ELSEVIER

Parallel Computing 28 (2002) 1257–1273

PARALLEL COMPUTING

www.elsevier.com/locate/parco

Greedy Givens algorithms for computing the rank-*k* updating of the QR decomposition ☆

Erricos John Kontoghiorghes *

Institut d'informatique, Université de Neuchâtel, Émile-Argand 11, Case Postale 2, CH-2007 Neuchâtel, Switzerland

Received 1 March 2001; received in revised form 16 February 2002; accepted 25 April 2002

Abstract

A Greedy Givens algorithm for computing the rank-1 updating of the QR decomposition is proposed. An exclusive-read exclusive-write parallel random access machine computational model is assumed. The complexity of the algorithms is calculated in two different ways. In the unlimited parallelism case a single time unit is required to apply a compound disjoint Givens rotation of any size. In the limited parallelism case all the disjoint Givens rotations can be applied simultaneously, but one time unit is required to apply a rotation to a two-element vector. The proposed Greedy algorithm requires approximately 5/8 the number of steps performed by the conventional sequential Givens rank-1 algorithm under unlimited parallelism. A parallel implementation of the sequential Givens algorithm outperforms the Greedy one under limited parallelism. An adaptation of the Greedy algorithm to compute the rank-*k* updating of the QR decomposition has been developed. This algorithm outperforms a recently reported parallel method for small *k*, but its efficiency decreases as *k* increases.

Keywords: QR decomposition; Givens rotations; Parallel model of computation; Parallel random access machine; Complexity

☆ This work is in part supported by the Swiss National Foundation Grants 1214-056900.99/1 and 2000-061875.00/1. Part of the work was done while the author was visiting the ALADIN group at INRIA-IRISA, Rennes, France under the support of the host institution and the Swiss National Foundation Grant 83R-065887.

* Tel.: +41-32-718-2738; fax: +41-32-718-2701.

E-mail address: erricos.kontoghiorghes@unine.ch (E.J. Kontoghiorghes).

0167-8191/02/$ - see front matter
PII: S0167-8191(02)00132-1

1. Introduction

Consider the QR decomposition (QRD) of the non-singular matrix $A \in \mathbb{R}^{n\times n}$

$$A = QR, \tag{1}$$

where $Q \in \mathbb{R}^{n\times n}$ is orthogonal and $R \in \mathbb{R}^{n\times n}$ is upper triangular. If $x, y \in \mathbb{R}^{n\times k}$, then the rank-$k$ updating of the QRD (UQRD) problem is the recalculation of the QRD

$$\tilde{A} \equiv A + xy^{\mathrm{T}} = \tilde{Q}\tilde{R}, \tag{2}$$

when (1) has already been computed [11,13]. Thus, the UQRD problem is equivalent to computing the QRD

$$Q(R + zy^{\mathrm{T}}) = \tilde{Q}\tilde{R}, \tag{3}$$

where $z = Q^{\mathrm{T}}x$. The sequential Givens method solves the rank-1 UQRD problem in two stages [4,5]. The first stage computes

$$G^{\mathrm{T}}\begin{pmatrix} Z & R & Q^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} \zeta e_1 & H & G^{\mathrm{T}}Q^{\mathrm{T}} \end{pmatrix}, \tag{4}$$

where G is the product of $n-1$ Givens rotations which annihilate the elements of z from bottom to the top, $\zeta^2 = z^{\mathrm{T}}z, e_1$ is the first column of the $n \times n$ identity matrix I_n, and H is upper Hessenberg. The second stage applies $n-1$ Givens rotations to retriangularize the upper Hessenberg matrix

$$\tilde{H} = H + \zeta e_1 y^{\mathrm{T}}. \tag{5}$$

That is, it computes

$$\hat{G}^{\mathrm{T}}\begin{pmatrix} \tilde{H} & G^{\mathrm{T}}Q^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} \tilde{R} & \tilde{Q} \end{pmatrix}, \tag{6}$$

where $\hat{G}^{\mathrm{T}}$ is the product of Givens rotations which annihilate the elements of the subdiagonal of $\tilde{H}$ from top to the bottom.

