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

Systolic Arrays for Identification of Systems with Variable Structure

No. 1817 November 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.

This work has been partially supported by the research grant No. 102/93/0897 of the Grant Agency of the Czech Republic: "Systolic Arrays of Processors for Identification of Systems with Variable Structure" and by the SERC Grant "Algorithm Engineering for Concurrent Control" (UK). J. Kadlec is with Control Engineering Research Centre, Department of Electrical and Electronic Engineering, The Queen's University of Belfast, Ashby Building, Stranmillis Road, Belfast BT9 5AH, Northern Ireland, Email: j.kadlec@ee.qub.ac.uk

Systolic Arrays for Identification of Systems with Variable Structure

Jiří Kadlec

November 4, 1994

Abstract

The paper explains the nesting of lower order problems, available in the square root information filters. It is shown how to find the nested estimates in the established QR and QRD information filters based on Givens rotations. Nesting enables the transformation of an aposteriori Givens algorithm to the normalised version of the aposteriori Givens algorithm. All normalised parameters and variables propagated in the corresponding systolic array have a guaranteed range of values. This provides an opportunity for low-cost, fixed-point systolic implementation. The performance of the fixed point array is assessed using computer simulations.

1 Introduction

Parallel array predictors, estimators and filters based on the recursive minimisation of an exponentially weighted least-squares criterion (RLS) have been studied extensively in the 80's. The wide use of these structures in various signal processing and control applications is mainly due to their good numerical properties [11] , and the possibility of pipelined, VLSI implementation.

This paper explains the nesting of lower order problems, present in square root information filters. The concept of nesting gives the solution to problem of effective systolic implementation of an identification of regression models with an unknown or time variable structure.

The understanding to the nested RLS subproblems can eliminate the parallel running of independent identification tasks and therefore considerably reduce the numeric complexity of the problem.

The concept of nesting is used in the derivation of a normalised version of the aposteriori Givens algorithm.This normalised recursive information filter allows a highly modular and numerically safe implementation of the recursive least square identification with a minimal latency. All normalised parameters and internal variables of the modular triangular array have the guaranteed range $[-1, 1]$.

Consequently, a significant part of the algorithm is suitable for implementation on a fast, parallel, low-cost, fixed-point hardware, based on look-up memory tables.

1.1 Existing Systolic Implementations

Algorithms, known as square root information filters, transform the estimated parameters by a data and time varying linear transformation. The data-regressor is transformed analogically to a vector of backward errors. The output prediction of the information filter is built up as a linear combination of transformed parameters and backward errors.

The recursive QR or QRD algorithm [3],[10], based on the orthogonal Givens rotations and the Recursive Gram–Schmidt (RGS) algorithm [7] belong to the group of square root information filters.

The second group of algorithms originates from UD factorised covariance filters [1] and provides the estimated parameters directly. Moonen and McWhirter [12] have shown the systolic array implementation of these algorithms.

In this paper, we concentrate on the first group of algorithms, due to the fact that they produce a range of nested lower order RLS estimates.

2 Nested RLS Estimation Problems

We consider the recursive least squares (RLS) identification of a single-output system described by the regression model

$$y(t) = \theta_p^T \varphi_p(t) + v(t) \tag{1}$$

where the p-vector, $\varphi(t)$, is the data regressor, $y(t)$ is the output signal and $v(t)$ represents the stochastic part of the transformation. The unknown p-vector, θ_p, of the regression parameters is estimated by $\hat{\theta}_p(t)$.

Let us denote $\varphi_{p+1}(t)$, the vector of data measurements as $\varphi_{p+1}(t) = [\varphi_p^T(t), y(t)]^T$. The scalar $\varphi_{p+1,m}(t)$ is the m-th element of $\varphi_{p+1}(t)$ with $m = 2, ..., p+1$.

We will operate with $p(p+1)/2$ different regression models (indexed by $m = 2, ...p+1; i = 1, 2, ...m-1$):

$$\varphi_{p+1,m}(t) = \theta_i^{(m)T} \varphi_i(t) + v_{m,i}(t) \tag{2}$$

where the i-vector $\varphi_i(t)$ is formed by the first i data items from the vector $\varphi_{p+1}(t)$. The scalars $v_{m,i}(t)$ represent the stochastic part of transformations. The unknown vectors of parameters $\theta_i^{(m)}; m = 2, ...p+1; i = 1, 2, ...m-1$ are to be recursively identified with estimates denoted by $\hat{\theta}_i^{(m)}(t); m = 2, ...p+1; i = 1, 2, ...m-1$.

Notice that (m) represents the position of the estimated data in the regressor and the index i defines the length of the data regressor used and the number of regression parameters. The standard, maximal order RLS model (1) is part of the set (2) for $m = p+1; i = p$.

The estimates $\hat{\theta}_i^{(m)}(t); m = 2, ...p+1; i = 1, .., m-1$ minimise the sum of exponentially weighted squares of the prediction errors:

$$J(\theta_i^{(m)}) = \sum_{j=1}^{j=t} \lambda^{t-j} \left(\varphi_{p+1,m}(j) - \theta_i^{(m)T} \varphi_i(j) \right)^2 \tag{3}$$

where $0 < \lambda \leq 1$ denotes the exponential weighting factor.

The sum (3) can be rearranged as:

$$J(\theta_i^{(m)}) = \left[\frac{-\theta_i^{(m)}}{1} \right]^T V_i^{(m)}(t) \left[\frac{-\theta_i^{(m)}}{1} \right] \tag{4}$$

where

$$V_i^{(m)}(t) = \sum_{j=1}^{j=t} \lambda^{t-j} \left[\frac{\varphi_i(j)}{\varphi_{p+1,m}(j)} \right] \left[\frac{\varphi_i(j)}{\varphi_{p+1,m}(j)} \right]^T . \tag{5}$$

The matrix $V_i^{(m)}(t)$ is composed from (i, i) matrix $V_i(t)$, vectors $[V_{1,m}(t), ...V_{i,m}(t)]^T$ and scalar $V_{m,m}(t)$.

The information matrix $V_i^{(m)}$ is a weighted sum of data measurement vectors $[\varphi_i^T(j), \varphi_{p+1,m}(j)]^T$.

Let us compare the time recursion of $V_i^{(m)}(t)$, defined in (5) with the time recursion of the maximal-dimension matrix $V_{p+1}(t)$:

$$V_{p+1}(t) = \sum_{j=1}^{j=t} \lambda^{t-j} \varphi_{p+1}(j) \varphi_{p+1}^T(j). \tag{6}$$

If the weight λ is identical for all compared recursions and if the initial value of the matrix $V_i^{(m)}(0)$ is identical with the matrices $V_i(0)$, $[V_{1,m}(0), ...V_{i,m}(0)]^T$ and $V_{m,m}(0)$ nested in the initial values of the matrix $V_{p+1}(0)$ then all elements of $V_i^{(m)}(t)$ are present (nested) in the maximal matrix $V_{p+1}(t)$ for $m = 2, 3...p+1; i = 1, 2, ..., m-1$ and all $t > 0$.

It is direct consequence of the definiton of $[\varphi_i^T(j), \varphi_{p+1,m}(j)]^T$ as a vector composed from data present in $\varphi_{p+1}(j)$.

