System Identification

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

Lecturer: Lucian Buşoniu

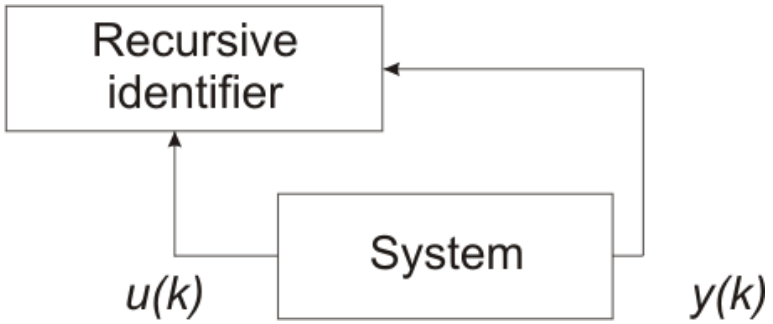
Part IX

Recursive identification methods

Table of contents

1. Introduction and motivation
2. Recursive least-squares and ARX
3. Recursive instrumental variables

Recursive identification: Idea



Recursive methods can work *online*, while the system is running.

At step k, compute *new parameter estimate* $\widehat{\theta}(k)$, based on the previous estimate $\widehat{\theta}(k-1)$ and newly available data $u(k)$, $y(k)$.

Remark: To contrast them with recursive identification, the previous methods, which used the whole data set at once, will be called *batch* identification.

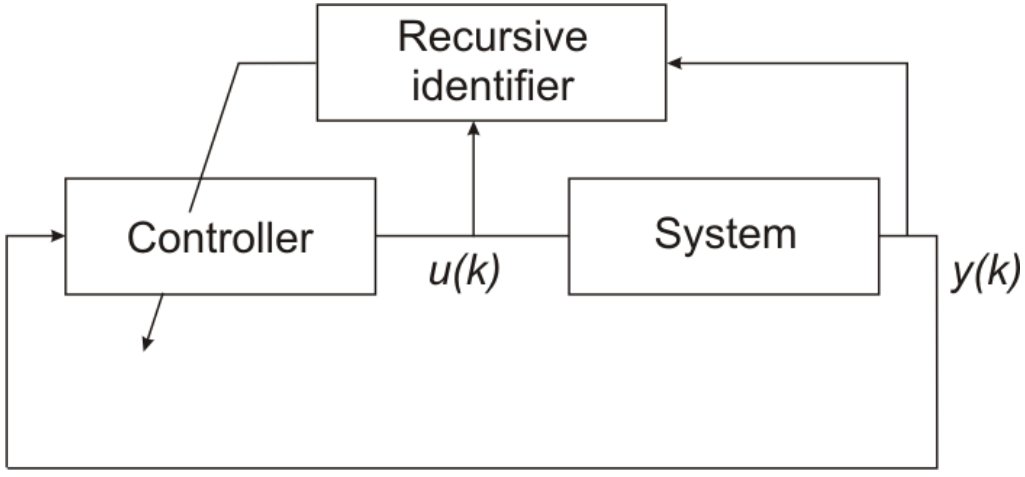
Motivation

Recursive methods:

- Require less memory, and less computation *for each update*, than the whole batch algorithm.

$\Rightarrow$ Easier to apply in real-time.
(total computation, after the entire dataset, may be larger)

- When properly modified, can deal with systems that change over time (*time-varying*).
- If model is used to tune a controller, we obtain *adaptive control*:



Disadvantage

Recursive methods are usually approximations of batch techniques
$\Rightarrow$ guarantees more difficult to obtain.

Motivating example: Estimating a scalar

Recall scalar estimation. Model:

$$y(k) = b + e(k) = 1 \cdot b + e(k) = \varphi(k)\theta + e(k)$$

where $\varphi(k) = 1 \forall k$, $\theta = b$.

For the data points up to and including k:

$$\begin{aligned} y(1) &= \varphi(1)\theta = 1 \cdot b \\ &\dots \\ y(k) &= \varphi(k)\theta = 1 \cdot b \end{aligned}$$

After calculation, the solution of this system is:

$$\widehat{\theta}(k) = \frac{1}{k}[y(1) + \ldots + y(k)]$$

(estimate is the average of all measurements, filtering out the noise).

