Akademie věd české republiky
ÚSTAV TEORIE INFORMACE A AUTOMATIZACE

Academy of Sciences of the Czech Republic
INSTITUTE OF INFORMATION THEORY AND AUTOMATION

RESEARCH REPORT

J. Schier

The Systolic Regularized Linear Quadratic Controller

No. 1815 October 1994

ÚTIA AV ČR P. O. Box 18, 182 08 Prague, Czech Republic
Telex: 122018 atom c, Fax: (+42)(2)6641 4903
e-mail: utia@utia.cas.cz

This report constitutes an unrefereed manuscript which is intended to be submitted for publication. Any opinions and conclusions expressed in this report are those of the author(s) and do not necessarily represent the views of the Institute.

The Systolic Regularized Linear Quadratic Controller

Jan Schier
Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
P.O. Box 18, 182 08 Prague 8, Czech Republic
e-mail: schier@utia.cas.cz

Abstract

Adaptive LQ controllers (Linear model, Quadratic criterion) have been successfully applied in a number of applications. However, due to the computational complexity of $\mathcal{O}(n^2)$, the standard sequential implementation is too slow to control some processes. This is even more the problem if using new, robustified identification methods, increasing the computational complexity yet in order.

A suitable solution is to use a parallel implementation of the controller on a systolic array. Giving a theoretical speedup of an order magnitude, this implementation makes the LQ selftuners attractive for fast applications.

The demands of the robustness and on the speed of the controller are, however, quite contradictory and the robustified systolic algorithms were not available until recently.

This paper aims at contributing to solution of this problem. After referring about various existing implementations of the systolic controller, the principle of a new, regularized, version of an adaptive controller is described.

1. Introduction

Linear quadratic multistep controllers [1, 2] represent one of the most advanced groups among the adaptive controllers. The maturity of their theory has been confirmed by successes in controlling variety of difficult processes like paper machines, cold-rolling mills, rotary kilns, heat exchangers, drum boilers etc. [3, 4], where their superior robustness with respect to ill–defined structural parameters and unmodelled dynamics has been used with advantage.

Nonetheless, the computational requirements imposed by the control applications often cannot be met by the standard sequential implementations of LQ control. This is the more the issue in the case of advanced implementations with robustified identification used to improve the stability of control [5]. This limitation has motivated research of *fast computational schemes*, exploiting special nature of the control algorithms. Over the last years, a number of fast algorithmic schemes was developed, achieving savings of an order of magnitude. This process was further stimulated by development of the digital signal processor (DSP) in 80s.

A major impact in the design of algorithms had the advent of *systolic/wavefront arrays*. The advances in VLSI technology, together with the notion of systolization, have made it possible to design special-purpose array processors. This, in turn, resulted in design of special algorithms with structural features that make them amenable to *VLSI systolic implementation*, like modularity, pipelineability and communications locality. Finally, the design of highly parallel/pipelined algorithms has also benefited from the development of programmable processors specially designed for parallel implementations, such as *transputers*.

Nonetheless, the problem of the controversial character of demands on the robustness of control and on the speed of control had no solution until recently, when the modification of the regularization, suitable for systolic implementation, has been developed [6].

In this paper, an adaptive controller will be presented, implementing the block regularization in the systolic square-root covariance filter [7, 8].

2. Adaptive LQ controller

Adaptive controller adapts to the dynamics of an a priori unknown and/or slowly varying system. A necessary prerequisite for the adaptive control is recursive identification of the controlled system.

The scheme of LQ controller is on Fig. 1. It consists of three processes:

Identification. For identification, we shall use recursive estimation of a linear regression model, with

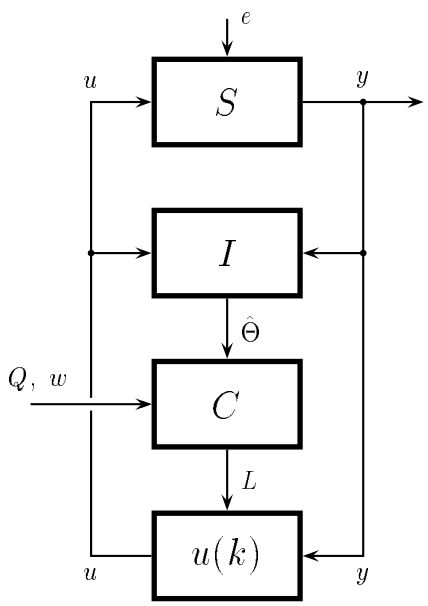


Figure 1: Basic scheme of an adaptive control loop

$O(n^2)$ operations per sample interval required; n being the model order. The usual way to achieve adaptivity is to weight the data samples by an exponential weight, which allocates the greatest significance to the newest samples. However, numerical difficulties may be encountered in the identification procedure in the conditions of the closed loop control, because of insufficiently exciting data, if only the exponential weighting is used.