Result: The matrices $V_i^{(m)}(t)$;$m = 2, 3...p+1; i = 1, 2, ..., m-1$ can be described by the set of elements and submatrices of the maximal order matrix $V_{p+1}(t)$:

$$V_i^{(m)}(t) = \left[\begin{array}{c|c} V_i(t) & V_{1,m}(t) \\ & \cdots \\ & V_{i,m}(t) \\ \hline V_{1,m}(t), ..., V_{i,m}(t) & V_{m,m}(t) \end{array} \right] \tag{7}$$

2.1 RLS Parameter Estimates

The minimum of the quadratic form (4) can be found from the partitioning of $V_i^{(m)}(t)$ outlined in (7):

Result: The vector

$$\hat{\theta}_i^{(m)}(t) = V_i^{-1}(t) \begin{bmatrix} V_{1,m}(t) \\ \dots \\ V_{i,m}(t) \end{bmatrix}. \tag{8}$$

minimises the quadratic form (4). It is the LS estimate of the vector $\theta_i^{(m)}$. The minimum value of the quadratic form (4) is equal to

$$\Lambda_{m,i+1}(t) = V_{m,m}(t) - \\ \begin{bmatrix} V_{1,m}(t) \\ \dots \\ V_{i,m}(t) \end{bmatrix}^T V_i^{-1}(t) \begin{bmatrix} V_{1,m}(t) \\ \dots \\ V_{i,m}(t) \end{bmatrix}. \tag{9}$$

2.2 Direct Time Recursions

The formulae (5) defining the information matrix $V_i^{(m)}(t)$ can be written in a recursive form

$$V_i^{(m)}(t) = \lambda V_i^{(m)}(t-1) + \\ \left[\frac{\varphi_i(t)}{\varphi_{p+1,m}(t)}\right] \left[\frac{\varphi_i(t)}{\varphi_{p+1,m}(t)}\right]^T. \tag{10}$$

Let us define the scalar variables for $m = 2, 3, ...p+1; i = 1, ..., m-1$:

$$e_{m,i+1}(t) = \varphi_{p+1,m}(t) - \hat{\theta}_i^{(m)T}(t-1)\varphi_i(t) \tag{11}$$

and

$$\gamma_{i+1}(t-1) = \varphi_i^T(t)\lambda^{-1}V_i^{-1}(t-1)\varphi_i(t) \tag{12}$$

The scalar $e_{m,i+1}(t)$ for $m = 2, 3, ...p+1; i = 1, ..., m-1$ denotes the prediction error associated with the nested problem $\theta_i^{(m)}(t)$.

The direct update formulas for $\hat{\theta}_i^{(m)}(t)$ and for $\Lambda_{m,i+1}(t)$ are given by:

$$\hat{\theta}_i^{(m)}(t) = \hat{\theta}_i^{(m)}(t-1) + \\ \lambda^{-1}V_i^{-1}(t-1)\varphi_i(t)(1+\gamma_{i+1}(t-1))^{-1}e_{m,i+1}(t) \tag{13}$$

$$\Lambda_{m,i+1}(t) = \lambda\Lambda_{m,i+1}(t-1) + \\ e_{m,i+1}^2(t)(1+\gamma_{i+1}(t-1))^{-1} \tag{14}$$

The direct recursions (13) and (14) can be proven by substitution of (10), into the definitions (8) and (9).

2.3 Nesting of Square Root Factors

We shall deal with a numerically robust LD decomposition of the matrices $V_i(t); i = 1, 2, ..p+1$ which leads to a highly modular Recursive (Modified) Gram–Schmidt (RGS, RMGS) algorithm [7]

$$V_i(t) = L_i(t)\Lambda_i(t)L_i^T(t) \tag{15}$$

In (15), $L_i(t)$ is a lower triangular matrix with units on the diagonal and $\Lambda_i(t)$ is a diagonal matrix with positive diagonal elements.

In addition, the LD decomposition of matrix $V_i^{(m)}(t)$ will be considered:

$$V_i^{(m)}(t) = L_i^{(m)}(t)\Lambda_i^{(m)}(t)L_i^{(m)T}(t) \tag{16}$$

In (16), $L_i^{(m)}(t)$ is a $(i+1, i+1)$ lower triangular matrix with unity diagonal elements and $\Lambda_i^{(m)}(t)$ is a $(i+1, i+1)$ diagonal matrix with positive diagonal elements.

Let us find the structure of $L_i^{(m)}(t)$ and $\Lambda_i^{(m)}(t)$ resulting from the fact that $V_i(t)$ is nested in $V_i^{(m)}(t), m = 2, ..., p+1; i = 1, 2, ..., m-1$. The partitioning

$$\left[\begin{array}{c|c} V_i(t) & \begin{matrix} V_{1,m}(t) \\ \cdots \\ V_{i,m}(t) \end{matrix} \\ \hline V_{1,m}(t), ..., V_{i,m}(t) & V_{m,m}(t) \end{array}\right] =$$

$$\left[\begin{array}{c|c} * & \\ \hline * & * \end{array}\right] \left[\begin{array}{c|c} * & \\ \hline & * \end{array}\right] \left[\begin{array}{c|c} * & * \\ \hline & * \end{array}\right] \tag{17}$$

and the comparison of both sides of (17) gives:

$$\begin{aligned} L_i^{(m)}(t) &= \left[\begin{array}{c|c} L_i(t) & \\ \hline [V_{1,m}..V_{i,m}](t)L_i^{-T}(t)\Lambda_i^{-1}(t) & 1 \end{array}\right] \\ \Lambda_i^{(m)}(t) &= \left[\begin{array}{c|c} \Lambda_i(t) & \\ \hline & \Lambda_{m,i+1}(t) \end{array}\right] \\ \Lambda_{m,i+1}(t) &= V_{m,m}(t) - \\ & \begin{bmatrix} V_{1,m}(t) \\ \cdots \\ V_{i,m}(t) \end{bmatrix}^T V_i^{-1}(t) \begin{bmatrix} V_{1,m}(t) \\ \cdots \\ V_{i,m}(t) \end{bmatrix} \end{aligned} \tag{18}$$

Result: The square root factor $L_i^{(m)}(t), m = 2, 3, ..., p+1; i = 1, 2, ...m-1$ related to the identification of the $\theta_i^{(m)}$ problem contains the matrix $L_i(t)$. The diagonal $\Lambda_i^{(m)}(t), m = 2, 3, ..., p+1; i = 1, 2, ...m-1$ is composed from the matrix $\Lambda_i(t)$ and last item on the diagonal is the scalar $\Lambda_{m,i+1}(t)$ defined in (9).

In particular, for $m = 2, 3, ..., p+1; i = m-1$ the result implies that $L_{p+1}(t)$ and $\Lambda_{p+1}(t)$ contain the nested matrices $L_i(t)$ and $\Lambda_i(t), i = 1, 2, ..., p$.

The diagonal matrix Λ_{p+1} contains the scalars $\Lambda_{i,i}(t), i = 1, 2, ..., p+1$.

2.4 Definition of Parcor Coefficients

The vector of the parcor coefficients, related to the problem of estimation of $\theta_i^{(m)}, m = 2, 3, ..., p+1; i = 1, 2, ...m-1$, can be defined as a vector with i elements denoted by $K_i^{(m)}(t)$:

$$\begin{aligned} K_i^{(m)}(t) &\hat{=} \Lambda_i(t) L_i^T(t) \hat{\theta}_i^{(m)}(t) && (19) \\ &= L_i^{-1}(t) \begin{bmatrix} V_{1,m}(t) \\ ... \\ V_{i,m}(t) \end{bmatrix} && (20) \end{aligned}$$

The relation (20) can be proven by substitution of (8) to (19).

