ISSN 0923-0408

INFORMATION AND DECISION TECHNOLOGIES

Volume 19 Number 1 1993

IDTEEI 19(1) (1993) 1–76

NORTH-HOLLAND

Recursive sliding window estimators

A. Pouliezos

Department of Production and Management Engineering, Technical University of Crete, University Campus, Kounoupidiana, 731 00 Chania, Greece

Received May 1992
Revised April 1993
Communicated by S. Tzafestas

Sliding window estimation is very useful in certain areas of automatic control. In this paper recursive sliding window versions of popular estimation algorithms are presented which greatly reduce computational load and response time, thus making them suitable for on-line applications. An illustrative example demonstrates the usefulness of the algorithms.

1. Introduction

Iterative parameter estimation methods form the basis for the solution of many problems in control practice. State estimation in noisy environments and fault detection are two such areas. In the linear world the *least squares estimator* (LSE) with its many variations/interpretations is widely used. Its simplest scalar output non-recursive form for n consecutive samples is,

$$\hat{\boldsymbol{\theta}} = [\boldsymbol{U}^T\boldsymbol{U}]^{-1}\boldsymbol{U}^T\boldsymbol{y}\,, \tag{1a}$$

where

$$y(k) = \boldsymbol{u}^T(k)\boldsymbol{\theta} + e(k)\,, \quad k = 1, \ldots, n \tag{1b}$$

is the model of the process with $y(k)$ being the scalar observed variable/regressand/output, $\boldsymbol{u}(k)$ the $(m \times 1)$ explanatory variables/regressors/inputs, $\boldsymbol{\theta}$ the $(m \times 1)$ unknown parameter vector and $e(k)$ the error due to measurement noise and modelling inaccuracies.

Equation (1a) uses all available information, lumped in $\boldsymbol{U}$ and $\boldsymbol{y}$, to produce a one-shot estimate of $\boldsymbol{\theta}$. One form for a recursive version of (1) is given by

$$\boldsymbol{P}(k) = \boldsymbol{P}(k-1) - \boldsymbol{P}(k-1)\boldsymbol{u}(k)\boldsymbol{u}^T(k)\boldsymbol{P}(k-1)/(1 + \boldsymbol{u}^T(k)\boldsymbol{P}(k-1)\boldsymbol{u}(k))\,, \tag{2a}$$

$$\hat{\boldsymbol{\theta}}(k) = \hat{\boldsymbol{\theta}}(k-1) + \boldsymbol{P}(k)\boldsymbol{u}(k)[y(k) - \boldsymbol{u}^T(k)\hat{\boldsymbol{\theta}}(k-1)]\,. \tag{2b}$$

Starting points for equations (2a), (2b) are an a priori estimate $\hat{\boldsymbol{\theta}}(0)$ of $\boldsymbol{\theta}$ and a $\boldsymbol{P}(0)$ reflecting the confidence in this initial estimate. It is common practice to take,

$$\hat{\boldsymbol{\theta}}(0) = \boldsymbol{0}\,, \quad \boldsymbol{P}(0) = \rho\boldsymbol{I}\,; \quad \rho \gg 0\,.$$

North-Holland
Information and Decision Technologies 19 (1993) 19–30

Equations (2a), (2b) may be expressed in other forms, each one exemplifying a particular aspect of the algorithm. For example, if (2b) is rewritten as

$$\hat{\boldsymbol{\theta}}(k) = [\boldsymbol{I} - \boldsymbol{P}(k)\boldsymbol{u}(k)\boldsymbol{u}^T(k)]\hat{\boldsymbol{\theta}}(k-1) + \boldsymbol{P}(k)\boldsymbol{u}(k)y(k)\,,$$

the nature of the estimate updating is pointed out.

The vector output versions of (1a), (1b), (2a), (2b) are given in Norton [4], as

$$\hat{\boldsymbol{x}}(k) = [\mathcal{H}^T(k)\mathcal{H}(k)]^{-1}\mathcal{H}^T(k)\mathcal{Y}(k)\,, \tag{3a}$$

$$\mathcal{Y}(k) = \mathcal{H}(k)\boldsymbol{x} + \mathcal{V}(k)\,, \tag{3b}$$

$$\boldsymbol{P}(k) = \boldsymbol{P}(k-1) - \boldsymbol{P}(k-1)\boldsymbol{H}^T(k)[\boldsymbol{I} + \boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^T(k)]^{-1}\boldsymbol{H}(k)\boldsymbol{P}(k-1)\,, \tag{4a}$$

$$\hat{\boldsymbol{x}}(k) = \hat{\boldsymbol{x}}(k-1) + \boldsymbol{P}(k)\boldsymbol{H}^T(k)[\boldsymbol{y}(k) - \boldsymbol{H}(k)\hat{\boldsymbol{x}}(k-1)]\,, \tag{4b}$$

where

$$\mathcal{V}(k) = \begin{bmatrix} \boldsymbol{v}(1) \\ \cdot\cdot \\ \cdot\cdot \\ \boldsymbol{v}(k) \end{bmatrix}, \quad \mathcal{Y}(k) = \begin{bmatrix} \boldsymbol{y}(1) \\ \cdot\cdot \\ \cdot\cdot \\ \boldsymbol{y}(k) \end{bmatrix}, \quad \mathcal{H}(k) = \begin{bmatrix} \boldsymbol{H}(1) \\ \cdot\cdot \\ \cdot\cdot \\ \boldsymbol{H}(k) \end{bmatrix},$$

and

$$\boldsymbol{P}(k) = [\boldsymbol{H}^T(k)\boldsymbol{H}(k)]^{-1}$$

is the covariance of $\hat{\boldsymbol{x}}(k)$. Initialisation of (4a), (4b) is performed analogously to initialisation for (2a), (2b).

