A UNIFIED SQRT-FREE RANK-1 UP/DOWN-DATING APPROACH FOR RECURSIVE LEAST-SQUARES PROBLEMS **V1.5**

S.F. Hsieh
Dept. of Communication Engineering
Nat'l Chiao Tung University
Hsinchu, Taiwan 30039

K.J.R. Liu
Electrical Engineering Dept.
Systems Research Center
University of Maryland
College Park, MD 20742

K. Yao
Electrical Engineering Dept.
UCLA
Los Angeles, CA 90024-1594

ABSTRACT

The QR-decomposition (QRD)-based recursive least-squares (RLS) methods have been shown to be useful and effective towards *adaptive signal processing* in modern communications, radar, and sonar systems implementable with various modern parallel and systolic array architectures. Planar (Givens) and hyperbolic rotations are the most commonly used methods in performing QRD up/downdating. But the generic formula for these rotations require explicit square-root (sqrt) computations, which are quite undesirable from the practical VLSI circuit design point of view. Since the sqrt operation takes up much area and its computational time is long (due to many iterations), the associated area/time efficiency is poor.

1 Planar and Hyperbolic Rotations

A 2×2 *planar* (Givens) rotation matrix is the most fundamental orthogonal matrix in performing QR decomposition. A *planar* (Givens) rotation is given by $\begin{bmatrix} c & s \\ -s & c \end{bmatrix}$, and is used to premultiply a two-row matrix

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_p \\ \beta_1 & \beta_2 & \cdots & \beta_p \end{bmatrix}$$

to introduce a zero element in the $(2,1)$ location such that it becomes

$$\begin{bmatrix} \alpha'_1 & \alpha'_2 & \cdots & \alpha'_p \\ 0 & \beta'_2 & \cdots & \beta'_p \end{bmatrix},$$

where

$$c = \alpha_1 / \sqrt{\alpha_1^2 + \beta_1^2}, \quad (1)$$

$$s = \beta_1 / \sqrt{\alpha_1^2 + \beta_1^2}, \quad (2)$$

$$\alpha'_1 = \sqrt{\alpha_1^2 + \beta_1^2}, \quad (3)$$

$$\alpha'_j = c\alpha_j + s\beta_j, \quad (4)$$

$$\beta'_j = -s\alpha_j + c\beta_j, \quad j = 2, \cdots, p. \quad (5)$$

Similarly, a *hyperbolic* rotation matrix is given by $\begin{bmatrix} \hat{c} & -\hat{s} \\ -\hat{s} & \hat{c} \end{bmatrix}$, and the corresponding parameters satisfy:

$$\hat{c} = \alpha_1 / \sqrt{\alpha_1^2 - \beta_1^2}, \quad (6)$$

This work is partially supported by a UC MICRO grant and the NSF grant NCR-8814407.

$$\hat{s} = \beta_1 / \sqrt{\alpha_1^2 - \beta_1^2}, \quad (7)$$

$$\alpha'_1 = \sqrt{\alpha_1^2 - \beta_1^2}, \quad (8)$$

$$\alpha'_j = \hat{c}\alpha_j - \hat{s}\beta_j, \quad (9)$$

$$\beta'_j = -\hat{s}\alpha_j + \hat{c}\beta_j, \quad j = 2, \cdots, p. \quad (10)$$

2 Prototypes of Generalized Sqrt-Free Algorithms

In VLSI circuit design, square-root operation is expensive, because it takes up much area or is slow (due to many iterations). Therefore, it is advantageous to avoid square-root operations or minimize the required number of such operations. We will focus on how to meet this goal for the 2×2 planar and hyperbolic rotations.

By taking out a scaling factor from each row, the two rows under consideration before and after the orthogonal transformations are given by

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_p \\ \beta_1 & \beta_2 & \cdots & \beta_p \end{bmatrix} = \begin{bmatrix} \sqrt{k_1} & 0 \\ 0 & \sqrt{k_2} \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \cdots & a_p \\ b_1 & b_2 & \cdots & b_p \end{bmatrix} \quad (11)$$

and

$$\begin{bmatrix} \alpha'_1 & \alpha'_2 & \cdots & \alpha'_p \\ 0 & \beta'_2 & \cdots & \beta'_p \end{bmatrix} = \begin{bmatrix} \sqrt{k'_1} & 0 \\ 0 & \sqrt{k'_2} \end{bmatrix} \begin{bmatrix} a'_1 & a'_2 & \cdots & a'_p \\ 0 & b'_2 & \cdots & b'_p \end{bmatrix}. \quad (12)$$

Now, our task is to find the expressions for k'_1, k'_2, a'_1, $\{(a'_j, b'_j),\ j = 2, \cdots, p\}$, in terms of $k_1, k_2, \{(a_j, b_j),\ j = 1, \cdots, p\}$, such that *NO* square-root operation is actually needed. The square-root expressions of $\sqrt{k_1}$, $\sqrt{k_2}$, $\sqrt{k'_1}$, and $\sqrt{k'_2}$ in (11) and (12) are used for representational purposes only and are not actually performed.