Using the vector of the parcor coefficients, the last row of the matrix $L_i^{(m)}(t)$ reads:

$$L_{i+1}^{(m)}(t) = \left[\begin{array}{c|c} L_i & \\ \hline K_i^{(m)T}(t) \Lambda_i^{-1}(t) & 1 \end{array} \right] \qquad (21)$$

and the scalar $\Lambda_{m,i+1}(t)$, defined in (9), can be expressed using the parcor vector as

$$\Lambda_{m,i}(t) = V_{m,m}(t) - K_i^{(m)T}(t) \Lambda_i^{-1}(t) K_i^{(m)}(t) \qquad (22)$$

2.5 Nesting of Parcor Coefficients

Notice first that the matrix $L_{m+1}^{-1}(t)$ contains the nested matrix $L_m^{-1}(t)$. This follows from the nesting of $L_m(t)$ in $L_{m+1}(t)$ and from the equation

$$L_{m+1}(t) L_{m+1}^{-1}(t) = I \qquad (23)$$

From the equation:

$$\left(V_m^{(m+1)}(t) \right)^{-1} = \left[\begin{array}{c|c} L_m^{-T}(t) & \\ \hline * & 1 \end{array} \right]$$
$$\left[\begin{array}{c|c} \Lambda_m^{-1}(t) & \\ \hline & \Lambda_{m+1,m+1}^{-1}(t) \end{array} \right] \left[\begin{array}{c|c} L_m^{-1} & * \\ \hline & 1 \end{array} \right]$$

and the definition of $\hat{\theta}_m^{(m+1)}(t)$, (8), can be derived the structure of the matrix $L_{m+1}^{-1}(t)$:

$$L_{m+1}^{-1}(t) = \left[\begin{array}{c|c} L_m^{-1} & \\ \hline -\hat{\theta}_m^{(m+1)T}(t) & 1 \end{array} \right] \qquad (24)$$

The nesting (24) enables to present the key relation of the vectors $K_{m-1}^{(m)T}(t)$ and $K_i^{(m)T}(t)$ for $m = 2, 3, ...p+1; i = 1, 2, ...m-1$:

$$K_{m-1}^{(m)}(t) = L_{m-1}^{-1}(t) \begin{bmatrix} V_{1,m}(t) \\ ... \\ V_{m-1,m}(t) \end{bmatrix}$$

$$= \left[\begin{array}{c|c} L_i^{-1} & 0 \\ \hline * & * \end{array}\right] \left[\begin{array}{c} V_{1,m}(t) \\ \cdots \\ V_{i,m}(t) \\ \hline * \end{array}\right] \tag{25}$$

From (25) follows:

Result: The vector $K_i^{(m)}(t)$ is nested in the vector $K_{m-1}^{(m)}(t)$ for $m = 2, 3, ...p + 1; i = 1, ..., m - 1$.

$$K_{m-1}^{(m)}(t) = \left[\begin{array}{c} K_i^{(m)}(t) \\ \hline * \end{array}\right] \tag{26}$$

2.6 Simplified Notation

Because of nesting (26) it is possible to drop out the upper index of the coefficients $K_{m-1}^{(m)}(t)$ for $m = 2, 3, ..., p + 1$:

$$\begin{aligned} K_{m-1}^{(m)}(t) &\equiv K_m(t) \\ &\equiv [k_{m,1}(t), ..., k_{m,m-1}(t)]^T \end{aligned} \tag{27}$$

The scalars $\Lambda_{m,i+1}(t)$ and $\Lambda_{m,i}(t)$ are related by the order recursion given by

$$\Lambda_{m,i+1}(t) = \Lambda_{m,i}(t) - k_{m,i}^2(t)\Lambda_{i,i}^{-1}(t) \tag{28}$$

or

$$\frac{\Lambda_{m,i+1}(t)}{\Lambda_{m,i}(t)} = 1 - \frac{k_{m,i}^2(t)}{\Lambda_{m,i}(t)\Lambda_{i,i}^{-1}(t)} \tag{29}$$

with

$$\Lambda_{m,1}(t) = V_{m,m}(t). \tag{30}$$

This follows from (22) and nesting (26). The order recursion (29) plays key role in the transformation of the Aposteriori Givens algorithm (48)-(49) to the Normalised Aposteriori Givens algorithm (50)-(52).

Result: The matrix $L_{p+1}(t)$ and the diagonal matrix $\Lambda_{p+1}(t)$ contain all the relevant information for all the $p(p + 1)/2$ RLS subproblems related to the identification of regression models $\theta_i^{(m)}$ for $m = 2, 3, ...p + 1; i = 1, ..., m - 1$.

3 Information Filter and Nested Filters

This section provides the summary of the information filter with a variable structure based on the nesting of the lower order subproblems. It highlights the nesting of lower order structures, and variables provided as a byproduct from the filter.

3.1 Prediction Errors

The error $e_{m,i+1}(t)$ can be expressed:

$$e_{m,i+1}(t) = \varphi_{p+1,m}(t) - \\ \varphi_i^T(t) L_i^{-T}(t-1) \Lambda_i^{-1}(t-1) K_i^{(m)}(t-1). \tag{31}$$

This follows from the substitution of (20) to (11).

The vector

$$E_i(t) = L_i^{-1}(t-1)\varphi_i(t) \tag{32}$$

is composed from the lower order errors defined in (11)

$$E_i(t) = [e_{1,1}(t), e_{2,2}(t), \ldots, e_{i,i}(t)]^T. \tag{33}$$

This follows from the special structure of the matrix $L_i^{-1}(t)$ derived in (24).

Using the nesting (26), the relation (31) can be computed by the recursion:

$$\begin{aligned} e_{m,i+1}(t) &= \varphi_{p+1,m}(t) - \\ &\quad E_i^T(t)\Lambda_i^{-1}(t-1)K_i^m(t-1) \\ &= e_{m,i}(t) - e_{i,i}(t)\Lambda_{i,i}^{-1}(t-1)k_{m,i}(t-1) \end{aligned} \tag{34}$$

This defines the highly modular recursion of the final information filter:

cycle for m = 1,2,. . . ,p+1:
  $e_{m,1}(t) = \varphi_{p+1,m}(t)$
  cycle for i = 1,2,. . . ,m-1: empty for $m = 1$
    $e_{m,i+1}(t) = e_{m,i}(t) -$
      $k_{m,i}(t-1)\Lambda_{i,i}^{-1}(t-1)e_{i,i}(t)$. (35)
  end of cycle for i
end of cycle for m

The computation (35) can be mapped on the parallel, extremely modular computing structure depicted in Fig. 1.

Result: The systolic array Fig. 1. produces all the lower-order prediction errors $e_{m,i}(t)$ as the intermediate data propagated in the array.

3.2 Time Update of Parcor Coefficients

The time update of $k_{m,i}(t)$ is given by:

$$k_{m,i}(t) = \lambda k_{m,i}(t-1) + \\ e_{m,i}(t)e_{i,i}(t)/(1+\gamma_i(t-1)) \tag{36}$$

It can be derived from (20) by substitution of (10) and by the substitution the time update formulas (13) to the rows of the matrix $L_i^{-1}(t)$.

