Várkonyi-Kóczy, A.R., T. Kovácsházy, O. Takács, and Cs. Benedecsik, "Anytime Evaluation of Regression-Type Algorithms." International Journal of Advanced Computational Intelligence, Vol. 5, No. 1, pp. 2-7, Feb. 2001.

Anytime Evaluation of Regression-Type Algorithms

Annamária R. Várkonyi-Kóczy,

Tamás Kovácsházy, Orsolya Takács, Csaba Benedecsik

Department of Measurement and Information Systems

Budapest University of Technology and Economics, H-1521 Budapest, Műegyetem rkp. 9.,

Hungary

Phone: + 36 1 463 2057, Fax: +36 1 463 4112

e-mail: koczy@mit.bme.hu

Abstract

Regression-type algorithms are widely used for system modeling and characterization. There are applications where such characterizations are to be performed "on-line" to support control mechanisms and other decisions. In embedded autonomous systems robustness considerations ask for techniques, which, in addition to reflecting the actual state of the system and its environment, can continuously provide immediate signal processing results even in case of abrupt changes and/or temporal shortage of computational power and/or loss of some data. With other words in such situations the actual processing should be continued to insure appropriate performance. Consequently there is a need for robust techniques called "anytime" algorithms, which can provide

short response time and be very flexible with respect to the available input information and computational power. The paper presents some considerations concerning such flexibility in the case of regression-type algorithms.

1. Introduction

Computer-based monitoring and diagnostic systems are designed to handle abrupt changes due to failures within the supervised system or in its environment. This capability involves on one hand different, simultaneously operated digital signal processors (DSPs), while on the other the corresponding information processing algorithms. These algorithms should be performed under prescribed response time conditions. It is an obvious requirement to provide enough computational power but the achievable processing speed is highly influenced by the precedence, timing and data access conditions of the processing itself. It seems to be unavoidable even in the case of extremely careful design to get into situations where the shortage of necessary data and/or processing time becomes serious. Such situations may result in a critical breakdown of the computer-based monitoring and/or diagnostic systems.

With the introduction of "anytime" signal processing algorithms [1] we try to handle the above changes and their consequences in larger scale embedded DSP systems [2]. The idea is that if there is a temporal shortage of computational power and/or there is a loss of some data the actual evaluation should be continued to provide appropriate overall performance. The solution should be signal processing of simpler complexity producing outputs of acceptable quality to continue the operation of the complete embedded system. The accuracy of the processing may be temporarily lower but possibly still enough to produce data for qualitative evaluations and supporting

unavoidable decisions. Consequently "anytime" algorithms provide short response time and are very flexible with respect to the available input information and computational power. Such and similar flexibility is investigated in a somewhat different environment in [3]. It is important to observe at this point that such flexibility is possible only if a set of "shortage indicators" is an additional input of the signal processing facilities in use. Obviously the shortage indicators are outputs of such information processing units which monitor the actual rate of sensory data and the rate of the computational load.

In Section II of this paper regression-type algorithms will be considered. Since the operation of the monitoring and/or diagnostic systems should run in parallel with the data acquisition, the investigations are restricted to recursive algorithms (see, e.g., [4] - [6]). These algorithms are typically one-step prediction-correction schemes, where the weighted difference between the measured and the predicted inputs corrects the previous estimates. Recursive techniques are advantageous concerning also the response time since they almost always can provide some kind of an estimate at a price of acceptable computational load. In this Section the key equations of the recursive least squares method are investigated more in detail from the above-mentioned point of view.

In Section III of this paper a "shortage controlled" mechanism is developed to adapt the recursive least

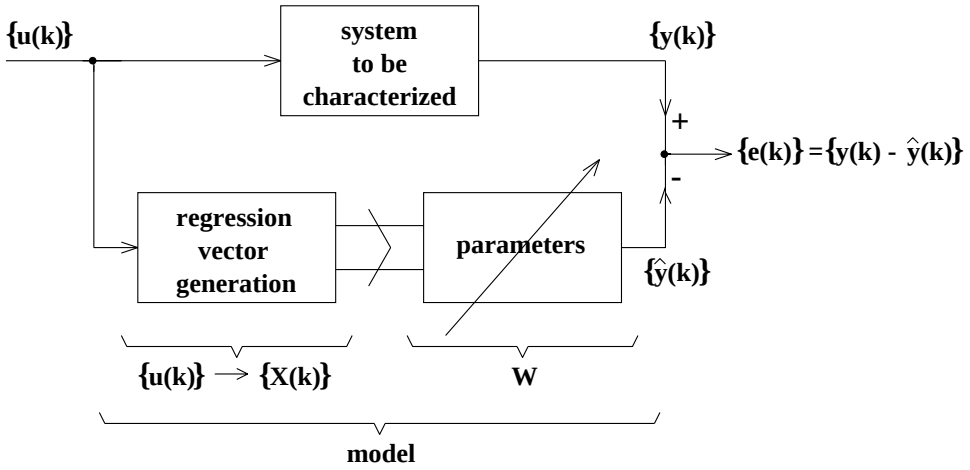

Figure 1. The signal-flow scheme of a typical regression-type algorithm. The goal of the procedure is to set the parameter vector W to minimize the modeling error

