

## Lectures 3

### Recursive (On-line) Identification Methods

- Recursive Least Squares (RLS) Methods
- Forgetting Factor and Tracking Time-Varying Parameters
- Identification condition problem
- Computational Aspects

## Course Outline

- Introduction and overview on system identification
- Non-recursive (off-line) identification methods
- *Recursive (on-line) identification methods (I)*
- Recursive (on-line) identification methods (II)
- Practical aspects and applications of system identification

## Why?

Why is recursive identification of interest?

- On-line Estimation.
- Adaptive Systems.
- Time Varying Parameters.
- Fault Detection and Diagnosis.
- Simulation.

## How?

How do we estimate time-varying parameters?

- Update the model regularly (once every sampling instant)
- Make use of previous calculations in an efficient manner.
- The basic procedure is to modify the corresponding off-line method, *e.g.*, the block/batch least squares method, the prediction error method.

## Desirable Properties

We desire our recursive algorithms to have the following properties:

- Fast convergence.
- Consistent estimates (time-invariant models).
- Good tracking (for time-varying parameters, e.g. in the event of fault occurrence or operating condition changes).
- Computationally simple (perform all calculations during one sampling interval).

## Trade-offs

No algorithm is perfect. The design is always based on trade-offs, such as:

- Convergence versus tracking.
- Computational complexity versus accuracy.

## Recursive Least Squares Method (RLS)

$$\hat{\boldsymbol{\theta}}(t) = \arg \min_{\boldsymbol{\theta}} V_t(\boldsymbol{\theta}), \quad V_t(\boldsymbol{\theta}) = \sum_{k=1}^t \varepsilon^2(k)$$

where  $\varepsilon(k) = y(k) - \boldsymbol{\varphi}^T(k)\boldsymbol{\theta}$ . The solution reads:

$$\hat{\boldsymbol{\theta}}(t) = \mathbf{R}_t^{-1} \mathbf{r}_t$$

where

$$\mathbf{R}_t = \sum_{k=1}^t \boldsymbol{\varphi}(k)\boldsymbol{\varphi}^T(k), \quad \mathbf{r}_t = \sum_{k=1}^t \boldsymbol{\varphi}(k)y(k)$$

- The criterion function  $V_t(\boldsymbol{\theta})$  changes every time step, hence the estimate  $\hat{\boldsymbol{\theta}}(t)$  changes every time step.
- How can we find a recursive implementation of  $\hat{\boldsymbol{\theta}}(t)$ ?

## RLS

### Algorithm:

At time  $t = 0$ : Choose initial values of  $\hat{\boldsymbol{\theta}}(0)$  and  $\mathbf{P}(0)$

At each sampling instant, update  $\boldsymbol{\varphi}(t)$  and compute

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + \mathbf{K}(t)\varepsilon(t)$$

$$\varepsilon(t) = y(t) - \boldsymbol{\varphi}^T(t)\hat{\boldsymbol{\theta}}(t-1)$$

$$\mathbf{K}(t) = \mathbf{P}(t)\boldsymbol{\varphi}(t)$$

$$\mathbf{P}(t) = \left[ \mathbf{P}(t-1) - \frac{\mathbf{P}(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^T(t)\mathbf{P}(t-1)}{1 + \boldsymbol{\varphi}^T(t)\mathbf{P}(t-1)\boldsymbol{\varphi}(t)} \right]$$

**Question:** How to obtain/derive this recursive version of LS from the block/batch LS?

## Tracking

How do we handle time-varying parameters? — two ways:

- Postulate a time-varying model for the parameters. Typically we let the parameters vary according to a random walk and use the Kalman filter as an estimator.
- Modify the cost function so that we gradually forget old data. Hence, the model is fitted to the most recent data (the parameters are adapted to describe the newest data).

- Modified cost function:

$$\hat{\boldsymbol{\theta}}(t) = \arg \min_{\boldsymbol{\theta}} V_t(\boldsymbol{\theta}), \quad V_t(\boldsymbol{\theta}) = \sum_{k=1}^t \beta(t, k) \varepsilon^2(k)$$

- Suppose that the weighting function  $\beta(t, k)$  satisfies

$$\begin{aligned} \beta(t, k) &= \lambda(t) \beta(t-1, k), \quad 0 \leq k < t \\ \beta(t, t) &= 1 \end{aligned}$$

A common choice is to let  $\lambda(t) = \lambda$ , where  $\lambda$  is referred to as a so-called *forgetting factor*. In this case we get:

$$\beta(t, k) = \lambda^{t-k}, \quad 0 < \lambda \leq 1$$

- $\lambda = 1$  corresponds to the standard RLS.

