Parallel processing computer implementation of a real time DC motor drive fault detection algorithm

Article *in* Electric Power Applications, IEE Proceedings B [see also IEE Proceedings-Electric Power Applications] · October 1990

DOI: 10.1049/ip-b.1990.0037 · Source: IEEE Xplore

3 authors, including:

G. Stavrakakis
Technical University of Crete
196 PUBLICATIONS **2,440** CITATIONS

SEE PROFILE

Anastasios D. Pouliezos
Technical University of Crete
62 PUBLICATIONS **599** CITATIONS

SEE PROFILE

Parallel processing computer implementation of a real time DC motor drive fault detection algorithm

G.S. Stavrakakis
C. Lefas
A. Pouliezos

Indexing terms: Fault detection, Robotics, Algorithms, Motors

Abstract: The present paper describes a fault detection system for robotic DC motor drives. First, the fault detection method is presented, which, by reducing the amount of computation, permits a higher sampling rate than other fault detection methods. Next, the effectiveness of the method is verified using simulation results and, finally, the implementation of the algorithm on a commercially available parallel processing machine is given.

1 Introduction

The continually increasing demands for reliability and safety of technical plant have steered the improvement of industrial process supervision and monitoring methods as part of the overall control scheme.

The early indication of failures can help avoid major breakdowns and catastrophies that could otherwise result in substantial material damage and human injuries. Similarly, failure detection and isolation has become a critical issue in the operation of high-performance ships, airplanes, space vehicles and structures where safety, mission accomplishment and material value are at stake. By assisting the human operator in assessing the nature and extent of the fault, automatic diagnostic systems may contribute significantly to the fast and proper reaction to failure situations, with such reactions ranging from immediate emergency actions to long-term modification of the maintenance schedule [2].

An essential prerequisite for the further development of automatic supervision is real time process fault detection. Because of restrictions in available computing power at reasonable cost, classical fault detection methods usually compare output signals with limit values only [3, 5].

A method with which to deal with this problem is the use of real time fault detection algorithms, which detect faults by estimating several parameters that are not directly measurable. An unexpected change in these parameters indicates the gradual buildup of the fault [1, 3, 8].

A major concern of such methods is the number of computations that have to be performed. Since computations have to be performed in real time, simplifications of initial theoretical methods are keenly sought after, even if this results in slightly degraded performance [1, 3, 5]. This effort is complemented, however, by the substantial increase in computing power, at the same cost.

A method suitable for real time implementation is given by Stavrakakis and Pouliezos [4]. The major strength of this method is the recursive form of all the necessary computations, resulting in a smaller memory requirement and increased speed. These ideas are further developed in this paper and the whole fault monitoring scheme is implemented on a parallel processing machine. The performance of the system is assessed by simulation.

2 Theoretical background

The considered unit under test (UUT) is regarded as a multi-input multioutput (MIMO) continuous time system. The system model parameters θ represent more or less intricate relationships between several physical process coefficients, e.g. length, mass, inertia, drag coefficient, viscosity, resistance and/or capacitance.

Since the physical process coefficients p, which indicate system faults, are not directly measurable, faults are detected via the change in the values of the process model parameters θ. An estimation of the model parameters can be made from measurement of signals $y(t)$ (system outputs) and $u(t)$ (control inputs), the process theoretical model and modern estimation theory. A necessary requirement of this procedure is the existence of the inverse relationship

$$p = F^{-1}(\theta) \tag{1}$$

In the case of engines, theoretical models can be found in almost every case [3].

The first stage of the fault detection procedure consists of the estimation of parameters θ from the measurements and is followed by the calculation of the physical process parameters p using eqn. 1. The last stage consists of the fault detection decision mechanism. Fig. 1 shows a block diagram for fault detection using parameter estimation.

The recursive nature of all computations, makes the algorithm especially suitable for online applications using multiprocessor computer hardware. Such systems are currently commercially available at a modest cost, which is rapidly decreasing. In this sense it is reasonable to expect that real time fault detection will become increasingly popular and will be implemented in dedicated multiprocessor systems.

