

24S2 EE6225

System Identification

Associate Professor Ling KV

School of Electrical and Electronic Engineering

Nanyang Technological University

- System Identification is about building mathematical models of dynamical system using measured input-output data.
- A common method is the so-called **Prediction-error method (PEM)**.
- The basic idea behind the PEM is very simple. Describe the model as a predictor of the next output (i.e. the one-step ahead predictor):

$$\hat{y}(t|\theta) = f(Z^{t-1}, \theta).$$

- Express the one-step ahead predictor of the output, as a function of past observed data, parameterized in terms of a finite-dimensional parameter vector.
- Then determine an estimate of the parameter vector so that the prediction errors are minimized in some sense (e.g. least squares).

The PEM has a number of advantages:

- It can be applied to a wide spectrum of model parameterizations (see Section 2).
- It gives models with excellent asymptotic properties, due to its kinship with maximum likelihood (see Sections 4 and 5).
- It can handle systems that operate in closed loop (the input is partly determined as output feedback, when the data are collected) without any special tricks and techniques (see Section 4).

It also has some drawbacks:

- It requires an explicit parameterization of the model. To estimate, say, an arbitrary linear, fifth-order model, some kind of parameterization, covering all fifth-order models, must be introduced.
- The search for the parameters that gives the best output prediction fit may be laborious and involve search surfaces that have many local minima.

Example

- Consider a physical system, with observed input and output signals obtained from test flights

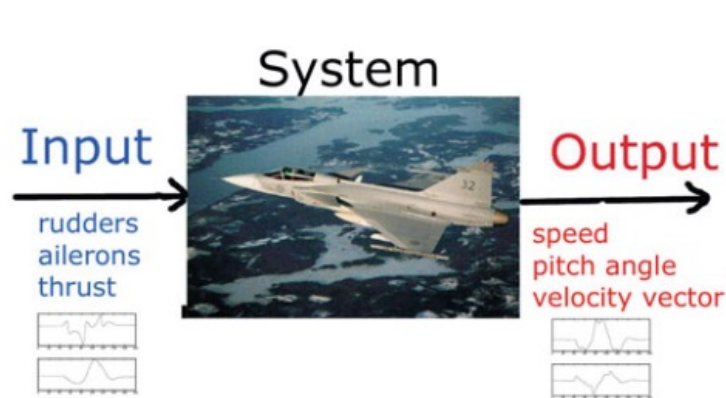
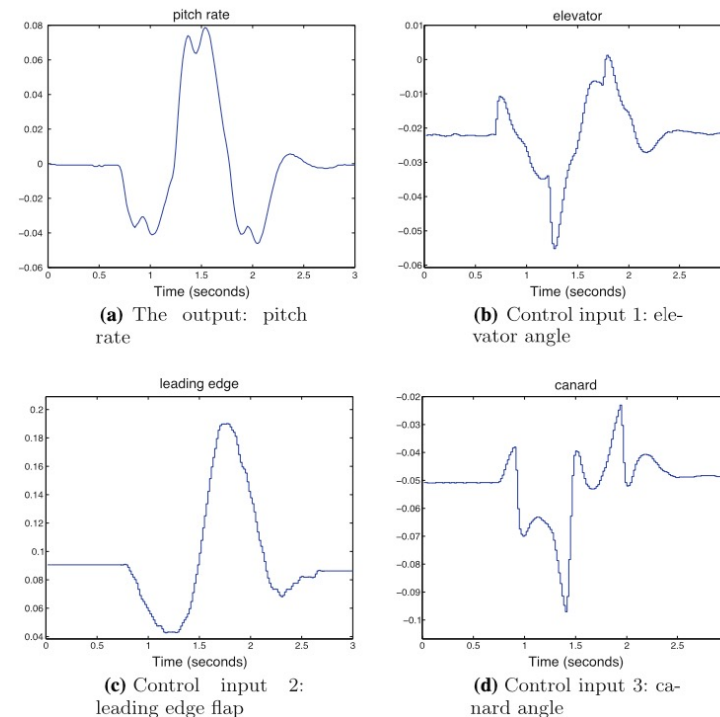


Fig. 1 The Swedish aircraft Gripen



- In order to be able to simulate the aircraft and to design an effective autopilot, it is necessary to understand how, in this case, the pitch rate is affected by the three inputs. We need mathematical expressions for this. A fair amount of knowledge exists about aircraft dynamics, but let us just try a simple difference equation relation.

$$\begin{aligned} y(t) = & a_1 y(t-1) - a_2 y(t-2) - a_3 y(t-3) \\ & + b_{1,1} u_1(t-1) + b_{1,2} u_1(t-2) \\ & + b_{2,1} u_2(t-1) + b_{2,2} u_2(t-2) \\ & + b_{3,1} u_3(t-1) + b_{3,2} u_3(t-2) \end{aligned} \quad (1)$$

- Here, we denote the output, the pitch rate at sample t as $y(t)$ and the three control inputs at the same time as $u_k(t)$, $k = 1, 2, 3$.
- We adjust the parameters a and b to fit the observed data as well as possible by a common least squares fit, giving ...

- We may note that this model is unstable – it has a pole in 1.0026, but this is in order because the pitch channel is unstable at the velocity and altitude in question.

$$\begin{aligned}y(t) - 1.15y(t-1) + 0.50y(t-2) - 0.35y(t-3) \\= -0.54u_1(t-1) + 0.04u_1(t-2) \\+ 0.15u_2(t-1) + 0.16u_2(t-2) \\+ 0.16u_3(t-1) + 0.07u_3(t-2)\end{aligned}\tag{2}$$

- How can we test if this model is OK?

... Example

- Since we used only half of the observed data for the estimation, we can test the model on the whole data record.
- Since the model is unstable and thus simulation is difficult, it is natural to let the model prediction future outputs, say five samples ahead and compared with the measured outputs.
- We see that the simple model provides quite a reasonable prediction over data it has not seen before. This could conceivably be improved if more elaborate model structures were tried out.

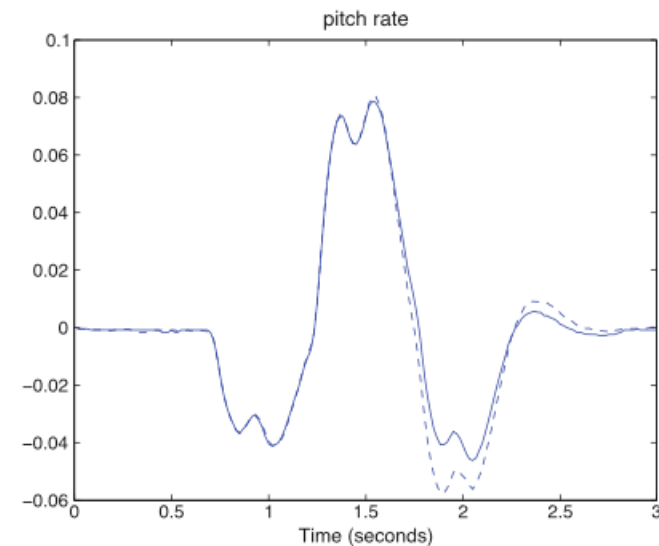


Fig. 3 The measured output (*solid line*) compared to the five step ahead predicted one (*dashed line*)

Basic Questions About System Identification (from Matlab SysID toolbox)

Basic Questions About System Identification

What is System Identification?

