Akademie věd české republiky
ÚSTAV TEORIE INFORMACE A AUTOMATIZACE

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

RESEARCH REPORT

F.M.F. Gaston, J. Kadlec, J. Schier

The Block Regularised Linear Quadratic Optimal Controller

No. 1789 January 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 BLOCK REGULARISED LINEAR QUADRATIC OPTIMAL CONTROLLER

F.M.F. Gaston*, J. Kadlec‡, J. Schier‡

* The Queen's University of Belfast, N.I.

‡ The Institute of Information Theory and Automation,
Czech Academy of Sciences, Czech Republic

INTRODUCTION

Linear quadratic multistep (LQ) adaptive controllers [1, 2] are important because of their superior robustness with respect to ill–defined structural parameters and unmodeled dynamics. However, large-order or high speed, industrial control applications impose computational requirements that cannot be met by the standard sequential implementations of LQ control. This limitation has motivated research into parallel and VLSI array implementations of this controller for these type of applications.

The classical configuration of the LQ controller consists of three procedures: identification, control law design and computation of the control action:

Identification. Identification is often carried out using adaptive recursive estimation of a linear regression model where $O(n^2)$ computations per sample interval are required; n being the model order. However, in closed loop control, if simple exponential weighting is used, numerical difficulties may occur in this procedure because the data is insufficiently exciting or some of the parameters are redundant. To prevent these difficulties regularisation is used [4], [5] has proposed a generalized framework for regularised parameter tracking but it does, however, increase the number of computations required to at least $O(n^3)$ per sample interval.

Control Law Design. The controller parameters are computed using the current parameter estimates and involves minimizing a quadratic cost function on the horizon $k+N, k+N-1, \ldots, k$ where k is the current discrete time and N is the length of the horizon. The dynamic programming updates a symmetric, positive definite kernel of a quadratic form, which can be implemented using square root factorization which requires $N * O(n^3)$ computations [8]. A reduction to $N * O(n^2)$ may be achieved by using an i/o canonical representation [7].

Calculation of the Control Input. The calculation of the control input from the control law which was computed in the previous procedure requires $O(n)$ computations per sample interval.

This means that the combined LQ controller has as much as $O(n^3) + N * O(n^3) + O(n)$ computations per sample interval. Clearly, this large computational burden limits sequential implementations to slow, low-order applications.

The objective of the paper. In this paper we will show how a sample interval of $N * O(1)$ for the LQ adaptive controller may be achieved using a pipelined parallel implementation.
We will present:

- A novel adaptive parameter estimator, whose data sample speed is $O(1)$ and generates the parameter estimates for the controller with a speed of $O(n)$ on $O(n^2)$ processing elements.
- An LQ control designer, whose speed is $N * O(1)$ on $O(n^2)$ processing elements.
- A systolic implementation of the control action computation, generating the control signal with a speed of $O(1)$ on $O(n)$ processing elements.

SYSTOLIC BLOCK REGULARISED IDENTIFICATION

The plant is described by an ARX model

$$\begin{aligned} y(k) &= b_1 u(k-1) + \\ &\quad + \textstyle\sum_{i=1}^{l} a_i y(k-i) + b_{i+1} u(k-i-1) + e(k) \\ &= [b_1, \Theta'][u(k-1), x'(k-1)]' + e(k); \\ \Theta' &= [a_1, b_2, \ldots, a_l, b_{l+1}]; \\ x(k-1) &= [y(k-1), u(k-2), \ldots \\ &\quad \ldots, y(k-l), u(k-l-1)]' \end{aligned} \tag{1}$$

where u are the inputs, y are the outputs, e is assumed to be a normal white noise, a_i are the plant coefficients related to the system output and b_i are the coefficients related to the system input. The length of the vector Θ is $n = 2 * l$.

The ARX model can be rewritten as a state-space model as in equations (2) and (3)

$$x(k+1) = Ax(k) + Bu(k) + Ce(k+1) \tag{2}$$

$$y(k) = C'x(k) \tag{3}$$

where the matrices A, B and C are

$$A = \left[\begin{array}{c|cc} \multicolumn{3}{c}{\Theta'} \\ \hline \multicolumn{3}{c}{0} \\ \hline I & 0 & 0 \end{array}\right]; B = \left[\begin{array}{c} b_1 \\ \hline 1 \\ \hline 0 \end{array}\right]; C = \left[\begin{array}{c} 1 \\ \hline 0 \\ \hline 0 \end{array}\right]. \tag{4}$$

Regularised recursive identification estimates the vector $[b_1, \Theta']$ from the data measurement $u(1), y(1), \ldots, u(k), y(k)$ by minimizing the criterion:

$$[b_1(k), \Theta'(k)] = \arg\left\{ \min_{[b_1,\Theta']} J_k([b_1, \Theta']) \right\} \tag{5}$$