Paper 7293B (P1), received 30th May 1989
The authors are with the Technical University of Crete, El. Venizelou 34, 73100 Chania, Greece

In the next Section, the development of a real time fault detection algorithm for robotic DC drives is described. It is implemented on a four-processor transputer system. The algorithm is broken down into four tasks, which are carried out by each processor. At the end of each task the processors are synchronised.

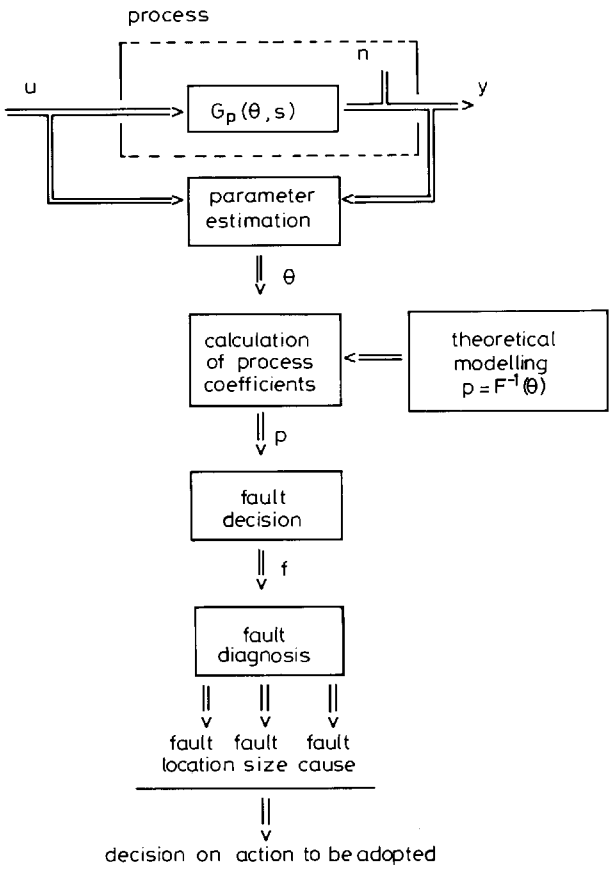


Fig. 1 *Generalised structure of fault detection method based on parameter estimation and theoretical modelling*

3 Fast fault detection for a DC motor drive

In this Section the application of the method to robotic systems driven by a speed-controlled direct current motor is considered.

Using the global dynamic model of a three degrees of freedom robotic manipulator derived by Tzafestas and Stavrakakis [7], the state-space representation for the actuator of the ith link of the robot can be written as

$$\boldsymbol{y}^{(1)}(t) + \boldsymbol{A}_1\boldsymbol{y}^{(1)}(t) + \boldsymbol{A}_0\boldsymbol{y}(t) = \boldsymbol{B}_0\boldsymbol{u}(t) \tag{2}$$

where

$$\boldsymbol{A}_1 = \boldsymbol{0} \in \boldsymbol{R}^{4\times 4}$$

$$\boldsymbol{A}_0 = \begin{bmatrix} \dfrac{R_i}{L_i} & \dfrac{K_{mi}N_i}{L_i} \\ -\dfrac{K_{mi}}{J_{mi}N_i} & \dfrac{\rho_i}{J_{mi}} \end{bmatrix}$$

$$\boldsymbol{B}_0 = \begin{bmatrix} \dfrac{1}{L_i} & 0 \\ 0 & -\dfrac{1}{J_{mi}N_i^2} \end{bmatrix}$$

$$\boldsymbol{u}^T(t) = [V_i(t) \quad T_{Li}(t)]$$

$$\boldsymbol{y}^T(t) = [i_i(t) \quad \omega_i(t)]$$

where

V_i = applied armature voltage
T_{Li} = disturbance torque referred to link side of drive shaft
i_i = armature current
ω_i = shaft angular velocity referred to link side of drive shaft
N_i = gear ratio
J_{mi} = moment of inertia of drive rotor
K_{mi} = electromechanical constant of motor (the back-EMF constant is equal to the torque constant)
R_i = armature resistance
L_i = armature inductance
ρ_i = viscous friction coefficient