System Identification allows you to build mathematical models of a dynamic system based on measured data.

How is that done?

Essentially by adjusting parameters within a given model until its output coincides as well as possible with the measured output.

How do you know if the model is any good?

A good test is to take a close look at the model's output compared to the measured one on a data set that wasn't used for the fit ("Validation Data").

Can the quality of the model be tested in other ways?

It is also valuable to look at what the model couldn't reproduce in the data ("the residuals"). This should not be correlated with other available information, such as the system's input.

What models are most common?

The techniques apply to very general models. Most common models are difference equations descriptions, such as ARX and ARMAX models, as well as all types of linear state-space models.

Do you have to assume a model of a particular type?

For parametric models, you have to specify the structure. However, if you just assume that the system is linear, you can directly estimate its impulse or step response using Correlation Analysis or its frequency response using Spectral Analysis. This allows useful comparisons with other estimated models.

What does the System Identification Toolbox contain?

It contains all the common techniques to adjust parameters in all kinds of linear models. It also allows you to examine the models' properties, and to check if they are any good, as well as to preprocess and polish the measured data.

Isn't it a big limitation to work only with linear models?

No, actually not. Most common model nonlinearities are such that the measured data should be nonlinearly transformed (like squaring a voltage input if you think that it's the power that is the stimuli). Use physical insight about the system you are modeling and try out such transformations on models that are linear in the new variables, and you will cover a lot!

How do I get started?

If you are a beginner, browse through Chapter and then try out a couple of the data sets that come with the toolbox. Use the graphical user interface (GUI) and check out the built-in help functions to understand what you are doing.

Is this really all there is to System Identification?

Actually, there is a huge amount written on the subject. Experience with real data is the driving force to understand more. It is important to remember that any estimated model, no matter how good it looks on your screen, has only picked up a simple reflection of reality. Surprisingly often, however, this is sufficient for rational decision making.

System Identification
Toolbox
For Use with MATLAB[®]
Licensee: Lincat

Computation
Visualization
Programming

User's Guide

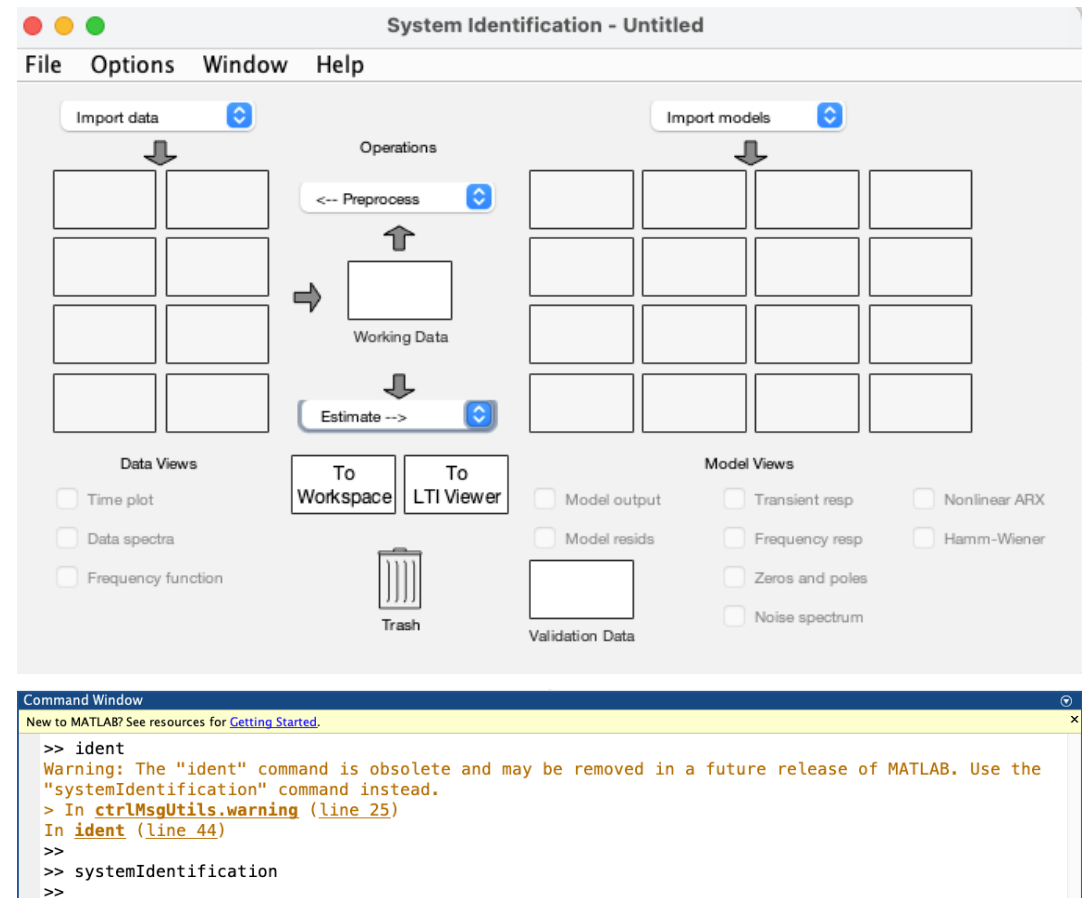
https://www.ladispe.polito.it/corsi/SIEF/material/MATLAB_S_I_Toolbox.pdf

Common Terms Used in SysID

Common Terms Used in System Identification

This section defines some of the terms that are frequently used in System Identification.

- **Estimation Data** is the data set that is used to fit a model to data. In the GUI this is the same as the **Working Data**.
- **Validation Data** is the data set that is used for model validation purposes. This includes simulating the model for these data and computing the residuals from the model when applied to these data.
- **Model Views** are various ways of inspecting the properties of a model. They include looking at zeros and poles, transient and frequency response, and similar things.
- **Data Views** are various ways of inspecting properties of data sets. A most common and useful thing is just to plot the data and scrutinize it. So-called *outliers* could be detected then. These are unreliable measurements, perhaps arising from failures in the measurement equipment. The frequency contents of the data signals, in terms of periodograms or spectral estimates, is also most revealing to study.
- **Model Sets or Model Structures** are families of models with adjustable parameters. **Parameter Estimation** amounts to finding the “best” values of these parameters. The System Identification problem amounts to finding both a good model structure and good numerical values of its parameters.
- **Parametric Identification Methods** are techniques to estimate parameters in given model structures. Basically it is a matter of finding (by numerical search) those numerical values of the parameters that give the best agreement between the model's (simulated or predicted) output and the measured one.
- **Nonparametric Identification Methods** are techniques to estimate model behavior without necessarily using a given parametrized model set. Typical nonparametric methods include **Correlation analysis**, which estimates a system's impulse response, and **Spectral analysis**, which estimates a system's frequency response.
- **Model Validation** is the process of gaining confidence in a model. Essentially this is achieved by “twisting and turning” the model to scrutinize all aspects of it. Of particular importance is the model's ability to reproduce the behavior of the Validation Data sets. Thus it is important to inspect the properties of the residuals from the model when applied to the Validation Data.



