Lecture 7

Recursive Identification Methods – (Ch. 9)

Why?

Why is recursive identification of interest?

- Online Estimation.
- Adaptive Systems.
- Time-varying Parameters.
- Fault Detection.

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 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 case).
- Good tracking (time-varying case).
- 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:

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

where

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

- 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 $\boldsymbol{P}(0)$

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

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

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

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

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

Tracking

How do we handle time-varying parameters?

- Postulate a time-varying model for the parameters. Typically 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

$$\beta(t,k) = \lambda(t)\beta(t-1,k), \quad 0 \le k < t$$
$$\beta(t,t) = 1$$

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

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

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

Weighted RLS

Algorithm:

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

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

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + \boldsymbol{K}(t)\varepsilon(t)$$
$$\varepsilon(t) = y(t) - \boldsymbol{\varphi}^T(t)\hat{\boldsymbol{\theta}}(t-1)$$
$$\boldsymbol{K}(t) = \boldsymbol{P}(t)\boldsymbol{\varphi}(t)$$
$$\boldsymbol{P}(t) = \frac{1}{\lambda(t)}\left[\boldsymbol{P}(t-1) - \frac{\boldsymbol{P}(t-1)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^T(t)\boldsymbol{P}(t-1)}{\lambda(t) + \boldsymbol{\varphi}^T(t)\boldsymbol{P}(t-1)\boldsymbol{\varphi}(t)}\right]$$

Initial Conditions

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

Forgetting Factor

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

- We must have $\lambda = 1$ to get convergence.
- λ small $\Rightarrow$ old data is forgotten fast, hence good tracking.
- λ small $\Rightarrow$ the algorithm is sensitive to noise (bad convergence).
- The memory constant is defined as $T_0 = \frac{1}{1-\lambda}$

The choice of λ 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.*,

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

The Kalman Filter

Consider the system:

$$\begin{aligned} \boldsymbol{x}(t+1) &= \boldsymbol{F}\boldsymbol{x}(t) + \boldsymbol{G}\boldsymbol{u}(t) + \boldsymbol{v}(t) \\ y(t) &= \boldsymbol{H}\boldsymbol{x}(t) + e(t) \end{aligned}$$

where $\boldsymbol{v}(t)$ and $e(t)$ are independent white noise sources with $Ee^2(t) = R_2$ and $E\boldsymbol{v}(t)\boldsymbol{v}^T(t) = \boldsymbol{R}_1$.

The optimal predictor of the state variable $\boldsymbol{x}(t)$ is given by the Kalman filter

$$\begin{aligned} \hat{\boldsymbol{x}}(t+1) &= \boldsymbol{F}\hat{\boldsymbol{x}}(t) + \boldsymbol{G}\boldsymbol{u}(t) + \boldsymbol{K}(t)\Big[y(t) - \boldsymbol{H}\hat{\boldsymbol{x}}(t)\Big] \\ \boldsymbol{K}(t) &= \frac{\boldsymbol{F}\boldsymbol{P}(t)\boldsymbol{H}^T}{R_2 + \boldsymbol{H}\boldsymbol{P}(t)\boldsymbol{H}^T} \end{aligned}$$

where

$$\boldsymbol{P}(t+1) = \boldsymbol{F}\boldsymbol{P}(t)\boldsymbol{F}^T - \frac{\boldsymbol{F}\boldsymbol{P}(t)\boldsymbol{H}^T\boldsymbol{H}\boldsymbol{P}(t)\boldsymbol{F}^T}{R_2 + \boldsymbol{H}\boldsymbol{P}(t)\boldsymbol{H}^T} + \boldsymbol{R}_1$$

Let us model the parameter variation according to

$$\begin{aligned} \boldsymbol{\theta}(t+1) &= \boldsymbol{\theta}(t) + \boldsymbol{v}(t) \\ y(t) &= \boldsymbol{\varphi}^T(t)\boldsymbol{\theta}(t) + e(t) \end{aligned}$$

Then

$$\begin{aligned} \hat{\boldsymbol{\theta}}(t+1) &= \hat{\boldsymbol{\theta}}(t) + \boldsymbol{K}(t)\Big[y(t) - \boldsymbol{\varphi}^T(t)\hat{\boldsymbol{\theta}}(t)\Big] \\ \boldsymbol{K}(t) &= \frac{\boldsymbol{P}(t)\boldsymbol{\varphi}(t)}{R_2 + \boldsymbol{\varphi}^T(t)\boldsymbol{P}(t)\boldsymbol{\varphi}(t)} \\ \boldsymbol{P}(t+1) &= \boldsymbol{P}(t) - \frac{\boldsymbol{P}(t)\boldsymbol{\varphi}(t)\boldsymbol{\varphi}^T(t)\boldsymbol{P}(t)}{R_2 + \boldsymbol{\varphi}^T(t)\boldsymbol{P}(t)\boldsymbol{\varphi}(t)} + \boldsymbol{R}_1 \end{aligned}$$

The tracking capability is affected by the covariance matrix $\boldsymbol{R}_1$ (let $\boldsymbol{R}_2 = 1$ for simplicity).

