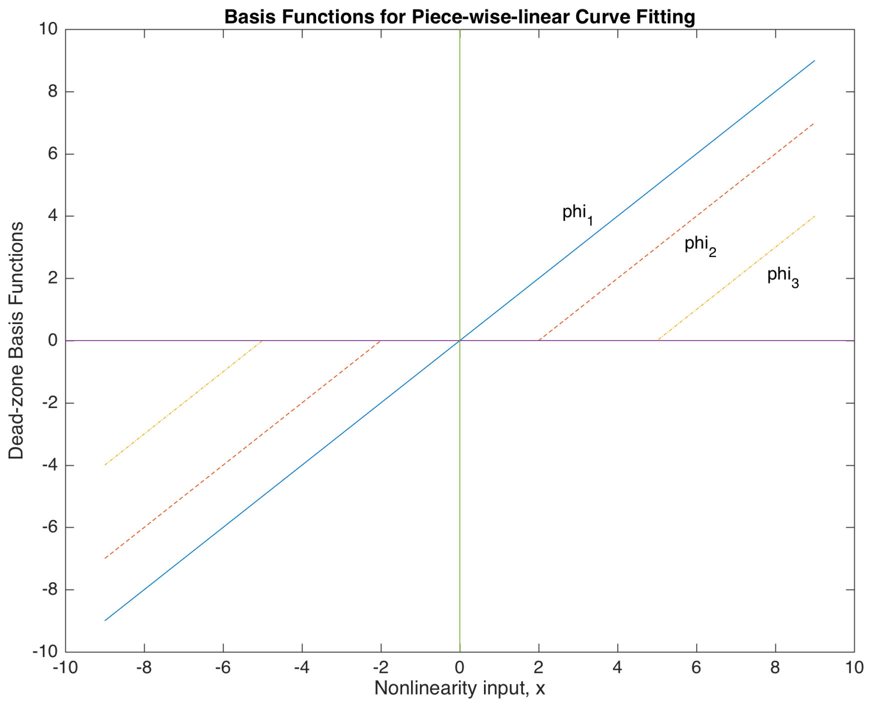
Example of **gross** overfitting:

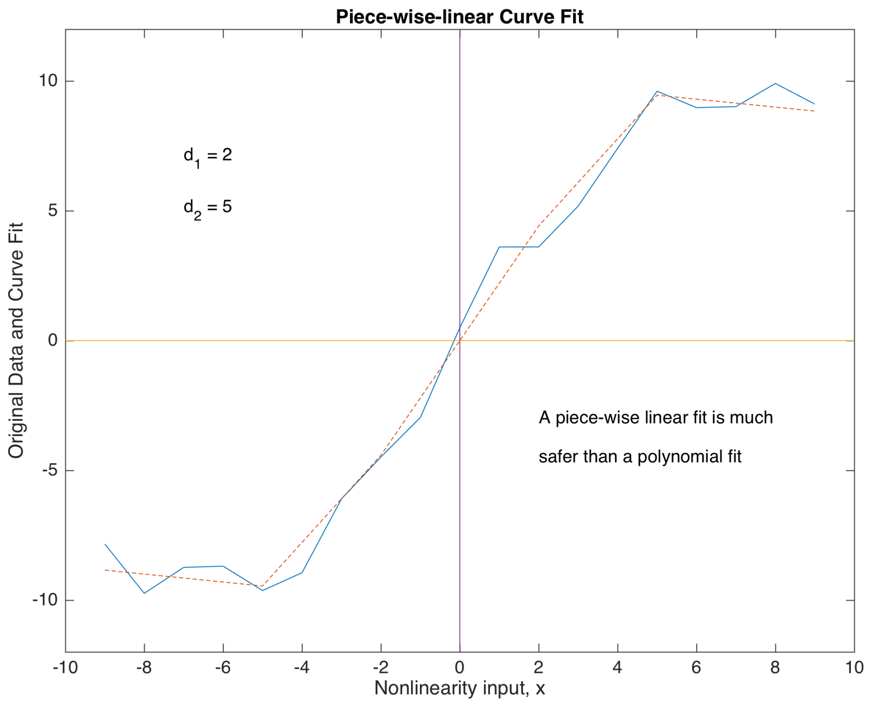


- This result fits the data **too well**, and it "goes crazy" beyond the first and last points.
- In **both of these cases** one should use the curve fit only for $-6 < x < 6$ (approximately) unless you know for certain that the nonlinearity curves down for $x > 6$ and *vice versa* for $x < -6$.
- It would usually be best to assume there is **limiting** outside the range $-6 < x < 6$, e.g., $y = 9.8\,\text{sign}(x),\ \text{sign}(x) > 6$.
- You can also identify **hysteretic nonlinearities** if you separate the experimental data into the "loading" and "unloading" cases

Static Nonlinearity Model ID (Cont'd)

But . . . we don't have to use polynomial basis functions – let's use **deadzone functions** instead





Static Nonlinearity Model ID (Cont'd)