For simplicity, let us focus on planar rotations only, similar derivations for the square-root-free hyperbolic rotation can also be obtained by replacing k_2 with $-k_2$ and k'_2 with $-k'_2$. Replacing $\alpha_j = \sqrt{k_1} a_j$, $\beta_j = \sqrt{k_2} b_j$, $\alpha'_j = \sqrt{k'_1} a'_j$, $\beta'_j = \sqrt{k_2} b'_j$, $j = 1, \ldots, p$, in (1) - (5) leads to

$$c = \sqrt{k_1}\, a_1 \,/\, \sqrt{k_1 a_1^2 + k_2 b_1^2}, \quad (13)$$

$$s = \sqrt{k_2}\, b_1 \,/\, \sqrt{k_1 a_1^2 + k_2 b_1^2}, \quad (14)$$

CH2977-7/91/0000-1017 $1.00 © 1991 IEEE

$$\sqrt{k_1'}\,a_1' = \sqrt{k_1 a_1^2 + k_2 b_1^2}\,, \tag{15}$$

$$\sqrt{k_1'}\,a_j' = \frac{k_1 a_1}{\sqrt{k_1 a_1^2 + k_2 b_1^2}}\,a_j + \frac{k_2 b_1}{\sqrt{k_1 a_1^2 + k_2 b_1^2}}\,b_j\,, \quad j = 2, \cdots, p. \tag{16}$$

$$\sqrt{k_2'}\,b_j' = \frac{-\sqrt{k_1 k_2}\,b_1}{\sqrt{k_1 a_1^2 + k_2 b_1^2}}\,a_j + \frac{\sqrt{k_1 k_2}\,a_1}{\sqrt{k_1 a_1^2 + k_2 b_1^2}}\,b_j\,, \tag{17}$$

After simplifications we have

$$a_1' = \sqrt{\frac{k_1 a_1^2 + k_2 b_1^2}{k_1'}}\,, \tag{18}$$

$$a_j' = \frac{1}{\sqrt{k_1'}\sqrt{k_1 a_1^2 + k_2 b_1^2}}[k_1 a_1 a_j + k_2 b_1 b_j]\,, \quad j = 2, \cdots, p. \tag{19}$$

$$b_j' = \frac{\sqrt{k_1 k_2}}{\sqrt{k_2'}\sqrt{k_1 a_1^2 + k_2 b_1^2}}[-b_1 a_j + a_1 b_j]\,, \tag{20}$$

To avoid square-roots, we need to determine k_1' and k_2' such that a_1', a_j' and b_j' will not require square-root operations. Let us express k_1' and k_2' as

$$k_1' = \frac{k_1 a_1^2 + k_2 b_1^2}{\mu^2}\,, \tag{21}$$

$$k_2' = \frac{k_1 k_2}{\nu^2(k_1 a_1^2 + k_2 b_1^2)}\,, \tag{22}$$

where μ and ν will be determined later to be any square-root-free functions of k_1, k_2, a_1, and b_1. Indeed, with (21) - (22), (18) - (20) can be computed with no square-roots, and we have the following updating formulas without square-roots:

$$k_1' = \frac{k_1 a_1^2 + k_2 b_1^2}{\mu^2}\,, \tag{23}$$

$$k_2' = \frac{k_1 k_2}{\nu^2(k_1 a_1^2 + k_2 b_1^2)} = \frac{k_1 k_2}{\mu^2 \nu^2 k_1'}\,, \tag{24}$$

$$a_1' = \mu\,, \tag{25}$$

$$a_j' = \frac{\mu}{k_1 a_1^2 + k_2 b_1^2}[k_1 a_1 a_j + k_2 b_1 b_j] = \frac{k_1 a_1 a_j + k_2 b_1 b_j}{\mu k_1'}\,, \tag{26}$$

