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.Kadlec

The Cell Level Description of Systolic Block Regularised QR Filter

No. 1792 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

The Cell Level Description of Systolic Block Regularised QR Filter

J. Kadlec

This paper presents the concept and the cell-level description of the block regularised parameter estimator compatible with the requirements of implementation on a parallel architecture.

1. INTRODUCTION

Adaptive recursive parameter estimation is a fundamental concept in control and signal processing. The standard regularisation of identification protects the square root adaptive recursive estimation from numerical difficulties (associated with not sufficiently exciting data). But an ad-hoc regularisation can affect the quality of estimation unpredictably.

This was one of the reasons why Kulhavý has proposed a generalised parameter tracking in [1]. In his concept the standard exponential weighting method is extended by the addition of an alternative weighted quadratic term, enriching the conventional exponentially weighted RLS method by an general concept of regularisation.

In [2] a novel concept of a fast, block regularised parameter estimator has been presented compatible with the requirements of implementation on parallel architectures.

This paper presents the detailed, cell-level description of a compacted systolic array implementation of the block regularised parameter estimator [2]. The regularisation avoids the potential numerical problems, caused by overparametrisation or poor data, which can eventually affect the classical QR recursive least squares systolic arrays [3], [4] without regularisation. The array provides periodically the regression parameter estimates (weights). It is shown in detail, how to implement the QR type of the processing cells.

In the presented array an additional column of processors realize a feedback associated with the periodical adjustment of the new position of the regularising cost function on the boundaries of blocks. This position follows (with delay) the current parameter estimates. The array is compatible with the pipelined implementation if the length of blocks is chosen at least equal to the number of identified parameters.

The Basic Model

The identified system is modelled by the linear regression $y_k = \theta_k^T \varphi_k + v_k$ where the output scalar measurement, y_k is assumed to be related through the unknown parameter vector θ_k to a known n-dimensional vector φ_k composed from the recent inputs, outputs, measurable disturbances contributing to the model output and v_k is scalar Gausian white noise with zero mean and a known constant covariance. A cost function is minimized so that

$$\hat{\theta}_k = \arg\left\{\min_{\theta_k} J_k(\theta_k)\right\} \tag{1}$$

where $\hat{\theta}_k$ is the estimate of the unknown parameter vector θ_k.

Regularised Exponential Algorithm

In the regularised exponential algorithm [1], [2] the recursively minimised cost function is modified, as shown in equation (2)

$$J_k(\theta_k) = \lambda_{k-1} J_{k-1}(\theta_k) + (1-\lambda_{k-1}) J^*_{k-1}(\theta_k) + (y_k - \theta_k^T \varphi_k)^2 \tag{2}$$

where the scalar $0 < \lambda_{k-1} \leq 1$ is the exponential forgetting factor.

The cost function $J_k(\theta_k)$ in (2) differs from the exponentially weighted RLS quadratic cost function by the addition of the term $(1-\lambda_{k-1})J^*_{k-1}(\theta_k)$. By introducing this additional quadratic term the algorithm is enriched by the description of the time update step and by the tools for regularising the recursive identification.

If the quadratic costs J_k (resp. J^*_{k-1}) are characterised by (n) vector and (n,n) matrix d_k, P_k^{-1} (resp. d^*_{k-1}, $(P^*_{k-1})^{-1}$) then the regression parameter estimates $\hat{\theta}_k$, minimizing (2), is given by $\hat{\theta}_k = P_k d_k$ and a similar vector θ^*_{k-1}, minimizing J^*_{k-1}, is given by $\theta^*_{k-1} = P^*_{k-1} d^*_{k-1}$.

The criterium (1) for the cost function (2) leads to time and data update based on the inverse error covariance as given in equations (3).

$$\begin{aligned} P_k^{-1} &= \lambda_{k-1} P_{k-1}^{-1} + (1-\lambda_{k-1})(P^*_{k-1})^{-1} + \varphi_k \varphi_k^T \\ d_k &= \lambda_{k-1} d_{k-1} + (1-\lambda_{k-1})(P^*_{k-1})^{-1}\theta^*_{k-1} + \varphi_k y_k \\ \hat{\theta}_k &= P_k d_k \end{aligned} \tag{3}$$

This defines the regularised exponential algorithm. For the detailed derivation see [2].