The subscript i denotes the ith joint of the robotic manipulator. Define

$$\theta_1 = \frac{R_i}{L_i} \quad \theta_2 = \frac{K_{mi}N_i}{L_i} \quad \theta_3 = \frac{1}{L_i}$$

$$\theta_4 = -\frac{K_{mi}}{J_{mi}N_i} \quad \theta_5 = \frac{\rho_i}{J_{mi}} \quad \theta_6 = -\frac{1}{J_{mi}N_i^2} \tag{3}$$

i.e.

$$\boldsymbol{\theta}^T = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6] \in \boldsymbol{R}^6$$

The following variables are measured for each motor: armature current, angular velocity, armature voltage and shaft torque.

The former two are the system outputs, whereas the latter are the system inputs. Input and output signal measurements are available at discrete times $t = kT_0$, $k = 0, 1, \ldots, N, \ldots$ (where T_0 is the sampling time) defined as $i_i(k)$, $\omega_i(k)$, $V_i(k)$, $T_{Li}(k)$. We therefore have the following observation equations:

$$\begin{aligned} y_1^{(1)}(k) &= \boldsymbol{\psi}_1^T(k)\boldsymbol{\theta}_a + e_1(t) \\ y_2^{(1)}(k) &= \boldsymbol{\psi}_2^T(k)\boldsymbol{\theta}_b + e_2(t) \end{aligned} \tag{4a}$$

where

$$\begin{aligned} \boldsymbol{\psi}_1^T(k) &= [-\boldsymbol{y}^T(k)\ u_1(k)] \in \boldsymbol{R}^3 \\ \boldsymbol{\psi}_2^T(k) &= [-\boldsymbol{y}^T(k)\ u_2(k)] \in \boldsymbol{R}^3 \end{aligned} \tag{4b}$$

The algorithm is implemented on a four-processor system operating as a two-stage pipeline. At the input, a measurement unit M feeds the first two processors. At the output, a fault decision unit operates as a separate unit, having, however, a light computational load and it is therefore a low cost processor system. The fault monitoring scheme is broken down into four tasks carried out by the four processors.

Specifically, the following steps are carried out at every sampling instant k:

Measurements: Measure $i_i(k)$, $\omega_i(k)$, $V_i(k)$ and $T_{Li}(k)$ and measure or compute the derivatives $i_i^{(1)}(k)$ and $\omega_i^{(1)}(k)$ by the third order backward formula:

$$\begin{aligned} i_i^{(1)}(k) &= \frac{1}{2h}\{3i_i(k) - 4i_i(k-1) + i_i(k-2)\} \\ \omega_i^{(1)}(k) &= \frac{1}{2h}\{3\omega_i(k) - 4\omega_i(k-1) + \omega_i(k-2)\} \end{aligned} \tag{5}$$

where $h = T_0$ is the sampling interval.

Task 1: Perform one iteration of the recursive least squares (RLS) parameter estimation algorithm for parameters

$$\boldsymbol{\theta}_a^T = [\theta_1 \theta_2 \theta_3]$$

Task 2: Perform one iteration of the parameter estimation algorithm for parameters

$$\theta_b^T = [\theta_4\ \theta_5\ \theta_6]$$

The algorithm and its computational load are described in Appendix 6.1.

Task 3(a): Calculate the physical parameters $p_i(k)$, $i = 1, 2, 3$ from the previously computed estimates for θ_a and θ_b, using

$$p_1(k) = R_i = \frac{\theta_1(k)}{\theta_3(k)}$$

$$p_2(k) = L_i = \frac{1}{\theta_3(k)} \tag{6}$$

$$p_3(k) = N_i K_{mi} = \frac{\theta_2(k)}{\theta_3(k)}$$

The case of a fault occurrence in the gearbox is considered as an event with probability 0.

