14. Recursive Identification Methods

- introduction
- recursive least-squares method
- recursive instrumental variable method
- recursive prediction error method

Introduction

features of recursive (online) identification

- $\hat{\theta}(t)$ is computed by some 'simple modification' of $\hat{\theta}(t-1)$
- used in central part of adaptive systems
- not all data are stored, so a small requirement on memory
- easily modified into real-time algorithms
- used in fault detection, to find out if the system has changed significantly

How to estimate time-varying parameters

- update the model regularly
- make use of previous calculations in an efficient manner
- the basic procedure is to modify the corresponding off-line method

Desirable properties of recursive algorithms

- fast convergence
- consistent estimates (time-invariant case)
- good tracking (time-varying case)
- computationally simple

Trade-offs

- convergence vs tracking
- computational complexity vs accuracy

Recursive least-squares method (RLS)

Recursive estimation of a constant: Consider the model

$$y(t) = b + \nu(t), \quad \nu(t) \text{ is a disturbance of variance } \lambda^2$$

the least-squares estimate of b is the arithmetic mean:

$$\hat{\theta}(t) = \frac{1}{t}\sum_{k=1}^{t} y(k)$$

this expression can be reformulated as

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{1}{t}[y(t) - \hat{\theta}(t-1)]$$

- the current estimate is equal to the previous estimate plus a correction
- the correction term is the deviation of the predicted value from what is actually observed

RLS algorithm for a general linear model

$$y(t) = H(t)\theta + \nu(t)$$

The recursive least-squares algorithm is given by

$$\begin{aligned}
e(t) &= y(t) - H(t)\hat{\theta}(t-1) \\
P(t) &= P(t-1) - P(t-1)H^T(t)[I + H(t)P(t-1)H(t)^T]^{-1}H(t)P(t-1) \\
K(t) &= P(t)H(t)^T \quad = \quad P(t-1)H(t)^T[I + H(t)P(t-1)H(t)^T]^{-1} \\
\hat{\theta}(t) &= \hat{\theta}(t-1) + K(t)e(t)
\end{aligned}$$

- interprete $e(t)$ as a prediction error and $K(t)$ as a gain factor
- the update rule in $P(t)$ has an efficient matrix inversion for scalar case

Proof of the update formula the least-square estimate is given by

$$\hat{\theta}(t) = \left(\sum_{k=1}^{t} H(k)^T H(k)\right)^{-1} \left(\sum_{k=1}^{t} H(k)^T y(k)\right)$$

denote $P(t)$ as

$$P(t) = \left(\sum_{k=1}^{t} H(k)^T H(k)\right)^{-1} \quad \Longrightarrow \quad P^{-1}(t) = P^{-1}(t-1) + H(t)^T H(t)$$

then it follows that

$$\begin{aligned} \hat{\theta}(t) &= P(t) \left[\sum_{k=1}^{t-1} H(k)^T y(k) + H(t)^T y(t)\right] \\ &= P(t) \left[P^{-1}(t-1)\hat{\theta}(t-1) + H(t)^T y(t)\right] \end{aligned}$$

$$
\begin{aligned}
\hat{\theta}(t) &= P(t)\left[(P^{-1}(t) - H(t)^T H(t))\hat{\theta}(t-1) + H(t)^T y(t)\right] \\
&= \hat{\theta}(t-1) + P(t)H(t)^T\left[y(t) - H(t)\hat{\theta}(t-1)\right]
\end{aligned}
$$

to obtain the update rule for $P(t)$, we apply the matrix inversion lemma:

$$
(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}
$$

to

$$
P^{-1}(t) = P^{-1}(t-1) + H(t)^T H(t)
$$

where we use

$$
A = P^{-1}(t-1), \quad B = H(t)^T, \quad C = I \quad D = H(t)
$$

Initial conditions

- $\hat{\theta}(0)$ is the initial parameter estimate
- $P(0)$ is an estimate of the covariance matrix of the initial parameter
- if $P(0)$ is small then $K(t)$ will be small and $\hat{\theta}(t)$ will not change much
- if $P(0)$ is large, $\hat{\theta}(t)$ will quickly jump away from $\hat{\theta}(0)$
- it is common in practice to choose

$$\hat{\theta}(0) = 0, \quad P(0) = \rho I$$

  where ρ is a constant
- using a large ρ is good if the initial estimate $\hat{\theta}(0)$ is uncertain