Estimating a scalar: Recursive formulation

Rewrite the formula:

$$\begin{aligned}
\widehat{\theta}(k) &= \frac{1}{k}[y(1) + \ldots + y(k)] \\
&= \frac{1}{k}[(k-1)\frac{1}{k-1}(y(1) + \ldots + y(k-1)) + y(k)] \\
&= \frac{1}{k}[(k-1)\widehat{\theta}(k-1) + y(k)]
\end{aligned}$$

(already a recursive formula, but we go on to gain more intuition)

$$\begin{aligned}
&= \frac{1}{k}[k\widehat{\theta}(k-1) + y(k) - \widehat{\theta}(k-1)] \\
&= \widehat{\theta}(k-1) + \frac{1}{k}[y(k) - \widehat{\theta}(k-1)]
\end{aligned}$$

Estimating a scalar: Features

$$\widehat{\theta}(k) = \widehat{\theta}(k-1) + \frac{1}{k}[y(k) - \widehat{\theta}(k-1)]$$

Method has many of the features of general recursive techniques:

- Recursive formula: new estimate $\widehat{\theta}(k)$ computed based on previous estimate $\widehat{\theta}(k-1)$ and new data $y(k)$.
- $[y(k) - \widehat{\theta}(k-1)]$ is a *prediction error* $\varepsilon(k)$, since $\widehat{\theta}(k-1) = \widehat{b} = \widehat{y}(k)$, a one-step-ahead prediction of the output.
- Update applies a correction proportional to $\varepsilon(k)$, weighted by $\frac{1}{k}$:
  - When the error $\varepsilon(k)$ is large (or small), a large (or small) adjustment is made.
  - The weight $\frac{1}{k}$ decreases with time k, so adjustments get smaller as $\widehat{\theta}$ becomes better.

Table of contents

1 Introduction and motivation

2 Recursive least-squares and ARX
- General recursive least-squares
- Recursive ARX
- Matlab example

3 Recursive instrumental variables

Recall: Least-squares regression

Model:

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

Dataset up to k gives a linear system of equations:

$$\begin{aligned} y(1) &= \varphi^\top(1)\theta \\ y(2) &= \varphi^\top(2)\theta \\ &\dots \\ y(k) &= \varphi^\top(k)\theta \end{aligned}$$

After some linear algebra and calculus, the least-squares solution can be written:

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

Least-squares: Recursive formula

With notation $P(k) = \sum_{j=1}^{k} \varphi(j)\varphi^\top(j)$:

$$
\begin{aligned}
\widehat{\theta}(k) &= P^{-1}(k) \left[\sum_{j=1}^{k} \varphi(j) y(j) \right] \\
&= P^{-1}(k) \left[\sum_{j=1}^{k-1} \varphi(j) y(j) + \varphi(k) y(k) \right] \\
&= P^{-1}(k) \left[P(k-1)\widehat{\theta}(k-1) + \varphi(k) y(k) \right] \\
&= P^{-1}(k) \left[[P(k) - \varphi(k)\varphi^\top(k)]\widehat{\theta}(k-1) + \varphi(k) y(k) \right] \\
&= \widehat{\theta}(k-1) + P^{-1}(k) \left[-\varphi(k)\varphi^\top(k)\widehat{\theta}(k-1) + \varphi(k) y(k) \right] \\
&= \widehat{\theta}(k-1) + P^{-1}(k)\varphi(k) \left[y(k) - \varphi^\top(k)\widehat{\theta}(k-1) \right]
\end{aligned}
$$

Least-squares: Features

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

$$\widehat{\theta}(k) = \widehat{\theta}(k-1) + W(k)\varepsilon(k)$$

- Recursive formula: new estimate $\widehat{\theta}(k)$ computed based on previous estimate $\widehat{\theta}(k-1)$ and new data $y(k)$.
- $\left[y(k) - \varphi^{\top}(k)\widehat{\theta}(k-1)\right]$ is a prediction error $\varepsilon(k)$, since $\varphi^{\top}(k)\widehat{\theta}(k-1) = \widehat{y}(k)$ is the one-step-ahead prediction using the previous parameter vector.
- $W(k) = P^{-1}(k)\varphi(k)$ is a weighting vector: elements of P grow large for large k, therefore $W(k)$ decreases.

Recursive matrix inversion

The previous formula requires matrix inverse $P^{-1}(k)$, which is computationally costly.

For P, an easy recursion exists: $P(k) = P(k-1) + \varphi(k)\varphi^\top(k)$, but this does not help; the matrix must still be inverted.