The Complexity

The price paid for the universality of the regularised exponential algorithm (3) is the increase of the number of arithmetic operations in comparison with the exponentially weighted recursive least squares algorithm.

Between two successive data measurements the regularised exponential algorithm approach needs (in the factorised information-filter implementation) $O(n^3)$ multiplications and divisions between data measurements (the inclusion of alternative covariance matrix $(P^*_{k-1})^{-1}$). In contrast, the exponentially weighted recursive least square information filter [3] needs only simple rank-one update leading to $O(n^2)$ complexity.

Such an increase of the computation burden, caused by regularisation, might be difficult to accept in many time intensive signal processing applications. Therefore a novel block regularised algorithm has been described in [2].

2. BLOCK REGULARISED ALGORITHM

The Block Regularised Algorithm [2] overcomes complexity problem by the additional assumption about the covariance P^*_{k-1} and the vector θ^*_{k-1} used for the regularisation in (3). If they both remain constant within an block of N data samples

$$P^* = P^*_{k-N} = P^*_{k-N+1} = \ldots = P^*_{k-1} \tag{4}$$

$$\theta^* = \theta^*_{k-N} = \theta^*_{k-N+1} = \ldots = \theta^*_{k-1} \tag{5}$$

then the alternative covariance matrix can be added only every N-th iteration, certainly with a new accumulated weight.

Under the assumption (4)-(5) the block regularised algorithm provides exactly the same parameter estimates as the original algorithm (3) at every N-th iteration. But the complexity is reduced from $O(n^3)$ to $O(n^2)$.

The block regularised algorithm has to be "square-rooted" in order to get sound results with the finite precision arithmetic. The standard factorisation [3] of matrices P_k^{-1} and $(P^*)^{-1}$ is used:

$$P_k^{-1} \hat{=} R_k^T\, R_k \; ; \; (P^*)^{-1} \hat{=} R^{*T}\, R^* \tag{6}$$

Here R_k ; R^* are upper triangular (n, n) matrices with nonnegative items on the diagonal.

From these factorisations the following relation can be defined

$$r_k \hat{=} R_k\, \hat{\theta}_k = R_k^{-T} d_k \; ; \; r^* \hat{=} R^*\, \theta^* = (R^*)^{-T} d^* \tag{7}$$

where r_k and r^* are the transformed parameters used in the array implementation. It is well known that the process can be implemented in parallel on the triangular Gentleman-Kung array [3], [4] updating in cels the items of the triangular matrix R_k and in the separate column the vector r_k. Data $[\varphi_k^T\,,\, y_k]$ is to be loaded into the array in parallel from the top of the array.

Using this QR rotation concept [3], the factorised block restricted algorithm can be rewritten into the final QR form (8). See [2] for the detailed derivation.

The QR block restricted algorithm (8) performs a block of N (reorganised) steps of the regularised algorithm (3) under the assumption (4), (5).

Block Regularised Algorithm as a sequence of the QR orthogonal rotations. It is specified by the parameters $R^* \geq 0; \lambda_{k-N}, .., \lambda_{k-1}; 0 < \lambda_j \leq 1; N \geq n$.

$$\bar{R} = R_{k-N}\,;\; \bar{r} = r_{k-N} \qquad (a)$$

cycle for $j = N-1, N-2, ..., 0:$

$$\begin{bmatrix} \bar{R} & \bar{r} \\ 0 & * \end{bmatrix} := Q_{k-j-1} \begin{bmatrix} \sqrt{\lambda_{k-j-1}}\,\bar{R} & \sqrt{\lambda_{k-j-1}}\,\bar{r} \\ \varphi_{k-j}^T & y_{k-j} \end{bmatrix} \qquad (b)$$

end of cycle for j

$$\bar{R}_k = \bar{R}\,;\; \bar{r}_k = \bar{r} \qquad (c)$$

$$\bar{\theta}_{k-N} = \bar{R}_{k-N}^{-1}\bar{r}_{k-N} \qquad (d) \qquad (8)$$

$$\theta^* \equiv \bar{\theta}_{k-N} \qquad (e)$$

$$r^* = R^*\theta^* \qquad (f)$$

cycle for $i = n, n-1, ..., 1:$

$$\begin{bmatrix} \bar{R} & \bar{r} \\ 0 & * \end{bmatrix} := Q_{i,k} \begin{bmatrix} \bar{R} & \bar{r} \\ \sqrt{\lambda_{[k-N,k]}}\,R^*[i] & \sqrt{\lambda_{[k-N,k]}}\,r^*[i] \end{bmatrix} \qquad (g)$$