Task 3(b): Redefine the data window by accepting the new estimates $p_i(k)$, $i = 1, 2, 3$, dropping the oldest estimates $p_i(k - N_w - 1)$ and recalculating the real time parameter mean and variance estimates (i.e. the parameter statistics are estimated over the $N_w + 1$ most recent parameter estimates). The relevant recursive equations (11–13) are given in Appendix 7.2.

Task 3(c): Compute the likelihood ratio for the fault/no fault hypothesis according to eqn. 14 or 15.

Task 3(d): Decide whether a fault condition exists. The decision is taken by comparing the likelihood ratio obtained in task 3(c), against a predefined threshold. To avoid false alarms, the fault condition is signalled if the threshold is exceeded in M consecutive instants. The optimal threshold value and M are best chosen by trial and error using simulation.

Task 4: Perform tasks 3(*a*) to (*d*) for parameters $p_4(k)$ and $p_5(k)$, using

$$p_4(k) = N_i^2 J_{mi} = -\frac{\theta_2(k)}{\theta_3(k)\theta_4(k)}$$

and

$$p_5(k) = N_i^2 \rho_i = -\frac{\theta_2(k)\theta_5(k)}{\theta_3(k)\theta_4(k)} \tag{7}$$

The above procedure assumes that the algorithm is run initially on a fault free DC motor. The nonerror statistics are obtained from this run and are used subsequently in tasks 3(*b*) and (*c*) and 4(*b*) and (*c*).

The initialisation of the algorithm in real time (until the first N_w measurements are processed) uses the following equations once:

$$\hat{\mu}_i(0) = \frac{1}{N_w}\sum_{j=1}^{N_w} \hat{p}_i(j)$$

$$\hat{\sigma}_i^2(0) = \frac{1}{N_w}\sum_{j=1}^{N_w} [\hat{p}_i(j) - \hat{\mu}_i(0)]^2 \tag{8}$$

$$\tilde{\sigma}_i^2(0) = \frac{1}{N_w}\sum_{j=1}^{N_w} [\hat{p}_i(j) - \bar{\mu}_i]^2$$

Alternatively, the recursive equations 9 and 10 of Appendix 7.2 can be used.

4 Simulation results: implementation

The effectiveness of the method was verified using simulated data. For this purpose the DC motor robotic actuator parameters are

$R = 1.04\ \Omega$ $\quad j_m = 0.00005\ \text{kgm}^2$

$L = 0.00089\ \text{H}$ $\quad \rho = 0.005\ \text{kgm}^2/\text{s}$

$K_m = 0.0224\ \text{Vs/rad}$ $\quad N = 64$

A 2 kHz sampling frequency is considered. The nonerror estimate statistics are calculated using $N_s = 300$ samples, whereas the detection window is $N_w = 50$. The first parameter estimate to be used by the detection procedure was taken at tine $k = 70$, giving a large initial sample. The likelihood ratio fault detection threshold value is 11.2, while M is set at 10. These values are determined by trial and error.

A normal operating DC drive was simulated from sample time $k = 1$ to $k = 130$. A fault occurs at sample number $k = 131$, indicated by a 4.8% change in the armature resistance R_1 (i.e. $R_{1f} = 1.09\ \Omega$). The results in Figs. 2*a* to *e* are obtained using an RLS estimator with a foregetting factor $\lambda_a = 0.95$ for the estimation of θ_a and $\lambda_b = 0.99$ for the estimation of θ_b. As can be seen, all estimates converge quickly to their respective values after the fault occurrence. The exact estimated values are shown in Table 1.