The Sherman-Morrison formula gives a recursion for the inverse P^{-1}:

$$P^{-1}(k) = P^{-1}(k-1) - \frac{P^{-1}(k-1)\varphi(k)\varphi^\top(k)P^{-1}(k-1)}{1 + \varphi^\top(k)P^{-1}(k-1)\varphi(k)}$$

Exercise: Prove the Sherman-Morrison formula! (linear algebra)

Overall algorithm

Recursive least-squares (RLS)

initialize $\widehat{\theta}(0)$, $P^{-1}(0)$

loop at every step $k = 1, 2, \ldots$

measure $y(k)$, form regressor vector $\varphi(k)$

find prediction error $\varepsilon(k) = y(k) - \varphi^\top(k)\widehat{\theta}(k-1)$

update inverse $P^{-1}(k) = P^{-1}(k-1) - \frac{P^{-1}(k-1)\varphi(k)\varphi^\top(k)P^{-1}(k-1)}{1+\varphi^\top(k)P^{-1}(k-1)\varphi(k)}$

compute weights $W(k) = P^{-1}(k)\varphi(k)$

update parameters $\widehat{\theta}(k) = \widehat{\theta}(k-1) + W(k)\varepsilon(k)$

end loop

Initialization

The RLS algorithm requires initial parameter $\widehat{\theta}(0)$, and initial inverse $P^{-1}(0)$.

Typical choices without prior knowledge:

- $\widehat{\theta}(0) = [0, \ldots, 0]^\top$ - a vector of n zeros.
- $P^{-1}(0) = \frac{1}{\delta} I$, with δ a small number such as 10^{-3}.
  An equivalent initial value of P would be $P(0) = \delta I$.

Intuition: P small corresponds to a low confidence in the initial parameters. This leads to large $P^{-1}(0)$, so initially the weights $W(k)$ are large and large updates are applied to $\widehat{\theta}$.

If a prior value $\widehat{\theta}$ is available, $\widehat{\theta}(0)$ can be initialized to it, and δ correspondingly increased. This means better confidence in the initial estimate, and smaller initial updates.

Sanity check: Recovering the scalar case

$$y(k) = b + e(k) = \varphi(k)\theta + e(k), \text{ where } \varphi(k) = 1, \theta = b$$

Take $\widehat{\theta}(0) = 0$, $P^{-1}(0) \to \infty$. We have with Sherman-Morrison:

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

$$P^{-1}(1) = 1$$

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

$$\dots$$

$$P^{-1}(k) = \frac{1}{k}$$

Also, $\varepsilon(k) = y(k) - \widehat{\theta}(k-1)$ and $W(k) = P^{-1}(k) = \frac{1}{k}$, leading to:

$$\widehat{\theta}(k) = \widehat{\theta}(k-1) + W(k)\varepsilon(k) = \widehat{\theta}(k-1) + \frac{1}{k}[y(k) - \widehat{\theta}(k-1)]$$

$\Rightarrow$ formula for the scalar case was recovered.

Table of contents

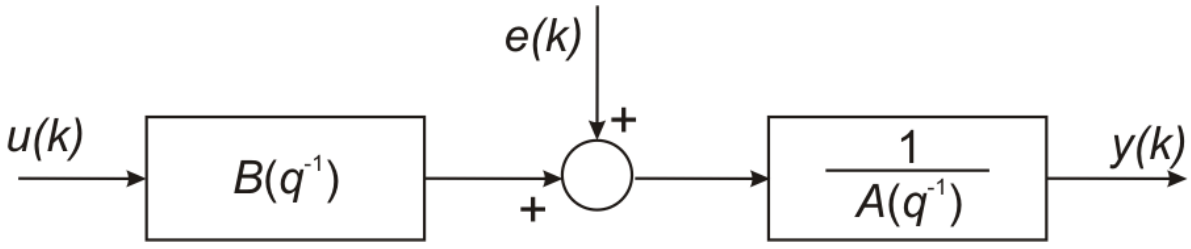
1 Introduction and motivation

2 Recursive least-squares and ARX
- General recursive least-squares
- Recursive ARX
- Matlab example

3 Recursive instrumental variables

Recall: ARX model

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



In explicit form:

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

where the model parameters are: $a_1, a_2, \ldots, a_{n_a}$ and $b_1, b_2, \ldots, b_{nb}$.

