Mikael Sternad 2003-03-24

Project Courses in Process Control and Adaptive Signal Processing

Recursive ARX identification/adaptive modeling

In previous lectures, we discussed methods for estimating linear black-box models for dynamic systems. These methods were based on *blocks* of measured input-output data. We discussed time-invariant models for single-output systems and the model adjustment was performed by *prediction error methods.* Various discrete-time model structures were expressed as one-step predictors, and the sum of squared prediction errors was minimized for the whole data block [3, 2].

An alternative to such block methods are discussed here: *recursive methods.* These algorithms modify a pre-existing model sample by sample, as new measurements become available [3, 4, 5]. Recursive adaptation algorithms can be expressed as time-varying filters, that modify the model by filtering the prediction errors. If the prediction error is small, the model remains unchanged. If it is significant, the model parameters are adjusted so that the error is decreased. Such time-varying self-adjusting models are called *adaptive models.*

To simplify the discussion, we here specialize the model structure to a scalar linear regression model of ARX type, for possibly complex-valued scalar data $y(t)$.

Given data $\{y(i), u(i)\}_{i=1}^{t}$ we are to adjust a time-varying linear regression model

$$y(t) = \varphi^H(t)\theta(t) + \varepsilon(t) \tag{1}$$

where the first right-hand term represents a one-step predictor for the scalar $y(t)$

$$\hat{y}(t|t-1, \theta(t)) = \varphi^H(t)\theta(t) \quad . \tag{2}$$

Here, $(\cdot)^H$ denotes Hermitian transpose (complex-conjugate transpose), while $(\cdot)^*$ will denote the complex conjugate of complex scalars. Thus, $(\cdot)^H = (\cdot)^{*T}$. Furthermore,

$$\begin{aligned} \theta(t) &= (\ a_1(t) \dots a_{na}(t)\ b_0(t) \dots b_{nb}(t)\ )^T & (3) \\ \varphi^H(t) &= (\ -y(t-1) \dots - y(t-na)\ u(t-k) \dots u(t-k-nb)\ ) \quad . & (4) \end{aligned}$$

The discrete time index t is used in $\varphi^H(t)$ to indicate dependence of data up to time t. In signal processing applications, we normally assume models with possibly no delay, $k \geq 0$. This model corresponds to a time-varying linear autoregressive model with external input (ARX model)

$$\hat{A}_t(q^{-1})y(t) = \hat{B}_t(q^{-1})u(t-k) + \varepsilon(t) \quad . \tag{5}$$

The one-step predictor (2) corresponds to

$$\hat{y}(t|t-1, \theta(t)) = [1 - \hat{A}_t(q^{-1})]y(t) + \hat{B}_t(q^{-1})u(t-k) \tag{6}$$

and

$$y(t) - \hat{y}(t|t-1, \theta(t)) = \varepsilon(t) \quad .$$

The model residual $\varepsilon(t)$ in (6) thus equals the one-step prediction error for this particular model structure. Special cases of (1)- (6) include FIR models ($na = 0$), all pole models ($nb = 0, \hat{B}_t(q^{-1}) = b_0(t)$) and autoregressive models of time-series ($\hat{B}_t(q^{-1}) = 0, u(t-i) = 0$).

We now assume a stochastic framework, in which the measurements $y(t)$ are regarded as stochastic variables. Repeated experiments, each performed by using the same input sequence $\{u(i)\}_{i=1}^{t}$, are thus expected to result in an ensemble (a set)
erent output trajectories. This means that the residual $\varepsilon(t)$ in the model (2),
regarded as a stochastic process. In a stochastic framework, an minimum
square error adjustment of the parameter vector $\theta(t)$ would be obtained by
izing the loss function

$$J(\theta) \triangleq \frac{1}{2}\, \mathrm{E}\, |\varepsilon(t,\theta)|^2 = \frac{1}{2}\, \mathrm{E}\, \varepsilon(t,\theta)^H \varepsilon(t,\theta) \quad , \tag{7}$$

where $\mathrm{E}\,(\cdot)$ denotes an ensemble average over all stochastic variables affecting the regressors $\{y(i)\}_{i=t-na}^{t-1}$ and the measurement $y(t)$. (The input $u(t)$ is regarded as deterministic and known.)

However, we are at this point faced with three complications.