end of cycle for i

$$R_k = \bar{R}\,;\; r_k = \bar{r}\,;\; \hat{\theta}_k = R_k^{-1} r_k \qquad (h)$$

where $0 < \lambda_{[k-N,k]}$ is a scalar accumulated weight given by (9)

$$\begin{aligned} \lambda_{[k-N,k]} &= \lambda_{k-1}...\lambda_{k-N+1}(1-\lambda_{k-N}) + \lambda_{k-1}...\lambda_{k-N+2}(1-\lambda_{k-N+1}) \\ &+ \; ... + \lambda_{k-1}(1-\lambda_{k-2}) + (1-\lambda_{k-1}). \end{aligned} \qquad (9)$$

Notice the automatic selection of θ^*. It requires a sequence of additional computations (8d)-(8f). The equation (8d) computes the parameter estimate $\bar{\theta}_{k-N}$. In (8e) the selection of θ^* is made. Finally the vector r^* as a matrix by vector product $R^*\theta^*$ is computed (8f). The vector r^*, weighted by $\lambda_{[k-N,k]}$ (9), is added to $\bar{r}_k$ in (8g) to realise the accumulated regularisation.

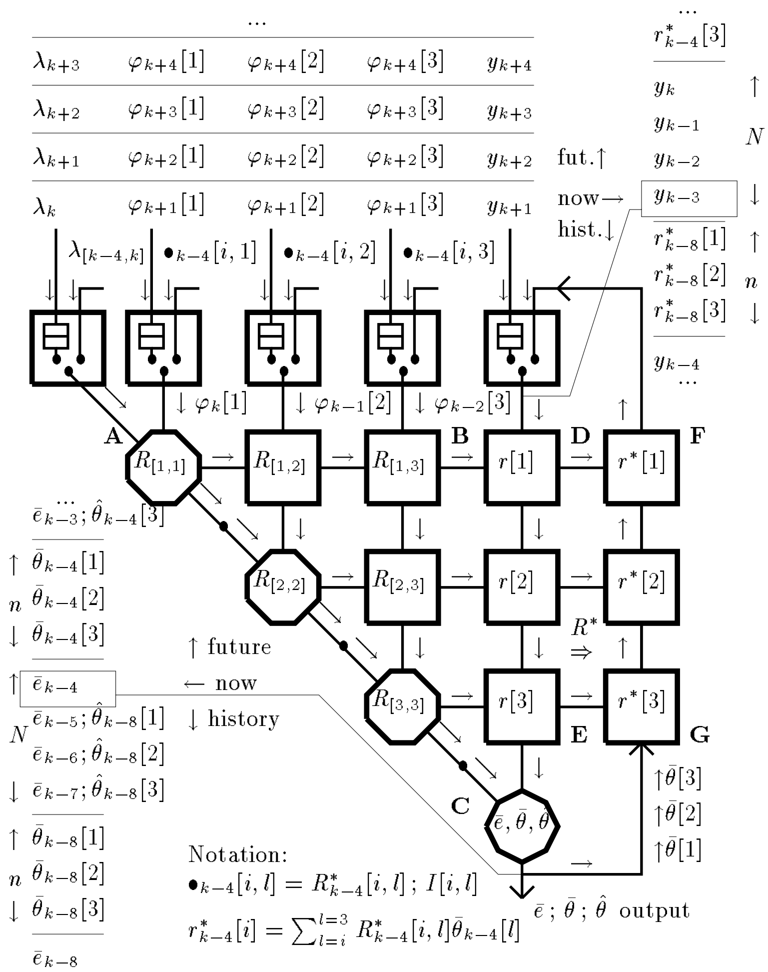
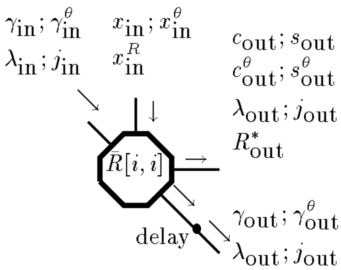


Figure 1. Compacted Block Regularised QR Array

3. SYSTOLIC ARRAY DESCRIPTION

The systolic array implementing the algorithm (8) is plotted in Fig. 1. The basic areas marked by ABC and BCDE store and update matrix R_k and r_k as in (8b) and (8g). The extended column DEFG performs the feedback associated with the block regularisation. The first-in-first-out buffers accumulate and synchronise the input data measurements. Multiplexers switch between the data from the buffer outputs and the regularisation data R^* and r^*.

