13. 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

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

- 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

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

Denote $P(t)$ as

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

Then it follows that

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

$$\hat{\theta}(t) = P(t)\left[(P^{-1}(t) - H(t)^*H(t))\hat{\theta}(t-1) + H(t)^*y(t)\right]$$
$$= \hat{\theta}(t-1) + P(t)H(t)^*\left[y(t) - H(t)\hat{\theta}(t-1)\right]$$

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

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

where we use

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

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

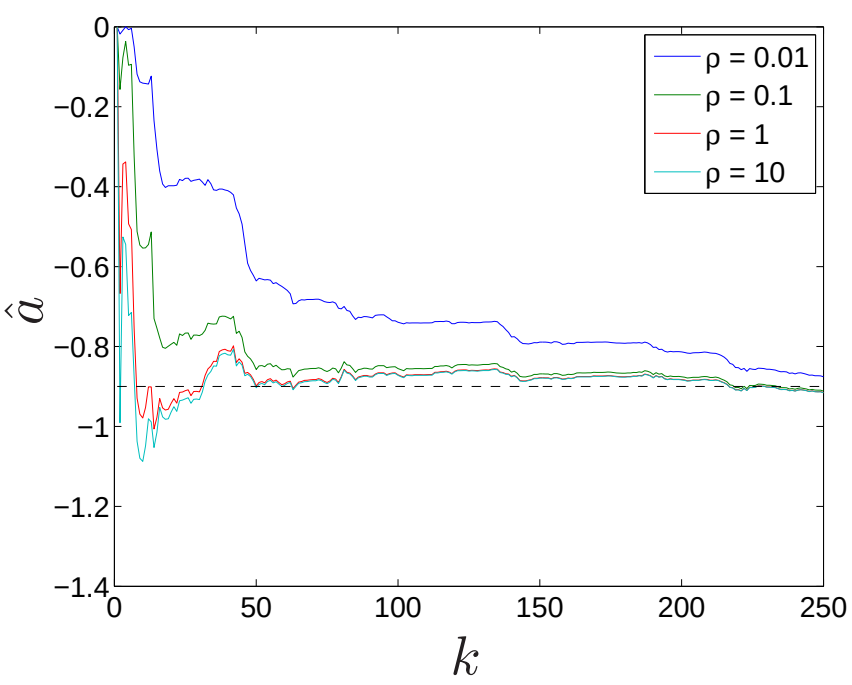
$$y(t) - 0.9y(t-1) = 1.0u(t-1) + \nu(t)$$

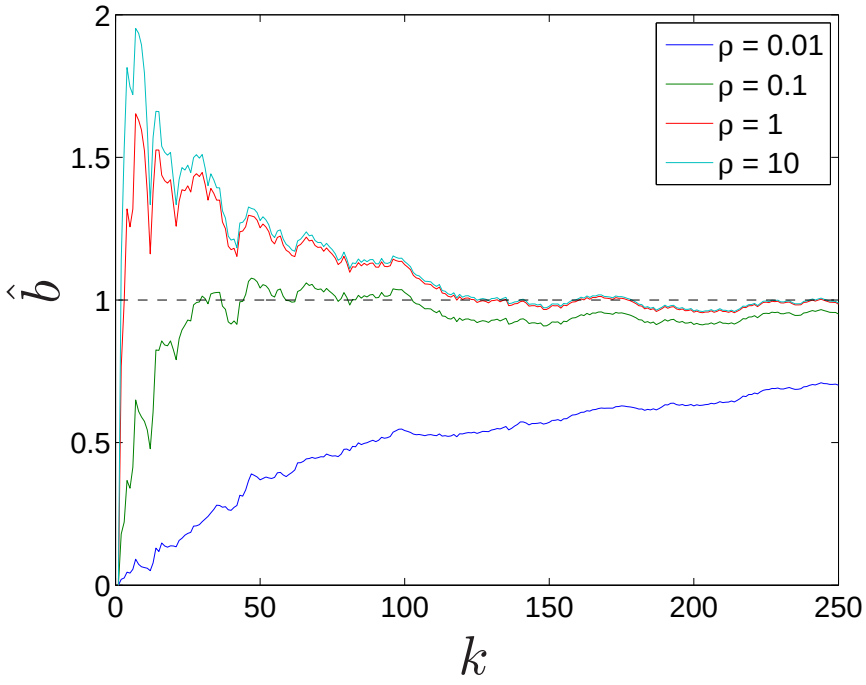
- $u(t)$ is binary white noise
- $\nu(t)$ is white noise of zero mean and variance 1
- Identify the system using RLS with 250 points of data
- The parameters are initialized by

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

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

The graphs show the influence of the initial values





- Large and moderate values of ρ ($i.e.$, $\rho = 1$ and $\rho = 10$) 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 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