## Weighted RLS

### Algorithm:

At time  $t = 0$ : Choose initial values of  $\hat{\boldsymbol{\theta}}(0)$  and  $\mathbf{P}(0)$

At each sampling instant, update  $\boldsymbol{\varphi}(t)$  and compute

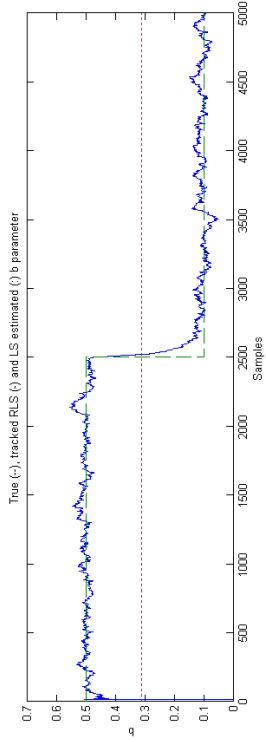
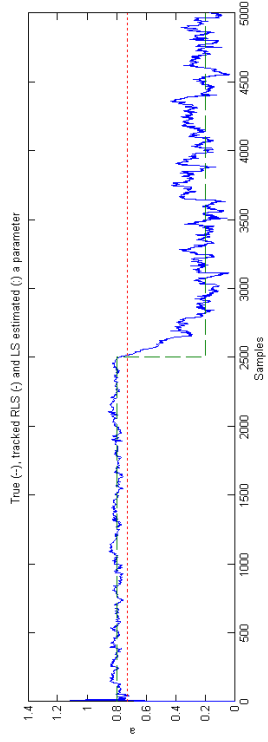
$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + \mathbf{K}(t) \varepsilon(t)$$

$$\varepsilon(t) = y(t) - \boldsymbol{\varphi}^T(t) \hat{\boldsymbol{\theta}}(t-1)$$

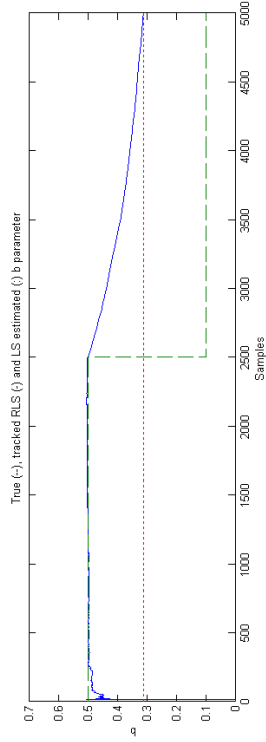
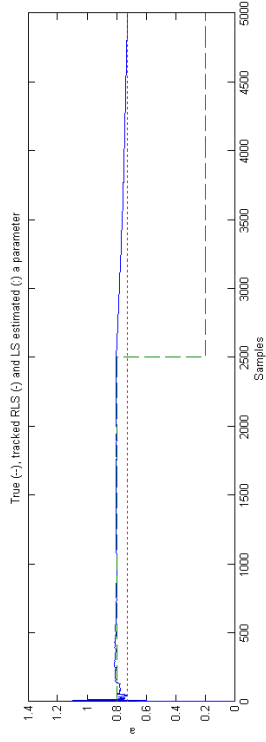
$$\mathbf{K}(t) = \mathbf{P}(t) \boldsymbol{\varphi}(t)$$