The Diagonal Cell $\bar{R}[i,i]$



Timing from cell $\bar{R}[1,1]$

$j_{\text{in}} = N-1, ..., 1, 0, -1, ..., -n$

for all j_{in} : $\gamma_{\text{in}} = 1$; $\gamma^{\theta}_{\text{in}} = 1$.

if $(N > j_{\text{in}} \geq N-n)$: $\lambda_{\text{in}} = \lambda_{k-j-1}$;
$x_{\text{in}} = \varphi_{k-j}[1]$; $x^{\theta}_{\text{in}} = I_{[N-j,1]}$.

if $(N-n > j_{\text{in}} \geq 0)$: $\lambda_{\text{in}} = \lambda_{k-j-1}$;
$x_{\text{in}} = \varphi_{k-j}[1]$.

if $(1 \geq j_{\text{in}} \geq -n)$: $\lambda_{\text{in}} = \lambda_{[k-N,k]}$;
$x_{\text{in}} = R^{*}_{[n+j+1,1]}$; $x^{\theta}_{\text{in}} = I_{[n+j+1,1]}$;
$x^{R}_{\text{in}} = R^{*}_{[n+j+1,1]}$.

The Computaion in Diagonal Cells $\bar{R}[i,i]\; i = 1, 2, .., n$.

for all j_{in} : $\lambda_{\text{out}} = \lambda_{\text{in}}$; $j_{\text{out}} = j_{\text{in}}$.

if $(N > j_{\text{in}} \geq 0)$:
$\bar{R}' = (\lambda_{\text{in}}\bar{R}^2 + x^2_{\text{in}})^{1/2}$;
$c_{\text{out}} = \lambda^{1/2}_{\text{in}}\bar{R}/\bar{R}'$;
$s_{\text{out}} = x_{\text{in}}/\bar{R}'$;
$\gamma_{\text{out}} = c_{\text{out}}\gamma_{\text{in}}$; $\bar{R} = \bar{R}'$.

if $(N > j_{\text{in}} \geq N-n))$:
$\bar{R}'_k = (\bar{R}^2_k + (x^{\theta}_{\text{in}})^2)^{1/2}$;
$c^{\theta}_{\text{out}} = \bar{R}_k/\bar{R}'_k$;
$s^{\theta}_{\text{out}} = x^{\theta}_{\text{in}}/\bar{R}'_k$;
$\gamma^{\theta}_{\text{out}} = c^{\theta}_{\text{out}}\gamma^{\theta}_{\text{in}}$.

if $(j_{\text{in}} = 0)$: $\bar{R}_k = \bar{R}'$.

if $(1 \geq j_{\text{in}} \geq -n)$:
$\bar{R}' = (\bar{R}^2 + \lambda_{\text{in}}x^2_{\text{in}})^{1/2}$;
$c_{\text{out}} = \bar{R}/\bar{R}'$;
$s_{\text{out}} = \lambda^{1/2}_{\text{in}}x_{\text{in}}/\bar{R}'$;
$\gamma_{\text{out}} = c_{\text{out}}\gamma_{\text{in}}$; $\bar{R} = \bar{R}'$.

if $(0 \geq j_{\text{in}} \geq -n)$:
$R'_k = (R^2_k + (x^{\theta}_{\text{in}})^2)^{1/2}$;
$c^{\theta}_{\text{out}} = R_k/R'_k$;
$s^{\theta}_{\text{out}} = x^{\theta}_{\text{in}}/R'_k$;
$\gamma^{\theta}_{\text{out}} = c^{\theta}_{\text{out}}\gamma^{\theta}_{\text{in}}$.

if $(j_{\text{in}} = -n-1+i)$: $R^{*} = x_{\text{in}}$.