Table 1: True and estimated values for test run

	R_1	L_1	$K_{m1}N_1$	$J_{m1}N_1^2$	$\rho_1 N_1^2$
True value	1.09	0.00089	1.4336	0.2048	20.48
Estimated values at sample time $k = 300$	1.1	0.000896	1.4476	0.2038	20.82

Fig. 3*b* shows the fault occurrence likelihood ratio for parameter R_1. A sharp increase in the likelihood is observed after $k = 131$ (the time when the fault occurs) enabling early and accurate detection of the fault. The index shown in Fig. 3*a* is introduced to show on which parameter the fault has occurred. Its value depends directly on the values of the fault occurrence likelihood ratios. The fault occurrence likelihood ratios for the other parameters are not disturbed, resulting in the desired fault distinguishability.

A major factor in the success of the above algorithm is the assumption of the 2 kHz sampling rate. This means that the algorithm must be implemented on a computer capable of performing all the above calculations in 0.5 ms, an almost impossible task for single processor architectures. The above procedure, however, is suitable for implementation on parallel processing machines, e.g. the INMOS transputer system. This algorithm is implemented on a system employing four transputers as shown in Fig. 4. The numbers shown in Fig. 4 correspond to the tasks performed by each machine according to the task partition described in Section 3. The implementation forms a two-stage pipeline. The first stage consists of machines 1 and 2 and the second stage of machines 3 and 4. Fault decision is performed by a separate machine which is underutilised by the algorithm, leaving power for suitable presentation of the results. The computational complexity (i.e. multiplications and divisions per recursion, MADPR) is given for all algorithms in Appendixes 6.1 and 6.2. It is seen that all machines perform

approximately the same number of operations for $m = 2$ and 3, so that the system operates utilising almost its full computing power.

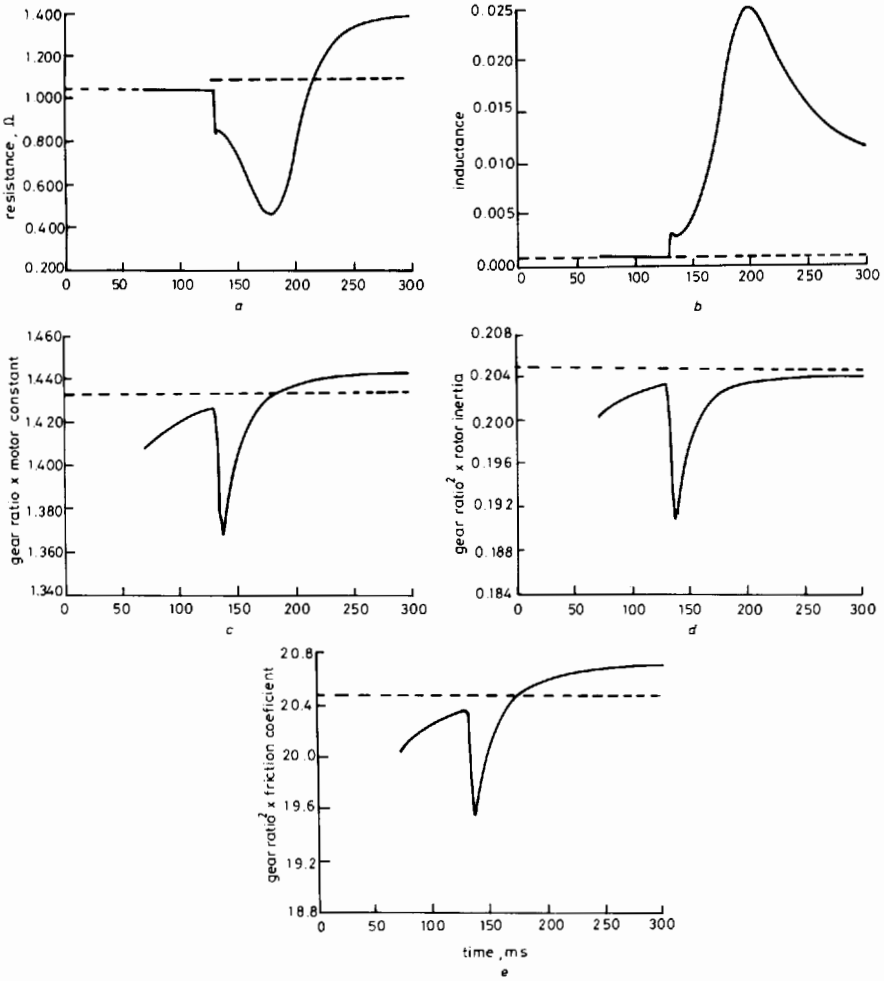
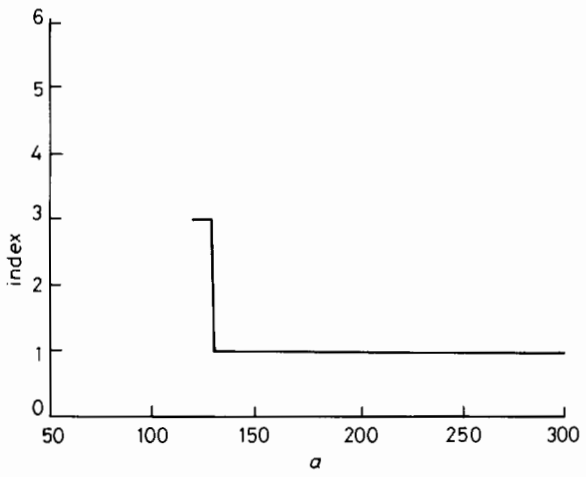


