System Identification and Control (BI3SI)

William Harwin
based on notes by X. Hong and W.S. Harwin

Lecture 1: Introduction

Books:

- Ljung, L and T. Söderström (1983): Theory and Practice of Recursive Identification, The MIT Press[1].
- Ljung, Lennart (1987): System identification: Theory for the user, Prentice Hall[2][3].
- Söderström, T and P. Stoica (1989): System Identification, Prentice Hall.[4]
- Aström, K. J. and B. Wittenmark (1989): Adaptive Control, Addison Wesley.

System Identification:

Assessment is via examination and an assignment (see blackboard)

There may be additional information on the websites `http://www.personal.rdg.ac.uk/~sis01xh/` `http://www.reading.ac.uk/~shshawin/LN`

Introduction to System identification

Systems and models

Systems are objects, of which we would like to study, control and affect the behavior. There are tasks typically related to the study and use of the systems. e.g.

- A ship: We would like to control its direction by manipulating the rudder angle.
- A telephone channel: We would like to construct a receiver with good reproduction of the transmitted signal.
- A time series of data (e.g. sales): We would like to predict the future values.

A model is the knowledge of the properties of a system. It is necessary to have a model of the system in order to solve problems such as control, signal processing, system design. Sometimes the aim of modelling is to aid in design. Sometimes a simple model is useful for explaining certain phenomena and reality.

A model may be given in any one of the following forms:

1. Mental, intuitive or verbal models: Knowledge of the system's behavior is summarized in a person's mind.
2. Graphs and tables: e.g. the step response, Bode plot.
3. Mathematical models: Here we confine this class of model to differential and difference equations.

There are two ways of constructing mathematical models:

1. Physical modelling: Basic laws from physics such as Newton's laws and balance equations are used to describe the dynamical behavior of a process. Yet in many cases the processes are so complex that it is not possible to obtain reasonable models using only physical insight, so we are forced to use the alternative:
2. System identification: Some experiments are performed on the system; A model is then fitted to the recorded data by assigning suitable values to its parameters.

System identification is the field of modeling dynamic systems from experimental data. A dynamical system can be conceptually described by the following Figure 1.

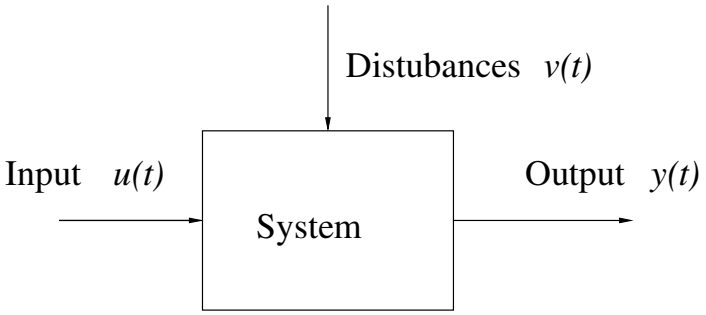


Figure 1: A dynamical system with input $u(t)$, output $y(t)$ and disturbance $v(t)$, where t denotes time.

The system is controlled by input $u(t)$. The user cannot control $v(t)$. The output $y(t)$ can be measured and gives information about the system. For a dynamical system the control action at time t will influence the output at time instants $s > t$.

How system identification is applied

1. Design of experiments.
2. Perform experiments, and collect data.

3. Determine/choose model structure.
4. Estimate model parameters.
5. Model validation.

In practice the procedure is iterated until an acceptable model is found. A good model should encompass essential information without becoming too complex.

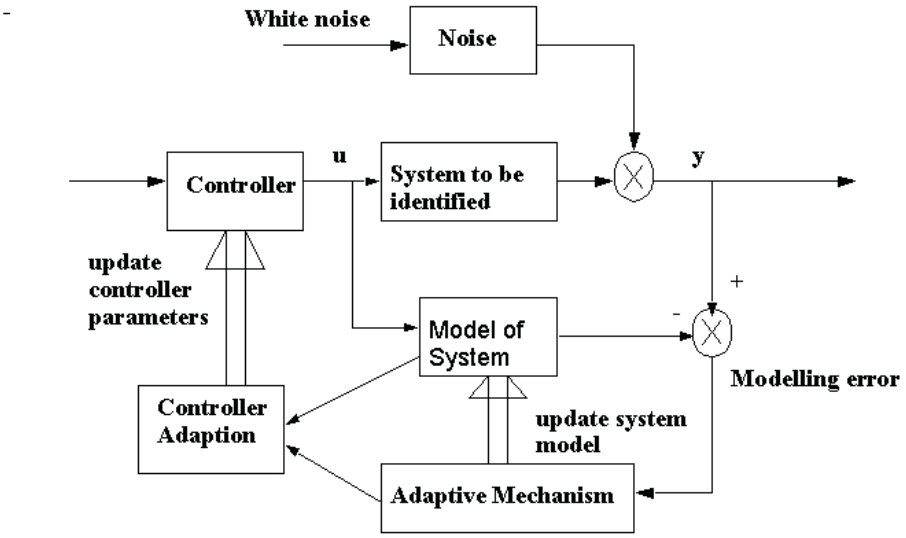


Figure 2: System identification and computer control

Shift-operator calculus

The forward-shift operator

$$qf(k) = f(k+1) \qquad \text{or} \qquad qf_k = f_{k+1} \tag{1}$$

The backward-shift operator

$$q^{-1}f(k) = f(k-1) \qquad \text{or} \qquad q^{-1}f_k = f_{k-1} \tag{2}$$

The shift operator is used to simplify the manipulation of higher-order difference equations. Consider the equation

$$\begin{aligned} & y(k) + a_1 y(k-1) + ... + a_{na} y(k-na) \\ = \; & b_0 u(t-d) + b_1 u(t-d-1) + ... + b_{nb} u(t-d-nb) \end{aligned} \tag{3}$$

The model can be expressed as

$$A(q^{-1})y(k) = B(q^{-1})u(k) \tag{4}$$

where

$$A(q^{-1}) = a_0 + a_1 q^{-1} + ...a_{na} q^{-na}, \quad \text{and} \quad a_0 = 1. \tag{5}$$

$$B(q^{-1}) = (b_0 + b_1 q^{-1} + ...b_{nb} q^{-nb}) q^{-d} \tag{6}$$

These discrete representations are thus linear and time invariant.

- Linear because $\alpha y = \alpha u$ for some real number α (scaling the input results in an identically scaled output).
- Time invarient because $q^n y = q^n u$ for some real number n, (the process does not change on different days).

Z-transform

Given a signal $x(t)$ sampled every T seconds then the z-transform is

$$X(z) = \mathcal{Z}[x(kT)] = \sum_{k=0}^{\infty} x(kT) z^{-k}$$

Lecture 2: Least squares method

Positive definite matrices

Consider the following function f, defined with a matrix $\mathbf{M}$ and input vector $\mathbf{x}$

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{M} \mathbf{x}$$

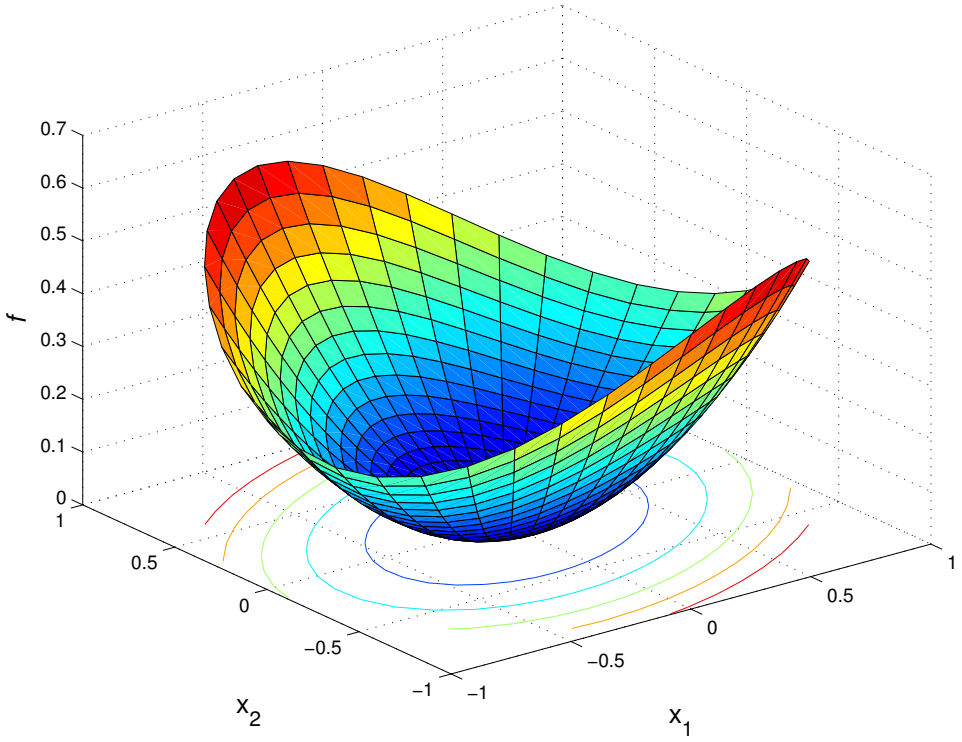


Figure 3: The function $f(\mathbf{x}) = \mathbf{x}^T\mathbf{M}\mathbf{x}$ is always positive therefore $\mathbf{M}$ is positive definate.

That is f is a quatratic function of $\boldsymbol{x}$.

If f is always positive (and 0 except when $\mathbf{x} = [0, 0]^T$) irrespective of $\mathbf{x}$ then $\mathbf{M}$ is positive definite. If $\mathbf{x}$ is a two valued vector $\mathbf{x} = [x_1, x_2]^T$, then f looks like a bowl resting on the origin as seen in the Figure below, for a positive definite $\mathbf{M}$.

Two other shapes can result from the quadratic form. If f is always negative then $\mathbf{M}$ is known as negative definite. If f is positive in some regions and negative on others then it describes a saddle. In all cases f is zero when $\mathbf{x} = [0, 0]^T$.

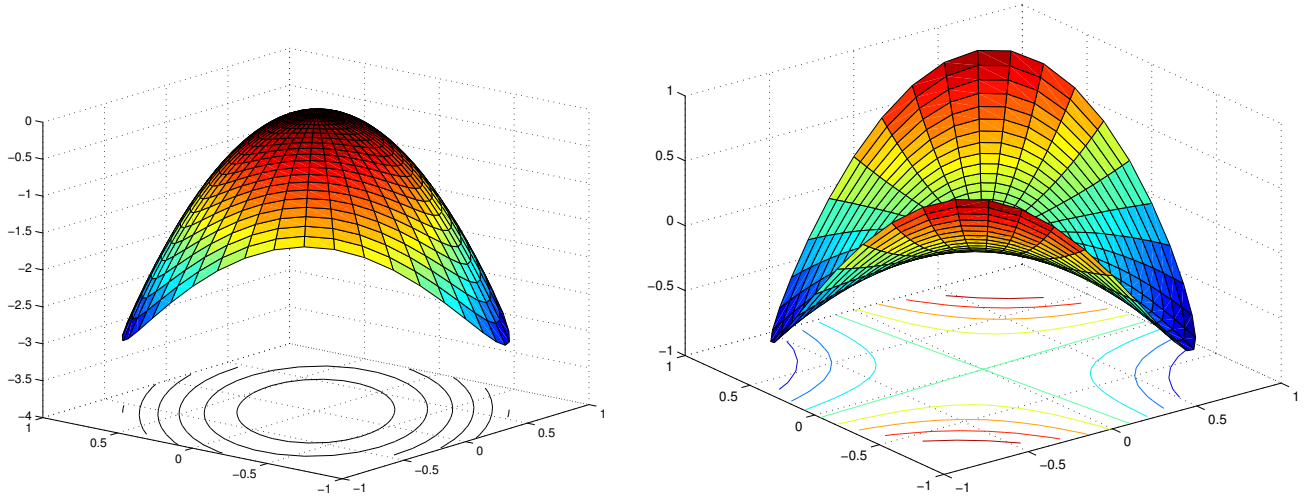


Figure 4: Left: Matrix $\mathbf{M}$ is negative definate. Right: Matrix $\mathbf{M}$ defines a saddle.

Theorem: If a matrix $\mathbf{M} = \boldsymbol{\phi}^T\boldsymbol{\phi}$, then it is positive definite.

Proof. The quadratic form is

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{M}\mathbf{x} = \mathbf{x}^T\boldsymbol{\phi}^T\boldsymbol{\phi}\mathbf{x}$$

If we can show that f is always positive then $\mathbf{M}$ must be positive definite.

$$f(\mathbf{x}) = (\boldsymbol{\phi}\mathbf{x})^T\boldsymbol{\phi}\mathbf{x}$$

Denote $\mathbf{z} = \boldsymbol{\phi}\mathbf{x}$ so

$$f(\mathbf{x}) = \mathbf{z}^T\mathbf{z} = \sum_i z_i^2$$

so f must always be positive (except when $\mathbf{x} = [0, 0]^T$).

Linear regression

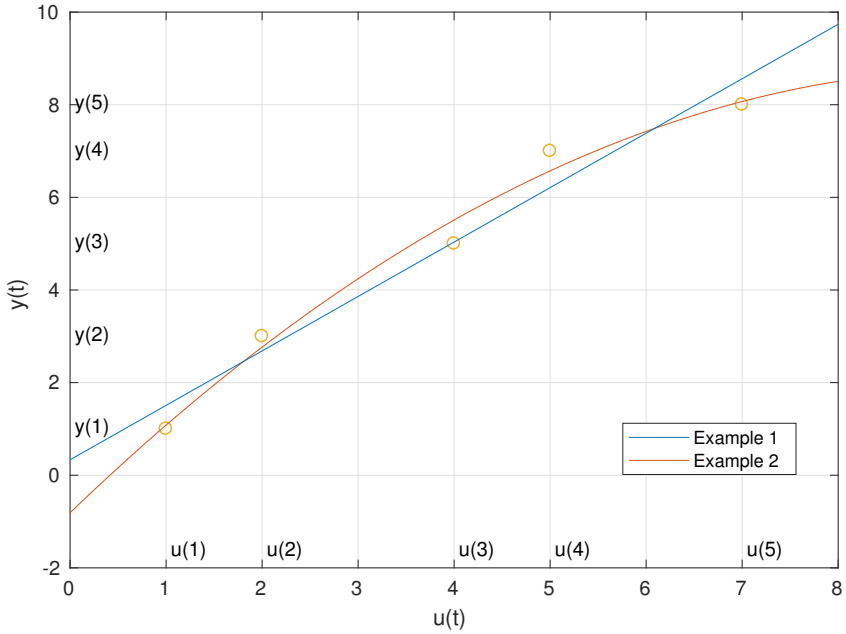


Figure 5: Two models fitted with least squares linear regression.

The simplest parametric model is linear regression.

$$y(t) = \hat{y}(t) + e(t) = \boldsymbol{\phi}^T(t)\boldsymbol{\theta} + e(t)$$

where $y(t)$ is a measurable quantity. $\boldsymbol{\phi}(t)$ is a vector of known quantities, and $\boldsymbol{\theta}$ is a vector of unknown parameters. $e(t)$ is modelling error. t denotes the label for data samples.

Given a model form and a data set, we wish to calculate the parameter $\boldsymbol{\theta}$. One way is to minimise the errors between the actual $y(t)$ and the model prediction $\hat{y}(t)$.

