System Identification

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

Lecturer: Lucian Buşoniu



Part VII

Prediction error methods

Table of contents

1. Model structures
2. General prediction error methods
3. Solving the optimization problem
4. Theoretical guarantees

Classification

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

By number of parameters:

1. Parametric models: have a fixed form (mathematical formula), with a known, often small number of parameters
2. Nonparametric models: cannot be described by a fixed, small number of parameters
   Often represented as graphs or tables

By amount of prior knowledge ("color"):

1. First-principles, white-box models: fully known in advance
2. Black-box models: entirely unknown
3. Gray-box models: partially known

Like ARX, general prediction error methods (PEM) produce *black-box*, *parametric* (polynomial) models.

Table of contents

1 Model structures

2 General prediction error methods

3 Solving the optimization problem

4 Theoretical guarantees

Motivation

General PEM can be seen as an extension of ARX to significantly more general model structures, and are therefore able to identify more classes of systems.

To clarify, we first introduce the general class of models to which PEM can be applied.

General model structure

$$y(k) = G(q^{-1})u(k) + H(q^{-1})e(k)$$

where G and H are discrete-time *transfer functions* – fractions of polynomials. Signal $e(t)$ is zero-mean white noise.

- G is the input-output transfer function.
- H is the disturbance transfer function.

General model structure: Explicit form

By making the fractions explicit and placing the common factors of the denominators of G and H in $A(q^{-1})$, we get the more detailed form:

$$y(k) = \frac{B(q^{-1})}{A(q^{-1})F(q^{-1})}u(k) + \frac{C(q^{-1})}{A(q^{-1})D(q^{-1})}e(k)$$

$$A(q^{-1})y(k) = \frac{B(q^{-1})}{F(q^{-1})}u(k) + \frac{C(q^{-1})}{D(q^{-1})}e(k)$$

where A, B, C, D, F are all polynomials, of orders na, nb, nc, nd, nf:

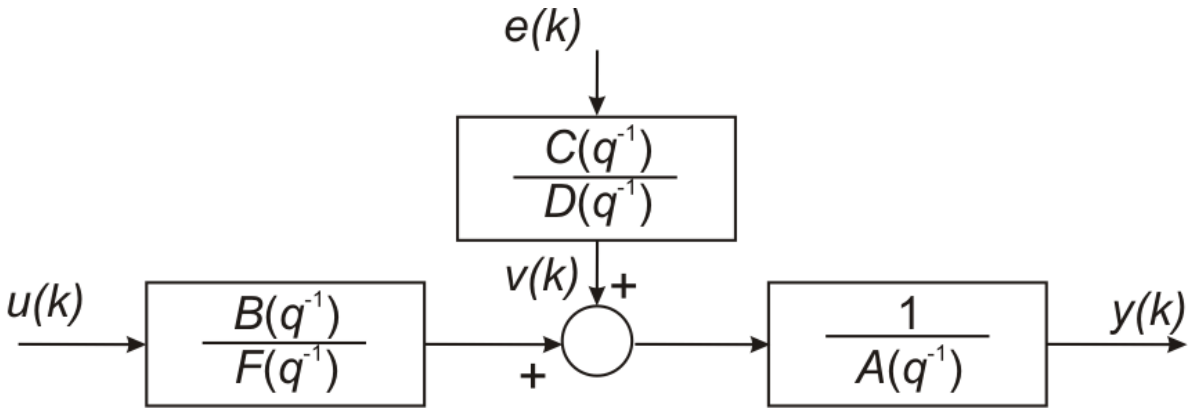
$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_{na} q^{-na}$$

$$B(q^{-1}) = b_1 q^{-1} + \ldots + b_{nb} q^{-nb} \qquad F(q^{-1}) = 1 + f_1 q^{-1} + \ldots + f_{nf} q^{-nf}$$

$$C(q^{-1}) = 1 + c_1 q^{-1} + \ldots + c_{nc} q^{-nc} \qquad D(q^{-1}) = 1 + d_1 q^{-1} + \ldots + d_{nd} q^{-nd}$$

General model structure: Graphical form

$$A(q^{-1})y(k) = \frac{B(q^{-1})}{F(q^{-1})}u(k) + \frac{C(q^{-1})}{D(q^{-1})}e(k)$$



Very general form, all other linear forms are special cases of this. Not for practical use, but to describe algorithms in a generic way. In practice, we use one of the special cases, as exemplified next.

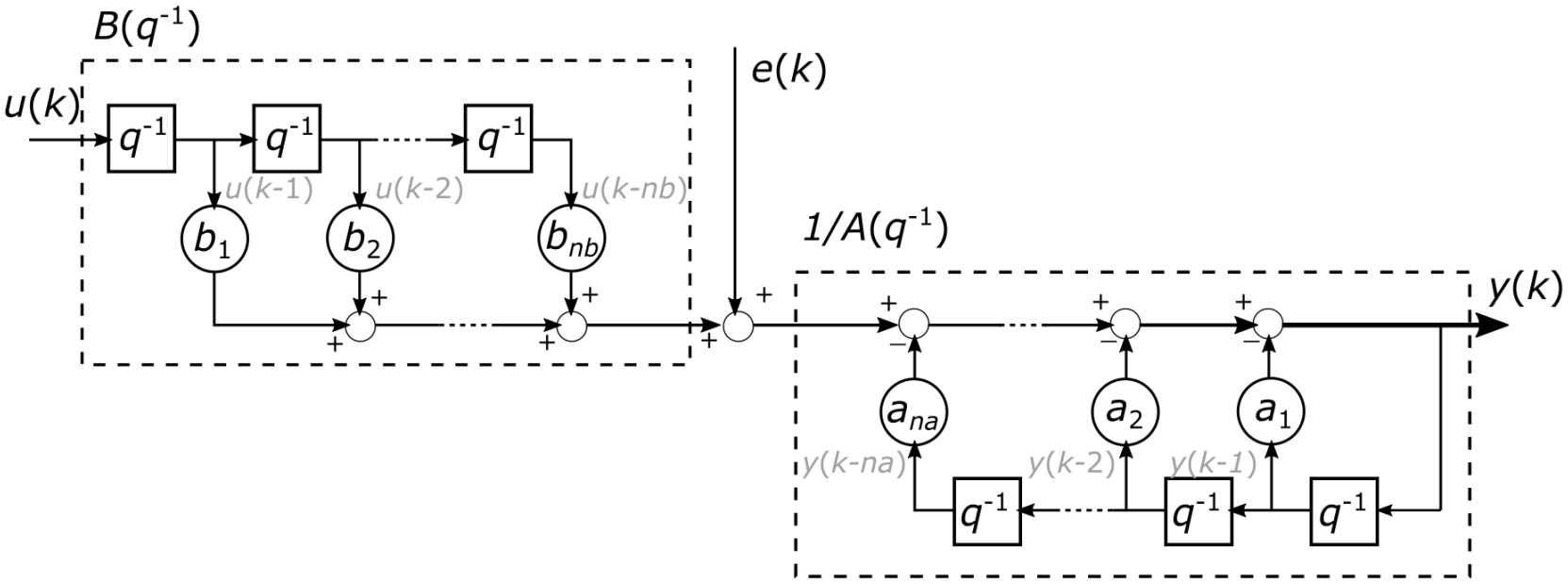
ARMAX model structure

Setting $F = D = 1$ (i.e. orders $nf = nd = 0$), we get:

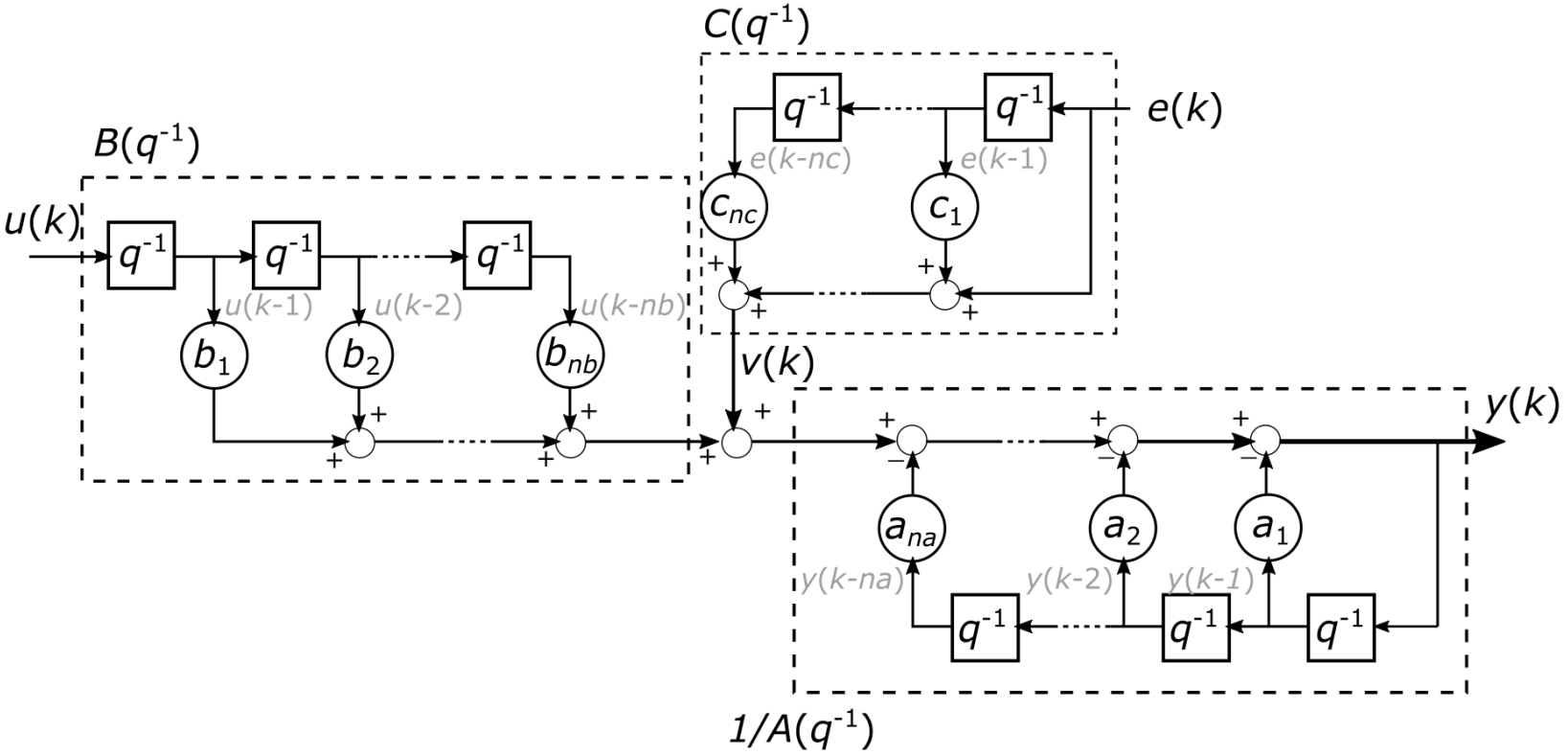
$$A(q^{-1})y(k) = B(q^{-1})u(k) + C(q^{-1})e(k)$$

Name: AutoRegressive (dependence on previous outputs), Moving Average (referring to noise model) with eXogenous input (dependence on u)

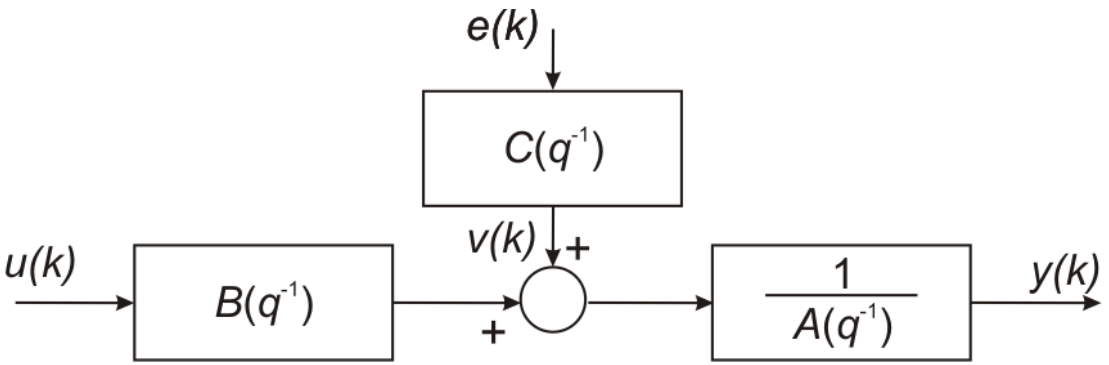
Recall ARX graphical form with delayed signals



ARMAX graphical form



Compact form:



ARMAX: explicit form

$$A(q^{-1})y(k) = B(q^{-1})u(k) + C(q^{-1})e(k)$$
$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_{na} q^{-na}$$
$$B(q^{-1}) = b_1 q^{-1} + \ldots + b_{nb} q^{-nb}$$
$$C(q^{-1}) = 1 + c_1 q^{-1} + \ldots + c_{nc} q^{-nc}$$

$$\begin{aligned} y(k) + a_1 y(k-1) + \ldots + a_{na} y(k-na) \\ = b_1 u(k-1) + \ldots + b_{nb} u(k-nb) \\ + e(k) + c_1 e(k-1) + \ldots + c_{nc} e(k-nc) \end{aligned}$$

with parameter vector:

$$\theta = [a_1, \ldots, a_{na}, b_1, \ldots, b_{nb}, c_1, \ldots, c_{nc}]^\top \in \mathbb{R}^{na+nb+nc}$$

Running example: 1st order ARMAX

Take $na = nb = nc = 1$. This leads to the 1st order ARMAX model:

$$(1 + aq^{-1})y(k) = bq^{-1}u(k) + (1 + cq^{-1})e(k)$$
$$y(k) = -ay(k-1) + bu(k-1) + ce(k-1) + e(k)$$

where we skipped the indices in a, b, c since there is a single parameter in each polynomial.

We will be using this as a running example throughout this part.

Special case of ARMAX: ARX

Setting $C = 1$ in ARMAX ($nc = 0$), we get:

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

precisely the ARX model we worked with before.

Compared to ARX, ARMAX can model more intricate disturbances ($C(q^{-1})e(k)$ instead of $e(k)$, which is often assumed to be zero-mean white-noise).

Recall: FIR special case of ARX

Further 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.

Recall however also that FIR and ARX are fundamentally different (nonparametric impulse response versus parametric transfer function).

