System Identification

Control Engineering EN, 3rd year B.Sc.
Technical University of Cluj-Napoca
Romania

Lecturer: Lucian Buşoniu

Part V

ARX identification

Table of contents

We stay in the single-output, single-input case for all but the last section.

Classification

Recall **Types of models** from Part I:

1. Mental or verbal models
2. Graphs and tables (nonparametric)
3. Mathematical models, with two subtypes:
   - First-principles, analytical models
   - Models from system identification

The ARX method produces *parametric*, polynomial models.

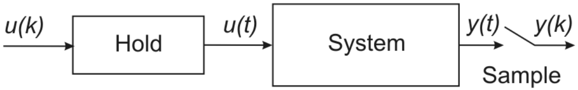
Why ARX?

- General-order, fully implementable method with guarantees – like correlation analysis
- Unlike correlation analysis, gives a *compact* model with a number of parameters proportional to the order of the system

Table of contents

Recall: Discrete time

We remain in the discrete-time setting:



ARX model structure

In the ARX model structure, the output $y(k)$ at the current discrete time step is computed based on previous input and output values:

$$\begin{aligned} y(k) &+ a_1 y(k-1) + a_2 y(k-2) + \ldots + a_{na} y(k-na) \\ &= b_1 u(k-1) + b_2 u(k-2) + \ldots + b_{nb} u(k-nb) + e(k) \end{aligned}$$

equivalent to

$$\begin{aligned} y(k) &= -a_1 y(k-1) - a_2 y(k-2) - \ldots - a_{na} y(k-na) \\ &\quad b_1 u(k-1) + b_2 u(k-2) + \ldots + b_{nb} u(k-nb) + e(k) \end{aligned}$$

$e(k)$ is the noise at step k.

Model parameters: $a_1, a_2, \ldots, a_{n_a}$ and $b_1, b_2, \ldots, b_{nb}$.

Name: AutoRegressive ($y(k)$ a function of previous y values) with eXogenous input (dependence on u)

Polynomial representation

Backward shift operator q^{-1}:

$$q^{-1}z(k) = z(k-1)$$

where $z(k)$ is any discrete-time signal.

Then:

$$\begin{aligned} & y(k) + a_1 y(k-1) + a_2 y(k-2) + \ldots + a_{na} y(k-na) \\ & \quad = (1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_{na} q^{-na}) y(k) =: A(q^{-1}) y(k) \end{aligned}$$

and:

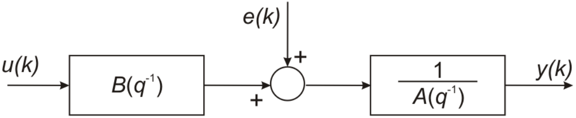
$$\begin{aligned} & b_1 u(k-1) + b_2 u(k-2) + \ldots + b_{nb} u(k-nb) \\ & \quad = (b_1 q^{-1} + b_2 q^{-2} + \ldots + b_{nb} q^{-nb}) u(k) =: B(q^{-1}) u(k) \end{aligned}$$

ARX model in polynomial form

Therefore, the ARX model is written compactly:

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

Or in graphical representation:



which holds because:

$$y(k) = \frac{1}{A(q^{-1})}[B(q^{-1})u(k) + e(k)]$$

Remark: The ARX model is quite general, it can describe arbitrary linear relationships between inputs and outputs. However, the noise enters the model in a restricted way, and later we introduce models that generalize this.

Linear regression model

Returning to the explicit recursive representation:

$$
\begin{aligned}
y(k) &= -a_1 y(k-1) - a_2 y(k-2) - \ldots - a_{na} y(k-na) \\
&\quad b_1 u(k-1) + b_2 u(k-2) + \ldots + b_{nb} u(k-nb) + e(k) \\
&= \begin{bmatrix} -y(k-1) & \cdots & -y(k-na) & u(k-1) & \cdots & u(k-nb) \end{bmatrix} \\
&\quad \cdot \begin{bmatrix} a_1 & \cdots & a_{na} & b_1 & \cdots & b_{nb} \end{bmatrix}^\top + e(k) \\
&=: \varphi^\top(k)\theta + e(k)
\end{aligned}
$$

So in fact ARX obeys the standard model structure in linear regression!

Regressor vector: $\varphi \in \mathbb{R}^{na+nb}$, previous output and input values.
Parameter vector: $\theta \in \mathbb{R}^{na+nb}$, polynomial coefficients.