- View $\boldsymbol{R}_1$ as a design variable.
- Let $\boldsymbol{R}_1$ be a diagonal matrix.
- Large elements of $\boldsymbol{R}_1 \Rightarrow$ large parameter variations, and vice versa.
- The Kalman filter gives better flexibility than the forgetting factor implementation. One can easily assume different variations in different parameters.

Recursive IVM (RIV)

As for the least squares method, it is straightforward to find a recursive version of the IV method

$$\hat{\boldsymbol{\theta}}(t) = \Big[\sum_{k=1}^{t} \boldsymbol{z}(t)\boldsymbol{\varphi}^T(t)\Big]^{-1}\Big[\sum_{k=1}^{t} \boldsymbol{z}(t)y(t)\Big]$$

as

$$\begin{aligned}
\hat{\boldsymbol{\theta}}(t) &= \hat{\boldsymbol{\theta}}(t-1) + \boldsymbol{K}(t)\varepsilon(t) \\
\boldsymbol{K}(t) &= \boldsymbol{P}(t)\boldsymbol{z}(t) \\
\varepsilon(t) &= y(t) - \boldsymbol{\varphi}^T(t)\hat{\boldsymbol{\theta}}(t-1) \\
\boldsymbol{P}(t) &= \frac{1}{\lambda(t)}\Big[\boldsymbol{P}(t-1) - \frac{\boldsymbol{P}(t-1)\boldsymbol{z}(t)\boldsymbol{\varphi}^T(t)\boldsymbol{P}(t-1)}{\lambda(t) + \boldsymbol{\varphi}^T(t)\boldsymbol{P}(t-1)\boldsymbol{z}(t)}\Big]
\end{aligned}$$

Some comments:

- The influence of the design variables $\boldsymbol{P}(0)$, $\hat{\boldsymbol{\theta}}(0)$ and $\lambda(t)$ is the same as for the RLS.
- RIV $\Rightarrow$ consistent estimates of $A(q^{-1})$ and $B(q^{-1})$ in a ARMAX model (time-invariant parameters). Consistent estimates even for colored noise.

Recursive PEM (RPEM)

To derive a recursive PEM, we begin by defining the cost function (SISO)

$$V_t(\boldsymbol{\theta}) = \frac{1}{2}\sum_{s=1}^{t} \lambda^{t-s}\varepsilon^2(s, \boldsymbol{\theta})$$

It is not possible to derive an exact recursive algorithm (the off-line method relies on a numerical minimization), and some form of approximations must be used.

Assume that $\hat{\boldsymbol{\theta}}(t-1)$ minimizes $V_{t-1}(\boldsymbol{\theta})$ and that the minimum point of $V_t(\boldsymbol{\theta})$ is close to $\hat{\boldsymbol{\theta}}(t-1)$. Then using a second-order Taylor expansion of $V_t(\boldsymbol{\theta})$, one obtains

$$V_t(\boldsymbol{\theta}) \approx V_t(\hat{\boldsymbol{\theta}}(t-1)) + V_t'(\hat{\boldsymbol{\theta}}(t-1))\big(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}(t-1)\big) + \frac{1}{2}\big(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}(t-1)\big)^T V_t''(\hat{\boldsymbol{\theta}}(t-1))\big(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}(t-1)\big)$$

Minimize this with respect to $\boldsymbol{\theta}$ and let the minimum be $\hat{\boldsymbol{\theta}}(t)$:

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) - \left[V_t''(\hat{\boldsymbol{\theta}}(t-1))\right]^{-1} V_t'(\hat{\boldsymbol{\theta}}(t-1))^T$$

Rem: We must find $V_t'(\hat{\boldsymbol{\theta}}(t-1))$ and $P(t) = [V_t''(\hat{\boldsymbol{\theta}}(t-1))]^{-1}$!