Example 2-1:

$y(t) = au(t) + b + e(t)$

$\boldsymbol{\theta} = [a, b]^T$, $\boldsymbol{\phi}(t) = [u(t), 1]^T$.

t	1	2	3	4	5
$u(t)$	1	2	4	5	7
$y(t)$	1	3	5	7	8

$$\hat{\mathbf{y}} = \begin{pmatrix} \hat{y}(1) \\ \hat{y}(2) \\ \hat{y}(3) \\ \hat{y}(4) \\ \hat{y}(5) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 4 & 1 \\ 5 & 1 \\ 7 & 1 \end{pmatrix} \boldsymbol{\theta} = \boldsymbol{\phi}\boldsymbol{\theta}$$

Example 2-2:

$y(t) = b_2u^2(t) + b_1u(t) + b_0 + e(t)$

$\boldsymbol{\theta} = [b_2, b_1, b_0]^T$, $\boldsymbol{\phi}(t) = [u^2(t), u(t), 1]^T$.

$$\begin{aligned} \hat{\mathbf{y}} &= \begin{pmatrix} \hat{y}(1) \\ \hat{y}(2) \\ \vdots \\ \hat{y}(N-1) \\ \hat{y}(N) \end{pmatrix} \\ &= \begin{pmatrix} u^2(1) & u(1) & 1 \\ u^2(2) & u(2) & 1 \\ \vdots & \vdots & \vdots \\ u^2(N-1) & u(N-1) & 1 \\ u^2(N) & u(N) & 1 \end{pmatrix} \boldsymbol{\theta} \\ &= \boldsymbol{\phi}\boldsymbol{\theta} \end{aligned}$$

Example 2-3:

$$y(t) = b_2 u(t) + b_1 u(t-1) + b_0 u(t-2) + e(t)$$

$\boldsymbol{\theta} = [b_2, b_1, b_0]^T$, $\boldsymbol{\phi}(t) = [u(t), u(t-1), u(t-2)]^T$.

$$\begin{aligned}
\hat{\mathbf{y}} &= \begin{pmatrix} \hat{y}(1) \\ \hat{y}(2) \\ \vdots \\ \hat{y}(N-1) \\ \hat{y}(N) \end{pmatrix} \\
&= \begin{pmatrix} u(1) & 0 & 0 \\ u(2) & u(1) & 0 \\ \vdots & \vdots & \vdots \\ u(N-1) & u(N-2) & u(N-3) \\ u(N) & u(N-1) & u(N-2) \end{pmatrix} \boldsymbol{\theta} \\
&= \boldsymbol{\phi}\boldsymbol{\theta}
\end{aligned}$$

Denote the actual output vector

$$\mathbf{y} = [y(1), y(2), ..., y(N)]^T.$$

A model can be estimated by using $\mathbf{y}$ (actual measurement of output) as a target of $\hat{\mathbf{y}}$ (the output that is expected by the model).

Derivation of the least squares algorithm

For $t = 1, ..., N$,

$$e(t) = y(t) - \hat{y}(t)$$

In vector form

$$\mathbf{e} = \begin{pmatrix} e(1) \\ e(2) \\ \vdots \\ e(N-1) \\ e(N) \end{pmatrix} = \mathbf{y} - \hat{\mathbf{y}}$$

SSE(sum of squared errors)=$\sum_{t=1}^{N} e^2(t) = \mathbf{e}^T\mathbf{e}$

The model parameter vector $\boldsymbol{\theta}$ is derived so that the distance between these two vectors is the smallest possible, i.e. SSE is minimized.

$$\begin{aligned}
\mathbf{e}^T\mathbf{e} =& (\mathbf{y} - \hat{\mathbf{y}})^T(\mathbf{y} - \hat{\mathbf{y}}) \\
=& (\mathbf{y}^T - \hat{\mathbf{y}}^T)(\mathbf{y} - \hat{\mathbf{y}}) \\
& \longleftarrow \hat{\mathbf{y}} = \boldsymbol{\phi}\boldsymbol{\theta}, \quad (\boldsymbol{\phi}\boldsymbol{\theta})^T = \boldsymbol{\theta}^T\boldsymbol{\phi}^T \\
=& (\mathbf{y}^T - \boldsymbol{\theta}^T\boldsymbol{\phi}^T)(\mathbf{y} - \boldsymbol{\phi}\boldsymbol{\theta}) \\
=& \mathbf{y}^T\mathbf{y} - 2\boldsymbol{\theta}^T\boldsymbol{\phi}^T\mathbf{y} + \boldsymbol{\theta}^T\boldsymbol{\phi}^T\boldsymbol{\phi}\boldsymbol{\theta} \\
=& \mathbf{y}^T\mathbf{y} - \underbrace{\mathbf{y}^T\boldsymbol{\phi}[\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y} + \mathbf{y}^T\boldsymbol{\phi}[\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y}}_{0} \\
& - 2\boldsymbol{\theta}^T\boldsymbol{\phi}^T\mathbf{y} + \boldsymbol{\theta}^T\boldsymbol{\phi}^T\boldsymbol{\phi}\boldsymbol{\theta} \\
=& \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\boldsymbol{\phi}[\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y} \\
& + (\boldsymbol{\theta} - [\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y})^T\boldsymbol{\phi}^T\boldsymbol{\phi}(\boldsymbol{\theta} - [\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y})
\end{aligned}$$

Note that only the last term contains $\boldsymbol{\theta}$. The SSE is minimal when the last term is minimised.

$$\begin{aligned}
\mathbf{e}^T\mathbf{e} =& \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\boldsymbol{\phi}[\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y} \\
& + (\boldsymbol{\theta} - [\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y})^T\boldsymbol{\phi}^T\boldsymbol{\phi}(\boldsymbol{\theta} - [\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y}) \\
=& \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\boldsymbol{\phi}[\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y} \\
& + \mathbf{x}^T\boldsymbol{\phi}^T\boldsymbol{\phi}\mathbf{x}
\end{aligned}$$

where $\mathbf{x} = \boldsymbol{\theta} - [\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y}$.

SSE has the minimum $\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\boldsymbol{\phi}[\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y}$ when

$$\mathbf{x} = \boldsymbol{\theta} - [\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

The solution of least squares parameter estimate is

$$\hat{\boldsymbol{\theta}} = [\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y}$$

where $\hat{\boldsymbol{\theta}}$ means the estimate of $\boldsymbol{\theta}$.

Example 2-4:

For Example 2-1 $\mathbf{y} = [1, 3, 5, 7, 8]^T$, and

$$\boldsymbol{\phi} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 4 & 1 \\ 5 & 1 \\ 7 & 1 \end{pmatrix}$$

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= [\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y} \\
&= [\begin{pmatrix} 1 & 2 & 4 & 5 & 7 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 4 & 1 \\ 5 & 1 \\ 7 & 1 \end{pmatrix}]^{-1} \\
&\quad \times \begin{pmatrix} 1 & 2 & 4 & 5 & 7 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 5 \\ 7 \\ 8 \end{pmatrix} \\
&= \begin{pmatrix} 95 & 19 \\ 19 & 5 \end{pmatrix}^{-1} \begin{pmatrix} 118 \\ 24 \end{pmatrix} \\
&= \begin{pmatrix} 1.1754 \\ 0.3333 \end{pmatrix}
\end{aligned}$$

and the model of least squares fit is

$$\hat{y}(t) = 1.1754u(t) + 0.3333$$

An alternative proof:

$$SSE = \mathbf{e}^T\mathbf{e} = \mathbf{y}^T\mathbf{y} - 2\boldsymbol{\theta}^T\boldsymbol{\phi}^T\mathbf{y} + \boldsymbol{\theta}^T\boldsymbol{\phi}^T\boldsymbol{\phi}\boldsymbol{\theta}$$

SSE has the minimum when

$$\frac{\partial(SSE)}{\partial\boldsymbol{\theta}} = \mathbf{0}$$

$$\begin{aligned}
\frac{\partial(SSE)}{\partial\boldsymbol{\theta}} &= -2[\frac{\partial(\mathbf{y}^T\boldsymbol{\phi}\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}]^T + \frac{\partial(\boldsymbol{\theta}^T\boldsymbol{\phi}^T\boldsymbol{\phi}\boldsymbol{\theta})}{\partial\boldsymbol{\theta}} \\
&= -2\boldsymbol{\phi}^T\mathbf{y} + 2\boldsymbol{\phi}^T\boldsymbol{\phi}\boldsymbol{\theta} \\
&= -2\boldsymbol{\phi}^T(\mathbf{y} - \boldsymbol{\phi}\boldsymbol{\theta}) = \mathbf{0}
\end{aligned}$$

yielding

$$\hat{\boldsymbol{\theta}} = [\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y}$$

Lecture 3: Model representations for dynamical systems

Consider a dynamical system with input signal $\{u(t)\}$ and output signal $\{y(t)\}$. Suppose that these signals are sampled in discrete time $t = 1, 2, 3...$ and the sample values can be related through a linear difference equation

$$y(t) + a_1 y(t-1) + ... + a_{n_a} y(t-n_a) = b_1 u(t-1)... + b_2 u(t-2) + ... + b_{n_b} u(t-n_b) + v(t)$$

where $v(t)$ is some disturbance.

Let q^{-1} be the backward shift (or delay) operator: $q^{-1}y(t) = y(t-1)$.

Then the system can be rewritten as

$$A(q^{-1})y(t) = B(q^{-1})u(t) + v(t)$$

where

$$A(q^{-1}) = 1 + a_1 q^{-1} + ... + a_{n_a} q^{-n_a}$$
$$B(q^{-1}) = b_1 q^{-1} + ... + b_{n_b} q^{-n_b}$$

The system can be expressed as a linear regression

$$y(t) = \boldsymbol{\phi}^T(t)\boldsymbol{\theta} + v(t)$$

with

$$\boldsymbol{\theta} = [a_1, ..., a_{n_a}, b_1, ..., b_{n_b}]^T$$

$$\boldsymbol{\phi}(t) = [-y(t-1), ..., -y(t-n_a), u(t-1), ..., u(t-n_b)]^T$$

If $\{v(t)\}$ is a sequence of independent random variable with zero mean values. We call it 'white noise'.

If no input is present and $\{v(t)\}$ is white, then the system becomes

$$y(t) + a_1 y(t-1) + ... + a_{n_a} y(t-n_a) = v(t)$$

which is known as an autoregressive (AR) processes of order n_a.

The ARMAX Model

See 11 for the acronym expansion of ARMAX, or read on.

Suppose the disturbance is described as a moving average (MA) of a white noise sequence $\{e(t)\}$:

$$v(t) = C(q^{-1})e(t)$$

$$C(q^{-1}) = 1 + c_1 q^{-1} + ... + c_{n_c} q^{-n_c}$$

Then the resultant model is

$$A(q^{-1})y(t) = B(q^{-1})u(t) + C(q^{-1})e(t)$$

known as an ARMAX model. The model is a combination of an autoregressive (AR) part

$$A(q^{-1})y(t),$$

a moving average (MA) part

$$C(q^{-1})e(t),$$

and a control part

$$B(q^{-1})u(t),$$

with $u(t)$ being referred as as an eXogeneous variable.

When there is no input, the model is

$$A(q^{-1})y(t) = C(q^{-1})e(t)$$

which is known as an ARMA process, a very common type of model for stochastic signals.

Another variant is

$$y(t) = B(q^{-1})u(t) + v(t)$$

referred as a finite impulse response model (FIR) model.

The ARMAX Model is a linear regression

$$y(t) = \boldsymbol{\phi}^T(t)\boldsymbol{\theta} + e(t)$$

with

$$\boldsymbol{\theta} = [a_1, ..., a_{n_a}, b_1, ..., b_{n_b}, c_1, ...c_{n_c}]^T$$

$$\boldsymbol{\phi}(t) = [-y(t-1), ..., -y(t-n_a), u(t-1)..., u(t-n_b), e(t-1), ..., e(t-n_c)]^T$$

Basic examples of ARMAX Systems

The basic examples to represent the systems are given by

$$y(t) + ay(t-1) = bu(t-1) + ce(t-1) + e(t)$$

where $\{e(t)\}$ is a white noise sequence with variance σ^2.

Suppose that two sets of parameters are used:

System 1: $a = -0.8, b = 1.0, c = 0.0, \sigma = 1.0$

$$y(t) - 0.8y(t-1) = u(t-1) + e(t)$$

System 2: $a = -0.8, b = 1.0, c = -0.8, \sigma = 1.0$

$$x(t) - 0.8x(t-1) = u(t-1)$$

$$y(t) = x(t) + e(t)$$

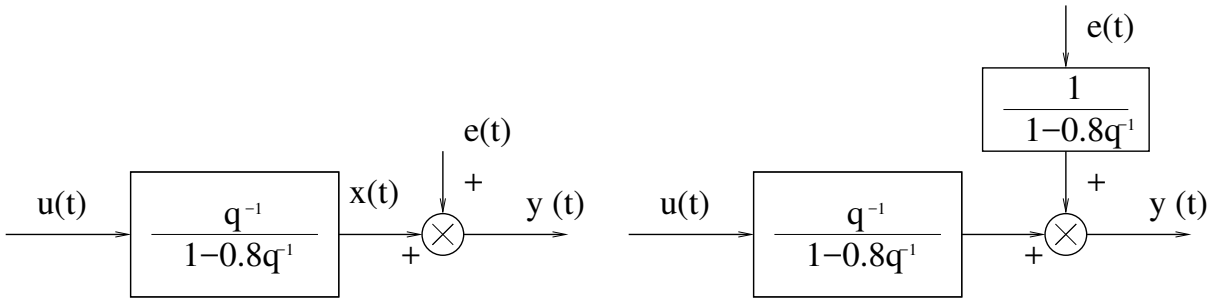


Figure 6: Left: System 1 Right: System 2

Example 3-1

System 1 is simulated generating 1000 data points. The input $u(t)$ is a uniformly distributed random signal in [-2,2].

Matlab code generating data (system 1):

```
>> u=2-4*rand(1,1000); % zero mean uniform distribution
>> e=randn(1,1000);
>> y(1)=0;
>> for  i=2:1000;
>>   y(i)=0.8*y(i-1) + u(i-1) +  e(i);
>> end;
>> figure(1); plot(u);xlabel('t');ylabel('u(t)');
>> figure(2); plot(y);xlabel('t');ylabel('y(t)');
```

Suppose that a model in the form of

$$\hat{y}(t) = -ay(t-1) + bu(t-1)$$

is used to model the system. The least squares solution is

$$\hat{\boldsymbol{\theta}} = [\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y}$$

based on $\mathbf{y} = [y(2), ..., y(1000)]^T$, and

$$\boldsymbol{\phi} = \begin{pmatrix} -y(1) & u(1) \\ -y(2) & u(2) \\ \vdots & \vdots \\ -y(998) & u(998) \\ -y(999) & u(999) \end{pmatrix}$$

Matlab code for least squares estimator:

```
>> for i=2: 1000;
>>   phi(i,:)=[y(i-1) u(i-1)];
>> end;
>> theta=(phi'*phi)\phi'*y';
```

The model output is generated using

$$\hat{y}(t) = 0.8142y(t-1) + 1.0413u(t-1)$$

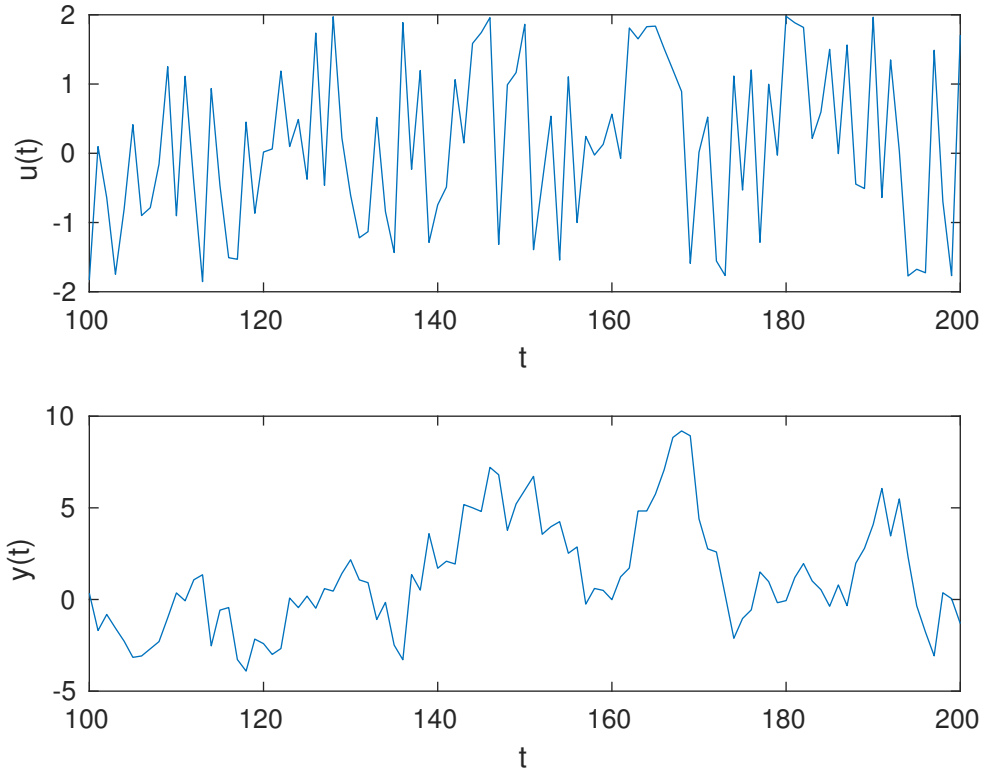


Figure 7: Upper: System 1 with a zero mean uniform signal in $[-2, 2]$ as input. Lower: Output of the ARMAX process described by System 1

Table 1: Parameter estimates for Example 3-1.