squares method to the available amount of computing capacity. The adaptation is a temporal reduction of the number of the model parameters to be updated within one iteration step. Those parameters should be omitted which have less influence on the model output.

In Section IV, as an illustrative example, the classical linear regression problem is considered for anytime evaluation. The analytical formulas help to understand the nature and complexity of the modified recursive least square technique.

2. Recursive evaluation of regression-type algorithms

The signal-flow scheme of a typical regression-type algorithm is given on Fig. 1. The system to be characterized is perturbed by a sequence of input samples $u(k)$, $k=0,1, \ldots$, and produces an output sequence of $y(k)$. The regression problem is to fit a model to the system using the set of inputs and outputs ($\{u(k)\}$ and $\{y(k)\}$) in such a way that the difference at the output $\{e(k)\}$ is minimum in a given sense. The typical approach fixes the model-structure

and offers a parameter vector *W* to set or adapt during the procedure. A widely used model form is where a fixed part generates the so-called regression vectors {*X(k)*}, and this is followed by a variable part producing output estimates $\{\hat{y}(k)\}$. Since for practical reasons it is advantageous to apply models which are linear in their parameters, usually this second part is simply a linear combiner with variable weights. The solution of such and similar regression problems is well known in the literature (see, e.g. [4], [5]), here only the key elements of the simplest least-squares procedure are repeated.

Notations:

$Y(n) = [y(0), y(1), \ldots, y(n-1)]^T$ observation vector

$\hat{Y}(n) = [\hat{y}(0), \hat{y}(1), \ldots, \hat{y}(n-1)]^T$ estimation vector

n identifies the number of observations

$\mathbf{X}(n) = [X(0), X(1), \ldots, X(n-1)]^T$ *n*x*m* regression matrix

m number of model parameters

W(*n*) *m*-dimensional parameter vector based on *n* observations

$\mathbf{P}(n) = [\mathbf{X}^T(n)\mathbf{X}(n)]^{-1}$ *m*x*m* matrix

G(*n*) *m*-dimensional correction-gain vector

ε(*n*) squared error after *n* observations

Off-line version: The first step of solving the regression problem off-line is the collection of *n* input and output samples ({*u(k)*}, {(*y(k)*}) and the calculation of *n* regression vectors {*X(k)*} to get *Y(n)* and **X**(*n*). The second step is to calculate *W(n)* which minimizes

$$\varepsilon(n) = \left[Y(n) - \hat{Y}(n)\right]^T \left[Y(n) - \hat{Y}(n)\right] = \sum_{k=0}^{n-1} e^2(k) = \sum_{k=0}^{n-1} \left[y(k) - \hat{y}(k)\right]^2 \tag{1}$$

where

$$\hat{Y}(n) = \mathbf{X}(n)W(n) \tag{2}$$

The optimum set of parameters based on *n* observations is given by

$$W(n) = \left[\mathbf{X}^T(n)\mathbf{X}(n)\right]^{-1}\mathbf{X}^T(n)Y(n) = \mathbf{P}(n)\mathbf{X}^T(n)Y(n) \tag{3}$$

On-line (recursive) version: The equation for the recursive evaluation can be interpreted as a one-step prediction-correction form, which having *W(n)* utilizes the new observation *y(n)* and the new regression vector *X(n)* to calculate *W(n+1):*

$$W(n+1) = W(n) + G(n+1)\left[y(n) - X^T(n)W(n)\right] \tag{4}$$

↑ ↑

new observation new regression vector

where the correction-gain vector is given by

$$G(n+1) = \frac{\mathbf{P}(n)X(n)}{1 + X^T(n)\mathbf{P}(n)X(n)} \tag{5}$$

Fortunately $\mathbf{P}(n+1)$ can be calculated recursively

$$\mathbf{P}(n+1) = \left[\mathbf{I} - G(n+1)X^T(n)\right]\mathbf{P}(n) \tag{6}$$

To use the recursive algorithms, initial values for their start-up are required. Hints can be found in the literature (see, e.g., [4]).