The standard exponentially weighted, recursive least squares error criterion with a scalar exponential weighting factor $0 < \lambda < 1$ in the normal adaptive controller is replaced with a recursive regularised criterion [5],[10]:

$$\begin{aligned} J_k([b_1, \Theta']) &= \lambda J_{k-1}([b_1, \Theta']) + (1-\lambda) J^*([b_1, \Theta']) + \\ +(y(k) &- [b_1, \Theta'][u(k-1), x'(k-1)]')^2 \end{aligned} \tag{6}$$

where the quadratic forms $J_k([b_1, \Theta'])$ and $J^*([b_1, \Theta'])$ are characterized by the matrices

$$\left[\begin{array}{c|c} V(k) & d(k) \\ \hline d'(k) & \sharp \end{array}\right] = \left[\begin{array}{c|c} L_1(k) & 0 \\ \hline L_2'(k) & \sharp \end{array}\right] \left[\begin{array}{c|c} L_1'(k) & L_2(k) \\ \hline 0 & \sharp \end{array}\right] \tag{7}$$

$$\left[\begin{array}{c|c} V^* & d^* \\ \hline d^{*T} & \sharp \end{array}\right] = \left[\begin{array}{c|c} L_1^* & 0 \\ \hline L_2^{*T} & \sharp \end{array}\right] \left[\begin{array}{c|c} L_1^{*T} & L_2^* \\ \hline 0 & \sharp \end{array}\right] \tag{8}$$

with the time recursion given by

$$\begin{aligned} \left[\begin{array}{c|c} V(k) & d(k) \\ \hline d'(k) & \sharp \end{array}\right] &= \lambda \left[\begin{array}{c|c} V(k-1) & d(k-1) \\ \hline d'(k-1) & \sharp \end{array}\right] + \\ +(1-\lambda) \left[\begin{array}{c|c} V^* & d^* \\ \hline d^{*T} & \sharp \end{array}\right] &+ \left[\begin{array}{c} u(k-1) \\ x(k-1) \\ \hline y(k) \end{array}\right] \left[\begin{array}{c} u(k-1) \\ x(k-1) \\ \hline y(k) \end{array}\right]' \end{aligned} \tag{9}$$

and also

$$[b_1(k), \Theta'(k)]' = V^{-1}(k) d(k) = (L_1'(k))^{-1} L_2(k) \tag{10}$$

where $V(k), V^*$ are $n+1$ symmetric, positive definite matrices, $L_1(k), L_1^*$ are their $n+1$ lower triangular square root factors, $d(k), d^*$ are $n+1$ vectors and $\sharp$ denotes unimportant scalars. This alternative quadratic form $J^*([b_1, \Theta'])$ has its minimum when

$$[b_1^*, \Theta^{*T}]' = (V^*)^{-1} d^* = (L_1^{*T})^{-1} L_2^*. \tag{11}$$

In recursive identification (9), the user specifies the exponential weight λ and the shape and position of the alternative, regularising quadratic form J^* which is characterized by V^*, b_1^*, Θ^* or L_1^*, L_2^*. This recursion (9) reduces to the standard recursive least squares estimation if $V^* = 0$.

Choosing $V^* > 0$ attracts the information matrix $V(k)$ to V^* and the parameter estimate $b_1(k), \Theta(k)$ to a preferred, selected value b_1^*, Θ^*. This represents the initial information about the parameter estimates in a safe region, leading to "well behaved " control law design and smooth control.

In general, the regularised estimator (9) - (10) tracks the parameter changes reflected in the data measurements. If this information is missing from the data for a longer period, the estimator forgets the information contained in the measurements and converges towards the final stage given by V^*, b_1^*, Θ^* or, identically L_1^*, L_2^*. This is extremely important in closed loop identification, where $u(k)$ is generated by the linear controller.

Block regularised identification

The parameter estimates are required only when the optimal control law is computed which is at the boundaries of the blocks of data.