1. The ensemble average $\mathrm{E}\,(\cdot)$ in (7) cannot be measured directly and exactly. We do not, in general, run an ensemble of experiments. A common way out of this dilemma is to *assume ergodicity*: ensemble averages equal time averages. We may then keep the parameter $\theta(t)$ unchanged for a time interval and approximate the ensemble average by a time average over this interval. Time averaging has been investigated in e.g. [5]. Interestingly, turns out that not much is gained by using long averaging intervals, compared to very short intervals. This phenomenon will be utilized in the algorithms that are discussed below.

2. The underlying system that generates our data could have time-varying properties. That possibility is in fact a main motivation for the use of recursive adaptation. If no time-variability is expected, we might just as well use block methods!

   For time-varying systems, we can *not*, without further assumptions, determine the sequence of parameter vectors $\theta(t)$ uniquely from a sequence of measurements

   $$y(1) = \varphi^H(1)\theta(1) + \varepsilon(1) \ , \ y(2) = \varphi^H(2)\theta(2) + \varepsilon(2) \ , \ \ldots \quad .$$

   This is not possible even in the noise-free case, unless the model has only one single parameter! We would end up with a set of unknowns $\theta(1)\ \theta(2) \ldots$ with more elements than the available measurements $y(1), y(2) \ldots$. To avoid this dilemma, assumptions must be introduced on the relationship between parameter vectors $\theta(t)$ and $\theta(\tau)$ at different points in time.

Here, we will use very simple assumptions on time variability, that turn out to be adequate for the Process Control course. We will assume the optimal parameter adjustment to *change only slowly, with relatively constant speed.* A more systematic methodology, based on Kalman estimation, will be introduced in later lectures within the Adaptive Signal Processing course.

3. The ARX model structure (5) may be inappropriate and/or the selected degrees na, nb may be too low. If the model structure is not sufficiently flexible, then the prediction error will remain correlated with the regressors.

   We must then modify or extend the parameter space, by increasing the model orders, by testing other model structures such as output error or ARMAX models, or by introducing models that are nonlinear in the signals or in the parameters. We will encounter problems with the ARX model structure within the Indirect Adaptive Control project in the Process Control course. There, the model structure will have to be extended to an ARMAX model structure
$$\hat{A}_t(q^{-1})y(t) = \hat{B}_t(q^{-1})u(t-k) + \hat{C}_t(q^{-1})\varepsilon(t) \quad .$$

A recursive algorithm for adjusting $\theta(t)$ can now be thought of as a numerical search for the point $\theta(t)$ in parameter space that minimizes J, defined by (7). For an ideal adjustment, the error $\varepsilon(t)$ would be white, zero mean and uncorrelated with the regressors. With a model adjusted in this way, no further improvement could be attained without new data.

Recursive algorithms for adjusting dynamic models are designed to start with any prior information that provides an initial model, with parameter vector $\theta(0)$. They then compute adjustment vectors $\Delta\theta(t)$
$$\theta(t+1) = \theta(t) + \Delta\theta(t) \tag{8}$$
designed to decrease the value of J, on average.

Among minimization schemes, it is natural to first consider the steepest descent algorithm, in which the update steps $\Delta\theta(t)$ are taken in the negative gradient direction. The gradient of a function $J(\theta)$ of a possibly complex-valued column vector $\theta = x + jy$ of dimension m is defined as the row vector
$$\nabla J = \left(\frac{\partial J}{\partial x_1} + \frac{\partial J}{\partial y_1} \quad \dots \quad \frac{\partial J}{\partial x_m} + \frac{\partial J}{\partial y_m} \right) \tag{9}$$

The gradient vector is perpendicular to the level curves of $J(\theta)$ at each point θ. For a given (fixed) parameter vector $\theta(t) = \theta$, the gradient vector of the criterion (7) can be shown to be given by
$$\begin{aligned}\nabla J = 2\left(\frac{\partial J(\theta)}{\partial \theta^*}\right)^T &= 2\left(\frac{\partial}{\partial \theta^*}\frac{1}{2}\left[\mathrm{E}\,(y(t) - \varphi^H(t)\theta)^*(y(t) - \varphi^H(t)\theta)\right]\right)^T \\ &= 2\mathrm{E}\left(\frac{\partial(-\varphi^H(t)\theta)^*}{\partial\theta^*}\right)^T \underbrace{(y(t) - \varphi^H(t)\theta)}_{\varepsilon(t)} = -2\mathrm{E}\,\varphi(t)\varepsilon(t) \quad ,\end{aligned} \tag{10}$$