Output error

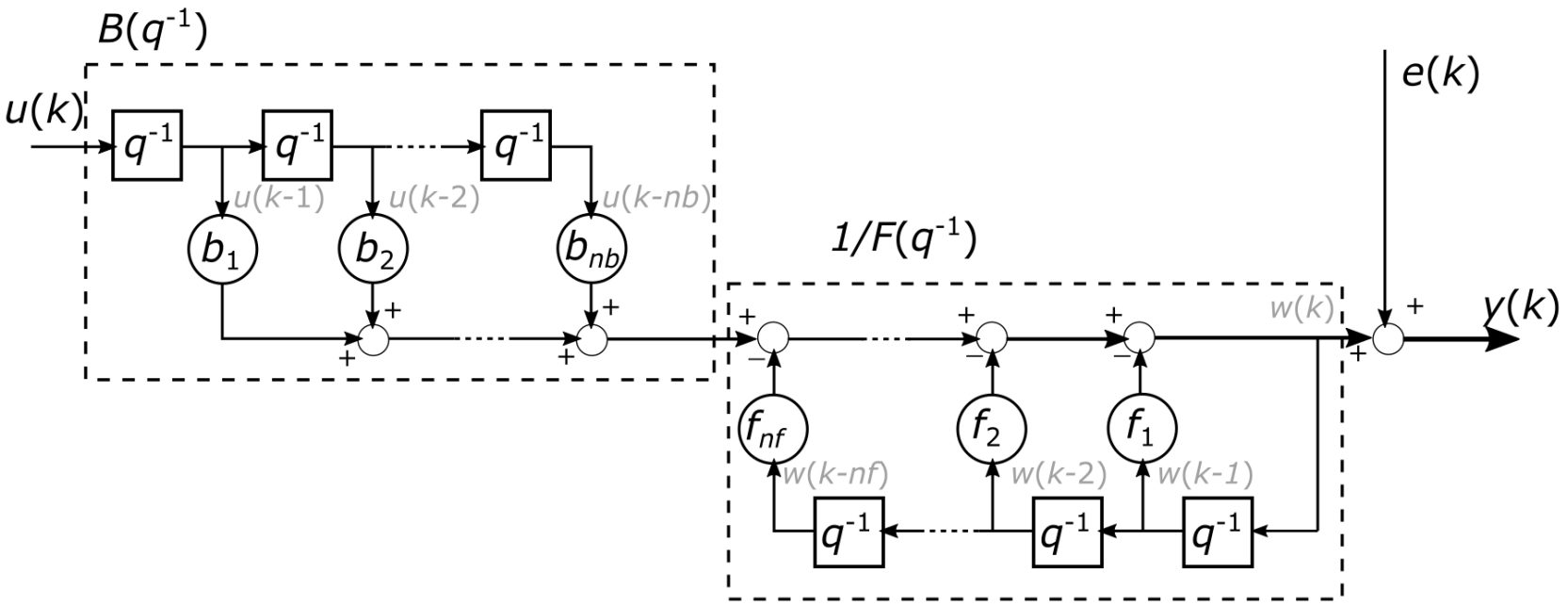
Output Error, OE:

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

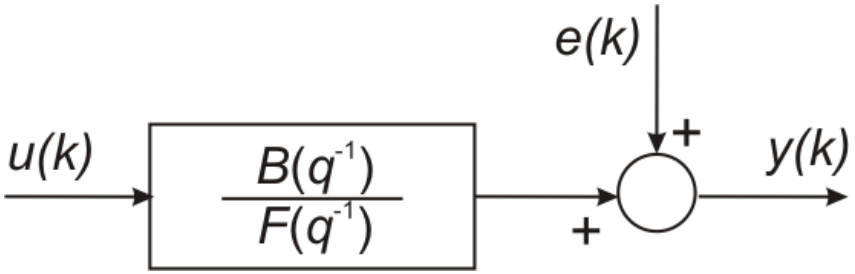
obtained for $na = nc = nd = 0$, i.e. $A = C = D = 1$.

This corresponds to simple, additive measurement noise on the output (the “output error”).

OE graphical form

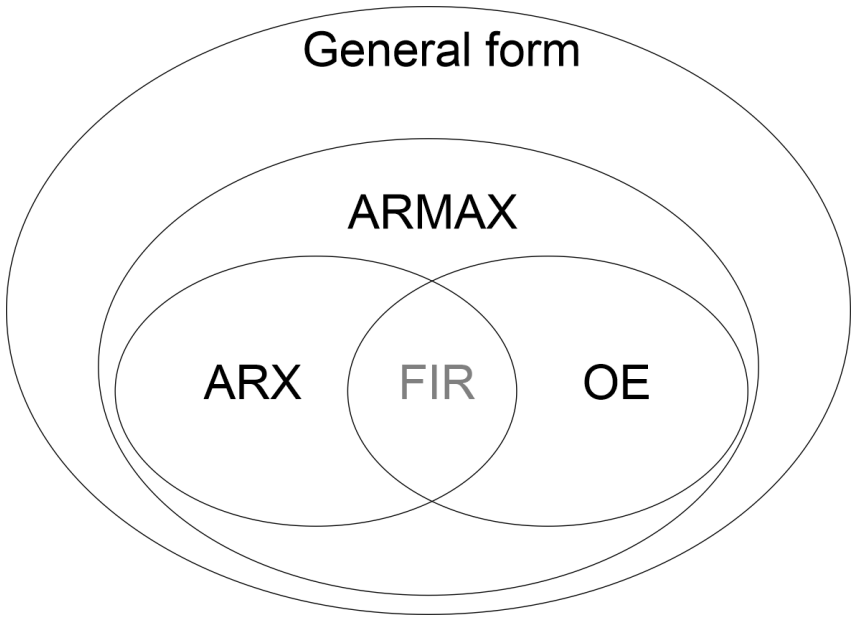


Compact form:



Summary and overall relationship between models

- General form of models used in PEM
- Specific structures, including some we already saw
- Relationship:



- ARMAX vs. ARX vs. OE: different disturbance.
- ARX vs. FIR: models of different nature.

Exercises

1. What is the explicit form (with delayed-signal equations) of the OE model?
2. Draw the graphical form of the FIR and of the general-case structure.
3. Search the literature for the Box-Jenkins model structure, and write its explicit and graphical form. What is the relationship of this structure with those discussed above?

Table of contents

1 Model structures

2 General prediction error methods
- Stepping stone: ARX revisited
- General case
- Example: 1st order ARMAX
- Matlab example

3 Solving the optimization problem

4 Theoretical guarantees

ARX reinterpreted as a PEM

1. Compute predictions at each step, $\widehat{y}(k) = \varphi^\top(k)\theta$ given parameters θ.
2. Compute prediction errors at each step, $\varepsilon(k) = y(k) - \widehat{y}(k)$.
3. Find a parameter vector θ minimizing criterion $V(\theta) = \frac{1}{N}\sum_{k=1}^{N} \varepsilon^2(k)$.

The procedure above is just a reinterpretation, equivalent to the algorithm discussed in the ARX lecture.

Prediction error methods are obtained by extending this procedure to general model structures.

ARX reinterpreted as a PEM (continued)

Remarks:

- For ARX, we already know how to minimize the MSE (from linear regression); for more general models new methods will be introduced.
- ARX predictor $\widehat{y}(k)$ is chosen to achieve the error $\varepsilon(k) = e(k)$, equal to the noise. We will aim to achieve the same error in the general PEM, intuitively because we cannot do better.
  Note different meanings of $\varepsilon(k)$ (prediction error) and $e(k)$ (noise)
- The prediction error is just a rearrangement of the equation $y(k) = \varphi^\top(k)\theta + \varepsilon(k) = \widehat{y}(k) + \varepsilon(k)$.