Parameter	True value	Estimated value
a	-0.8	-0.8142
b	1.0	1.0413

Matlab code for model output:

```
>> yhat=phi*theta;
>> figure(3);
>> i=1:1000;
>> plot(i,y(i),'-',i,yhat(i),'-.');
>> xlabel('t');
```

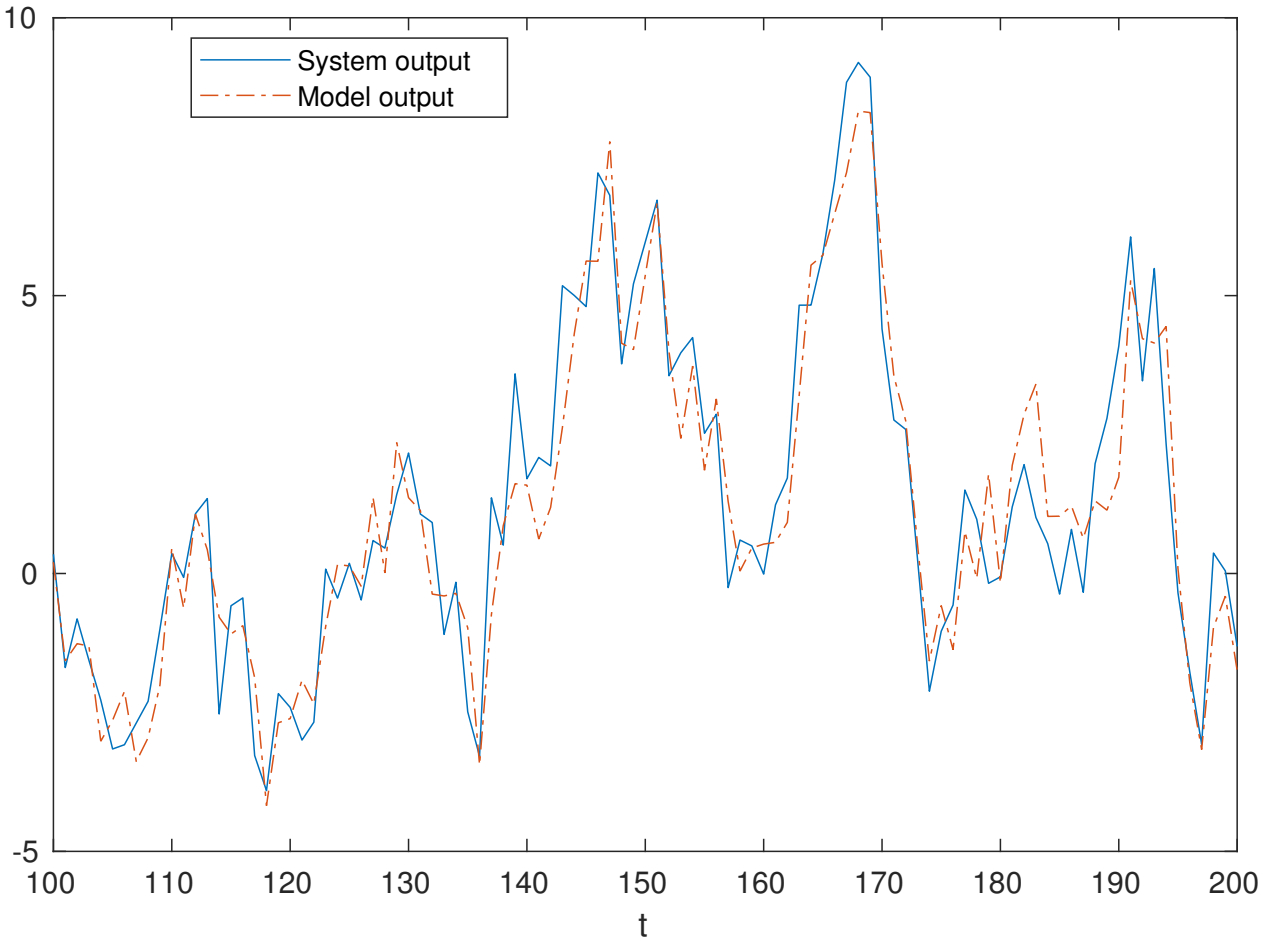


Figure 8: The model parameters computed by the least squares algorithm are used to generate the model output and show good agreement with the system output.

Recursive identification

Suppose at time instant $(n-1)$, we already calculated $\hat{\boldsymbol{\theta}}$, but now new output value $y(t)$ is available at time n, we need to recalculate $\hat{\boldsymbol{\theta}}$.

We would like a way of efficiently recalculating the model each time we have new data. Ideal form would be

$$\hat{\boldsymbol{\theta}}(n) = \hat{\boldsymbol{\theta}}(n-1) + correction \quad factor$$

Thus if the model is correct at time $(n-1)$ and the new data at time n is indicative of the model then the correction factor would be close to zero.

Example 3-2

Estimation of a constant. Assume that a model is

$$y(t) = b$$

This means a constant is to be determined from a number of noisy measurements $y(t)$, $t = 1, ..., n$.

$\boldsymbol{\theta} = b$, $\boldsymbol{\phi}(t) = 1$. $\mathbf{y} = [y(1), ..., y(n)]^T$, and

$$\boldsymbol{\phi} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}$$

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= [\boldsymbol{\phi}^T\boldsymbol{\phi}]^{-1}\boldsymbol{\phi}^T\mathbf{y} \\ &= \frac{1}{n}\sum_{t=1}^{n} y(t) \end{aligned}$$

This is simply the mean of the signal.

If we introduce a label $(n-1)$ to represent the fact that $n-1$ data points are used in deriving the mean, such that

$$\begin{aligned}\hat{\boldsymbol{\theta}}(n-1) =& [\boldsymbol{\phi}^T(n-1)\boldsymbol{\phi}(n-1)]^{-1}\boldsymbol{\phi}^T(n-1)\mathbf{y}(n-1)\\ =& \frac{1}{n-1}\sum_{t=1}^{n-1} y(t)\end{aligned}$$

Thus for n data points,

$$\begin{aligned}\hat{\boldsymbol{\theta}}(n) =& \frac{1}{n}\sum_{t=1}^{n} y(t)\\ =& \frac{1}{n}\{\sum_{t=1}^{n-1} y(t) + y(n)\}\\ =& \frac{1}{n}\{(n-1)\hat{\boldsymbol{\theta}}(n-1) + y(n)\}\\ =& \frac{1}{n}\{n\hat{\boldsymbol{\theta}}(n-1) - \hat{\boldsymbol{\theta}}(n-1) + y(n)\}\\ =& \hat{\boldsymbol{\theta}}(n-1) + \frac{1}{n}(y(n) - \hat{\boldsymbol{\theta}}(n-1))\\ =& \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n)(y(n) - \hat{\boldsymbol{\theta}}(n-1))\end{aligned}$$

with $\mathbf{K}(n) = \frac{1}{n}$. The above equation is the least squares algorithm in recursive form. General form of recursive algorithms is

$$\hat{\boldsymbol{\theta}}(n) = \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n)\varepsilon(n)$$

where $\hat{\boldsymbol{\theta}}(n-1)$ is a vector of model parameters estimates at time $(n-1)$.

$\hat{\boldsymbol{\theta}}(n)$ is a vector of model parameters estimates at time n.

$\varepsilon(n)$ is the difference between the measured output $y(n)$ and the model output using previous model with parameter vector $\hat{\boldsymbol{\theta}}(n-1)$.

$\mathbf{K}(n)$ is the scaling - also known as the Kalman Gain.

Advantages of recursive model estimation

- Gives an estimate of the model from the first time step.
- Can be computationally more efficient and less memory intensive, especially if we can avoid doing large matrix inverse calculations.
- Can be made to adapt to a changing system, e.g. online system identification allows telephone systems to do echo cancellation on long distance lines.
- Can be used for fault detection (model estimates start to differ radically from a norm).
- Forms the core of adaptive control strategies and adaptive signal processing.
- Ideal for real-time implementations.

Lecture 4: Recursive identification algorithm

In off-line or batch identification, data up to some $t = N$ is first collected, then the model parameter vector $\hat{\boldsymbol{\theta}}$ is computed.

In on-line or recursive identification, the model parameter vector $\hat{\boldsymbol{\theta}}(n)$ is required for every $t = n$. It is important that memory and computational time for updating the model parameter vector $\hat{\boldsymbol{\theta}}(n)$ do not increase with t. This poses constraints on how the estimates may be computed.

General form of recursive algorithms is

$$\hat{\boldsymbol{\theta}}(n) = \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n)\varepsilon(n)$$

In the following we will derive the recursive form of the least squares algorithm by modifying the off-line algorithm.

Derivation of recursive least squares

Suppose we have all the data collected up to time n. Then consider the formation of the regression matrix $\boldsymbol{\Phi}(n)$ as

$$\boldsymbol{\phi}(n) = \begin{pmatrix} \boldsymbol{\phi}^T(1) \\ \boldsymbol{\phi}^T(2) \\ \vdots \\ \boldsymbol{\phi}^T(n) \end{pmatrix}$$

and output vector $\mathbf{y}(n) = [y(1), ..., y(n)]^T$. Thus the least squares solution is

$$\hat{\boldsymbol{\theta}}(n) = [\boldsymbol{\Phi}(n)^T \boldsymbol{\Phi}(n)]^{-1} \boldsymbol{\Phi}(n)^T \mathbf{y}(n)$$

When a new data point comes, n is increased by 1, the least squares solution as above requires repetitive calculation including recalculating the inverse (expensive in computer time and storage)

Let's look at the expression $\boldsymbol{\Phi}(n)^T \boldsymbol{\Phi}(n)$ and $\boldsymbol{\Phi}(n)^T \mathbf{y}(n)$ and define $\mathbf{P}^{-1}(n) = \boldsymbol{\Phi}(n)^T \boldsymbol{\Phi}(n)$

$$\begin{aligned}
\mathbf{P}^{-1}(n) &= \boldsymbol{\Phi}(n)^T \boldsymbol{\Phi}(n) \\
&= [\boldsymbol{\phi}(1), \boldsymbol{\phi}(2), ..., \boldsymbol{\phi}(n)] \begin{pmatrix} \boldsymbol{\phi}^T(1) \\ \boldsymbol{\phi}^T(2) \\ \vdots \\ \boldsymbol{\phi}^T(n) \end{pmatrix} \\
&= \sum_{t=1}^{n} \boldsymbol{\phi}(t)\boldsymbol{\phi}^T(t) \\
&= \sum_{t=1}^{n-1} \boldsymbol{\phi}(t)\boldsymbol{\phi}^T(t) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n) \\
&= \mathbf{P}^{-1}(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n) \\
\boldsymbol{\Phi}(n)^T \mathbf{y}(n) &= [\boldsymbol{\phi}(1), \boldsymbol{\phi}(2), ..., \boldsymbol{\phi}(n)] \begin{pmatrix} y(1) \\ y(2) \\ \vdots \\ y(n) \end{pmatrix} \\
&= \sum_{t=1}^{n} \boldsymbol{\phi}(t) y(t) \\
&= \sum_{t=1}^{n-1} \boldsymbol{\phi}(t) y(t) + \boldsymbol{\phi}(n) y(n) \\
&= \boldsymbol{\Phi}(n-1)^T \mathbf{y}(n-1) + \boldsymbol{\phi}(n) y(n)
\end{aligned}$$

$$\mathbf{P}^{-1}(n) = \mathbf{P}^{-1}(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n) \tag{7}$$

$$\boldsymbol{\Phi}(n)^T \mathbf{y}(n) = \boldsymbol{\Phi}(n-1)^T \mathbf{y}(n-1) + \boldsymbol{\phi}(n) y(n) \tag{8}$$

The least squares estimate at time n

$$\begin{aligned}
\hat{\boldsymbol{\theta}}(n) &= [\boldsymbol{\Phi}(n)^T \boldsymbol{\Phi}(n)]^{-1} \boldsymbol{\phi}(n)^T \mathbf{y}(n) \\
&= \mathbf{P}(n) \boldsymbol{\phi}(n)^T \mathbf{y}(n) \\
&= \mathbf{P}(n) \left(\boldsymbol{\Phi}(n-1)^T \mathbf{y}(n-1) + \boldsymbol{\phi}(n) y(n) \right)
\end{aligned} \tag{9}$$

Because the least squares estimate at time $(n-1)$ is given by

$$\hat{\boldsymbol{\theta}}(n-1) = \mathbf{P}(n-1) \boldsymbol{\phi}(n-1)^T \mathbf{y}(n-1),$$

$$\text{so} \quad \mathbf{P}^{-1}(n-1) \hat{\boldsymbol{\theta}}(n-1) = \boldsymbol{\phi}(n-1)^T \mathbf{y}(n-1) \tag{10}$$

Substitute (10) into (9)

$$\begin{aligned}
\hat{\boldsymbol{\theta}}(n) =& \mathbf{P}(n) \left(\mathbf{P}^{-1}(n-1) \hat{\boldsymbol{\theta}}(n-1) + \boldsymbol{\phi}(n) y(n) \right) \\
& \longleftarrow \text{ Applying (7)} \\
=& \mathbf{P}(n) \left(\mathbf{P}^{-1}(n) \hat{\boldsymbol{\theta}}(n-1) - \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n) \hat{\boldsymbol{\theta}}(n-1) + \boldsymbol{\phi}(n) y(n) \right) \\
=& \hat{\boldsymbol{\theta}}(n-1) + \mathbf{P}(n) \boldsymbol{\phi}(n) \left(y(n) - \boldsymbol{\phi}^T(n) \hat{\boldsymbol{\theta}}(n-1) \right) \\
=& \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n) \varepsilon(n)
\end{aligned}$$

Thus the recursive least squares (RLS) are

$$\begin{aligned}
\varepsilon(n) &= y(n) - \boldsymbol{\phi}^T(n)\hat{\boldsymbol{\theta}}(n-1) && (11)\\
\mathbf{P}(n) &= (\mathbf{P}^{-1}(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n))^{-1} && (12)\\
\mathbf{K}(n) &= \mathbf{P}(n)\boldsymbol{\phi}(n) && (13)\\
\hat{\boldsymbol{\theta}}(n) &= \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n)\varepsilon(n) && (14)
\end{aligned}$$

Yet we still require a matrix inverse to be calculated in (6).