where we used $[\varphi^H(t)]^{*T} = \varphi(t)$ in the last equality. The first equality relates the gradient to the so-called *conjugate derivative* of $J(\theta)$ (12), a relation explained in detail in Appendix B of [4].[1] When considering more advanced optimization schemes than steepest descent, the Hessian, i.e the matrix of second derivatives at the point θ, becomes of interest. A matrix proportional to the Hessian can be shown to be obtained by operating with (11) on the conjugate derivative:

$$\frac{\partial}{\partial\theta}\left(\frac{\partial J(\theta)}{\partial\theta^*}\right) = \frac{\partial}{\partial\theta}\left[-\mathrm{E}\,\varphi(t)(y(t) - \varphi^H(t)\theta)\right] = \mathrm{E}\,\varphi(t)\varphi^H(t) \quad . \tag{13}$$

Thus, the Hessian matrix will for linear regression ARX models equal the regressor covariance matrix. (This will not be true in more general cases.)

If the gradient (10) were measurable directly, we could construct a recursive algorithm that decreases the criterion J by changing θ along the negative gradient direction in parameter space:

$$\theta(t+1) = \theta(t) + \mu(t)\mathrm{E}\{\varphi(t)\varepsilon(t)\} \;=\; \theta(t) + \mu(t)\mathrm{E}\{\varphi(t)[\,y(t) - \varphi^H(t)\theta(t)]\} \;. \tag{14}$$

Here, we have introduced a scalar and possibly time-varying step length parameter $\mu(t)$ that scales the steps taken in the negative gradient direction. The step length could be tuned differently for different elements of $\theta(t)$. We then substitute $\mu(t)$ by a diagonal matrix, with possibly unequal diagonal elements.

A method that provides much faster initial convergence towards the optimum is the **Newton method:**

$$\theta(t+1) = \theta(t) + \mu(t)\left(\mathrm{E}\,\varphi(t)\varphi(t)^H\right)^{-1}\mathrm{E}\,\varphi(t)\varepsilon(t) \quad . \tag{15}$$

The instantaneous negative gradient $\mathrm{E}\,\varphi(t)\varepsilon(t)$ has here been multiplied by the inverse Hessian matrix (13), which equals the inverse regressor covariance matrix. This matrix will shift the search direction from a gradient direction to a Newton direction, which aims directly towards the minimum. The regressor covariance matrix

$$\mathbf{R} \triangleq \mathrm{E}\,\varphi(t)\varphi(t)^H \tag{16}$$

will be invertible (have full rank) if the input sequence $u(t)$ is persistently exciting. This condition is fulfilled if $u(t)$ has nonzero spectral density at least at

[1]Functions of complex variables have unique derivatives only if the Cauchy-Riemann equations are fulfilled, see e.g. [7]. This will *not* be the case for $J(\theta)$ at the origin $\theta = 0$. Therefore, there exist several possible derivative vector functions that can be defined based on the derivatives with respect to real and imaginary parts, and which all have reasonable properties. One of them is

$$\frac{\partial}{\partial\theta} = \frac{1}{2}\left(\frac{\partial}{\partial x_1} - \frac{\partial}{\partial y_1} \quad \cdots \quad \frac{\partial}{\partial x_m} - \frac{\partial}{\partial y_m}\right)^T \tag{11}$$

and another one is the conjugate derivative

$$\frac{\partial}{\partial\theta^*} = \frac{1}{2}\left(\frac{\partial}{\partial x_1} + \frac{\partial}{\partial y_1} \quad \cdots \quad \frac{\partial}{\partial x_m} + \frac{\partial}{\partial y_m}\right)^T \tag{12}$$

The relation of the gradient (9) to the conjugate derivative is evident.

$n\theta = na + nb + 1$ frequencies.

We now need estimates of the (ensemble) averages appearing in (10) and (13). As mentioned earlier, a time average may be used if ergodicity is assumed. An extremely simple but crude estimate of the gradient, which works surprisingly well, is to use *the instantaneous value* as approximation of the average:

$$\mathrm{E}\,\varphi(t)\varepsilon(t) \approx \varphi(t)\varepsilon(t) \quad . \tag{17}$$

When (17) is used in the gradient law (14), we obtain the LMS adaptation law [5, 6, 4]

$$\theta(t+1) = \theta(t) + \mu(t)\,\underbrace{\varphi(t)(y(t) - \varphi^H(t)\theta(t))}_{\text{Approx. negative gradient}} \quad . \tag{18}$$

This adaptation rule was introduced by Widrow and Hoff around 1960. It is effective in many applications and it is still widely used.