The recursive algorithms are widely used also in non-stationary/time-varying systems. In such case regression-type problem turns to a parameter adaptation problem, where the characterization of the system becomes time-dependent and older observations loose their importance. The introduction of special weights ("forgetting factors") into (1), or as a special case the application of the sliding window slightly influences the above algorithms, but the

key elements of the solution do not change.

3. Temporal computational complexity reduction

In the subsequent discussion we will suppose that $n>>m$, i.e., $W(n)$ is based on a large amount of observations. We will also suppose that a temporal shortage of computational power requires temporal complexity reduction. The computational power needed to evaluate equations (4)-(6) can be estimated if the sampling rate and the required number of parameters m is known. The computational load can be lowered if either the sampling rate or the number of parameters to be updated within one iteration step is lower. There are several possible alternatives:

1. The input/output sampling rate can be reduced if the system characterization with lower "bandwidth" is tolerable.
2. The number of parameters can be reduced by neglecting regression vector components having small contribution to $\hat{y}(n)$. In this case all the vectors and matrices in equations (4)-(6) will have reduced size. To avoid non-tolerable transient oscillations of the parameter values, the withdrawal of regression vector components might be performed step-by-step. Similarly, the reduction of the regression component values to zero might be performed in multiple steps. On the way back to the original number of parameters similar considerations may help.
3. Defining subsets of parameter vector components and to solve the optimization problem for the subsets separately and sequentially.
4. Defining a subset of such parameter vector components, which show relatively small variations during adaptation, and fix them for a given number of iterations.

For the last two alternatives equations (2)-(6) can be rewritten, but we need some new

notations. The ideas will be explained for two subsets.

New notations:

$\mathbf{X}_1(n) = \left[X_1(0), X_1(1), \ldots, X_1(n-1)\right]^T$ nxm_1 regression matrix

$\mathbf{X}_2(n) = \left[X_2(0), X_2(1), \ldots, X_2(n-1)\right]^T$ nxm_2 regression matrix

m_1, m_2 ($m_1+m_2=m$) number of model parameters

$W_1(n)$ m_1-dimensional parameter vector

$W_2(n)$ m_2-dimensional parameter vector

$\mathbf{P}(n) = \begin{bmatrix} \mathbf{P}_{11}(n) & \mathbf{P}_{12}(n) \\ \mathbf{P}_{21}(n) & \mathbf{P}_{22}(n) \end{bmatrix}$ mxm matrix, where

dim $\mathbf{P}_{11}(n) = m_1$xm_1, dim $\mathbf{P}_{12}(n) = m_1$xm_2,

dim $\mathbf{P}_{21}(n) = m_2$xm_1, dim $\mathbf{P}_{22}(n) = m_2$xm_2

$G_1(n)$ m_1-dimensional correction-gain vector

$G_2(n)$ m_2-dimensional correction-gain vector

$\mathbf{P}_1(n) = m_1$xm_1 dimensional matrix

$\mathbf{P}_2(n) = m_2$xm_2 dimensional matrix

If we follow alternative 3, and separate the regression and the parameter vector elements into two groups, equation (2) can be rewritten as

$$\hat{Y}(n) = \left[\mathbf{X}_1(n)\mathbf{X}_2(n)\right]\begin{bmatrix} W_1(n) \\ W_2(n) \end{bmatrix} = \quad \mathbf{X}_1(n)W_1(n) + \mathbf{X}_2(n)W_2(n) \tag{7}$$

If we could consider the two sets independently, then the ***off-line version*** of the solution would have the form

$$W_1(n) = \underbrace{\left[\mathbf{X}_1^T(n)\mathbf{X}_1(n)\right]^{-1}}_{\mathbf{P}_1(n)}\mathbf{X}_1^T\left[Y(n) - \mathbf{X}_2(n)W_2(n)\right] \tag{8}$$

$$W_2(n) = \underbrace{\left[\mathbf{X}_2^T(n)\mathbf{X}_2(n)\right]^{-1}}_{\mathbf{P}_2(n)}\mathbf{X}_2^T\left[Y(n) - \mathbf{X}_1(n)W_1(n)\right] \tag{9}$$

The ***on-line (recursive) version*** can also be developed using (7) from equations (4)-(6). Unfortunately, however, the two sets can not be handled separately, because they are interrelated via equation (7). Since from previous iterations both parameter vectors are known, we can fix one of them, and continue the iteration only for the other, and then fix this latter, and iterate the first one. All these calculations may be performed within one iteration step at a lower price, since matrix manipulations of (m_1+m_2)x(m_1+m_2) size are replaced by manipulations of m_1xm_1 and m_2xm_2 size. With this approach the optimization problem is divided into two "conditional" optimizations, which are performed separately with the assumption that the "other" parameter set is appropriate. A weak point of this technique is that unlike the original method, it separates also the original regression vector components, and therefore only their subsets will improve the corresponding subsets of parameters.