The recursive expressions (2a), (2b), (4a), (4b) utilize the whole set of available information, providing optimum estimates. There are cases however, where this is either undesirable or ineffective. These are:

(1) Situations where it is desirable to track time-varying parameters. In such cases the model is described as

$$y(k) = \boldsymbol{u}^T(k)\boldsymbol{\theta}(k) + e(k)\,, \quad k = 1, \ldots, n\,. \tag{5}$$

The inability of the preceding algorithms to track time variation of parameters stems from the fact that the estimate error covariance matrix $\boldsymbol{P}(k)$ settles to a small value (meaning great confidence to estimates) and therefore new information is not taken into account quickly. There are a number of different approaches to overcome this problem:

(1a) A *weighting matrix* $\boldsymbol{W}$ (also called *forgetting factor*) can be used, resulting in the weighted least squares estimate (WLSE),

$$\hat{\boldsymbol{\theta}}_W = [\boldsymbol{U}^T\boldsymbol{W}\boldsymbol{U}]^{-1}\boldsymbol{U}^T\boldsymbol{W}\boldsymbol{y}\,. \tag{6}$$

Its recursive counterpart for diagonal $\boldsymbol{W}$ is

$$\boldsymbol{P}(k) = \boldsymbol{P}(k-1) - \boldsymbol{P}(k-1)\boldsymbol{u}(k)\boldsymbol{u}^T(k)\boldsymbol{P}(k-1)[w(k)^{-1} + \boldsymbol{u}^T(k)\boldsymbol{P}(k+1)\boldsymbol{u}(k)]^{-1}\,, \tag{6a}$$

$$\hat{\boldsymbol{\theta}}(k) = \hat{\boldsymbol{\theta}}(k-1) + \boldsymbol{P}(k)\boldsymbol{u}(k)w(k)[y(k) - \boldsymbol{u}^T(k)\hat{\boldsymbol{\theta}}(k-1)]\,, \tag{6b}$$

where $w(k)$ is the corresponding diagonal element of $\boldsymbol{W}$, usually increasing with time. In this way the estimator gain (through the error covariance) is prohibited from getting too small, and thus enabling the

tracking of the parameter variations. Typical values for $(1/w(k))$ are 0.95, 0.99. Alternatively, in a Markovian context, $\boldsymbol{W}$ may be interpreted as $\boldsymbol{R}^{-1}$, where $\boldsymbol{R} = \text{cov}\,\{\boldsymbol{e}(k)\}$ is usually diagonal.

(1b) Goodwin and Sin [2], proposed a *covariance resetting* method which has the same effect, i.e. $\boldsymbol{P}(k)$ is prevented from getting too small. This is accomplished by resetting it to a fixed large value whenever a measure of its size, for example its trace, falls below a prespecified threshold.

(1c) Greater flexibility can be achieved by basing the estimator on an explicit model of the parameter variation. This leads to the idea of *state estimation* and *Kalman filtering* [3]. In this approach equation (5) is supplemented by

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + \boldsymbol{v}(k)\,, \tag{7}$$

where $\boldsymbol{v}(k)$ is white noise with $\text{cov}\,\{\boldsymbol{v}(k)\} = \boldsymbol{Q}$. The covariance matrix $\boldsymbol{Q}$ can be used to describe how fast the different components of $\boldsymbol{\theta}$ are expected to vary. Applying the Kalman filter equations to the system described by (5) and (7) yields

$$\boldsymbol{P}(k) = \boldsymbol{P}(k-1) - \boldsymbol{P}(k-1)\boldsymbol{u}(k)\boldsymbol{u}^T(k)\boldsymbol{P}(k-1)[w(k)^{-1} + \boldsymbol{u}^T(k)\boldsymbol{P}(k+1)\boldsymbol{u}(k)]^{-1} + \boldsymbol{Q}\,, \tag{8a}$$

$$\hat{\boldsymbol{\theta}}(k) = \hat{\boldsymbol{\theta}}(k-1) + \boldsymbol{P}(k)\boldsymbol{u}(k)w(k)[y(k) - \boldsymbol{u}^T(k)\hat{\boldsymbol{\theta}}(k-1)]\,. \tag{8a}$$

Observe that in this context $\boldsymbol{P}(k)$ is prevented from getting small by replacing it with,

$$\boldsymbol{M}(k) = \boldsymbol{P}(k) + \boldsymbol{Q}\,, \quad \boldsymbol{Q} > \boldsymbol{0}\,.$$

(2) In *fault detection schemes* based on parameter estimation [5] or Kalman filtering techniques [8]. In this framework the system is described by equation (5) but time variation of the parameters is modeled to occur at unspecified times during the process operation as:

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + \sigma_{k,\delta}\nu\,, \tag{9}$$

where δ denotes the change time and $\sigma_{k,\delta}$ is the unit step which is zero if $k < \delta$ and 1 otherwise. If no change (fault) occurs, δ is infinite.

Now, since the algorithms defined by (2a), (2b) or (4a), (4b) have infinite memory, a system fault may take a prohibitively long time to be detected if these estimators are used as primary fault detectors. One way out of this problem is to use a sliding window of data values, thus making the detection mechanism more responsive to the onset of failure while maintaining good estimation properties as proposed by Basseville [1], Tanaka and Müller [7] and Willsky [8].