To prevent these difficulties, regularization may be used [9]. In [5], a generalized framework for regularized parameter tracking was proposed, which does, however, increase the number of required operations to at least $O(n^3)$ per sampling period.

Control Law Design. The computation of the controller parameters involves minimization of a quadratic cost function on a horizon $k_f = k+N, k+N-1, \ldots, k_0 = k$ (k is the current discrete time, k_0 is the beginning, k_f is the end of the control horizon and N is the length of the horizon).

In the dynamic programming, a symmetric positive definite kernel of a quadratic form is updated. This procedure may be implemented with advantage using the square root factorization [3], which requires $N * O(n^3)$ operations. A reduction to $N * O(n^2)$ may be achieved by using an ARX model [10].

Computation of the System Input. It requires $O(n)$ operations per sampling interval to compute the system input from the control law and data.

Counting the above given figures, we find that the adaptive LQ controller requires as much as $O(n^3) + N * O(n^3) + O(n)$ operations per sampling period. Clearly, this large computational burden limits the sequential implementation to slow and/or low-order applications.

2.1. Existing systolic implementations

Systolic estimator. For the systolic implementation of the identification algorithm, a *square-root information filter* (SRIF) has often been used, as an algorithm "well-fitted" for systolization. A good overview of the parallel implementations may be found in [11, 12]. We can name the algorithms of the QR and QRD family [13] as typical representatives of this group.

Another group of the systolic estimators [14] originates from the UD factorized covariance filters. The advantage of this group is in providing directly the estimated parameters. Also, since the algorithms work with the factor of the covariance matrix, they may be used for implementation of the cautious control strategy [3, 7].

A regularized modification, developed by the author of this paper, will be used here.

Systolic controller. Many of the systolic implementations of the LQ controller were based on the relation of the LQ optimization with the Kalman filtering [15]. A simplification of the systolic array, resulting in a further speedup, may be achieved [3, 10] by exploitation of the special structure of the ARX model.

Also in this paper an array based on the ARX model will be used, using a modified feeding of the penalties and setpoints, developed by the author in [7]. This method gives the user possibility to decide the penalty immediately before it is used. This is expected to be advantageous for solving the problems of the control design with input restrictions [16].

3. Systolic block-regularized identification

3.1. ARX model

Let the plant be described by an ARX model

$$\begin{aligned} y(k) &= \textstyle\sum_{i=1}^{l} a_i y(k-i) + \\ &\quad + \textstyle\sum_{i=1}^{l+1} b_i u(k-i) + e(k) \qquad (1) \\ &= \Theta' \varphi(k-1) + e(k); \end{aligned}$$

$$\begin{aligned} \Theta' &= [b_1, a_1, b_2, \ldots, a_l, b_{l+1}] = [b_1 \ \Theta'_x], \\ \varphi'(k-1) &= [u(k-1), y(k-1), u(k-2), \ldots \\ &\quad \ldots, y(k-l), u(k-l-1)]. \qquad (2) \end{aligned}$$

where u are the inputs, y are the outputs, e is assumed to be a normal white noise, and a and b are the plant coefficients.

For later use we introduce

$$m = \dim(\Theta) = 2l + 1.$$

3.2. Regularized identification

Let us introduce the *regularized loss criterion* [5, 6]:

$$\begin{aligned} \bar{J}(\Theta) &= \lambda\tilde{J}(\Theta) + (1-\lambda)J^*(\Theta) + \\ &\quad +(y(k) - \hat{\Theta}'\varphi(k-1))^2, \qquad (3) \\ &\quad 0 < \lambda \leq 1. \end{aligned}$$

The 'tilde' ($\tilde{\bullet}$), 'bar-tilde' ($\bar{\tilde{\bullet}}$) and 'bar' ($\bar{\bullet}$) symbols denote quantities before the data update, before the time update and after the time update, respectively.

The quadratic forms $J(\Theta)$ and $J^*(\Theta)$ with positive definite kernels are defined by the formulae

$$J = [\Theta - \hat{\Theta}]'P^{-1}[\Theta - \hat{\Theta}] + \Lambda, \qquad (4)$$

$$J^* = [\Theta - \Theta^*]'(P^*)^{-1}[\Theta - \Theta^*] + \Lambda^*. \qquad (5)$$

Λ and Λ^* are scalar quantities independent of $\hat{\Theta}_\bullet$. They do not need to be known explicitly because the estimates are independent of them.

$\hat{\Theta}$ denotes the parameter estimate, P denotes the covariance matrix.

Note on notation. As we shall work only with the estimates, we shall not use the 'hat' sign. Instead, we shall use the 'tilde' and 'bar' signs to refer the values before and after updates.