Recall: Linear regression representation

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

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

Parameter vector: $\theta \in \mathbb{R}^{na+nb}$, polynomial coefficients.

Recursive ARX

With this representation, just an instantiation of RLS template.

Recursive ARX

initialize $\widehat{\theta}(0)$, $P^{-1}(0)$
loop at every step $k = 1, 2, \ldots$
measure $u(k)$, $y(k)$
form regressor vector
$\varphi(k) = [-y(k-1), \cdots, -y(k-na), u(k-1), \cdots, u(k-nb)]^\top$
find prediction error $\varepsilon(k) = y(k) - \varphi^\top(k)\widehat{\theta}(k-1)$
update inverse $P^{-1}(k) = P^{-1}(k-1) - \frac{P^{-1}(k-1)\varphi(k)\varphi^\top(k)P^{-1}(k-1)}{1+\varphi^\top(k)P^{-1}(k-1)\varphi(k)}$
compute weights $W(k) = P^{-1}(k)\varphi(k)$
update parameters $\widehat{\theta}(k) = \widehat{\theta}(k-1) + W(k)\varepsilon(k)$
end loop

Remark: Outputs more than *na* steps ago, and inputs more than *nb* steps ago, are not used so they can be forgotten, reducing memory usage.

Guarantees idea

With $\widehat{\theta}(0) = 0$, $P^{-1}(0) = \frac{1}{\delta} I$ and $\delta \to 0$, recursive ARX at step k is equivalent to running batch ARX on the dataset $u(1), y(1), \ldots, u(k), y(k)$.

$\Rightarrow$ the same guarantees as batch ARX:

Theorem

Under appropriate assumptions (including the existence of true parameters θ_0), ARX identification is consistent: the estimated parameters $\widehat{\theta}$ tend to the true parameters θ_0 as $k \to \infty$.

The condition on P is to ensure the inverses are the same as in the offline case.

Table of contents

1 Introduction and motivation

2 Recursive least-squares and ARX
- General recursive least-squares
- Recursive ARX
- Matlab example

3 Recursive instrumental variables

System

To illustrate recursive ARX, we take a known system:

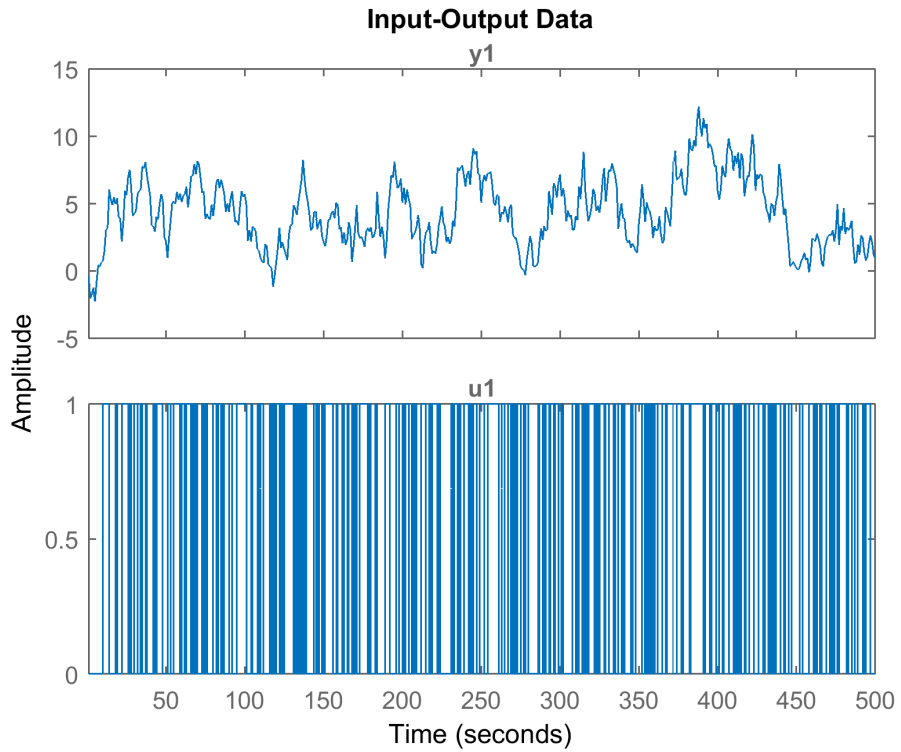
$$y(k) + ay(k-1) = bu(k-1) + e(k), \quad a = -0.9, b = 1$$