Effect of the initial values

we simulate the following system

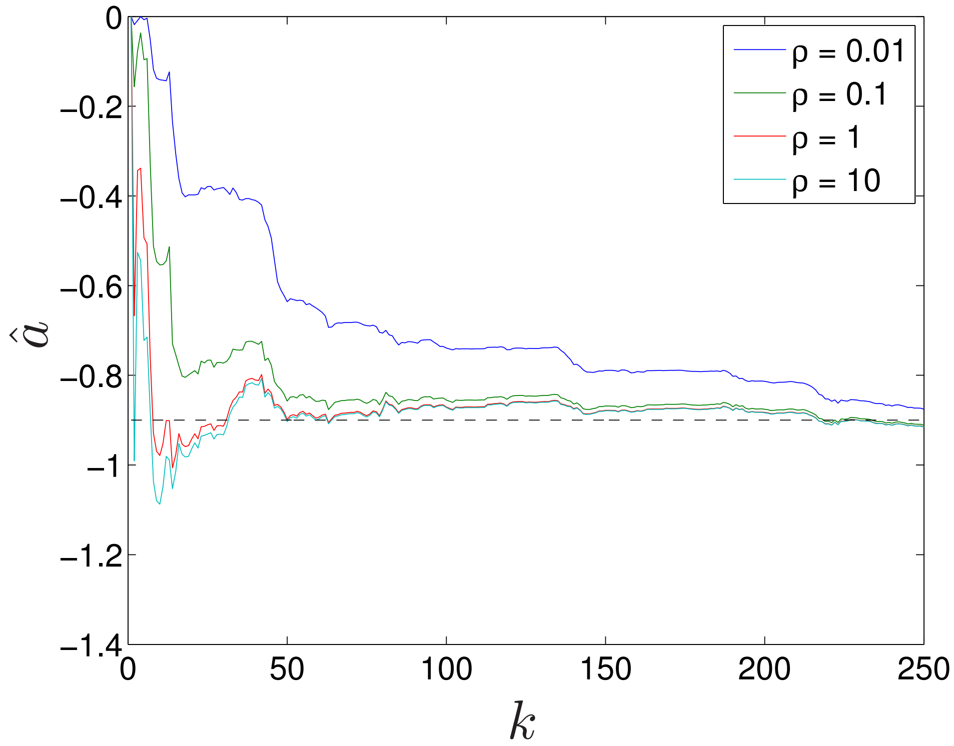
$$y(t) - 0.9y(t-1) = 1.0u(t-1) + \nu(t)$$

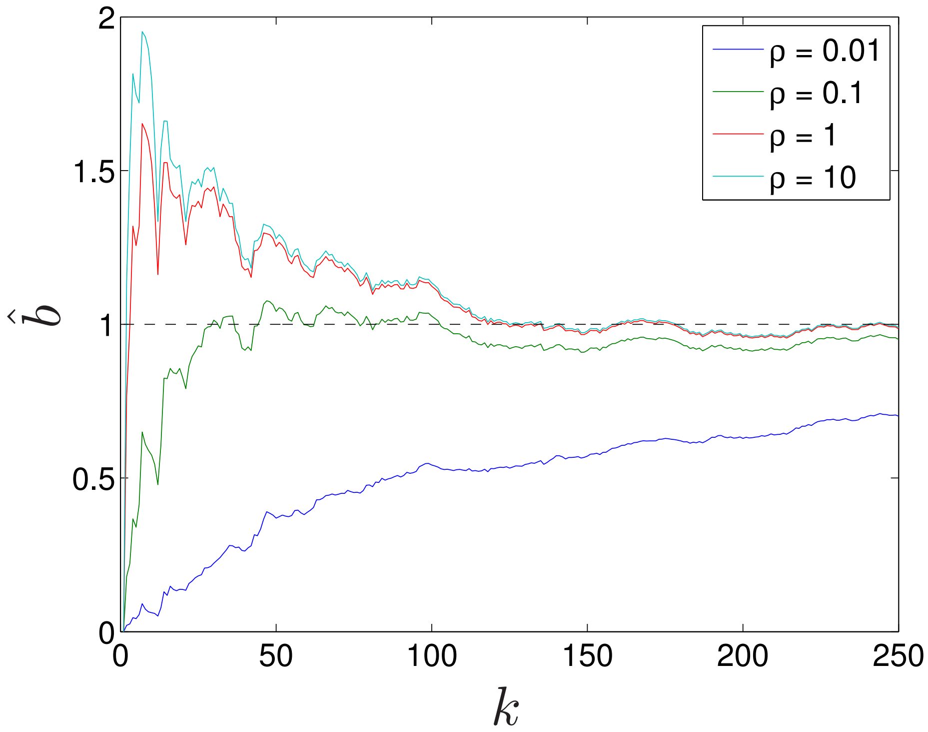
- $u(t)$ is binary white noise
- $\nu(t)$ is white noise of zero mean and variance 1
- identify the system using RLS with 250 points of data
- the parameters are initialized by

$$\hat{\theta}(0) = 0, \quad P(0) = \rho \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

for $\rho = 0.01, 0.1, 1, 10$

the graphs show the influence of the initial values





- large and moderate values of ρ ($i.e.$, $\rho = 1, 10$) lead to similar results
- for large ρ, little confidence is given to $\hat{\theta}(0)$, so quick transient response
- a small value of ρ leads to a small $K(t)$, so it gives a slower convergence