Algorithm:

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) + \boldsymbol{K}(t)\varepsilon(t)$$
$$\boldsymbol{K}(t) = \boldsymbol{P}(t)\boldsymbol{\psi}(t)$$
$$\boldsymbol{P}(t) = \frac{1}{\lambda}\left[\boldsymbol{P}(t-1) - \frac{\boldsymbol{P}(t-1)\boldsymbol{\psi}(t)\boldsymbol{\psi}^T(t)\boldsymbol{P}(t-1)}{\lambda + \boldsymbol{\psi}^T(t)\boldsymbol{P}(t-1)\boldsymbol{\psi}(t)}\right]$$

where the actual way of implementing the approximations

$$\varepsilon(t) \approx \varepsilon(t, \hat{\boldsymbol{\theta}}(t-1))$$
$$\boldsymbol{\psi}(t) \approx -\left[\frac{\partial}{\partial \boldsymbol{\theta}}\varepsilon(t, \hat{\boldsymbol{\theta}}(t-1))\right]$$

depend on the model structure.

Example: ARMAX

Recursive Pseudolinear Regression (RPLR)

Consider the ARMAX model

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

Rewrite the model as

$$y(t) = \boldsymbol{\varphi}^T(t)\boldsymbol{\theta} + e(t)$$
$$\boldsymbol{\varphi}^T(t) = [-y(t-1)\ \cdots\ -y(t-n_a)\ u(t-1)\ \cdots\ u(t-n_b)\ e(t-1)\ \cdots\ e(t-n_c)]$$
$$\boldsymbol{\theta} = [a_1\ \cdots\ a_{n_a}\ b_1\ \cdots\ b_{n_b}\ c_1\ \cdots\ c_{n_c}]^T$$

Here $e(t-1), \ldots, e(t-n_c)$ are unknown! Replacing these by the prediction errors $\varepsilon(t-1), \ldots, \varepsilon(t-n_c)$ and applying RLS yields RPLR. Notice $\varepsilon(t) = y(t) - \boldsymbol{\varphi}^T(t)\hat{\boldsymbol{\theta}}(t-1)$.

Comparison between RPEM and RPLR:

- The computational demand is similar for the methods.
- The RPEM converges under weak assumptions, while for the RPLR convergence is not always assured (depends on $C_0(q^{-1})$).
- In some cases, it seems as the RPLR has a better/faster transient behavior than the RPEM.

Comparing the Methods

In the following, we will examine the following simulated system

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

where $u(t)$ and $e(t)$ are independent white noise with zero mean and unit variance. For RLS and RIV, we use the model structure (ignoring the noise color)

$$y(t) + ay(t-1) = bu(t-1) + e(t)$$
$$\boldsymbol{\theta} = [a\ b]^T$$

with RIV using the instruments $\boldsymbol{z}(t) = [u(t-1)\ u(t-2)]^T$.

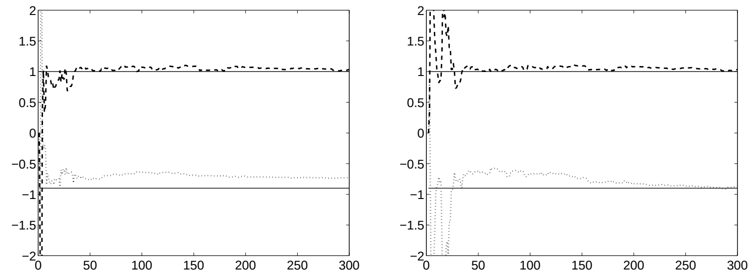

Figure 1: RLS (left) and RIV (right)

For RPEM and PLR, we take the noise into account and use the model

$$y(t) + ay(t-1) = bu(t-1) + e(t) + ce(t-1)$$
$$\boldsymbol{\theta} = [a\ b\ c]^T$$

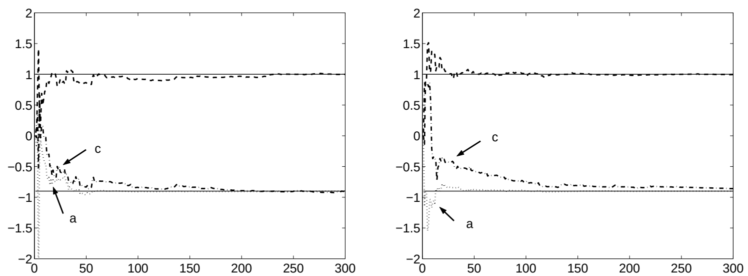

Figure 2: RPEM (left) and RPLR (right)

Effect of Initial Value

Using the same system, we examine the effect of the initial values using RLS, setting $\boldsymbol{P}(0) = \rho \boldsymbol{I}$. Larger ρ gives faster response.

