Chapter 8

Recursive Identification

"Given a current estimated model and a new observation, how should we update this model in order to take this new piece of information into account?"

In many cases it is beneficial to have a model of the system available online while the system is in operation. The model should then be based on the observations up till the current time. A naive way to go ahead is to use all observations up to t to compute an estimate $\hat{\theta}_t$ of the system parameters. In recursive identification methods, the parameter estimates are computed recursively over time: suppose we have an estimate $\hat{\theta}_{t-1}$ at iteration $t-1$, then recursive identification aims to compute a new estimate $\hat{\theta}_t$ by a 'simple modification' of $\hat{\theta}_{t-1}$ when a new observation becomes available at iteration t. The counterpart to online methods are the so-called offline or batch methods in which all the observations are used simultaneously to estimate the model.

Recursive methods have the following general features:

- They are a central part in adaptive systems where the next action is based on the latest estimated parameters. Typical examples are found in adaptive control or adaptive filtering applications.

- Memory and computational requirements at any timestep has to be modest. Specifically, one often requires that both are independent to the length of the history at any timestep.

- They are often applied to real-time settings, where the 'true' underlying parameters are changing over time (i.e. tracking applications).

- They are often used for fault detection systems. Here one wants to detect when the observed signals or the underlying system differs significantly from what one would associate from a normal operation modus.

In general, the techniques go with the same statistical properties as their counterparts in 'batch' setting. For example, the RLS gives consistent estimates under the conditions as discussed in Section 5.3. That is, the discussion on the recursive estimators is often concerned with computational issues.

8.1 Recursive Least Squares

Let us start this section with perhaps the simplest application possible, nevertheless introducing ideas.

Example 50 (RLS for Estimating a Constant) *Given the following system*

$$y_t = \theta_0 + e_t, \ \forall t = 1, 2, \dots. \tag{8.1}$$

In chapter 2, example 1 we derive how the least squares estimate of θ_0 using the first t observations is given as the arithmetic (sample) mean, i.e.

$$\hat{\theta}_t = \frac{1}{t}\sum_{i=1}^{t} y_i. \tag{8.2}$$

Now it is not too difficult to rewrite this in a recursive form.

$$\hat{\theta}_t = \frac{1}{t}\left(\sum_{i=1}^{t-1} y_i + y_t\right) = \frac{1}{t}\left((t-1)\hat{\theta}_{t-1} + y_t\right) = \hat{\theta}_{t-1} + \frac{1}{t}\left(y_t - \hat{\theta}_{t-1}\right). \tag{8.3}$$

This result is quite appealing: the new estimate $\hat{\theta}_t$ equals the previous estimate $\hat{\theta}_{t-1}$ plus a small correction term. The correction term is proportional to the deviation of the prediction $\hat{\theta}_{t-1}$ and the observation y_t. Moreover the correction term is weighted by the term $\frac{1}{t}$, which implies that the magnitude of the correction will decrease in time. Instead the estimate $\hat{\theta}_{t-1}$ will become more reliable. In case a proper stochastic framework is assumed (see chapter 5, section 3), the variance of $\hat{\theta}_t$ becomes

$$P_t = \frac{1}{t}, \tag{8.4}$$

which can in turn be computed recursively as

$$P_t = \frac{1}{P_{t-1}^{-1} + 1} = \frac{P_{t-1}}{1 + P_{t-1}}. \tag{8.5}$$

In order to generalize the result, we need the following well-known matrix properties.

Lemma 9 (Matrix Inversion Lemma) *Let $\mathbf{Z} \in \mathbb{R}^{d\times d}$ be a positive definite matrix with unique inverse $\mathbf{Z}^{-1}$, and let $\mathbf{z} \in \mathbb{R}^d$ be any vector, then*

$$\mathbf{Z}_+^{-1} = (\mathbf{Z} + \mathbf{z}\mathbf{z}^T)^{-1} = \mathbf{Z}^{-1} - \frac{\mathbf{Z}^{-1}\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-T}}{1 + \mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}}, \tag{8.6}$$

where $\mathbf{Z}_+^{-1} = \mathbf{Z} + \mathbf{z}\mathbf{z}^T$.