Forgetting factor

the loss function in the least-squares method is modified as

$$f(\theta) = \sum_{k=1}^{t} \lambda^{t-k} \|y(k) - H(k)\theta\|_2^2$$

- λ is called **the forgetting factor** and take values in $(0, 1)$
- the smaller the value of λ, the quicker the previous info will be forgotten
- the parameters are adapted to describe the newest data

Update rule for RLS with a forgetting factor

$$P(t) = \frac{1}{\lambda}\left\{P(t-1) - P(t-1)H(t)^T[\lambda I + H(t)P(t-1)H(t)^T]^{-1}H(t)P(t-1)\right\}$$

$$K(t) = P(t)H(t)^T = P(t-1)H(t)^T[\lambda I + H(t)P(t-1)H(t)^T]^{-1}$$

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)[y(t) - H(t)\hat{\theta}(t-1)]$$

the solution $\hat{\theta}(t)$ that minimizes $f(\theta)$ is given by

$$\hat{\theta}(t) = \left(\sum_{k=1}^{t} \lambda^{t-k} H(k)^T H(k) \right)^{-1} \left(\sum_{k=1}^{t} \lambda^{t-k} H(k)^T y(k) \right)$$

the update formula follow analogously to RLS by introducing

$$P(t) = \left(\sum_{k=1}^{t} \lambda^{t-k} H(k)^T H(k) \right)^{-1}$$

the choice of λ is a trade-off between convergence and tracking performance

- λ small $\Longrightarrow$ old data is forgotten fast, hence good tracking
- λ close to 1 $\Longrightarrow$ good convergence and small variances of the estimates

Effect of the forgetting factor

consider the problem of tracking a time-varying system

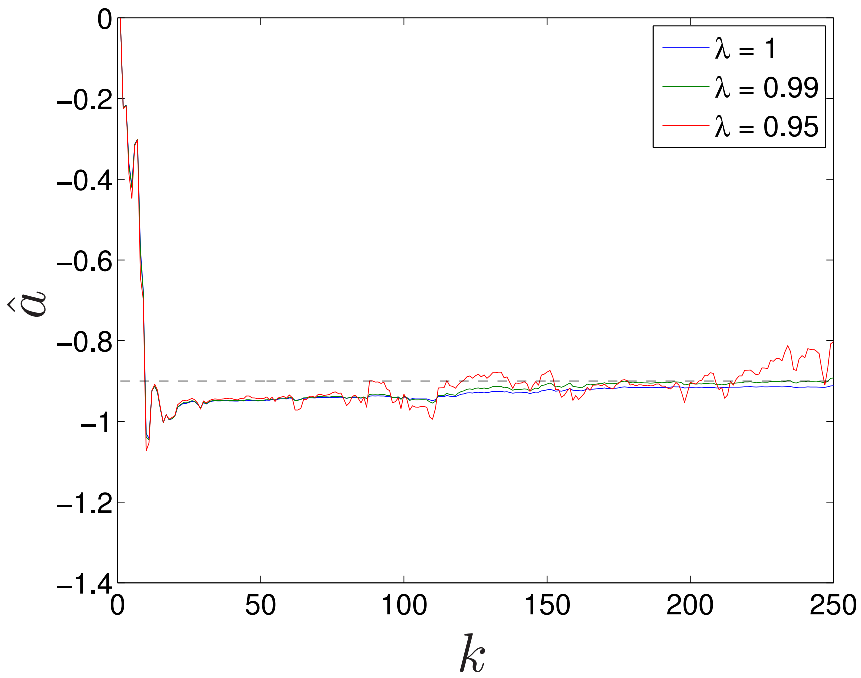
$$y(t) - 0.9y(t-1) = b_0 u(t) + \nu(t), \quad b_0 = \begin{cases} 1.5 & t \leq N/2 \\ 0.5 & t > N/2 \end{cases}$$