In the rest of the paper sliding window equivalents of the above recursions are developed, which reduce the amount of computations needed, thus making them especially suitable for incorporation in online computerised fault detection systems. In addition the estimates are optimal with respect to the current data window, providing a more attractive alternative to the algorithms described by (6a), (6b) or (8a), (8b). Furthermore, sliding window estimators require the specification of only one design parameter, namely window length, thus providing a simple solution to the detection problem.

2. The algorithms

The algorithms are developed in the same sequence as outlined above. The scalar output case described by

$$y(k) = \boldsymbol{u}^T(k)\boldsymbol{\theta} + e(k)$$

is considered first. In the usual way, define

$$\boldsymbol{U} = \begin{bmatrix} u_1(1) & \cdots & u_m(1) \\ \vdots & & \vdots \\ u_1(k) & \cdots & u_m(k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}^T(1) \\ \vdots \\ \boldsymbol{u}^T(k) \end{bmatrix},$$

and

$$\boldsymbol{y} = [y(1) \quad \cdots \quad y(k)]^T .$$

Furthermore, for a moving window of length n_w, define

$$\boldsymbol{U}_k = \begin{bmatrix} \boldsymbol{u}^T(k-n_w+1) \\ \vdots \\ \boldsymbol{u}^T(k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}^T(k-n_w+1) \\ \hline \boldsymbol{U}(k, k-n_w+2) \end{bmatrix}.$$

Then

$$\boldsymbol{U}_{k+1} = \begin{bmatrix} \boldsymbol{u}^T(k-n_w+2) \\ \hline \vdots \\ \hline \boldsymbol{u}^T(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}(k, k-n_w+2) \\ \hline \boldsymbol{u}^T(k+1) \end{bmatrix}.$$

The iteration for $\boldsymbol{U}_k^T\boldsymbol{U}_k$ is considered first.

$$\begin{aligned} \boldsymbol{U}_k^T\boldsymbol{U}_k &= [\boldsymbol{u}(k-n_w+1) \,|\, \boldsymbol{U}^T(k, k-n_w+2)] \begin{bmatrix} \boldsymbol{u}^T(k-n_w+1) \\ \hline \boldsymbol{U}(k, k-n_w+2) \end{bmatrix} \\ &= \boldsymbol{u}(k-n_w+1)\boldsymbol{u}^T(k-n_w+1) + \boldsymbol{U}^T(k, k-n_w+2)\boldsymbol{U}(k, k-n_w+2) , \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{U}_{k+1}^T\boldsymbol{U}_{k+1} &= [\boldsymbol{U}^T(k, k-n_w+2) \,|\, \boldsymbol{u}(k+1)] \begin{bmatrix} \boldsymbol{U}(k, k-n_w+2) \\ \hline \boldsymbol{u}^T(k+1) \end{bmatrix} \\ &= \boldsymbol{u}(k+1)\boldsymbol{u}^T(k+1) + \boldsymbol{U}^T(k, k-n_w+2)\boldsymbol{U}(k, k-n_w+2) \\ &= \boldsymbol{U}_k^T\boldsymbol{U}_k + \boldsymbol{u}(k+1)\boldsymbol{u}^T(k+1) - \boldsymbol{u}(k-n_w+1)\boldsymbol{u}^T(k-n_w+1) \\ &= \boldsymbol{U}_k^T\boldsymbol{U}_k + \boldsymbol{\Gamma}(k+1) , \end{aligned}$$

where

$$\boldsymbol{\Gamma}(k+1) = \boldsymbol{u}(k+1)\boldsymbol{u}^T(k+1) - \boldsymbol{u}(k-n_w+1)\boldsymbol{u}^T(k-n_w+1) .$$

Secondly, the iteration for $\boldsymbol{U}_k^T\boldsymbol{y}_k$ is considered. Define,

$$\boldsymbol{y}_k = \begin{bmatrix} y(k-n_w+1) \\ \vdots \\ y(k) \end{bmatrix} = \begin{bmatrix} y(k-n_w+1) \\ \hline y(k, k-n_w+2) \end{bmatrix},$$

then

$$\boldsymbol{y}_{k+1} = \begin{bmatrix} y(k-n_w+2) \\ \vdots \\ y(k+1) \end{bmatrix} = \begin{bmatrix} y(k, k-n_w+2) \\ \hline y(k+1) \end{bmatrix}.$$

Hence

$$\boldsymbol{U}_k^T \boldsymbol{y}_k = [\boldsymbol{u}(k-n_w+1) \,|\, \boldsymbol{U}^T(k, k-n_w+2)] \left[\begin{array}{c} y(k-n_w+1) \\ \hline \boldsymbol{y}(k, k-n_w+2) \end{array}\right]$$

$$= \boldsymbol{u}(k-n_w+1) y(k-n_w+1) + \boldsymbol{U}^T(k, k-n_w+2) \boldsymbol{y}(k, k-n_w+2) ,$$

and

$$\boldsymbol{U}_{k+1}^T \boldsymbol{y}_{k+1} = [\boldsymbol{U}^T(k, k-n_w+2) \,|\, \boldsymbol{u}(k+1)] \left[\begin{array}{c} \boldsymbol{y}(k, k-n_w+2) \\ \hline y(k+1) \end{array}\right]$$

$$= \boldsymbol{u}(k+1) y(k+1) + \boldsymbol{U}^T(k, k-n_w+2) \boldsymbol{y}(k, k-n_w+2)$$

$$= \boldsymbol{U}_k^T \boldsymbol{y}_k + \boldsymbol{u}(k+1) y(k+1) - \boldsymbol{u}(k-n_w+1) y(k-n_w+1)$$

$$= \boldsymbol{U}_k^T \boldsymbol{y}_k + \boldsymbol{\delta}(k+1) , \tag{10}$$

where

$$\boldsymbol{\delta}(k+1) = \boldsymbol{u}(k+1) y(k+1) - \boldsymbol{u}(k-n_w+1) y(k-n_w+1) .$$