Fig. 2 *Time evolution of motor characteristics with fault at time $k = 0$*



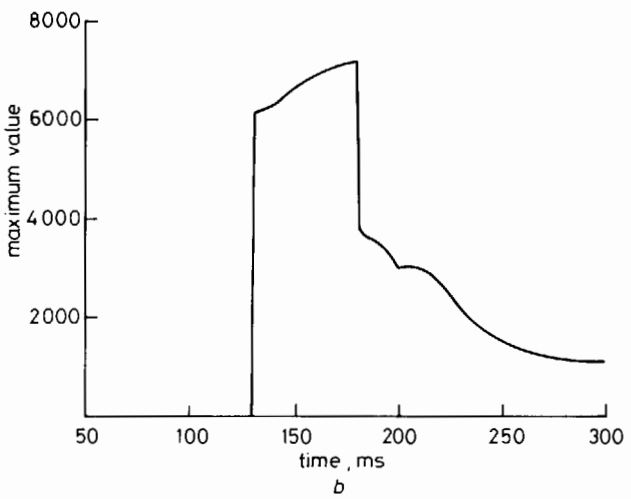


Fig. 3 *Algorithm parameter response to fault*

a Index of faulty system parameter Fault occurs at time $k = 130$
b Likelihood ratio time evolution

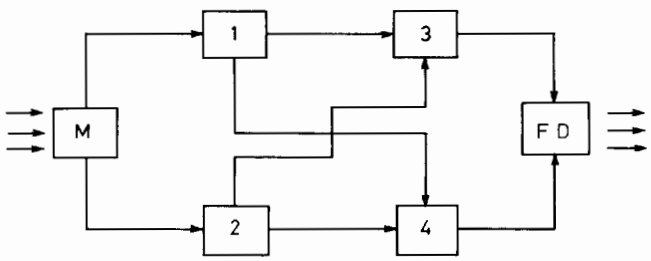


Fig. 4 *Four-processor real time computer implementation of DC drive fault detection algorithm*

5 Conclusion

The described fault detection algorithm for DC motors is suitably implemented on commercial parallel processing machines. The partitioning of the algorithm described makes it suitable for implementation on the INMOS transputer architecture and requires four machines. Such a system offers a powerful tool for fault detection in robotic DC motor drives in real time at a modest cost.