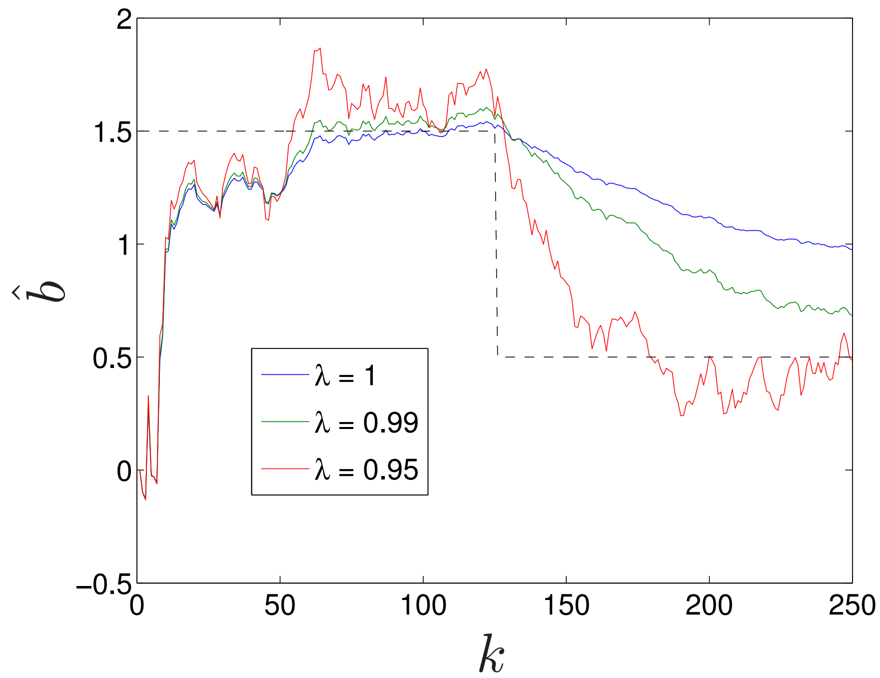
- $u(t)$ is binary white noise
- $\nu(t)$ is white noise of zero mean and variance 1
- identify the system using RLS with 250 points of data
- the parameters are initialized by

$$\hat{\theta}(0) = 0, \quad P(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- the forgetting factors are varied by these values $\lambda = 1, 0.99, 0.95$

graphs show the influence of the forgetting factors





a decrease in the forgetting factor leads to two effects:

- the estimates approach the true value more rapidly
- the algorithm becomes more sensitive to noise

as λ decreases, the oscillations become larger

summary:

- one must have $\lambda = 1$ to get convergence
- if $\lambda < 1$ the parameter estimate can change quickly, and the algorithm becomes more sensitive to noise

for this reason, it is often to allow the forgetting factor to vary with time

a typical choice is to let $\lambda(t)$ tends exponentially to 1

$$\lambda(t) = 1 - \lambda_0^t(1 - \lambda(0))$$

this can be easily implemented via a recursion

$$\lambda(t) = \lambda_0\lambda(t-1) + (1 - \lambda_0)$$

typical values for $\lambda_0 = 0.99$ ($|\lambda_0|$ must be less than 1) and $\lambda(0) = 0.95$

Kalman Filter interpretation

consider a state-space of a time-varying system

$$\begin{aligned} x(t+1) &= A(t)x(t) + Bu(t) + \nu(t) \\ y(t) &= C(t)x(t) + \eta(t) \end{aligned}$$

where $\nu(t), \eta(t)$ are independent white noise with covariances R_1, R_2

Kalman filter:

$$\begin{aligned} \hat{x}(t+1) &= A(t)\hat{x}(t) + B(t)u(t) + K(t)[y(t) - C(t)\hat{x}(t)] \\ K(t) &= A(t)P(t)C(t)^T[C(t)P(t)C(t)^T + R_2]^{-1} \\ P(t+1) &= A(t)P(t)A(t)^T + R_1 \\ &\quad -A(t)P(t)C(t)^T[C(t)P(t)C(t)^T + R_2]^{-1}C(t)P(t)A(t)^T \end{aligned}$$

the linear regression model

$$y(t) = H(t)\theta + \nu(t)$$

can be written as a state-space equation

$$\begin{aligned} \theta(t+1) &= \theta(t) \quad (= \theta) \\ y(t) &= H(t)\theta(t) + \nu(t) \end{aligned}$$

apply the Kalman filter to the state-space equation with

$$A(t) = I, \quad B(t) = 0, \quad C(t) = H(t), \quad R_1 = 0$$

when $R_2 = I$, it will give precisely the basic RLS algorithm in page 14-5

the tracking capability is affected by R_2

Recursive instrument variable method

the IV estimate of a scalar linear system

$$y(t) = H(t)\theta + \nu(t)$$

is given by

$$\hat{\theta}(t) = \left[\sum_{k=1}^{t} Z(k)^T H(k)\right]^{-1} \left[\sum_{k=1}^{t} Z(k)^T y(k)\right]$$

the IV estimate can be computed recursively as

$$\begin{aligned}
\hat{\theta}(t) &= \hat{\theta}(t-1) + K(t)[y(t) - H(t)\hat{\theta}(t-1)] \\
K(t) &= P(t)Z(t)^T \quad = \quad P(t-1)Z(t)^T[I + H(t)P(t-1)Z(t)^T] \\
P(t) &= P(t-1) - P(t-1)Z(t)^T[I + H(t)P(t-1)Z(t)^T]^{-1}H(t)P(t-1)
\end{aligned}$$