Now

$$\hat{\boldsymbol{\theta}}(k+1) = (\boldsymbol{U}_{k+1}^T \boldsymbol{U}_{k+1})^{-1} \boldsymbol{U}_{k+1}^T \boldsymbol{y}_{k+1} .$$

Defining

$$\boldsymbol{P}(k+1) = (\boldsymbol{U}_{k+1}^T \boldsymbol{U}_{k+1})^{-1} ,$$

which is the covariance of the estimate $\hat{\boldsymbol{\theta}}(k+1)$, we get

$$\hat{\boldsymbol{\theta}}(k+1) = \boldsymbol{P}(k+1)[\boldsymbol{U}_k^T \boldsymbol{y}_k + \boldsymbol{\delta}(k+1)]$$

$$= \boldsymbol{P}(k+1)[\boldsymbol{P}^{-1}(k) \hat{\boldsymbol{\theta}}(k) + \boldsymbol{\delta}(k+1)]$$

$$= \boldsymbol{P}(k+1)[(\boldsymbol{P}^{-1}(k+1) - \boldsymbol{\Gamma}(k+1)) \hat{\boldsymbol{\theta}}(k) + \boldsymbol{\delta}(k+1)]$$

$$= \hat{\boldsymbol{\theta}}(k) - \boldsymbol{P}(k+1)[\boldsymbol{\Gamma}(k+1) \hat{\boldsymbol{\theta}}(k) - \boldsymbol{\delta}(k+1)] , \tag{11}$$

and

$$\boldsymbol{P}^{-1}(k+1) = \boldsymbol{P}^{-1}(k) + \boldsymbol{\Gamma}(k+1) . \tag{12}$$

It should be remembered that $\hat{\boldsymbol{\theta}}(i)$ is estimated using information from the last n_w samples. Equations (11) and (12) form the sliding window least squares estimator (SWLSE). Note that in this simple case a further reduction of (12) is not needed since only one inversion is required. The reduction in speed is proportional to the length of the window since the dimensions of $\boldsymbol{P}$, $\boldsymbol{\Gamma}$ and $\boldsymbol{\delta}$ are independent of the window size (they are $m \times m$, $m \times m$ and $m \times 1$ respectively).

The improvement in speed over the classical batch sliding window LSE is shown in the operations count table for the scalar case (Table 1). No special methods for better performance of individual operations (matrix inversion) are taken into account, since these would apply equally well to both cases. It should be noted however that memory requirements are not reduced, since at any one time all the window values must be accessible. The scalar case considered may serve as a guideline for speed improvement in the vector versions that follow.

The sliding window weighted least squares estimator (SWWLSE) is similarly obtained.

For the vector output case, where

$$\boldsymbol{y}(k) = \boldsymbol{H}(k) \boldsymbol{x} + \boldsymbol{v}(k) ,$$

Table 1
Operations count for window size n_w (scalar case)

Recursive version			Batch		
	Additions	Multiplies		Additions	Multiplies
Estimate updating	$3m^2$	$4m^2+2m$	U^TU	$(n_w-1)m$	n_wm^2
Covariance updating	m^2	0	$[U^TU]^{-1}U^Ty$	$m^2+m(n_w-2)$	m^2+mn_w
Total	$4m^2$	$4m^2+2m$	Total	m^2+2mn_w-3m	$m^2(n_w+1)+mn_w$
Inversions		one $(m\times m)$	Inversions		one $(m\times m)$

define

$$\mathcal{H}=\begin{bmatrix}\boldsymbol{H}(1)\\ \vdots\\ \boldsymbol{H}(k)\end{bmatrix},\quad \mathcal{Y}=\begin{bmatrix}\boldsymbol{y}(1)\\ \vdots\\ \boldsymbol{y}(k)\end{bmatrix},$$

and for a moving window of length n_w

$$\mathcal{H}_k=\begin{bmatrix}\boldsymbol{H}(k-n_w+1)\\ \vdots\\ \boldsymbol{H}(k)\end{bmatrix}=\left[\begin{array}{c}\boldsymbol{H}(k-n_w+1)\\ \hline \mathcal{H}(k,k-n_w+2)\end{array}\right].$$

Then

$$\mathcal{H}_{k+1}=\begin{bmatrix}\boldsymbol{H}(k-n_w+2)\\ \vdots\\ \boldsymbol{H}(k+1)\end{bmatrix}=\left[\begin{array}{c}\mathcal{H}(k,k-n_w+2)\\ \hline \boldsymbol{H}(k+1)\end{array}\right],$$

and

$$\begin{aligned}\mathcal{H}_{k+1}^T\mathcal{H}_{k+1}&=[\mathcal{H}^T(k,k-n_w+2)\,|\,\boldsymbol{H}^T(k+1)]\left[\begin{array}{c}\mathcal{H}(k,k-n_w+2)\\ \hline \boldsymbol{H}(k+1)\end{array}\right]\\ &=\boldsymbol{H}^T(k+1)\boldsymbol{H}(k+1)+\mathcal{H}^T(k,k-n_w+2)\mathcal{H}(k,k-n_w+2)\\ &=\mathcal{H}_k^T\mathcal{H}_k+\boldsymbol{H}^T(k+1)\boldsymbol{H}(k+1)-\boldsymbol{H}^T(k-n_w+1)\boldsymbol{H}(k-n_w+1)\\ &=\mathcal{H}_k^T\mathcal{H}_k+\boldsymbol{G}(k+1),\end{aligned}$$

where

$$\boldsymbol{G}(k+1)=\boldsymbol{H}^T(k+1)\boldsymbol{H}(k+1)-\boldsymbol{H}^T(k-n_w+1)\boldsymbol{H}(k-n_w+1).$$