$$b_j' = \nu[-b_1 a_j + a_1 b_j]\,, \quad j = 2, \cdots, p\,, \tag{27}$$

$$c = \frac{a_1}{\mu}\sqrt{\frac{k_1}{k_1'}}\,, \tag{28}$$

$$s = \frac{b_1}{\mu}\sqrt{\frac{k_2}{k_1'}}\,. \tag{29}$$

Notice that square-root operations disappear in our formulas of (23) - (27) needed in the planar and hyperbolic rotations. The use of the rotation parameter c in (28) (with a square-root operation) will be further considered in the next section when the optimum residual e is desired. Later work will show that it is possible to obtain e without any square-root operation where the explicit computation of the rotation parameter c can be bypassed. To avoid repetitive computations and take the advantage of previous computed results, (24), (26), (28) and (29) use the newly updated k_1' of (23). As stated earlier, we are still free to choose those two parameters μ and ν. Different choices of μ and ν will affect the number of multiplications and divisions, as well as the numerical stability and parallelism of computations.

It can be shown that this newly derived approach generalizes *all* of the previously known researches on the square-root-free algorithms via an proper choice of μ and ν. Among them are Gentleman [4], Hammarling [6], Bareiss [1], Kalson and Yao [7], Ling, *etc.* [8], Barlow and Ispen [2], Chen and Yao [3], Götze and Schwiegelshohn [5]. Table 1 lists various square-root-free algorithms and the corresponding choices of μ and ν.

3 Sqrt-Free Triangular Array Updating and Optimum Residual Acquisition

In this section, we will apply the prototypes of sqrt-free rotations developed above to QRD-based RLS filtering problems. To be specific, we are interested in updating from

$$\begin{bmatrix} R & \mathbf{u} \\ \mathbf{x}^T & y \end{bmatrix} \tag{30}$$

to

$$\begin{bmatrix} R' & \mathbf{u}' \\ \mathbf{0}^T & v \end{bmatrix}. \tag{31}$$

It can be shown [9] that the $p \times p$ upper triangular matrix R' can be obtained through a sequence of p Givens rotations and the optimum residual e for the newly appended data $[\mathbf{x}^T \vdots y]$ is given by

$$e = -(\prod_{i=1}^{p} c_i)\,v, \tag{32}$$

with c_i representing the cosine value of the i-th rotation angles.

Factoring out the scaling constants into the pre-multiplying diagonal matrix leads (30) to the form of

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} & u_1 \\ & r_{22} & \cdots & r_{2p} & u_2 \\ & & \ddots & \vdots & \vdots \\ & & & r_{pp} & u_p \\ x_1 & x_2 & \cdots & x_p & y \end{bmatrix} \tag{33}$$

$$= \begin{bmatrix} \sqrt{k_1} & & & & \\ & \ddots & & & \\ & & \sqrt{k_p} & & \\ & & & \sqrt{k_q} & \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} & a_{1,p+1} \\ & a_{22} & \cdots & a_{2p} & a_{2,p+1} \\ & & \ddots & \vdots & \vdots \\ & & & a_{pp} & a_{p,p+1} \\ b_1 & b_2 & \cdots & b_p & b_{p+1} \end{bmatrix}. \tag{34}$$

Unlike the previously developed formula, where we are only interested in updating k_i, a_{ij}, to k'_i, a'_{ij} and zeroing out all the b_i's, this time we do also need to know the cosine values explicitly as required in the optimum residual given in (32).

After the first rotation, b_1 will be zeroed-out and we have

$$k'_1 = \frac{k_1 a_{11}^2 + k_q b_1^2}{\mu_1^2}, \tag{35}$$

$$k_q^{(1)} = \frac{k_1 k_q}{\mu_1^2 \nu_1^2 k'_1}, \tag{36}$$

$$a'_{11} = \mu_1, \tag{37}$$

$$a'_{1j} = \frac{1}{\mu_1 k'_1}[k_1 a_{11} a_{1j} + k_q b_1 b_j], \tag{38}$$

$$b_j^{(1)} = \nu_1[-b_1 a_{1j} + a_{11} b_j], \quad j = 2, \cdots, p+1 \tag{39}$$

$$c_1 = \frac{a_{11}}{\mu_1}\sqrt{\frac{k_1}{k'_1}}, \tag{40}$$

with (μ_1, ν_1) being the parameter pair which are still free to be chosen later. Note the close analogy of (35) - (40) to those of (23) - (28). Similarly, after the i-th rotation $(1 < i \le p)$, we have

$$k'_i = \frac{k_i a_{ii}^2 + k_q^{(i-1)} b_i^{(i-1)^2}}{\mu_i^2}, \tag{41}$$

$$k_q^{(i)} = \frac{k_i k_q^{(i-1)}}{\mu_i^2 \nu_i^2 k'_i}, \tag{42}$$

$$a'_{ii} = \mu_i, \tag{43}$$

$$a'_{ij} = \frac{1}{\mu_i k'_i}[k_i a_{ii} a_{ij} + k_q^{(i-1)} b_1^{(i-1)} b_j^{(i-1)}], \tag{44}$$