3.3 Order Recursion of $\gamma_{i+1}(t-1)$

Finally, the scalar $\gamma_{i+1}(t-1)$ can be computed by the order recursion

$$\gamma_{i+1}(t-1) = \gamma_i(t-1) + e_{i,i}^2(t)\lambda^{-1}\Lambda_{i,i}^{-1}(t-1) \tag{37}$$

with $\gamma_1(t-1) = 0$.

It follows from the definition (12) and the special structure of the matrix $L_{m+1}^{-1}(t)$ derived in(24).

Using the recursion in (37), the variable $\gamma_{i+1}(t-1)$ can be computed in the diagonal processors of the array Fig. 1.

Result: The derived information filter is summarised in an algorithmic form in (38). The filter can be mapped on the systolic array presented in Fig. 1. The detailed processor specification is given by the formulas (39) and (40).

Result: Because of the nesting, the systolic array Fig. 1. contains and recursively estimates all the $p(p+1)/2$ identification subproblems related to the estimation of the regression models.

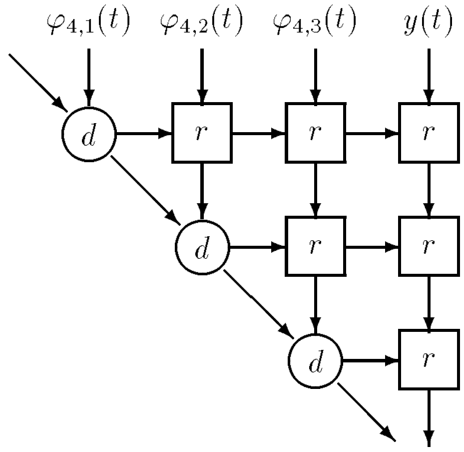

Figure 1. Systolic array with a variable structure implementing the information filter. The array structure is identical with the QRD, QR and for the aposteriori QR systolic array. These arrays differ in the internal data representation. This paper highlights, that these arrays carry (in transformed forms) all the relevant information related to the nested lower order identification problems.

Information Filter

$$
\begin{array}{l}
e_{p+1,1}(t) = \varphi_{p+1,p+1}(t) = y(t) \\
\gamma_1(t-1) = 0 \\
\text{cycle for m = 1,2,...,p:} \\
\quad e_{m,1}(t) = \varphi_{p+1,m}(t) \\
\quad \text{cycle for i = 1,2,...,m-1: (empty for } m=1) \\
\quad\quad e_{m,i+1}(t) = e_{m,i}(t) - \\
\quad\quad\quad\quad k_{m,i}(t-1)\Lambda_{i,i}^{-1}(t-1)e_{i,i}(t) \\
\quad\quad k_{m,i}(t) = \lambda k_{m,i}(t-1) + \\
\quad\quad\quad\quad e_{m,i}(t)e_{i,i}(t)/(1+\gamma_i(t-1)) \\
\quad \text{end of cycle for } i \\
\quad \gamma_{m+1}(t-1) = \gamma_m(t-1) + \\
\quad\quad\quad e_{m,m}^2(t)\lambda^{-1}\Lambda_{m,m}^{-1}(t-1) \\
\quad \Lambda_{m,m}(t) = \lambda\Lambda_{m,m}(t-1) + \\
\quad\quad\quad e_{m,m}^2(t)/(1+\gamma_m(t-1)) \\
\quad e_{p+1,m+1}(t) = e_{p+1,m}(t) - \\
\quad\quad\quad k_{p+1,m}(t-1)\Lambda_{m,m}^{-1}(t-1)e_{m,m}(t) \\
\quad k_{p+1,m}(t) = \lambda k_{p+1,m}(t-1) + \\
\quad\quad\quad e_{p+1,m}(t)e_{m,m}(t)/(1+\gamma_m(t-1)) \\
\text{end of cycle for } m
\end{array}
\tag{38}
$$

Processors (Fig. 1.) $\boxed{d_{i,i}}$

$$
\begin{array}{rcl}
\text{Input NW:} & & \gamma_i(t-1) \\
\text{Input N:} & & e_{i,i}(t) \\
\text{Compute:} & & \\
\gamma_{i+1}(t-1) & := & \gamma_i(t-1) \\
& & +e_{i,i}^2(t)\lambda^{-1}\Lambda_{i,i}^{-1}(t-1) \\
\Lambda_{i,i}(t) & := & \lambda\Lambda_{i,i}(t-1) + \\
& & e_{i,i}^2(t)/(1+\gamma_i(t-1)) \\
\text{Output SE:} & & \gamma_{i+1}(t-1) \\
\text{Output E:} & & e_{i,i}(t)\,;\ \gamma_{i+1}(t-1)\,;\ \Lambda_{i,i}(t-1)
\end{array}
\tag{39}
$$

Processors (Fig. 1.) $\boxed{r_{m,i}}$

$$
\begin{array}{rl}
\text{Input W:} & e_{i,i}(t)\,;\ \gamma_i(t-1)\,;\ \Lambda_{i,i}(t-1) \\
\text{Input N:} & e_{m,i}(t) \\
\text{Compute:} &
\end{array}
$$

$$
\begin{aligned}
e_{m,i+1}(t) &:= e_{m,i}(t) - \\
&\quad k_{m,i}(t-1)\Lambda_{i,i}^{-1}(t-1)e_{i,i}(t) \\
k_{m,i}(t) &:= \lambda k_{m,i}(t-1) + \\
&\quad e_{m,i}(t)e_{i,i}(t)/(1+\gamma_i(t-1)) \\
\text{Output S:} &\quad e_{m,i+1}(t) \\
\text{Output E:} &\quad e_{i,i}(t)\,;\ \gamma_i(t-1)\,;\ \Lambda_{i,i}(t-1)
\end{aligned}
\tag{40}
$$

4 Nesting in Established Systolic Filters

The algorithms based on Givens rotations are typically used for estimation of one RLS estimate with one fixed structure.

The existence of the side products or nested RLS subproblems is often overlooked, because of the transformed nature of the internal variables.

In this section, the above given analysis is used for the highlighting how the well established systolic arrays based on the Givens orthogonal rotations carry (in a transformed forms) the information related to the nested lower order identification problems.

The notation used for the descriotion of the processors is identical to the standard descriptions of cells based on the orthogonal Givens rotations [11].

The inputs, internal states and outputs of the cells are interpreted, using the variables defined in this paper. This enables to exploite the nested transformed information.

The main objective is to provides a guide for the users of the different QR arrays based on the Givens rotations. The guide is summarised in the appendix of the paper. It explains how to reconstruct from the transformed variables the nested lower order RLS subproblems $\theta_i^{(m)}$ for $m = 2, 3, ...p + 1; i = 1, ..., m - 1$.

The appendix provides cell descriptions for following algorithms:

1. The Square-Root-Free Givens Algorithm

2. The Givens with Square-Roots Algorithm

3. The Aposteriori Square-Roots Givens Algorithm

The cell description formulas can be proven simply by substitution of the specified values and comparison with the information algorithm (38) and the corresponding processor descriptions (39) and (40).

5 Normalised Aposteriori Square-Roots Givens Algorithm

The formulae (36) for the time update of the coefficients $k_{m,i}(t)$ (partial correlation of different error signals), supports the idea of normalising them into the range $[-1, 1]$ by suitable scaling factors.

Normalised coefficients defined as

$$\bar{r}_{m,i}(t) = k_{m,i}(t) / \left(\Lambda_{m,i}^{1/2}(t) \Lambda_{i,i}^{1/2}(t) \right) . \tag{41}$$

have guaranteed the range of values $[-1, 1]$. It follows from the inspection of the relation (29).