6 References

1 GEIGER, G.: 'Fault identification using a discrete square root method', *Int. J. Model. Simul.*, 1988, **6**, (1), pp. 26–31
2 GERTLER, J.: 'Survey of model-based failure detection and isolation in complex plants', *IEEE Control Syst. Mag.*, **5**, (6), pp. 3–11

3 ISERMAN, R.: 'Process fault detection based on modelling and estimation methods — A survey', *Automatica*, **20**, (4), pp. 387–404
4 POULIEZOS, A., and STAVRAKAKIS, G.: 'Fast fault detection and location for reliable operation of controlled robotic systems', *Int. J. Syst. Sci.*, 1989
5 RAULT, A., and BASKIOTIS, C.: 'Fault detection, diagnosis and predictive maintenance'. Proceedings of 25th Conference on Decision and Control, Athens, Greece, December 1986
6 TZAFESTAS, S.G. *et al.* (Eds.): 'Reliability and fault detection techniques of large scale systems' (Springer, 1985), pp. 35–65
7 TZAFESTAS, S.G., and STAVRAKAKIS, G.S.: 'Model reference adaptive control of industrial robots with actuator dynamics'. IFAC/IFIP/IMACS International Symposium on Theory of Robots, Vienna, December 3–5, 1986
8 WILLSKY, S.: 'A survey of design methods for failure detection in dynamical systems', *Automatica*, **12**, pp. 601–611

7 Appendixes

7.1 Conventional mth order RLS algorithm with forgetting factor

Information available at time k:

$\boldsymbol{P}(k)$: estimation error covariance matrix, $m \times m$
$\boldsymbol{\psi}(k)$: measurement vector, $m \times 1$, according to eqn. 4*b*
$\hat{\boldsymbol{\theta}}(k)$: unknown parameter estimate vector, $m \times 1$, according to eqn. 3
$\boldsymbol{W}^*(k)$: Kalman gain vector, $m \times 1$

New information at time k + 1:

$$\boldsymbol{\psi}(k+1) \in \boldsymbol{R}^m \quad \text{and} \quad y^{(1)}(k+1) \in \mathbb{R}^1$$

according to eqns. 4*a* and *b*.

The necessary MADPR for the algorithm are shown in Table 2.

Remarks: The initialization of the RLS algorithm can be done either by evaluating $\boldsymbol{P}(0)$ for an initial block of data or by simply setting $\boldsymbol{P}(0) = \sigma \boldsymbol{I}$, $\sigma \gg 0$.

The initial vector $\hat{\boldsymbol{\theta}}(0)$ can be zero or equal to the values of the estimated parameters used in the process model.

Table 2: MADPR for RLS algorithm

Time updating of gain vector	MADPR
$\boldsymbol{W}(k+1) = (1/\lambda)\boldsymbol{P}(k)\boldsymbol{\psi}(k+1)$	$m^2 + m$
$a(k+1) = 1 + \boldsymbol{\psi}^T(k+1)\boldsymbol{W}(k+1)$	m
$\boldsymbol{W}^*(k+1) = \boldsymbol{W}(k+1)/a(k+1)$	m
$\boldsymbol{P}(k+1) = (1/\lambda)\boldsymbol{P}(k) - \boldsymbol{W}^*(k+1)\boldsymbol{W}^T(k+1)$	$0.5(m^2+m)$
Time updating of the forward predictor	
$e(k+1) = y^{(1)}(k+1) - \boldsymbol{\psi}^T(k)\hat{\boldsymbol{\theta}}(k)$	m
$\hat{\boldsymbol{\theta}}(k+1) = \hat{\boldsymbol{\theta}}(k) + \boldsymbol{W}^*(k+1)e(k+1)$	m
Total number of operations	$(3/2)m^2 + (11/2)m$

For the present case of RLS estimation of $\hat{\theta}_a$ and $\hat{\theta}_b$ the order is $m = 3$. Thus, the total number of MADPR = 30 for each estimator.