Checklist

For each model structure, we will have to run through the three steps: prediction, prediction error, and minimization of the MSE.

We keep a running checklist:

	ARX	General PEM	1st order ARMAX
prediction $\widehat{y}(k)$	✓		
prediction error $\varepsilon(k)$	✓		
minimization of $V(\theta)$	✓		

Table of contents

1 Model structures

2 General prediction error methods
- Stepping stone: ARX revisited
- General case
- Example: 1st order ARMAX
- Matlab example

3 Solving the optimization problem

4 Theoretical guarantees

Recall: General model structure

$$y(k) = G(q^{-1})u(k) + H(q^{-1})e(k)$$

where G and H are discrete-time transfer functions:

$$y(k) = \frac{B(q^{-1})}{A(q^{-1})F(q^{-1})}u(k) + \frac{C(q^{-1})}{A(q^{-1})D(q^{-1})}e(k)$$

In the sequel, we skip argument q^{-1} to make the equations readable, and implicitly understand that each capital letter is a transfer function or polynomial of q^{-1}.

Prediction error

We start by deriving $e(k)$.

$$y(k) = Gu(k) + He(k)$$
$$\Rightarrow e(k) = H^{-1}(y(k) - Gu(k))$$

where $H^{-1} = \frac{AD}{C}$ is the inverse of polynomial fraction H.

The predictor will be derived so that the prediction error $\varepsilon(k) = y(k) - \widehat{y}(k) = e(k)$. So, the same formula can also be used to compute $\varepsilon(k)$:

$$\boxed{\varepsilon(k) = H^{-1}(y(k) - Gu(k))}$$

This is a dynamical system that can be simulated to compute $\varepsilon(k)$.

Predictor

To achieve error $e(k)$, the predictor dynamics must be:

$$\begin{aligned}\widehat{y}(k) &= y(k) - e(k) \\ &= y(k) - H^{-1}(y(k) - Gu(k)) \\ &= \boxed{(1 - H^{-1})y(k) + H^{-1}Gu(k)}\end{aligned}$$

Remark: In order to have a *causal* predictor, that only depends on past values of the output and input, we require $G(0) = 0$ and $H(0) = 1$. This is true because $B(0) = 0, A(0) = C(0) = D(0) = 1$.

Preview: Finding the parameters & using the model

Finding the parameters: Once the procedure to compute the errors is available, the parameters θ are found by minimizing criterion $V(\theta) = \frac{1}{N}\sum_{k=1}^{N} \varepsilon^2(k)$. This will usually require multiple evaluations of error signal $\varepsilon(k)$, for multiple values of the parameters θ.

We do not yet go into specific computational methods to solve the error minimization problem. We will study them in detail in the next section.

Using the model: Once an estimate $\widehat{\theta}$ of the optimum is found, the predictor formula is applied to compute the model outputs $\hat{y}(k)$. The model can also be adapted to run in simulation mode.

Checklist

	ARX	General PEM	1st order ARMAX
prediction $\widehat{y}(k)$	✓	✓	
prediction error $\varepsilon(k)$	✓	✓	
minimization of $V(\theta)$	✓	?	

Table of contents

1 Model structures

2 General prediction error methods
- Stepping stone: ARX revisited
- General case
- Example: 1st order ARMAX
- Matlab example

3 Solving the optimization problem

4 Theoretical guarantees

Running example: 1st order ARMAX

Recall ARMAX:

$$Ay(k) = Bu(k) + Ce(k)$$

Placing it in the standard form, we get:

$$\begin{aligned} y(k) &= \frac{B}{A}u(k) + \frac{C}{A}e(k) \\ &= Gu(k) + He(k) \end{aligned}$$

So $G = \frac{B}{A}$, $H = \frac{C}{A}$

For 1st order: $A = 1 + aq^{-1}$, $B = bq^{-1}$, $C = 1 + cq^{-1}$.

1st order ARMAX: Prediction error

Recall general error formula:

$$\varepsilon(k) = H^{-1}(y(k) - Gu(k))$$

For our case, since $G = \frac{B}{A}$, $H = \frac{C}{A}$:

$$\begin{aligned}
\varepsilon(k) &= \frac{A}{C}\left(y(k) - \frac{B}{A}u(k)\right) \\
C\varepsilon(k) &= Ay(k) - Bu(k) \\
(1 + cq^{-1})\varepsilon(k) &= (1 + aq^{-1})y(k) - bq^{-1}u(k) \\
\varepsilon(k) + c\varepsilon(k-1) &= y(k) + ay(k-1) - bu(k-1) \\
\varepsilon(k) &= -c\varepsilon(k-1) + y(k) + ay(k-1) - bu(k-1)
\end{aligned}$$

This is a *dynamical*, recursive formula which needs to be simulated over time! Requires initialization of $\varepsilon(0)$, usually taken 0.

1st order ARMAX: Predictor

Recall general predictor formula:

$$\widehat{y}(k) = (1 - H^{-1})y(k) + H^{-1}Gu(k)$$

For our case, since $G = \frac{B}{A}$, $H = \frac{C}{A}$:

$$\begin{aligned}
\widehat{y}(k) &= \left(1 - \frac{A}{C}\right) y(k) + \frac{\cancel{A}}{C}\frac{B}{\cancel{A}}u(k) \\
C\widehat{y}(k) &= (C - A)y(k) + Bu(k) \\
(1 + cq^{-1})\widehat{y}(k) &= (\cancel{1} + cq^{-1} - \cancel{1} - aq^{-1})y(k) + bq^{-1}u(k) \\
\widehat{y}(k) + c\widehat{y}(k-1) &= (c - a)y(k-1) + bu(k-1) \\
\widehat{y}(k) &= -c\widehat{y}(k-1) + (c - a)y(k-1) + bu(k-1)
\end{aligned}$$

Again, a *dynamical*, recursive formula. Requires initialization at $\widehat{y}(0)$, usually taken 0.

Checklist

	ARX	General PEM	1st order ARMAX
prediction $\widehat{y}(k)$	✓	✓	✓
prediction error $\varepsilon(k)$	✓	✓	✓
minimization of $V(\theta)$	✓	?	?

Table of contents

1 Model structures

2 General prediction error methods
- Stepping stone: ARX revisited
- General case
- Example: 1st order ARMAX
- Matlab example