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

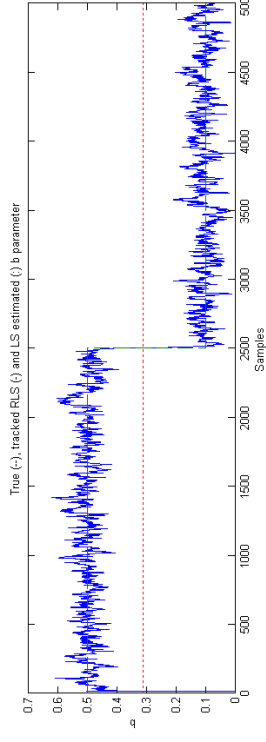
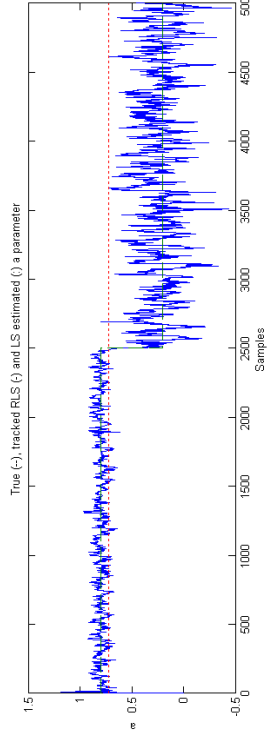
$\lambda = 0.98$



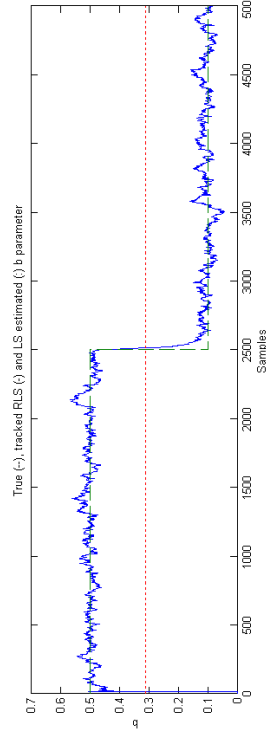
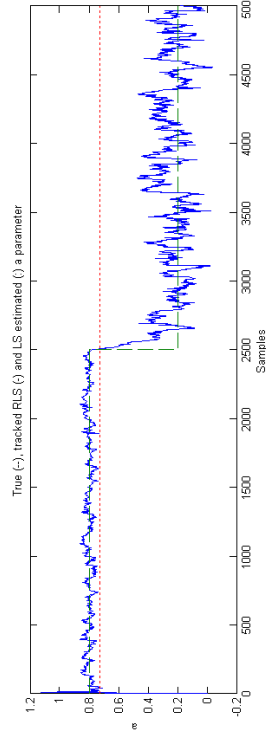
$\lambda = 1.0$



$\lambda = 0.90$



$\lambda = 0.97$



### Initial Conditions

- $\hat{\theta}(0)$  is the initial parameter estimate.
- View  $P(0)$  as an estimate of the covariance matrix of the initial parameter estimate.
  - $P(0)$  (and  $P(t)$ ) are covariance matrices, and must be symmetric and positive definite.
  - Choose  $P(0) = \rho I$ .
  - $\rho$  large  $\Rightarrow$  large initial response. Good if initial estimate  $\hat{\theta}(0)$  is uncertain.

### Forgetting Factor

Let  $\lambda(t) = \lambda$ . The forgetting factor  $\lambda$  will then determine the tracking capability.

- We must have  $\lambda = 1$  to get convergence.
- $\lambda$  small  $\Rightarrow$  Old data is forgotten faster, hence better tracking.
- $\lambda$  small  $\Rightarrow$  the algorithm is more sensitive to noise (bad convergence).
- The memory constant is defined as  $T_0 = \frac{1}{1-\lambda}$ . If  $\lambda = 0.95$ ,  $T_0 = 20$

The choice of  $\lambda$  is consequently a trade-off between tracking capability and noise sensitivity. A typical choice is  $\lambda \in (0.95, 0.99)$ . It is common to let  $\lambda(t)$  tend exponentially to 1, *e.g.*,

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



## Conclusions

- In practical scenarios, one often need to use recursive identification (time-varying systems, online identification, fault diagnosis).
- Both the LS and the IVM can easily be recast in recursive forms. The PEM can only be approximated.
- The properties of the on-line methods are comparable with the off-line case.
- Tracking capability can be incorporated by using forgetting factor techniques, or by model the parameter variations.
- There is always a tradeoff between convergence speed and tracking properties, as well as computational complexity and accuracy.
- In practice, one can make simplifications and modifications to make the recursion cheaper and more numerically robust.