The RLS method with a forgetting factor is

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

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

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

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

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

The update formula follow analogously to RLS by introducing

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

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

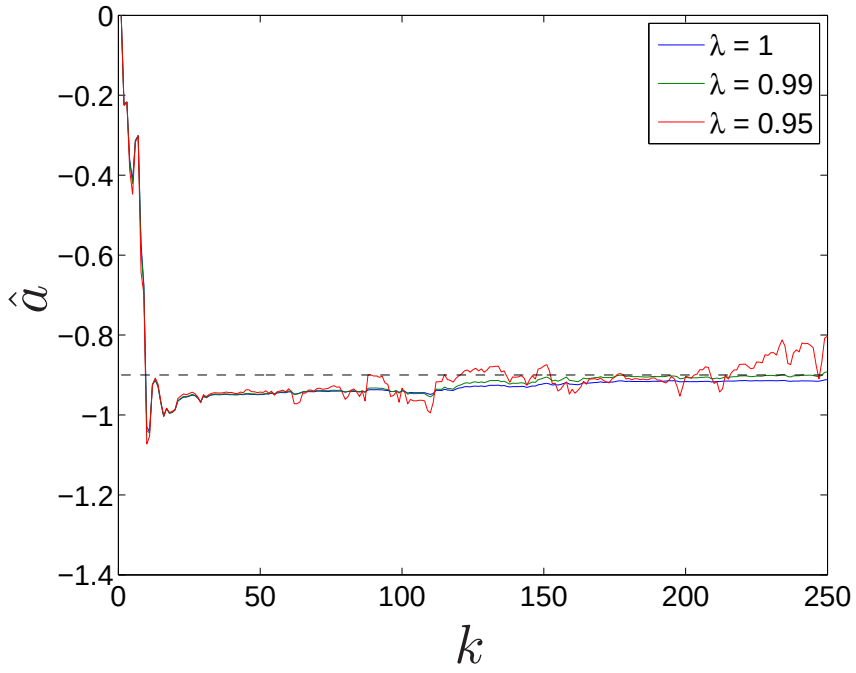
$$y(t) - 0.9y(t-1) = b_0 u(t) + \nu(t), \quad b_0 = \begin{cases} 1.5 & t \leq N/2 \\ 0.5 & t > N/2 \end{cases}$$

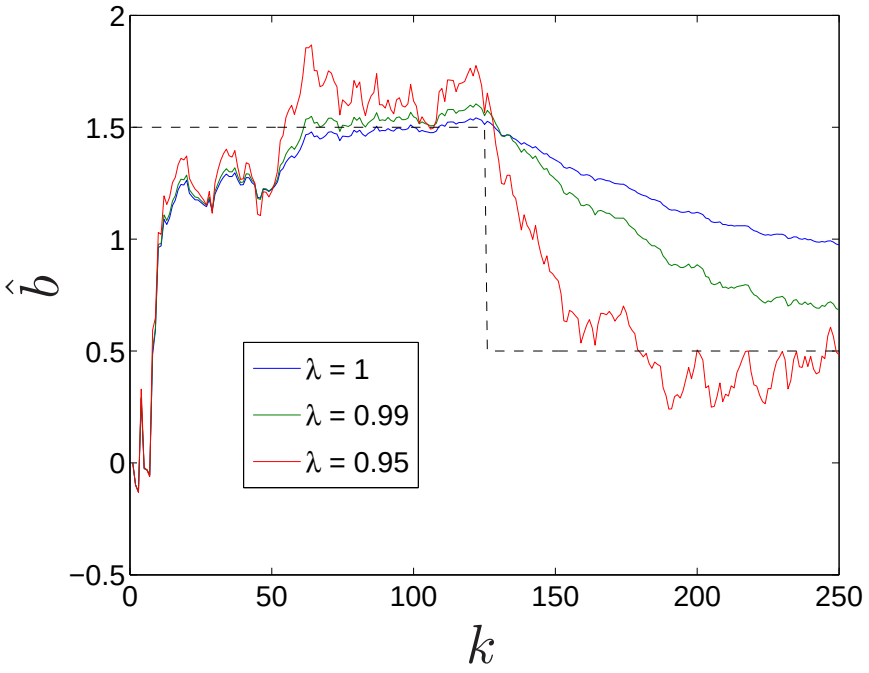
- $u(t)$ is binary white noise
- $\nu(t)$ is white noise of zero mean and variance 1
- Identify the system using RLS with 250 points of data
- The parameters are initialized by

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

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

The graphs show the influence of the forgetting factors





A decrease in the forgetting factor leads to two effects:

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

As λ decreases, the oscillations become larger

In 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 typically choice is to let $\lambda(t)$ tend exponentially to 1