This allows V^* and b_1^*, Θ^* to be constant within the blocks. In addition, the parameters b_1^*, Θ^* can be automatically adjusted for each new block in order to track the current parameter estimates. Because V^* and b_1^*, Θ^* are constant within the blocks of $m > n$ data measurements, the addition of the regularising matrix

$$\left[\begin{array}{c|c} V^* & d^* \\ \hline d^{*T} & \sharp \end{array}\right] = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \sharp \end{array}\right] + \sum_{i=1}^{i=n+1} \left[\begin{array}{c} L_1^*[i] \\ \hline L_2^*[i] \end{array}\right] \left[\begin{array}{c} L_1^*[i] \\ \hline L_2^*[i] \end{array}\right]' \tag{12}$$

where

$$\left[\begin{array}{c} L_1^*[i] \\ \hline L_2^{*T}[i] \end{array}\right] \hat{=}\ i\text{-th column of } \left[\begin{array}{c|c} L_1^* & 0 \\ \hline L_2^{*T} & \sharp \end{array}\right] \tag{13}$$

can be computed just once in the block as a set of $n+1$ artificial rank-one data updates with a new weight

$$\lambda_{k-m,k} = (\lambda^{m-1} + .. + \lambda + 1)(1 - \lambda). \tag{14}$$

Therefore, the block regularised algorithm is a sequence of rank one data updates of positive definite symmetric matrices. These updates can be implemented as a square root information filter [3] using orthogonal rotations $Q'Q = I$ and this is suitable for systolic implementation. This completes all the definitions required for the block regularised algorithm (15).

BLOCK REGULARISED ALGORITHM

$$
\begin{array}{lr}
\textbf{cycle for } j = m, m-1, \ldots, 1: & \\
\left[\begin{array}{c|c} L_1' & L_2 \\ \hline 0 & \sharp \end{array}\right] = Q \left[\begin{array}{c|c} \lambda^{1/2} L_1' & \lambda^{1/2} L_2 \\ \hline u_{(k-j)}, x'_{(k-j)} & y_{(k-j+1)} \end{array}\right] & (a) \\
\textbf{end of cycle for } j & \\
\left\{ \begin{array}{rcl} \left[\bar{b}_{1(k)}, \bar{\Theta}'_{(k)}\right]' & = & L_1^{-T} L_2 \text{ if aut. adjust:} \\ \left[b_1^*, \Theta^{*T}\right] & \equiv & \left[\bar{b}_{1(k-m)}, \bar{\Theta}'_{(k-m)}\right] \end{array} \right\} & (b) \\
L_2^* = L_1^{*T} \left[b_1^*, \Theta^{*T}\right]' & (c) \\
\textbf{cycle for } i = n+1, n, \ldots, 1: & \\
\left[\begin{array}{c|c} L_1' & L_2 \\ \hline 0 & \sharp \end{array}\right] = Q \left[\begin{array}{c|c} L_1' & L_2 \\ \hline \lambda_{k-m,k}^{1/2} L_1^{*T}[i] & \lambda_{k-m,k}^{1/2} L_2^{*}[i] \end{array}\right] & (d) \\
\textbf{end of cycle for } i & \\
[b_1(k), \Theta'(k)]' = L_1^{-T} L_2 & (e)
\end{array}
\qquad (15)
$$

A detailed derivation and a systolic implementation of the block regularised identification algorithm (15) is given in [10]. The integration of the block regularised identification algorithm into the complete adaptive controller is described after the control algorithm is derived.

SYSTOLIC LQ CONTROLLER

A square root algorithm to synthesize LQ control is described here briefly. An LQ output regulation problem for a SISO plant, using the receding-horizon control strategy, is considered.

The derivation of the square-root algorithm for LQ optimization

The following *cost function* K is assumed

$$K = \sum_{k=k_0}^{k_f} Q_y e_y^2(k) + Q_u e_u^2(k-1), \qquad (16)$$

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

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

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

Using the ARX model, the cost function (16) can be written

$$\begin{aligned} K &= Q_y[b_1u(k_f-1)+\Theta' x(k_f-1)- \\ &\quad -w_y(k_f)]^2+Q_u[u(k_f-1)-w_u(k_f-1)]^2+ \\ &\quad +\textstyle\sum_{k=k_0}^{k_f-1} Q_y e_y^2(k)+Q_u e_u^2(k-1). \end{aligned} \tag{17}$$

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

$$\begin{aligned} K_1 &= Q_y[b_1u(k_f-1)+ \\ &\quad \Theta' x(k_f-1)-w_y(k_f)]^2+ \\ &\quad Q_u[u(k_f-1)-w_u(k_f-1)]^2 \\ &= [u(k_f-1), x'(k_f-1), 1]\Phi'(k_f)\Phi(k_f)\begin{bmatrix} u(k_f-1) \\ x(k_f-1) \\ 1 \end{bmatrix}, \end{aligned} \tag{18}$$

where

$$\Phi(k_f)=\left[\begin{array}{cc|c} \multicolumn{2}{c|}{Q_y^{1/2}[b_1\ \Theta']} & d_y(k_f) \\ \hline Q_u^{1/2} & \multicolumn{1}{|c|}{0_{(1,n)}} & d_u(k_f-1) \\ \hline \multicolumn{2}{c|}{0_{(n,n+1)}} & 0_{(n,1)} \end{array}\right]=\begin{bmatrix} v_1 & v_2 \\ Z_1 & Z_2 \end{bmatrix} \tag{19}$$

$$d_y(k_f) = -Q_y^{1/2}w_y(k_f), \tag{20}$$

$$d_u(k_f-1) = -Q_u^{1/2}w_u(k_f-1). \tag{21}$$

and denote the upper row of matrix Φ as vector v and the rest as matrix Z.