(analogous proof to RLS by using $P(t) = (\sum_{k=1}^{t} Z(k)^T H(k))^{-1}$)

Recursive prediction error method

we will use the cost function

$$f(t,\theta) = \frac{1}{2}\sum_{k=1}^{t} \lambda^{t-k} e(k,\theta)^T W e(k,\theta)$$

where $W \succ 0$ is a weighting matrix

- for $\lambda = 1$, $f(\theta) = \mathbf{tr}(WR(\theta))$ where $R(\theta) = \frac{1}{2}\sum_{k=1}^{t} e(k,\theta)e(k,\theta)^T$
- the off-line estimate of $\hat{\theta}$ cannot be found analytically (except for the LS case)
- it is *not* possible to derive an exact recursive algorithm
- some approximation must be used, and they hold exactly for the LS case

main idea: assume that

- $\hat{\theta}(t-1)$ minimizes $f(t-1,\theta)$
- the minimum point of $f(t,\theta)$ is close to $\hat{\theta}(t-1)$

using a second-order Taylor series approximation around $\hat{\theta}(t-1)$ gives

$$\begin{aligned} f(t,\theta) \approx f(t,\hat{\theta}(t-1)) + \nabla f(t,\hat{\theta}(t-1))^T(\theta-\hat{\theta}(t-1)) \\ + \frac{1}{2}[\theta-\hat{\theta}(t-1)]^T\nabla^2 f(t,\hat{\theta}(t-1))[\theta-\hat{\theta}(t-1)] \end{aligned}$$

minimize the RHS w.r.t. θ and let the minimizer be $\hat{\theta}(t)$:

$$\hat{\theta}(t) = \hat{\theta}(t-1) - [\nabla^2 f(t,\hat{\theta}(t-1))]^{-1}\nabla f(t,\hat{\theta}(t-1))$$

(Newton-Raphson step)

we must find $\nabla f(t,\hat{\theta}(t-1))$ and $P(t) = [\nabla^2 f(t,\hat{\theta}(t-1))]^{-1}$

details: to proceed, the gradients of $f(t,\theta)$ w.r.t θ are needed

$$
\begin{aligned}
f(t,\theta) &= \lambda f(t-1,\theta) + \frac{1}{2}e(t,\theta)^T W e(t,\theta) \\
\nabla f(t,\theta) &= \lambda \nabla f(t-1,\theta) + e(t,\theta)^T W \nabla e(t,\theta) \\
\nabla^2 f(t,\theta) &= \lambda \nabla^2 f(t-1,\theta) + \nabla e(t,\theta)^T W \nabla e(t,\theta) + e(t,\theta)^T W \nabla^2 e(t,\theta)
\end{aligned}
$$

first approximations:

- $\nabla f(t-1,\hat{\theta}(t-1)) = 0$ ($\hat{\theta}(t-1)$ minimizes $f(t-1,\theta)$)
- $\nabla^2 f(t-1,\hat{\theta}(t-1)) = \nabla^2 f(t-1,\hat{\theta}(t-2))$ ($\nabla^2 f$ varies slowly with θ)
- $e(t,\theta)^T W \nabla^2 e(t,\theta)$ is negligible

after inserting the above equations to

$$
\hat{\theta}(t) = \hat{\theta}(t-1) - [\nabla^2 f(t,\hat{\theta}(t-1))]^{-1} \nabla f(t,\hat{\theta}(t-1))
$$

we will have

$$\hat{\theta}(t) = \hat{\theta}(t-1) - [\nabla^2 f(t, \hat{\theta}(t-1)]^{-1}[e(t, \hat{\theta}(t-1))^T W \nabla e(t, \hat{\theta})(t-1)]$$
$$\nabla^2 f(t, \hat{\theta}(t-1)) = \lambda \nabla^2 f(t-1, \hat{\theta}(t-2)) + \nabla e(t, \hat{\theta}(t-1))^T W \nabla e(t, \hat{\theta}(t-1))$$