Here is the MATLAB code:

```
%% matlab script to estimate a1 and a2 for fitting a piecewise linear
%% function to static nonlinearity data using the LS approach
%% Jim Taylor - 4 May 2017
clear; close all;
load_noisy_data;
d1 = 3; d2 = 6;
nx = length(fofx);
phi1 = x(:);
for ii = 1:nx
   if phi1(ii) < 0,
      phi2(ii) = phi1(ii) + d1;
      phi3(ii) = phi1(ii) + d2;
      if phi2(ii) > 0,  phi2(ii) = 0; end;
      if phi3(ii) > 0,  phi3(ii) = 0; end;
   else
      phi2(ii) = phi1(ii) - d1;
      phi3(ii) = phi1(ii) - d2;
      if phi2(ii) < 0,  phi2(ii) = 0; end;
      if phi3(ii) < 0,  phi3(ii) = 0; end;
   end
end
figure(2); plot(x,phi1,x,phi2,'--',x,phi3,'-.')
title ('Basis Functions for Piece-wise-linear Curve Fitting')
xlabel('Nonlinearity input, x')
ylabel('Dead-zone Basis Functions')
phi = [ phi1(:) phi2(:) phi3(:) ];
PI = inv((phi'*phi))*phi';     %% manual implementation
theta = PI*fofx'; y_id = phi*theta;
figure(3); plot(x,fofx,x,y_id,'--');
title ('Piece-wise-linear Curve Fit')
xlabel('Nonlinearity input, x')
ylabel('Original Data and Curve Fit')
```

Linear Dynamic Model Identification

It is not possible to obtain a linear dynamic model directly; rather it is necessary to identify a **linear difference equation** or **discrete-time model** and transform that into a continous-time model. First, here is the formulation for the discrete-time case:

If $\{\, u_i \,\}$, $\{\, y_i \,\}$ represent input and output data sequences for a dynamic system and we wish to replicate the dynamic behaviour with a first-order difference equation,

$$\hat{y_k} = -a\, y_{k-1} + b\, u_{k-1}$$

then we have $N-1$ instances of the model relationship:

$$\hat{y} = \begin{bmatrix} y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} -y_1 & u_1 \\ -y_2 & u_2 \\ \vdots & \vdots \\ -y_{N-1} & u_{N-1} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \triangleq \Phi\, \theta$$

where, of course, $\theta_1 = a$, $\theta_2 = b$.

We create the array Φ as above, so our model is again concisely written $\hat{y} = \Phi\, \theta$ and we can solve the parameter identification problem easily using the least squares approach. Note that we have only $N-1$ instances above (not a problem).

Linear Dynamic Model ID – MATLAB Code

Assume that unit step response data has been obtained with a single experiment and that $\{ u_i \}$, $\{ y_i \}$ are loaded into the MATLAB work space in the form of vectors `uk` (known sampled input data) and `ynk` (output data with measurement noise). MATLAB code to carry out the task is as follows:

```
% matlab script to estimate a and b for fitting
% step-response data with a first-order diff. eq.

 yc = ynk(:);       % make yc a col vector
 len = length(yc); % determine number of samples
 uc = uk(:);        % uk = input data
 Phi = [ -yc uc ];
 % eliminate the first element of y and the last
 % row of Phi: to create N - 1 model instances
 yc(1) = [];       % first element is "empty"
 Phi(len,:) = []; % last row is "empty"
 % calculate the pseudo-inverse, estimate theta:
 PI = inv((Phi'*Phi))*Phi';
 theta = PI*yc; a = theta(1); b = theta(2)
```

Linear Dynamic Model ID – Example

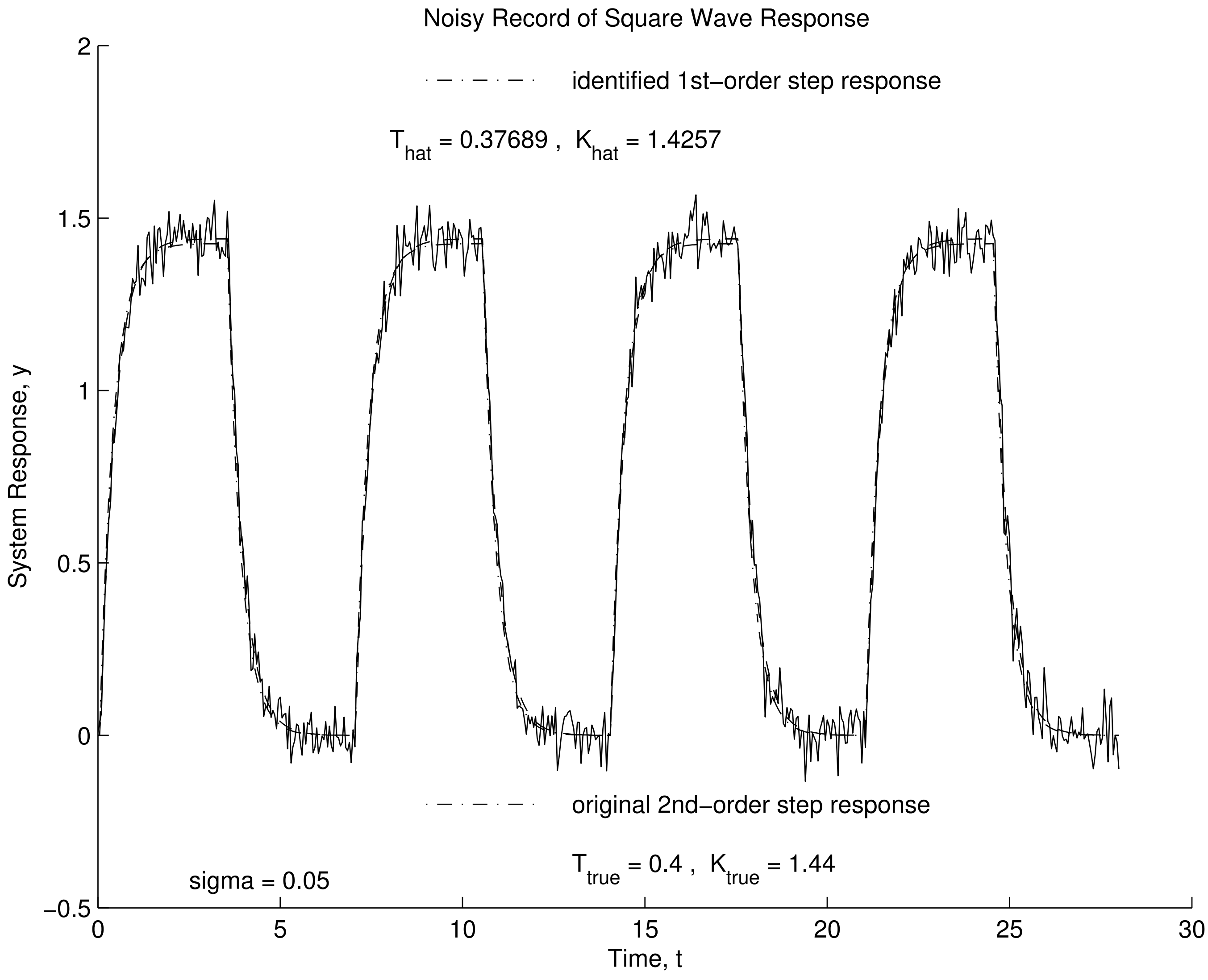
We complete the task by transforming the difference equation into a *continuous-time* model as follows:

$$T = -h/\ln(-a)\,;\;\; K = \theta_2/(1+a)$$

We augment the script above to carry this out:

```
h = tk(2) - tk(1);     %% determine the sampling time
T = - h/log(-theta(1)) %%  or: theta(1) = exp(-h/T)
K = theta(2)/(1 + theta(1)) %%  or: theta(2) = K(1 - exp(-h/T))
```

This step allows us to check the quality of our model identification procedure, which turned out rather well:



Discrete- to Continuous-time Transformation

In case you are interested, here is a derivation of the discrete- to continuous-time transformation:

Given a continuous-time system defined by the transfer function $G_s(s) = K/(1 + Ts)$, or, in ordinary differential equation form:

$$\dot{y} = (Ku - y)/T \tag{1}$$

which you want to approximate by a discrete-time system of the form

$$y_k = -\theta_1 y_{k-1} + \theta_2 u_{k-1}\,; \;\; h = \text{sampling time} \tag{2}$$

(note that the model parameters are also called $\theta_1 = a$ and $\theta_2 = b$ in least squares notation). We discretize Eqn. (1) by assuming that u is constant over the interval $t_{k-1} < t \le t_k$ and use the standard (convolution integral) solution,

$$y(t_k) = y(t_{k-1})e^{-h/T} + K(1 - e^{-h/T})u_{k-1} \tag{3}$$

The relations between $(K,\ T)$ and $(\theta_1,\ \theta_2)$ are thus:

$$\theta_1 = -e^{-h/T} \tag{4}$$

$$\theta_2 = K\,(1 - e^{-h/T}) \tag{5}$$

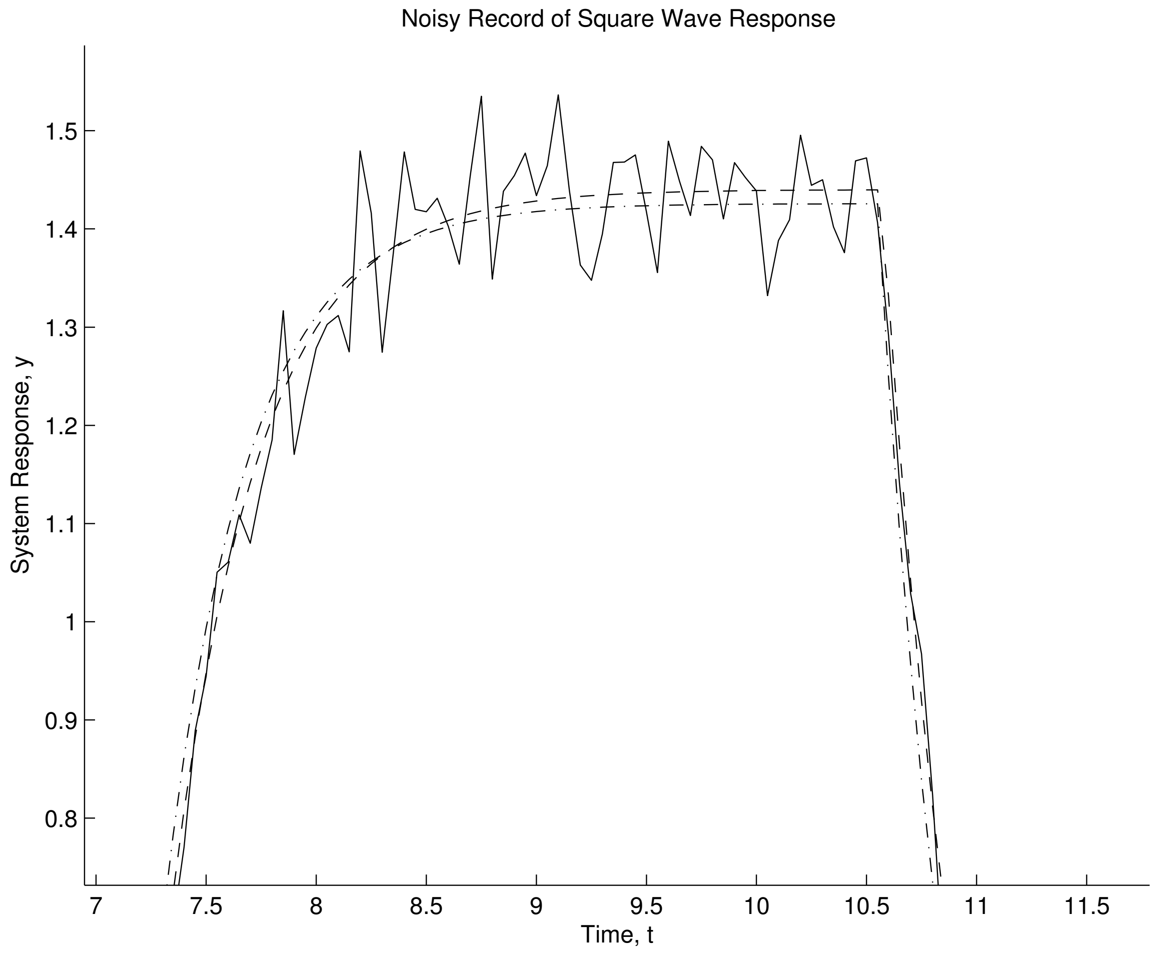
or, conversely,

$$T = -h/\ln(-\theta_1)\,; \;\; K = \theta_2/(1 + \theta_1) \tag{6}$$

(note '`log`' = ln or natural log in MATLAB).

Linear Dynamic Model ID Results (Cont'd)

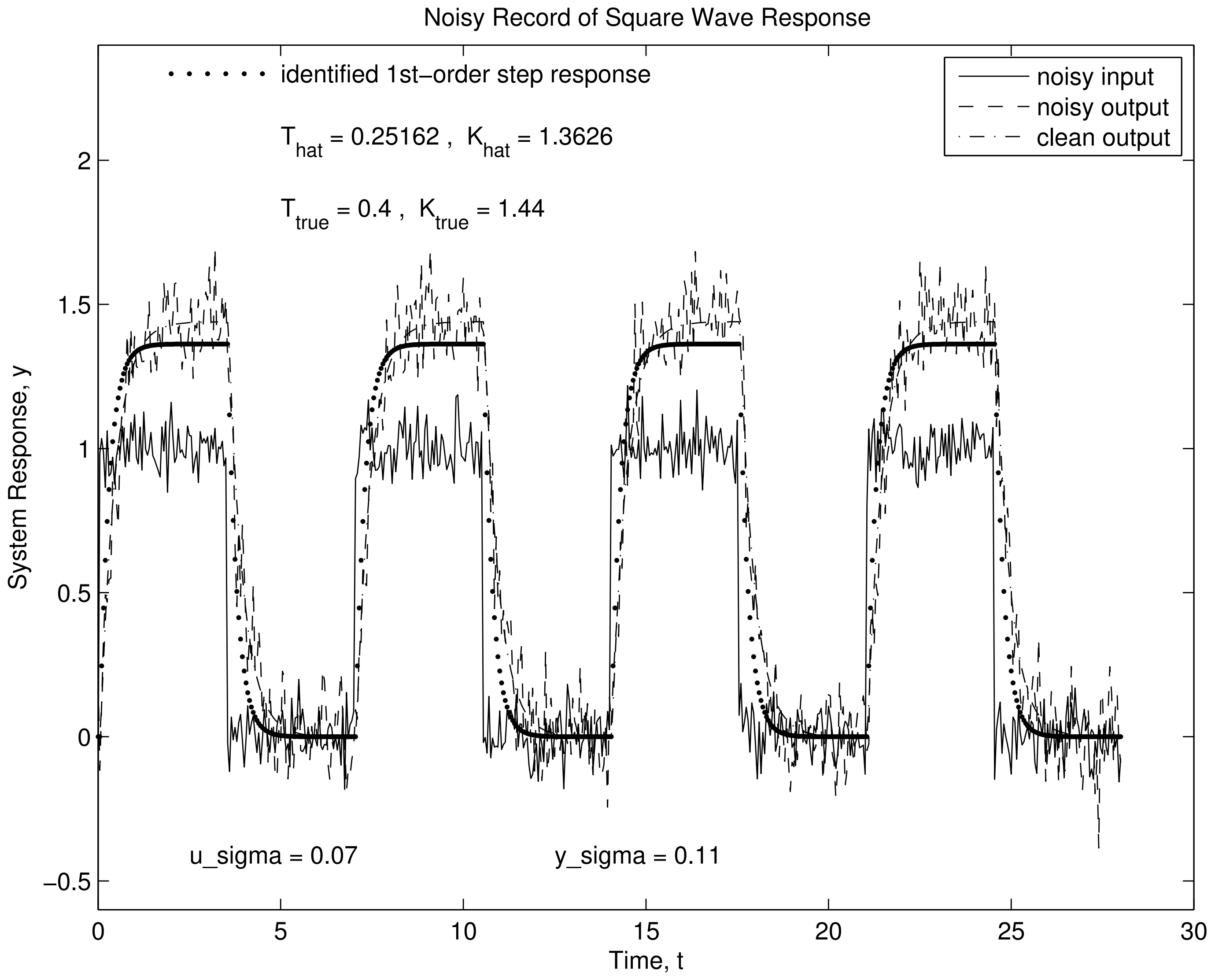
Four cycles of square wave were used in this example; longer data sequences tend to give better results. Let's inspect the quality of fit more closely:



Small errors in estimating the gain and time constant are evident; considering the noise amplitude this is quite good.