$$b_j^{(i)} = \nu_i[-b_i^{(i-1)} a_{ij} + a_{ii} b_j^{(i-1)}], \quad j = 2, \cdots, p+1 \tag{45}$$

$$c_i = \frac{a_{ii}}{\mu_i}\sqrt{\frac{k_i}{k'_i}}. \tag{46}$$

After p rotations are finished, (34) becomes

$$\begin{bmatrix} \sqrt{k'_1} & & & \\ & \ddots & & \\ & & \sqrt{k'_p} & \\ & & & \sqrt{k'_q} \end{bmatrix} \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1p} & a'_{1,p+1} \\ & a'_{22} & \cdots & a'_{2p} & a'_{2,p+1} \\ & & \ddots & \vdots & \vdots \\ & & & a'_{pp} & a'_{p,p+1} \\ 0 & 0 & \cdots & 0 & b_{p+1}^{(p)} \end{bmatrix} \tag{47}$$

which has the form of $\begin{bmatrix} R' & \mathbf{u}' \\ 0^T & v \end{bmatrix}$ in (31). The optimum residual e in (32) now becomes

$$e = -\left(\prod_{i=1}^{p} \frac{a_{ii}}{\mu_i}\sqrt{\frac{k_i}{k'_i}}\right)\sqrt{k_q^{(p)}}\, b_{p+1}^{(p)}. \tag{48}$$

To further simplify the expression in (48), we notice that $k_q^{(p)}$ can be computed recursively as follows,

$$k_q^{(p)} = \left(\frac{\sqrt{k_p/k'_p}}{\mu_p \nu_p}\right)^2 k_q^{(p-1)} \tag{49}$$

$$= \left(\frac{\sqrt{k_p/k'_p}}{\mu_p \nu_p}\right)^2 \left(\frac{\sqrt{k_{p-1}/k'_{p-1}}}{\mu_{p-1}\nu_{p-1}}\right)^2 k_q^{(p-2)} \tag{50}$$

$$\vdots$$

$$= \left[\prod_{i+1}^{p}\left(\frac{\sqrt{k_i/k'_i}}{\mu_i \nu_i}\right)^2\right] k_q, \tag{51}$$

where (42) is used in the recursion.

With (51) substituted into (48), we have

$$e = -\left(\prod_{i=1}^{p} \frac{a_{ii}}{\mu_i^2 \nu_i}\frac{k_i}{k'_i}\right)\sqrt{k_q}\, b_{p+1}^{(p)}. \tag{52}$$

Because $\sqrt{k_q}[b_1, b_2, \ldots, b_p, b_{p+1}] = [x_1, x_2, \ldots, x_p, y]$ is the appended new data row, we are certainly free to choose $k_q = 1$ to reduce the arithmetic complexity and simplify the expression in (52). Therefore, a lemma on obtaining the sqrt-free optimum residual is given below.

Lemma 1 *(Sqrt-free optimum residual)*
The optimum residual e can be computed with no square-root operations as follows:

$$e = -\left(\prod_{i=1}^{p} \frac{a_{ii}}{\mu_i^2 \nu_i}\frac{k_i}{k'_i}\right) b_{p+1}^{(p)}. \tag{53}$$

McWhirter[9] successfully employed Gentleman's proposition [4] in computing the residual e without sqrt operations. By choosing

$$\mu_i = \nu_i = a_{ii} = 1, \quad 1 \le i \le p, \tag{54}$$

the optimum residual can be reduced to

$$e = -\left(\prod_{i=1}^{p} \frac{k_i}{k'_i}\right) b_{p+1}^{(p)} \qquad \text{(Gentleman/McWhirter)}. \tag{55}$$

This result was first observed by McWhirter.