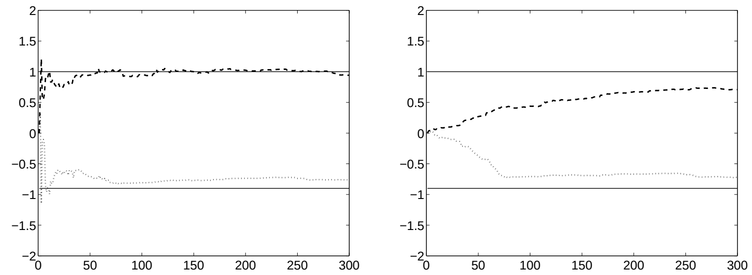

Figure 3: $\rho = 10$ (left) and $\rho = 0.01$ (right)

Effect of the forgetting factor

Using the ARMA system $y(t) - 0.9y(t-1) = e(t) + 0.9e(t-1)$ and the RPEM. Correction steps and rate of convergence increase when λ decreases. For $\lambda < 1$, the estimates do not converge, but oscillates around the true value.

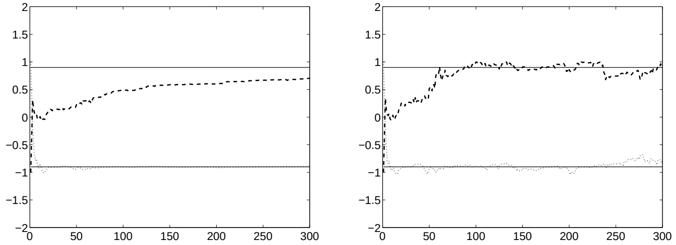

Figure 4: $\lambda = 1$ (left) and $\lambda = 0.95$ (right)

Common Problems for Recursive Identification

- Excitation.
- Estimator windup.
- $\boldsymbol{P}(t)$ becomes indefinite.

Excitation

Just as for the off-line case, it is important that the input is persistently excitation off sufficiently high order. This applies during the whole identification period.

Estimator Windup

Often, some periods of an identification experiment exhibit poor excitation. This causes problems for the identification algorithms. Consider the extreme situation when $\varphi(t) = 0$ in the RLS algorithm. Then

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1)$$

$$\boldsymbol{P}(t) = \frac{1}{\lambda}\boldsymbol{P}(t-1)$$

Notice:

- $\hat{\boldsymbol{\theta}}$ is constant as t increases.
- $\boldsymbol{P}$ increases exponentially with time for $\lambda < 1$.

When the system is excited again ($\varphi(t) \neq 0$), then the estimator gain $\boldsymbol{K}(t) = \boldsymbol{P}(t)\boldsymbol{\varphi}(t)$ will be very large, and there will be an abrupt change in the estimate $\hat{\boldsymbol{\theta}}$, despite the fact that the system has not changed. This is referred to as estimator windup.

Solution:

- Do not update $\boldsymbol{P}(t)$ if we have poor excitation. There exist several algorithms for doing this automatically.

$\boldsymbol{P}(t)$ Indefinite

$\boldsymbol{P}(t)$ is a covariance matrix $\Rightarrow$ must be symmetric and positive definite.

Rounding errors may accumulate to make $\boldsymbol{P}(t)$ indefinite (which will make the estimate diverge). The solution is to note that every positive definite matrix can be written as

$$\boldsymbol{P}(t) = \boldsymbol{S}(t)\boldsymbol{S}^T(t)$$

One then rewrites the algorithm to recursively update $\boldsymbol{S}(t)$ instead of $\boldsymbol{P}(t)$ (*Potter's Square Root Algorithm*).

Approximate Algorithms

The structure of the RPEM (Newton-Raphson)

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) - \left[V_t^{''}(\hat{\boldsymbol{\theta}}(t-1))\right]^{-1} V_t'(\hat{\boldsymbol{\theta}}(t-1))^T$$

- Cumbersome to compute the Hessian $V_t^{''}(\hat{\boldsymbol{\theta}}(t-1))$.
- Approximate algorithms that are less computationally demanding. For instance, ignoring the Hessian:

$$\hat{\boldsymbol{\theta}}(t) = \hat{\boldsymbol{\theta}}(t-1) - \gamma_t V_t'(\hat{\boldsymbol{\theta}}(t-1))^T$$

  This leads to the steepest decent algorithm, least-mean-square algorithm (LMS), ...

Conclusions

- In practical scenarios, one often need to use recursive identification (time-varying systems, on-line identification, fault detection).
- Both the LSM and the IVM can easily be recast in recursive forms. The PEM can only be approximated to a recursive algorithm.
- The properties of the on-line methods are comparable with the off-line case.
- Tracking capability can be incorporated by using a forgetting factor, or by modeling the parameter variations.
- Tradeoffs between convergence speed and tracking properties, as well as between computational complexity and accuracy.
- In practise, one can simplify and modify to make the recursion cheaper and more numerically robust.