Linear Dynamic Model ID Results (Cont'd)

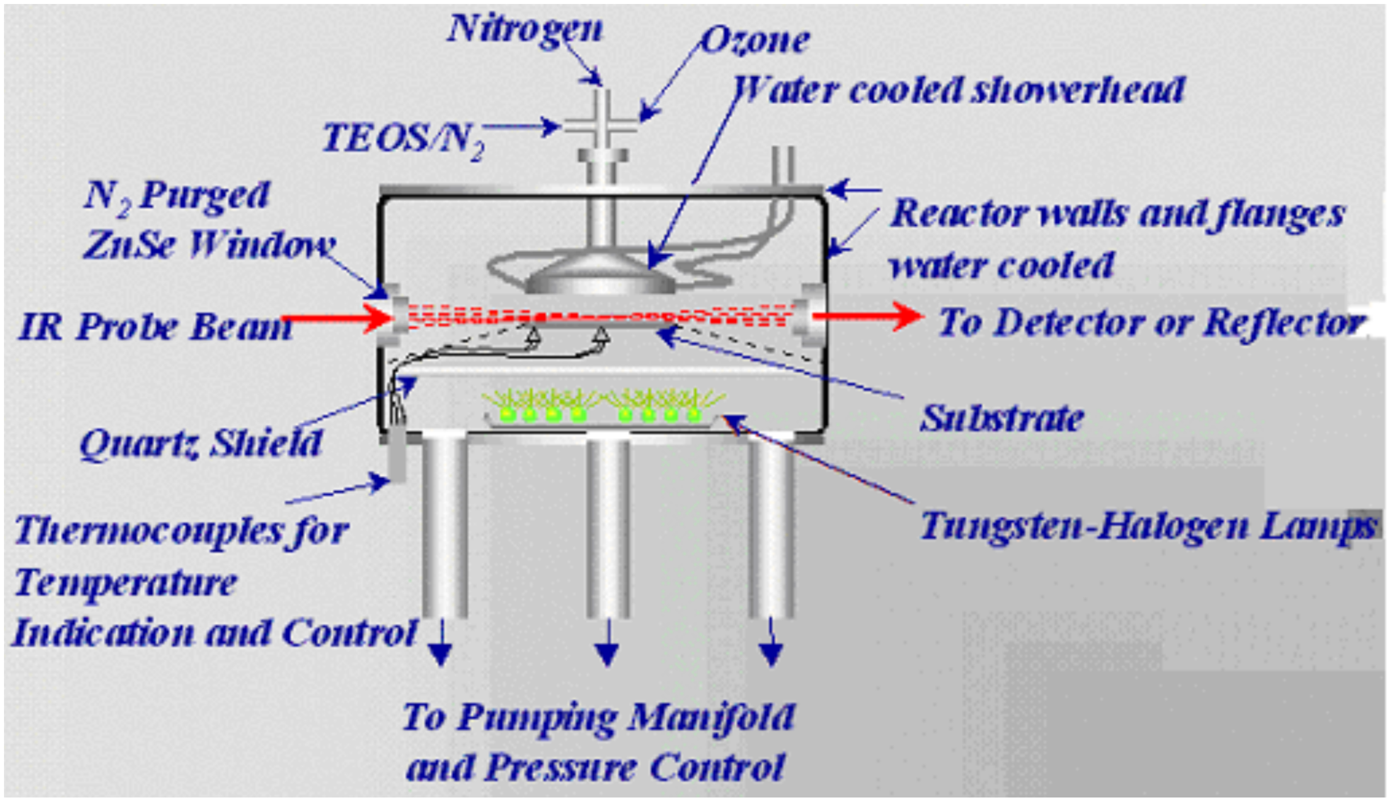
Finally, even with high measurement noise levels on both the input and output data sequences we still obtain good results:



Higher noise levels decrease the accuracy of the model ID process

Linear Dynamic Model ID – Real World Project

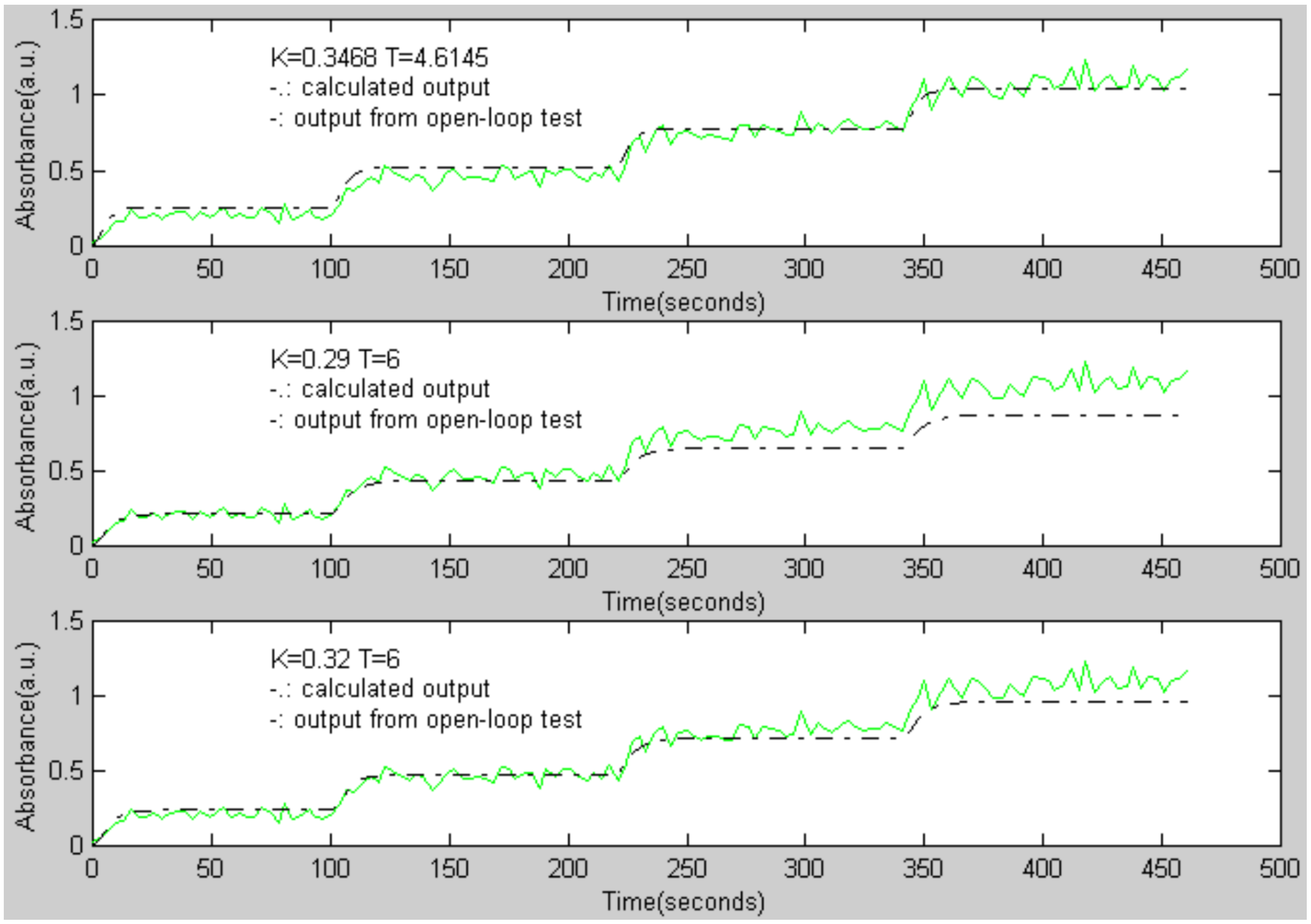
Mr. Zhao Xiaozhong, a Masters student of mine, developed a controller for a semiconductor chemical vapor deposition process, and used least squares model ID in order to understand its dynamic behaviour:



In this work we showed that least squares model ID worked well, **and** that the process was somewhat nonlinear

This work was done for a small start-up company, where the concept was to improve the semiconductor yield by monitoring the chemistry above the wafer using laser spectography

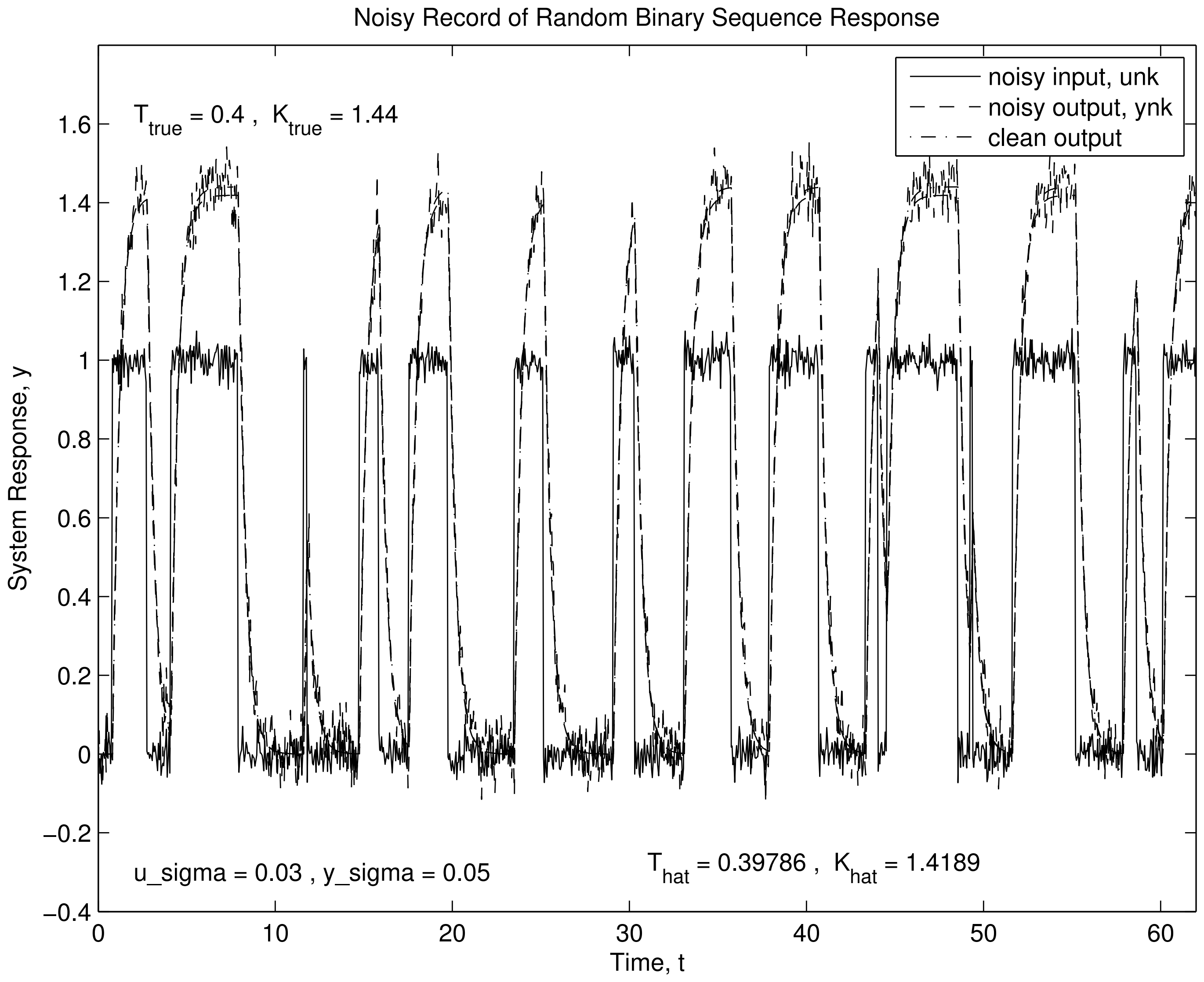
Linear Dynamic Model ID – Real World Project (Cont'd)