In the rank-k UQRD problem, the orthogonal factorizations (4) and (6) are replaced by the factorizations

$$G_B^{\mathrm{T}}\begin{pmatrix} z & R & Q^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} R_z & H & G_B^{\mathrm{T}}Q^{\mathrm{T}} \end{pmatrix} \tag{7}$$

and

$$\hat{G}_B^{\mathrm{T}}\begin{pmatrix} \tilde{H} & G_B^{\mathrm{T}}Q^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} \tilde{R} & \tilde{Q} \end{pmatrix}, \tag{8}$$

where $G_B R_z$ is the QRD of z and $\tilde{H} = H + R_z y^{\mathrm{T}}$. A parallel algorithm—the SK algorithm—solves the rank-k ($1 \leqslant k < n$) UQRD problem using $2(n+k-2)$ compound disjoint Givens rotations (CDGRs) [9]. The SK algorithm employs the Sameh and Kuck annihilation scheme in [12] to compute the QRD of z using $(n+k-2)$ CDGRs. This results H in (7), and consequently $\tilde{H}$, to have zero in its last $n-k-1$ subdiagonals. The retriangularization of $\tilde{H}$ is obtained using $(n+k-2)$ CDGRs. A CDGR is the product of a number of disjoint Givens rotations that can simultaneously annihilate elements of a matrix [3,7]. A Givens rotation in plane (i,j) that reduces to zero the element $B_{i,k}$ when it is applied from the left of $B \in \mathbb{R}^{m\times n}$ will be denoted by $G_{i,j}^{(k)}$, where $1 \leqslant i, j \leqslant m$ and $1 \leqslant k \leqslant n$.

The complexity analyses of the algorithms to be discussed are based on an exclusive-read exclusive-write (EREW) parallel random access machine (PRAM) computational model [2,8]. It is assumed that sufficient processors are available to apply simultaneously all the disjoint Givens rotations. The complexity of applying a single Givens rotation will be defined in two different ways. In the unlimited parallelism case a single time unit is defined to be the time required to apply a Givens rotation on any matrix conformable with the transformation. In the limited parallelism case a single time unit is defined to be the time required to apply a Givens rotation to a two-element vector [9]. Thus, in the second case, the complexity of a Givens rotation is determined by the length of the vectors affected by the rotation. The complexity based on the unlimited parallelism will also hold for SIMD systems if all the matrices (i.e. z, R and Q) can be stored in a single memory layer [7]. The times to construct a Givens rotation and to compute the updating $H + R_z y^{\mathrm{T}}$ will not be taken into account. Thus, the complexity of the rank-1 algorithms includes the computations of (4) and (6). Similarly, for the complexity of the rank-k algorithms the computations (7) and (8) are considered.

In this work Greedy parallel Givens strategies are investigated which use a smaller number of CDGRs than the sequential and SK algorithms for solving the rank-1 and rank-k ($k < \log_2 n$) UQRD problems, respectively. In the Section 2 the Greedy algorithm for solving the rank-1 UQRD problem is presented. The adaptation and modification of a Greedy sequence to solve the rank-k UQRD problem is considered in Section 3. Finally, in Section 4 some conclusions are presented.

2. The Greedy rank-1 updating algorithm

Using the Greedy, or recursive doubling, method the reduction of the n-element vector z to the form ζe_1 can be achieved by applying the minimum number of $\lceil \log_2 n \rceil$ CDGRs. It is assumed that there are enough processors to apply all the disjoint rotations simultaneously. This implies that at least $\lfloor n/2 \rfloor$ processors are available. The Givens rotations are not performed on adjacent planes and, generally, the application of the single Givens rotation $G^{(1)}_{i,j}$ $(i > j)$ from the left of the augmented matrix $(z\ R)$ annihilates the ith element of z and fills in the elements j to $(i-1)$ of the ith row of R. However, a sequence of CDGRs can be found which retriangularizes R fast enough so that the complexity of solving the UQRD problem is less than that of the sequential algorithm.