(Söderström & Stoica)

```
system = idpoly([1 -0.9], [0 1]);
```

Identification data obtained in simulation:

```
sim(system, u, 'noise');
```



Recursive ARX

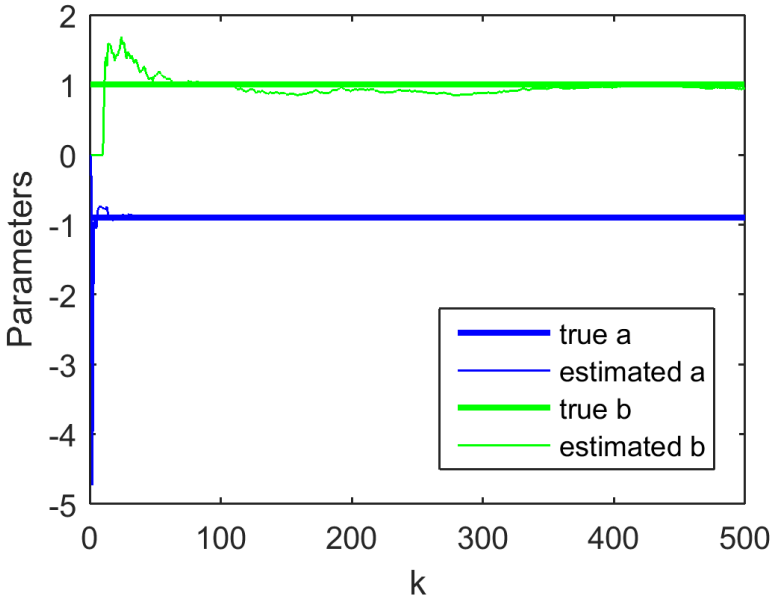
```
model = rarx(id, [na, nb, nk], 'ff', 1, th0, Pinv0);
```

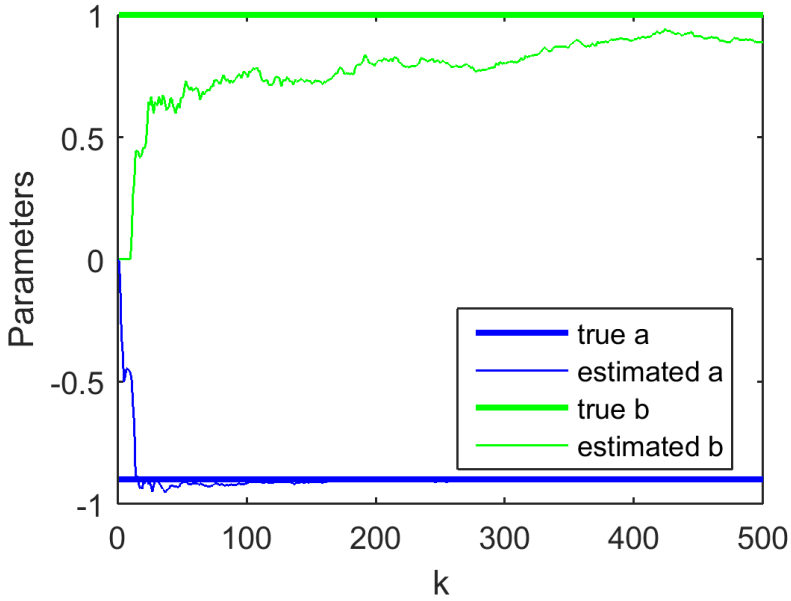
Arguments:

1. Identification data.
2. Array containing the orders of A and B and the delay nk.
3. `'ff', 1` selects the algorithm variant presented in this lecture.
4. `th0` is the initial parameter value.
5. `Pinv0` is the initial inverse $P^{-1}(0)$.

Results

$na = nb = nk = 1$
$P^{-1}(0) = \frac{1}{\delta} I$, left: $\delta = 10^{-3}$, right: $\delta = 100$.
For all the experiments, $\theta_0 = [0, 0]^\top$.