We determined that the process was nonlinear based on the observation that the first trace (the result from LS model ID) shows that the fit for the last steps is good but for the first step the gain K is too high and the time constant T is too low. For the second case we manually reduced K and increased T, significantly improving the fit for the first step but degrading the fit for the last step. The third case was an attempt to find a compromise

Linear Dynamic Model ID Using a More "Exciting" Input

As mentioned, square-wave excitation does not excite the system dynamics very well – higher-frequencies are rather weak. Random binary sequences have better higher-frequency content, and thus produce more accurate models, especially if you want to identify dynamic models of order greater than one.



Advanced Linear Dynamic Model ID Methods

The technique presented here is the most rudimentary linear dynamic model ID approach using statistical methods – it is a "black-box" approach, meaning that no assumptions are made (no knowledge is available) about the **structure**. Many linear dynamic model ID approaches are of this type.

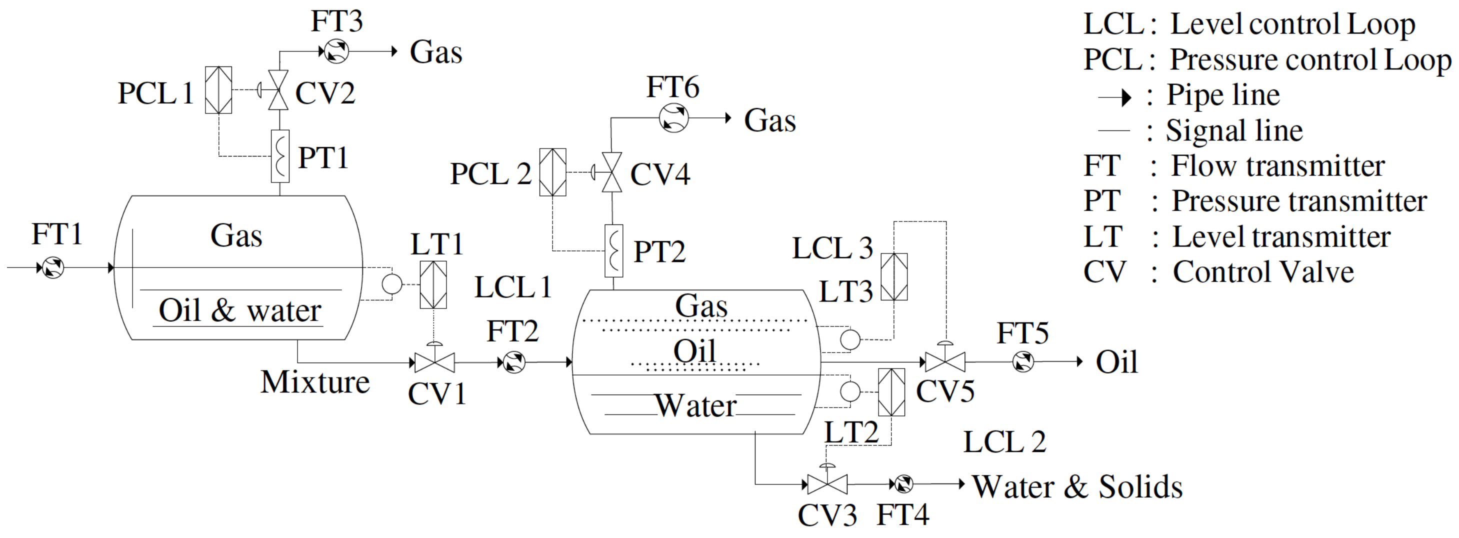
Other more advanced techniques utilize different approaches in several aspects; most importantly:

- **Model formulations** – we used a first-order difference equation with noise injected in the output data; more general models, e.g., the autoregressive moving average model, will produce better results when the noise situation is not so simple;
- **Performance criteria** – we used minimizing the square of the fitting error; more sophisticated criteria, e.g., the prediction error/maximum likelihood method are superior; and
- **Different types of input** – we used square wave signals, which have litte high frequency content; more "exciting" inputs, e.g., pseudo random binary sequences or generalized binary noise, provide full spectrum excitation.