Another example can be taken from Hammarling's suggestion [6] as follows,

$$\mu_i = \frac{k_i a_{ii}^2 + k_q^{(i-1)} b_i^{(i-1)^2}}{k_i a_{ii}}, \tag{56}$$

$$\nu_i = 1 / a_{ii}, \qquad i = 1, \cdots, p. \tag{57}$$

then it follows that $k'_i = k_i a_{ii}/\mu_i$ and the optimum residual is given by

$$e = -\left(\prod_{i=1}^{p} \frac{a_{ii}}{\mu_i^2 \nu_i}\frac{\mu_i}{a_{ii}}\right) b_{p+1}^{(p)} \tag{58}$$

$$= -\left(\prod_{i=1}^{p} \frac{1}{\mu_i \nu_i}\right) b_{p+1}^{(p)} \quad (59)$$

$$= -\left(\prod_{i=1}^{p} \frac{a_{ii}}{a'_{ii}}\right) b_{p+1}^{(p)} \quad \text{(Hammarling)}. \quad (60)$$

4 Conclusions

Planar (Givens) and hyperbolic rotations are the most commonly used methods in performing QRD up/downdating. Most of these rotation-based methods require explicit square-root computations, which are undesirable from the practical VLSI circuit design point of view. Since the square-root operation takes up much area and its computational time is slow (due to many iterations), the associated area/time efficiency is poor. This is the first effort to establish the basic understanding toward all known square-root-free QRD algorithms, from which the basic criterion is seen to be simple. This unified approach also provides a fundamental framework for the square-root-free RLS algorithms which are essential for practical VLSI implementations.

References

[1] E. H. Bareiss, "Numerical solution of the weighted least squares problems by G-transformations," Tech. Report 82-03-NAM-03, Dept. of Elec. Engr. and Comp. Sci., Northwestern University, Evanston, IL, Apr. 1982.

[2] J. L. Barlow and I. C. F. Ipsen, "Scaled Givens rotations for the solution of linear least squares problems on systolic arrays," *SIAM J. Sci. Stat. Comput.*, Vol. 8, No. 5, pp. 716–733, Sept. 1987.

[3] M. J. Chen and K. Yao, "Comparison of QR least-squares algorithms for systolic array processing," *Proc. Conf. on Information Science and Systems*, pp. 683–688, Mar. 1988.

[4] W. M. Gentleman, "Least squares computations by Givens transformations without square roots," *J. Inst. Maths Applics*, Vol. 12, pp. 329–336, 1973.

[5] J. Götze and U. Schwiegelshohn, "An orthogonal method for solving systems of linear equations without square roots and with few divisions," *Proc. IEEE ICASSP*, pp. 1298–1301, 1989.

[6] S. Hammarling, "A note on modifications to the Givens plane rotation," *J. Inst. Maths Applics*, Vol. 13, pp. 215–218, 1974.

[7] S. Kalson and K. Yao, "A Class of Least-Squares Filtering and Identification Algorithms with Systolic Array Architectures," *IEEE Trans. on Inform. Theory*, pp. 43–52, Jan. 1991. An earlier version appeared as S. Kalson and K. Yao, "Systolic array processing for order and time recursive generalized least-squares estimation," *Proc. SPIE 564, Real-Time Signal Processing VIII*, pp. 28–38, 1985.

[8] F. Ling, D. Manolakis, and J. G. Proakis, "A recursive modified Gram-Schmidt algorithm for least-squares estimation," *IEEE Trans. on Acous., Speech, and Signal Processing*, Vol. ASSP-34, No. 4, pp. 829–836, Aug. 1986.

[9] J. G. McWhirter, "Recursive least-squares minimization using a systolic array," *Proc. SPIE 431, Real time signal processing VI*, pp. 105–112, 1983.

Table 1: Choices of μ and ν for various sqrt-free Givens rotations

μ	ν	author(year)
1	1	Gentleman('73)
$\frac{k_1 a_1^2 + k_2 b_1^2}{k_2 b_1}$	$\frac{-1}{b_1}$	"
$\frac{k_1 a_1^2 + k_2 b_1^2}{k_1 a_1}$	$\frac{1}{a_1}$	"
$\frac{k_1 a_1^2 + k_2 b_1^2}{k_1 a_1}$	$\frac{1}{a_1}$	Hammarling('74)
$\frac{k_1 a_1^2 + k_2 b_1^2}{k_1 a_1}$	$\frac{-1}{b_1}$	"
$\frac{k_1 a_1^2 + k_2 b_1^2}{k_1 a_1}$	$\frac{k_2 b_1}{k_1 a_1^2}$	"
1	$\frac{1}{a_1}$	Bareiss('82)
$a_1 + k_2 b_1^2$	$\frac{1}{a_1 + k_1 b_1^2}$	Ling('86),Kalson/Yao('85)
$\frac{k_1 a_1^2 + k_2 b_1^2}{k_1 a_1}$	$\frac{1}{a_1}$	Chen/Yao ('88)
$\frac{k_1 a_1^2 + k_2 b_1^2}{k_1 k_2}$	1	Götze/Schwiegelshohn('89)
$2^{r_1}(k_1 a_1^2 + k_2 b_1^2)$	2^{r_2}	Barlow/Ispen('87)