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 shall perform a QR-update of matrix as in equation (19) $\Phi(k_f)$):

$$T(k_f)\Phi(k_f)=\left[\begin{array}{cc|c} \multicolumn{3}{c}{0} \\ \hline P_1(k_f) & P_2'(k_f) & m(k_f) \\ 0 & R(k_f) & n(k_f) \\ 0 & 0 & o(k_f) \end{array}\right], \tag{22}$$

P_1, m and o are scalars, P_2 and n are vectors and R is an upper triangular matrix (in this step, only the first line of R will be non-zero).

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

$$\begin{aligned} K_1 &= [u\ x'\ 1][P_1\ P_2'\ m]'[P_1\ P_2'\ m][u\ x'\ 1]'+ \\ &\quad +[x'\ 1][R\ n]'[R\ n][x'\ 1]'+o^2 \end{aligned}$$

We minimize this expression by setting

$$[P_1 \; P_2' \; m][u \; x \; 1]' \stackrel{!}{=} 0, \tag{23}$$

i.e.

$$\begin{aligned}
u(k_f-1) &= -\frac{1}{P_1(k_f)}[P_2'(k_f)x(k_f-1)+m(k_f)] && (24)\\
&= S(k_f)x(k_f-1)+S_0(k_f) && (25)\\
S(k_f) &= -\frac{1}{P_1(k_f)}P_2'(k_f) && (26)\\
S_0(k_f) &= -\frac{1}{P_1(k_f)}m(k_f) && (27)
\end{aligned}$$

The residual value of K_1 after the minimization is

$$K_1^* = [x' \; 1][R \; n][R \; n]'[x \; 1]' + o^2. \tag{28}$$

For further derivations, the matrix $R(k_f)$ is split into three parts:

$$R(k_f) = \begin{bmatrix} R_1 & R_2 \\ 0 & R_3 \end{bmatrix}, \tag{29}$$

where R_1 is a scalar, R_2 a transposed vector and R_3 an upper triangular matrix.

Standard step of optimization. Let us show one more step of the optimization; rewrite the cost function in the form:

$$\begin{aligned}
K &= K_1^* + Q_y e_y^2(k_f-1) + Q_u e_u^2(k_f-2)+ \\
&\quad + \textstyle\sum_{k=k_0}^{k_f-2} Q_y e_y^2(k) + Q_u e_u^2(k-1).
\end{aligned} \tag{30}$$

and denote the first three terms of K as K_2:

$$\begin{aligned}
K_2 &= [x'(k_f-1) \; 1]\begin{bmatrix} R' \\ n' \end{bmatrix}[R \; n]\begin{bmatrix} x(k_f-1) \\ 1 \end{bmatrix}+ \\
&\quad + Q_y e_y^2(k_f-1) + Q_u e_u^2(k_f-2) + o^2(k_f)
\end{aligned} \tag{31}$$

Using the state-space model (3), we can express the square root of the first term in this relation as a function of $u(k_f-2)$ and $x(k_f-2)$:

$$[R \; n]\begin{bmatrix} x(k_f-1) \\ 1 \end{bmatrix} = R(Bu(k_f-2)+Ax(k_f-2))+n.$$

Exploiting the structure of the matrices A and B (4) and using the partitioning of the matrix R (29), we get (omitting the time index (k_f-2) of u and x):

$$
\begin{gathered}
R(Bu+Ax)+n=\left[\begin{array}{cc} R_1 & R_2 \\ 0 & R_3 \end{array}\right]\left\{\left[\begin{array}{c} b_1 \\ 1 \\ 0_{(n-2,1)} \end{array}\right]u+\right. \\
\left.+\left[\begin{array}{ccc} \Theta_1 & \ldots & \Theta_n \\ 0 & \ldots & 0 \\ I_{(n-2,n-2)} & 0 & 0 \end{array}\right]x\right\}+n=\left[\begin{array}{cc} \tilde{v}_1 & \tilde{v}_2 \\ \tilde{Z}_1 & \tilde{Z}_2 \end{array}\right]\left[\begin{array}{c} u \\ x \\ 1 \end{array}\right],
\end{gathered}
$$

where the auxiliary variables $\tilde{v}$ and $\tilde{Z}$ are composed in the following way ($(R_2)_{1\ldots n}$ denotes the elements of R_2, $n_{1\ldots n}$ denotes the elements of n):

$$
\begin{aligned}
\tilde{v}_1 \;=\; & [R_1 b_1+(R_2)_1,\ R_1\Theta_1+(R_2)_2,\ \ldots \\
& \ldots,\ R_1\Theta_{n-2}+(R_2)_n,\ R_1\Theta_{n-1},\ R_1\Theta_n]
\end{aligned} \tag{32}
$$

$$
\tilde{v}_2=n_1,\ \ \tilde{Z}_1=\left[\begin{array}{ccc} R_3 & 0 & 0 \end{array}\right],\ \ \tilde{Z}_2=[n_2\ \ldots\ n_n]' \tag{33}
$$