If we follow alternative 4, the parameter adaptation can be written as

$$\begin{aligned} W_1(n+1) &= W_1(n) + G_1(n+1)\left[y(n) - X^T(n)W(n)\right] \\ \hat{W}_2(n+1) &= W_2(n) \end{aligned} \tag{10}$$

where the parameter vector $W_2(n)$ is fixed, and therefore instead of $G_2(n+1)$ its approximation $\hat{G}_2(n+1) = 0$ is applied. $G_1(n+1)$ is evaluated according to the original method:

$$G_1(n+1) = \frac{\mathbf{P}_{11}(n)X_1(n) + \mathbf{P}_{21}(n)X_2(n)}{1 + X^T(n)\mathbf{P}(n)X(n)}; \hat{G}_2(n+1) = 0 \tag{11}$$

The updated value of $\mathbf{P}(n+1)$ will be replaced by an approximation $\hat{\mathbf{P}}_{(n+1)}$, where the last m_2 rows remain fixed:

$$\hat{\mathbf{P}}(n+1)=\begin{bmatrix}\mathbf{P}_{11}(n+1) & \mathbf{P}_{12}(n+1)\\ \hat{\mathbf{P}}_{21}(n+1) & \hat{\mathbf{P}}_{22}(n+1)\end{bmatrix}=$$
$$\begin{bmatrix}\mathbf{I}-G_1(n+1)X_1^T(n) & -G_1(n+1)X_2^T(n)\\ -\hat{G}_2(n+1)X_1^T(n) & \mathbf{I}-\hat{G}_2(n+1)X_2^T(n)\end{bmatrix}*\begin{bmatrix}\mathbf{P}_{11}(n) & \mathbf{P}_{12}(n)\\ \mathbf{P}_{21}(n) & \mathbf{P}_{22}(n)\end{bmatrix}=\begin{bmatrix}\mathbf{P}_{11}(n+1) & \mathbf{P}_{12}(n+1)\\ \mathbf{P}_{21}(n) & \mathbf{P}_{22}(n)\end{bmatrix} \tag{12}$$

The required computational power can be reduced drastically, if $m_1 << m_2$. An important feature is that the information of the regression vector is fully utilized. Another advantage is that with this approach we continuously have an approximation to $\mathbf{P}(\ )$, therefore the "on-line" change of the fixed and adapted parameter subsets is always possible.

4. Illustrative example

In this Section the temporal complexity reduction alternatives 3 and 4 will be investigated via the least-squares evaluation/approximation of the one-dimensional linear regression problem. The results are in analytical form, therefore we can have more insight into the details of the propositions. For the one-dimensional linear regression equation (7) has the form of

$$\hat{Y}(n)=\begin{bmatrix}1 & u(0)\\ 1 & u(1)\\ \vdots & \vdots\\ 1 & u(n-1)\end{bmatrix}\begin{bmatrix}W_1(n)\\ W_2(n)\end{bmatrix} \tag{13}$$

where the parameter vector consists of two elements. If we define two one-dimensional subsets, then using (8) and (9) we get the ***off-line*** solution in the form of a set of linear equations with two unknowns.

$$W_1(n) = \frac{1}{n}\sum_{k=0}^{n-1} y(k) - W_2(n)\frac{1}{n}\sum_{k=0}^{n-1} u(k)$$
$$W_2(n) = \frac{\sum_{k=0}^{n-1} u(k)y(k)}{\sum_{k=0}^{n-1} u^2(k)} - W_1(n)\frac{\sum_{k=0}^{n-1} u(k)}{\sum_{k=0}^{n-1} u^2(k)} \tag{14}$$

The side-effect of the separation is that this equation remained to be solved. The solution is obviously the same as without separation. The ***on-line (recursive)*** evaluation for this separated case is based on the following equations:

$$W_1(n+1) = W_1(n) + G_1(n+1)\left[y(n) - W_1(n) - W_2(n)u(n)\right] \tag{15}$$

$$G_1(n+1) = \frac{1}{n+1}; \qquad P_1(n+1) = \frac{n}{n+1}P_1(n) \tag{16}$$

$$W_2(n+1) = W_2(n) + G_2(n+1)\left[y(n) - W_1(n) - W_2(n)u(n)\right] \tag{17}$$

$$G_2(n+1) = \frac{u(n)}{\sum_{k=0}^{n} u^2(k)}; \qquad P_2(n+1) = \frac{\sum_{k=0}^{n-1} u^2(k)}{\sum_{k=0}^{n} u^2(k)} P_2(n) \tag{18}$$

Here it is important to note that since we have two "conditional" optimization problems, the regression vector is not fully utilized. E.g., $G_1(\ )$ and $\mathbf{P}(\ )$ are independent of $u(\ )$ in (16). Consequently the parameter set produced by the actual iteration will be sub-optimal.