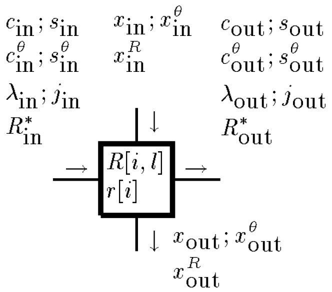
if $(j_{\text{in}} = N-1-n+i)$: $R^{*}_{\text{out}} = R^{*}$.

if $(j_{\text{in}} = -n)$: $R_k = \bar{R}'$.

Figure 2. Diagonal Cells of the Block Regularised QR Array

Arrays ABC and BCDE (Fig. 1) provide the parameter estimate $\bar{\theta}_{k-N}$ and $\hat{\theta}_{k-N}$ by "weight flushing" [4]. An unit matrix and a column of zeros is inserted on the array data inputs to compute (8d) and (8h). These "filters" work with the "frozen" values of $\bar{R}_{k-N}$, $\bar{r}_{k-N}$ and $\hat{R}_{k-N}$, $\hat{r}_{k-N}$. Estimates $\bar{\theta}_{k-N}$ are used for the automatic positioning of the regularising quadratic form ($\theta^{*} = \bar{\theta}_{k-N}$). The vector θ^{*} moves (the item $\theta^{*}[n]$ first) back to the block DEFG from the south to realize (8f).

The Cells $R[i,l]\,;\,r[i]$



Inputs of cells $R[1,l]$; $l = 2,3..n$

if $(N > j_{\text{in}} \geq 0)$:
$x_{\text{in}} = \varphi_{k-j}[l]\,;\, x^{\theta}_{\text{in}} = I_{[N-j,l]}$.
if $(0 > j_{\text{in}} \geq -n)$: $x_{\text{in}} = R^*_{[n+j+1,l]}$;
$x^{R}_{\text{in}} = R^*_{[n+j+1,l]}\,;\, x^{\theta}_{\text{in}} = I_{[n+j+1,l]}$.

Input of cell $r[1]$

if $(N > j_{\text{in}} \geq 0)$: $x_{\text{in}} = y_{k-j}; x^{\theta}_{\text{in}} = 0$.
if $(0 > j_{\text{in}} \geq -n)$: $x_{\text{in}} = r^*_{[n+j+1]}$;
$x^{\theta}_{\text{in}} = 0$.

The Internal Cells
For Cells $r[i]$; $i = 1..n$ replace R by r.

for all j_{in} : $\lambda_{\text{out}} = \lambda_{\text{in}}\,;\, j_{\text{out}} = j_{\text{in}}\,;\, c_{\text{out}} = c_{\text{in}}\,;\, s_{\text{out}} = s_{\text{in}}$.

if $(N > j_{\text{in}} \geq 0)$:
$$x_{\text{out}} = c_{\text{in}} x_{\text{in}} - s_{\text{in}} \bar{R};$$
$$\bar{R} = s_{\text{in}} x_{\text{in}} + c_{\text{in}} \bar{R}.$$

if $(N > j_{\text{in}} \geq N - n)$:
$$x^{\theta}_{\text{out}} = c^{\theta}_{\text{in}} x^{\theta}_{\text{in}} - s^{\theta}_{\text{in}} R_k;$$
$$s^{\theta}_{\text{out}} = s^{\theta}_{\text{in}}\,;\, c^{\theta}_{\text{out}} = c^{\theta}_{\text{in}}.$$

if $(j_{\text{in}} = 0)$: $\bar{R}_k = \bar{R}$.

if $(1 \geq j_{\text{in}} \geq -n)$:
$$x_{\text{out}} = c_{\text{in}} x_{\text{in}} - s_{\text{in}} \bar{R};$$
$$\bar{R} = s_{\text{in}} x_{\text{in}} + c_{\text{in}} \bar{R}.$$

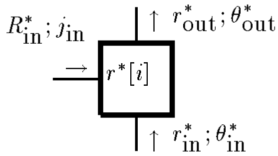
if $(0 \geq j_{\text{in}} \geq -n)$:
$$x^{\theta}_{\text{out}} = c^{\theta}_{\text{in}} x^{\theta}_{\text{in}} - s^{\theta}_{\text{in}} \bar{R}_k;$$
$$s^{\theta}_{\text{out}} = s^{\theta}_{\text{in}}\,;\, c^{\theta}_{\text{out}} = c^{\theta}_{\text{in}}.$$

if $(-1 > j_{\text{in}} > -n - 1 + i)$: $x^{R}_{\text{out}} = x^{R}_{\text{in}}$.
if $(j_{\text{in}} = -n - 1 + i)$: $R^* = x^{R}_{\text{in}}$.
if $(-n - 1 + 2i + n - 1 > j_{\text{in}} \geq -n - 1 + 2i)$: $R^*_{\text{out}} = R^*\,;\, R^* = R^*_{\text{in}}$.
if $(j_{\text{in}} = -n)$: $R_k = \bar{R}$.