In words, the inverse of a matrix with a rank-one update can be written in closed form using the inverse of the matrix and a small correction. *Proof:* The proof is instrumental.

$$\begin{aligned}
(\mathbf{Z}+\mathbf{z}\mathbf{z}^T)\left(\mathbf{Z}^{-1}-\frac{\mathbf{Z}^{-1}\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}}{1+\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}}\right) &= I_d-\mathbf{Z}\left(\frac{\mathbf{Z}^{-1}\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}}{1+\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}}\right)+\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}-(\mathbf{z}\mathbf{z}^T)\left(\frac{\mathbf{Z}^{-1}\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}}{1+\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}}\right)\\
&= I_d-\left(\frac{\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}}{1+\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}}\right)+\left(\frac{\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}}{1+\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}}\right)(1+\mathbf{z}\mathbf{Z}^{-1}\mathbf{z})-\left(\frac{\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}}{1+\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}}\right)\\
&= I_d+\left(\frac{\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}}{1+\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}}\right)(\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1})-\left(\frac{\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}}{1+\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}}\right).
\end{aligned}$$

Now, note that $(\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z})$ is a scalar, and thus

$$= I_d+\left(\frac{\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}}{1+\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}}\right)(\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z})-\left(\frac{(\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z})\mathbf{z}\mathbf{z}^T\mathbf{Z}^{-1}}{1+\mathbf{z}^T\mathbf{Z}^{-1}\mathbf{z}}\right)=I_d, \tag{8.7}$$

as desired.

□

The previous example serves as a blueprint of the Recursive Least Squares (RLS) algorithm, which we now will develop in full. Given a model for the observations $\{(\mathbf{x}_t, y_t)\}_t \subset \mathbb{R}^{d\times 1}$ given as

$$y_t = \theta_0^T\mathbf{x}_t + e_t,\ \forall t = 1,2,\ldots, \tag{8.8}$$

where $\theta_0 \in \mathbb{R}^d$ and the terms $\{e_t\}_t$ are the corresponding residuals. Then chapter 2 learns us that the LS solution based on the observations $x_i : i = 1,\ldots,t$ will be given as the solution to the normal equations

$$\left(\sum_{i=1}^t \mathbf{x}_i\mathbf{x}_i^T\right)\hat\theta_t = \left(\sum_{i=1}^t y_i\mathbf{x}_i\right). \tag{8.9}$$

Assume for now that the solution $\hat\theta_t$ is unique, i.e. the matrix $R_t = \left(\sum_{i=1}^t \mathbf{x}_i\mathbf{x}_i^T\right)$ can be inverted. Since trivially one has

$$\mathbf{R}_{t-1} = \mathbf{R}_t - \mathbf{x}_t\mathbf{x}_t^T, \tag{8.10}$$

if follows that

$$\begin{aligned}
\hat\theta_t &= \mathbf{R}_t^{-1}\left(\sum_{i=1}^{t-1} y_i\mathbf{x}_i + y_t\mathbf{x}_t\right)\\
&= \mathbf{R}_t^{-1}\left(\mathbf{R}_{t-1}\hat\theta_{t-1} + y_t\mathbf{x}_t\right)\\
&= \hat\theta_{t-1} + \mathbf{R}_t^{-1}\left(y_t\mathbf{x}_t - (\mathbf{x}_t\mathbf{x}_t^T)\hat\theta_{t-1}\right)\\
&= \hat\theta_{t-1} + \mathbf{R}_t^{-1}\mathbf{x}_t\left(y_t - \hat\theta_{t-1}^T\mathbf{x}_t\right),
\end{aligned} \tag{8.11}$$

and in summary

$$\begin{cases}\epsilon_t = (y_t - \mathbf{x}_t^T\hat\theta_{t-1})\\ \mathbf{K}_t = \mathbf{R}_t^{-1}\mathbf{x}_t\\ \hat\theta_t = \hat\theta_{t-1} + \mathbf{K}_t\epsilon_t.\end{cases} \tag{8.12}$$

Here the term ϵ_t will be interpreted as the prediction error: it is the difference between the observed sample y_t and the predicted value $\mathbf{x}_t^T\hat{\theta}_{t-1}$. If ϵ_t is 'small', the estimate $\hat{\theta}_{t-1}$ is good and should not be modified much. The matrix $\mathbf{K}_t$ is interpreted as the weighting or 'gain' matrix characterizing how much each element of the parameter vector $\hat{\theta}_{t-1}$ should be modified by ϵ_t.