(still not suited well as an online algorithm due to the term $e(t, \hat{\theta}(t-1))$)

second approximations: let

$$e(t) \approx e(t, \hat{\theta}(t-1)), \quad H(t) \approx -\nabla e(t, \hat{\theta}(t-1))$$

(the actual way of computing these depends on model structures), then

$$\hat{\theta}(t) = \hat{\theta}(t-1) + P(t)H^T(t)We(t)$$

where we denote $P(t) = [\nabla^2 f(t, \hat{\theta}(t-1))]^{-1}$ which satisfies

$$P^{-1}(t) = \lambda P^{-1}(t-1) + H(t)^T W H(t)$$

apply the matrix inversion lemma to the recursive formula of $P^{-1}(t)$

we arrive at recursive prediction error method (RPEM)

algorithm:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)e(t)$$
$$K(t) = P(t)H(t)^T$$
$$P(t) = \frac{1}{\lambda}\left\{P(t-1) - P(t-1)H(t)^T[\lambda W^{-1} + H(t)P(t-1)H(t)^T]^{-1}P(t-1)\right\}$$

where the approximations

$$e(t) \approx e(t, \hat{\theta}(t-1)), \quad H(t) \approx -\nabla e(t, \hat{\theta}(t-1))$$

depend on the model structure

Example of RPEM: ARMAX models

consider the scalar ARMAX model

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

where all the polynomials have the same order

$$\begin{aligned}
A(q^{-1}) &= 1 + a_1 q^{-1} + \cdots + a_n q^{-n} \\
B(q^{-1}) &= b_1 q^{-1} + \cdots + b_n q^{-n} \\
C(q^{-1}) &= 1 + c_1 q^{-1} + \cdots + c_n q^{-n}
\end{aligned}$$

define

$$\tilde{y}(t,\theta) = \frac{1}{C(q^{-1})}y(t), \quad \tilde{u}(t,\theta) = \frac{1}{C(q^{-1})}u(t), \quad \tilde{e}(t,\theta) = \frac{1}{C(q^{-1})}e(t)$$

we can derive the following relations

$$\begin{aligned} e(t,\theta) &= \frac{A(q^{-1})y(t) - B(q^{-1})u(t)}{C(q^{-1})} \\ \nabla e(t,\theta) &= (\tilde{y}(t-1,\theta),\ldots,\tilde{y}(t-n,\theta), -\tilde{u}(t-1,\theta),\ldots,-\tilde{u}(t-n,\theta), \\ &\qquad -\tilde{e}(t-1,\theta),\ldots,-\tilde{e}(t-n,\theta)) \end{aligned}$$

to compute $e(t,\theta)$, we need to process all data up to time t

we use the following approximations

$$
\begin{aligned}
e(t,\theta) \approx e(t) = y(t) + \hat{a}_1(t-1)y(t-1) + \cdots + \hat{a}_n(t-1)y(t-n) \\
- \hat{b}_1(t-1)u(t-1) - \cdots - \hat{b}_n(t-1)u(t-n) \\
- \hat{c}_1(t-1)e(t-1) - \cdots - \hat{c}_n(t-1)e(t-n)
\end{aligned}
$$

$$
\begin{aligned}
-\nabla e(t,\theta) \approx H(t) = (-\bar{y}(t-1), \ldots, -\bar{y}(t-n), \\
\bar{u}(t-1), \ldots, \bar{u}(t-n), \bar{e}(t-1), \ldots, \bar{e}(t-n))
\end{aligned}
$$

where

$$
\begin{aligned}
\bar{y}(t) &= y(t) - \hat{c}_1(t)\bar{y}(t-1) - \cdots - \hat{c}_n(t)\bar{y}(t-n) \\
\bar{u}(t) &= u(t) - \hat{c}_1(t)\bar{u}(t-1) - \cdots - \hat{c}_n(t)\bar{u}(t-n) \\
\bar{e}(t) &= e(t) - \hat{c}_1(t)\bar{e}(t-1) - \cdots - \hat{c}_n(t)\bar{e}(t-n)
\end{aligned}
$$

Comparison of recursive algorithms

we simulate the following system

$$y(t) = \frac{1.0q^{-1}}{1 - 0.9q^{-1}} u(t) + \nu(t)$$

- $u(t), \nu(t)$ are indepentdent white noise with zero mean and variance 1
- we use RLS,RIV, RPEM to identify the system