Conclusions:

- Convergence to true parameter values...
- ...slower when δ is larger (in this case, misplaced large confidence in the initial solution)

System outside model class

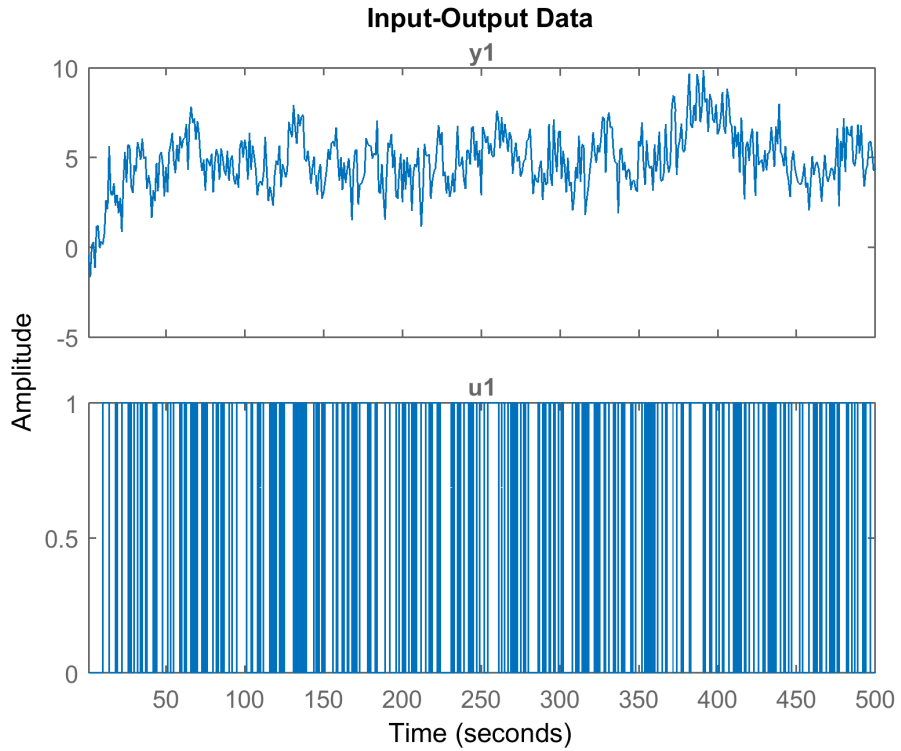
Take now an *output-error* system (so it is not ARX with white noise):

$$y(k) = \frac{bq^{-1}}{1 + fq^{-1}} u(k) + e(k), \quad f = -0.9, b = 1$$

(Söderström & Stoica)

```
system = idpoly([], [0 1], [], [], [1 -.9]);
```

Identification data:

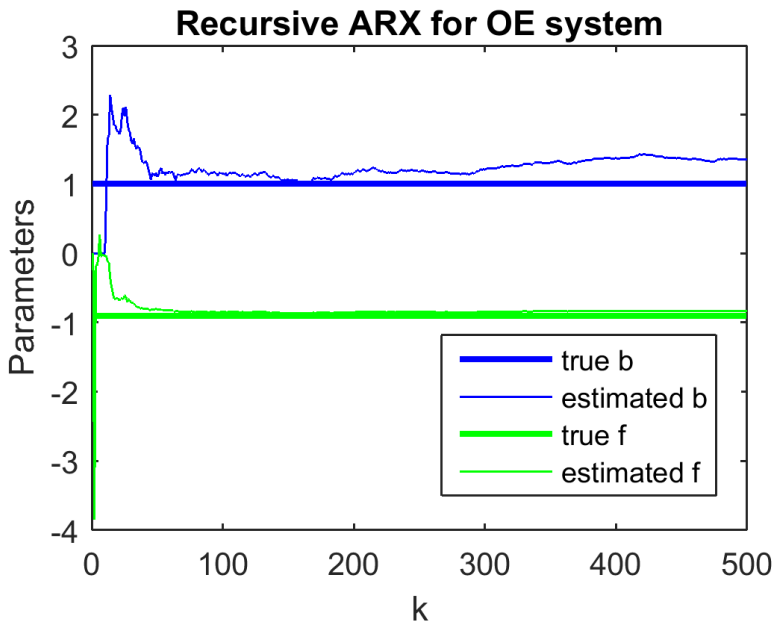


Results

$nf = nb = nk = 1$, so we attempt the ARX model:

$$y(k) + fy(k-1) = bu(k-1) + e(k)$$

Also, $P^{-1}(0) = \frac{1}{\delta} I$, $\delta = 10^{-3}$.