The scaling factors $\Lambda_{m,i+1}(t)$ are also used for the definition if normalised prediction errors

$$e_{m,i+1}(t) / \left((1 + \gamma_{i+1}(t-1))^{1/2} \Lambda_{m,i+1}^{1/2}(t) \right) \tag{42}$$

propagated by the normalised array (Fig. 2.). These normalised errors have the guaranteed range of values $[-1, 1]$. It follows from (14).

The key step, enabling to perform the transformation of the Aposteriori Square-Root Givens Algorithm (48),(49) to the Normalised Square-Root Givens Algorithm (50)-(52) is the relation (29). It expresses the fractions resulting from the transformation by the normalised parameters, present in the processors of the normalised array (Fig. 2.)

$$\frac{\Lambda_{m,i+1}(t)}{\Lambda_{m,i}(t)} = 1 - \bar{r}_{m,i}^2(t) \tag{43}$$

The final normalised algorithm is specified by (50)-(52) in the appendix. The corresponding normalised systolic array is shown in Fig. 2.

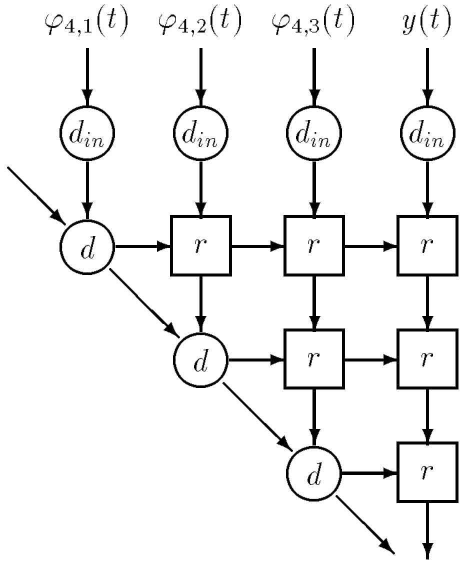

Figure 2. Normalised aposteriori Givens with square roots

6 Computer Simulation

Figure 3. and Fig. 4 plot the results of computer simulation. A 3-rd order time variable FIR regression model $\theta_3^{(4)}$ was simulated with the first 2 parameters time variable. The system was identified by an information filter related to a 5-th order FIR regression model $\theta_5^{(6)}$. Fig. 3. plots the 3-rd order RLS estimates $\hat{\theta}_3^{(6)}(t)$ reconstructed from the variables nested in all of the presented filters. The filters were all implemented in the double precision floating point arithmetic in this case. All filters provide the estimates $\hat{\theta}_3^{(6)}(t)$ identical with the estimates generated by the, independently working, 3-rd order FIR RLS identification, estimating directly $\hat{\theta}_3^{(4)}(t)$ parameters.

Figure 3 and Fig. 4 plot: the system input, the first 2 of 3 simulated and estimated parameters and finally the prediction error $e_{6,4}(t)$. The exponential weigting factor was in all cases $\lambda = 0.99$. The initial condition were defined by a diaginal matrix $V_6(0) = 0.001I$. This corresponds to $\hat{\theta}_i^{(m)} = 0$, $L_i^{(m)} = I$ and $\Lambda_i^{(m)} = 0.001I$. The data $\varphi_6(t) = [\varphi_5^T(t), y(t)]^T$ was quantised by the 12 bit A/D D/A converters in all experiments.

The periodical section of the input signal was selected to generate ill conditioned matrices. This helped to demonstrate the effects of the short word-length, 14-bit fixed-point implementation of the normalised aposteriori Givens algorithm.

7 Conclusions

We have shown the existence of the nested lower order RLS estimates in the systolic implementation of the information filters.

The second result of the paper are processor descriptions presented in the appendix. They can serve as a guideline for the users of the established QR and QRD algorithms based on the Givens rotations. The processor descriptions explain how to obtain the lower order nested subproblems from the QR systolic arrays based on the standard versions of Givens rotations.

The final result of the paper is the introduction of the normalised version of the aposteriori Givens algorithm. The definition of nested lower order subproblems is the key to the derivation of the normalised algorithm.

All normalised parameters and variables propagated in the main section of the corresponding triangular array have the guaranteed range $[-1, 1]$.

The normalisation opens the prospect of systolic implementation with the considerable saving of silicon, by the reduced requirement on the word-length of the fixed-point implementation due to the transformation of variables into the guaranteed range.

References

[1] Bierman, G. J. (1977). *Factorisation Methods for Discrete Sequential Estimation.*

Academic Press, New York.

[2] Friedlander, B. (1982). Lattice Filters for Adaptive Processing. *Proc. IEEE,*, **70**, no. 8, pp. 829-867,

[3] Gentleman,W.M. and H.T.Kung. (1981). Matrix Triangularisation by Systolic Arrays. Proc. SPIE, Vol.298, *Real Time Signal Processing IV*, 1981, pp. 19-26.

[4] Kadlec, J., F.M.F. Gaston and G. W. Irwin, (1992b). Regularised Lattice - Ladder Adaptive Filter. *Mutual Impact of Computing Power and Control Theory*, Plenum Inc., New York, NY, pp. 235-247.

[5] Kadlec, J. (1993). The Cell level Description of Systolic Block Regularised QR Filter. *1993 IEEE Workshop on VLSI Signal Processing VI*, Veldhoven, The Netherlands. pp. 298-306.

[6] Lee, D., M. Morf and B. Friedlander. (1981). Recursive Least Squares Ladder Estimation Algorithms. *IEEE Trans., ASSP-29*, pp. 627-641.

[7] Ling, F., D. Manolakis, and J.G. Proakis. (1986a). A recursive modified Gram-Schmidt algorithm for least squares estimation. *IEEE Trans., ASSP-34*, pp. 829-835.

[8] Ling, F., D. Manolakis, and J.G. Proakis. (1986b). Numerically Robust Leasr-Squares Lattice-Ladder Algorithms with Direct Updating of the Reflection Coefficients *IEEE Trans., ASSP-34*, pp. 837-845.

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

[10] McWhirter, J.G. (1983). Recursive Least Squares Minimisation Using a Systolic Array, Proc. *SPIE*, Vol. 431, *Real Time Signal Processing VI*, pp. 105-109.

[11] McWhirter, J.G. (1989). Algorithmic engineering - an emerging discipline, *SPIE* Vol. 1152, Advanced Algorithms and Architectures for Signal Processing IV, pp. 2-15.

[12] Moonen, M. and J.G. McWhirter. (1993). Systolic Array for Recursive Least Squares by Inverse Updating. *Electronics Letters*, **29** no. 13., pp. 1217-1218.

[13] Walther, J. S. (1971). A unified algorithm for elementary functions. *Soring Joint Computer Conference* pp. 379-385.