The Basic Steps of System Identification

The System Identification problem is to estimate a model of a system based on observed input-output data. Several ways to describe a system and to estimate such descriptions exist. This section gives a brief account of the most important approaches.

The procedure to determine a model of a dynamical system from observed input-output data involves three basic ingredients:

- The input-output data
- A set of candidate models (the model structure)
- A criterion to select a particular model in the set, based on the information in the data (the identification method)

The identification process amounts to repeatedly selecting a model structure, computing the best model in the structure, and evaluating this model's properties to see if they are satisfactory. The cycle can be itemized as follows:

- 1 Design an experiment and collect input-output data from the process to be identified.
- 2 Examine the data. Polish it so as to remove trends and outliers, select useful portions of the original data, and apply filtering to enhance important frequency ranges.
- 3 Select and define a model structure (a set of candidate system descriptions) within which a model is to be found.
- 4 Compute the best model in the model structure according to the input-output data and a given criterion of fit.
- 5 Examine the obtained model's properties
- 6 If the model is good enough, then stop; otherwise go back to Step 3 to try another model set. Possibly also try other estimation methods (Step 4) or work further on the input-output data (Steps 1 and 2).

The System Identification Toolbox offers several functions for each of these steps.

For Step 2 there are routines to plot data, filter data, and remove trends in data.

For Step 3 the System Identification Toolbox offers a variety of nonparametric models, as well as all the most common black-box input-output and state-space structures, and also general tailor-made linear state-space models in discrete and continuous time.

For Step 4 general prediction error (maximum likelihood) methods as well as instrumental variable methods and sub-space methods are offered for parametric models, while basic correlation and spectral analysis methods are used for nonparametric model structures.

To examine models in Step 5, many functions allow the computation and presentation of frequency functions and poles and zeros, as well as simulation and prediction using the model. Functions are also included for transformations between continuous-time and discrete-time model descriptions and to formats that are used in other MATLAB toolboxes, like the Control System Toolbox and the Signal Processing Toolbox.

Three basic ingredients:
input/output data, model structure, selection criteria

SysID cycle:

Design expt to collect data, Examine/clean up data,
Select model structure, Compute/fit data to model,
Validate model, repeat to improve model.

A Startup Identification Procedure

There are no standard and secure routes to good models in System Identification. Given the number of possibilities, it is easy to get confused about what to do, what model structures to test, and so on. This section describes one route that often works well, but there are no guarantees. The steps refer to functions within the GUI, but you can also go through them in command mode. See Chapter for the basic commands.

Step 1 Looking at the Data

Plot the data. Look at them carefully. Try to see the dynamics with your own eyes. Can you see the effects in the outputs of the changes in the input? Can you see nonlinear effects, like different responses at different levels, or different responses to a step up and a step down? Are there portions of the data that appear to be “messy” or carry no information. Use this insight to select portions of the data for estimation and validation purposes.

Do physical levels play a role in your model? If not, detrend the data by removing their mean values. The models will then describe how changes in the input give changes in output, but not explain the actual levels of the signals. This is the normal situation.

The default situation, with good data, is that you detrend by removing means, and then select the first half or so of the data record for estimation purposes, and use the remaining data for validation. This is what happens when you apply Quickstart under the pop-up menu Preprocess in the main ident window.

Step 2 Getting a Feel for the Difficulties

Apply Quickstart under pop-up menu Estimate in the main ident window. This will compute and display the spectral analysis estimate and the correlation analysis estimate, as well as a fourth order ARX model with a delay

estimated from the correlation analysis and a default order state-space model computed by `n4sid`. This gives three plots. Look at the agreement between the

- Spectral Analysis estimate and the ARX and state-space models' frequency functions
- Correlation Analysis estimate and the ARX and state-space models' transient responses
- Measured Validation Data output and the ARX and state-space models' simulated outputs

If these agreements are reasonable, the problem is not so difficult, and a relatively simple linear model will do a good job. Some fine tuning of model orders, and noise models have to be made and you can proceed to Step 4. Otherwise go to Step 3.

Step 3 Examining the Difficulties

There may be several reasons why the comparisons in Step 2 did not look good. This section discusses the most common ones, and how they can be handled:

Model Unstable

The ARX or state-space model may turn out to be unstable, but could still be useful for control purposes. Change to a 5- or 10-step ahead prediction instead of simulation in the Model Output View.

Feedback in Data

If there is feedback from the output to the input, due to some regulator, then the spectral and correlations analysis estimates are not reliable. Discrepancies between these estimates and the ARX and state-space models can therefore be disregarded in this case. In the Model Residuals View of the parametric models, feedback in data can also be visible as correlation between residuals and input for negative lags.

Noise Model

If the state-space model is clearly better than the ARX model at reproducing the measured output, this is an indication that the disturbances have a substantial influence, and it will be necessary to model them carefully.

... A Startup SysID Procedure

Model Order

If a fourth order model does not give a good **Model Output** plot, try eighth order. If the fit clearly improves, it follows that higher order models will be required, but that linear models could be sufficient.

Additional Inputs

If the **Model Output** fit has not significantly improved by the tests so far, think over the physics of the application. Are there more signals that have been, or could be, measured that might influence the output? If so, include these among the inputs and try again a fourth order ARX model from all the inputs. (Note that the inputs need not at all be control signals, anything measurable, including disturbances, should be treated as inputs).

Nonlinear Effects

If the fit between measured and model output is still bad, consider the physics of the application. Are there nonlinear effects in the system? In that case, form the nonlinearities from the measured data. This could be as simple as forming the product of voltage and current measurements, if you realize that it is the electrical power that is the driving stimulus in, say, a heating process, and temperature is the output. This is of course application dependent. It does not take very much work, however, to form a number of additional inputs by reasonable nonlinear transformations of the measured ones, and just test if inclusion of them improves the fit.

Still Problems?

If none of these tests leads to a model that is able to reproduce the Validation Data reasonably well, the conclusion might be that a sufficiently good model cannot be produced from the data. There may be many reasons for this. The most important one is that the data simply do not contain sufficient information, e.g., due to bad signal to noise ratios, large and nonstationary disturbances, varying system properties, etc. The reason may also be that the system has some quite complicated nonlinearities, which cannot be realized on physical grounds. In such cases, nonlinear, black box models could be a solution. Among the most used models of this character are the Artificial Neural Networks (ANN).