Analogously to the scalar case, it is obtained,

$$\mathcal{H}_{k+1}^T\mathcal{Y}_{k+1}=\mathcal{H}_k^T\mathcal{Y}_k+\boldsymbol{D}(k+1), \tag{13}$$

where

$$\boldsymbol{D}(k+1)=\boldsymbol{H}^T(k+1)\boldsymbol{y}(k+1)-\boldsymbol{H}^T(k-n_w+1)\boldsymbol{y}(k-n_w+1),$$

and finally

$$\hat{\boldsymbol{x}}(k+1)=\hat{\boldsymbol{x}}(k)-\boldsymbol{P}(k+1)[\boldsymbol{G}(k+1)\hat{\boldsymbol{x}}(k)-\boldsymbol{D}(k+1)], \tag{14}$$

$$\boldsymbol{P}^{-1}(k+1)=\boldsymbol{P}^{-1}(k)+\boldsymbol{G}(k+1). \tag{15}$$

Equations (14) and (15) form the vector equivalent of the SWLSE. A form similar to the Kalman filter equations is not possible to be obtained since the matrix $\boldsymbol{D}(k+1)$ cannot be guaranteed to be positive definite. However, the reduction in computation is still high.

For the general case where the errors are weighted by a non-diagonal matrix, the so called Markov estimate is given by equation (6a). In this case,

$$\mathscr{R}^{-1} = \begin{bmatrix} \boldsymbol{R}^{-1}(1) & & \\ & \boldsymbol{R}^{-1}(2) & \\ & & \ddots \\ & & & \boldsymbol{R}^{-1}(k) \end{bmatrix}.$$

Recall that in a Marcovian context the weights, possibly time-varying, are interpreted as the covariances of the error sequence

$$\boldsymbol{R}(k) = \operatorname{cov}(\boldsymbol{v}(k)) = \boldsymbol{W}^{-1}(k)$$

Define

$$\mathscr{R}_k^{-1} = \begin{bmatrix} \boldsymbol{R}^{-1}(k-n_w+1) & & \\ & \boldsymbol{R}^{-1}(k-n_w+2) & \\ & & \ddots \\ & & & \boldsymbol{R}^{-1}(k) \end{bmatrix} = \left[\begin{array}{c:c} \boldsymbol{R}^{-1}(k-n_w+1) & \\ \hdashline & \mathscr{R}(k, k-n_w+2) \end{array}\right],$$

and

$$\mathscr{R}_{k+1}^{-1} = \begin{bmatrix} \boldsymbol{R}^{-1}(k-n_w+2) & & \\ & \boldsymbol{R}^{-1}(k-n_w+3) & \\ & & \ddots \\ & & & \boldsymbol{R}^{-1}(k+1) \end{bmatrix} = \left[\begin{array}{c:c} \mathscr{R}(k, k-n_w+2) & \\ \hdashline & \boldsymbol{R}^{-1}(k+1) \end{array}\right].$$

The affected iterative equations are the ones involving the estimate covariance $\boldsymbol{P}$. They are calculated as follows:

$$\begin{aligned}(\mathscr{H}_{k+1}^T \mathscr{R}_{k+1}^{-1} \mathscr{H}_{k+1}) &= \mathscr{H}_{k+1}^T \mathscr{R}(k, k-n_w+2)\mathscr{H}_{k+1} + \mathscr{H}_{k+1}^T \boldsymbol{R}^{-1}(k+1)\mathscr{H}_{k+1} \\ &= \mathscr{H}_k^T \mathscr{R}_k^{-1} \mathscr{H}_k + \boldsymbol{H}^T(k+1)\boldsymbol{R}^{-1}(k+1)\boldsymbol{H}(k+1) \\ &\quad + \boldsymbol{H}^T(k-n_w+1)\boldsymbol{R}^{-1}(k-n_w+1)\boldsymbol{H}(k-n_w+1)\,.\end{aligned}$$

Therefore

$$\mathscr{P}_{k+1}^{-1} = \mathscr{P}_k^{-1} + \boldsymbol{H}^T(k+1)\boldsymbol{R}^{-1}(k+1)\boldsymbol{H}(k+1) + \boldsymbol{H}^T(k-n_w+1)\boldsymbol{R}^{-1}(k-n_w+1)\boldsymbol{H}(k-n_w+1)\,. \quad (16)$$

Equations (14) and (16) form the sliding window estimation formulae for the multivariable Markov estimate.

All the above estimation schemes are started using the corresponding one-shot formulae for the data of the starting window (first n_w measurements).