The exponential weight λ and the shape and position of the alternative, regularizing quadratic form J^* (which is characterized by P^*, Θ^*), are specified by the user.

The regularized time update attracts the information matrix P and the parameter estimate $\Theta(k)$ to the preferred, preselected values P^* and Θ^*.

Using the inverse update, the parameter estimates are related with the covariance matrix by the following formulae of the *data update*:

$$\begin{aligned} \varepsilon &= y - \tilde{\Theta}'\varphi \\ \zeta &= \varphi'\tilde{P}\varphi \\ \kappa &= -(1+\zeta)^{-1}\tilde{P}\varphi \\ \bar{\tilde{\Theta}} &= \tilde{\Theta} - \kappa\varepsilon \\ \bar{\tilde{P}} &= \tilde{P} - (1+\zeta)\kappa\kappa' \end{aligned}$$

where ε is the prediction error and κ is the Kalman gain vector.

The Direct time update of the regularized exponential algorithm is specified by the relations:

$$\begin{aligned} \bar{P}^{-1} &= \lambda\bar{\tilde{P}}^{-1} + (1-\lambda)(P^*)^{-1} \\ A &= [I + (1/\lambda - 1)\bar{\tilde{P}}(P^*)^{-1}]^{-1} \\ \bar{\Theta} &= A\bar{\tilde{\Theta}} + (I - A)\Theta^* \end{aligned}$$

In general, the regularized estimator tracks the parameter changes reflected in the data measurements. If this information is missing in the data for a longer period, the estimator forgets the information contained in the measurements and converges towards the stage given by P^*, Θ^*. This is important in the closed loop identification, where $u(k)$, generated by the linear controller, may be nearly linearly dependent on the data.

3.3. Block regularization

To preserve pipelining in the systolic implementation, the block regularized algorithm is used [6, 7].

The block regularization is based on the assumption that the regularizing parameters will not change over $M \geq m$ periods of identification. Under this assumption, it is possible to split the regularized identification into the phases: the first, where the identification with exponential forgetting (over a data block M periods long) is performed; and the second, where the regularizing data are added, weighted by so called *block weight* (sum of weights of the regularizing matrix in each step in the block):

$$\lambda_M = \sum_{i=0}^{M-1} [\lambda^i(1-\lambda)]$$

The regularization may be also seen as a sequence of rank one data updates of a positive definite symmetric matrix.

3.4. Square-root implementation

Usually we work with a square-root factor R of the covariance matrix, rather than with the matrix itself,

$$P = RR',$$

where R is an upper triangular matrix.

Let us denote the square-root factor of $(P^*)^{-1}$ as U^*:

$$(P^*)^{-1} = (U^*)'U^*$$

The data update is realized by orthogonal rotations:

$$\begin{bmatrix} s^{\frac{1}{2}} & s^{\frac{1}{2}}\kappa' \\ 0 & \bar{\tilde{R}}' \end{bmatrix} = \lambda_n Q \begin{bmatrix} 1 & 0 \\ a & \tilde{R}' \end{bmatrix},$$

where $a = -\tilde{R}'\varphi$, $s = 1 + aa' = 1 + \zeta$, Q is an orthogonal matrix and $\lambda_n = \mathrm{diag}\{1\ \lambda I_n\}$, I_n is an (n, n) unit matrix.

In the regularization phase, the regularizing data is entered, $\varphi_i^* = \lambda_M U_i^*$, $i = 1, \ldots, n$. The regularizing parameters Θ^* are entered through a product $y^* = U^*\Theta^*$. The forgetting is switched off during processing of the alternative data.

In our implementation, we choose the parameter estimates m steps old for the regularizing parameters. The advantages of this choice are easy implementation and preservation of the old value of estimates if the input data is not sufficiently exciting.

A detailed derivation of the mechanism is beyond the scope of this paper. Reader may find it in [6, 8].

3.5. Systolic implementation

The mapping of the estimation algorithm to the systolic array and loading of the data and control matrix (used to switch the inputs and forgetting) is shown on Fig. 2.

The data samples enter the array through the diagonal cells. The parameter estimates emerge from the bottom row. In the same row is also entered, skewed in time, the regularizing data. It is multiplied there with the regularizing estimates (stored in the memory cells during multiplication), and then shifted up through the array. The product of multiplication is shifted rightwards to the cell denoted as "1". When the regularizing data reaches the diagonal cells, the cell inputs are switched to it.

Together with the regularizing data, a control matrix is propagated through the array, to control the multiplication with the estimates and switching of the entries and of forgetting.

The detailed description of this array may be found in [7, 8].

4. Systolic LQ controller

In this section, a square-root algorithm for the LQ control synthesis will be derived and its systolic implementation will be described briefly. An LQ output regulation problem for a single input, single output (SISO) plant, using the receding-horizon control strategy, will be considered.