Identification problem

Consider now that we are given a vector of data $u(k), y(k)$, $k = 1, \dots, N$, and we have to find the model parameters θ.

Then for any k:

$$y(k) = \varphi^\top(k)\theta + \varepsilon(k)$$

where $\varepsilon(k)$ is now interpreted as an equation error (hence the changed notation).

Objective: minimize the mean squared error:

$$V(\theta) = \frac{1}{N}\sum_{k=1}^{N} \varepsilon(k)^2$$

Remark: When $k \leq na, nb$, zero- and negative-time values for u and y are needed to construct φ. They can be taken equal to 0 (assuming the system is in zero initial conditions).

Linear system of equations

$$y(1) = \begin{bmatrix} -y(0) & \cdots & -y(1-na) & u(0) & \cdots & u(1-nb) \end{bmatrix} \theta$$
$$y(2) = \begin{bmatrix} -y(1) & \cdots & -y(2-na) & u(1) & \cdots & u(2-nb) \end{bmatrix} \theta$$
$$\cdots$$
$$y(N) = \begin{bmatrix} -y(N-1) & \cdots & -y(N-na) & u(N-1) & \cdots & u(N-nb) \end{bmatrix} \theta$$

Matrix form:

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} -y(0) & \cdots & -y(1-na) & u(0) & \cdots & u(1-nb) \\ -y(1) & \cdots & -y(2-na) & u(1) & \cdots & u(2-nb) \\ \vdots & \vdots & \vdots & \vdots & & \\ -y(N-1) & \cdots & -y(N-na) & u(N-1) & \cdots & u(N-nb) \end{bmatrix} \cdot \theta$$
$$Y = \Phi\theta$$

with notations $Y \in \mathbb{R}^N$ and $\Phi \in \mathbb{R}^{N \times (na+nb)}$.

ARX solution

From linear regression, to minimize $\frac{1}{2}\sum_{k=1}^{N}\varepsilon(k)^2$ the parameters are:

$$\widehat{\theta} = (\Phi^\top \Phi)^{-1}\Phi^\top Y$$

Since the new $V(\theta) = \frac{1}{N}\sum_{k=1}^{N}\varepsilon(k)^2$ is proportional to the criterion above, the same solution also minimizes $V(\theta)$.

However, the form above is impractical in system identification, since the number of data points N can be very large. Better form:

$$\Phi^\top \Phi = \sum_{k=1}^{N}\varphi(k)\varphi^\top(k), \quad \Phi^\top Y = \sum_{k=1}^{N}\varphi(k)y(k)$$

$$\Rightarrow \widehat{\theta} = \left[\sum_{k=1}^{N}\varphi(k)\varphi^\top(k)\right]^{-1}\left[\sum_{k=1}^{N}\varphi(k)y(k)\right]$$

(Recall the similar “trick” from linear regression.)

ARX solution (continued)

Remaining issue: the sum of N terms can grow very large, leading to numerical problems: (matrix of very large numbers)$^{-1}\cdot$ vector of very large numbers.

Solution: Normalize element values by diving them by N. In equations, N simplifies so it has no effect on the analytical development, but in practice it keeps the numbers reasonable.

$$\widehat{\theta} = \left[\frac{1}{N}\sum_{k=1}^{N}\varphi(k)\varphi^\top(k)\right]^{-1}\left[\frac{1}{N}\sum_{k=1}^{N}\varphi(k)y(k)\right]$$

Using the ARX model

One-step ahead prediction: The true output sequence is known, so all the delayed signals are available and we can simply plug them in the formula, together with the coefficients taken from θ:

$$\begin{aligned}\hat{y}(k) = & - a_1 y(k-1) - a_2 y(k-2) - \ldots - a_{na} y(k-na) \\ & b_1 u(k-1) + b_2 u(k-2) + \ldots + b_{nb} u(k-nb)\end{aligned}$$

Signals at negative and zero time can be taken equal to 0.

Example: On day $k-1$, predict weather for day k.

Simulation: True outputs unknown, so we must use previously simulated outputs. Each $y(k-i)$ is replaced by $\hat{y}(k-i)$:

$$\begin{aligned}\hat{y}(k) = & - a_1 \hat{y}(k-1) - a_2 \hat{y}(k-2) - \ldots - a_{na} \hat{y}(k-na) \\ & b_1 u(k-1) + b_2 u(k-2) + \ldots + b_{nb} u(k-nb)\end{aligned}$$

(predicted outputs at negative and zero time can also be taken 0.)