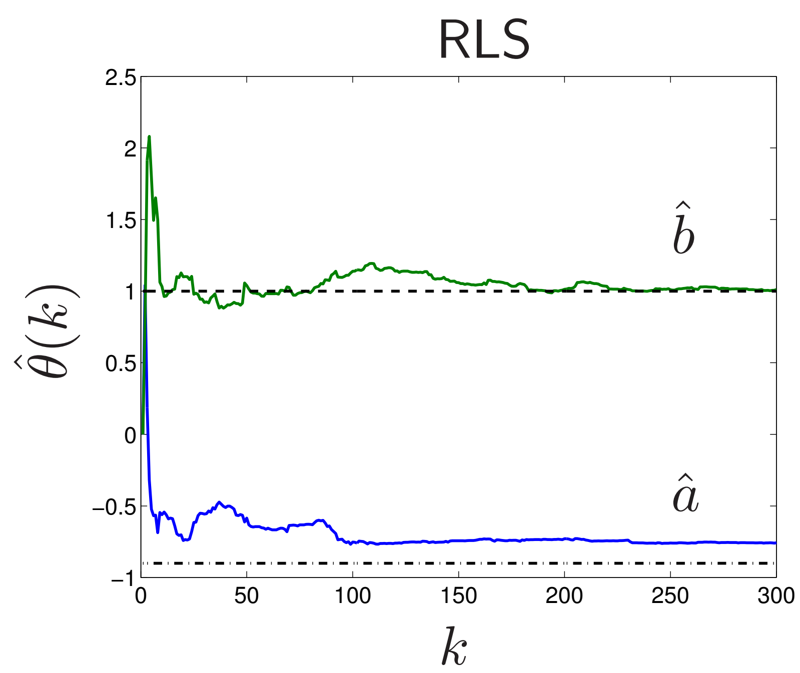
model structure for RLS and RIV:

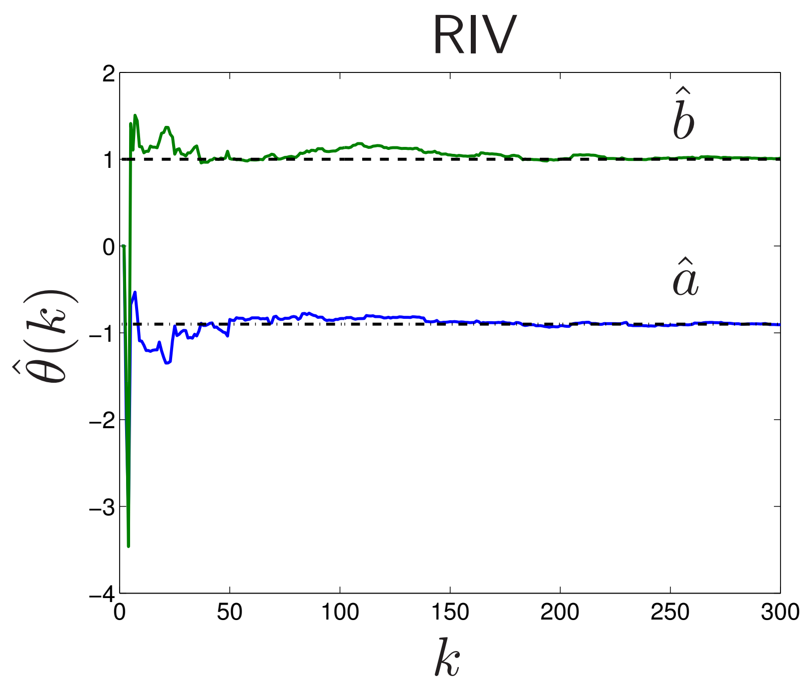
$$y(t) + ay(t-1) = bu(t-1) + \nu(t), \quad \theta = (a, b)$$

model structure for RPEM:

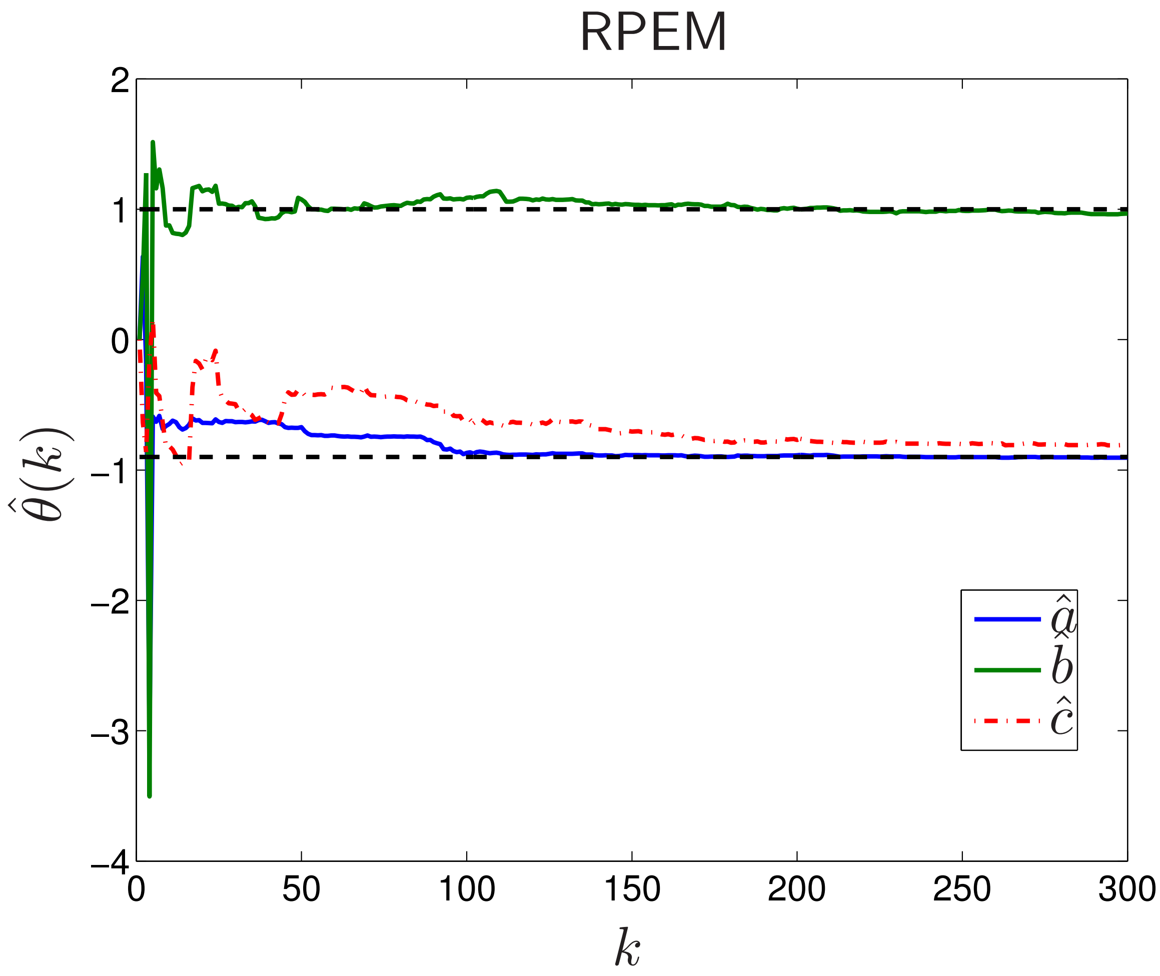
$$y(t) + ay(t-1) = bu(t-1) + \nu(t) + c\nu(t-1), \quad \theta = (a, b, c)$$

Numerical results





- RLS does not give consistent estimates for systems with correlated noise
- this is because RLS is equivalent to an off-line LS algorithm
- in contrast to RLS, RIV gives consistent estimates
- this result follows from that RIV is equivalent to an off-line IV method



- RPEM gives consistent estimates of a, b, c
- the estimates $\hat{a}$ and $\hat{b}$ converge more quickly than $\hat{c}$

Common problems for recursive identification

- excitation
- estimator windup
- $P(t)$ becomes indefinite

excitation it is important that the input is persistently excitation of sufficiently high order

Estimator windup

some periods of an identification experiment exhibit poor excitation

consider when $H(t) = 0$ in the RLS algorithm, then

$$\hat{\theta}(t) = \hat{\theta}(t-1), \quad P(t) = \frac{1}{\lambda}P(t-1)$$

- $\hat{\theta}$ becomes constant as t increases
- P increases exponentially with time for $\lambda < 1$
- when the system is excited again ($H(t) \neq 0$), the gain

$$K(t) = P(t)H(t)^T$$

  will be very large and causes an abrupt change in $\hat{\theta}$

- this is referred to as *estimator windup*

Solution: do not update $P(t)$ if we have poor excitation

Indefinite $P(t)$

$P(t)$ represents a covariance matrix

therefore, it must be symmetric and positive definite

rounding error may accumulate and make $P(t)$ indefinite

this will make the estimate diverge

the solution is to note that any positive definite matrix can be factorized as

$$P(t) = S(t)S(t)^T$$

and rewrite the algorithm to update $S(t)$ instead

References

Chapter 9 in
T. Söderström and P. Stoica, *System Identification*, Prentice Hall, 1989

Chapter 11 in
L. Ljung, *System Identification: Theory for the User*, 2nd edition, Prentice Hall, 1999

Lecture on
Recursive Identification Methods, System Identification (1TT875), Uppsala University, `http://www.it.uu.se/edu/course/homepage/systemid/vt05`