4.1. Derivation of the square-root algorithm for the LQ optimization

The following *cost function* K is assumed for optimization

$$K = \sum_{k=k_0}^{k_f} Q_y e_y^2(k) + Q_u e_u^2(k-1), \qquad (6)$$

where $N = k_f - k_0 + 1$ is the length of the control horizon, k_0 is the beginning and k_f is the end of the control horizon, and the terms e_y and e_u are defined as

$$e_y(k) = y(k) - w_y(k), \;\; e_u(k) = u(k) - w_y(k).$$

$w_y(k)$ is the setpoint of y and $w_u(k)$ is the setpoint of u (both setpoints are the user's choice).

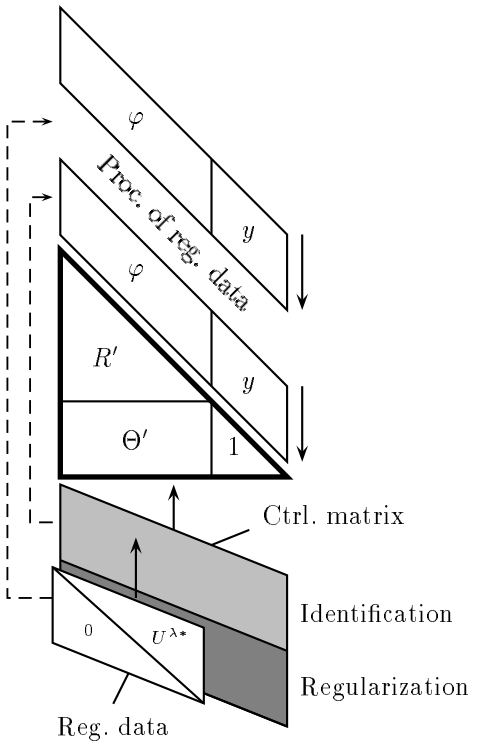


Figure 2: Mapping of variables and inputs in the regularized estimator

Since we suppose the use of the certainty-equivalence strategy [3], we need not count with the noise in (1).

Let us split now the data samples vector φ (2) in the following way:

$$\varphi(k) = [u(k)\; x(k)]. \qquad (7)$$

The ARX model may be rewritten in the state-space form:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + Ce(k+1), \quad (8)\\ y(k) &= C'x(k), \quad (9)\end{aligned}$$

$$A = \begin{bmatrix} \Theta'_x \\ 0_{1,2l} \\ I \;\; 0 \;\; 0 \end{bmatrix}; B = \begin{bmatrix} b_1 \\ 1 \\ 0 \end{bmatrix}; C = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \qquad (10)$$

Using the ARX model, the cost function (6) can be written as

$$\begin{aligned} K = {} & Q_y[b_1 u(k_f - 1) + \Theta'_x x(k_f - 1) - \\ & - w_y(k_f)]^2 + Q_u[u(k_f - 1) - w_u(k_f - 1)]^2 + \\ & + \sum_{k=k_0}^{k_f - 1} Q_y e_y^2(k) + Q_u e_u^2(k-1). \qquad (11)\end{aligned}$$

Initial step of optimization. Consider the first two terms of K alone and call them K_1, i.e.

$$K_1 \;\; = \;\; Q_y[b_1 u(k_f - 1) +$$

$$+\Theta'_x x(k_f-1) - w_y(k_f)]^2 + \\ +Q_u[u(k_f-1) - w_u(k_f-1)]^2. \quad (12)$$

This expression may be rewritten as a quadratic form with a kernel in the square-root form:

$$K_1 = [u(k_f-1), x'(k_f-1), 1]\Phi'(k_f) \\ \Phi(k_f)[u(k_f-1), x'(k_f-1), 1]' \quad (13)$$

$$\Phi(k_f) = \left[\begin{array}{cc|c} Q_y^{1/2}\Theta' & & d_y(k_f) \\ \hline Q_u^{1/2} & 0_{(1,2l)} & d_u(k_f-1) \\ \hline 0_{(2l,m)} & & 0_{(2l,1)} \end{array}\right] \quad (14)$$

$$d_y(k_f) = -Q_y^{1/2} w_y(k_f), \quad (15)$$

$$d_u(k_f-1) = -Q_u^{1/2} w_u(k_f-1). \quad (16)$$

In order to minimize K_1, we multiply the matrix $\Phi(k_f)$ by an orthogonal matrix $T(k_f)$ ($T'(k_f)T(k_f) = I$), so that the first row of matrix $\Phi(k_f)$ is zeroed (i.e. we perform a *QR-update* on matrix $\Phi(k_f)$):

$$T(k_f)\Phi(k_f) = \left[\begin{array}{ccc} & 0 & \\ \hline P_1(k_f) & P_2'(k_f) & m(k_f) \\ 0 & R(k_f) & n(k_f) \\ 0 & 0 & o(k_f) \end{array}\right]. \quad (17)$$

In the resulting matrix, P_1, m and o are scalars, P_2 and n are vectors and R is an upper triangular matrix.