Example: Simulation of an aircraft's response to emergency pilot inputs, that may be dangerous to apply to the real system.

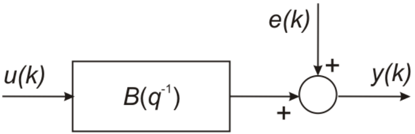
Special case of ARX: FIR

Setting $A = 1$ ($na = 0$) in ARX, we get:

$$y(k) = B(q^{-1})u(k) + e(k) = \sum_{j=1}^{nb} b_j u(k-j) + e(k)$$
$$= \sum_{j=0}^{M-1} h(j)u(k-j) + e(k)$$

the FIR model from correlation analysis!

To see this, take $nb = M - 1$, and $b_j = h(j)$. Note $h(0)$,the impulse response at time 0, is assumed 0 – i.e. system does not respond instantaneously to changes in input.



Fundamental difference between ARX and FIR

$$\begin{aligned} \text{ARX:} \quad & A(q^{-1})y(k) = B(q^{-1})u(k) + e(k) \\ \text{FIR:} \quad & y(k) = B(q^{-1})u(k) + e(k) \end{aligned}$$

Since ARX includes recursive relationships between current and previous outputs, it will be sufficient to take orders *na* and *nb* equal to the order of the dynamical system.

FIR needs a sufficiently large order *nb* (or length M) to model the entire transient regime of the impulse response (in principle, we only recover the correct model as $M \to \infty$).

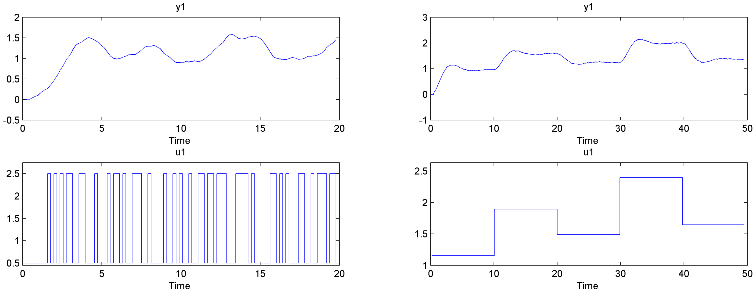
$\Rightarrow$ more parameters $\Rightarrow$ more data needed to identify them.

Table of contents

Experimental data

Consider we are given the following, separate, identification and validation data sets.

`plot(id);` and `plot(val);`



Remarks: Identification input: a so-called *pseudo-random binary signal*. Validation input: a sequence of steps.

Identifying an ARX model

```
model = arx(id, [na, nb, nk]);
```

Arguments:

1. Identification data.
2. Array containing the orders of A and B and the *delay nk*.

Structure different from theory: includes explicitly a minimum delay *nk* between inputs and outputs, useful for systems with time delays.

$$
\begin{aligned}
y(k) + a_1 y(k-1) + a_2 y(k-2) + \ldots + a_{na} y(k-na) \\
= b_1 u(k-nk) + b_2 u(k-nk-1) + \ldots + b_{nb} u(k-nk-nb+1) + e(k)
\end{aligned}
$$

$$A(q^{-1})y(k) = B(q^{-1})u(k-nk) + e(k), \text{ where:}$$

$$A(q^{-1}) = (1 + a_1 q^{-1} + a_2 q^{-2} + \ldots + a_{na} q^{-na})$$

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

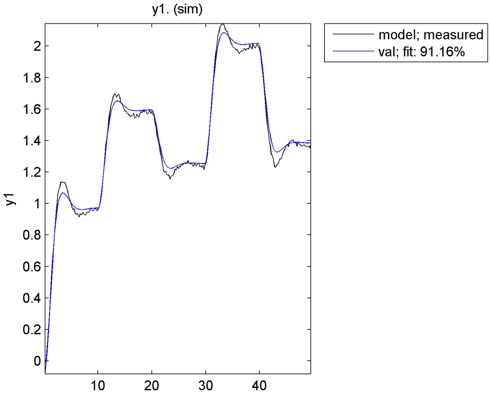
The theoretical structure is obtained by setting $nk = 1$. For $nk > 1$, we can also transform the new structure into the theoretical one by using a B polynomial of order $nk + nb - 1$, with $nk - 1$ leading zeros:

$$B_{\text{theor}}(q^{-1}) = 0q^{-1} + \ldots 0q^{-nk+1} + b_1 q^{-nk} + \ldots + b_{nb} q^{-nk-nb+1}$$

Model validation

Assuming the system is second-order, *in the ARX form*, and without time delay, we take $na = 2, nb = 2, nk = 1$. Validation:

```
compare(model, val);
```



Results are quite bad.