Assume for simplicity that $n = 2^g$. At the ith ($i = 1, \ldots, g$) step of the Greedy algorithm the elements $2^{(g-i)} + 1$ to $2^{(g+1-i)}$ of z are annihilated by the $2^g \times 2^g$ CDGR

$$C^{(i,g)} = \prod_{j=2^{(g-i)}+1}^{2^{(g+1-i)}} G^{(1)}_{j,j-2^{(g-i)}},$$

which can also be written as

$$C^{(i,g)} = \begin{pmatrix} C^{(i-1,g-1)} & 0 \\ 0 & I_2^{(g-1)} \end{pmatrix} \quad \text{for } i > 1.$$

The orthogonal matrix G^{T} in (4) is given by $C^{(g,g)}\cdots C^{(1,g)}$. It can be proved that the matrices

$$H^{(g)} = C^{(g,g)}\cdots C^{(1,g)}R$$

and

$$\tilde{H}^{(g)} = H^{(g)} + \zeta e_1 y^{\mathrm{T}} \tag{9}$$

have special recursive structures which facilitate the development of an efficient Givens algorithm for triangularizing $\tilde{H}^{(g)}$.

Theorem 1. *The structure of the matrix $H^{(g+1)} = C^{(g+1,g+1)}\cdots C^{(1,g+1)}R$ is given by*

$$H^{(g+1)} = \begin{matrix} \\ H^{(g)} \\ R^{(g)} \end{matrix}\!\!\begin{matrix} \begin{matrix} 2^g & 2^g \end{matrix} \\ \begin{pmatrix} \tilde{B}^{(g)} & \\ \hat{B}^{(g)} & \end{pmatrix} \end{matrix}\!\!\begin{matrix} \\ 2^g, \\ 2^g \end{matrix} \tag{10}$$

where $g \geqslant 0$, $\tilde{B}^{(g)}$ and $\hat{B}^{(g)}$ are full matrices, $R^{(g)}$ is upper triangular, $H^{(g)}$ has the same structure as $H^{(g+1)}$ but with g replaced by $g-1$ and $H^{(0)}$ is a non-zero scalar.

Proof. The proof is by induction. For $g = 0$ the matrix $H^{(1)} = C^{(1,1)}R = G^{(1)}_{2,1}R$ is a 2×2 full matrix which satisfies the structure (10). The inductive hypothesis is that $H^{(g)}$ has a structure defined in (10). It is required to show that $H^{(g+1)}$ has the structure (10). Recall that $H^{(g)}$ is of order $n\times n$ with $n = 2^g$. Consider the case with $n = 2^{(g+1)}$. The application of the first CDGR from the left of the R gives

$$C^{(1,g+1)}R = \begin{matrix} \\ \bar{R}^{(g)} \\ R^{(g)} \end{matrix}\!\!\begin{matrix} \begin{matrix} 2^g & 2^g \end{matrix} \\ \begin{pmatrix} \check{B}^{(g)} & \\ \hat{B}^{(g)} & \end{pmatrix} \end{matrix}\!\!\begin{matrix} \\ 2^g, \\ 2^g \end{matrix} \tag{11}$$

where $R^{(g)}$ and $\bar{R}^{(g)}$ are upper triangular, and $\hat{B}^{(g)}$ and $\check{B}^{(g)}$ are full matrices. This is obvious since the Givens rotation $G^{(1)}_{j,j-2^g}$ fills in the elements $(j-2^g)$ to $(j-1)$ of the jth row of R $(j = 2^g+1,\ldots,2^{g+1})$. That is, the fill-in of R from the application of $C^{(1,g+1)}$ is enclosed in a parallelogram of height 2^g. For $\tilde{B}^{(g)} = \prod_{i=1}^{g} C^{(i,g)}\check{B}^{(g)}$ the application of the CDGRs from the left of R can be written as

$$\begin{aligned} H^{(g+1)} = \prod_{i=1}^{g+1} C^{(i,g+1)}R &= \begin{pmatrix} \prod_{i=1}^{g} C^{(i,g)} & 0 \\ 0 & I_{2^g} \end{pmatrix}\begin{pmatrix} \bar{R}^{(g)} & \check{B}^{(g)} \\ R^{(g)} & \hat{B}^{(g)} \end{pmatrix} \\ &= \begin{pmatrix} \prod_{i=1}^{g} C^{(i,g)}\bar{R}^{(g)} & \tilde{B}^{(g)} \\ R^{(g)} & \hat{B}^{(g)} \end{pmatrix}. \end{aligned}$$