Otherwise, use the insights of which inputs to use and which model orders to expect and proceed to Step 4.

Step 4 Fine Tuning Orders and Noise Structures

For real data there is no such thing as a “correct model structure.” However, different structures can give quite different model quality. The only way to find this out is to try out a number of different structures and compare the properties of the obtained models. There are a few things to look for in these comparisons:

Fit Between Simulated and Measured Output

Keep the **Model Output View** open and look at the fit between the model's simulated output and the measured one for the Validation Data. Formally, you could pick that model, for which this number is the lowest. In practice, it is better to be more pragmatic, and also take into account the model complexity, and whether the important features of the output response are captured.

Residual Analysis Test

You should require of a good model, that the cross correlation function between residuals and input does not go significantly outside the confidence region. A clear peak at lag k shows that the effect from input $u(t-k)$ on $y(t)$ is not properly described. A rule of thumb is that a slowly varying cross correlation function outside the confidence region is an indication of too few poles, while sharper peaks indicate too few zeros or wrong delays.

Pole Zero Cancellations

If the pole-zero plot (including confidence intervals) indicates pole-zero cancellations in the dynamics, this suggests that lower order models can be used. In particular, if it turns out that the orders of ARX models have to be increased to get a good fit, but that pole-zero cancellations are indicated, then the extra poles are just introduced to describe the noise. Then try ARMAX, OE, or BJ model structures with an A or F polynomial of an order equal to that of the number of noncanceled poles.

What Model Structures Should be Tested?

Well, you can spend any amount of time to check out a very large number of structures. It often takes just a few seconds to compute and evaluate a model in a certain structure, so that you should have a generous attitude to the testing. However, experience shows that when the basic properties of the system's behavior have been picked up, it is not much use to fine tune orders in absurdum just to press the fit by fractions of percents.

Many ARX models: There is a very cheap way of testing many ARX structures simultaneously. Enter in the **Orders** text field many combinations of orders, using the colon (":") notation. When you select **Estimate**, models for all combinations (easily several hundreds) are computed and their (prediction error) fit to Validation Data is shown in a special plot. By clicking in this plot the best models with any chosen number of parameters will be inserted into the Model Board, and evaluated as desired.

Many State-space models: A similar feature is also available for black-box state-space models, estimated using `n4sid`. When a good order has been found, try the PEM estimation method, which often improves on the accuracy.

ARMAX, OE, and BJ models: Once you have a feel for suitable delays and dynamics orders, it is often useful to try out ARMAX, OE, and/or BJ with these orders and test some different orders for the noise transfer functions (C and D). Especially for poorly damped systems, the OE structure is suitable.

There is a quite extensive literature on order and structure selection, and anyone who would like to know more should consult the references.

Multivariable Systems

Systems with many input signals and/or many output signals are called *multivariable*. Such systems are often more challenging to model. In particular systems with several outputs could be difficult. A basic reason for the difficulties is that the couplings between several inputs and outputs lead to more complex models. The structures involved are richer and more parameters will be required to obtain a good fit.

Available Models

The System Identification Toolbox as well as the GUI handles general, linear multivariable models. All earlier mentioned models are supported in the single output, multiple input case. For multiple outputs ARX models and state-space models are covered. Multi-output ARMAX and OE models are covered via state-space representations: ARMAX corresponds to estimating the K-matrix, while OE corresponds to fixing K to zero. (These are pop-up options in the GUI model order editor.)

Generally speaking, it is preferable to work with state-space models in the multivariable case, since the model structure complexity is easier to deal with. It is essentially just a matter of choosing the model order.

Working with Subsets of the Input Output Channels

In the process of identifying good models of a system, it is often useful to select subsets of the input and output channels. Partial models of the system's behavior will then be constructed. It might not, for example, be clear if all measured inputs have a significant influence on the outputs. That is most easily tested by removing an input channel from the data, building a model for how the output(s) depends on the remaining input channels, and checking if there is a significant deterioration in the model output's fit to the measured one. See also the discussion under Step 3 above.

Generally speaking, the fit gets better when more inputs are included and worse when more outputs are included. To understand the latter fact, you should realize that a model that has to explain the behavior of several outputs has a tougher job than one that just must account for a single output. If you have difficulties obtaining good models for a multi-output system, it might be wise to model one output at a time, to find out which are the difficult ones to handle.

Models that are just to be used for simulations could very well be built up from single-output models, for one output at a time. However, models for prediction and control will be able to produce better results if constructed for all outputs simultaneously. This follows from the fact that knowing the set of all previous output channels gives a better basis for prediction, than just knowing the past outputs in one channel.

Some Practical Advice

The GUI is particularly suited for dealing with multivariable systems since it will do useful bookkeeping for you, handling different channels. You could follow the steps of this agenda:

- Import data and create a data set with all input and output channels of interest. Do the necessary preprocessing of this set in terms of detrending, prefiltering, etc., and then select a Validation Data set with all channels.
- Then select a Working Data set with all channels, and estimate state-space models of different orders using `n4sid` for these data. Examine the resulting model primarily using the **Model Output** view.
- If it is difficult to get a good fit in all output channels or you would like to investigate how important the different input channels are, construct new data sets using subsets of the original input/output channels. Use the pop-up menu **Preprocess > Select Channels** for this. Don't change the Validation

Data. The GUI will keep track of the input and output channel numbers. It will “do the right thing” when evaluating the channel-restricted models using the Validation Data. It might also be appropriate to see if improvements in the fit are obtained for various model types, built for one output at a time.

- If you decide for a multi-output model, it is often easiest to use state-space models. Use `n4sid` as a primary tool and try out `pem` when a good order has been found. Note that `n4sid` does not provide confidence intervals for the model views.

Reading More About System Identification

There is substantial literature on System Identification. The following textbook deals with identification methods from a similar perspective as this toolbox, and also describes methods for physical modeling.

- Ljung L. and T. Glad. *Modeling of Dynamic Systems*, Prentice Hall, Englewood Cliffs, N.J. 1994.

For more details about the algorithms and theories of identification:



- Ljung L.. *System Identification - Theory for the User*, Prentice Hall, Englewood Cliffs, N.J. 1987.
- Söderström T. and P. Stoica. *System Identification*, Prentice Hall International, London. 1989.

For more about system and signals:

- Oppenheim J. and A.S. Willsky. *Signals and Systems*, Prentice Hall, Englewood Cliffs, N.J. 1985.

The following textbook deals with the underlying numerical techniques for parameter estimation.

- Dennis, J.E. Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewood Cliffs, N.J. 1983.