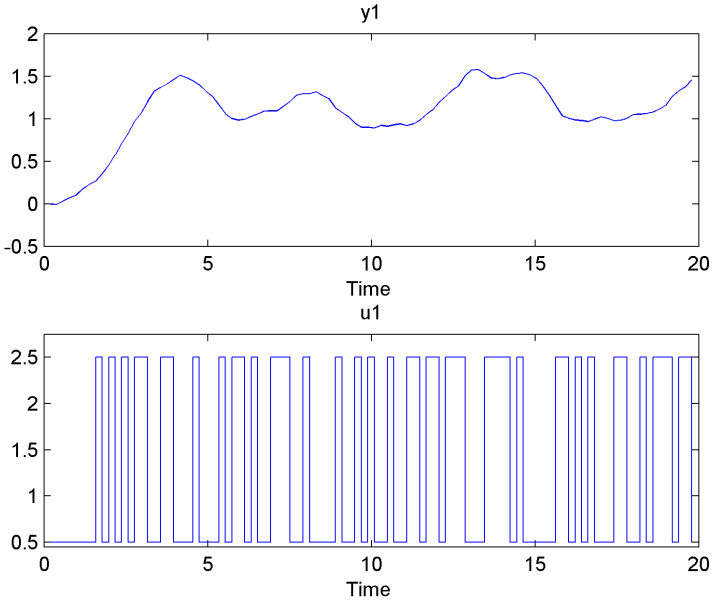
3 Solving the optimization problem

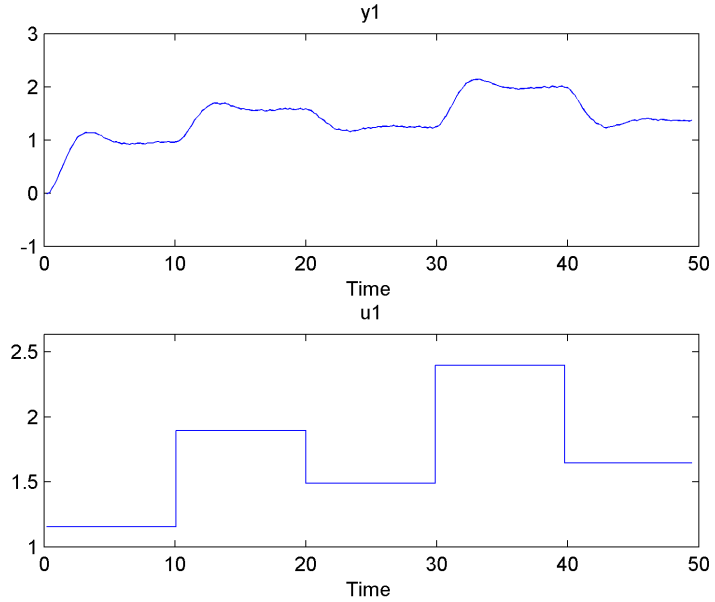
4 Theoretical guarantees

Experimental data

Consider again the experimental data on which ARX was applied.

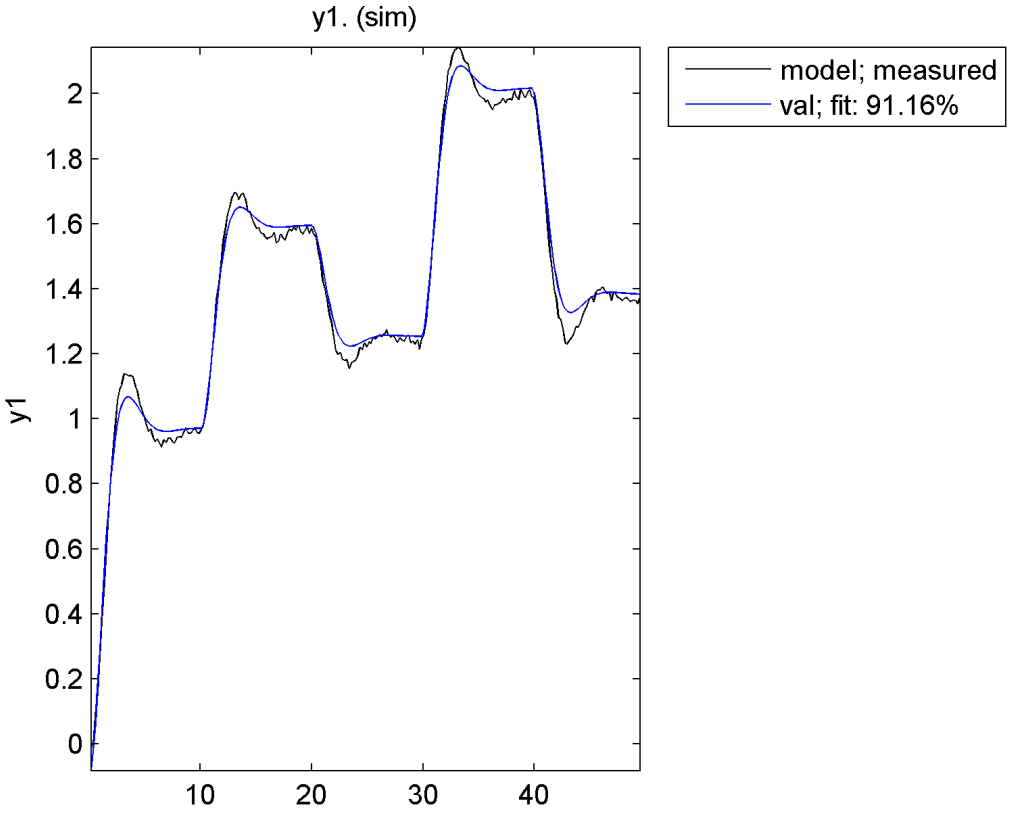
`plot(id);` and `plot(val);`





Recall: ARX result

Assuming the system is second-order and without time delay, we take $na = 2, nb = 2, nk = 1$.



Results are quite bad.

Identifying an ARMAX model

```
mARMAX = armax(id, [na, nb, nc, nk]);
```

Arguments:

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

Like for ARX, structure includes the explicit minimum delay nk between inputs and outputs.

$$\begin{aligned} y(k) + a_1 y(k-1) + a_2 y(k-2) + \ldots + a_{na} y(k-na) \\ = b_1 u(k-nk) + b_2 u(k-nk-1) + \ldots + b_{nb} u(k-nk-nb+1) \\ + e(k) + c_1 e(k-1) + c_2 e(k-2) + \ldots + c_{nc} e(k-nc) \end{aligned}$$

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

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

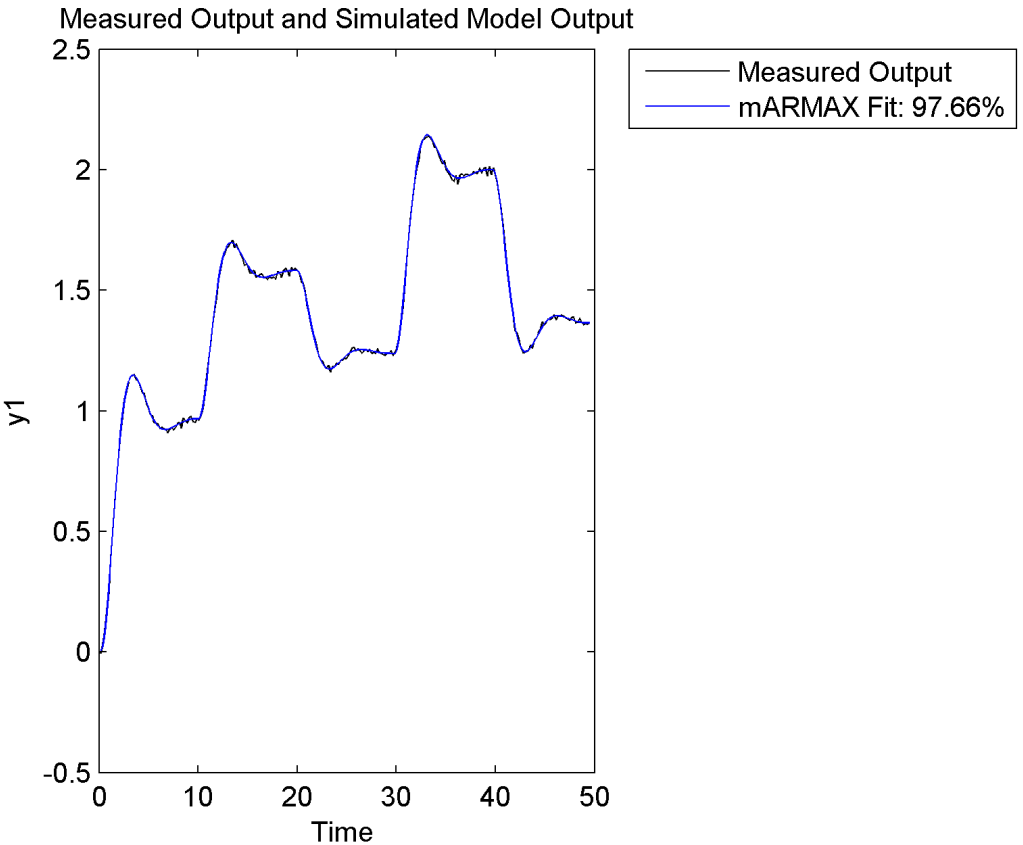
$$B(q^{-1}) = (b_1 + b_2 q^{-1} + b_{nb} q^{-nb+1})$$