Now, using the inductive hypothesis $H^{(g)} = \prod_{i=1}^{g} C^{(i,g)}\bar{R}^{(g)}$, it follows from the latter that $H^{(g+1)}$ can be expressed in the form

$$H^{(g+1)} = \begin{pmatrix} H^{(g)} & \tilde{B}^{(g)} \\ R^{(g)} & \hat{B}^{(g)} \end{pmatrix},$$

which completes the proof. □

Clearly $\tilde{H}^{(g)}$ (see (9)) has the same structure as $H^{(g)}$. Only the first row of $\tilde{H}^{(g)}$ differs from $H^{(g)}$. Fig. 1 shows the recursive structure of $H^{(g)}$ and the process of reducing $(z\ R)$ to $(\zeta_1 e_1\ H^{(g)})$ for $g = 4$. In the first picture the numeral i $(i = 1, \ldots, g)$ denotes the elements of z and R which are annihilated and filled in, respectively, by the ith CDGR.

The reduction of the matrix $\tilde{H}^{(g)} (g > 1)$ to upper triangular form can be obtained by constructing the orthogonal factorizations

$$\tilde{G}^{\mathrm{T}}\tilde{H}^{(g)} = \tilde{G}^{\mathrm{T}}\begin{pmatrix} \tilde{H}^{(g-1)} & \tilde{B}^{(g-1)} \\ R^{(g-1)} & \hat{B}^{(g-1)} \end{pmatrix} = \begin{pmatrix} W^{(g-1)} & \tilde{B}_*^{(g-1)} \\ \hat{R}^{(g-1)} & \hat{B}_*^{(g-1)} \end{pmatrix} \begin{matrix} 2^{g-1} \\ 2^{g-1} \end{matrix} \tag{12a}$$

and

$$\tilde{G}_*^{\mathrm{T}}\begin{pmatrix} W^{(g-1)} & \tilde{B}_*^{(g-1)} \\ \hat{R}^{(g-1)} & \hat{B}_*^{(g-1)} \end{pmatrix} = \tilde{R}, \tag{12b}$$

where $W^{(g-1)}$ and $\hat{R}^{(g-1)}$ are upper triangular matrices and $(\tilde{G}\tilde{G}_*)\tilde{R}$ is the QRD of $\tilde{H}^{(g)}$. That is, $\hat{G}^{\mathrm{T}}$ in (6) is given by $\tilde{G}_*^{\mathrm{T}}\tilde{G}^{\mathrm{T}}$.

The factorization of (12a) can be computed in $2^{(g-2)}$ steps using a Givens sequence similar to that in [7]. At the ith $(i = 1, \ldots, 2^{(g-2)})$ step the pth subdiagonal of $\tilde{H}^{(g-1)}$ is annihilated by applying the CDGR $C^{(i,g)}$, where $p = 2^{(g-2)} - i + 1$. The jth element of the pth subdiagonal is zeroed by the Givens rotation $G^{(j)}_{p+j,j+2^{(g-1)}}$, where $j = \sigma_i, \ldots, 2^{(g-2)} + i - 1$ and σ_i, is the index of the first non-zero element. Thus, $C^{(i,g)}$ is defined by

$$C^{(i,g)} = \prod_{j=\sigma_i}^{2^{(g-2)}+i-1} G^{(j)}_{p+j,j+2^{(g-1)}}. \tag{13}$$

The $2^{(g-2)}$-element vector σ can be written as $\sigma = (\sigma^{(1)}, \ldots, \sigma^{(g-1)})$, where $\sigma^{(\mu)}_\lambda = \lambda$, $\mu = 1, \ldots, g-1$ and $\lambda = 1, \ldots, \lceil 2^{g-2-\mu} \rceil$. The subvector $\sigma^{(\mu)}$ starts at position $q+1$ of σ, where $q = (2^{\mu-1} - 1)2^{g-\mu-1}$. Thus, $\sigma_{q+\lambda} \equiv \sigma^{(\mu)}_\lambda = \lambda$. For example, if $g = 5$, then $\sigma = (1, 2, 3, 4, 1, 2, 1, 1)$.

Notice that the factorization (12a) can start after the second CDGR on z has been applied. Fig. 2 illustrates the factorization of (12a), where $g = 4$. The full matrices $\tilde{B}^{(g-1)}$ and $\hat{B}^{(g-1)}$ are not shown. Arcs connecting the elements (and subrows) of $\tilde{H}^{(g-1)}$ and $R^{(g-1)}$ indicate those affected by Givens rotations.