MATLAB Help Center


CONTENTS

« Documentation Home

« Control Systems

« System Identification Toolbox

Category

Get Started with System Identification Toolbox

Data Preparation

Linear Model Identification

Nonlinear Model Identification

Grey-Box Model Estimation

Model Validation

Model Analysis

Time Series Analysis

Online Estimation

Documentation
Examples
Functions
Blocks
Apps

Get Started with System Identification Toolbox

Create linear and nonlinear dynamic system models from input-output data

R2024a

System Identification Toolbox™ provides MATLAB® functions, Simulink® blocks, and an app for dynamic system modeling, time-series analysis, and forecasting. You can learn dynamic relationships among measured variables to create transfer functions, process models, and state-space models in either continuous or discrete time while using time- or frequency-domain data. You can forecast time series using AR, ARMA, and other linear and nonlinear autoregressive modeling techniques.

The toolbox lets you estimate nonlinear system dynamics using Hammerstein-Wiener and Nonlinear ARX models with machine learning techniques such as Gaussian Processes (GP), Support Vector Machines (SVM), and other representations. Alternatively, you can create neural ordinary differential equation (ODE) models using deep learning to capture nonlinear system dynamics. The toolbox lets you perform grey-box system identification for estimating parameters of a user-defined model. You can integrate identified models into Simulink for rapid simulations to enable control design and diagnostic and prognostic applications.

You can perform online parameter and state estimation using extended or unscented Kalman filters and particle filters for adaptive control, fault detection, and soft sensing applications. The toolbox lets you generate C/C++ code for online estimation algorithms to target embedded devices.

[Acknowledgments](#)

Tutorials

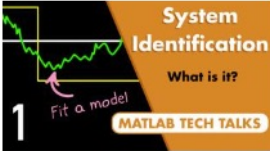
Identify Linear Models Using System Identification App
Identify linear black-box models from single-input/single-output (SISO) data using the System Identification app.

Identify Linear Models Using the Command Line
Identify linear models from multiple-input/single-output (MISO) data using System Identification Toolbox commands.

Identify Low-Order Transfer Functions (Process Models) Using System Identification App
Identify continuous-time transfer functions from single-input/single-output (SISO) data using the app.

Estimate Continuous-Time Grey-Box Model for Heat Diffusion
This example shows how to estimate the heat conductivity and the heat-

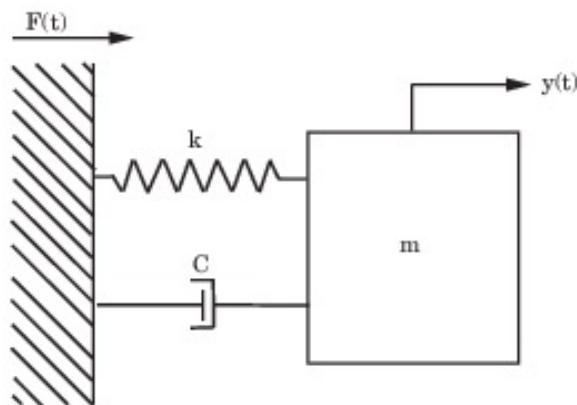
Videos



System Identification Part 1: What is System Identification?
System identification is the process of using data rather than physics to develop a model of a dynamic system. Explore what system

Example: Spring-Mass-Damper System

An often-used example of a dynamic model is the equation of motion of a spring-mass-damper system. As the following figure shows, the mass moves in response to the force $F(t)$ applied on the base to which the mass is attached. The input and output of this system are the force $F(t)$ and displacement $y(t)$, respectively.



Continuous-Time Dynamic Model Example

You can represent the same physical system as several equivalent models. For example, you can represent the mass-spring-damper system in continuous time as a second-order differential equation:

$$m \frac{d^2 y}{dt^2} + c \frac{dy}{dt} + ky(t) = F(t)$$

Here, m is the mass, k is the stiffness constant of the spring, and c is the damping coefficient. The solution to this differential equation lets you determine the displacement of the mass $y(t)$, as a function of external force $F(t)$ at any time t for known values of constant m , c , and k .

... Example: Spring-Mass-Damper System

Continuous-Time Dynamic Model Example

You can represent the same physical system as several equivalent models. For example, you can represent the mass-spring-damper system in continuous time as a second-order differential equation:

$$m \frac{d^2 y}{dt^2} + c \frac{dy}{dt} + ky(t) = F(t)$$

Here, m is the mass, k is the stiffness constant of the spring, and c is the damping coefficient. The solution to this differential equation lets you determine the displacement of the mass $y(t)$, as a function of external force $F(t)$ at any time t for known values of constant m , c , and k .

Consider the displacement $y(t)$ and velocity $v(t) = \frac{dy(t)}{dt}$ as state variables:

$$x(t) = \begin{bmatrix} y(t) \\ v(t) \end{bmatrix}$$

You can express the previous equation of motion as a state-space model of the system:

$$\frac{dx}{dt} = Ax(t) + BF(t)$$

$$y(t) = Cx(t)$$

The matrices A , B , and C are related to the constants m , c , and k as follows:

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}$$

$$C = [1 \quad 0]$$

You can also obtain a *transfer function model* of the spring-mass-damper system by taking the Laplace transform of the differential equation:

$$G(s) = \frac{Y(s)}{F(s)} = \frac{1}{(ms^2 + cs + k)}$$

Here, s is the Laplace variable.

Discrete-Time Dynamic Example

Discrete-Time Dynamic Model Example

Suppose you can observe only the input and output variables $F(t)$ and $y(t)$ of the mass-spring-damper system at discrete time instants $t = nT_s$, where T_s is a fixed time interval and $n = 0, 1, 2, \dots$. The variables are said to be *sampled* with sample time T_s . Then, you can represent the relationship between the sampled input-output variables as a second-order difference equation, such as

$$y(t) + a_1y(t - T_s) + a_2y(t - 2T_s) = bF(t - T_s)$$

Often, for simplicity, T_s is taken as one time unit, and the equation can be written as

$$y(t) + a_1y(t - 1) + a_2y(t - 2) = bF(t - 1)$$

Here, a_1 and a_2 are the model parameters. The model parameters are related to the system constants m , c , and k , and the sample time T_s .

This difference equation shows the dynamic nature of the model. The displacement value at the time instant t depends not only on the value of force F at a previous time instant, but also on the displacement values at the previous two time instants $y(t-1)$ and $y(t-2)$.

You can use this equation to compute the displacement at a specific time. The displacement is represented as a weighted sum of the past input and output values:

$$y(t) = bF(t - 1) - a_1y(t - 1) - a_2y(t - 2)$$

This equation shows an iterative way of generating values of the output $y(t)$ starting from initial conditions $y(0)$ and $y(1)$ and measurements of input $F(t)$. This computation is called *simulation*.

Alternatively, the output value at a given time t can be computed using the *measured* values of output at the previous two time instants and the input value at a previous time instant. This computation is called *prediction*. For more information on simulation and prediction using a model, see topics on the [Simulation and Prediction](#) page.

You can also represent a discrete-time equation of motion in state-space and transfer-function forms by performing the transformations similar to those described in [Continuous-Time Dynamic Model Example](#).

Use Measured Data in System Identification

System identification uses the input and output signals you measure from a system to estimate the values of adjustable parameters in a given model structure. You can build models using time-domain input-output signals, frequency response data, time-series signals, and time-series spectra.