An estimate of the Hessian matrix, which is completely accurate only when the system is time-invariant and if the parameter vector $\theta(t)$ is held fixed, is obtained by the time average:

$$\mathrm{E}\,\varphi(t)\varphi(t)^H \approx \frac{1}{t}\sum_{\tau=0}^{t}\varphi(\tau)\varphi(\tau)^H \triangleq \frac{1}{t}\bar{\mathbf{R}}(t) \quad . \tag{19}$$

This sum can be updated recursively,

$$\bar{\mathbf{R}}(t) = \bar{\mathbf{R}}(t-1) + \varphi(t)\varphi(t)^H \quad . \tag{20}$$

If we now use the approximations (17) and (19) in the Newton adaptation scheme (15) and if a decreasing step length

$$\mu(t) = \frac{1}{t} \quad ,$$

is introduced, then the recursive least squares algorithm (RLS) is obtained:

$$\theta(t+1) = \theta(t) + \frac{1}{t}t\,\bar{\mathbf{R}}(t)^{-1}\varphi(t)\,\underbrace{(y(t) - \varphi^H(t)\theta(t))}_{\varepsilon(t)} \quad . \tag{21}$$

Thus, the Recursive Least Squares method for adapting ARX models can be seen as a stochastic Newton method which approaches the optimum with decreasing step length.

An alternative way of deriving (21) is to start from the ordinary off-line least squares estimate for the model (5) and rewrite it in recursive form. See p7-9 in [8].

The presence of the inverse of $\bar{\mathbf{R}}(t)$ in (21) is inconvenient, since this inverse would have to be computed at each model update. However, instead of updating the estimated regressor covariance matrix $\bar{\mathbf{R}}(t)$ via (19), we may directly update the inverse which appears in (21). The inverse may be expressed as

$$\bar{\mathbf{R}}(t)^{-1} = \left[\bar{\mathbf{R}}(t-1) + \varphi(t)\varphi^H(t)\right]^{-1} \triangleq \mathbf{P}(t) \quad . \tag{22}$$

A direct update of (22) is accomplished by using the *Matrix Inversion Lemma*: If $\mathbf{A}$ and $\mathbf{B}$ are matrices of appropriate dimensions, then

$$\left[\mathbf{A}+\mathbf{B}\mathbf{B}^H\right]^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{B}[\mathbf{I}+\mathbf{B}^H\mathbf{A}^{-1}\mathbf{B}]^{-1}\mathbf{B}^H\mathbf{A}^{-1} \qquad (23)$$

The use of (23) in (22) with $\mathbf{A}^{-1}=\bar{\mathbf{R}}(t-1)^{-1}=\mathbf{P}(t-1)$ gives the so-called *Riccati equation* for updating $\mathbf{P}(t)$,

$$\mathbf{P}(t)=\mathbf{P}(t-1)-\frac{\mathbf{P}(t-1)\varphi(t)\varphi^H(t)\mathbf{P}(t-1)}{1+\varphi^H(t)\mathbf{P}(t-1)\varphi(t)} \qquad (24)$$

and the RLS parameter update equation (21) is

$$\theta(t+1)=\theta(t)+\mathbf{P}(t)\underbrace{\varphi(t)(y(t)-\varphi^H(t)\theta(t))}_{\text{Approx. negative gradient}}\quad . \qquad (25)$$

The point in using the update (24) instead of (20) is that a matrix inversion is then avoided. The denominator of the right-hand term in (24) is scalar, for scalar measurements $y(t)$.

Initially, $\mathbf{P}(t)$ will be uncertain and sensitive to new data, since it is based on few samples. As more data is accumulated, the estimate improves and stabilizes. The matrix $\mathbf{P}(t)$ will be an estimate of the inverse Hessian. When $t\rightarrow\infty$, a comparison with the theory for off-line LS estimates shows that $\mathbf{P}(t)\mathrm{E}\,|\varepsilon(t)|^2$ is an estimate of the *parameter error covariance matrix*, when a model with correct structure is adapted and when a time-invariant system generates $y(t)$.