4. Illustrative example

To illustrate the effectiveness of the proposed method, consider the fluid pumping circuit of Fig. 1, which is to be monitored for possible leaks. The equations describing the dynamic behavior of this system are [6]:

$$\frac{dM_1}{dt} = c_1 M_1^2(t) + c_2 f(M_1, \omega)\,, \tag{17a}$$

$$\frac{M_1(t) - M_6(t)}{M_1(t)} = \sigma\,, \tag{17b}$$

where

$$c_1 = -\frac{\xi_1 + \frac{\lambda l_{1L}}{D} + \left(1 + \xi_2 + \frac{\lambda l_{L6}}{D}\right)(1-\sigma^2)}{2\rho(\sigma l_{1L} + l(1-\sigma))A}\,, \tag{18}$$

$$c_2 = \frac{A}{\sigma l_{1L} + l(1-\sigma)}\,, \tag{19}$$

and the physical parameters denote

$M_i(t)$: fluid flow rate at point i ($\mathrm{kg\,s^{-1}}$)
$\omega(t)$: pump rotational speed ($\mathrm{s^{-1}}$)
$f(M_1, \omega)$: characteristic of pump
ρ: fluid density ($\mathrm{kg/m^3}$)
λ: friction coefficient
σ: leak flow rate as percentage of M_1 (%)
D: diameter of pipe (m)
A: cross-sectional area of pipe ($\mathrm{m^2}$)
ξ_1: loss coefficient of valve positioned before the leak
ξ_2: loss coefficient of valve positioned after the leak
l: total length of pipe (m)
l_{1L}: length of pipe from point 1 to leak point (m)
l_{L6}: length of pipe from leak point to point 6 (m)

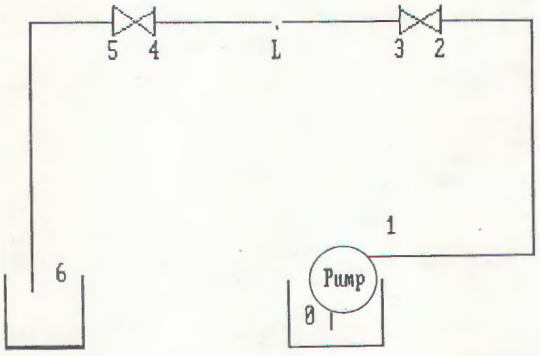

Fig. 1. Pumping system.

In particular, if the characteristic of the pump is expressed in the form

$$f(M_1(t), \omega(t)) = h_1 M_1^2(t) + h_2 M_1(t)\omega(t) + h_3 \omega^2(t)\,, \tag{20}$$

where h_i are appropriate known pump constants and $\omega(t)$ is taken as the input to the system, equations (17a)–(17b) can be conveniently written as,

$$\frac{\mathrm{d}M_1}{\mathrm{d}t} = [M_1^2(t) \quad \omega(t)M_1(t) \quad \omega^2(t)]\begin{bmatrix}\theta_{11}\\ \theta_{12}\\ \theta_{13}\end{bmatrix}, \tag{21a}$$

$$\frac{M_1(t) - M_6(t)}{M_1(t)} = \theta_{21}\,, \tag{21b}$$

where

$$\theta_{11} = -\frac{\xi_1 + \dfrac{\lambda l_{1L}}{D} + \left(1 + \xi_2 + \dfrac{\lambda l_{L6}}{D}\right)(1-\sigma^2)}{2\rho(\sigma l_{1L} + l(1-\sigma))A} + \frac{Ah_1}{\sigma l_{1L} + l(1-\sigma)}\,, \tag{22a}$$

$$\theta_{12} = \frac{Ah_2}{\sigma l_{1L} + l(1-\sigma)}\,, \tag{22b}$$

$$\theta_{13} = \frac{Ah_3}{\sigma l_{1L} + l(1-\sigma)}\,, \tag{22c}$$

$$\theta_{21} = \sigma\,. \tag{22d}$$

Equations (21a), (21b) and (22a)–(22d) fully describe the dynamic behaviour of the system and the values of their parameters can be used to decide whether a leak has occurred or not. This is accomplished by estimating the values of the parameters l_{1L} and σ which should be zero in normal (no-leak) operation. Equations (21a)–(21b) are linear-in-the-parameters differential and algebraic equations respectively and therefore the parameter vectors $\boldsymbol{\theta}_i$ can be estimated from input–output data using the proposed techniques. To this end, for every measurement pair $y_1(k)$, $y_2(k)$, equations (21a)–(21b) can be written in discrete time in the form of equation (1b) as follows:

$$y_i(k) = \boldsymbol{u}_i^T(k)\boldsymbol{\theta}_i + e_i(k)\,; \quad k = 0, t_h, \ldots, nt_h, \ldots, \tag{23}$$

where

$$y_1(k) \equiv \left.\frac{\mathrm{d}M_1}{\mathrm{d}t}\right|_{t=k} \qquad \text{(output data)}$$

$$y_2(k) \equiv \frac{M_1(t) - M_6(t)}{M_1(t)} \qquad \text{(output data)}$$

$$\boldsymbol{u}_1(k) \equiv [M_1^2(k) \quad \omega(k)M_1(k) \quad \omega^2(K)] \qquad \text{(input data)}$$

$$\boldsymbol{u}_2(k) \equiv 1 \qquad \text{(input data)}$$

$$\boldsymbol{\theta}_1 = [\theta_{11} \quad \theta_{12} \quad \theta_{13}] \qquad \text{(unknown parameter vector)}$$

$$\boldsymbol{\theta}_2 = [\theta_{21}] \qquad \text{(unknown parameter vector)}$$

and t_h is the sampling interval.