The RLS algorithm is completed by circumventing the matrix inversion of $\mathbf{R}_t$ in each timestep. Hereto, we can use the matrix inversion Lemma.

$$\mathbf{R}_t^{-1} = \mathbf{R}_{t-1}^{-1} - \frac{\mathbf{R}_{t-1}^{-1}\mathbf{x}_t\mathbf{x}_t^T\mathbf{R}_{t-1}^{-1}}{1+\mathbf{x}_t^T\mathbf{R}_{t-1}^{-1}\mathbf{x}_t}. \tag{8.13}$$

Note that as such we substitute the matrix inversion by a simple scalar division.

$$\mathbf{K}_t = \mathbf{R}_t^{-1}\mathbf{x}_t = \mathbf{R}_{t-1}^{-1}\mathbf{x}_t - \frac{\mathbf{R}_{t-1}^{-1}\mathbf{x}_t(\mathbf{x}_t^T\mathbf{R}_{t-1}^{-1}\mathbf{x}_t)}{1+\mathbf{x}_t^T\mathbf{R}_{t-1}^{-1}\mathbf{x}_t} = \left(\frac{1}{1+\mathbf{x}_t^T\mathbf{R}_{t-1}^{-1}\mathbf{x}_t}\right)\mathbf{R}_{t-1}^{-1}\mathbf{x}_t. \tag{8.14}$$

Given initial values $\mathbf{R}_0^{-1}$ and $\hat{\theta}_0$, the final RLS algorithm can as such be written as

$$\begin{cases} \epsilon_t = (y_t - \mathbf{x}_t^T\hat{\theta}_{t-1}) \\ \mathbf{P}_t = \mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1}\mathbf{x}_t\mathbf{x}_t^T\mathbf{P}_{t-1}}{1+\mathbf{x}_t^T\mathbf{P}_{t-1}\mathbf{x}_t} \\ \mathbf{K}_t = \mathbf{P}_t\mathbf{x}_t = \left(\frac{1}{1+\mathbf{x}_t^T\mathbf{P}_{t-1}\mathbf{x}_t}\right)\mathbf{P}_{t-1}\mathbf{x}_t \\ \hat{\theta}_t = \hat{\theta}_{t-1} + \mathbf{K}_t\epsilon_t, \end{cases} \tag{8.15}$$

where we use $\mathbf{P}_t = \mathbf{R}_t^{-1}$ for any t. For efficiency reasons, one can We will come back to the important issue on how to choose the initial values $\mathbf{P}_0$ and $\hat{\theta}_0$ in Subsection 8.1.2.

8.1.1 Real-time Identification

This subsection presents some ideas which are useful in case the RLS algorithm is applied for tracking time-varying parameters. This is for example the case when the 'true' parameter vector θ_0 varies over time, and are as such denoted as $\{\theta_{0,t}\}_t$. This setting is referred to as the *real-time identification* setting. There are two common approaches to modify the RLS algorithm to handle such case: (i) use of a forgetting factor (this subsection); (ii) use of a Kalman filter as a parameter estimator (next subsection).

The 'forgetting factor' approach starts from a slightly modified loss function

$$V_t(\theta) = \sum_{s=1}^{t}\lambda^{t-s}(y_t - \theta^T\mathbf{x}_t)^2. \tag{8.16}$$

The Squared Loss function lying at the basis of RLS is recovered when $\lambda = 1$. If λ is set to some value slightly smaller than 1 (say $\lambda = 0.99$ or $\lambda = 0.95$), one has that for increasing t past observations are discounted. The smaller λ got, the quicker information obtained from previous data will be forgotten, and hence the name. It is now straightforward to re-derive the RLS based on (8.16), and the modified RLS becomes:

$$\begin{cases} \epsilon_t = (y_t - \mathbf{x}_t^T \hat{\theta}_{t-1}) \\ \mathbf{P}_t = \frac{1}{\lambda}\left(\mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1}\mathbf{x}_t\mathbf{x}_t^T\mathbf{P}_{t-1}}{\lambda + \mathbf{x}_t^T\mathbf{P}_{t-1}\mathbf{x}_t}\right) \\ \mathbf{K}_t = \mathbf{P}_t\mathbf{x}_t = \left(\frac{1}{\lambda + \mathbf{x}_t^T\mathbf{P}_{t-1}\mathbf{x}_t}\right)\mathbf{P}_{t-1}\mathbf{x}_t \\ \hat{\theta}_t = \hat{\theta}_{t-1} + \mathbf{K}_t\epsilon_t. \end{cases} \tag{8.17}$$