$$C(q^{-1}) = (1 + c_1 q^{-1} + c_2 q^{-2} + \ldots + c_{nc} q^{-nc})$$

Remark: As for ARX, the theoretical structure is obtained by setting $nk = 1$ (and to represent $nk > 1$ in the theoretical structure, change B like in the ARX Matlab example).

ARMAX model

Considering the system is 2nd order with no time delay, take $na = 2$, $nb = 2$, $nc = 2$, $nk = 1$. Validation: `compare(val, mARMAX);`

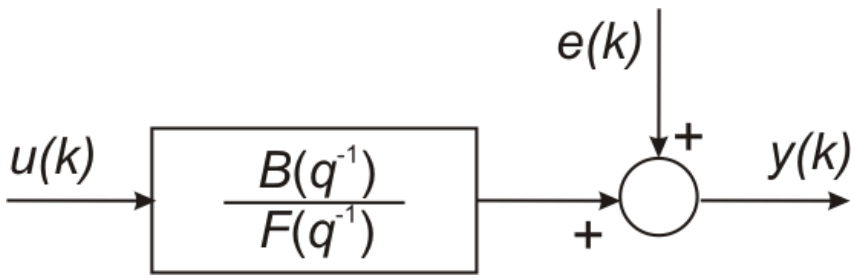


In contrast to ARX, results are good. Flexible noise model pays off.

Identifying an OE model

Recall OE model structure:

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



Identifying an OE model (continued)

```
mOE = oe(id, [nb, nf, nk]);
```

Arguments:

1. Identification data.
2. Array containing the orders of B, F, and the delay nk.

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

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

$$F(q^{-1}) = (1 + f_1 q^{-1} + f_2 q^{-2} + \ldots + f_{nf} q^{-nf})$$

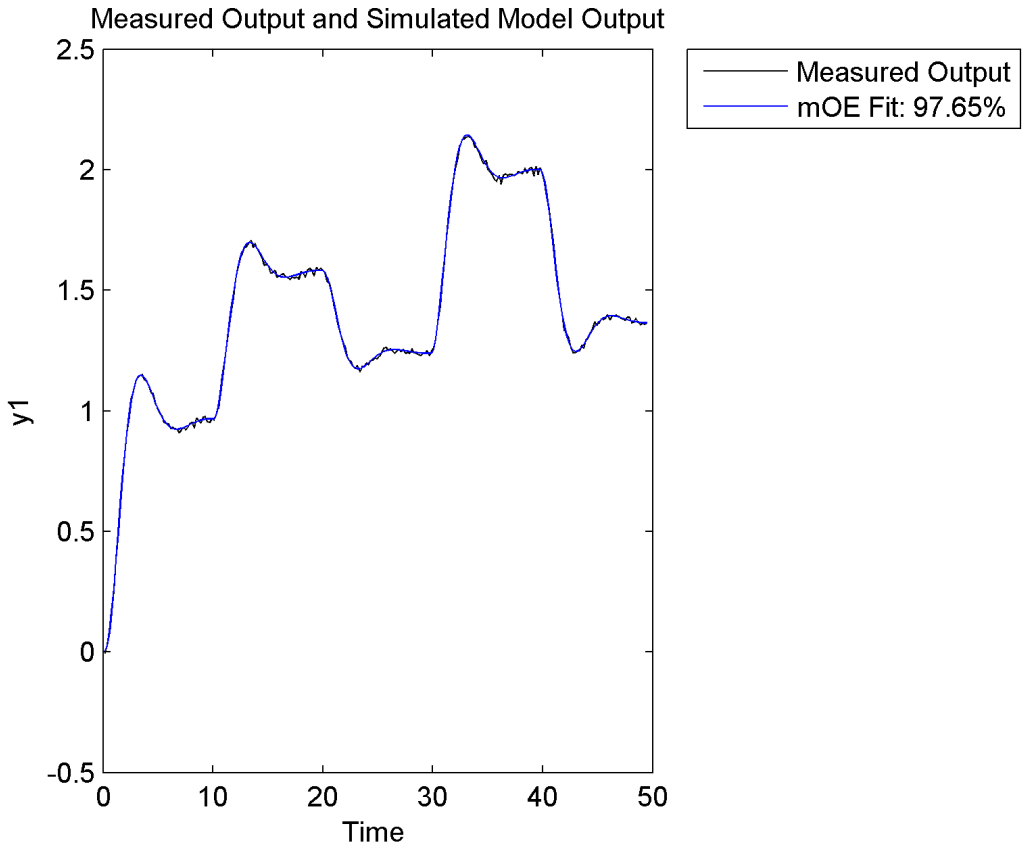
Explicit formula:

$$\begin{aligned} &y(k) + f_1 y(k-1) + f_2 y(k-2) + \ldots + f_{nf} y(k-nf) \\ &\quad = b_1 u(k-nk) + b_2 u(k-nk-1) + \ldots + b_{nb} u(k-nk-nb+1) \\ &\qquad + e(k) + f_1 e(k-1) + f_2 e(k-2) + \ldots + f_{nf} e(k-nf) \end{aligned}$$

Remark: Like before, can transform into theoretical structure by setting $nk = 1$ (or changing B if $nk > 1$).

OE model

Considering the system is second-order with no time delay, we take $nb = 2, nf = 2, nk = 1$. Validation: `compare(val, mOE);`



Results as good as ARMAX. System turns out to obey both model structures. Question: What is the true structure then?

Summary so far

- ARX reinterpreted as a PEM, isolating the predictor and prediction error
- Derived PEM in the general-model case: prediction error, predictor
- Example application to first-order ARMAX: prediction error, predictor
- Matlab example illustrating ARX, ARMAX, and OE identification
- Error-minimization method remains open

Table of contents

1 Model structures

2 General prediction error methods

3 Solving the optimization problem

4 Theoretical guarantees

Checklist

	ARX	General PEM	1st order ARMAX
prediction $\widehat{y}(k)$	✓	✓	✓
prediction error $\varepsilon(k)$	✓	✓	✓
minimization of $V(\theta)$	✓	?	?

Optimization problem

Objective of identification procedure: Minimize mean squared error

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

where $\varepsilon(k)$ are the prediction errors. In the general case:

$$\varepsilon(k) = H^{-1}(q^{-1})(y(k) - G(q^{-1})u(k))$$

Solution: $\widehat{\theta} = \arg\min_{\theta} V(\theta)$

So far we took this solution for granted and investigated its properties. While in ARX linear regression could be applied to find $\widehat{\theta}$, in general this does not work. Main implementation question:

How to solve the optimization problem?