The whole scheme was simulated on a microcomputer with the following data: $l = 130$ m, $D = 0.07$ m, $\rho = 1000$ kg/m^3, $\xi = 2.25$, $\lambda = 0.0035$. A valve was placed at a distance of $l_{12} = 5$ m from the beginning of the pipe, while the pump was operated using equation (20) with $h_1 = -314.97$, $h_2 = 0.316$, $h_3 = 0.02053$. An artificial leak of $\sigma = 2\%$ developed at $t = 750$ s at a point $l_{1l} = 60$ m resulting in $\xi_1 = 2.25$, $\xi_2 = 0$. As input to the system a harmonic signal given by $\omega(t) = 1500 + 500\sin(\pi t/25)$ was applied. The system was simulated for 2000 s. Equation (21a) was solved using a fifth order Runge–Kutta with variable step length. For the numerical differentiation a nine-point central difference formula with step $h = 0.5$ was used. The detection window size was $n_w = 100$. If the above parameter values are substituted in equations (22a)–(22c), the parameter vectors for the no fault situation are

$$\boldsymbol{\theta}_1 = [-1.90683849 \times 10^{-2} \quad 9.3547 \times 10^{-6} \quad 6.087 \times 10^{-7}],$$

$$\boldsymbol{\theta}_2 = [0],$$

while when the leak occurs

$$\boldsymbol{\theta}_1 = [-1.95553081 \times 10^{-2} \quad 10.4837 \times 10^{-6} \quad 6.811 \times 10^{-7}],$$

$$\boldsymbol{\theta}_2 = [2].$$

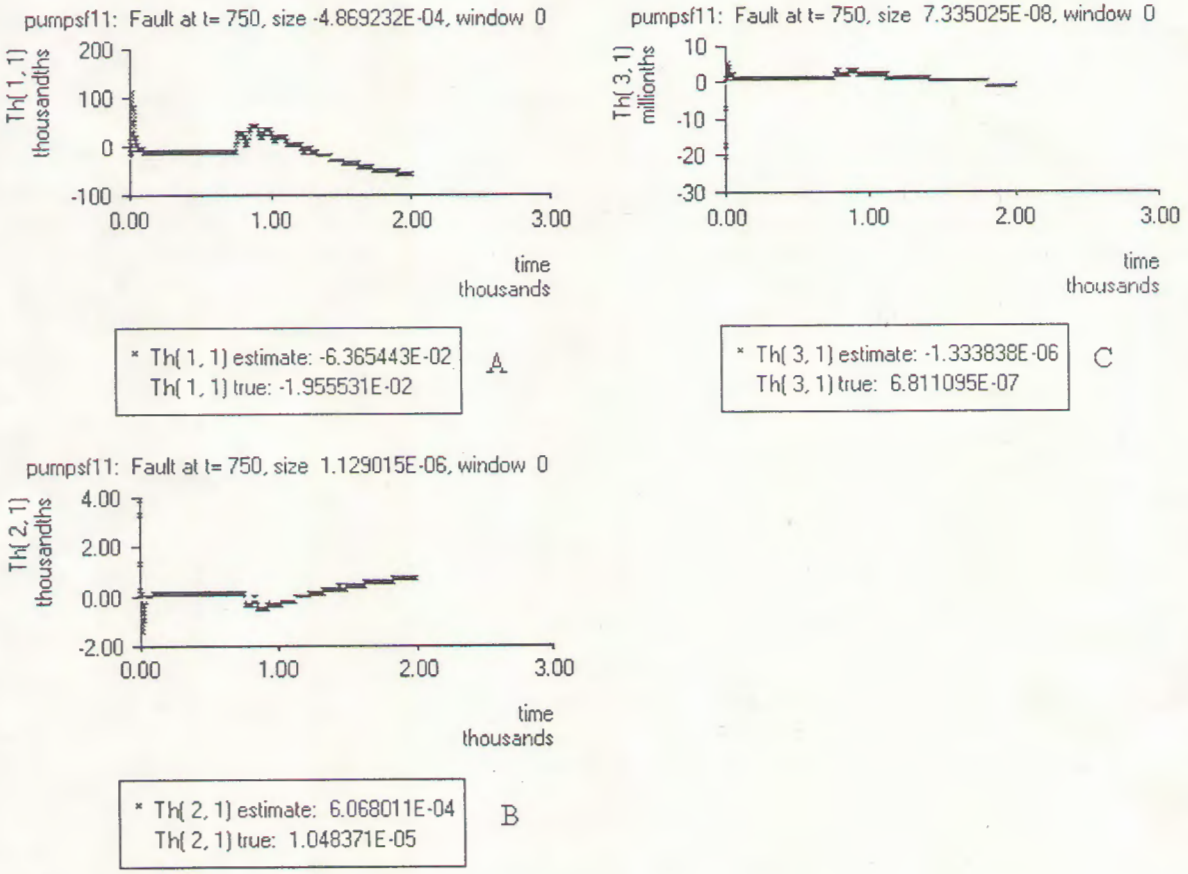

Fig. 2. Parameter estimates with no window.