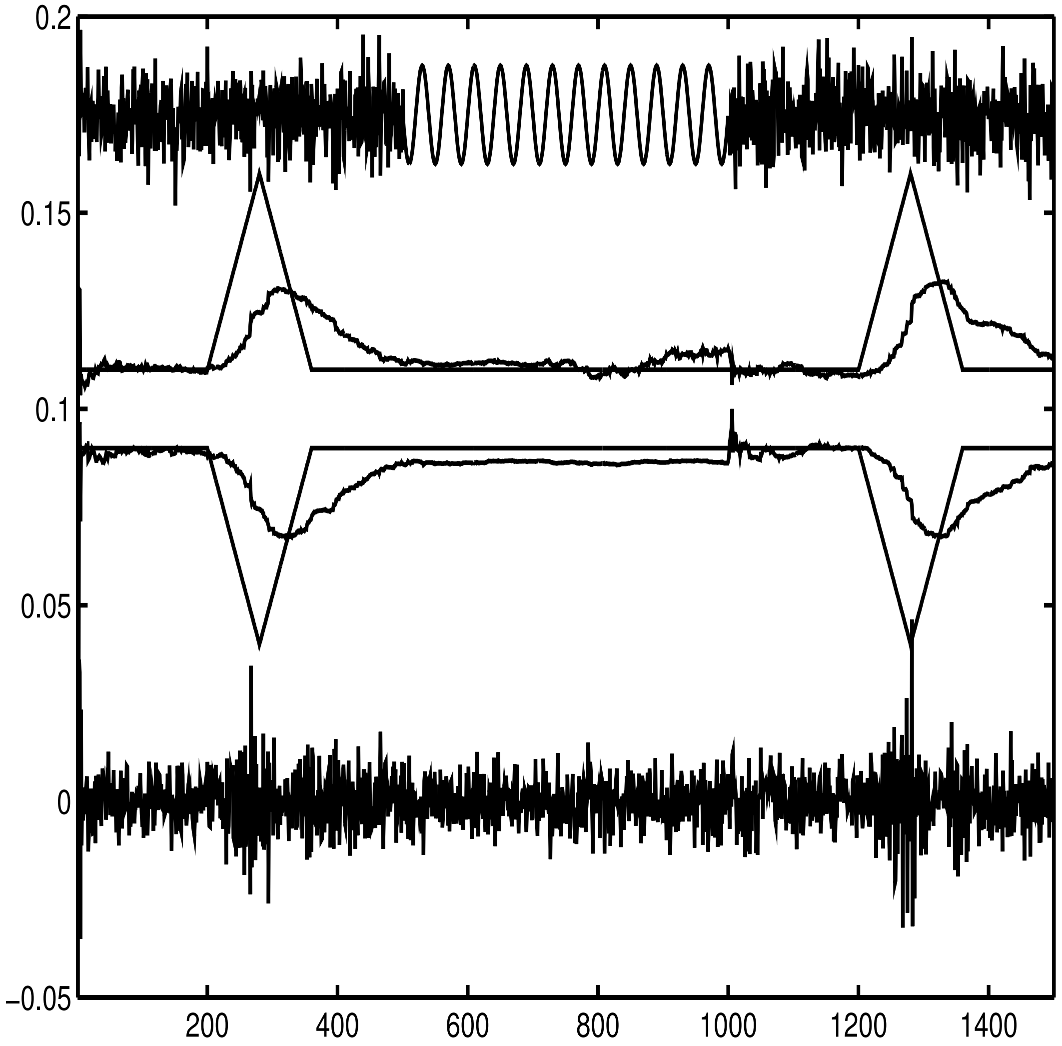

Figure 3. Figure plots the system input $u(t) = \varphi_{6,1}(t)$; the first two simulated parameters , the first two estimated parameters and the prediction error. It presents the estimation of a 3-rd order model reconstructed from the data nested in the 5-th order array. The results from the double precision floating point implementation are identical for all presented filters.

The exponential weigting factor was in all cases $\lambda = 0.99$. The initial condition were defined by a diaginal matrix $V_6(0) = 0.001I$. This corresponds to $\hat{\theta}_i^{(m)} = 0$, $L_i^{(m)} = I$ and $\Lambda_i^{(m)} = 0.001I$. The data $\varphi_6(t) = [\varphi_5^T(t), y(t)]^T$ was quantised by the 12 bit A/D D/A converters in all experiments.

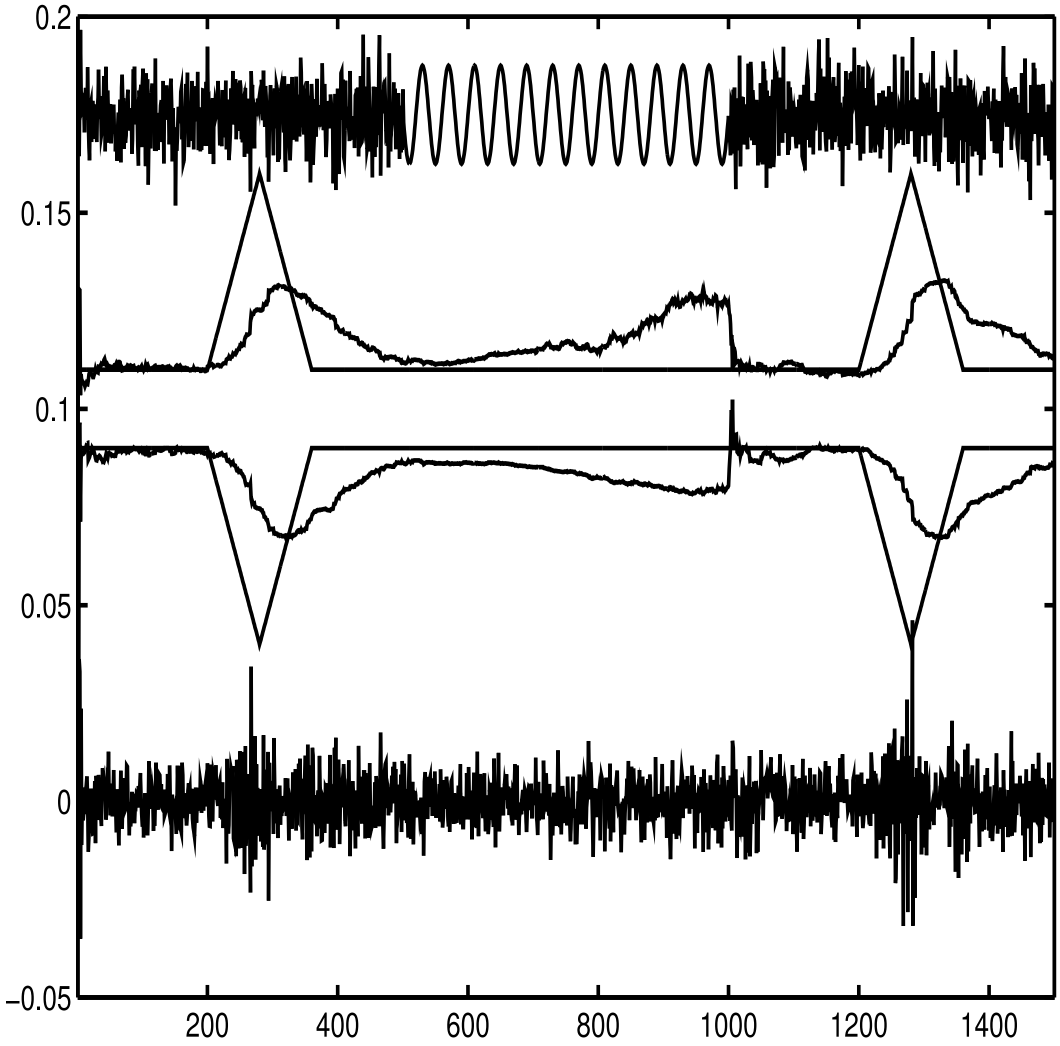

Figure 4. Figure plots the parameter estimates and prediction error of a 3-rd order model reconstructed from the internal normalised data nested in the 5-th order normalised systolic array of the normalised aposteriori Givens algorithm (Fig. 2). The array was simulated in 14 bit fixed point arithmetic.
The periodical section of the input signal was selected to generate ill conditioned matrices. This helped to demonstrate the effects of the short word-length, 14-bit fixed-point implementation of the normalised aposteriori Givens algorithm.