Structure selection

Alternate idea: try many different structures and choose the best one.

```
Na = 1:15;
Nb = 1:15;
Nk = 1:5;
NN = struc(Na, Nb, Nk);
V = arxstruc(id, val, NN);
```

- `struc` generates all combinations of orders in `Na, Nb, Nk`.
- `arxstruc` identifies for each combination an ARX model (on the data in 1st argument), simulates it (on the data in the 2nd argument), and returns all the MSEs on the first row of `V` (see `help arxstruc` for the format of `V`).

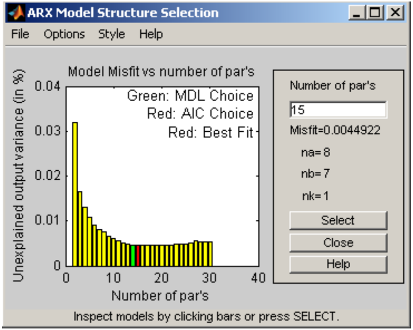
Structure selection (continued)

To choose the structure with the smallest MSE:

```
N = selstruc(V, 0);
```

For our data, `N`$=[8, 7, 1]$.

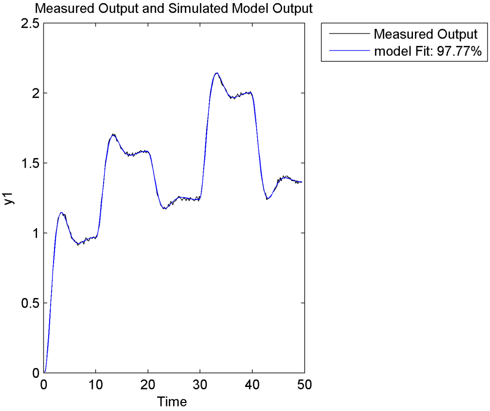
Alternatively, graphical selection: `N = selstruc(V, 'plot');`
Then click on bar corresponding to best (red) model and "Select", "Close".



(Later we learn other structure selection criteria than smallest MSE.)

Validation of best ARX model

```
model = arx(id, N); compare(model, val);
```



A better fit is obtained. However, 8th order systems are rare in real life, so something else is likely going on... we will see later.

Table of contents

Main result

Assumptions

1. There exists a true parameter vector θ_0 so that:

$$y(k) = \varphi(k)\theta_0 + v(k)$$

   with $v(k)$ a stationary stochastic process independent from $u(k)$.
2. $\mathrm{E}\left\{\varphi(k)\varphi^\top(k)\right\}$ is a nonsingular matrix.
3. $\mathrm{E}\left\{\varphi(k)v(k)\right\} = 0$.

Theorem

ARX identification is consistent: the estimated parameters $\widehat{\theta}$ converge to the true parameters θ_0, in the limit as $N \to \infty$.

Discussion of assumptions

1. Assumption 1 is equivalent to the existence of true polynomials $A_0(q^{-1})$, $B_0(q^{-1})$ so that:

$$A_0(q^{-1})y(k) = B_0(q^{-1})u(k) + v(k)$$

To motivate Assumption 2, recall

$$\widehat{\theta} = \left[\frac{1}{N}\sum_{k=1}^{N}\varphi(k)\varphi^\top(k)\right]^{-1}\left[\frac{1}{N}\sum_{k=1}^{N}\varphi(k)y(k)\right]$$

As $N \to \infty$, $\frac{1}{N}\sum_{k=1}^{N}\varphi(k)\varphi^\top(k) \to \mathrm{E}\left\{\varphi(k)\varphi^\top(k)\right\}$.