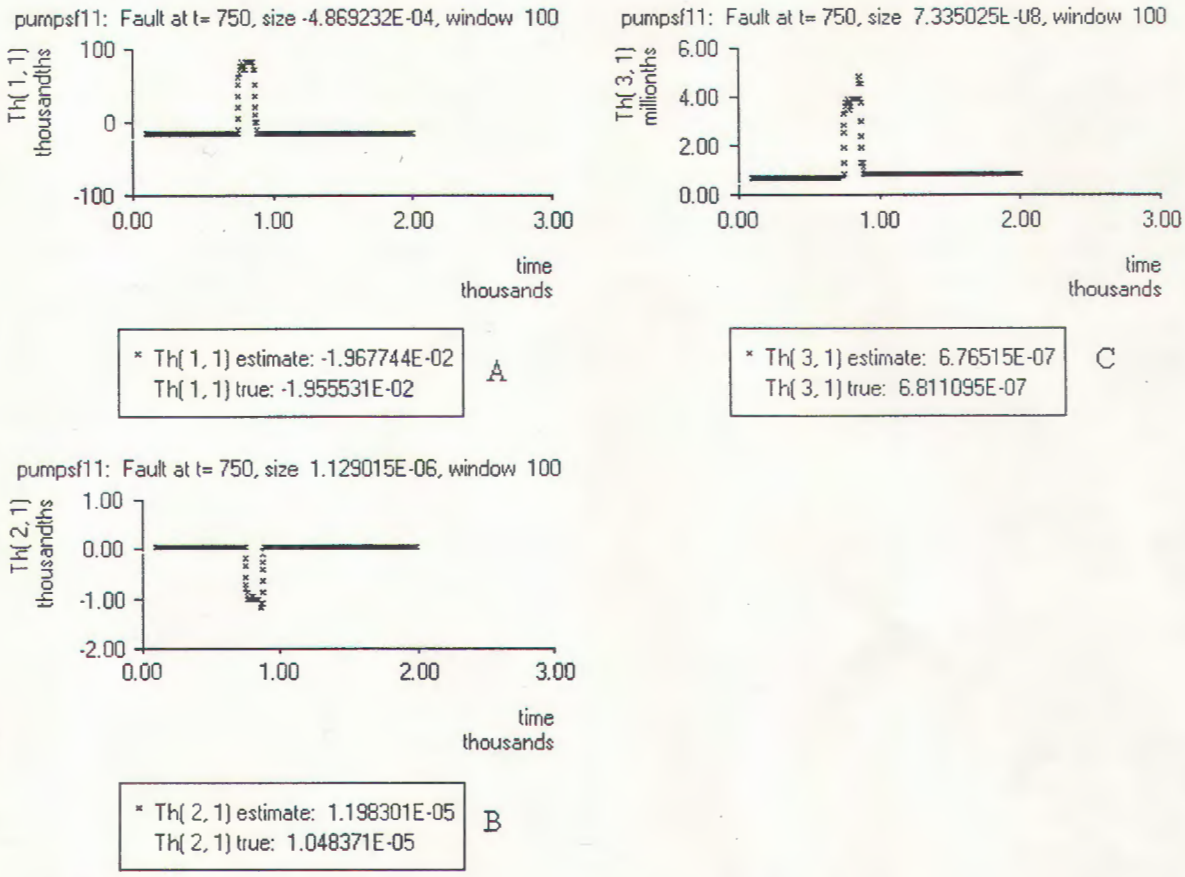

Fig. 3. Parameter estimates with window = 100.

In Figs. 2 and 3 the results of a comparison simulation are shown. The noise variances were $\text{var}\{e_1(k)\} = (0.0005)^2$, $\text{var}\{e_2(k)\} = (0.05)^2$. As seen from Fig. 2, if no window is used, RLS cannot track the parameter change. On the contrary, as Fig. 3 illustrates, the proposed sliding window estimator works very satisfactorily, giving very good leak size and position estimates. In the case of a real system greater accuracy is expected since the sample will exceed the 2000 mark used in the simulations. Furthermore, it must be emphasized that the harmonic test input used is the worst possible case and covers every combination of actual inputs.

4. Conclusions

Iterative sliding window estimators are presented which may be used in cases where infinite memory estimators are not desirable or inappropriate. They are particularly useful in computerised fault detection implementations where increased sensitivity to new data is needed for fast failure alarm rates. The choice of the window length, n_w, is usually a tradeoff between false alarm and missed detection rates. This issue has not been addressed in this paper but is currently under research. The decrease in computational load resulting from the iterative nature of the formulae, make these algorithms attractive for large dimension systems where on-line fault detection schemes were not possible to be applied.

References

[1] M. Basseville, Two examples of application of the GLR method in signal processing, in: M. Basseville and A. Benveniste (Eds.), Detection of Abrupt Changes in Signals and Dynamical Systems (Springer, Berlin, 1986), 50–73.
[2] G.C. Goodwin and K.S. Sin, Adaptive Filtering Prediction and Control (Prentice-Hall, Englewood Cliffs, NJ, 1984).
[3] R.E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME, J. Basic Eng., Ser. D 82 (1960) 342–345.
[4] J.P. Norton, An Introduction to Identification (Academic Press, New York, 1986).
[5] A. Pouliezos, G. Stavrakakis and Ch. Lefas, Fault detection using parameter estimation—a survey, Quality and Reliability Int. 5 (4) (1989) 283–290.
[6] A. Preve, Real-time malfunction diagnosis of pumping systems. Diploma thesis, Dept. of Production and Management Engineering, Technical University of Crete, Greece (1992).
[7] S. Tanaka and P.C. Müller, Diagnosability and failure detection in linear discrete dynamical systems, in: Proceedings 1st European Workshop on Fault Diagnostics, Reliability and Related Knowledge-Based Approaches, Rhodes (1986), 161–175.
[8] A.S. Willsky, Detection of abrupt changes in dynamic systems, in: M. Basseville and A. Benveniste (Eds.), Detection of Abrupt Changes in Signals and Dynamical Systems (Springer, Berlin, 1986), 27–49.