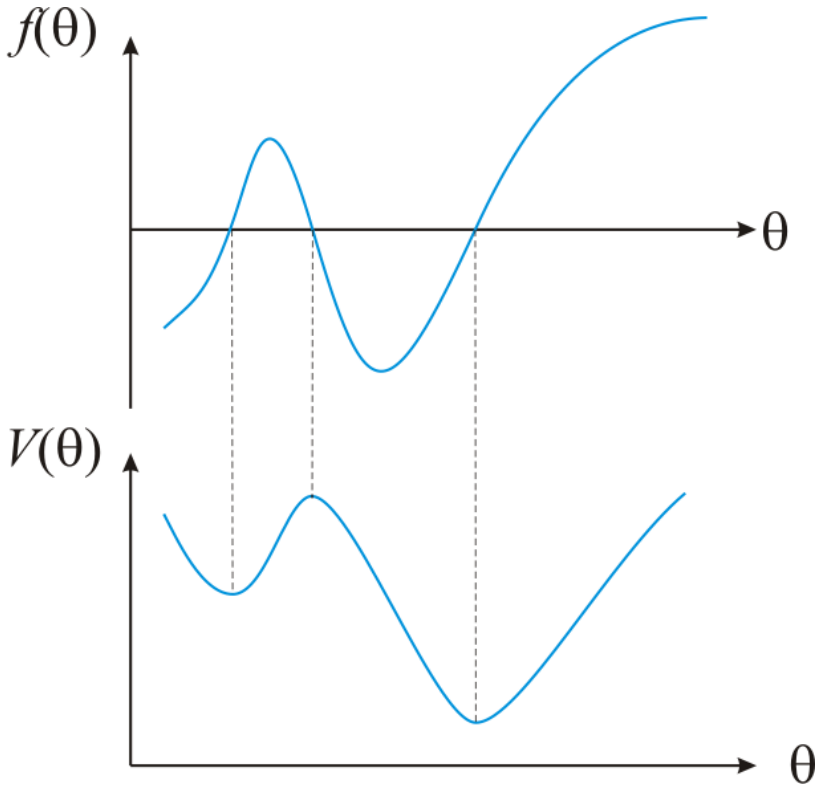
Minimization via derivative root

Consider first the scalar case, $\theta \in \mathbb{R}$.

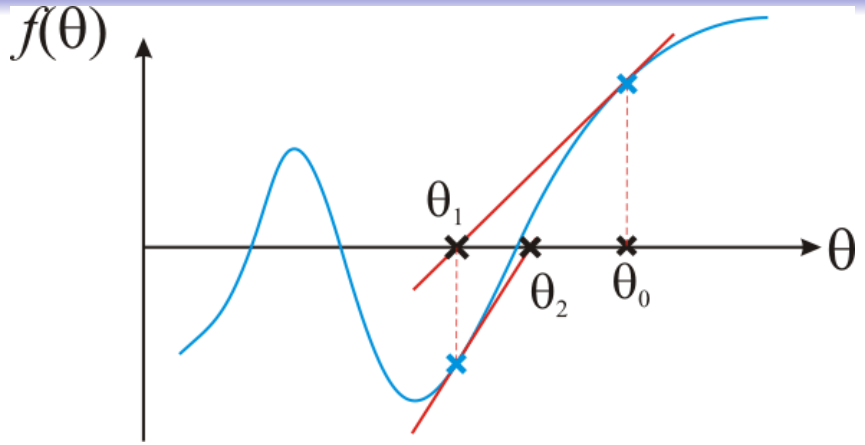
Idea: at any minimum, the derivative $f(\theta) = \frac{dV}{d\theta}$ is zero. So, find a root of $f(\theta)$.

Remarks:

- Care must be taken to find a minimum and not a maximum or inflexion. This can be checked with the second derivative, $\frac{d^2V}{d\theta^2} = \frac{df}{d\theta} > 0$.
- We may also find a local minimum which is larger (worse) than the global one.



Newton's method for root finding



- Start from some initial point θ_0.
- At iteration ℓ, next point $\theta_{\ell+1}$ is the intersection between abscissa and tangent at f in current point θ_ℓ. By geometry arguments:

$$\theta_{\ell+1} = \theta_\ell - \frac{f(\theta_\ell)}{\frac{df(\theta_\ell)}{d\theta}}$$

Remarks:

- Notation $\frac{df(\theta_\ell)}{d\theta}$ means the value of derivative $\frac{df}{d\theta}$ at point θ_ℓ.
- The slope of the tangent is $\frac{df(\theta_\ell)}{d\theta}$.
- $\theta_{\ell+1}$ is the "best guess" for root given current point θ_ℓ.

Newton's method for optimization

Replace $f(\theta)$ by $\frac{dV}{d\theta}$ to get back to optimization problem:

$$\theta_{\ell+1} = \theta_\ell - \frac{\frac{dV(\theta_\ell)}{d\theta}}{\frac{d^2V(\theta_\ell)}{d\theta^2}}$$

Gradient and Hessian

To extend from the scalar case to $\theta \in \mathbb{R}^n$, we need first and second-order derivatives of $V(\theta)$, taking into account that $V : \mathbb{R}^n \to \mathbb{R}$. Then:

$$\frac{dV}{d\theta} = \begin{bmatrix} \frac{\partial V}{\partial \theta_1} \\ \frac{\partial V}{\partial \theta_2} \\ \vdots \\ \frac{\partial V}{\partial \theta_n} \end{bmatrix}, \quad \frac{d^2V}{d\theta^2} = \begin{bmatrix} \frac{\partial^2 V}{\partial \theta_1^2} & \frac{\partial^2 V}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 V}{\partial \theta_1 \theta_n} \\ \frac{\partial^2 V}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 V}{\partial \theta_2^2} & \cdots & \frac{\partial^2 V}{\partial \theta_2 \theta_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 V}{\partial \theta_n \partial \theta_1} & \frac{\partial^2 V}{\partial \theta_n \theta_2} & \cdots & \frac{\partial^2 V}{\partial \theta_n^2} \end{bmatrix}$$

Gradient $\frac{dV}{d\theta}$ is an n-vector, and Hessian $\frac{d^2V}{d\theta^2}$ an $n \times n$-matrix.

Newton's method for optimization – general case

Start from scalar-case formula:

$$\theta_{\ell+1} = \theta_\ell - \frac{\frac{dV(\theta_\ell)}{d\theta}}{\frac{d^2V(\theta_\ell)}{d\theta^2}}$$

and extend to use the gradient vector and Hessian matrix:

$$\theta_{\ell+1} = \theta_\ell - \left[\frac{d^2V(\theta_\ell)}{d\theta^2}\right]^{-1} \frac{dV(\theta_\ell)}{d\theta}$$

Add a step size $\alpha_\ell > 0$. Final formula:

$$\theta_{\ell+1} = \theta_\ell - \alpha_\ell \left[\frac{d^2V(\theta_\ell)}{d\theta^2}\right]^{-1} \frac{dV(\theta_\ell)}{d\theta}$$

Remark:

- The stepsize helps with the convergence of the method, by keeping updates in the local area where the function is approximately quadratic.

Stopping criterion

Algorithm can be stopped:

- When the difference between consecutive parameter vectors is small, e.g. $\max_{i=1}^{n} |\theta_{i,\ell+1} - \theta_{i,\ell}|$ smaller than some preset threshold.

or

- When the number of iterations ℓ exceeds a preset maximum.

Remark: Nothing so far in Newton's method was specific to system identification. The method works in general for any optimization problem! Next, we return to system identification.