To obtain a good model of your system, you must have measured data that reflects the dynamic behavior of the system. The accuracy of your model depends on the quality of your measurement data, which in turn depends on your experimental design.

Time-Domain Data

Time-domain data consists of the input and output variables of the system that you record at a uniform sampling interval over a period of time.

For example, if you measure the input force $F(t)$ and mass displacement $y(t)$ of the spring-mass-damper system illustrated in [Dynamic Systems and Models](#) at a uniform sampling frequency of 10 Hz, you obtain the following vectors of measured values:

$$u_{meas} = [F(T_s), F(2T_s), F(3T_s), \dots, F(NT_s)]$$

$$y_{meas} = [y(T_s), y(2T_s), y(3T_s), \dots, y(NT_s)]$$

Here, $T_s = 0.1$ seconds and NT_s is the time of the last measurement.

If you want to build a discrete-time model from this data, the data vectors u_{meas} and y_{meas} and the sample time T_s provide sufficient information for creating such a model.

If you want to build a continuous-time model, you must also know the intersample behavior of the input signals during the experiment. For example, the input can be piecewise constant (zero-order hold) or piecewise linear (first-order hold) between samples.

Frequency-Domian Data (Not Covered)

Frequency-Domain Data

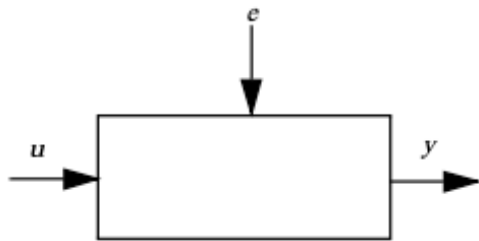
Frequency-domain data represents measurements of the system input and output variables that you record or store in the frequency domain. The frequency-domain signals are Fourier transforms of the corresponding time-domain signals.

Frequency-domain data can also represent the frequency response of the system, represented by the set of complex response values over a given frequency range. The *frequency response* describes the outputs to sinusoidal inputs. If the input is a sine wave with frequency ω , then the output is also a sine wave of the same frequency, whose amplitude is $A(\omega)$ times the input signal amplitude and a phase shift of $\Phi(\omega)$ with respect to the input signal. The frequency response is $A(\omega)e^{i\Phi(\omega)}$.

In the case of the mass-spring-damper system, you can obtain the frequency response data by using a sinusoidal input force and measuring the corresponding amplitude gain and phase shift of the response over a range of input frequencies.

You can use frequency-domain data to build both discrete-time and continuous-time models of your system.

The SysID problem



The basic input-output configuration is depicted in the figure above. Assuming unit sampling interval, there is an input signal

$$u(t); t=1,2,\dots,N$$

and an output signal

$$y(t); t=1,2,\dots,N$$

Assuming the signals are related by a linear system, the relationship can be written

$$y(t) = G(q)u(t) + v(t) \quad (3.1)$$

where q is the shift operator and $G(q)u(t)$ is short for

$$G(q)u(t) = \sum_{k=1}^{\infty} g(k)u(t-k) \quad (3.2)$$

and

$$G(q) = \sum_{k=1}^{\infty} g(k)q^{-k}; \quad q^{-1}u(t) = u(t-1) \quad (3.3)$$

Models of Linear Dynamic Systems

The numbers $\{g(k)\}$ are called the *impulse response* of the system. Clearly, $g(k)$ is the output of the system at time k if the input is a single (im)pulse at time zero. The function $G(q)$ is called the *transfer function* of the system. This function evaluated on the unit circle ($q = e^{i\omega}$) gives the *frequency function*

$$G(e^{i\omega}) \quad (3.4)$$

In (3.1) $v(t)$ is an additional, unmeasurable disturbance (noise). Its properties can be expressed in terms of its (auto) spectrum

$$\Phi_v(\omega) \quad (3.5)$$

which is defined by

$$\Phi_v(\omega) = \sum_{\tau=-\infty}^{\infty} R_v(\tau) e^{-i\omega\tau} \quad (3.6)$$

where $R_v(\tau)$ is the covariance function of $v(t)$:

$$R_v(\tau) = E v(t) v(t-\tau) \quad (3.7)$$

and E denotes mathematical expectation. Alternatively, the disturbance $v(t)$ can be described as filtered white noise:

$$v(t) = H(q)e(t) \quad (3.8)$$

where $e(t)$ is white noise with variance λ and

$$\Phi_v(\omega) = \lambda |H(e^{i\omega})|^2 \quad (3.9)$$

Equations (3.1) and (3.8) together give a *time domain description* of the system:

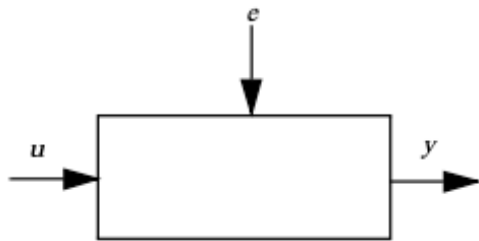
$$y(t) = G(q)u(t) + H(q)e(t) \quad (3.10)$$

while (3.4) and (3.5) constitute a *frequency domain description*:

$$G(e^{i\omega}); \quad \Phi_v(\omega) \quad (3.11)$$

The impulse response (3.3) and the frequency domain description (3.11) are called *nonparametric model descriptions* since they are not defined in terms of a finite number of parameters. The basic description (3.10) also applies to the multivariable case; i.e., to systems with several (say nu) input signals and several (say ny) output signals. In that case $G(q)$ is an ny by nu matrix while $H(q)$ and $\Phi_v(\omega)$ are ny by ny matrices.

The SysID problem (We only deal with Time Domain Models)



The basic input-output configuration is depicted in the figure above. Assuming unit sampling interval, there is an input signal

$$u(t); t=1,2,\dots,N$$

and an output signal

$$y(t); t=1,2,\dots,N$$

Assuming the signals are related by a linear system, the relationship can be written

$$y(t) = G(q)u(t) + v(t) \quad (3.1)$$

where q is the shift operator and $G(q)u(t)$ is short for

$$G(q)u(t) = \sum_{k=1}^{\infty} g(k)u(t-k) \quad (3.2)$$

and

$$G(q) = \sum_{k=1}^{\infty} g(k)q^{-k}; \quad q^{-1}u(t) = u(t-1) \quad (3.3)$$

the disturbance $v(t)$ can be described as filtered white noise:

$$v(t) = H(q)e(t) \quad (3.8)$$

Equations (3.1) and (3.8) together give a *time domain description* of the system:

$$y(t) = G(q)u(t) + H(q)e(t) \quad (3.10)$$

Polynomial Representation of Transfer Functions

Polynomial Representation of Transfer Functions

Rather than specifying the functions G and H in (3.10) in terms of functions of the frequency variable ω , you can describe them as rational functions of q^{-1} and specify the numerator and denominator coefficients in some way.