Using the transformed matrix, K_1 reads (time index omitted for brevity):

$$K_1 = [u\ x'\ 1][P_1\ P_2'\ m]'[P_1\ P_2'\ m][u\ x'\ 1]' + \\ +[x'\ 1][R\ n]'[R\ n][x'\ 1]' + o^2.$$

This expression is minimized by setting

$$[P_1\ P_2'\ m][u\ x\ 1]' \stackrel{!}{=} 0, \quad (18)$$

i.e.

$$u = -\frac{1}{P_1}[P_2'x + m] = Sx + S_0, \quad (19)$$

$$S = -\frac{1}{P_1}P_2';\ \ S_0 = -\frac{1}{P_1}m. \quad (20)$$

The residual value of K_1 after the minimization is

$$K_1^* = [x'\ 1][R\ n][R\ n]'[x\ 1]' + o^2. \quad (21)$$

Standard step of optimization. To show the standard step of optimization, we shall rewrite the cost function in the form:

$$K = K_1^* + Q_y e_y^2(k_f-1) + Q_u e_u^2(k_f-2) + \\ +\sum_{k=k_0}^{k_f-2} Q_y e_y^2(k) + Q_u e_u^2(k-1). \quad (22)$$

and denote the first three terms as K_2:

$$K_2 = [x'(k_f-1)\ 1][R\ n]' \\ [R\ n][x'(k_f-1)\ 1]' + o^2(k_f) + \\ +Q_y e_y^2(k_f-1) + Q_u e_u^2(k_f-2) \quad (23)$$

For further use, we shall denote the matrix R, resulting from the previous step of optimization, as $\tilde{R}$ and vector n as $\tilde{n}$.

Using the state-space model (9), we can express the square roots of the first and third terms in the above relation for K_2 as functions of $u(k_f-2)$ and $x(k_f-2)$ (we work only with the square roots now, not with the whole quadratic form):

$$\begin{aligned} [\tilde{R}\ \tilde{n}][x(k_f-1)\ 1] = \\ &= \tilde{R}(Bu(k_f-2) + Ax(k_f-2)) + \tilde{n}, \\ Q_y^{1/2}e_y &= Q_y^{1/2}[b_1 u(k_f-2) + \\ &\quad +\Theta'_x x(k_f-2) - w_y(k_f-1)] \\ &= Q_y^{1/2}[b_1\ \Theta'_x]\left[\begin{array}{c} u(k_f-2) \\ x(k_f-2) \end{array}\right] + \\ &\quad +d_y(k_f-1), \\ d_y(k_f-1) &= -Q_y^{1/2} w_y(k_f-1). \end{aligned}$$

Using this expression, we can reformulate the addition of $Q_y e_y^2$ to the residual of the criterion from the last step as a QR-update on the square-root form:

$$Q_1\left[\begin{array}{cc} Q_y^{1/2}0_{1,2l} & d_y \\ \tilde{R} & \tilde{n} \\ 0_{1,m} & \tilde{o} \end{array}\right]\underline{M} = \left[\begin{array}{cc} 0 & 0 \\ \bar{\tilde{R}} & \bar{\tilde{n}} \\ 0 & \bar{\tilde{o}} \end{array}\right]\underline{M}, \quad (24)$$

$$\underline{M} = \left[\begin{array}{ccc} & \Theta' & 0 \\ I & 0_{2l-1,2} & 0 \\ & 0_{1,m} & 1 \end{array}\right].$$

Let us split matrix $\bar{\tilde{R}}$ into three parts:

$$\bar{\tilde{R}} = \left[\begin{array}{cc} \bar{\tilde{R}}_1 & \bar{\tilde{R}}_2 \\ 0 & \bar{\tilde{R}}_3 \end{array}\right], \quad (25)$$

where $\bar{\tilde{R}}_1$ is a scalar, $\bar{\tilde{R}}_2$ is a transposed vector and $\bar{\tilde{R}}_3$ is an upper triangular matrix.