Computing the derivatives

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

Keeping in mind that $\varepsilon(k)$ depends on θ, from matrix calculus:

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

$$\frac{d^2V}{d\theta^2} = \frac{2}{N}\sum_{k=1}^{N}\frac{d\varepsilon(k)}{d\theta}\left[\frac{d\varepsilon(k)}{d\theta}\right]^\top + \frac{2}{N}\sum_{k=1}^{N}\varepsilon(k)\frac{d^2\varepsilon(k)}{d\theta^2}$$

where:

- $\frac{d\varepsilon(k)}{d\theta}$ is the vector derivative and $\frac{d^2\varepsilon(k)}{d\theta^2}$ the Hessian of $\varepsilon(k)$.
- $\frac{d\varepsilon(k)}{d\theta}[\frac{d\varepsilon(k)}{d\theta}]^\top$ is an $n \times n$ matrix.

Gauss-Newton

Ignore the second term in the Hessian of V and just use the first term:

$$\mathcal{H} = \frac{2}{N} \sum_{k=1}^{N} \frac{d\varepsilon(k)}{d\theta} \left[\frac{d\varepsilon(k)}{d\theta} \right]^\top$$

leading to the Gauss-Newton algorithm:

$$\theta_{\ell+1} = \theta_\ell - \alpha_\ell \mathcal{H}^{-1} \frac{dV(\theta_\ell)}{d\theta}$$

Motivation:

- Quadratic form of $\mathcal{H}$ gives better algorithm behavior.
- Simpler computation.

The details of how to compute $\frac{d\varepsilon(k)}{d\theta}$ depend on the model structure chosen.

Checklist

	ARX	General PEM	1st order ARMAX
prediction $\widehat{y}(k)$	✓	✓	✓
prediction error $\varepsilon(k)$	✓	✓	✓
minimization of $V(\theta)$	✓	✓	?

Example: 1st order ARMAX

Recall model and prediction error for 1st order ARMAX:

$$y(k) = -ay(k-1) + bu(k-1) + ce(k-1) + e(k)$$
$$\varepsilon(k) = -c\varepsilon(k-1) + y(k) + ay(k-1) - bu(k-1)$$

We need $\frac{d\varepsilon(k)}{d\theta} = [\frac{\partial\varepsilon(k)}{\partial a}, \frac{\partial\varepsilon(k)}{\partial b}, \frac{\partial\varepsilon(k)}{\partial c}]^\top$. Differentiating second equation:

$$\frac{\partial\varepsilon(k)}{\partial a} = -c\frac{\partial\varepsilon(k-1)}{\partial a} + y(k-1)$$
$$\frac{\partial\varepsilon(k)}{\partial b} = -c\frac{\partial\varepsilon(k-1)}{\partial b} - u(k-1)$$
$$\frac{\partial\varepsilon(k)}{\partial c} = -c\frac{\partial\varepsilon(k-1)}{\partial c} - \varepsilon(k-1)$$

So, $\frac{\partial\varepsilon(k)}{\partial a}, \frac{\partial\varepsilon(k)}{\partial b}, \frac{\partial\varepsilon(k)}{\partial c}$ are dynamical signals! They can be computed using the recursions above, starting e.g. from 0 initial values.

General prediction error method

initialize θ_0, iteration counter $\ell = 0$

repeat

given current value of parameter vector θ_ℓ,

apply recursions to find $\frac{d\varepsilon(k)}{d\theta}$, $k = 1, \ldots, N$

plug $\frac{d\varepsilon(k)}{d\theta}$ into equations for $\frac{dV}{d\theta}$, $\mathcal{H}$:

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

$$\mathcal{H} = \frac{2}{N} \sum_{k=1}^{N} \frac{d\varepsilon(k)}{d\theta} \left[\frac{d\varepsilon(k)}{d\theta} \right]^\top$$

apply Gauss-Newton update to find $\theta_{\ell+1}$:

$$\theta_{\ell+1} = \theta_\ell - \alpha_\ell \mathcal{H}^{-1} \frac{dV(\theta_\ell)}{d\theta}$$

increment counter: $\ell = \ell + 1$

until $\|\theta_{\ell+1} - \theta_\ell\|$ is small enough, or maximum ℓ was reached

Checklist

	ARX	General PEM	1st order ARMAX
prediction $\widehat{y}(k)$	✓	✓	✓
prediction error $\varepsilon(k)$	✓	✓	✓
minimization of $V(\theta)$	✓	✓	✓

Summary for optimization

- Need for nonlinear regression
- Newton's method for root finding, scalar case
- Newton's method for optimization, scalar case
- Generalization to n dimensions; computing the gradient and Hessian
- Gauss-Newton optimization
- Example on first-order ARMAX, with complete formulas and algorithm

Table of contents

1 Model structures

2 General prediction error methods

3 Solving the optimization problem

4 Theoretical guarantees

Assumptions

Assumptions

1. The true system satisfies the model structure chosen, i.e., there exists at least one θ_0 so that for any input $u(k)$ and the corresponding output $y(k)$ of the true system, we have:

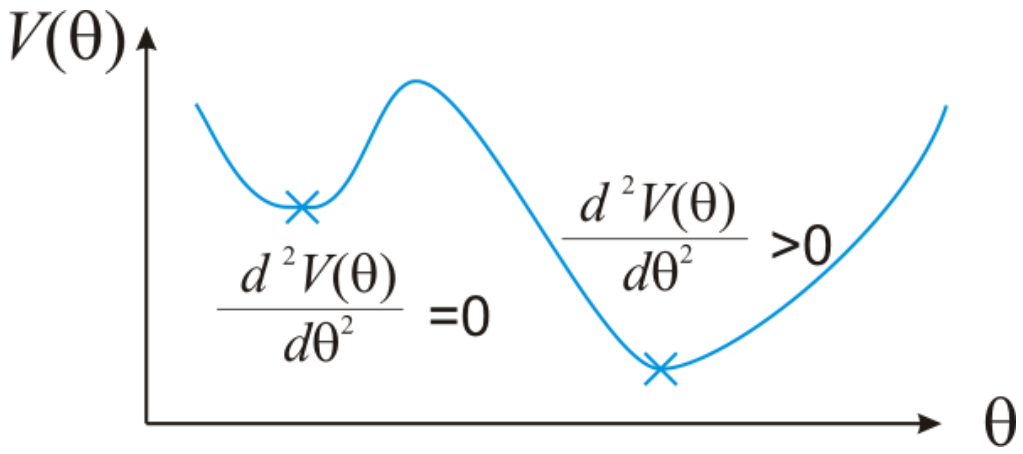
$$y(k) = G(q^{-1}; \theta_0)u(k) + H(q^{-1}; \theta_0)e(k)$$

   with $e(k)$ *white noise*.
2. Signals $u(k)$ and $y(k)$ are stationary stochastic processes.
3. The input signal $u(k)$ has a sufficiently high order of persistent excitation and is independent from the noise $e(k)$ (the experiment is performed in open loop).
4. The Hessian $\frac{d^2V}{d\theta^2}$ is nonsingular at the minimum points of V.

Hessian discussion

4. The Hessian $\frac{d^2V}{d\theta^2}$ is nonsingular at the minimum points of V.

Recall $V(\theta) = \frac{1}{N}\sum_{k=1}^{N}\varepsilon^2(k)$, the MSE. This assumption ensures V is not "flat" around minima.



Guarantee

Theorem

Under Assumptions 1-4, $\widehat{\theta}$ converges to a true parameter vector θ_0 as $N \to \infty$.

Remark: Consistency guarantee.