Example 51 (Estimator Windup) *Often, some periods of the identification experiment exhibit poor excitation. This causes problems for the identification algorithms. Consider the situation where $\varphi_t = 0$ in the RLS algorithm, then*

$$\begin{cases} \hat{\theta}_t = \hat{\theta}_{t-1} \\ \mathbf{P}_t = \frac{1}{\lambda}\mathbf{P}_{t-1}, \end{cases} \tag{8.18}$$

- *Notice that $\hat{\theta}$ remains constant during this period,*
- *... and $\mathbf{P}$ increases exponentially with time when $\lambda < 1$.*

When the system is excited again ($\varphi_t \neq 0$), the estimation gain $\mathbf{K}$ will be very large, and there will be an abrupt change in the estimate, despite the fact that the system has not changed. This effect is referred to as 'estimator windup'.

Since the study of Kalman filters will come back in some detail in later chapters, we treat the Kalman filter interpretation as merely an example here.

Example 52 (RLS as a Kalman Filter) *A stochastic state-space system takes the form*

$$\begin{cases} X_{t+1} = \mathbf{F}_t X_t + V_t \\ Y_t = \mathbf{H}_t X_t + W_t \end{cases} \quad \forall t = 1, 2, 3, \ldots, \tag{8.19}$$

where

- $\{X_t \in \mathbb{R}^n\}_t$ *denote the stochastic states,*
- $\{Y_t \in \mathbb{R}^m\}_t$ *denote the observed outcomes.*
- $\{V_t \in \mathbb{R}^n\}_t$ *denote the process noise.*
- $\{W_t \in \mathbb{R}^m\}_t$ *denote the observation noise.*
- $\{\mathbf{F}_t \in \mathbb{R}^{n \times n}\}_t$ *are called the* system matrices
- $\{\mathbf{H}_t \in \mathbb{R}^{m \times n}\}_t$

Now it is easily seen that the problem of time-invariant RLS estimation can be written as

$$\begin{cases} \theta_{t+1} = \theta_t \\ Y_t = \mathbf{x}_t^T\theta_t + E_t \end{cases} \quad \forall t = 1, 2, 3, \ldots, \tag{8.20}$$

where $\theta = \theta_1 = \dots = \theta_t = \dots$ is the unknown state one wants to estimate based on observations $\{Y_t\}_t$. Hence one can phrase the problem as a filtering problem, where the Kalman filter provides the optimal solution to under appropriate assumptions, eventually reducing to (8.15). The benefit of this is that one can extend the model straightforwardly by including unknown process noise terms $\{V_t\}_t$, modeling the drifting true values as a random walk - approaching effectively the real-time identification problem. Suppose $\{V_1, \dots, V_t, \dots\}$ are sampled independently from a Gaussian with mean zero and covariance $\mathbf{V} \in \mathbb{R}^{n\times n}$, then the Kalman filter would become

$$\begin{cases} \epsilon_t = (y_t - \mathbf{x}_t^T \hat{\theta}_{t-1}) \\ \mathbf{P}_t = \left(\mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1}\mathbf{x}_t\mathbf{x}_t^T\mathbf{P}_{t-1}}{1+\mathbf{x}_t^T\mathbf{P}_{t-1}\mathbf{x}_t}\right) + \mathbf{V} \\ \mathbf{K}_t = \mathbf{P}_t\mathbf{x}_t \\ \hat{\theta}_t = \hat{\theta}_{t-1} + \mathbf{K}_t\epsilon_t. \end{cases} \tag{8.21}$$

Observe that both in case (8.17) as in (8.21) the basic RLS algorithm is modified such that $\mathbf{P}_t$ will no longer tend to zero. In this way $\mathbf{K}_t$ also is prevented from decreasing to zero. The parameter estimate will therefore change continuously.