If we do not apply separation, then the ***off-line*** evaluation requires equation (3) with

$$P(n) = \frac{1}{n\sum_{k=0}^{n-1} u(k) - \left[\sum_{k=0}^{n-1} u(k)\right]^2} \begin{bmatrix} \sum_{k=0}^{n-1} u^2(k) & -\sum_{k=0}^{n-1} u(k) \\ -\sum_{k=0}^{n-1} u(k) & n \end{bmatrix} \tag{19}$$

and results in

$$W(n) = \left[\frac{\overline{y}.\overline{u^2} - \overline{u}.\overline{uy}}{\overline{u^2} - [\overline{u}]^2}; \qquad \frac{\overline{uy} - \overline{u}.\overline{y}}{\overline{u^2} - [\overline{u}]^2}\right]^T \tag{20}$$

where "$\bar{\ }$" stands for the linear averaging operator (average of n samples with indices running from 0 to n-1).

Obviously the same results can be obtained by the ***on-line (recursive) version***. The parameter update can be performed by equation (4) using

$$G(n+1) = \frac{1}{(n+1)\sum_{k=0}^{n} u^2(k) - \left[\sum_{k=0}^{n} u(k)\right]^2} \begin{bmatrix} \sum_{k=0}^{n-1} u^2(k) - u(n)\sum_{k=0}^{n-1} u(k) \\ nu(n) - \sum_{k=0}^{n-1} u(k) \end{bmatrix} \tag{21}$$

as gain vector. If we try to apply alternative 4, and we fix $W_2(n)$, then this corresponds to the assumption that $u(n)$ equals its average (see equation (21)). Since this condition is not necessarily a valid assumption, the analytical forms of (19)-(21) will not be valid for further n.

The analytical expressions above, especially (20), offer a further "level" of recursive evaluations, since averaging can be performed recursively, as well. This means, that we can decide to use (20) instead of (4)-(6) simply by "combining" the recursively evaluated averages of $u(\ )$, $y(\ )$, $u^2(\ )$ and $u(\ )y(\ )$.

5. Conclusions

In this paper temporal computational complexity reduction methods are proposed for recursively evaluated regression-type signal processing algorithms. Such and similar algorithms are intensively used in computer-based monitoring and control systems for on-line characterization of the application environment. The purpose of temporal computational complexity reduction is to increase the flexibility of embedded information systems. Increased flexibility enables adaptation and evolution within the application, which may

require temporarily additional computing power at the price of limiting other on-going activities. However, even if the available computing power is limited, some parts of the actual data evaluation and signal processing should be continued possibly using algorithms of lower performance. In this paper such reconfiguration problems are investigated for the case of recursive least squares algorithms.

Acknowledgment

This work was supported by the Hungarian Fund for Scientific Research (OTKA T 026254) and the Office of Bilateral Intergovernmental Scientific Co-operation Programs (GR-32/96).

References

1) Várkonyi-Kóczy, A.R., T. Kovácsházy, "Anytime Algorithms in Embedded Signal Processing Systems," Proc. of the IX. European Signal Processing Conference, EUSIPCO-98, Rhodes, Greece, 169-172, (1998).
2) Baron, C., J.-C. Geffroy, G. Motet, ed., "Embedded System Applications." Kluwer Academic Publishers, (1997).
3) Sztipanovits, J., D.M. Wilkes, G. Karsai, Cs. Biegl, L.E. Lynd, "The Multigraph and structural adaptivity," IEEE Transactions on Signal Processing, 41-8, 2695-2716, (1993).
4) Ljung, L., T. Söderström, "Theory and Practice of Recursive Identification." The MIT Press, Cambridge, Massachusetts, (1983).
5) Ljung, L., "System Identification: Theory for the User." Prentice-Hall, Inc., Englewood Cliffs, NJ., (1987).
6) Diniz, P.S.R., "Adaptive Filtering. Algorithms and Practical Implementation." Academic Publishers, (1997).