The factorization (12b) is divided into two stages. In the first stage the upper triangular matrix $\hat{R}^{(g-1)}$ is reduced to zero by applying $2^{(g-1)}$ CDGRs. This is performed using the strategies employed to update a QL decomposition by a lower triangular matrix [6,7]. The CDGRs annihilate, diagonal by diagonal, the non-zero entries of $\hat{R}^{(g-1)}$ while preserving the triangular structure of $W^{(g-1)}$. Specifically, at the ith $(i = 1, \ldots, 2^{(g-1)})$ step the CDGR

$$C^{(i,g)} = \prod_{j=1}^{2^{(g-1)}-i+1} G^{(i+j-1)}_{j+2^{(g-1)},i+j-1}$$

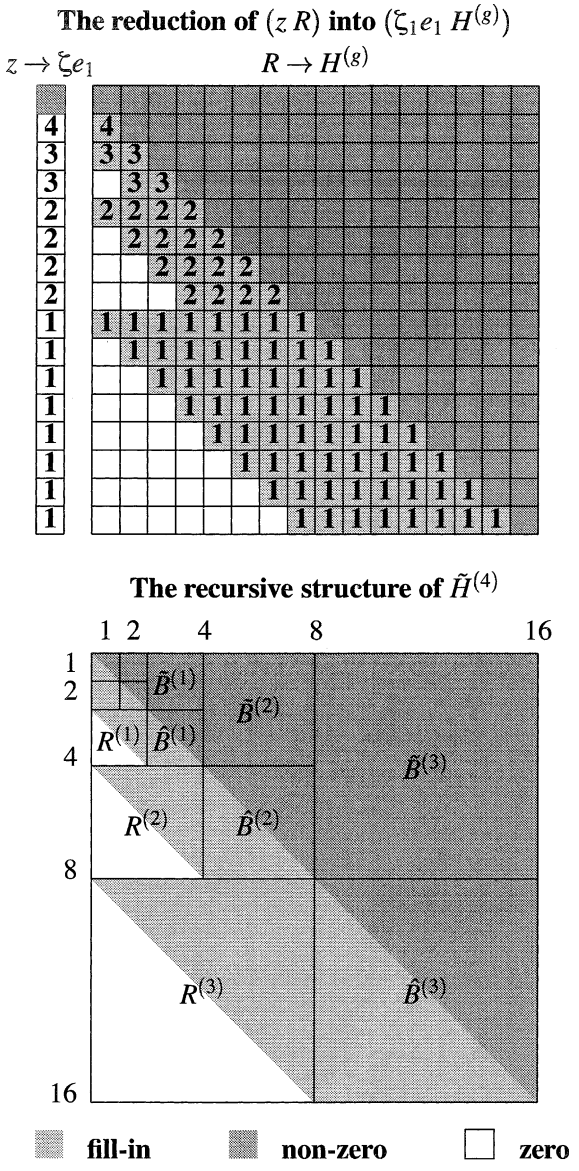

Fig. 1. The computationi of $(\zeta_1 e_1\ H^{(4)})$ using the Greedy algorithm and the structure of $\tilde{H}^{(4)}$.

is applied in order to annihilate the ith superdiagonal of $\hat{R}^{(g-1)}$ where, in this context, the main diagonal is equivalent to the first superdiagonal. Fig. 3 illustrates this Givens sequence.

In the second stage of computing (12b) the matrix $\hat{B}_*^{(g-1)}$ is triangularized using $2^g - 3$ CDGRs by employing the Sameh and Kuck (SK) annihilation scheme in

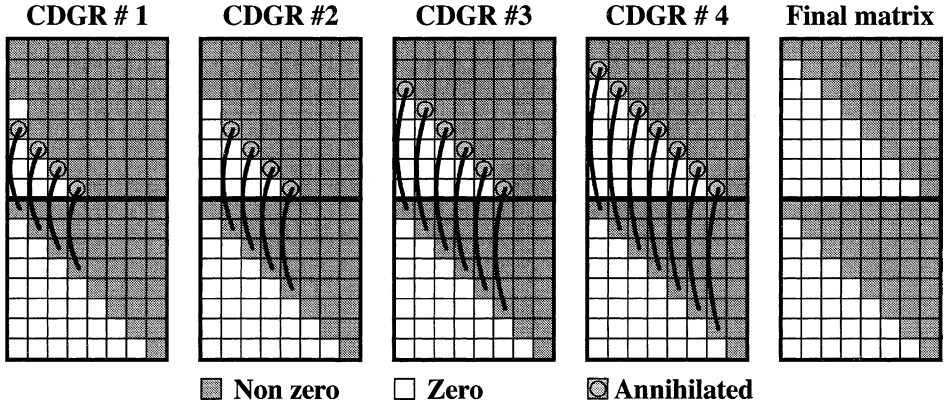

Fig. 2. Factorization of (12a) using CDGRs, where $g = 4$ and $\sigma = (1, 2, 1, 1)$.