However this can be avoided through matrix inversion lemma (to be derived next week).

Lecture 5: Derivation of recursive least squares (continued)

Matrix inversion Lemma: If $\mathbf{A}$, $\mathbf{C}$, $\mathbf{BCD}$ are nonsingular square matrix (the inverse exists) then

$$[\mathbf{A}+\mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}]^{-1}\mathbf{DA}^{-1}$$

The best way to prove this is to multiply both sides by $[\mathbf{A}+\mathbf{BCD}]$.

$$\begin{aligned}
& [\mathbf{A}+\mathbf{BCD}][\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}]^{-1}\ \mathbf{DA}^{-1}]\\
=\ & \mathbf{I} + \mathbf{BCDA}^{-1} - \mathbf{B}[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}]^{-1}\mathbf{DA}^{-1}\\
& -\mathbf{BCDA}^{-1}\mathbf{B}[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}]^{-1}\mathbf{DA}^{-1}\\
=\ & \mathbf{I} + \mathbf{BCDA}^{-1} - \mathbf{B}\underbrace{\mathbf{CC}^{-1}}_{\mathbf{I}}[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}]^{-1}\mathbf{DA}^{-1}\\
& -\mathbf{BCDA}^{-1}\mathbf{B}[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}]^{-1}\mathbf{DA}^{-1}\\
=\ & \mathbf{I} + \mathbf{BCDA}^{-1}\\
& -\mathbf{BC}\underbrace{\{\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}\}[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}]^{-1}}_{\mathbf{I}}\mathbf{DA}^{-1}\\
=\ & \mathbf{I}
\end{aligned}$$

Now look at the derivation of RLS algorithm so far and consider applying the matrix inversion lemma to (2) below

$$\begin{aligned}
\varepsilon(n) &= y(n) - \boldsymbol{\phi}^T(n)\hat{\boldsymbol{\theta}}(n-1) && (15)\\
\mathbf{P}(n) &= (\mathbf{P}^{-1}(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n))^{-1} && (16)\\
\mathbf{K}(n) &= \mathbf{P}(n)\boldsymbol{\phi}(n) && (17)\\
\hat{\boldsymbol{\theta}}(n) &= \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n)\varepsilon(n) && (18)
\end{aligned}$$

with

$$\mathbf{A} = \mathbf{P}^{-1}(n-1), \quad \mathbf{B} = \boldsymbol{\phi}(n)$$
$$\mathbf{C} = 1, \quad \mathbf{D} = \boldsymbol{\phi}^T(n)$$

$$\begin{aligned}
\mathbf{P}(n) &= (\mathbf{P}^{-1}(n-1) + \boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n))^{-1}\\
&= \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}]^{-1}\mathbf{DA}^{-1}\\
&= \mathbf{P}(n-1)\\
&\quad -\mathbf{P}(n-1)\boldsymbol{\phi}(n)[1 + \boldsymbol{\phi}^T(n)\mathbf{P}(n-1)\boldsymbol{\phi}(n)]^{-1}\boldsymbol{\phi}^T(n)\mathbf{P}(n-1)\\
&= \mathbf{P}(n-1) - \frac{\mathbf{P}(n-1)\boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n)\mathbf{P}(n-1)}{1 + \boldsymbol{\phi}^T(n)\mathbf{P}(n-1)\boldsymbol{\phi}(n)}
\end{aligned}$$

The RLS algorithm is

$$\begin{aligned}
\varepsilon(n) &= y(n) - \boldsymbol{\phi}^T(n)\hat{\boldsymbol{\theta}}(n-1)\\
\mathbf{P}(n) &= \mathbf{P}(n-1) - \frac{\mathbf{P}(n-1)\boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n)\mathbf{P}(n-1)}{1 + \boldsymbol{\phi}^T(n)\mathbf{P}(n-1)\boldsymbol{\phi}(n)}\\
\mathbf{K}(n) &= \mathbf{P}(n)\boldsymbol{\phi}(n)\\
\hat{\boldsymbol{\theta}}(n) &= \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n)\varepsilon(n)
\end{aligned}$$

Here the term $\varepsilon(n)$ should be interpreted as a prediction error. It is the difference between the measured output $y(n)$ and the one-step-ahead prediction

$$\hat{y}(n|n-1, \hat{\boldsymbol{\theta}}(n-1)) = \boldsymbol{\phi}^T(n)\hat{\boldsymbol{\theta}}(n-1)$$

made at time $t = (n-1)$. If $\varepsilon(n)$ is small $\hat{\boldsymbol{\theta}}(n-1)$ is good and should not be modified very much.

$\mathbf{K}(n)$ should be interpreted as a weighting factor showing how much the value of $\varepsilon(n)$ will modify the different elements of the parameter vector.

The algorithm also needs initial values of $\hat{\boldsymbol{\theta}}(0)$ and $\mathbf{P}(0)$. It is convenient to set the initial values of $\hat{\boldsymbol{\theta}}(0)$ to zeros and the initial value of $\mathbf{P}(0)$ to $LN \times \mathbf{I}$, where $\mathbf{I}$ is the identity matrix and LN is a large number.

Example 5-1:

(see Example 3-2) Estimation of a constant.

Assume that a model is

$$y(t) = b$$

This means a constant is to be determined from a number of noisy measurements $y(t)$, $t = 1, ..., n$.

Since $\boldsymbol{\phi}(n) = 1$

$$\mathbf{P}(n) = \mathbf{P}(n-1) - \frac{\mathbf{P}(n-1)\boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n)\mathbf{P}(n-1)}{1 + \boldsymbol{\phi}^T(n)\mathbf{P}(n-1)\boldsymbol{\phi}(n)}$$

$$= \mathbf{P}(n-1) - \frac{\mathbf{P}^2(n-1)}{1 + \mathbf{P}(n-1)}$$

$$= \frac{\mathbf{P}(n-1)}{1 + \mathbf{P}(n-1)}$$

i.e.

$$\mathbf{P}^{-1}(n) = \mathbf{P}^{-1}(n-1) + 1 = n$$

$$\mathbf{K}(n) = \mathbf{P}(n) = \frac{1}{n}$$

Thus $\hat{\boldsymbol{\theta}}(n) = \hat{\boldsymbol{\theta}}(n-1) + \frac{1}{n}\varepsilon(n)$ coincides with the results of Example 2 in Lecture 3.

Example 5-2:

The system input/output of a process is measured as in the following Table. Suppose the process can be described by a model $y(t) = ay(t-1) + bu(t-1) = \boldsymbol{\phi}^T(t)\boldsymbol{\theta}$. If a recursive least squares algorithm is applied for on-line parameter estimation and at time

$t = 2, \hat{\boldsymbol{\theta}}(2) = [0.8, 0.1]^T, \mathbf{P}(2) = \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix}$, find $\hat{\boldsymbol{\theta}}(3)$, $\mathbf{P}(3)$.

t	1	2	3	4
$y(t)$	0.5	0.6	0.4	0.5
$u(t)$	0.3	0.4	0.5	0.7

Note $\boldsymbol{\phi}(t) = [y(t-1), u(t-1)]^T$

$$\begin{aligned} \varepsilon(3) &= y(3) - \boldsymbol{\phi}^T(3)\hat{\boldsymbol{\theta}}(2) \\ &= 0.4 - [0.6, 0.4]\begin{bmatrix} 0.8 \\ 0.1 \end{bmatrix} = -0.12 \end{aligned}$$

$$\mathbf{P}(3) = \mathbf{P}(2) - \frac{\mathbf{P}(2)\boldsymbol{\phi}(3)\boldsymbol{\phi}^T(3)\mathbf{P}(2)}{1+\boldsymbol{\phi}^T(3)\mathbf{P}(2)\boldsymbol{\phi}(3)}$$

$$= \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix} - \frac{\begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix}\begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}_{[0.6,0.4]}\begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix}}{1+_{[0.6,0.4]}\begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix}\begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}}$$

$$= \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix} - \frac{\begin{bmatrix} 600 \\ 400 \end{bmatrix}^{[600,\ 400]}}{1+[600,400]\begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}}$$

$$= \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix} - \frac{\begin{bmatrix} 360000 & 240000 \\ 240000 & 360000 \end{bmatrix}}{521}$$

$$= \begin{bmatrix} 309.0211 & -460.6526 \\ -460.6526 & 309.0211 \end{bmatrix}$$

$$\begin{aligned} \mathbf{K}(3) &= \mathbf{P}(3)\boldsymbol{\phi}(3) \\ &= \begin{bmatrix} 309.0211 & -460.6526 \\ -460.6526 & 309.0211 \end{bmatrix} \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix} \\ &= \begin{bmatrix} 1.1516 \\ -152.7831 \end{bmatrix} \\ \hat{\boldsymbol{\theta}}(3) &= \hat{\boldsymbol{\theta}}(2) + \mathbf{K}(3)\varepsilon(3) \\ &= \begin{bmatrix} 0.8 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 1.1516 \\ -152.7831 \end{bmatrix}(-0.12) \\ &= \begin{bmatrix} 0.6618 \\ 18.4340 \end{bmatrix} \end{aligned}$$

RLS algorithm with a forgetting factor

The RLS algorithm can be modified for tracking time varying parameters. One approach is the RLS algorithm with a forgetting factor. For a linear regression model

$$y(t) = \boldsymbol{\phi}^T(t)\boldsymbol{\theta} + e(t)$$

The loss function to be minimized in a least squares algorithm is

$$V(\boldsymbol{\theta}) = \sum_{t=1}^{n} [y(t) - \boldsymbol{\phi}^T(t)\boldsymbol{\theta}]^2$$

then we minimize this with respect to $\boldsymbol{\theta}$. Consider modifying the loss function to

$$V(\boldsymbol{\theta}) = \sum_{t=1}^{n} \lambda^{(n-t)} [y(t) - \boldsymbol{\phi}^T(t)\boldsymbol{\theta}]^2$$

where $\lambda < 1$ is the forgetting factor. e.g. $\lambda = 0.99$ or 0.95. This means that as n increases, the measurements obtained previously are discounted. Older data has less effect on the coefficient estimation, hence "forgotten".

The RLS algorithm with a forgetting factor is

$$\begin{aligned} \varepsilon(n) &= y(n) - \boldsymbol{\phi}^T(n)\hat{\boldsymbol{\theta}}(n-1) \\ \mathbf{P}(n) &= \frac{1}{\lambda}\{\mathbf{P}(n-1) - \frac{\mathbf{P}(n-1)\boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n)\mathbf{P}(n-1)}{\lambda + \boldsymbol{\phi}^T(n)\mathbf{P}(n-1)\boldsymbol{\phi}(n)}\} \\ \mathbf{K}(n) &= \mathbf{P}(n)\boldsymbol{\phi}(n) \\ \hat{\boldsymbol{\theta}}(n) &= \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n)\varepsilon(n) \end{aligned}$$

When $\lambda = 1$ this is simply the RLS algorithm.

The smaller the value of λ, the quicker the information in previous data will be forgotten.

Using the RLS algorithm with a forgetting factor, we may speak about "real time identification". When the properties of the process may change (slowly) with time, the algorithm is able to track the time-varying parameters describing such process.

Summary of the RLS algorithm:

1. Initialization: Set n_a, n_b, λ, $\hat{\boldsymbol{\theta}}(0)$ and $\mathbf{P}(0)$. Step 2-5 are repeated starting from $t = 1$.
2. At time step $t = n$, measure current output $y(n)$.
3. Recall past y's and u's and form $\boldsymbol{\phi}(n)$.
4. Apply RLS algorithm for $\hat{\boldsymbol{\theta}}(n)$ and $\mathbf{P}(n)$
5. $\hat{\boldsymbol{\theta}}(n) \to \hat{\boldsymbol{\theta}}(n-1)$ and $\mathbf{P}(n) \to \mathbf{P}(n-1)$
6. $t = n + 1$, go to step 2.

Lecture 6

Example 6-1:

Consider a system described by

$$y(t) + a_1 y(t-1) + a_2 y(t-2) = b_1 u(t-1) + e(t)$$

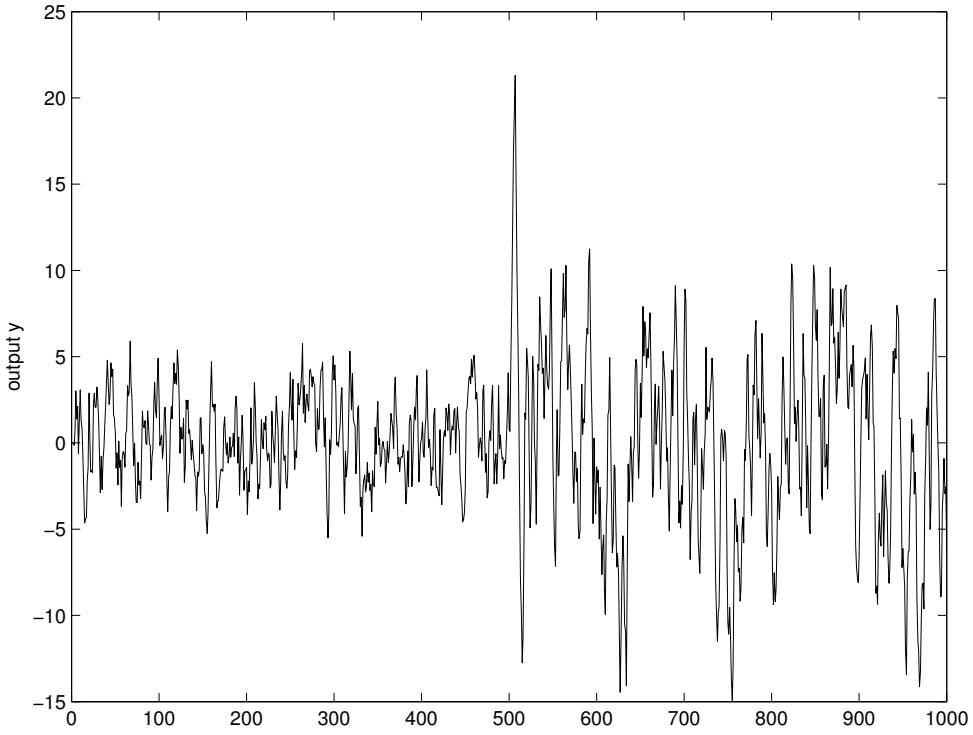
where $\{e(t)\}$ is a white noise sequence with variance 1. The input $u(t)$ is a uniformly distributed random signal in [-2,2]. The system parameters are

$$a_1 = -0.8, a_2 = 0, b_1 = 1.0$$

for the first 500 data samples and then these change to

$$a_1 = -1.1, a_2 = 0.3, b_1 = 2.0$$

for the next 500 data samples.



Bias,consistency and model approximation

An estimate $\hat{\boldsymbol{\theta}}$ is biased if its expected value deviates from the true value.

$$E(\hat{\boldsymbol{\theta}}) \neq \boldsymbol{\theta}$$

where E notation is the expected value. The difference $E(\hat{\boldsymbol{\theta}}) - \boldsymbol{\theta}$ is called the bias.

(The expected value of a variable is the average value that the variable would have if averaged over an extremely long period of time ($N \to \infty$), $E \approx \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N}$)

Example 6-2:

Suppose a system (System 2 in Lecture 3) is given by

$$y(t) + ay(t-1) = bu(t-1) + ce(t-1) + e(t)$$

where $\{e(t)\}$ is a white noise sequence with variance σ^2, with $a = -0.8, b = 1.0, c = -0.8, \sigma = 1.0$

System 2 is simulated generating 1000 data points. The input $u(t)$ is a uniformly distributed random signal in [-2,2].