Conclusion: Convergence to the true values is not attained! – due to colored noise (system not in model class)

Other recursive PEM algorithms

Other recursive PEM methods available in Matlab, e.g.:

- ARMAX, `rarmax`
- output-error, `roe`
- generic prediction error method, `rpem`

Table of contents

1 Introduction and motivation

2 Recursive least-squares and ARX

3 Recursive instrumental variables

- Recursive IV method
- Matlab example

Recall: Instrumental variable method

IV methods find the parameter vector using:

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

which is the solution to the system of equations:

$$\left[\frac{1}{N}\sum_{k=1}^{N} Z(k)[\varphi^{\top}(k)\theta - y(k)]\right] = 0$$

Regressor vector:
$\varphi(k) = [-y(k-1), \cdots, -y(k-na), u(k-1), \cdots, u(k-nb)]^{\top}$,
instrument vector usually:
$Z(k) = [-x(k-1), \cdots, -x(k-na), u(k-1), \cdots, u(k-nb)]^{\top}$.

Since instrument vector $Z(k)$ uncorrelated with the noise,
IV methods can handle colored noise.

Recursive formulation (1)

Averaging by the number of data points can be removed, leading to:

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

Averaging was needed in batch IV since summations over many data points could grow very large, leading to numerical problems. Recursive IV will, in the end, add terms one by one, and will therefore be numerically stable.

Rewrite the equation for the recursive case:

$$\widehat{\theta}(k) = P^{-1}(k) \left[\sum_{j=1}^{k} Z(j)y(j)\right]$$

where $P(k) = \sum_{j=1}^{k} Z(j)\varphi^{\top}(j)$.

Recursive formulation (2)

With this definition of P, the recursive update is found similarly to RLS:

$$
\begin{aligned}
\widehat{\theta}(k) &= P^{-1}(k)\left[\sum_{j=1}^{k} Z(j)y(j)\right] \\
&= P^{-1}(k)\left[\sum_{j=1}^{k-1} Z(j)y(j) + Z(k)y(k)\right] \\
&= P^{-1}(k)\left[P(k-1)\widehat{\theta}(k-1) + Z(k)y(k)\right] \\
&= P^{-1}(k)\left[[P(k) - Z(k)\varphi^{\top}(k)]\widehat{\theta}(k-1) + Z(k)y(k)\right] \\
&= \widehat{\theta}(k-1) + P^{-1}(k)\left[-Z(k)\varphi^{\top}(k)\widehat{\theta}(k-1) + Z(k)y(k)\right] \\
&= \widehat{\theta}(k-1) + P^{-1}(k)Z(k)\left[y(k) - \varphi^{\top}(k)\widehat{\theta}(k-1)\right]
\end{aligned}
$$

Recursive formulation (3)

Final formula:

$$\boxed{\widehat{\theta}(k) = \widehat{\theta}(k-1) + P^{-1}(k)Z(k)\left[y(k) - \varphi^{\top}(k)\widehat{\theta}(k-1)\right]}$$

$$\widehat{\theta}(k) = \widehat{\theta}(k-1) + W(k)\varepsilon(k)$$

with the Sherman-Morrison update of the inverse:

$$P^{-1}(k) = P^{-1}(k-1) - \frac{P^{-1}(k-1)Z(k)\varphi^{\top}(k)P^{-1}(k-1)}{1 + \varphi^{\top}(k)P^{-1}(k-1)Z(k)}$$

Recursive IV: Overall algorithm

Recursive IV

initialize $\widehat{\theta}(0)$, $P^{-1}(0)$

loop at every step $k = 1, 2, \ldots$

measure $y(k)$, form regressor vector $\varphi(k)$ and instrument vector $Z(k) = [-x(k-1), \cdots, -x(k-na), u(k-1), \cdots, u(k-nb)]^\top$

find prediction error $\varepsilon(k) = y(k) - \varphi^\top(k)\widehat{\theta}(k-1)$

update inverse $P^{-1}(k) = P^{-1}(k-1) - \frac{P^{-1}(k-1)Z(k)\varphi^\top(k)P^{-1}(k-1)}{1+\varphi^\top(k)P^{-1}(k-1)Z(k)}$

compute weights $W(k) = P^{-1}(k)Z(k)$

update parameters $\widehat{\theta}(k) = \widehat{\theta}(k-1) + W(k)\varepsilon(k)$