8.1.2 Choosing Initial Values

The choice of the initial values is paramount in real life application of RLS schemes. Close inspection of the meaning of $\mathbf{P}_t$ helps us here. In the Kalman filter interpretation of RLS $\mathbf{P}_t$ plays the role of the covariance matrix of the estimate $\hat{\theta}_t$, as such suggesting that in case one is not at all certain of a certain choice of $\hat{\theta}_0$, one should take a large $\mathbf{P}_0$; if one is fairly confident in a certain choice of $\hat{\theta}_0$, $\mathbf{P}_0$ should be taken small. If $\mathbf{P}_0$ is small, so will $\{\mathbf{K}_t\}_{t>0}$ and the estimate $\{\hat{\theta}_t\}_t$ will not change too much from $\hat{\theta}_0$. If $\mathbf{P}_0$ would be large, $\hat{\theta}_t$ will quickly jump away from $\hat{\theta}_0$. Without a priori knowledge, it is common practice to take the following initial values

$$\begin{cases} \hat{\theta} = 0_d \\ \mathbf{P}_0 = \rho I_d, \end{cases} \tag{8.22}$$

with $I_d = \text{diag}(1, \dots, 1) \in \mathbb{R}^{d\times d}$ the identity matrix, and $\rho > 0$ a 'large number'.

The effect on the choice of the initial values (or the 'transient behavior) can be derived algebraically. Consider the basic RLS algorithm (8.15). Then

$$\mathbf{R}_t = \mathbf{R}_0 + \sum_{s=1}^{t} \mathbf{x}_t\mathbf{x}_t. \tag{8.23}$$

Now set

$$\mathbf{z}_t = \mathbf{R}_t\hat{\theta}_t. \tag{8.24}$$

Then

$$\mathbf{z}_t = \mathbf{R}_t\hat{\theta}_{t-1} + \mathbf{x}_t\epsilon_t = \left(\mathbf{R}_{t-1} + \mathbf{x}_t\mathbf{x}_t^T\right)\hat{\theta}_{t-1} + \mathbf{x}_t\left(y_t - \hat{\theta}_{t-1}^T\mathbf{x}_t\right) = \mathbf{z}_{t-1} + \mathbf{x}_t y_t = \mathbf{z}_0 + \sum_{s=1}^{t} \mathbf{x}_s y_s. \tag{8.25}$$

And hence

$$\hat{\theta}_t = \mathbf{P}_t\mathbf{z}_t = \left(\mathbf{R}_0 + \sum_{s=1}^{t} \mathbf{x}_s\mathbf{x}_s^T\right)^{-1}\left(\mathbf{R}_0\hat{\theta}_0 + \sum_{s=1}^{t} \mathbf{x}_s y_s\right). \tag{8.26}$$

So, if $\mathbf{R}_0$ is small (i.e. $\mathbf{P}_0$ is large), then $\hat{\theta}_t$ is close to the offline estimate

$$\theta_t^* = \operatorname*{argmin}_\theta \sum_{s=1}^t (y_s - \theta^T \mathbf{x}_t)^2, \tag{8.27}$$

as seen by comparison of (8.26) with the normal equations associated to (8.27)

$$\theta_t^* = \left(\sum_{s=1}^t \mathbf{x}_s \mathbf{x}_s^T\right)^{-1} \left(\sum_{s=1}^t \mathbf{x}_s y_s\right). \tag{8.28}$$

The methods discussed in the above subsections are appropriate to systems that are known to change slowly over time. In such cases λ is chosen close to 1, or $\mathbf{V}$ is chosen as a small non-negative positive definite matrix. If the system exhibits more likely from time to time some abrupt changes of the parameters, techniques based on fault detection might be more suitable.

8.1.3 An ODE Analysis

Simulation no doubt gives useful insight. However, it is also clear that it does not permit generally valid conclusions to be drawn, and therefore it is only a complement to theory. The scope of a theoretical derivation would in particular be to study whether the parameter estimates $\hat{\theta}_t$ *converge* as t tends to infinity. If so, to what limit? And if possible also to establish the limiting distribution of $\hat{\theta}_t$.