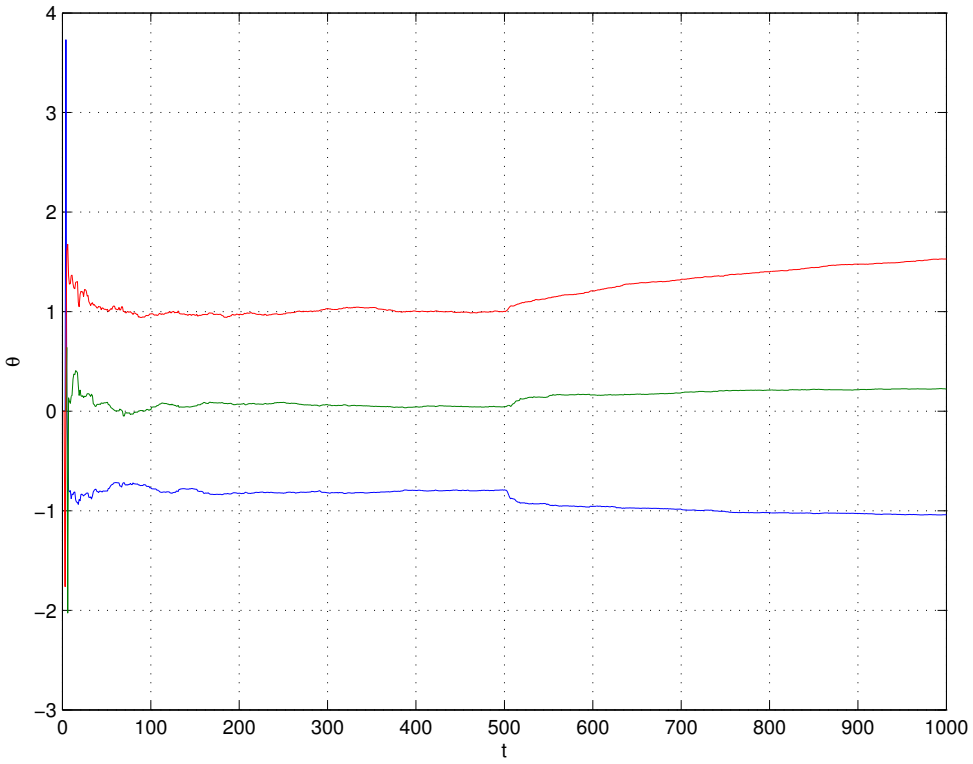


Figure 9: Evolution of parameter estimates with RLS algorithm ($\lambda = 1$)

Matlab code generating data (system 1):

```
>> u=2-4*rand(1,1000); % zero mean uniform distribution
>> e=randn(1,1000);
>> y(1)=0;
>> for i=2:1000
>>    y(i)=0.8*y(i-1) + u(i-1) -0.8*e(i-1)+ e(i);
>> end
>> figure(1); plot(u);xlabel('t');ylabel('u(t)');
>> figure(2); plot(y);xlabel('t');ylabel('y(t)');
```

Suppose that we still use the model

$$\hat{y}(t) = -ay(t-1) + bu(t-1)$$

to model the system (see Example 1, Lecture 3).

Table 2: Parameter estimates for System 1 (taken from Example 1, Lecture 3).

Parameter	True value	Estimated value
a	-0.8	-0.8142
b	1.0	1.0413

Table 3: Parameter estimates for System 2.

Parameter	True value	Estimated value
a	-0.8	-0.6122
b	1.0	0.9581

Thus for System 2 the estimate $\hat{a}$ will deviate from the true value. We say that an estimate is consistent if

$$E(\hat{\boldsymbol{\theta}}) = \boldsymbol{\theta}$$

It seems that the parameter estimator for System 1 is consistent, but not System 2. The models can be regarded as the approximation to the system. The model using the derived LS parameter estimator for System 2 is not good.

Recall that System 1 and System 2 are given by

$$y(t) - 0.8y(t-1) = u(t-1) + v(t)$$

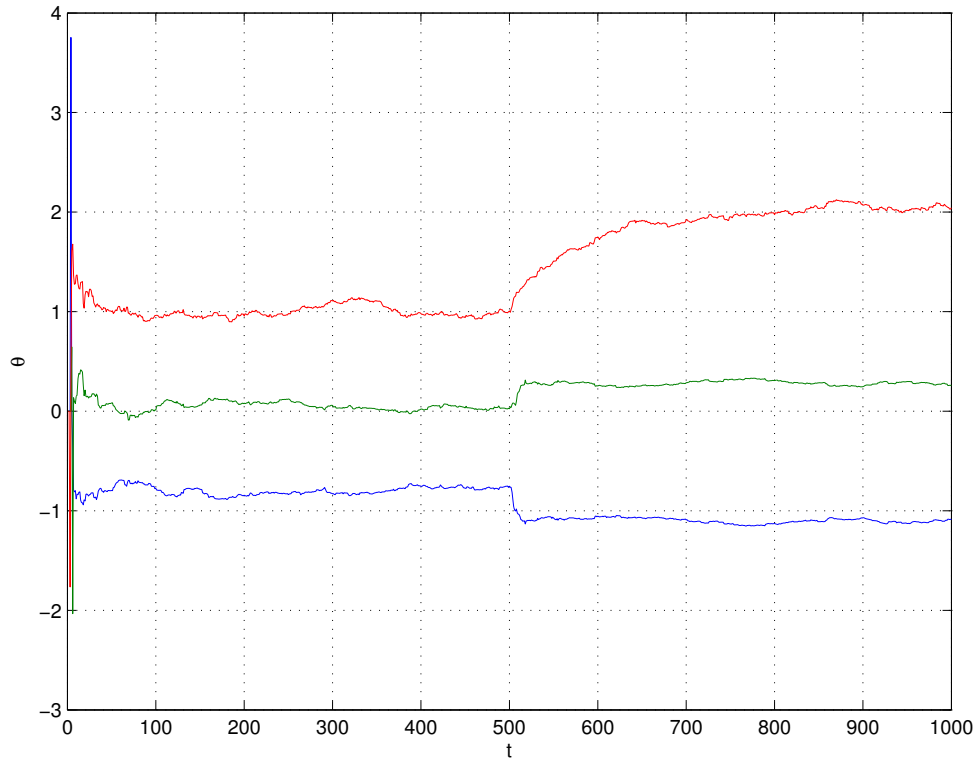


Figure 10: Evolution of parameter estimates with RLS algorithm ($\lambda = 0.99$)

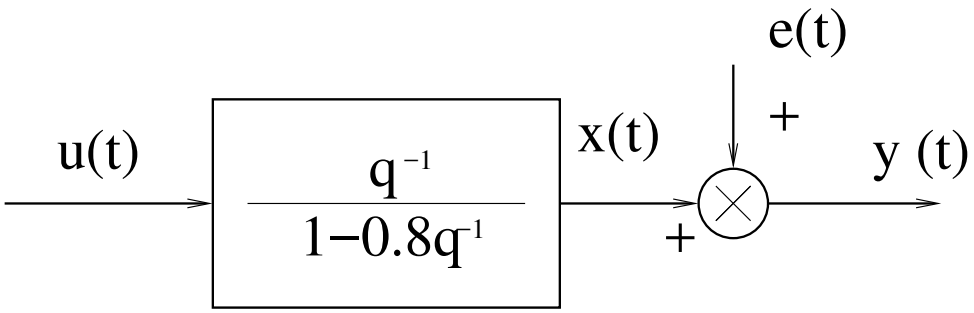


Figure 11: System 2

with

System 1: $v(t) = e(t)$ as a white noise.

System 2: $v(t) = -0.8e(t-1) + e(t)$ as a moving average (MA) of a white noise sequence $\{e(t)\}$.

Properties of the least squares estimate

Assume that the data obey the linear regression

$$y(t) = \hat{y}(t) + e(t) = \boldsymbol{\phi}^T(t)\boldsymbol{\theta} + e(t)$$

or in matrix form

$$\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{e}$$

where $e(t)$ is a stochastic variable with zero mean and variance σ^2.

Lemma 1: Assume further that $e(t)$ is a white noise sequence and consider the estimate

$$\hat{\boldsymbol{\theta}} = [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{y}.$$

The following properties hold :

(i) $\hat{\boldsymbol{\theta}}$ is an unbiased estimate of $\boldsymbol{\theta}$.

(ii) The covariance matrix of $\hat{\boldsymbol{\theta}}$ is given by

$$\text{cov}(\hat{\boldsymbol{\theta}}) = \sigma^2[\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}.$$

Proof: (i)

$$\hat{\boldsymbol{\theta}} = [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{y}$$
$$= [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{e})$$
$$= \boldsymbol{\theta} + [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{e}$$
$$E(\hat{\boldsymbol{\theta}}) = \boldsymbol{\theta} + E([\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{e})$$
$$= \boldsymbol{\theta} + [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T E(\mathbf{e}) = \boldsymbol{\theta}$$

(ii)

$$E\left[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^T\right]$$
$$= E\left[[\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{e}([\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{e})^T\right]$$
$$= E\left[[\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{e}\mathbf{e}^T\boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\right]$$
$$= [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T E(\mathbf{e}\mathbf{e}^T)\boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}$$

Note that the white noise assumption implies that $E(\mathbf{e}\mathbf{e}^T) = \sigma^2\mathbf{I}$

$$\text{cov}(\hat{\boldsymbol{\theta}}) = [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\sigma^2\mathbf{I}\boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}$$
$$= \sigma^2[\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}$$

Assuming that $\frac{\boldsymbol{\Phi}^T\boldsymbol{\Phi}}{N}$ is a finite full rank matrix, we have

$$\text{cov}(\hat{\boldsymbol{\theta}}) = \frac{\sigma^2}{N}[\frac{\boldsymbol{\Phi}^T\boldsymbol{\Phi}}{N}]^{-1} \to 0 \quad \text{as} \quad N \to \infty$$

Input signals

The input signal used in a system identification experiment can have a significant influence on the resulting parameter estimates. Some conditions on the input sequence must be introduced to yield a reasonable identification results. As a trivial illustration, we may think of an input that is identically zero. Such an input will not be able to yield full information about the input/output relationship of the system.

The input should excite the typical modes of the system, and this is called *persistent exciting*. In mathematical term this is related to the full rank condition of $\frac{\boldsymbol{\Phi}^T\boldsymbol{\Phi}}{N}$.

1. Step function: A step function is given by

$$u(t) = \begin{cases} 0 & t < 0 \\ u_0 & t \geq 0 \end{cases}$$

The users has to choose the amplitude u_0. For systems with a large signal-to-noise ratio, and step input can given information about the dynamics. Rise time, overshoot and static gain are directly related to the step response. Also the major time constants and a possible resonance can be at least crudely estimated from the step response. (*ones.m*)

2. Uniformly distributed pseudo-random numbers: *rand.m* generating a sequence drawn from a uniform distribution on the unit interval.

3. ARMA – Autoregressive moving average process:

$$C(q^{-1})u(t) = D(q^{-1})e(t)$$

where

$$C(q^{-1}) = 1 + c_1q^{-1} + ... + c_{n_c}q^{-n_c}$$
$$D(q^{-1}) = 1 + d_1q^{-1} + ... + d_{n_d}q^{-n_d}$$

are set by the user. Different choices of the filter parameters c_i and d_i lead to input signal with various frequency contents, thus emphasizing the identification in different frequency ranges. (*filter.m*)

4. Sum of sinusoids

$$u(t) = \sum_{j=1}^{m} a_j \sin(\omega_j t + \varphi_j)$$

where the angular frequency ω_j are distinct. $0 \leq \omega_1 < \omega_2 < ... < \omega_m \leq \pi$. The user has to choose a_j ω_j, and φ_j.

5. The input can be (partly) a feedback signal from the output (closed loop identification using a known feedback)

Lecture 7: Pseudo linear regression for ARMAX model

Consider the ARMAX model

$$A(q^{-1})y(t) = B(q^{-1})u(t) + C(q^{-1})e(t)$$

or

$$\begin{aligned} y(t) &= -a_1 y(t-1) - ... - a_{n_a} y(t-n_a) \\ &\quad + b_1 u(t-1) + ... + b_{n_b} u(t-n_b) \\ &\quad + c_1 e(t-1) + ... + c_{n_c} e(t-n_c) + e(t) \end{aligned}$$

To simply the notation, we study a first order system:

$$y(t) = -ay(t-1) + bu(t-1) + ce(t-1) + e(t) \tag{19}$$

If we rewrite (1) as

$$y(t) = \boldsymbol{\phi}^T(t)\boldsymbol{\theta} + v(t)$$

in which $v(t) = ce(t-1) + e(t)$ is moving average of $e(t)$.

and

$$\boldsymbol{\phi} = [-y(t-1), u(t-1)]^T, \quad \boldsymbol{\theta} = [a, b]^T$$

In matrix form

$$\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{v}$$

The least squares estimate

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{y} \\ &= [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T(\boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{v}) \\ &= \boldsymbol{\theta} + [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{v} \\ E(\hat{\boldsymbol{\theta}}) &= \boldsymbol{\theta} + E([\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{v}) \end{aligned}$$

The second term $E([\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{v}) \neq 0$ is the bias. This is due to $v(t)$ is a correlated signal (colored noise)

However (1) can be rewritten as

$$y(t) = \boldsymbol{\phi}^T(t)\boldsymbol{\theta} + e(t)$$

with

$$\begin{aligned} \boldsymbol{\phi} &= [-y(t-1), u(t-1), e(t-1)]^T \\ \boldsymbol{\theta} &= [a, b, c]^T \end{aligned}$$

Here in $\boldsymbol{\phi}$, $y(t-1)$ and $u(t-1)$ are known at time t, but $e(t-1)$ is unknown. However we can replace it using the prediction error $\varepsilon(t-1)$. $\varepsilon(0)$ can be initialized as 0.

For this system, the RLS algorithm is given by

$$\begin{aligned} \boldsymbol{\phi}(n) &= [-y(n-1), u(n-1), \varepsilon(n-1)]^T \\ \varepsilon(n) &= y(n) - \boldsymbol{\phi}^T(n)\hat{\boldsymbol{\theta}}(n-1) \\ \mathbf{P}(n) &= \frac{1}{\lambda}\{\mathbf{P}(n-1) - \frac{\mathbf{P}(n-1)\boldsymbol{\phi}(n)\boldsymbol{\phi}^T(n)\mathbf{P}(n-1)}{\lambda + \boldsymbol{\phi}^T(n)\mathbf{P}(n-1)\boldsymbol{\phi}(n)}\} \\ \mathbf{K}(n) &= \mathbf{P}(n)\boldsymbol{\phi}(n) \\ \hat{\boldsymbol{\theta}}(n) &= \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n)\varepsilon(n) \end{aligned}$$

Summary of the RLS algorithm for ARMAX:

1. Initialization: Set n_a,n_b, n_c, λ, $\hat{\boldsymbol{\theta}}(0)$ and $\mathbf{P}(0)$. $\varepsilon(0), ..., \varepsilon(1-n_c) = 0$. Step 2-5 are repeated starting from $t = 1$.
2. At time step $t = n$, measure current output $y(n)$.
3. Recall past y's and u's and ε's form $\boldsymbol{\phi}(n)$.
4. Apply RLS algorithm for $\varepsilon(n)$, $\hat{\boldsymbol{\theta}}(n)$ and $\mathbf{P}(n)$
5. $\hat{\boldsymbol{\theta}}(n) \to \hat{\boldsymbol{\theta}}(n-1)$ and $\mathbf{P}(n) \to \mathbf{P}(n-1)$ and $\varepsilon(n) \to \varepsilon(n-1)$.
6. $t = n+1$, go to step 2.