Also, the square root of the second term in (31) (i.e. $Q_y e_y^2(k_f-1)$) can be expressed as a function of $u(k_f-2)$ and $x(k_f-2)$, similar to the first line of $\Phi(k_f)$ (19):

$$
\begin{aligned}
Q_y^{1/2}e_y \;&=\; Q_y^{1/2}[b_1 u(k_f-2)+\Theta' x(k_f-2)- \\
&\quad -w_y(k_f-1)] \\
&=\; Q_y^{1/2}[b_1\ \Theta']\left[\begin{array}{c} u(k_f-2) \\ x(k_f-2) \end{array}\right]+d_y(k_f-1) \\
d_y(k_f-1)&=-Q_y^{1/2}w_y(k_f-1)
\end{aligned} \tag{34}
$$

The square root of the third term of (31) reads exactly as in the second row of $\Phi(k_f)$ (19), except that the time is shifted:

$$
\begin{aligned}
Q_u^{1/2}e_u(k_f-2) \;&=\; Q_u^{1/2}[u(k_f-2)-w_u(k_f-2)] \\
&=\; Q_u^{1/2}u(k_f-2)+d_u(k_f-2), \\
d_u(k_f-2) \;&=\; -Q_u^{1/2}w_u(k_f-2).
\end{aligned} \tag{35}
$$

The sum of the terms in (31) we shall realize by orthogonal transformations of matrix, composed from their square roots. One transformation should be made to add (34) and $\tilde{v}$ (33), second transformation should be made to add (35) and $\tilde{Z}$ (33).

These transformations, however, may be prepared in advance, by filling the diagonal of matrix Z_1 (19) with the penalties and Z_2 with the setpoints, before the minimization step is performed [8]. The modified matrix $\bar{Z}_1(k_f)$ reads:

$$
\begin{aligned}
\bar{Z}_1(k_f) &= [\mathrm{diag}[Q_u^{1/2}\ Q_y^{1/2}\ \dots\ Q_u^{1/2}]\ 0_{(n+1,1)}]' && (36)\\
\bar{Z}_2(k_f) &= [d_u(k_f-1)\ d_y(k_f-1)\ \dots \\
&\quad \dots\ d_u(k_f-l-2)\ 0]' && (37)
\end{aligned}
$$

After the minimization (22), we get modified vector $\bar{n}(k_f)$ and matrix $\bar{R}(k_f)$:

$$
\begin{aligned}
\bar{R}(k_f) &= \begin{bmatrix} \bar{R}_1 & \bar{R}_2 \\ 0 & \bar{R}_3 \end{bmatrix}, && (38)\\
\bar{n}(k_f) &= n + [d_y(k_f-1)\ \dots\ d_u(k_f-l-2)]' && (39)
\end{aligned}
$$

for which holds

$$
[\bar{R}\ \bar{n}]'[\bar{R}\ \bar{n}] = [R\ \bar{n}]'[R\ \bar{n}] + \mathrm{diag}[Q_y Q_u \dots Q_y Q_u]. \tag{40}
$$

Using this matrix and vector for preparation of the next step of minimization, we may write:

$$
\begin{aligned}
\Phi(k_f-1) &= \begin{bmatrix} v_1(k_f-1) & v_2(k_f-1) \\ \bar{Z}_1(k_f-1) & \bar{Z}_2(k_f-1) \end{bmatrix}, && (41)\\
v_1(k_f-1) &= \bar{R}_1(k_f)[b_1\ \Theta] + [\bar{R}_2(k_f)\ 0\ 0] && (42)\\
v_2(k_f-1) &= \bar{n}_1(k_f) && (43)\\
\bar{Z}_1(k_f-1) &= \begin{bmatrix} \bar{R}_3(k_f) & & \\ & Q_y^{1/2} & \\ & & Q_u^{1/2} \\ 0 & \dots & 0 \end{bmatrix} && (44)\\
\bar{Z}_2(k_f-1) &= [\bar{n}_{2\dots n},\ d_y,\ d_u,\ o(k_f)]' && (45)
\end{aligned}
$$

Minimization and computation of the control law are performed in exactly the same way as in the first step.