Using this partitioning, it is possible to rewrite the right-hand side of (24) as follows:

$$\left[\begin{array}{cc} \bar{\tilde{R}} & \bar{\tilde{n}} \\ 0 & \bar{\tilde{o}} \end{array}\right]\underline{M} = \left[\begin{array}{cc} \Psi & \bar{\tilde{n}} \\ 0 & \bar{\tilde{o}} \end{array}\right], \quad (26)$$

$$\Psi = \left[\begin{array}{c} \bar{\tilde{R}}_1\Theta' + [\bar{\tilde{R}}_2\ 0\ 0] \\ \hline \tilde{R}_3 \quad 0_{2l-1,2} \\ 0_{1,m} \end{array}\right] = \left[\begin{array}{c} \Psi_1 \\ \Psi_2 \end{array}\right], \quad (27)$$

$$\bar{\tilde{n}} = [\bar{\tilde{n}}_1 \; \bar{\tilde{n}}_2 \; 0 \; 0]. \qquad (28)$$

The row vector Ψ_1 contains the first row, and the upper triangular matrix Ψ_2 the rest of matrix Ψ. Scalar $\bar{\tilde{n}}_1$ contains the first element, and vector $\bar{\tilde{n}}_2$ the rest of vector $\bar{\tilde{n}}$.

The square root of the last term in (23) reads exactly as in the first step of optimization:

$$\begin{aligned} & Q_u^{1/2} e_u(k_f - 2) = \\ & \quad = Q_u^{1/2}[u(k_f - 2) - w_u(k_f - 2)] \\ & \quad = Q_u^{1/2} u(k_f - 2) + d_u(k_f - 2), \qquad (29) \\ & d_u(k_f - 2) = -Q_u^{1/2} w_u(k_f - 2). \end{aligned}$$

This expression will be added using an orthogonal rotation w.r.t. Ψ_2, $\bar{\tilde{n}}_2$, $\bar{\tilde{o}}$ (see (26), (27) and (28)).

$$Q_2 \begin{bmatrix} Q_u^{1/2} & 0_{2l} & d_u \\ \Psi_1 & & \bar{\tilde{n}}_1 \\ \Psi_2 & & \bar{\tilde{n}}_2 \\ 0 & & \bar{\tilde{o}} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ \Psi_1 & \bar{\tilde{n}}_1 \\ \bar{\Psi}_2 & \bar{n}_2 \\ 0 & \bar{o} \end{bmatrix} = \Phi \qquad (30)$$

The third orthogonal rotation will be used to minimize the cost function by reducing the vector $[\Psi_1 \; \bar{\tilde{n}}_1]$ w.r.t. the upper triangular part of the matrix:

$$T\Phi = T \begin{bmatrix} 0 & 0 \\ \Psi_1 & \bar{\tilde{n}}_1 \\ \bar{\Psi}_2 & \bar{n}_2 \\ 0 & \bar{o} \end{bmatrix} = \left[\begin{array}{cc|c} \multicolumn{2}{c|}{0_{2\times(m+1)}} & \\ \hline P_1 & P_2' & m \\ 0 & R & n \\ 0 & 0 & o \end{array}\right]. \qquad (31)$$

The resulting submatrices are the same as in (17).

By repeating the above described procedure until the beginning of the control horizon, we get the control law $[S(k_0) \; S_0(k_0)]$ to be used to compute the control action u.

4.2. Remarks on algorithm

In the step described by (26), the matrix R_3 is simply shifted two positions up along the main diagonal. The vector $\bar{\tilde{n}}_2$ is shifted two positions up.

The scalar o (17, 31) contributes to the cost function only passively and does not have any influence on the control law. It is not needed in the computation of the control law and can be ignored.

4.3. Systolic implementation of the LQ controller

The above described control algorithm will be mapped onto an $m \times m$ triangular systolic array.

Mapping of Ψ_1 and Ψ_2. To eliminate an unnecessary latency, both the vector Ψ_1 and the upper row of the matrix Ψ_2 (27) are mapped onto the same row of the systolic array (see Fig. 3).

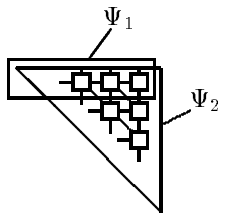


Figure 3: Mapping of Ψ_1 and Ψ_2

Mapping of n_1 and n_2. The setpoint processing column ($[n_1 \; n_2]$) (30) is mapped onto the diagonal of the array (see Fig 4).

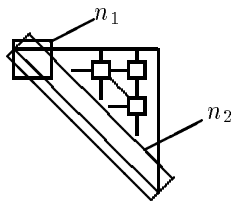


Figure 4: Mapping of n_1 and n_2

Using this mapping, the elements of $\bar{\tilde{n}}$ are, between the iterations, shifted along the diagonal in the same way as the elements of $\tilde{\bar{R}}_3$. Also, since d_y and d_u are calculated in the upper left cell of the array, $Q_u^{1/2}$ and $Q_y^{1/2}$ do not have to be sent through the array.

Operation modes and their propagation. The orthogonal rotations propagate through the array from left to right. The result of reduction of $Q_y^{1/2}$ in (24) (matrix $\tilde{\bar{R}}$ in (25)) is immediately shifted up along the main diagonal.