A1: Square-Root-Free Givens

Processors (Fig. 1.) $\boxed{d_{i,i}}$

$$
\begin{array}{lll}
\text{Input NW:} & & \delta_{\text{in}} \equiv \dfrac{1}{1+\gamma_i(t-1)} \\
\text{Input N:} & & x_{\text{in}} \equiv e_{i,i}(t) \\
\text{Internal:} & & d \equiv \Lambda_{i,i}(t-1);\, d' \equiv \Lambda_{i,i}(t) \\
\beta \equiv \Lambda^{1/2} & & \bar{c} \equiv \lambda \dfrac{\Lambda_{i,i}(t-1)}{\Lambda_{i,i}(t)} \\
 & & \bar{s} \equiv \dfrac{e_{i,i}(t)}{\Lambda_{i,i}(t)(1+\gamma_i(t-1))} \\
\text{Compute:} & & \\
z & := & x_{\text{in}} \\
d' & := & \beta^2 d + \delta_{\text{in}} x_{\text{in}}^2 \\
\bar{c} & := & \beta^2 d / d' \\
\bar{s} & := & \delta_{\text{in}}\, x_{\text{in}} / d' \\
\delta_{\text{out}} & := & \bar{c}\, \delta_{\text{in}} \\
\text{Output SE:} & & \delta_{\text{out}} \equiv \dfrac{1}{1+\gamma_{i+1}(t-1)} \\
\text{Output E:} & & z\,;\, \bar{c}\,;\, \bar{s}
\end{array}
\tag{44}
$$

Processors (Fig. 1.) $\boxed{r_{m,i}}$

$$
\begin{array}{lll}
\text{Input W:} & & z \equiv e_{i,i}(t)\,;\, \bar{c} \equiv \lambda \dfrac{\Lambda_{i,i}(t-1)}{\Lambda_{i,i}(t)} \\
 & & \bar{s} \equiv \dfrac{e_{i,i}(t)}{\Lambda_{i,i}(t)(1+\gamma_i(t-1))} \\
\text{Input N:} & & x_{\text{in}} \equiv e_{m,i}(t) \\
\text{Internal:} & & r \equiv \dfrac{k_{m,i}(t-1)}{\Lambda_{i,i}(t-1)}\,;\, r' \equiv \dfrac{k_{m,i}(t)}{\Lambda_{i,i}(t)} \\
\text{Compute:} & & \\
x_{\text{out}} & := & x_{\text{in}} - r\, z \\
r' & := & \bar{c}\, r + \bar{s}\, x_{\text{in}} \\
\text{Output S:} & & x_{\text{out}} \equiv e_{m,i+1}(t) \\
\text{Output E:} & & z\,;\, \bar{c}\,;\, \bar{s}
\end{array}
\tag{45}
$$

Note: The implementation (44),(45) avoids the square roots. The division is performed only in the diagonal cells.

A2: Givens with Square-Roots

Processor (Fig. 1.) $\boxed{d_{i,i}}$

$$
\begin{array}{ll}
\text{Input NW:} & \hat{\delta}_{\text{in}} \equiv \dfrac{1}{(1+\gamma_i(t-1))^{1/2}} \\
\text{Input N:} & \hat{x}_{\text{in}} \equiv \dfrac{e_{i,i}(t)}{(1+\gamma_i(t-1))^{1/2}} \\
\text{Internal:} & \hat{d} \equiv \Lambda_{i,i}^{1/2}(t-1); \hat{d}' \equiv \Lambda_{i,i}^{1/2}(t) \\
\beta \equiv \Lambda^{1/2} & \hat{c} \equiv \lambda^{1/2}\dfrac{\Lambda_{i,i}^{1/2}(t-1)}{\Lambda_{i,i}^{1/2}(t)} \\
 & \hat{s} \equiv \dfrac{e_{i,i}(t)}{\Lambda_{i,i}^{1/2}(t)(1+\gamma_i(t-1))^{1/2}}
\end{array}
$$

Compute:

$$
\begin{aligned}
\hat{d}' &:= ((\beta\, \hat{d})^2 + \hat{x}_{\text{in}}^2)^{1/2} \\
\hat{c} &:= \beta\, \hat{d}/\hat{d}' \\
\hat{s} &:= \hat{x}_{\text{in}}/\hat{d}' \\
\delta_{\text{out}}^{1/2} &:= \hat{c}\, \delta_{\text{in}}^{1/2}
\end{aligned}
$$

$$
\begin{array}{ll}
\text{Output SE:} & \delta_{\text{out}}^{1/2} \equiv \dfrac{1}{(1+\gamma_{i+1}(t-1))^{1/2}} \\
\text{Output E:} & \hat{c}\,;\ \hat{s}
\end{array}
\tag{46}
$$

Processors (Fig. 1.) $\boxed{r_{m,i}}$

$$
\begin{array}{ll}
\text{Input W:} & \hat{c} \equiv \lambda^{1/2}\dfrac{\Lambda_{i,i}^{1/2}(t-1)}{\Lambda_{i,i}^{1/2}(t)} \\
 & \hat{s} \equiv \dfrac{e_{i,i}(t)}{\Lambda_{i,i}^{1/2}(t)(1+\gamma_i(t-1))^{1/2}} \\
\text{Input N:} & x_{\text{in}} \equiv \dfrac{e_{m,i}(t)}{(1+\gamma_i(t-1))^{1/2}} \\
\text{Internal:} & \hat{r} \equiv \dfrac{k_{m,i}(t-1)}{\Lambda_{i,i}^{1/2}(t-1)}\,;\ \hat{r}' \equiv \dfrac{k_{m,i}(t)}{\Lambda_{i,i}^{1/2}(t)}
\end{array}
$$

Compute:

$$
\begin{aligned}
\hat{x}_{\text{out}} &:= \hat{c}\, \hat{x}_{\text{in}} - \beta\, \hat{s}\, \hat{r} \\
\hat{r}' &:= \beta\, \hat{c}\, \hat{r} + \hat{s}\, \hat{x}_{\text{in}}
\end{aligned}
$$

$$
\begin{array}{ll}
\text{Output S:} & \hat{x}_{\text{out}} \equiv \dfrac{e_{m,i+1}(t)}{(1+\gamma_{i+1}(t-1))^{1/2}} \\
\text{Output E:} & \hat{c}\,;\ \hat{s}
\end{array}
\tag{47}
$$

Note: The implementation (46),(47) has division only in the diagonal cells.