A successful approach considers the sequence $\{\hat{\theta}_t\}_{t=0,1,2}$ as approximating a continuous vector valued function $\{\theta : \mathbb{R}_+ \to \mathbb{R}^d\}$. This continuos function evaluated at a time instant $\tau > 0$ is denoted as $\theta(\tau)$, and the whole sequence is described as an Ordinary Differential Equation. Such approach typically adopts a stochastic setting where $\{Y_t\}_t$is a stochastic process, and $\{X_t\}_t$ is a vector valued stochastic process, and both have bounded first- and second moments. Recall that the minimal MMSE $\theta_* \in \mathbb{R}^d$ is then given as the solution to

$$\mathbb{E}\left[X_t X_t^T\right] \theta_* = \mathbb{E}\left[X_t Y_t\right]. \tag{8.29}$$

Define again $\mathbf{R} = \mathbb{E}\left[X_t X_t^T\right]$, and suppose this one is invertible. Define the functional r (recall that θ here is a function) as:

$$r(\theta) = \mathbb{E}\left[X_t(Y_t - X_t^T\theta)\right]. \tag{8.30}$$

Now, consider the following ODE

$$\frac{\partial\theta(\tau)}{\partial\tau} = \mathbf{R}^{-1} r(\theta(\tau)). \tag{8.31}$$

If this ODE is solved numerically by an Euler method on discretization steps $\tau_1, \tau_2, \ldots$ one gets

$$\theta_{\tau_k} \approx \theta_{\tau_{k-1}} + (\tau_k - \tau_{k-1})\mathbf{R}^{-1}\mathbb{E}[Y_t - X_t^T \theta_{\tau_{k-1}}]. \tag{8.32}$$

Note the similarity between (8.32) and the algorithm (8.15), suggesting that the solutions to the deterministic ODE will be close in some sense. Specifically, consider the following recursion described by the algorithm

$$\hat{\theta}_t = \hat{\theta}_{t-1} + \gamma_t \mathbf{R}^{-1} X_t (Y_t - X_t^T \hat{\theta}_t). \tag{8.33}$$

with $\hat{\theta}_0$ given. Then the paths described by this discrete recursion will be similar to the solutions $\{\theta_{\tau_k}\}_k$ using the timescale $(\tau_k - \tau_{k-1}) = \gamma_t$. The above statements can be made quite precise as was done in [**?**]. The study of this ODE gives us new insight in the RLS algorithm, including:

1. The trajectories which solve the ODE are the expected paths of the algorithm.
2. Assume that there is a positive function $V(\theta, \mathbf{R})$ such that along along the solutions of the ODE we have that $\frac{\partial}{\partial\tau} V(\theta(\tau), \mathbf{R}) \le 0$. Then as $\tau \to \infty$, $\theta(\tau)$ either tend to the set
$$D_c = \left\{ \theta_* \;\middle|\; \frac{\partial}{\partial\tau} V(\theta(\tau), \mathbf{R}) = 0 \right\}, \tag{8.34}$$
or to the boundary of the set of feasible solutions. In other words, $\theta(\tau)$ for $\tau \to \infty$ go to the stable stationary points of the ODE. Equivalently, $\hat{\theta}_t$ converge locally to a solution in D_c.

8.2 Other Algorithms

Since the problem of recursive identification, adaptive filtering or online estimation is so ubiquitous, it comes as no surprise that many different approaches exist. This section reviews three common variations.