The *cells work in two modes* – in the *first mode*, they perform the reduction (24) and shift the resulting matrix $\tilde{\bar{R}}$. In the *second mode*, they perform the reductions (30) and (31). The operation modes propagate through the array as sketched on Fig. 5.

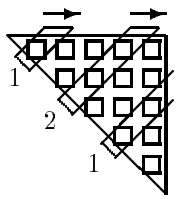


Figure 5: Switching of the function modes

The function of the cells in **the first row** of the array is more complicated. In the *first mode*, they perform the rotation (24), using the lower-right diagonal entry as input and passing the result to

the left, and multiply Θ by $\bar{\tilde{R}}_1$ (27). In the *second mode*, the computation of Ψ_1 is completed by adding $[\bar{\tilde{R}}_2\ 0\ 0]$ and the rotations (30) and (31) are computed. If the beginning of the control horizon was reached, the control law is computed and passed to the systolic array computing the system input u.

Loading of the array. The parameter estimates are entered to the upper row of the array, skewed in time. As soon as the parameter b_1 (leftmost in the vector) enters the array, the array restarts the operation. The penalties and setpoints are entered through the upper left cell of the array (see Fig. 6).

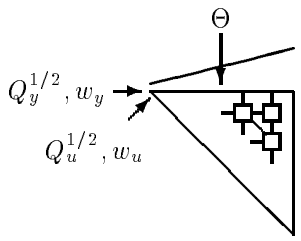


Figure 6: Input of parameters, penalties and setpoints

5. Complete scheme of the controller

On Fig. 7, the complete scheme of the regularized adaptive controller is presented. Let us give some explanations to this scheme.

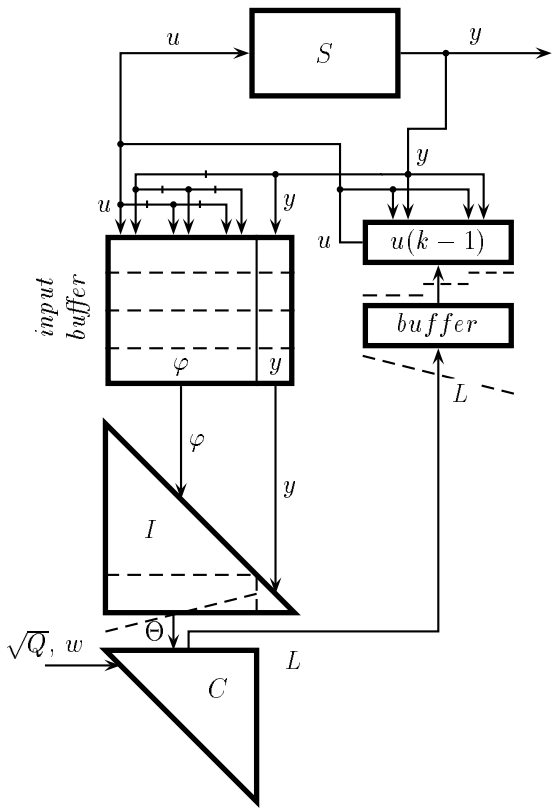
The delay cells at the entry of the input buffer form the vector $x(k-1)$ from $y(k)$ and $u(k-1)$. The *input buffer* is necessary since the processing of the input data temporarily stops during the regularization. The array marked in Fig. 7 by I is the regularized estimator. The input of the regularizing data (see Fig. 2) is not drawn for brevity.

The parameter estimates, emerging from the bottom row of the estimator, are passed to the LQ controller (denoted by C).

5.1. Array for the system input generation, buffering of the control law

The computed control law parameters emerging from the top of the controller are latched into a *buffer*, and passed on into the systolic *generator of the system input*, denoted by $u(k-1)$ in Fig. 7. The system input generator implements (19) for $S(k)$ and $S_0(k)$ by utilizing the shifting structure of $x(k-1)$ to generate $u(k-1)$.

The reason for use of the buffer between the output of the controller and the input of the input generator $u(k-1)$ is not to introduce additional transport delay to the control loop when generating the system input. The buffer reverts the time skew of the control law on the output of the controller, so that the product $S_1 y(k-1)$ (20) is computed last, immediately before $u(k-1)$ is applied.



Symbol —|— denotes one step delay element

Figure 7: Complete scheme of the adaptive LQ controller

The input generator works with a constant control law until the new law is available in the buffer.

The estimator and controller run at the same speed. They run, however, faster than the input generator, because the estimator has to process the regularizing data in addition to the samples from the system.

Remarks on timing. We assume that all of the processing elements of the arrays need an equal, constant time Δ for the elementary systolic operation. The LQ design array (C in Fig. 7) needs time $T_C = 6N\Delta$ to compute the new control law parameters for the control horizon N. This time defines the basic period of adjustment of the control law for the system input generator ($u(k-1)$ in Fig. 7).