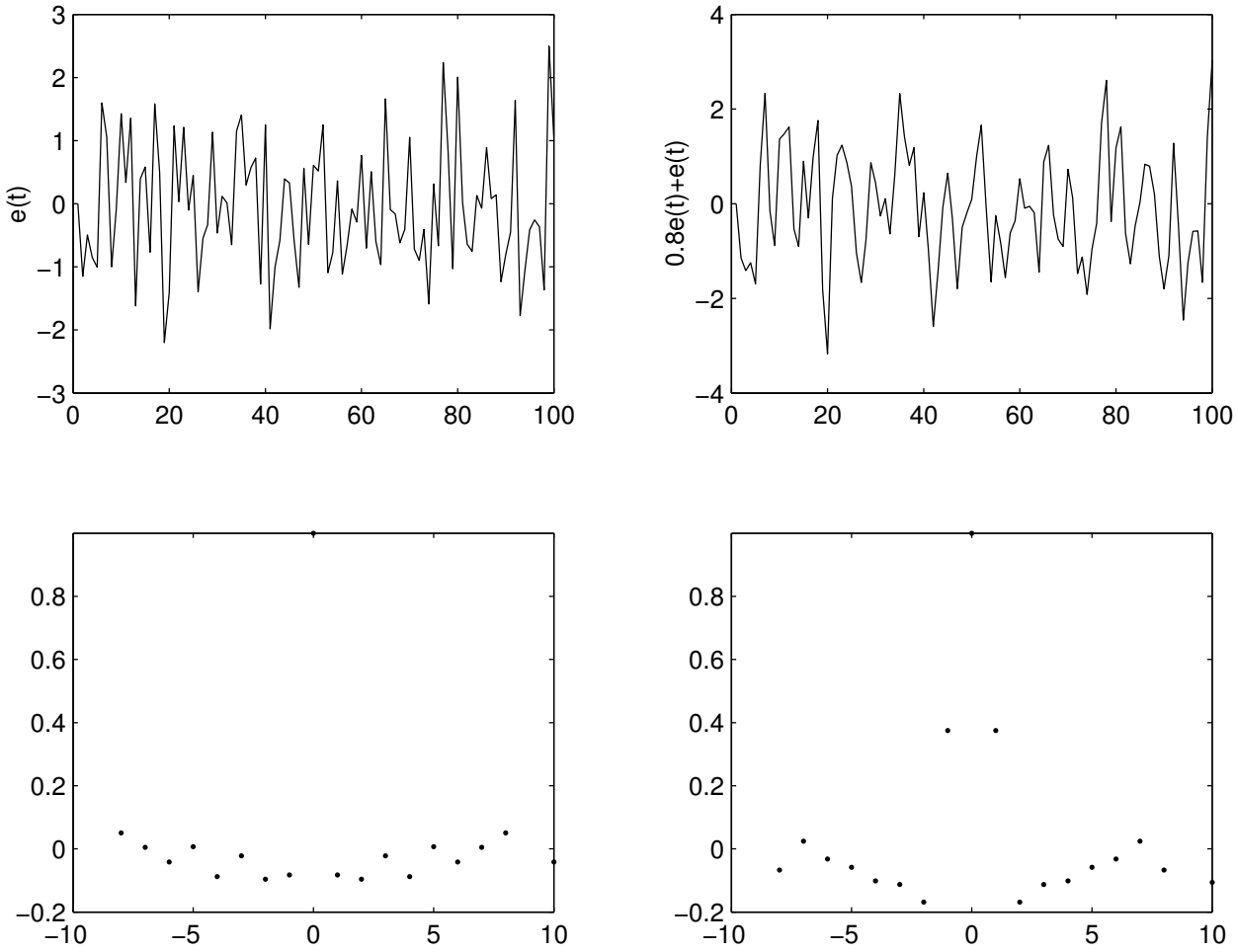


Figure 12: Comparison of autocorrelation coefficients

Example 7-1:

Consider a system described by

$$y(t) + ay(t-1) = bu(t-1) + ce(t-1) + e(t)$$

where $\{e(t)\}$ is a white noise sequence with variance 1. The system parameters are

$$a = -0.9, b = 1.0, c = 0.3$$

The input

$$u(t) = 2\sin(\frac{\pi t}{50} + \frac{\pi}{3}) + \sin(\frac{\pi t}{30})$$

1000 data samples are generated.

Determining the model structure

For a given data set, consider using the linear regression

$$y(t) \quad = \quad \boldsymbol{\phi}^T(t)\boldsymbol{\theta} + e(t)$$

with a sequence of model structures of increasing dimension to obtain the best fitted models within each of the model structures.

e.g. The first model is $(n_a = 1, n_b = 1)$

$$y(t) \quad = \quad -a_1 y(t-1) + b_1 u(t-1) + e(t)$$

and the second model is $(n_a = 2, n_b = 2)$

$$\begin{aligned} y(t) \quad = \quad & -a_1 y(t-1) - a_2 y(t-2) \\ & +b_1 u(t-1) + b_2 u(t-2) + e(t) \end{aligned}$$

With more free parameters in the second model, a better fit will be obtained. In determining the best model structure, the important thing is to investigate whether or not the improvement in the fit is significant.

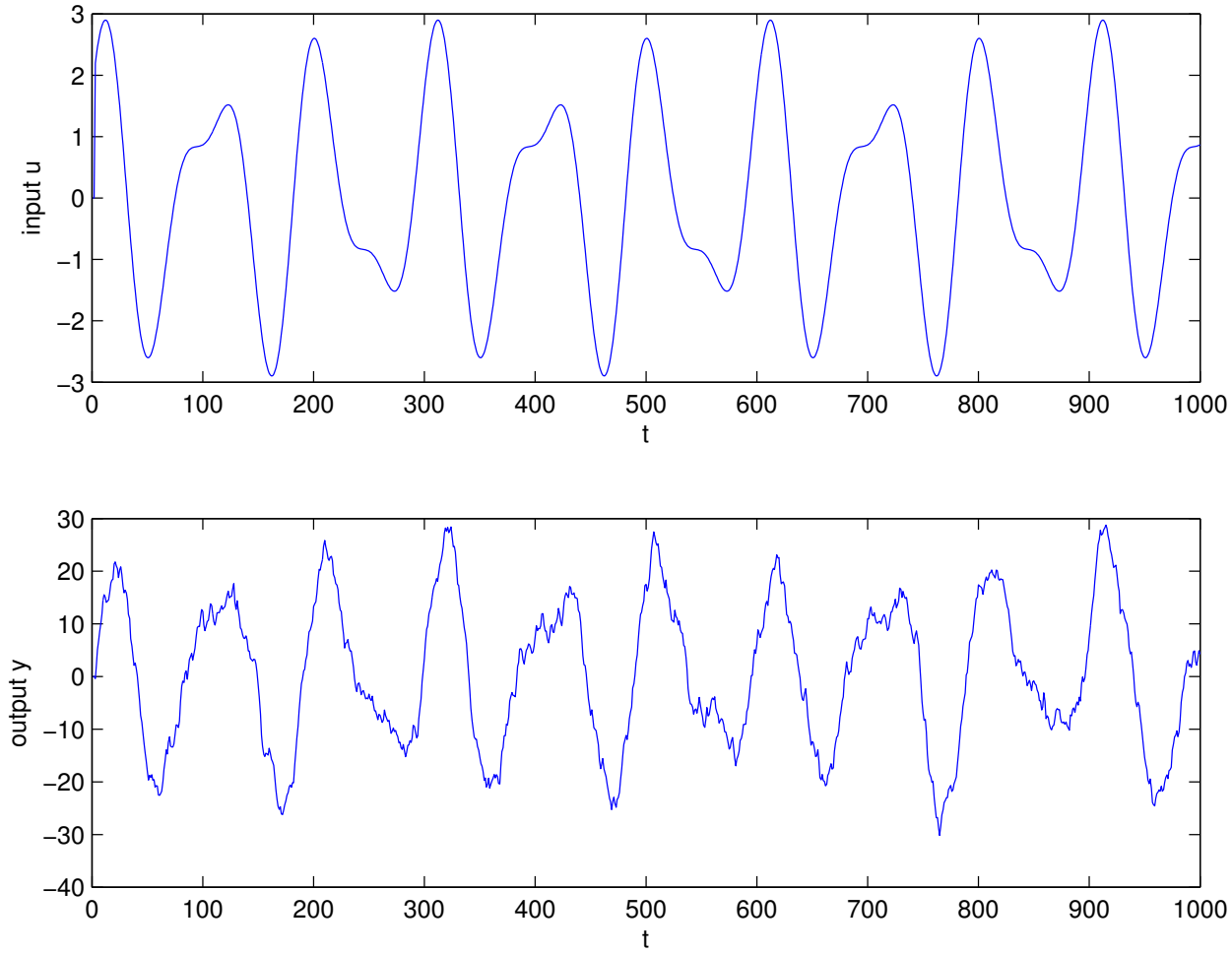


Figure 13: System input and output (dataset 7-3).

The model fit is given by the loss function $V(\hat{\boldsymbol{\theta}}) = \frac{1}{N}\sum_{t=1}^{N}\varepsilon^2(t)$ for each model. Depending on the data set, we may obtain either of the following Figures:

If a model is more complex than is necessary, the model may be over-parameterized.

Example 7-2.

(Over-parameterization) Suppose that the true signal $y(t)$ is described by the ARMA process

$$y(t) + ay(t-1) = e(t) + ce(t-1)$$

Let the model set be given by

$$\begin{aligned} &y(t) + a_1 y(t-1) + a_2 y(t-2) \\ &= e(t) + c_1 e(t-1) + c_2 e(t-2) \end{aligned}$$

we see that all a_i, c_i such that

$$\begin{aligned} 1 + a_1 q^{-1} + a_2 q^{-2} &= (1 + aq^{-1})(1 + dq^{-1}) \\ 1 + c_1 q^{-1} + c_2 q^{-2} &= (1 + cq^{-1})(1 + dq^{-1}) \end{aligned}$$

with an arbitrary number d give the same description of the system. a_i, c_i can not be uniquely determined, we may say that model is over-parameterized. Consequently $\frac{\boldsymbol{\Phi}^T\boldsymbol{\Phi}}{N}$ is singular.

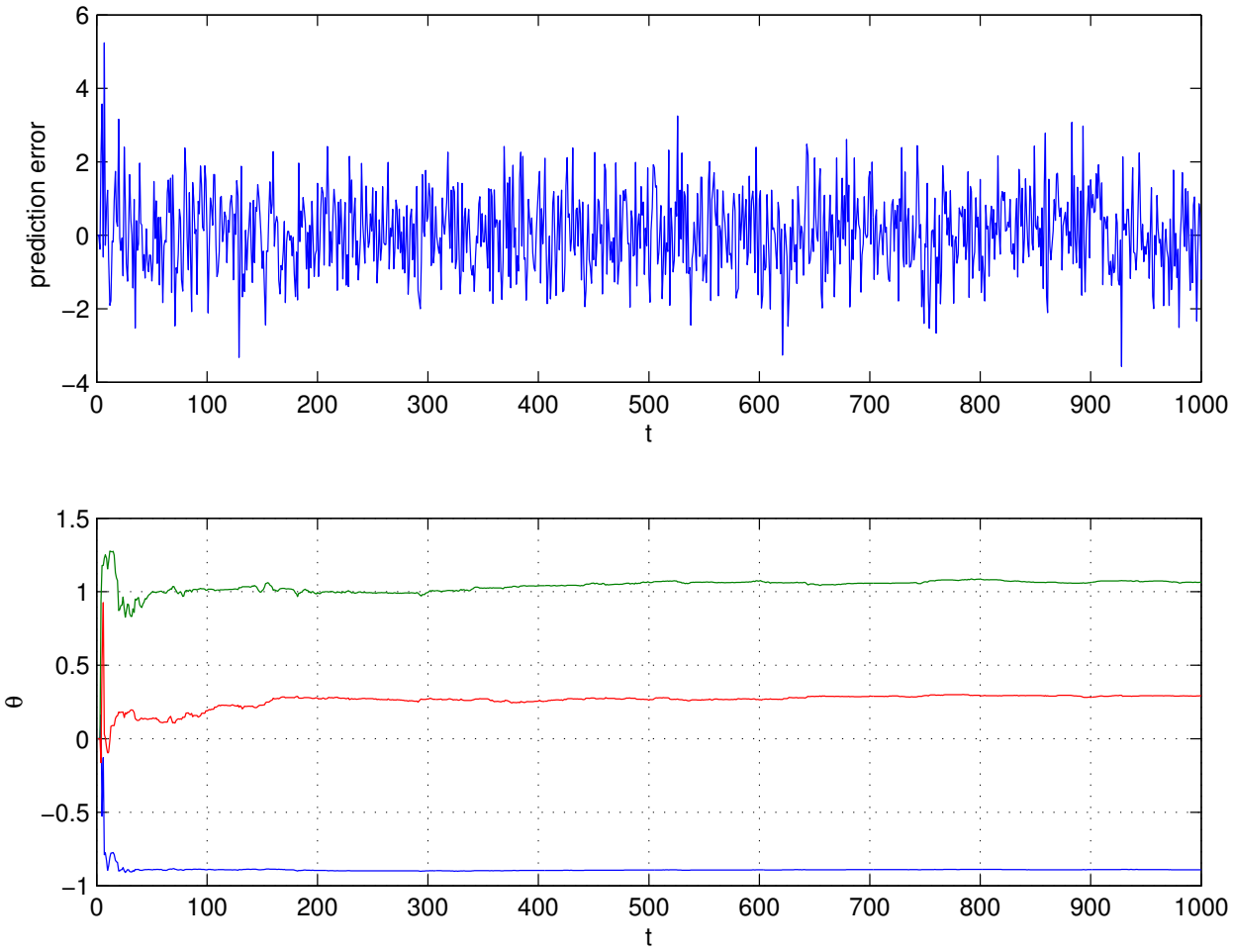


Figure 14: Results of RLS algorithm ($\lambda = 1$).

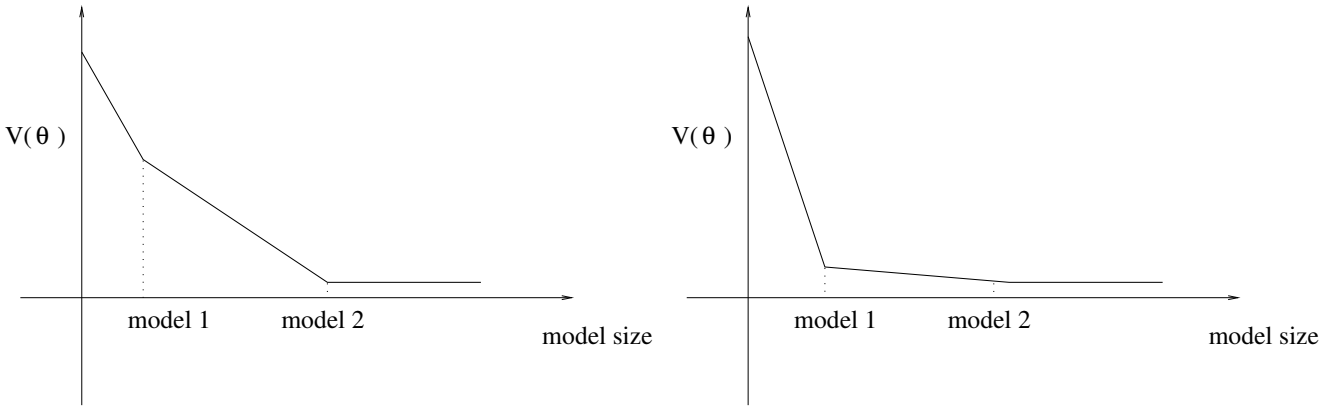


Figure 15: Left: Model 2 is preferable as Model 1 is not large enough to cover the true system. Right: Model 1 is preferable as Model 2 is more complex, but not significantly better than Model 1.