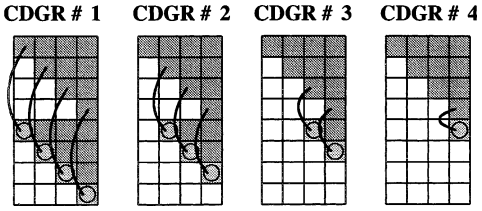

Fig. 3. Assimilation of $\hat{R}^{(g-1)}$ in the factorization of (12b) using CDGRS, where $g = 3$.

[12]. The elements of a column of $\hat{B}_*^{(g-1)}$ are annihilated by successive CDGRs from bottom to the top with the $(2i-1)$th CDGR starting to annihilate the elements of the ith column. Notice, however, that the annihilation of $\hat{B}_*^{(g-1)}$ can commence after the last two rows of $\hat{R}^{(g-1)}$ have been zeroed. That is, the annihilation of $\hat{B}_*^{(g-1)}$ starts after the second CDGR of the first stage has been applied. Hence, $2^g - 1$ CDGRs are needed to compute factorization (12b).

Fig. 4 illustrates the annihilation pattern of the Greedy parallel Givens strategies for solving the rank-1 UQRD problem, where $g = 4$. In this figure, a numeral i denotes the elements annihilated by the ith CDGR. The steps of the Greedy algorithm are shown in Algorithm 1. The standard colon notation has been used to represent sequences of adjacent elements. The loop at lines 3–19 reduces z to ζe_1 while simultaneously computing the factorization (12a). The loop at lines 20–37 computes the factorization (12b). Note, the two conditional statements (lines 5–11 and lines 12–17) controlled by the *parallel do* at line 4 are executed simultaneously. This also holds for the conditional statements controlled by the *parallel do* at line 21. A Givens rotation is applied from the left of a two-row matrix and annihilates the first element of the second row. A number of Givens rotations are applied simultaneously using a *for all* loop. For notational simplicity G_j denotes the 2×2 rotation applied at the jth step of the *for all* loop.

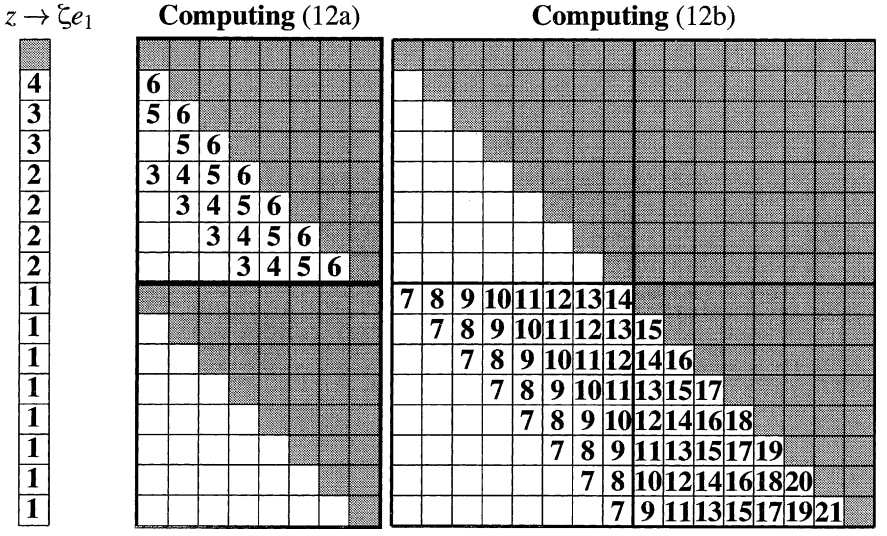

Fig. 4. Annihilation pattern of the Greedy rank-1 UQRD algorithm, where $g = 4$.