2. $\mathrm{E}\left\{\varphi(k)\varphi^\top(k)\right\}$ is nonsingular if the data is “sufficiently informative” (e.g., $u(k)$ should not be a simple feedback from $y(k)$; see Söderström & Stoica for more discussion).
3. $\mathrm{E}\left\{\varphi(k)v(k)\right\} = 0$ e.g. if $v(k)$ is white noise. Later on, we will discuss in more detail Assumption 3 and the role of $\mathrm{E}\left\{\varphi(k)v(k)\right\} = 0$.

Table of contents

MIMO system

So far we considered $y(k) \in \mathbb{R}$, $u(k) \in \mathbb{R}$,
Single-Input, Single-Output (SISO) systems

Many systems are Multiple-Input, Multiple-Output (MIMO).
E.g., aircraft. Inputs: throttle, aileron, elevator, rudder.
Outputs: airspeed, roll, pitch, yaw.

MIMO ARX

Consider next $y(k), e(k) \in \mathbb{R}^{ny}$, $u(k) \in \mathbb{R}^{nu}$. MIMO ARX model:

$$\begin{aligned} A(q^{-1})y(k) &= B(q^{-1})u(k) + e(k) \\ A(q^{-1}) &= I + A_1 q^{-1} + \ldots + A_{na} q^{-na} \\ B(q^{-1}) &= B_1 q^{-1} + \ldots + B_{nb} q^{-nb} \end{aligned}$$

where I is the $ny \times ny$ identity matrix, $A_1, \ldots, A_{na} \in \mathbb{R}^{ny \times ny}$, $B_1, \ldots, B_{nb} \in \mathbb{R}^{ny \times nu}$.

Concrete example

Take $na = 1$, $nb = 2$, $ny = 2$, $nu = 3$. Then:

$$
\begin{aligned}
A(q^{-1})y(k) &= B(q^{-1})u(k) + e(k) \\
A(q^{-1}) &= I + A_1 q^{-1} \\
&= I + \begin{bmatrix} a_1^{11} & a_1^{12} \\ a_1^{21} & a_1^{22} \end{bmatrix} q^{-1} \\
B(q^{-1}) &= B_1 q^{-1} + B_2 q^{-2} \\
&= \begin{bmatrix} b_1^{11} & b_1^{12} & b_1^{13} \\ b_1^{21} & b_1^{22} & b_1^{23} \end{bmatrix} q^{-1} + \begin{bmatrix} b_2^{11} & b_2^{12} & b_2^{13} \\ b_2^{21} & b_2^{22} & b_2^{23} \end{bmatrix} q^{-2}
\end{aligned}
$$

Concrete example (continued)

$$\left(\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}a_1^{11} & a_1^{12}\\ a_1^{21} & a_1^{22}\end{bmatrix} q^{-1}\right)\begin{bmatrix}y_1(k)\\ y_2(k)\end{bmatrix}$$
$$= \left(\begin{bmatrix}b_1^{11} & b_1^{12} & b_1^{13}\\ b_1^{21} & b_1^{22} & b_1^{23}\end{bmatrix} q^{-1} + \begin{bmatrix}b_2^{11} & b_2^{12} & b_2^{13}\\ b_2^{21} & b_2^{22} & b_2^{23}\end{bmatrix} q^{-2}\right)\begin{bmatrix}u_1(k)\\ u_2(k)\\ u_3(k)\end{bmatrix} + \begin{bmatrix}e_1(k)\\ e_2(k)\end{bmatrix}$$

Explicit relationship:

$$\begin{aligned}
y_1(k) &+ a_1^{11} y_1(k-1) + a_1^{12} y_2(k-1)\\
&= b_1^{11} u_1(k-1) + b_1^{12} u_2(k-1) + b_1^{13} u_3(k-1)\\
&+ b_2^{11} u_1(k-2) + b_2^{12} u_2(k-2) + b_2^{13} u_3(k-2) + e_1(k)\\
y_2(k) &+ a_1^{21} y_1(k-1) + a_1^{22} y_2(k-1)\\
&= b_1^{21} u_1(k-1) + b_1^{22} u_2(k-1) + b_1^{23} u_3(k-1)\\
&+ b_2^{21} u_1(k-2) + b_2^{22} u_2(k-2) + b_2^{23} u_3(k-2) + e_2(k)
\end{aligned}$$

Matlab example

Consider a continuous stirred-tank reactor:

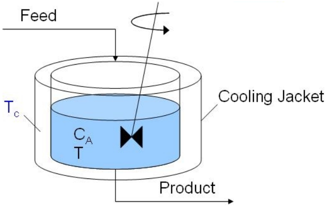


Image credit: mathworks.com

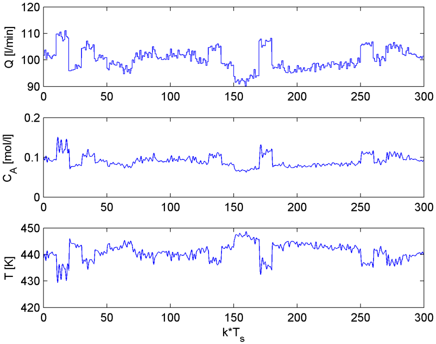
Input: coolant flow Q

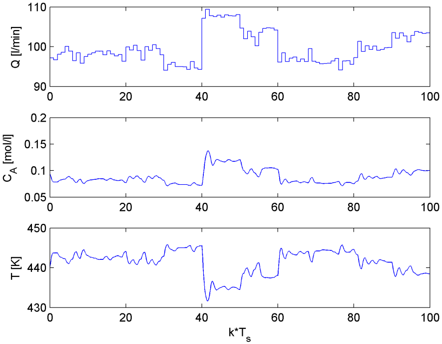
Outputs:

- Concentration C_A of substance A in the mix
- Temperature T of the mix

Matlab: Experimental data

Left: identification, Right: validation





Matlab: MIMO ARX, different from theory

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

$$A(q^{-1}) = \begin{bmatrix} a^{11}(q^{-1}) & a^{12}(q^{-1}) & \dots & a^{1ny}(q^{-1}) \\ a^{21}(q^{-1}) & a^{22}(q^{-1}) & \dots & a^{2ny}(q^{-1}) \\ \dots & & & \\ a^{ny1}(q^{-1}) & a^{ny2}(q^{-1}) & \dots & a^{nyny}(q^{-1}) \end{bmatrix}$$

$$a^{ij}(q^{-1}) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} + a_1^{ij} q^{-1} + \ldots + a_{na_{ij}}^{ij} q^{-na_{ij}}$$

$$B = \begin{bmatrix} b^{11}(q^{-1}) & b^{12}(q^{-1}) & \dots & b^{1nu}(q^{-1}) \\ b^{21}(q^{-1}) & b^{22}(q^{-1}) & \dots & b^{2nu}(q^{-1}) \\ \dots & & & \\ b^{ny1}(q^{-1}) & b^{ny2}(q^{-1}) & \dots & b^{nynu}(q^{-1}) \end{bmatrix}$$

$$b^{ij}(q^{-1}) = b_1^{ij} q^{-nk_{ij}} + \ldots + b_{nb_{ij}}^{ij} q^{-nk_{ij} - nb_{ij} + 1}$$

Matlab: Identifying the model

```
m = arx(id, [Na, Nb, Nk]);
```

Arguments:

1. Identification data.
2. Matrices with orders of polynomials in A, B, and *delays* nk:

$$Na = \begin{bmatrix} na_{11} & \dots & na_{1ny} \\ \dots & & \\ na_{ny1} & \dots & na_{nyny} \end{bmatrix}$$

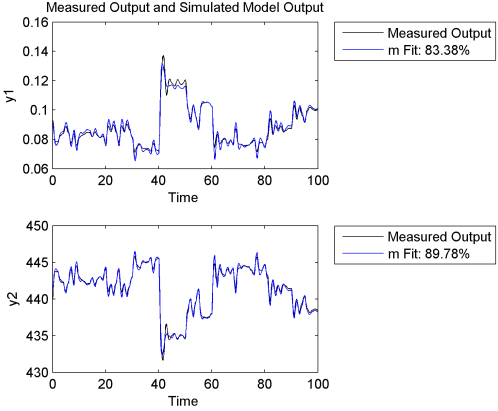
$$Nb = \begin{bmatrix} nb_{11} & \dots & nb_{1nu} \\ \dots & & \\ nb_{ny1} & \dots & nb_{nynu} \end{bmatrix}$$

$$Nk = \begin{bmatrix} nk_{11} & \dots & nk_{1nu} \\ \dots & & \\ nk_{ny1} & \dots & nk_{nynu} \end{bmatrix}$$