This provides an interpretation that aids the selection of an initial value for $\mathbf{P}(0)$: If the initial model parameter vector $\theta(0)$ is reasonably accurate, then $\mathbf{P}(0)$ should be selected to approximate the covariance matrix of the initial parameter error. On the other hand, if no information on a good initial model is present, $\theta(0)$ might as well be initialized as a zero vector. The initial $\mathbf{P}$-matrix is then typically selected as a diagonal matrix with large diagonal elements. The approximate negative gradient in (25) will then initially be multiplied by high gains for all parameters. This results in a fast initial convergence. As the model improves, the norm of $\mathbf{P}(t)$ will decrease quickly, and so will the gain in the RLS algorithm.

However, the RLS algorithm (24), (25) is essentially just a recursive approximation of the off-line (batch) least squares estimate, designed for a time-invariant system. As the data batch increases in length, each new prediction errors will have less and less influence compared to the already accumulated information. The step length goes towards zero and the ability to react to new information vanishes.

If the underlying system is time-varying, this property is clearly unsatisfactory. The best approach would then be to design an adaptation law that is tuned to the expected types of time-variations. A suboptimal approach, which often (but far from always) gives acceptable results for slow and steady parameter drifts is to modify the RLS law. The modification must prevent the adaptation gain from converging towards zero. The most common modification is to substitute the average in the criterion (7) by a time-average which is *exponentially discounted*: old values are weighted less in the criterion:

$$J_t(\theta) = \sum_{\tau=1}^{t} \lambda^{t-\tau} \frac{1}{2} |\varepsilon(\tau)|^2 \tag{26}$$

An exponential data window with *forgetting factor* $0 \leq \lambda \leq 1$ corresponds approximately to the use of a rectangular data window of size

$$N \approx \frac{2}{1-\lambda} \quad .$$

The introduction of the modification (26) will leave the parameter update equation (25) unchanged. The Riccati update is however modified to

$$\mathbf{P}(t) = \frac{1}{\lambda}\left[\mathbf{P}(t-1) - \frac{\mathbf{P}(t-1)\varphi(t)\varphi^H(t)\mathbf{P}(t-1)}{\lambda + \varphi^H(t)\mathbf{P}(t-1)\varphi(t)}\right] \tag{27}$$

We have then obtained the Recursive Least Squares algorithm with exponential forgetting, or *exponentially weighted RLS*. The term $\mathbf{P}(t-1)/\lambda$ in (27) tends to blow up the matrix. It can be seen as representing the increase in parameter uncertainty due to forgetting of old data. The last (negative) right-hand term balances this increase. It represents the decrease in uncertainty due to new information via the latest prediction error.[2]

The choice of window length (forgetting factor) will be a compromise between tracking ability and noise sensitivity. In some problems such as adaptive control, the use of a time-varying forgetting factor improves the performance. The compromise between tracking ability and noise sensitivity will in a similar way affect the choice of the step length parameter $\mu(t)$ in the LMS algorithm (18).

The RLS algorithm will provide a much faster initial convergence towards the optimum than LMS. However, it should be emphasized that neither LMS nor RLS can be considered consistently superior relative to the other when tracking parameters of time-varying systems.

Practical aspects on the use of RLS algorithms are discussed on separate slides.

[2]Note, however, that when $\lambda < 1$, a strict interpretation of $\mathbf{P}(t)$ in the RLS algorithm as a parameter error covariance matrix is no longer possible.

References

[1] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification.* Prentice-Hall 1983 (out of print).

[2] L. Ljung *System Identification: Theory for the User.* Second Ed. Prentice-Hall 1999.

[3] T. Söderström and P. Stoica, *System Identification* Prentice Hall International, 1989.

[4] S. Haykin *Adaptive Filter Theory* Third ed. Prentice-Hall 1996.

[5] B. Widrow and S. D. Stearns, *Adaptive Signal Processing.* Prentice-Hall 1985

[6] M.H. Hayes, *Statistical Digital Signal Processing and Modeling.* Wiley, New York, 1996.

[7] N. Levinson and R.M. Redheffer *Complex Variables* Holden-Day, San Francisco, 1970.

[8] A. Ahlen and M. Sternad *Introduktion till adaptiv regrering.* Kompendium, Uppsala Universitet 1988.