By repeating the procedure described for K_2 until the beginning of the control horizon, we finally get the control law $S(k_0)$, $S_0(k_0)$, which is used to compute the control action u.

Remarks on algorithm

Constructing new matrix $\bar{Z}$, the matrix R_3 is simply shifted two positions up along the diagonal and completed with penalties. The vector $\bar{n}$ is shifted two positions up and completed with setpoints multiplied by penalties.

The scalar o (the last element in $\bar{Z}_2$ (45)) only passively contributes to the cost function and does not have any influence on the control law. Therefore, it is not required in the computation of the control law and the bottom row of matrix Φ, containing only this element, can be ignored.

Systolic array for the LQ control algorithm

The algorithm to compute the LQ control law can be mapped onto an $(n+1) \times (n+1)$ triangular systolic array.

Mapping of v_1 and $\bar{Z}_1$. To eliminate the unnecessary latency, both the vector v_1 (42) and the first row of matrix Z_1 (44) are mapped onto the same row of the systolic array (see fig 1).

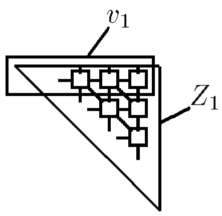

Fig 1. Mapping of v_1 and $\bar{Z}_1$

Mapping of v_2 and $\bar{Z}_2$. The setpoint processing column is mapped onto the diagonal processing elements (PE) of the array (see fig 2.).

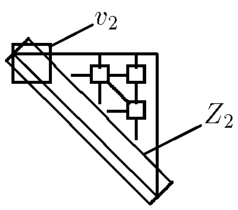

Fig 2. Mapping of v_2 and $\bar{Z}_2$

With this mapping, the elements of $\bar{n}$ are shifted up along the diagonal between the iterations in the same way as the elements of $\bar{R}_3$. Also, since d_y and d_u are calculated on the diagonal, we do not have to send $Q_u^{1/2}$ through the array.

The orthogonal transformations used in the minimization step, propagate through the array in waves from the upper left corner (see fig 3). From each PE the result of the transformation (updated matrix R) is immediately passed up along the main diagonal. As it must be shifted by two positions, the *PEs work* *in two modes* – in the *first mode* they only shift the matrix R and in the *second mode* they perform the minimization (see fig 3).

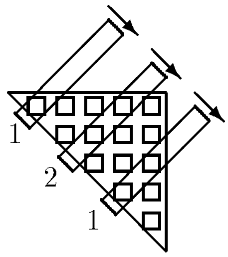

Fig 3. Propagation of transformations

The function of the PEs in the first row of the array is slightly more complicated. In the *first mode*, the PEs in this row multiply the plant parameters by $\bar{R}_1$, received from the next PE on the diagonal. In the *second mode*, the computation of v_1 is completed by adding $\bar{R}_2$. If the beginning of the control horizon has been reached then the control law is calculated and passed to the systolic array computing the control action.

Initialization of the array. The parameter estimates are entered at 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 starts to work (i.e. *there is no latency necessary for the initialization*).

The penalties and setpoints are entered along the diagonal. For the further iterations they are entered at the lower left corner of the array (see fig 4).

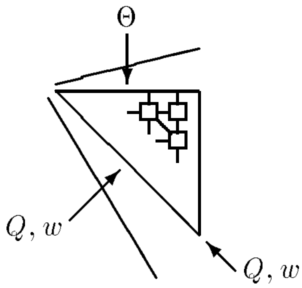

Fig 4. Entering of penalties and setpoints

Array for block regularised identification

The array implementing the algorithm (15) is shown in Fig. 5 as a part of the whole adaptive controller. The detailed cell description depends on the specification of the basic "rank-one" update at the cell level. See [3], [10] for different options.

The delay cells (labelled on Fig. 5. by **A**), form the vector $x(k-1)$ from $y(k)$ and $u(k-1)$. The FIFO buffer **B** is necessary to handle the regularising phase when the processing of the input data temporarily stops. The arrays marked in Fig. 5. by **C** and **D** store and update the matrix L_1' and the vector L_2 as in (15). The extended row of $n+1$ cells, **E**, performs the matrix by vector multiplication (15c). The matrix L_1^{*T} is loaded from the top into the **C** array and it is also used as input for the regularisation process (15d).

When loaded, L_1^{*T} starts to move (with the element $L_1^{*T}[n+1, n+1]$ first) through the **D** array to the right to realize (15c) in **E**. Simultaneously, vector Θ^* moves (the item $\Theta^*[n]$ first) to **E** from the bottom to realize (15f). The resulting product L_2^* emerges serially from the top output of the **E** array. Its elements $L_2^*[i]$ are directly loaded back into the top input of the array **D** (synchronously with the rows $L_1^{*T}[i]$ loaded **C**) to realize the accumulated regularisation (15d).