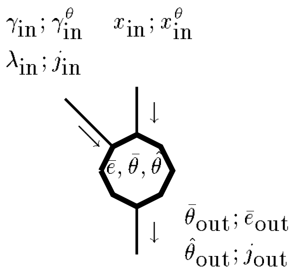
Figure 3. Internal Cell of the Block Regularised QR Array

The extended row of n cells, DEFG (Fig.1.), performs the matrix by vector multiplication (8f). The matrix R^* is loaded from the north to ABC area. When loaded, R^* starts to move (with the element $R^*[n,n]$ first) through BCDE in the easterly direction, to realize (8f) in DEFG. The resulting product $r^*[i]$ emerges from the northerly output of the DEFG array. It is directly loaded back to the north input of the array BCDE to realize (8g).

The Feedback Cells $r^*[i]$



The Final Cell



Feedback Cell

if $((-n-1+2i)+(n-i) >$
$\quad j_{\text{in}} \geq -n-1+2i)$:
$\qquad r^* = r^* + R^*_{\text{in}}\theta^*_{\text{in}}.$

if $(j_{\text{in}} = -n-1+2i)$:
$\qquad r^*_{\text{out}} = r^*\ ;\ r^* = 0.$

if $((-n-1+2i)+(n-i) \geq$
$\quad j_{\text{in}} > -n-1+2i)$:
$\qquad r^*_{\text{out}} = r^*_{\text{in}}.$

if $((-n-1+2i)+i \geq j_{\text{in}}$
$\quad > -n-1+2i)$:
$\qquad \theta^*_{\text{out}} = \theta^*_{\text{in}}.$

Final Cell

if $(N > j_{\text{in}} \geq 0)$:
$\qquad \bar{e} = x_{\text{in}}\gamma_{\text{in}}.$

if $(N > j_{\text{in}} \geq N-n)$:
$\qquad \hat{\theta}_{\text{out}} = x^{\theta}_{\text{in}}/\gamma^{\theta}_{\text{in}}.$

if $(1 \geq j_{\text{in}} \geq -n)$:
$\qquad \bar{\theta}_{\text{out}} = x^{\theta}_{\text{in}}/\gamma^{\theta}_{\text{in}}.$

Figure 4. Feedback and Final Cells of Block Reg. QR Array

Notice, that cells (Fig. 2.-Fig. 4.) work in two modes distinguished by the sign of an index j propagated through the array. The value of the index gives to the cells the timing information within both modes.

4. CONCLUSION

In this paper the block regularised algorithm is described in its square root form. The algorithm allows parallel, systolic array implementation.

It is shown up to the detailed cell level how it could be implemented on a systolic architecture. Fig.1. presents the compacted array structure, with the input data buffering and multiplexers, and Fig.2. - Fig.4. give the detailed internal cell description.

The architecture is only half as fast as the standard architecture of McWhirter [3] but it provides a novel and relatively universal concept of regularisation.

The cells perform nearest-neighbor, one-directional communication. There are no long global connections or data flows. The feedback associated with adjustment of the cost function used for regularisation is completely compatible with the pipelined implementation.

References

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

[2] Kadlec, J., F.M.F. Gaston, G. W. Irwin, (1992). Parallel Implementation of Restricted Parameter Tracking. Poster na *Third IMA International conference on Mathematics in Signal Processing*, by University of Warwick, UK, December 15-17, 1992.

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

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

4.1. Acknowledgments

The author gratefully acknowledge the support from The Royal Society UK–Czechoslovak Fellowship Program.

Ji Kadlec
Instute of Information Theory and Automation
Czech Academy of Sciences
P.O. Box 18, 182 08 Prague 8, Czech Republic,
e-mail: adapt@cspgas11.bitnet