8.2.1 Recursive Instrumental Variables

Recall that instrumental variable techniques come into the picture when the noise is known to be strongly colored, and a plain LSE is not consistent. An instrumental variable estimator uses random instruments $\{Z_t\}$ which are known to be independent to the noise of the system. Then we look for the parameters which match this property using the sample correlations instead. Formally, consider the statistical system
$$Y_t = \mathbf{x}_t^T \theta_0 + D_t, \tag{8.35}$$
where $\{D_t\}$ is colored noise, and $\mathbf{x}_t \in \mathbb{R}^d$, with deterministic but unknown vector θ_0. Suppose we have d-dimensional instruments Z_t such that
$$\mathbb{E}\left[Z_t D_t\right] = 0_d. \tag{8.36}$$
That is, the instruments are orthogonal to the noise. Then the (batch) IV estimator θ_n is given as the solution of $\theta \in \mathbb{R}^d$ to
$$\sum_{t=1}^{n} Z_t (Y_t - \mathbf{x}_t^T \theta) = 0_d, \tag{8.37}$$
which look similar to the normal equations. If $\sum_{t=1}^{n} (Z_t \mathbf{x}_t^T)$ were invertible, then the solution is unique and can be written as
$$\theta_n = \left(\sum_{t=1}^{n} Z_t \mathbf{x}_t^T \right)^{-1} \left(\sum_{t=1}^{n} Z_t Y_t^T \right), \tag{8.38}$$
Now we can use the techniques used for RLS to construct a recursive method to estimate θ_t when the data comes in. It is a simple example to derive the algorithm, which is given as
$$\begin{cases} \epsilon_t = (y_t - \mathbf{x}_t^T \hat{\theta}_{t-1}) \\ \mathbf{P}_t = \mathbf{P}_{t-1} - \frac{\mathbf{P}_{t-1} Z_t \mathbf{x}_t^T \mathbf{P}_{t-1}}{1 + \mathbf{x}_t^T \mathbf{P}_{t-1} Z_t} \\ \mathbf{K}_t = \mathbf{P}_t Z_t \\ \hat{\theta}_t = \hat{\theta}_{t-1} + \mathbf{K}_t \epsilon_t. \end{cases} \tag{8.39}$$

The discussion on the behavior of RLS w.r.t. initial variables and forgetting factor remains valid.

8.2.2 Recursive Prediction Error Method

Recall that a PEM method bases inference on maximizing performance of the best predictor corresponding to a model. Also this technique is straightforwardly to phrase in a recursive form.

$$\theta_t = \operatorname*{argmin}_{\theta} V_t(\theta) = \sum_{k=1}^{t} \lambda^{t-k}\epsilon_k(\theta), \tag{8.40}$$

where $0 < \lambda \leq 1$ is typically chosen as $0.99, 0.95, 0.9$. As before $\epsilon_t(\theta)$ denotes the prediction errors of corresponding to model parameters θ, that is $\epsilon_t(\theta) = y_t - \hat{y}_t(\theta)$ where $\hat{y}_t(\theta)$ is the optimal predictor at the tth instance. Now, unlike the previous algorithms, no closed form solution of (8.40) exists in general, and one resorts to numerical optimization tools. But there is an opportunity here: it is not too difficult to integrate -say- a Gauss-Newton step in the optimizer with the online protocol.

To see how this goes, consider again the second order Taylor decomposition of the loss function. Lets assume we have a fairly good estimate $\hat{\theta}_{t-1}$ at the previous instance

$$V_t(\theta) = V_t(\hat{\theta}_{t-1}) + V'(\hat{\theta}_{t-1})^T(\theta - \hat{\theta}_{t-1}) + \frac{1}{2}(\theta - \hat{\theta}_{t-1})^T V_t''(\hat{\theta}_{t-1})(\theta - \hat{\theta}_{t-1}). \tag{8.41}$$

Now, the challenge is to compute gradient V_t' and Hessian V_t'' recursively. Details can be found in the book (Söderström, Stoica, 1989), but are necessarily tied to the adapted model and are often approximative in nature.

8.2.3 Recursive Pseudo-linear Least Squares

The following example expresses an ARMAX as a pseudo-linear model as follows.

Example 53 (ARMAX) *Given an ARMAX system*

$$A(q^{-1})y_t = B(q^{-1})u_t + C(q^{-1})e_t, \tag{8.42}$$

of orders n_a, n_b, n_c. Then this system can 'almost' be written as a LIP model as follows

$$y_t = \varphi_t^T\theta_0 + e_t, \tag{8.43}$$

where

$$\begin{cases} \varphi_t = (-y_{t-1}, \dots, -y_{t-n_a}, u_{t-1}, \dots, u_{t-n_b}, \hat{e}_{t-1}, \dots, \hat{e}_{t-n_c})^T \\ \theta_0 = (a_1, \dots, a_{t-n_a}, b_1, \dots, b_{t-n_b}, c_1, \dots, c_{t-n_c}), \end{cases} \tag{8.44}$$

where $\hat{e}_t$ is the prediction error computed based on the model parameters $\hat{\theta}_{t-1}$. The rationale is that in case $\theta_{t-1} \approx \theta_t$, $\hat{e}_t$ is a good proxy to the prediction errors e_t based on the parameters θ_t. Then the Recursive Partial Least Squares algorithm implements a RLS strategy based on this 'linearized' model.