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

which can be easily implemented via a recursion

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

Typical values for $\lambda_0 = 0.99$ and $\lambda(0) = 0.95$

Kalman Filter interpretation

The Kalman filter for estimating the state 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, resp., is given by

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

The linear regression model

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

can be written as a state-space equation

$$\theta(t+1) = \theta(t) \quad (= \theta)$$
$$y(t) = H(t)\theta(t) + \nu(t)$$

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 13-5

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

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

The IV estimate can be computed recursively as

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

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

Recursive prediction error method

We will use the cost function

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

where $W \succ 0$ is a weighting matrix

- For $\lambda = 1$, $f(\theta) = \mathbf{tr}(W R(\theta))$ where $R(\theta) = \frac{1}{2}\sum_{k=1}^{t} e(k,\theta)e^*(k,\theta)$
- 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

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

Minimize this 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 gradient of $f(t,\theta)$ w.r.t θ are needed

$$f(t,\theta) = \lambda f(t-1,\theta) + \frac{1}{2}e(t,\theta)^* W e(t,\theta)$$
$$\nabla f(t,\theta) = \lambda \nabla f(t-1,\theta) + e(t,\theta)^* W \nabla e(t,\theta)$$
$$\nabla^2 f(t,\theta) = \lambda \nabla^2 f(t-1,\theta) + \nabla e(t,\theta)^* W \nabla e(t,\theta) + e(t,\theta)^* W \nabla^2 e(t,\theta)$$

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)
- $e(t,\theta)^* W \nabla^2 e(t,\theta)$ 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

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

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

(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

$$\hat{\theta}(t) = \hat{\theta}(t-1) + P(t) H^*(t) W e(t)$$

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

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

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

we arrive at recursive prediction error method (RPEM)

Algorithm:

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

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

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

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

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

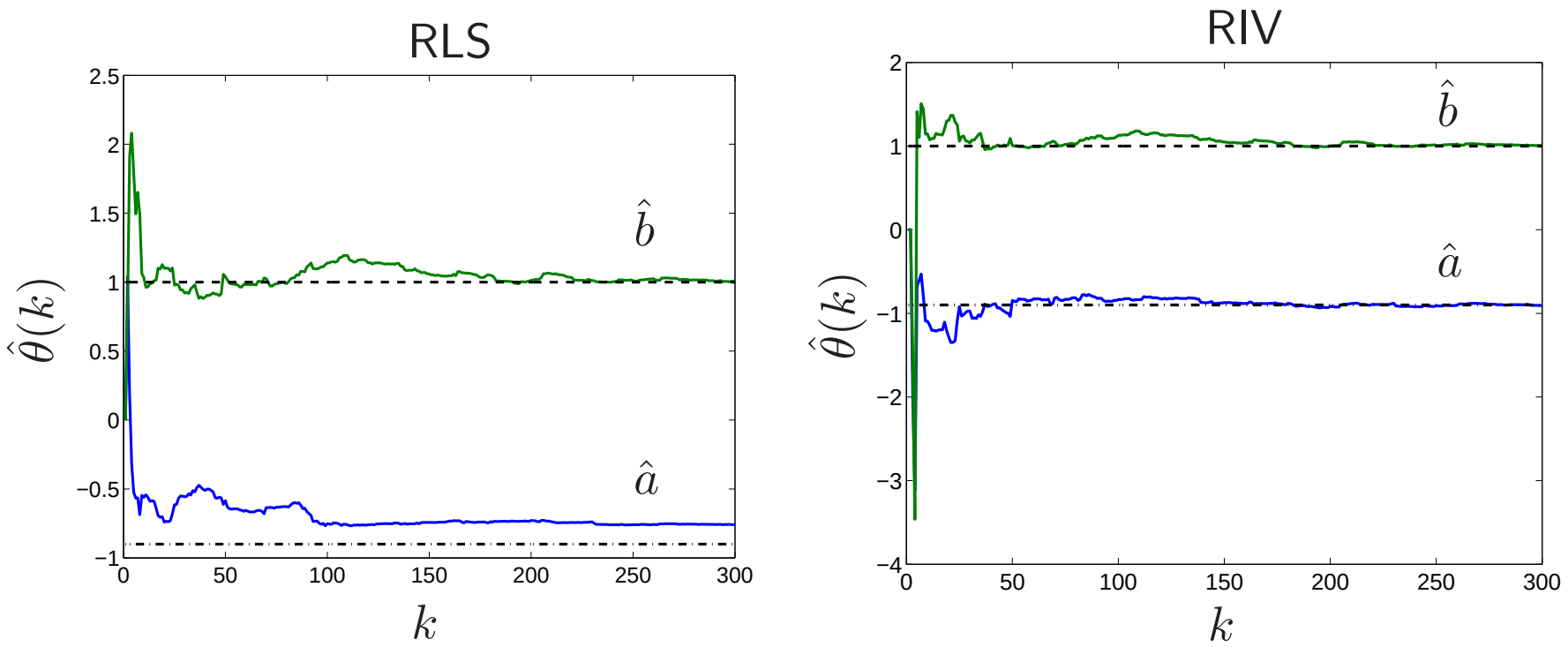
The model structure for RLS and RIV:

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

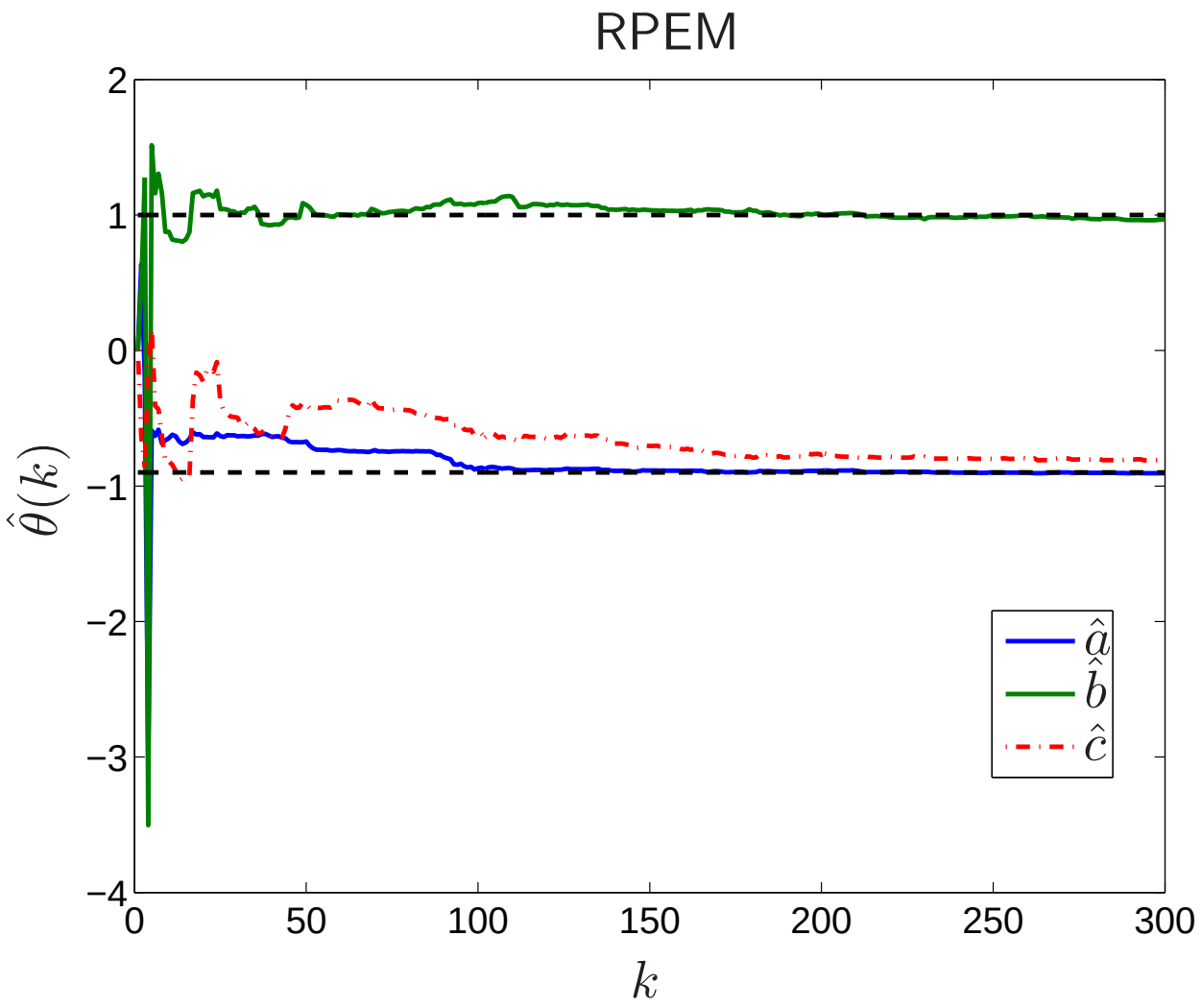
The model structure for RPEM:

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

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 and $\hat{c}$

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 $\big(H(t) \neq 0\big)$, the gain

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

  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

$P(t)$ indefinite

$P(t)$ represents a covariance matrix

Therefore, it must be symmetric and positive definite

Rounding error may accumulate to make $P(t)$ indefinite

This will make the estimate diverge

The solution is to note that every positive definite matrix can be factorized as

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

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`