A3: Aposteriori Square-Roots Givens

Processor (Fig. 1.) $\boxed{d_{i,i}}$

$$
\begin{array}{ll}
\text{Input NW:} & \hat{\delta}_{\text{in}} \equiv \dfrac{1}{(1+\gamma_i(t-1))^{1/2}} \\
\text{Input N:} & \hat{x}_{\text{in}} \equiv \dfrac{e_{i,i}(t)}{(1+\gamma_i(t-1))^{1/2}} \\
\text{Internal:} & \hat{d} \equiv \Lambda_{i,i}^{1/2}(t-1)\,;\ \hat{d}' \equiv \Lambda_{i,i}^{1/2}(t) \\
\beta \equiv \lambda^{1/2} & \hat{c} \equiv \lambda^{1/2}\dfrac{\Lambda_{i,i}^{1/2}(t-1)}{\Lambda_{i,i}^{1/2}(t)} \\
 & \hat{s} \equiv \dfrac{e_{i,i}(t)}{\Lambda_{i,i}^{1/2}(t)(1+\gamma_i(t-1))^{1/2}}
\end{array}
$$

Compute:

$$
\begin{array}{rcl}
\hat{d}' & := & ((\beta\,\hat{d})^2 + \hat{x}_{\text{in}}^2)^{1/2} \\
\hat{s} & := & \hat{x}_{\text{in}}/\hat{d}' \\
\hat{c} & := & \beta\,\hat{d}/\hat{d}' \equiv (1-\hat{s}^2)^{1/2} \\
\hat{\delta}_{\text{out}} & := & \hat{c}\,\hat{\delta}_{\text{in}}
\end{array}
$$

$$
\begin{array}{ll}
\text{Output SE:} & \hat{\delta}_{\text{out}} \equiv \dfrac{1}{(1+\gamma_{i+1}(t-1))^{1/2}} \\
\text{Output E:} & \hat{c}\,;\ \hat{s}
\end{array}
\tag{48}
$$

Processors (Fig. 1.) $\boxed{r_{m,i}}$

$$
\begin{array}{ll}
\text{Input W:} & \hat{c} \equiv \lambda^{1/2}\dfrac{\Lambda_{i,i}^{1/2}(t-1)}{\Lambda_{i,i}^{1/2}(t)} \\
 & \hat{s} \equiv \dfrac{e_{i,i}(t)}{\Lambda_{i,i}^{1/2}(t)(1+\gamma_i(t-1))^{1/2}} \\
\text{Input N:} & \hat{x}_{\text{in}} \equiv \dfrac{e_{m,i}(t)}{(1+\gamma_i(t-1))^{1/2}} \\
\text{Internal:} & \hat{r} \equiv \dfrac{k_{m,i}(t-1)}{\Lambda_{i,i}^{1/2}(t-1)}\,;\ \hat{r}' \equiv \dfrac{k_{m,i}(t)}{\Lambda_{i,i}^{1/2}(t)}
\end{array}
$$

Compute:

$$
\begin{array}{rcl}
\hat{r}' & := & \beta\,\hat{c}\,\hat{r} + \hat{s}\,\hat{x}_{\text{in}} \\
\hat{x}_{\text{out}} & := & (\hat{x}_{\text{in}} - \hat{s}\,\hat{r}')/\hat{c}
\end{array}
$$

$$
\begin{array}{ll}
\text{Output S:} & \hat{x}_{\text{out}} \equiv \dfrac{e_{m,i+1}(t)}{(1+\gamma_{i+1}(t-1))^{1/2}} \\
\text{Output E:} & \hat{c}\,;\ \hat{s}
\end{array}
\tag{49}
$$

Note: The implementation (48),(49) is the included to make a link with the normalised aposteriori Givens algorithm cells (50)-(52).

A4: Normalised Aposteriori Square-Roots Givens

Processors (Fig. 2.) $\boxed{d_{\text{in},i}}$

$$
\begin{array}{ll}
\text{Input N:} & \bar{x}_{\text{in}} \equiv \varphi_{p+1,i}(t) \\
\text{Internal:} & \hat{d}_{\text{in}} \equiv \Lambda_{i,1}^{1/2}(t-1)\,;\ \hat{d}'_{\text{in}} \equiv \Lambda_{i,1}^{1/2}(t) \\
 & \beta \equiv \lambda^{1/2}
\end{array}
$$

Compute:

$$
\begin{aligned}
\hat{d}'_{\text{in}} &:= \left((\beta\hat{d}_{\text{in}})^2 + \hat{x}_{\text{in}}^2\right)^{1/2} \\
\bar{x}_{\text{out}} &:= \bar{x}_{\text{in}}/\hat{d}'_{\text{in}}
\end{aligned}
$$

$$
\text{Output E:} \quad \bar{x}_{\text{out}} \equiv \frac{e_{i,1}(t)}{\Lambda_{i,1}^{1/2}(t)(1+\gamma_1(t-1))^{1/2}} \tag{50}
$$

Processors (Fig. 2.) $\boxed{d_{i,i}}$

$$
\begin{array}{ll}
\text{Input NW:} & \hat{\delta}_{\text{in}} \equiv \dfrac{1}{(1+\gamma_i(t-1))^{1/2}} \\
\text{Input N:} & \bar{x}_{\text{in}} \equiv \dfrac{e_{i,i}(t)}{\Lambda_{i,i}^{1/2}(t)(1+\gamma_i(t-1))^{1/2}}
\end{array}
$$

Compute:

$$
\begin{aligned}
\hat{s} &:= \bar{x}_{\text{in}} \\
\hat{c} &:= (1-\hat{s}^2)^{1/2} \\
\hat{\delta}_{\text{out}} &:= \hat{c}\,\hat{\delta}_{\text{in}}
\end{aligned}
$$

$$
\begin{array}{ll}
\text{Output SE:} & \hat{\delta}_{\text{out}} \equiv \dfrac{1}{(1+\gamma_{i+1}(t-1))^{1/2}} \\
\text{Output E:} & \hat{c}\,;\ \hat{s}
\end{array} \tag{51}
$$

Processors (Fig. 2.) $\boxed{r_{m,i}}$

$$
\begin{array}{ll}
\text{Input W:} & \hat{c} \equiv \lambda^{1/2}\dfrac{\Lambda_{i,i}^{1/2}(t-1)}{\Lambda_{i,i}^{1/2}(t)} \\
 & \hat{s} \equiv \dfrac{e_{i,i}(t)}{\Lambda_{i,i}^{1/2}(t)(1+\gamma_i(t-1))^{1/2}} \\
\text{Input N:} & \bar{x}_{\text{in}} \equiv \dfrac{e_{m,i}(t)}{\Lambda_{m,i}^{1/2}(t)(1+\gamma_i(t-1))^{1/2}} \\
\text{Internal:} & \bar{r} \equiv \dfrac{k_{m,i}(t-1)}{\Lambda_{m,i}^{1/2}(t-1)\Lambda_{i,i}^{1/2}(t-1)} \\
 & \bar{r}' \equiv \dfrac{k_{m,i}(t)}{\Lambda_{m,i}^{1/2}(t)\Lambda_{i,i}^{1/2}(t)}
\end{array}
$$

Compute:

$$\begin{aligned}\bar{r}' &:= \hat{c}\,\bar{r}\,(1-\bar{x}_{\text{in}}^2)^{1/2} + \hat{s}\,\bar{x}_{\text{in}} \\ \bar{x}_{\text{out}} &:= (\bar{x}_{\text{in}} - \hat{s}\,\bar{r}')\,/\,\left(\hat{c}\,(1-(\bar{r}')^2)^{1/2}\right)\end{aligned}$$

Output S: $\bar{x}_{\text{out}} \equiv \dfrac{e_{m,i+1}(t)}{\Lambda_{m,i+1}^{1/2}(t)(1+\gamma_{i+1}(t-1))^{1/2}}$

Output E: $\hat{c}\,;\,\hat{s}$ (52)