Lecture 8: The instrumental variable method

The instrumental variable method is a modification of the least squares method designed to overcome the convergence problems when the disturbance $v(t)$ is not white noise.

For linear regression model

$$y(t) = \boldsymbol{\phi}^T(t)\boldsymbol{\theta} + v(t)$$

or in vector form

$$\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{v}$$

The least squares solution

$$\hat{\boldsymbol{\theta}} = [\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{y}$$

with

$$E(\hat{\boldsymbol{\theta}}) = \boldsymbol{\theta} + E([\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{v})$$

The term $E([\boldsymbol{\Phi}^T\boldsymbol{\Phi}]^{-1}\boldsymbol{\Phi}^T\mathbf{v})$ is zero if

$$E(\boldsymbol{\Phi}^T\mathbf{v}) = 0$$

This is true in the following typical cases:

(i) $v(t)$ is white noise.

(ii) $\boldsymbol{\phi}(t)$ contains only input terms $u(t)$ which is uncorrelated $v(t)$, such that $E(\boldsymbol{\Phi}^T\mathbf{v}\ ) = 0$. (If $v(t)$ is not white noise, and $\boldsymbol{\phi}(t)$ contains $y(t-1)$, ...,$y(t-n_a)$, then $y(t-1)$ contains $v(t-1)$ which in turn is correlated with $v(t)$, then $E(\boldsymbol{\Phi}^T\mathbf{v}) \neq 0$, and $\hat{\boldsymbol{\theta}}$ is biased).

The instrumental variable method is given by

$$\hat{\boldsymbol{\theta}} = [\mathbf{Z}^T\boldsymbol{\Phi}]^{-1}\mathbf{Z}^T\mathbf{y}$$

$$= [\sum_{t=1}^{N}\mathbf{z}(t)\boldsymbol{\phi}^T(t)]^{-1}\sum_{t=1}^{N}\mathbf{z}^T(t)y(t)$$

where $\mathbf{z}(t)$ is a vector with the same dimension as $\boldsymbol{\phi}(t)$.

We see that $\hat{\boldsymbol{\theta}}$ tends to $\boldsymbol{\theta}$ as N tends to infinity under the following three conditions.

(i) $\mathbf{z}(t)$ and $v(t)$ are uncorrelated.

(ii) $\mathbf{Z}^T\boldsymbol{\Phi}$ is invertible.

(iii) $v(t)$ has zero mean.

The vector $\mathbf{z}(t)$ is referred to as the instrumental variable (IV). We now show that instrumental variable estimate is unbiased:

$$\hat{\boldsymbol{\theta}} = [\mathbf{Z}^T\boldsymbol{\Phi}]^{-1}\mathbf{Z}^T\mathbf{y}$$
$$= [\mathbf{Z}^T\boldsymbol{\Phi}]^{-1}\mathbf{Z}^T(\boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{v})$$
$$= \boldsymbol{\theta} + [\mathbf{Z}^T\boldsymbol{\Phi}]^{-1}\mathbf{Z}^T\mathbf{v}$$
$$E(\hat{\boldsymbol{\theta}}) = \boldsymbol{\theta} + E([\mathbf{Z}^T\boldsymbol{\Phi}]^{-1}\mathbf{Z}^T\mathbf{v})$$
$$= \boldsymbol{\theta} + [\mathbf{Z}^T\boldsymbol{\Phi}]^{-1}E(\mathbf{Z}^T\mathbf{v}) = \boldsymbol{\theta}$$

The recursive IV algorithm is

$$\begin{aligned}
\varepsilon(n) &= y(n) - \boldsymbol{\phi}^T(n)\hat{\boldsymbol{\theta}}(n-1) \\
\mathbf{P}(n) &= \mathbf{P}(n-1) - \frac{\mathbf{P}(n-1)\mathbf{z}(n)\phi^T(n)\mathbf{P}(n-1)}{1+\boldsymbol{\phi}^T(n)\mathbf{P}(n-1)\mathbf{z}(n)} \\
\mathbf{K}(n) &= \mathbf{P}(n)\mathbf{z}(n) \\
\hat{\boldsymbol{\theta}}(n) &= \hat{\boldsymbol{\theta}}(n-1) + \mathbf{K}(n)\varepsilon(n)
\end{aligned}$$

Choice of the instrumental variable $\mathbf{z}(t)$

Loosely speaking the choice of the instrumental variable $\mathbf{z}(t)$ is that they should be sufficiently correlated to $\phi(t)$ but uncorrelated with the system noise $v(t)$.

Suppose that the following model is used

$$\begin{aligned}
y(t) &= -a_1y(t-1) - ... - a_{n_a}y(t-n_a) \\
&\quad +b_1u(t-1) + ... + b_{n_b}u(t-n_b) + v(t) \\
&= \boldsymbol{\phi}^T(t)\boldsymbol{\theta} + v(t)
\end{aligned}$$

with

$$\boldsymbol{\phi}(t) = [-y(t-1), ... - y(t-n_a),$$
$$u(t-1), ..., u(t-n_b)]^T$$

$\boldsymbol{\theta} = [a_1, ..., a_{n_a}, b_1, ..., b_{n_b}]^T$

A common choice of $\mathbf{z}(t)$ is

$$\mathbf{z}(t) = [-y_M(t-1), ... - y_M(t-n_a),$$
$$u(t-1), ..., u(t-n_b)]^T$$

where $y_M(t-1)$ is noise free output of a system driven by the actual input $u(t)$ using $\hat{a}_i$ and $\hat{b}_i$ which are current estimates obtained in the recursive IV algorithm.

Example 8-1:

Consider a system described by

$$y(t) + ay(t-1) = bu(t-1) + ce(t-1) + e(t)$$

where $\{e(t)\}$ is a white noise sequence with variance 1. The system parameters are

$$a = -0.9, b = 1.0, c = 0.3$$

The input

$$u(t) = 0.1\sin(\frac{\pi t}{50} + \frac{\pi}{3}) + 0.2\sin(\frac{\pi t}{30})$$

1000 data samples are generated.

Choose

$$\boldsymbol{\phi}(t) = [-y(t-1), u(t-1)]^T$$
$$\mathbf{z}(t) = [-\hat{y}(t-1), u(t-1)]^T$$

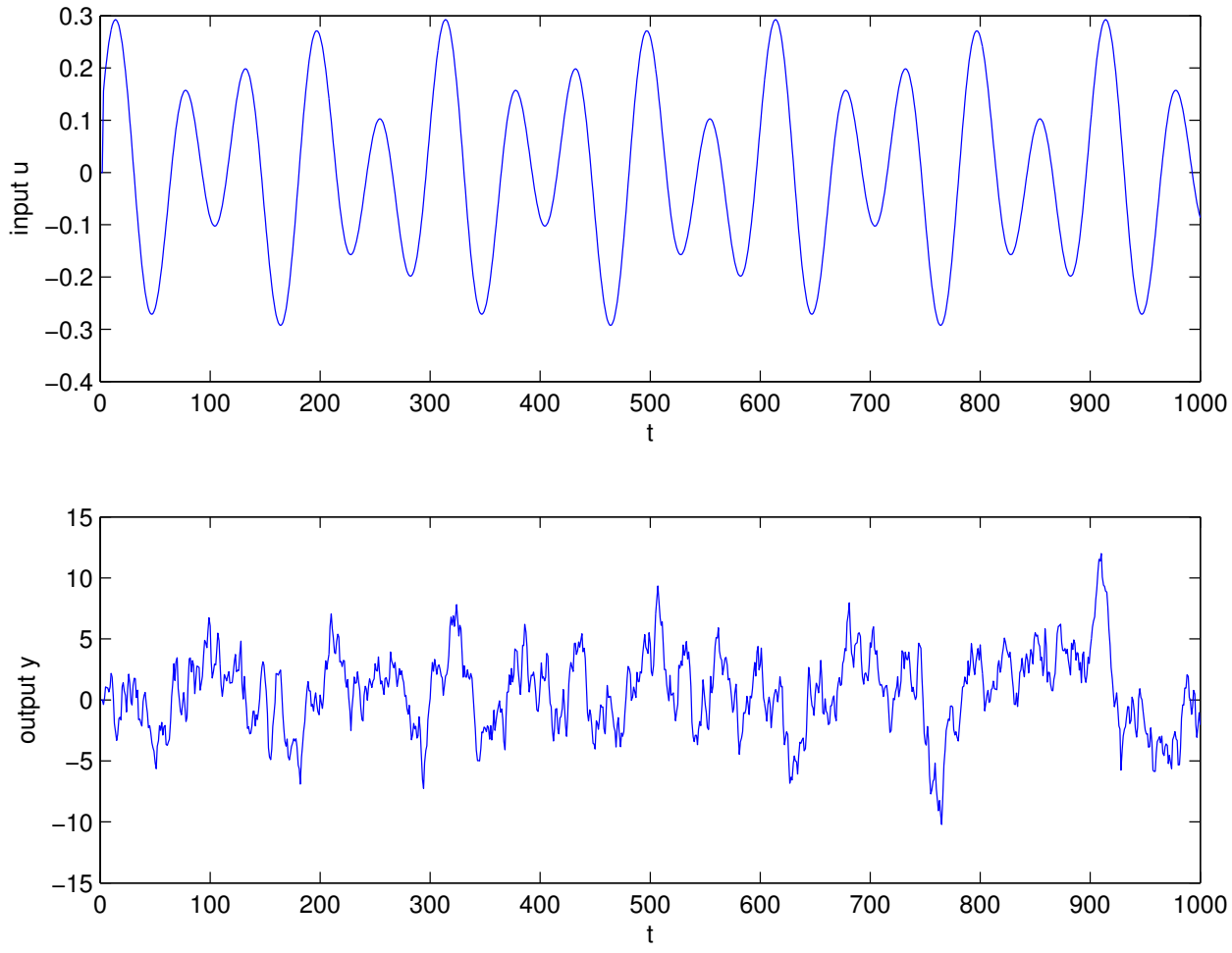


Figure 16: System input and output.8-1

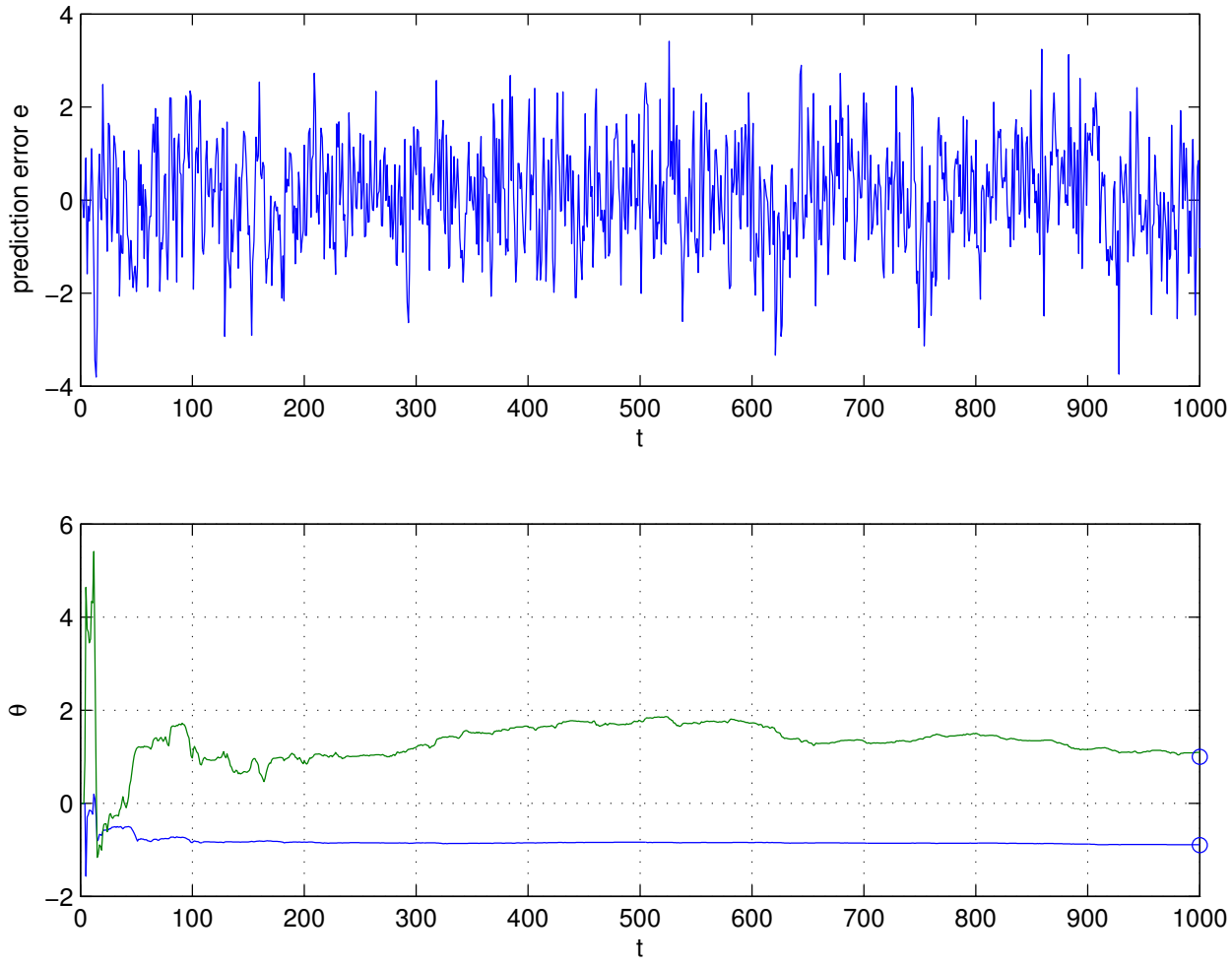


Figure 17: Results of recursive instrumental algorithm

Maximum likelihood estimation

There is a further important issue to mention, namely the relation of LS estimator with the Maximum Likelihood method in estimation theory.

Suppose that the process noise $e(t)$ is a discrete random process, and we know its probability density function as a functional of the parameter vector $\boldsymbol{\theta}$, such that we know $p(e;\boldsymbol{\theta})$.

Now we have N data samples, given just as before $\{y(t), u(t)\ \}$, how do we estimate $\boldsymbol{\theta}$.

The idea of maximum likelihood estimation is to maximize a likelihood function which is often defined as the joint probability of $e(t)$.

Suppose $e(t)$ is uncorrelated, the likelihood function L can be written as (the joint probability of $e(t)$)

$$L(\boldsymbol{\theta}) = \prod_{t=1}^{N} p(e(t), \boldsymbol{\theta})$$

This means that the Likelihood function is the product of data each samples pdf.

Consider using log likelihood function $\log L$. Log function is a monotonous function. This means when L is maximum, so is $\log L$.