The estimator I needs time $T_I = 2(M+m)\Delta$ to complete one identification cycle [7]. It is necessary to specify the length of the data M so that one or several identification cycles fit exactly into the control law computation time T_C. Denoting the count of the cycles by $j \geq 1$, the value of j comes

from $T_C = jT_I$ and $T_I = 2(M+m)\Delta$ with $M \geq m$.

The identification section samples jM data pairs $y(k), u(k-1)$ in each control design period T_C. This defines the clock period k of the system input generator $(u(k-1))$ as $k = T_C/(jM) = 2\Delta(M+m)/M$.

The controller interacts with the controlled system sampled with the period k. Hence, we can measure the period of the control law readjustment $T_C = jMk$ by the count of the sampling periods k.

6. CONCLUSIONS

An adaptive controller, made up of a robust block regularized parallel estimator and of a linear quadratic controller, was described in the paper. While meeting the speed demands, often encountered in the practical applications of adaptive control, the algorithm has increased robustness and hence improves stability of the control loop.

Using an ARX model of the system, an iteration period independent on the system order was achieved for the control law design.

The derivation of the controller treats the case of a non-zero setpoint, which is important for practical applications. A modified mechanism of entering the penalties and setpoints was used, which is expected to be more suitable for the control under constraints, than the standard mechanism [10].

The implementation can be used both for the preprogrammed tracking with the setpoints $w_y(k)$, $w_u(k)$ not constant, but known beforehand, and for the regulation problem ($w_y(k) = w_y = const.$, $w_u(k) = w_u = const.$). The penalties Q_y, Q_u, used in the optimization, can be time variable.

The efficiency of the array, implementing the LQ controller, is about 33%. A higher efficiency may be achieved by grouping several systolic cells on one processor.

The detailed description of the systolic algorithms used in this paper is available by an anonymous ftp to *ftp.utia.cas.cz*.

References

[1] V. Peterka, "Predictor-based self-tuning control," *Automatica—J. IFAC*, vol. 20, no. 5, pp. 39–50, 1984.

[2] D. W. Clarke, C. Mohtadi, and P. C. Tuffs, "Generalized predictive control- part I. the basic algorithm," *Automatica—J. IFAC*, vol. 23, no. 2, pp. 137–148, 1987.

[3] M. Kárný *et al.*, "Design of linear quadratic adaptive control: theory and algorithms for practice," *Kybernetika (Prague)*, vol. 21, 1985. Supplement to No. 3, 4, 5, 6.

[4] J. Böhm and R. Kulhavý, "Towards more credible self-tuners...," in *Advanced Methods in Adaptive Control for Industrial Applications* (K. Warwick, M. Kárný, and A. Halousková, eds.), vol. 158 of *Lecture Notes in Control and Information Sciences*, pp. 189–205, Springer-Verlag, 1990.

[5] R. Kulhavý, "Restricted exponential forgetting in real-time identification," *Automatica—J. IFAC*, vol. 23, pp. 589–600, 1987.

[6] J. Kadlec, "The cell level description of systolic block regularised QR filter," in *IEEE Workshop on VLSI Signal Processing*, (Koningshof, Veldhoven, Netherlands), pp. 298–306, 1993.

[7] J. Schier, *Parallel algorithms for robust adaptive identification and square-root LQG control synthesis*. PhD thesis, ÚTIA AVČR, Prague, Czech Republic, 1994.

[8] J. Schier, "A systolic algorithm for the block-regularized RLS identification," Res. Report 1807, ÚTIA AVČR, 1994. Accepted for publication in Kybernetika (Prague) journal.

[9] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. Cambridge, MA: MIT Press, 1983.

[10] L. Chisci and G. Zappa, "Architecture for iterative lq optimization," *Automatica—J. IFAC*, vol. 27, no. 5, pp. 799–810, 1991.

[11] B. Friedlander, "Lattice filters for adaptive processing," *Proc. IEEE*, vol. 70, no. 8, pp. 829–867, 1982.

[12] N. Kalouptsidis and S. Theodoris, *Adaptive System Identification and Signal Processing Algorithms*. Prentice Hall International Series in Acoustics, Speech and Signal Processing, UK: Prentice-Hall, 1993.

[13] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*. New York: Academic Press, 1977.

[14] M. Moonen and J. McWhirter, "Systolic array for recursive least squares by inverse updating," *Electronics Letters*, vol. 29, no. 13, pp. 1217–1218, 1993.

[15] F. Gaston and G. Irwin, "Systolic kalman filtering: an overview," *Proc. IEE*, vol. 137, no. 4, pp. 235–244, 1990.

[16] J. Böhm and M. Kárný, "Self-tuning regulators with restricted inputs," *Kybernetika (Prague)*, vol. 18, no. 6, pp. 529–544, 1982.