Illuminating these issues is beyond the scope of this course; these are mentioned to provide a context for an example that follows.

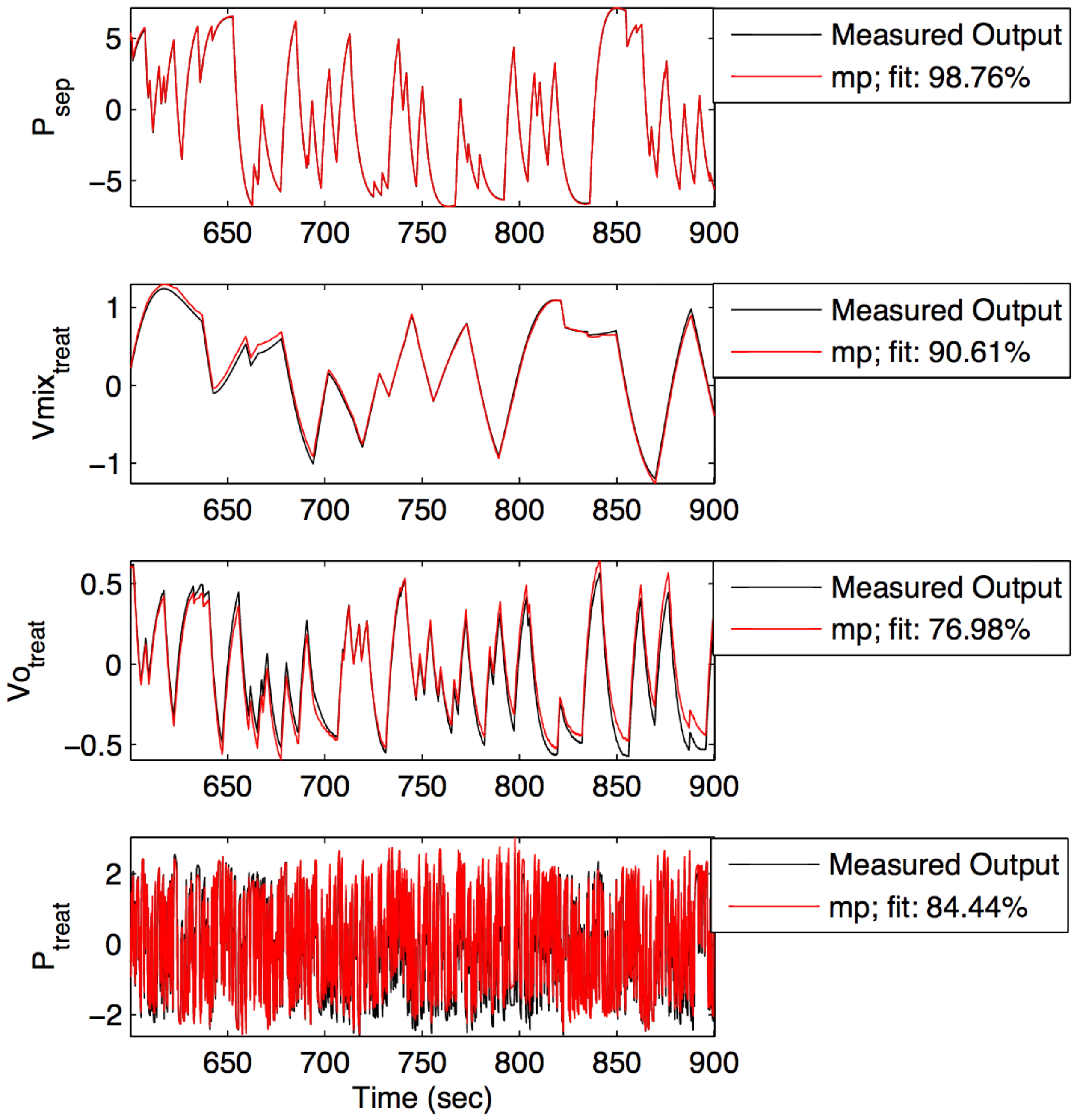
An Advanced Real World Linear Model ID Example

Dr. Maira Omana, a PhD student of mine, developed a fault detection, isolation and accommodation (FDIA) procedure and applied it to a complex process found in the petroleum industry, a **two-stage crude oil separator**:



Her FDIA procedure required a linearized model of the process, and Dr. Omana devised a much more sophisticated algorithmic approach than presented above. The five controller input signals were **generalized binary noise** and the resulting input/output signals were analyzed using the **prediction error/maximum likelihood method** - here is a typical result (next slide):

An Advanced Real World Example (Cont'd)



The resulting linearized model proved to serve well in her development of a powerful new FDIA procedure. Creating a realistic *physics-based* model took a different PhD student and me **one year**