The vector Θ^* is necessary for the above process of regularisation. In Fig. 5, it can be defined by the user (hence the switch at the bottom entry of the **E** array), or it can be generated for each block automatically by the array based on the information stored in L_1', L_2 at the boundary of the previous block.

Therefore the arrays **C**, **D** provide, simultaneously with the standard QR update, the extraction of the parameter estimate Θ. This is known as "weight flushing". Both, (15b) and (15e) are implemented by inserting an $(n+1, n+1)$ unit matrix and a column of zeros into the second set of the array data inputs. The (15b) is computed simultaneously with (15d) and (15e) with (15a).

This second function of the array (the parameter extraction filter) works with constant values of the matrix L_1' and vector L_2 which were sampled and stored by the internal logic of the cells after (15a) and (15d) have been executed. The parameters emerge as a serial stream of $n+1$ items, (b_1 first) and those generated in (15e) are loaded into the upper row of the array for the LQ control algorithm (**H** in Fig. 5).

Array for optimal output generation

The computed control law parameters emerging from the top of the **H** array are latched into cells **G**, and passed onto the systolic generator for optimal control output denoted by **F** in Fig.5. It implements (25) for $S(k)$ and $S_0(k)$ by utilizing the shifting structure of $x(k-1)$ to generate $u(k-1)$ with minimal latency. The controller works with constant parameters until new updates are available in **G**.

Remarks on timing. We assume that all the processing elements (cells) of the arrays (denoted **C,D,E,H** in Fig. 5.) need an equal, constant time t for the elementary systolic operation. The LQ design array (**H** in Fig. 5.) needs time $T_C = 6N * t$ to compute the new control law parameters for the control horizon N. This time defines the basic period of the adjustment of the parameters for the array generating the control action (**F** in Fig. 5.). The identification section (**C,D,E** in Fig. 5.) needs time $T_I = (m+n+1)*t$ to complete one identification cycle (15). It is possible to specify the length of the data m (15), so that several identification cycles (15) fit precisely into the control law computation time T_C. Let us denote the count of these cycles by $j \geq 1$. The value of j comes from $T_C = j * T_I$ and $T_I = (m+n+1)*t$ with $m > n$.

The identification section samples only $j * m$ data pairs $y(k), u(k-1)$ in each control design period T_C. This defines the sampling period of the clock period k for array **F** as $k = T_C/(j*m) = t*(m+n+1)/m$. Notice that $2*t \geq k \geq t$ for all N, n, m, j.

The controller interacts with the controlled system with the clock period k. Therefore it is realistic to measure the period of the control law readjustment $T_C = (j*m)*k$ by the count of the clock periods k. To illustrate this point, let us consider a controller with $n+1 = 5$ parameters implemented on array given in Fig. 5. Parameters of the controller for different lengths of the control horizon N are summarized in a table below:

Length of control horizon (N)	5	8	12
Time for control comput. $t*(6N)$	$30t$	$48t$	$72t$
Number of identification cycles (j)	3	4	4
Number of data in ident. blocks (m)	5	7	13
Data sampl. time $k = t*(m+5)/m$	$2t$	$1.7t$	$1.4t$
Period of cont. law adjust. (in k cl.)	$15k$	$28k$	$52k$
Latency of cont. law adjust. (in k)	$30k$	$56k$	$104k$
Latency of control action generation	$1t$	$1t$	$1t$

CONCLUSIONS

In this paper we have presented an LQ adaptive controller which has a data sample and control action generation interval of $O(1)$, and the readjustment interval of control law of $N * O(1)$, where N is the length of the horizon used for the LQ control computation, by using a pipelined parallel implementation. This controller has a number of important properties:

- The regularised identification procedure is robust because it prevents covariance of the estimates from becoming reduced-rank and causing the estimator to "blow up" and also limits the estimates from varying beyond a "safe" region.
- The identification is implemented on $O(n^2)$ processing cells and produces an estimate every $O(n)$ time intervals.
- The computation of the LQ control law also uses $O(n^2)$ processing cells and has a speed of $N * O(1)$, which is independent of the system order. This is an order of magnitude faster than that proposed in [11] and has the added advantage of no feedback loops on the systolic array.
- The computation of the control law allows for the setpoints to be non-zero and, also time-varying, which is important for practical applications. The penalties can also be time-varying.
- Lastly, the computation required to implement the control law is very simple and has a speed of $O(1)$ on $O(n)$ processing elements.