Matlab: Results

Take $na = 2$, $nb = 2$, and $nk = 1$ everywhere in matrix elements:

```
Na = [2 2; 2 2]; Nb = [2; 2]; Nk = [1; 1];
m = arx(id, [Na Nb Nk]);
compare(m, val);
```



Appendix: Nonlinear ARX (for project)

Nonlinear ARX structure

Recall standard ARX:

$$
\begin{aligned}
y(k) = & - a_1 y(k-1) - a_2 y(k-2) - \ldots - a_{na} y(k-na) \\
& b_1 u(k-1) + b_2 u(k-2) + \ldots + b_{nb} u(k-nb) + e(k)
\end{aligned}
$$

Linear dependence on delayed outputs $y(k-1), \ldots, y(k-na)$ and inputs $u(k-1), \ldots, u(k-nb)$.

Nonlinear ARX (NARX) generalizes this to any nonlinear dependence:

$$
\begin{aligned}
y(k) =& g\big(y(k-1), y(k-2), \ldots, y(k-na), \\
& u(k-1), u(k-2), \ldots, u(k-nb); \theta\big) + e(k)
\end{aligned}
$$

Function g is parameterized by $\theta \in \mathbb{R}^n$, and these parameters can be tuned to fit identification data and thereby model a particular system.

Polynomial NARX

In our particular case, each constant a, b is replaced by a polynomial of the delayed outputs and inputs:

$$\begin{aligned} y(k) = & - p_1(d(k))y(k-1) - p_2(d(k))y(k-2) - \ldots - p_{na}(d(k))y(k-na) \\ & + z_1(d(k))u(k-1) + z_2(d(k))u(k-2) + \ldots + z_{nb}(d(k))u(k-nb) \\ & + e(k) \end{aligned}$$

with $d(k) = [y(k-1), \ldots, y(k-na), u(k-1), \ldots, u(k-nb)]^\top$

Remarks:

- Do not confuse with polynomial form $A(q^{-1})y(k) = B(q^{-1})u(k) + e(k)$
- The parameters are now the coefficients of all polynomials
- If all the polynomials have degree 0, model reduces to linear ARX
- Negative and zero-time y and u can be taken 0, assuming system in zero initial conditions
- By computing the multiplication, we end up with an overall polynomial $g(d(k); \theta)$

Example

Take $na = nb = 1$, then $d(k) = [y(k-1), u(k-1)]^T$ and the model is:

$$y(k) = -p_1(d(k))y(k-1) + z_1(d(k))u(k-1) + e(k)$$

Take degree 1 for both polynomials:

$$\begin{aligned} y(k) = &-[a_1 + a_2 y(k-1) + a_3 u(k-1)]y(k-1) \\ &+ [b_1 + b_2 y(k-1) + b_3 u(k-1)]u(k-1) + e(k) \\ = &-a_1 y(k-1) - a_2 y(k-1)^2 - a_3 u(k-1)y(k-1) \\ &+ b_1 u(k-1) + b_2 y(k-1)u(k-1) + b_3 u(k-1)^2 + e(k) \end{aligned}$$

Linear regression works as usual, finding the parameters a_i, b_j that minimize the MSE!

Example: Polynomial reduction

However: duplicate regressors, $y(k-1)u(k-1)$ will lead to singular system!

$\Rightarrow$ group the duplicates together:

$$\begin{aligned} y(k) =& -a_1 y(k-1) - a_2 y(k-1)^2 + b_1 u(k-1) + b_3 u(k-1)^2 \\ & + c y(k-1)u(k-1) + e(k) \\ =& g(d(k);\theta) + e(k) \end{aligned}$$

where $c = -a_3 + b_2$, and the reduced parameter vector is: $\theta = [a_1, a_2, b_1, b_3, c]^\top$

Recall prediction versus simulation

One-step ahead prediction: True output sequence is known, delays vector $d(k)$ is fully available:

$$x(k) = [y(k-1), \dots, y(k-na), u(k-1), \dots, u(k-nb)]^\top$$
$$\widehat{y}(k) = g(d(k); \widehat{\theta})$$

Simulation: True outputs unknown, use the previously simulated outputs to construct an *approximation* of $d(k)$:

$$\widehat{x}(k) = [\widehat{y}(k-1), \dots, \widehat{y}(k-na), u(k-1), \dots, u(k-nb)]^\top$$
$$\widehat{y}(k) = g(\widehat{d}(k); \widehat{\theta})$$