A commonly used parametric model is the ARX model that corresponds to

$$G(q) = q^{-nk} \frac{B(q)}{A(q)}; \quad H(q) = \frac{1}{A(q)} \quad (3.12)$$

where B and A are polynomials in the delay operator q^{-1} :

$$A(q) = 1 + a_1 q^{-1} + \dots + a_{na} q^{-na} \quad (3.13)$$

$$B(q) = b_1 + b_2 q^{-1} + \dots + b_{nb} q^{-nb+1} \quad (3.14)$$

Here, the numbers na and nb are the orders of the respective polynomials. The number nk is the number of delays from input to output. The model is usually written

$$A(q)y(t) = B(q)u(t-nk) + e(t) \quad (3.15)$$

or explicitly

$$y(t) + a_1 y(t-1) + \dots + a_{na} y(t-na) = b_1 u(t-nk) + b_2 u(t-nk-1) + \dots + b_{nb} u(t-nk-nb+1) + e(t) \quad (3.16)$$

Note that (3.15)-(3.16) apply also to the multivariable case, where $A(q)$ and the coefficients a_i become ny by ny matrices, $B(q)$ and the coefficients b_i become ny by nu matrices.

Another very common, and more general, model structure is the ARMAX structure

$$A(q)y(t) = B(q)u(t-nk) + C(q)e(t) \quad (3.17)$$

Here, $A(q)$ and $B(q)$ are as in (3.13)-(3.14), while

$$C(q) = 1 + c_1 q^{-1} + \dots + c_{nc} q^{-nc} \quad (3.18)$$

An *Output-Error* (OE) structure is obtained as

$$y(t) = \frac{B(q)}{F(q)} u(t-nk) + e(t) \quad (3.19)$$

with

$$F(q) = 1 + f_1 q^{-1} + \dots + f_{nf} q^{-nf} \quad (3.20)$$

The so-called *Box-Jenkins* (BJ) model structure is given by

$$y(t) = \frac{B(q)}{F(q)} u(t-nk) + \frac{C(q)}{D(q)} e(t) \quad (3.21)$$

with

$$D(q) = 1 + d_1 q^{-1} + \dots + d_{nd} q^{-nd} \quad (3.22)$$

... Polynomial Representation of Transfer Functions

All these models are special cases of the general parametric model structure:

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t-nk) + \frac{C(q)}{D(q)}e(t) \quad (3.23)$$

The variance of the white noise $\{e(t)\}$ is assumed to be λ .

Within the structure of (3.23), virtually all of the usual linear black-box model structures are obtained as special cases. The ARX structure is obviously obtained for $nc = nd = nf = 0$. The ARMAX structure corresponds to $nf = nd = 0$. The ARARX structure (or the “generalized least-squares model”) is obtained for $nc = nf = 0$, while the ARARMAX structure (or “extended matrix model”) corresponds to $nf = 0$. The Output-Error model is obtained with $na = nc = nd = 1$, while the Box-Jenkins model corresponds to $na = 0$. (See Section 4.2 in Ljung (1987) for a detailed discussion.)

The same type of models can be defined for systems with an arbitrary number of inputs. They have the form

$$A(q)y(t) = \frac{B_1(q)}{F_1(q)}u_1(t-nk_1) + \dots + \frac{B_{nu}(q)}{F_{nu}(q)}u_{nu}(t-nk_{nu}) + \frac{C(q)}{D(q)}e(t)$$

Estimating Parametric Models

Given a description (3.10) and having observed the input-output data u, y , the (prediction) errors $e(t)$ in (3.10) can be computed as:

$$e(t) = H^{-1}(q)[y(t) - G(q)u(t)] \quad (3.38)$$

These errors are, for given data y and u , functions of G and H . These in turn are parametrized by the polynomials in (3.15)–(3.23) or by entries in the state-space matrices defined in (3.29)–(3.32). The most common parametric identification method is to determine estimates of G and H by minimizing

$$V_N(G, H) = \sum_{t=1}^N e^2(t) \quad (3.39)$$

that is

$$[\hat{G}_N, \hat{H}_N] = \underset{G, H}{\operatorname{argmin}} \sum_{t=1}^N e^2(t) \quad (3.40)$$

This is called a *prediction error method*. For Gaussian disturbances it coincides with the maximum likelihood method. (See Chapter 7 in Ljung (1987).)

A somewhat different philosophy can be applied to the ARX model (3.15). By forming filtered versions of the input

$$N(q)s(t) = M(q)u(t) \quad (3.41)$$

and by multiplying (3.15) with $s(t-k)$, $k = 1, 2, \dots, na$ and $u(t-nk+1-k)$, $k = 1, 2, \dots, nb$ and summing over t , the noise in (3.15) can be correlated out and solved for the dynamics. This gives the *instrumental variable* method, and $s(t)$ are called the instruments. (See Section 7.6 in Ljung (1987).)

A Simple SysID Demo

```
% 20 Feb 2025  
% SysID example  
% LKV
```

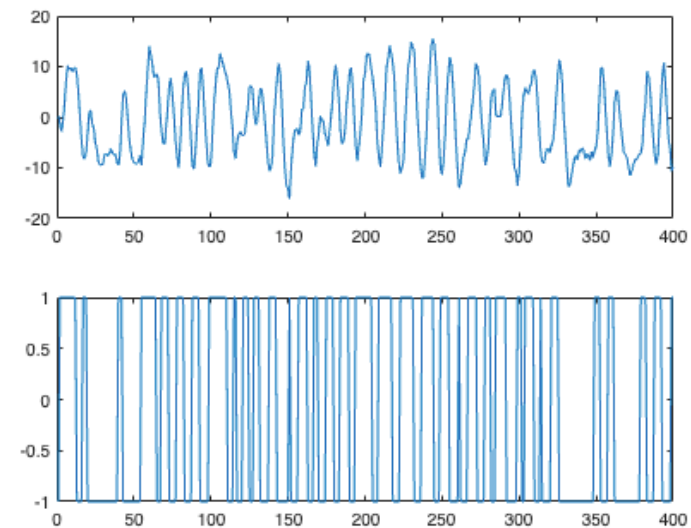
The System Identification Toolbox also has several commands for simulation. You should check `idinput`, `idsim`, `poly2th`, `modstruc`, and `ms2th` in Chapter 4, "Command Reference," for details. The following example shows how the ARMAX model

$$y(t) - 1.5y(t-1) + 0.7y(t-2) = u(t-1) + 0.5u(t-2) + e(t) - e(t-1) + 0.2e(t-2)$$

is simulated with a binary random input u :

```
model1 = poly2th([1 -1.5 0.7],[0 1 0.5],[1 -1 0.2]);  
u = idinput(400,'rbs',[0 0.3]);  
e = randn(400,1);  
y = idsim([u e],model1);
```

```
model = poly2th([1 -1.5 0.7],[0 1 0.5],[1 -1 0.2]);  
u = idinput(400,'rbs',[0 0.3]);  
e = randn(400,1);  
y = idsim([u e],model);  
subplot(211),plot(y)  
subplot(212),plot(u)  
N=200;  
Y = y(1:N);  
Yold = [0; y(1:N-1)];  
Yold2 = [0; 0; y(1:N-2)];  
U = u(1:N);  
Uold = [0; u(1:N-1)];  
Uold2 = [0; 0; u(1:N-2)];  
Phi = [Yold -Yold2 Uold Uold2];  
th = Phi\Y
```



```
th = 4x1  
    1.2867  
    0.5152  
    0.8625  
    1.0889
```