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

References

[1] Peterka, V., 1984, Predictor-based self-tuning control, Automatica, 20, 5, 39-50

[2] Clarke D. W., Mohtadi C. and Tuffs P.C., 1987, Generalized predictive control- Part I. The basic algorithm, Automatica, 23, 2, 137-148

[3] Mc Whirter, J.G., 1983, Recursive Least Squares minimization using a Systolic Array, Proc. SPIE, Real Time Signal Processing VI, 431, 105

[4] Ljung, L., Söderström, T., 1983, "Theory and Practice of Recursive Identification", Cambridge, MA, MIT Press

[5] Kulhavý, R., 1987, Restricted exponential forgetting in real-time identification, Automatica, 23, 589-600

[6] Böhm J., Kulhavy, R., 1990, Towards more credible self-tuners... "Lecture Notes in Control and Information Sciences", K.Warwick, M.Kárný, A.Halousková editors, 158, Springer-Verlag, 189-205

[7] Kárný, M., Halousková, A., Böhm, J., Kulhavý, R. and Nedoma, P., 1985, Design of linear quadratic adaptive control: theory and algorithms for practice, Kybernetika, 21, Supplement to No. 3, 4, 5, 6

[8] Chisci,L., Zappa G., 1991, A Systolic Architecture for Iterative LQ Optimization Automatica, 27 No. 5. 799-810.

[9] Kadlec, J., Gaston, F.M.F., Irwin, G. W., 1992, Parallel Implementation of Restricted Parameter Tracking, To be published in "Mathematics in Signal Processing", J. Mc Whirter editor, Oxford publishers press.

[10] Kadlec,J. 1993, The Cell Level Description of Systolic Block Regularised QR Filter, Proc. IEEE Workshop on VLSI Signal Processing, Koningshof, Veldhoven, Netherlands, 298-306

[11] Gaston, F.M.F; Irwin, G.W., 1990, Linear quadratic optimal controller realization on systolic arrays. Proc. UK Information Technology Conf., Southampton, 407-412.

[12] Schier, J., 1993, Parallel algorithms for robust adaptive identification and square-root LQG control synthesis, Ph.D. Thesis, ÚTIA AVČR, Prague, Czech Republic

[13] Shepherd,T.J., Mc Whirter, J. G., and Hudson, J. E., 1988, Parallel Weight Extraction from a Systolic Adaptive Beamformer. Proc. IMA Int. Conf. on Mathematics in Signal Processing, Warwick, Dec. 1988, 775-790.

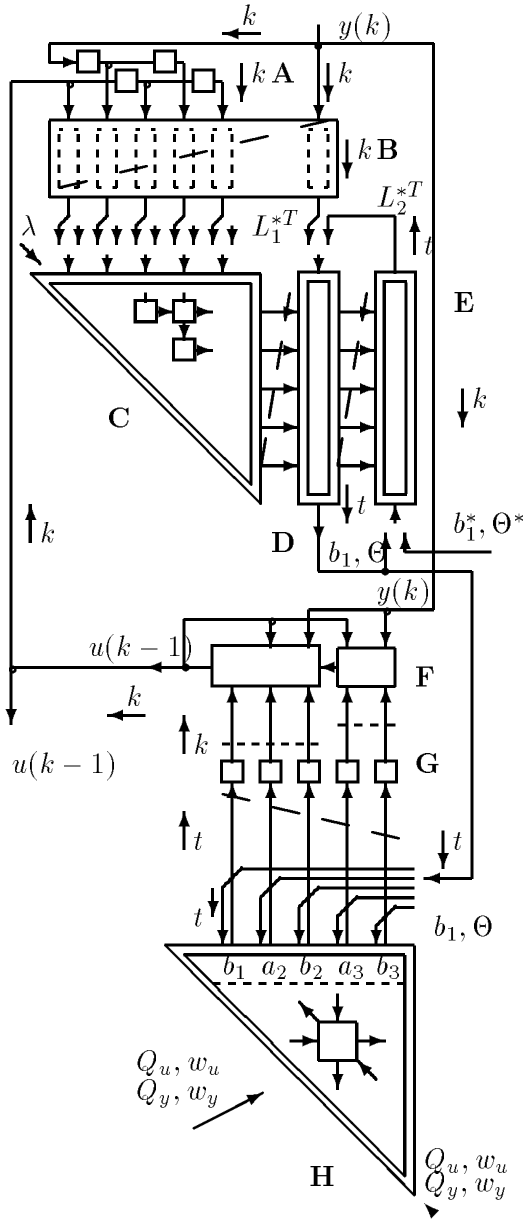

Fig 5. Systolic Controller.

Notation: **A** - delay cells; **B** - FIFO buffer; **C, D, E** - The block regularised identification array; **F** - systolic optimal control output generator; **G** - latch cells; **H** - LQ design array (see fig 1-4 for its details).