end loop

Table of contents

1 Introduction and motivation

2 Recursive least-squares and ARX

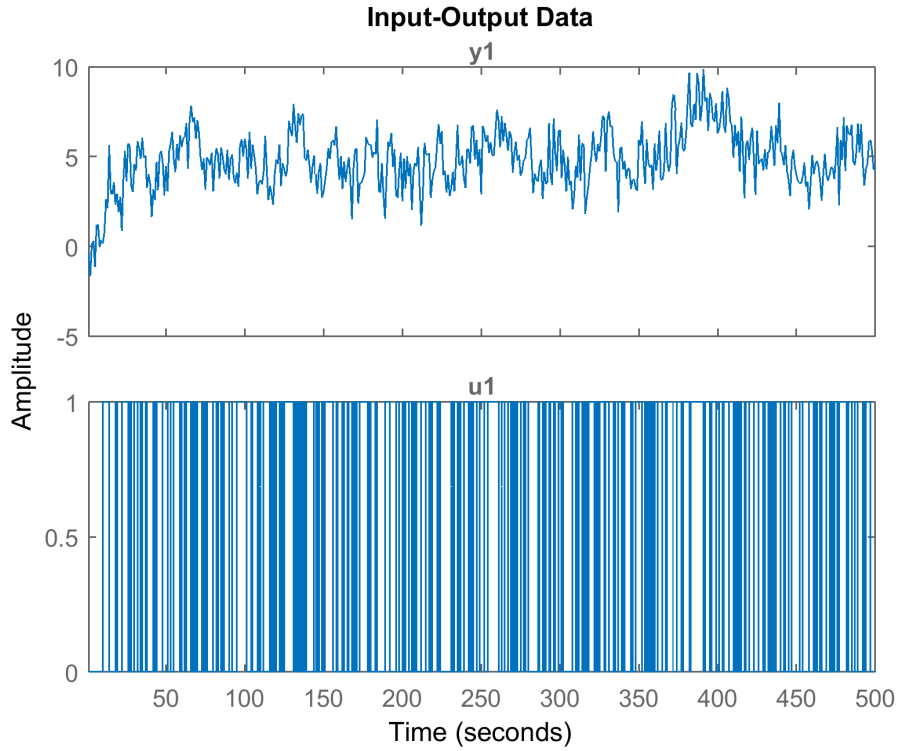
3 Recursive instrumental variables
- Recursive IV method
- Matlab example

OE system

Take the same OE system we used in the ARX example:

$$y(k) = \frac{bq^{-1}}{1 + fq^{-1}} u(k) + e(k), \quad f = -0.9, b = 1$$

with the same dataset:

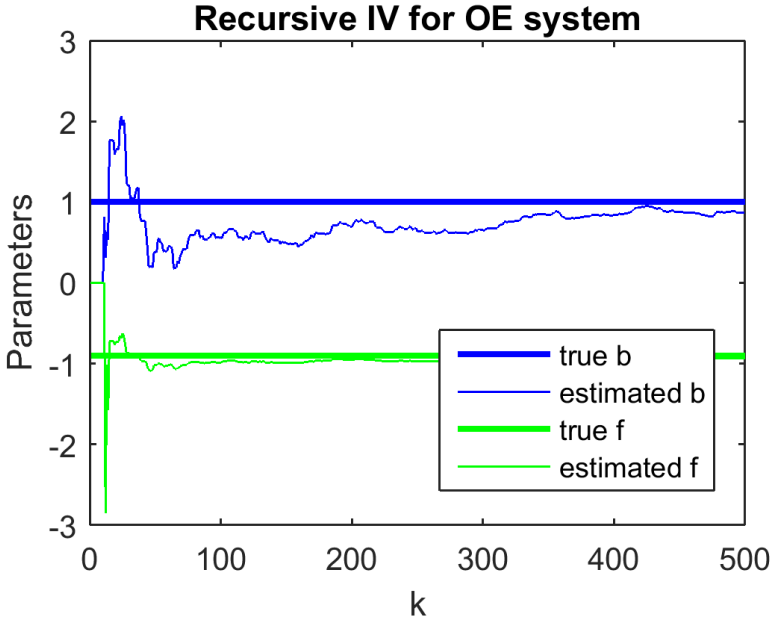


Results with recursive IV

We use the IV model with colored noise $v(k)$:

$$y(k) + fy(k-1) = bu(k-1) + v(k)$$

Instrument vector: $Z(k) = [u(k-2), u(k-1)]$.
As before, $P^{-1}(0) = \frac{1}{\delta} I$, $\delta = 10^{-3}$.



Conclusion: Better than recursive ARX