SysID Dryer Example

```
% SysID Dryer Example
% 20 Feb 2025
% LKV

load dryer2
z2 = [y2(1:300) u2(1:300)];
idplot(z2)
%idplot(z2,200:300,0.08)
z2 = dtrend(z2);
```

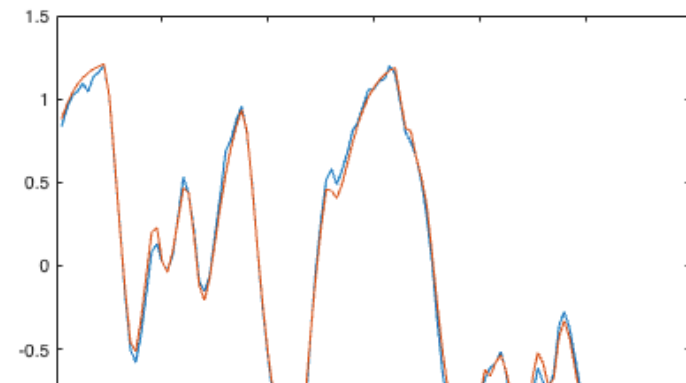
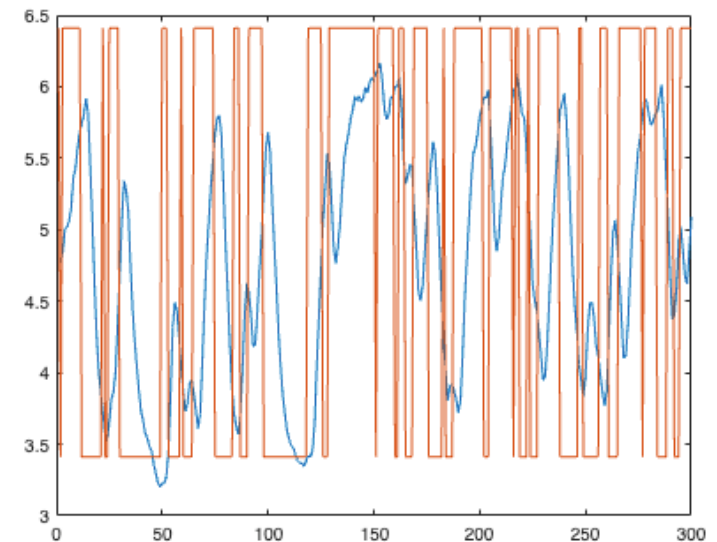
Now, fit to the data a model of the form:

$$y(t) + a_1 y(t-T) + a_2 y(t-2T) = b_1 u(t-3T) + b_2 u(t-4T)$$

```
th = arx(z2,[2 2 3]);
%present(th);
```

Next, you might ask how to evaluate how well the model fits the data. A simple test is to run a simulation whereby real input data is fed into the model, and compare the simulated output with the actual measured output. For this, select a portion of the data that was not used to build the model, for example, from sample 700 to 900:

```
u = dtrend(u2(700:900));
y = dtrend(y2(700:900));
ysim = idsim(u,th);
plot([y(100:200), ysim(100:200)])
```



Extracting information from data is not an entirely straightforward task. In addition to the decisions required for model structure selection and validation, the data may need to be handled carefully. This section gives some advice on handling several common situations.

Offset Levels

When the data have been collected from a physical plant, they are typically measured in physical units. The levels in these raw inputs and outputs may not match in any consistent way. This will force the models to waste some parameters correcting the levels.

Offsets are easily dealt with; always subtract the mean levels from the input and output sequences before the estimation. It is best if the mean levels correspond to a physical equilibrium, but if such values are not known, use the sample means:

```
z = dtrend(z);
```

Section 14.6 in Ljung (1987) discusses this in more detail. With the `dtrend` command, you can also remove piece-wise linear trends.

Outliers

Real data are also subject to possible bad disturbances; an unusually large disturbance, a temporary sensor or transmitter failure, etc. It is important that such outliers are not allowed to affect the models too strongly.

The robustification of the error criterion (described under `lim` in “Optional Variables” on page 3-26) helps here, but it is always good practice to examine the residuals for unusually large values, and to go back and critically evaluate the original data responsible for the large values. If the raw data are obviously in error, they can be smoothed, and the estimation procedure repeated.

Filtering Data

Depending upon the application, interest in the model can be focused on specific frequency bands. Filtering the data before the estimation, through filters that enhance these bands, improves the fit in the interesting regions. This is used in the System Identification Toolbox function `idfilt`. For

example, to enhance the data in the frequency band between $0.02 * \pi$ and $0.1 * \pi$, execute

```
zf = idfilt(z,5,[0.02 0.1]);
```

This computes and uses a fifth order Butterworth bandpass filter with passband between the indicated frequencies. Chapter 13 in Ljung (1987) discusses the role of filtering in more detail.

The SITB contains other useful commands for related problems. For example, if you want to lower the sampling rate by a factor of 5, use

```
z5 = idresamp(z,5);
```

Feedback in Data

If the system was operating in closed loop (feedback from the past outputs to the current input) when the data were collected, some care has to be exercised.

Basically, all the prediction error methods work equally well for closed-loop data. Note, however, that the Output-Error model (3.19) and the Box-Jenkins model (3.21) are normally capable of giving a correct description of the dynamics G , even if H (which equals 1 for the output error model) does not allow a correct description of the noise properties. This is no longer true for closed-loop data. You then need to model the noise properties carefully.

The spectral analysis method and the instrumental variable techniques (with default instruments) give unreliable results when applied to closed-loop data. These techniques should be avoided when feedback is present.

To detect if feedback is present, use the basic method of applying `cra` to estimate the impulse response. Significant values of the impulse response at negative lags is a clear indication of feedback. When a parametric model has been estimated and the `resid` command is applied, a graph of the correlation between residuals and inputs is given. Significant correlation at negative lags again indicates output feedback in the generation of the input. Testing for feedback is, therefore, a natural part of model validation.

Delays

The selection of the delay n_k in the model structure is a very important step in obtaining good identification results. You can get an idea about the delays in the system by the impulse response estimate from `cra`.

Incorrect delays are also visible in parametric models. Underestimated delays (n_k too small) show up as small values of leading b_k estimates, compared to their standard deviations. Overestimated delays (n_k too large) are usually visible as a significant correlation between the residuals and the input at the lags corresponding to the missing b_k terms.