Indeed one can prove that the resulting estimates do converge if the system is obeys some regularity conditions. Specifically, if the system is almost unstable the recursive estimates are often unstable (and diverging) as well. In practice, the resulting algorithm needs monitoring of the resulting estimates in order to detect such divergent behavior.

8.2.4 Stochastic Approximation

The class of stochastic approximation techniques take a quite different perspective on the recursive identification problem. Here the parameter estimate $\hat{\theta}_{t-1}$ obtained previously is modified such that it is better suited for explaining the new sample related to (φ_t, y_t). Formally, a new estimate $\hat{\theta}_t$ is obtained from the given $\hat{\theta}_{t-1}$ and the sample (φ_t, y_t) by solving for a given $\gamma > 0$ the optimization problem

$$\hat{\theta}_t = \operatorname*{argmin}_{\theta} J_t(\theta) = (\theta - \hat{\theta}_{t-1})^T(\theta - \hat{\theta}_{t-1}) + \gamma \left(\mathbf{x}_t^T\theta - y_t\right)^2. \tag{8.45}$$

The optimal result is then given directly as

$$\hat{\theta}_t = \hat{\theta}_{t-1} - \gamma \left(\mathbf{x}^T\theta - y_t\right)\varphi_t, \tag{8.46}$$

obtained by equating the derivative of $J_t(\theta)$ to zero. The algorithm is then completed by specification of the initial estimate $\hat{\theta}_0$. This recursion gives then what is called the Least Mean Squares (LMS) algorithm. This is the building stone of many implementations of adaptive filtering. The naming convention 'stochastic approximation' is motivated as follows. The correction at instance t is based on the gradient of a single point $(\mathbf{x}_t, y_t)$, and is a very 'noisy' estimate of the overall gradient. A variation of this algorithm is given by the recursion

$$\hat{\theta}_t = \hat{\theta}_{t-1} - \frac{\gamma}{\|\mathbf{x}_t\|_2 + \epsilon}\left(\mathbf{x}_t^T\theta - y_t\right), \tag{8.47}$$

with $\epsilon > 0$ small, and where $\hat{\theta}_0$ is given. This recursion is the basis of the Normalized LMS algorithm. The rationale is that here each sample modifies the present estimate proportional how close the estimate is to the working point 0_d.

8.3 Model Selection

As in the batch setting, it is paramount to be able to qualify and quantify how well our recursive algorithms succeeds in its task. But the conceptual and practical ways to do turn out to be entirely different. As it stands there is no comprehensive theoretical framework for this question, but some insight is gained in the following example.

Example 54 (Predicting Random noise) *As seen, a lot of fancy mathematics can be brought in to form complex recursive schemes, but at the end of the day the methods implemented need 'merely' may good predictions. It helps to reason about this objective by considering the prediction of random white noise: by construction this is impossible to do better than $\hat{y}_t = 0$ (why?). A method trying to fit a complex model to such data will necessarily do worse than this simple predictor, and the example is often used as a validity check of a new method.*

Except for the traditional considerations of bias and variance of a model, and the statistical uncertainty associated with estimating parameters, other issues include the following:

- Initialization of the parameters. If the initial guess of the parameters is not adequate, the recursive algorithm might take much samples before correcting this (transient effect).
- Forgetting Factor. The choice of a forgetting factor makes a trade-off between flexibility and accuracy.

- Window. If the window used for estimating then one must decide on how many samples are used for estimating at a certain instance t.
- Stability of the estimate. If the algorithm at hand is not well-tuned to the task at hand, it may display diverging estimates. This is clearly undesirable, and some algorithms go with guarantees that no such unstable behavior can occur.
- Gain. A typical parameter which needs t be tuned concerns the size of the update made at a new sample. If the gain is too low, a resulting algorithm will not converge fastly. If the gain is too large, one may risk unstable behavior.

In order to check wether a recursive identification is well-tuned for a certain application, it is instrumental to monitor closely the online behavior of the method, and to make appropriate graphical illustrations of the method.