7.2 Recursive statistics formulas

7.2.1 Nonerror case

For the nonerror case, H_0 (offline statistics):

$$(a)\ \bar{\mu}_i(k) = (1/k)[(k-1)\bar{\mu}_i(k-1) + \hat{p}_i(k)]$$
$$i = 1, \ldots, m;\ k = 1, \ldots, N_s \quad (9)$$

The recursion is initialised with $\bar{\mu}_i(1) = \hat{p}_i(1)$.

$$(b)\ \bar{\sigma}_i^2(k) = \frac{k-2}{k-1}\bar{\sigma}_i^2(k-1) + \frac{1}{k}[\hat{p}_i(k) - \bar{\mu}_i(k-1)]^2$$
$$i = 1, \ldots, m;\ k = 1, \ldots, N_s \quad (10)$$

The recursion is self-initialised by

$$\bar{\sigma}_i^2(2) = \tfrac{1}{2}[\hat{p}_i(2) - \hat{p}_i(1)]^2$$

The sample size N_s is chosen sufficiently large that an accurate estimate for the above statistics is obtained.

7.2.2 Error case

For the error case H_i, $i = 1, \ldots, m$, three quantities are needed (real time statistics):

(*a*) Window mean:

$$\hat{\mu}_i(k) = \frac{1}{N_w}\sum_{j=k-N_w+1}^{k}\hat{p}_i(j)$$
$$= \hat{\mu}_i(k-1) - \frac{1}{N_w}\gamma_i(k) \quad i = 1, \ldots, m \quad (11)$$

where

$$\gamma_i(k) = \hat{p}_i(k - N_w) - \hat{p}_i(k)$$
$$i = 1, \ldots, m;\ \text{MADPR} = m$$

(*b*) Window variance:

$$\hat{\sigma}_i^2(k) = \frac{1}{N_w}\sum_{j=k-N_w+1}^{k}[\hat{p}_i(j) - \hat{\mu}_i(k)]^2 = \hat{\sigma}_i^2(k-1)$$
$$+ \frac{1}{N_w}\left[2\gamma_i(k)\hat{\mu}_i(k-1) - \frac{1}{N_w}\gamma_i^2(k) - \hat{p}_i^2(k-N_w) + \hat{p}_i^2(k)\right]$$
$$i = 1, \ldots, m;\ \text{MADPR} = 7m \quad (12)$$

(*c*) Window variance based on the nonerror mean $\bar{\mu}_i$:

$$\tilde{\sigma}_i^2(k) = \frac{1}{N_w}\sum_{j=k-N_w+1}^{k}[\hat{p}_i(j) - \bar{\mu}_i]^2 = \tilde{\sigma}_i^2(k-1)$$
$$+ \frac{1}{N_w}[-2\gamma_i(k)\bar{\mu}_i + \hat{p}_i^2(k-N_w) - \hat{p}_i^2(k)]$$
$$i = 1, \ldots, m;\ \text{MADPR} = 2\text{m} \quad (13)$$

The total number of operations is $10m$ per iteration.

These three iterative schemes need a starting window of N_w sample values $\hat{p}_i$ for initialisation. The window size N_w is chosen so that reasonable rates for missed alarms and false detections are achieved. The advantage of using a moving window of sample parameter values $\hat{p}_i(k)$, $i = 1, \ldots, m$, is in the improved speed of detection.

A fault in the ith parameter is declared at time k, if the quantity (MADPR = $3m$)

$$\Lambda_i(k) = \frac{N_w}{2}\left[\frac{\tilde{\sigma}_i(k)}{\bar{\sigma}_i(k)} - 2\ln\left(\frac{\hat{\sigma}_i(k)}{\bar{\sigma}_i(k)}\right) - 1\right] \quad (14)$$

exceeds a predetermined threshold in M consecutive time instants. The threshold value and M may be chosen by simulation.

Alternatively, the comparable quantity may be (MADPR = $2m$)

$$\Lambda_i(k) = \frac{\tilde{\sigma}_i(k)}{\bar{\sigma}_i(k)} - 2\ln\left(\frac{\hat{\sigma}_i(k)}{\bar{\sigma}_i(k)}\right) \quad (15)$$

with the threshold value modified accordingly.