Instead of looking for $\hat{\boldsymbol{\theta}}$ that maximizes L, we now look for $\hat{\boldsymbol{\theta}}$ that maximizes $\log L$, the result will be the same, but computation is simpler.

$$\frac{\partial \log L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} = \mathbf{0}$$

If $e(t)$ is Gaussian distributed with zero mean, and variance σ^2

$$p(e(t), \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{e^2(t)}{2\sigma^2}\}$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{[y(t) - \boldsymbol{\theta}^T \boldsymbol{\phi}(t)]^2}{2\sigma^2}\}$$

$$\log L(\boldsymbol{\theta}) = \log \prod_{t=1}^{N} p(e(t), \boldsymbol{\theta})$$

$$= \sum_{t=1}^{N} \log p(e(t), \boldsymbol{\theta})$$

$$= -\sum_{t=1}^{N} \frac{[y(t) - \boldsymbol{\theta}^T \boldsymbol{\phi}(t)]^2}{2\sigma^2} - \frac{N}{2} \log(2\pi\sigma^2)$$

$$= -\frac{[\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}]^T [\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}]}{2\sigma^2} + c$$

By setting

$$\frac{\partial \log L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} = \mathbf{0}$$

we obtain

$$\frac{\partial [\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}]^T [\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}]}{\partial \boldsymbol{\theta}} = \mathbf{0}$$

$$\frac{\partial [\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}]^T [\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}]}{\partial \boldsymbol{\theta}} = -2[\frac{\partial (\mathbf{y}^T \boldsymbol{\Phi}\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}]^T + \frac{\partial (\boldsymbol{\theta}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi}\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

$$= -2\boldsymbol{\Phi}^T \mathbf{y} + 2\boldsymbol{\Phi}^T \boldsymbol{\Phi}\boldsymbol{\theta}$$

$$= -2\boldsymbol{\Phi}^T (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) = \mathbf{0}$$

yielding

$$\hat{\boldsymbol{\theta}} = [\boldsymbol{\Phi}^T \boldsymbol{\Phi}]^{-1} \boldsymbol{\Phi}^T \mathbf{y}$$

which is simply the LS estimates.

The conclusion is that under the assumption that the noise is Gaussian distributes, then the LS estimates is equivalent to maximum likelihood estimates.

References

[1] Lennart Ljung et al. *Theory and practice of recursive identification*. The MIT press, 1983 (cit. on p. 1).

[2] Lennart Ljung. *System identification*. Wiley Online Library, 1999. DOI: 10.1002/047134608X.W1046.pub2 (cit. on p. 1).

[3] Lennart Ljung. *System Identification: Theory For the User*. (**UoR call 003.1.LJU**). Second Edition, Upper Saddle River, N.J: Prentice Hall, 1999 (cit. on p. 1).

[4] Torsten Söderström and Petre Stoica. *System identification*. Prentice-Hall, Inc., 1988 (cit. on p. 1).

[5] Cleve B. Moler. *Numerical Computing with MATLAB*. SIAM, 2004. DOI: 10.1137/1.9780898717952. URL: https://uk.mathworks.com/moler/index_ncm.html.

[6] Cleve Moler. *Professor SVD*. http://www.mathworks.com/tagteam/35906_91425v00_clevescorner.pdf. 2006 (cit. on p. 30).

Appendix: Matrix Algebra

Matrix quick reference guide/databook etc

Matrix and vector definitions

A matrix is a two dimensional array that contains expressions or numbers. In general matrices will be notated as bold uppercase letters, sometimes with a trailing subscript to indicate its size.

For example:

$$\mathbf{A}_{4\times 3} = \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

is a matrix with 4 rows and 3 columns containing 12 articles (the location of which is given by the article subscript).

A vector is a collection of symbols or numbers, and can be represented as a matrix with a single column. Vectors are sometimes notated as a bold lowercase symbol, or with a short line or arrow that may be below or above the symbol.

For example

$$\mathbf{b} = \boldsymbol{b} = \bar{b} = \underline{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = [b_1, b_2, b_3, b_4]^T$$

is a vector with 4 elements. Note that ^{T} denotes transpose.

Matrix operations

1. Addition:

$\mathbf{C} = \mathbf{A} + \mathbf{B}$ such that $c_{ij} = a_{ij} + b_{ij}$ ($\mathbf{A}$ and $\mathbf{B}$ must be the same size)

2. Transpose:

The transpose of a matrix interchanges rows and columns.

$\mathbf{C} = \mathbf{A}^T$ such that $c_{ij} = a_{ji}$

3. Identity matrix:

The identity matrix $\mathbf{I}$ is square and has 1s on the major diagonal, elsewhere 0s.

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{bmatrix}$$

4. Multiplication:

The definition of matrix multiplication is $\mathbf{C} = \mathbf{AB}$ such that $c_{ij} = \sum_k a_{ik} b_{kj}$

(The number of columns of $\mathbf{A}$ must be the same as number of rows of $\mathbf{B}$)

Matrices are associative so that

$$A + AB = A(I + B) \neq A + BA = (I + B)A$$

Matrices are not commutative in multiplication, i.e.

$$AB \neq BA$$

Example: let

$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$

Then $\mathbf{C} = \mathbf{AB} = \mathbf{A}_{4\times3}\mathbf{B}_{3\times2}$

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \\ c_{41} & c_{42} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} \\ a_{41}b_{11} + a_{42}b_{21} + a_{43}b_{31} & a_{41}b_{12} + a_{42}b_{22} + a_{43}b_{32} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

Subscript notation

To work out if two matrices will multiply it is useful to subscript the matrix variable with row and column information, thus

This can be determined by considering the dimensions of each matrix. For example if we write $A_{2\times3}$ to indicate that it has 2 rows and 3 columns, then

$$A_{2\times3}B_{3\times5} = C_{2\times5}$$

is realised by cancelling the inner 3's resulting in a 2 by 5 matrix.

5. Multiplication of partitioned matrices

Any matrix multiplication can also be carried out on sub-matrices and vectors as long as each sub-matrix/vector is a valid matrix calculation. (You can colour in the individual articles in the equation above to demonstrate.)

For example if $A = \left[\begin{array}{c|c} \mathbf{D} & \underline{f} \end{array}\right]$ and $B = \left[\begin{array}{c} \mathbf{G} \\ \hline \underline{h}^T \end{array}\right]$ then

$$AB = \begin{bmatrix} \mathbf{DG} \\ \underline{f}\underline{h}^T \end{bmatrix}$$

6. Matrix inverse:

If a matrix is square and not singular it has an inverse such that

$$AA^{-1} = A^{-1}A = I$$

Where I is the identity matrix.

$\mathbf{AA}^{-1} = \mathbf{I}$

$\mathbf{A}^{-1}$ exists only if $\mathbf{A}$ is square and not singular. $\mathbf{A}$ is singular if the determinant $|\mathbf{A}| = 0$.

A simple example is

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$|A| = ad - bc$$

$$\mathbf{A}^{-1} = \frac{\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}}{|\mathbf{A}|} = \frac{\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}}{ad - bc}$$

Computation of the inverse can be done by computing the matrix determinant of $|A|$, the matrix of minors, the co-factor, and the ajoint/adjugate. Thus if X is the ajoint/ajugate of A

$$XA = AX = I|A|$$

Computing the determinant and the ajoint requires computing a matrix of minors and hence the co-factor matrix. The co-factor matrix is then simply the transpose of the ajoint/ajugate.

Note on numerical computation

Some matrices are do not have independent rows so do not have an inverse. This can be tested by computing a matrix rank.

Some matrices are ill-conditioned so that the inverse is difficult to compute. A measure called the condition number, defined as the ratio of the maximum singular to the minimum singular value, indicate whether the matrix inverse will be sensitive with respect to the accuracy of the computer.

In Matlab the condition number can be calculated as

```
>> max(svd(A))/min(svd(A))
```

or via the cond command

Note on Cayley-Hamilton Theorem

The Cayley-Hamilton Theorem says that 'Every square matrix satisfies its own characteristic equation.'

The Characteristic equation is the solution to

$$|sI - A| = 0$$

This gives a way to calculating the inverse of A by substituting into the characteristic equation and multiplying by A^{-1}.

Algebra rules :

$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$ (Addition is commutative)

$\mathbf{AB} \neq \mathbf{BA}$ (Multiplication not commutative)

$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$ (Associative)

$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$ (Associative)

$\mathbf{AI} = \mathbf{AI} = \mathbf{A}$

$\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ (for non-singular and square matrix)

$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$

$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$

Matrix decomposition

Matrix types

$\mathbf{C} = \mathbf{C}^T$ such that $c_{ij} = c_{ji}$ (If $\mathbf{C}$ is symmetric it must be square)
A matrix is symmetric if $B = B^T$
A matrix is skew symmetric if $B = -B^T$
A matrix is orthogonal if $B^{-1} = B^T$
A matrix is positive definite if $x^T Bx$ is positive for all values of the vector x
The scalar $(Ax)^T(Ax)$ will always be zero or positive. Thus if the matrix B can be partitioned such that $B = A^T A$ it will be positive definite.

Orthogonal matrices

If a vector is transformed by an orthogonal matrix then Euclidean metrics are conserved. That is if $L(x) = x^T x$ and $y = Ax$ then evidently $L(y) = y^T y = (Ax)^T Ax = x^T A^T Ax$

If the measures are conserved then $L(x) = L(y)$ so $A^T A = I$ and hence orthogonality requires $A^T = A^{-1}$

Eigenvalues and Eigenvectors

$$|\lambda I - A| = 0$$

- If A is real then complex Eigenvalues, if they occur, will be as complex conjugate pairs.
- A symmetric matrix has only real Eigenvalues and the Eigenvectors are orthogonal.

Given the Eigenvalues, the Eigenvectors are such that

$$AV = V \mathrm{diag} \lambda_i$$

Where the columns of V are the Eigenvectors and diag lambda is a diagonal matrix of the eigenvalues.

If A is symmetric and $[V, D]$ are the Eigenvectors and diagonalised Eigenvalues such that $AV = VD$. Then since $V^{-1} = V^T$, $A^{-1} = V^T D^{-1} V$ and since D is diagonal the inverse is now trivial...

Rank vs Eigenvalues and Eigenvectors

Singular value decomposition

For a matrix A we can partition A such that $A = UDV^T$ where $U^T U = I$, $V^T V = I$ and D is diagonal. The diagonal elements of D are known as singular values σ

$$\sigma_i = \sqrt{\lambda_i (A^T A)}$$

where $\lambda_i(A^T A)$ is are the Eigenvalues of $A^T A$

See [6] for a good overview.

Transfer functions

Interesting identity

$$A(I + A)^{-1} = (I + A)^{-1} A$$

Things to do with 2x2 matrices

Let

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

determinate $|A| = ad - bc$

inverse

$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \begin{bmatrix} \frac{d}{ad-bc} & -\frac{b}{ad-bc} \\ -\frac{c}{ad-bc} & \frac{a}{ad-bc} \end{bmatrix}$$

Eigenvalues

$$\lambda = \begin{bmatrix} 1/2\,a + 1/2\,d + 1/2\,\sqrt{a^2 - 2\,ad + d^2 + 4\,bc} \\ 1/2\,a + 1/2\,d - 1/2\,\sqrt{a^2 - 2\,ad + d^2 + 4\,bc} \end{bmatrix}$$

Useful Matlab functions

A*B A+B A\B A/B	matrix multiplication, addition and division (left and right)
eye(n)	create an identity matrix size n
svg(A)	compute the singular value decomposition matrices of A
eig(A)	compute the Eigenvalues and Eigenvectors of A
rank(A)	compute rank (number of independent rows or columns in A)
cond(A)	compute condition number (an indication of numerical error in the inverse)
det(A)	compute determinant
inv(A)	compute inverse
pinv(A)	compute pseudo inverse = $(AA^T)^{-1}A^T$
adjoint(A)	compute adjoint/adjugate
eigshow	demonstration of Eigenvalues, (can drag the x vector)

References

[1] Lennart Ljung et al. *Theory and practice of recursive identification*. The MIT press, 1983 (cit. on p. 1).

[2] Lennart Ljung. *System identification*. Wiley Online Library, 1999. DOI: 10.1002/047134608X.W1046.pub2 (cit. on p. 1).

[3] Lennart Ljung. *System Identification: Theory For the User*. (**UoR call 003.1.LJU**). Second Edition, Upper Saddle River, N.J: Prentice Hall, 1999 (cit. on p. 1).

[4] Torsten Söderström and Petre Stoica. *System identification*. Prentice-Hall, Inc., 1988 (cit. on p. 1).

[5] Cleve B. Moler. *Numerical Computing with MATLAB*. SIAM, 2004. DOI: 10.1137/1.9780898717952. URL: https://uk.mathworks.com/moler/index_ncm.html.

[6] Cleve Moler. *Professor SVD*. http://www.mathworks.com/tagteam/35906_91425v00_clevescorner.pdf. 2006 (cit. on p. 30).

William Harwin Oct 2017

Acronyms

ARIMA	Autoregressive Integrated Moving Average
ARMAX	Autoregressive Moving Average eXogeneous
ARX	Autoregressive eXogeneous
CARMA	Controlled ARMA
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
RLS	Recursive Least Squares

Programs

Least squares example

```
%% System ID example 1
u=[1  2  4  5 7]';
y=[1  3  5  7 8]';
Phi=[u ones(size(u))];
theta=Phi\y
tu=[0 8]';
% figure(1);plot(tu,[tu ones(2,1)]*theta,u,y,'o')
% grid;xlabel('u(t)');ylabel('y(t)')
% text(u,0.5*ones(1,5),{'u(1)','u(2)','u(3)','u(4)','u(5)'});

%%
Phi=[u.^2 u ones(size(u))];
theta2=Phi\y
tu2=(0:.1:8)';
figure(1);plot(tu,[tu ones(2,1)]*theta,tu2,[tu2.^2 tu2 ones(size(tu2))]*theta2,u,y,'o')
grid;xlabel('u(t)');ylabel('y(t)')
text(u+.05,-1.6*ones(1,5),{'u(1)','u(2)','u(3)','u(4)','u(5)'})
text(.05*ones(1,5),y+.05,-1.6*ones(1,5),{'y(1)','y(2)','y(3)','y(4)','y(5)'})
legend({'Example 1','Example 2'},'Location','best')
%
```

Positive definate example

```
%% Examples of positive definate matrices
if exist('k','var')==0;k=8;end
if k >7;
   k=menu('Choose quadratic form','random','rand pd','pos def','saddle','identity','set to m');
end

if k==1;M=randn(2);end

% random pd
if k==2;M=randn(2);M=M*M';end

% Illustrative pos def
if k==3;M=[.32  -.11;-.11 .52];end
% if k==4;M=[1.08 0.229;2.37 -.267];end
% illustrative saddle
if k==4;M=[-.072 1.373;.28 .18];end
if k==5;M=eye(2);end
if k==6;M=m;end
if k==7;M=1;end

z=cplxgrid(20);
X=real(z);
Y=imag(z);
[n,m]=size(X);
f=zeros(n,m);
for i=1:n
   for j=1:m;
      x=[X(i,j);Y(i,j)];
      f(i,j)=x'*M*x;
   end
end
surfc(X,Y,f);
%meshc(X,Y,f);
if k>2; clear; end
```

Basic ARMAX example

```
%% Basic examples of ARMAX
% approx page 7 of notes

% System 1
u=2-4*rand(1,1000); % zero mean uniform distribution
e=randn(1,1000);
y(1)=0;y2(1)=0;
for  i=2:1000;
  y(i)=0.8*y(i-1) + u(i-1) +  e(i); % system 1
  y2(i)=0.8*y2(i-1) + u(i-1) - 0.8*e(i-1) + e(i); % system 2
end;
sr=100:200;% subrange of the data

%figure(1); plot(sr,u(sr));xlabel('t');ylabel('u(t)');
%figure(2); plot(sr,y(sr));xlabel('t');ylabel('y(t)');
figure(1);subplot(211); plot(sr,u(sr));xlabel('t');ylabel('u(t)');
subplot(212);plot(sr,y(sr));xlabel('t');ylabel('y(t)');

% Matlab code for least squares estimator:
```

```
% phi2=[[0 y(1:end-1)]' [0 0 y(1:end-2)]' [0 u(1:end-1)]'];

for i=2: 1000;
  phi(i,:)=[y(i-1) u(i-1)];
end;
theta=(phi'*phi)\phi'*y';
```

```
% Model output
yhat=phi*theta;
figure(3);
plot(sr,y(sr),'-',sr,yhat(sr),'-.');
legend({'System output','Model output'},'Location','Best')
xlabel('t');
```