Algorithm 1. The Greedy algorithm for solving the rank-1 UQRD problem (3), where $n = 2^g$.

1. Let $z = Q^{\mathrm{T}}x$ and $\tilde{R} = (R \;\; Q^{\mathrm{T}}) \in \mathbb{R}^{n\times 2n}$, where $n = 2^g$
2. Let $\sigma_{q+\lambda} = \lambda$, where $\lambda = 1, \ldots, \lceil 2^{g-2-\mu} \rceil$, $\mu = 1, \ldots, g-1$ and $q = (2^{\mu-1} - 1)2^{g-\mu-1}$
3. **for** $i = 1, 2, \ldots, 2^{(g-2)} + 2$ **do**
4. **parallel do**
5. **if** $(i \leqslant g)$ **then**
6. **for all** $j = 1, 2, \ldots, 2^{(g-i)}$ **do-in-parallel**
7. Let $r = 2^{(g-i)} + j$
8. $$\begin{pmatrix} z_j & \tilde{R}_{j,j:2n} \\ z_r & \tilde{R}_{r,j:2n} \end{pmatrix} \leftarrow G_j \begin{pmatrix} z_j & \tilde{R}_{j,j:2n} \\ z_r & \tilde{R}_{r,j:2n} \end{pmatrix}$$
9. **end for all**
10. **if** $(i = g)$ **then** $\tilde{R}_{1,1:n} \leftarrow \tilde{R}_{1,1:n} + z_1 y^{\mathrm{T}}$ **end if**
11. **end if**
12. **if** $(i > 2)$ **then**
13. **for all** $j = \sigma_{i-2}, \ldots, 2^{(g-2)} + i - 3$ **do-in-parallel**
14. Let $s = 2^{(g-2)} + j - i + 3$ and $r = 2^{(g-1)} + j$
15. $$\begin{pmatrix} \tilde{R}_{r,j:2n} \\ \tilde{R}_{s,j:sn} \end{pmatrix} \leftarrow G_j \begin{pmatrix} \tilde{R}_{r,j:2n} \\ \tilde{R}_{s,j:2n} \end{pmatrix}$$
16. **end for all**
17. **end if**
18. **end parallel**
19. **end for**
20. **for** $i = 1, 2, \ldots, 2^g - 1$ **do**

21. **parallel do**
22. **if** $(i \leqslant 2^{(g-1)})$ **then**
23. **for all** $j = 1, 2, \ldots, 2^{(g-1)} - i + 1$ **do-in-parallel**
24. Let $r = i + j - 1$
25. $\begin{pmatrix} \tilde{R}_{r,r:2n} \\ \tilde{R}_{2^{(g-1)}+j,r:2n} \end{pmatrix} \leftarrow G_j \begin{pmatrix} \tilde{R}_{r,r:2n} \\ \tilde{R}_{2^{(g-1)}+j,r:2n} \end{pmatrix}$
26. **end for all**
27. **end if**
28. **if** $(i > 2)$ **then**
29. **for all** $j = 1, 2, \ldots, 2^{(g-1)}$ **do-in-parallel**
30. Let $r = 2^{(g-1)} + 2j - i + 1$ and $s = 2^{(g-1)} + r$
31. **if** $(j < r \leqslant 2^{(g-1)})$ **then**
32. $\tilde{R}_{s-1:s,2^{(g-1)}+j:2n} \leftarrow G_j \tilde{R}_{s-1:s,2^{(g-1)}+j:2n}$
33. **end if**
34. **end for all**
35. **end if**
36. **end parallel**
37. **end for**

2.1. Complexity

The total number of CDGRs required to solve the rank-1 UQRD using the Greedy algorithm is given by

$$T_{\mathrm{ULgr}}(g) = 2 + 2^{(g-2)} + 2^g - 1 = 5 \times 2^{(g-2)} + 1 \quad \text{for } g > 1.$$

This corresponds to the unlimited parallelism case. The number of processors required is given by

$$2^{g-1} + \sum_{i=1}^{2^{g-1}} (3 \times 2^{g-1} + i) = 2^{g-2}(7 \times 2^{g-1} + 3).$$

Notice that this is the number of processors needed to perform the first CDGR of the Greedy algorithm.

As shown in [9], the complexity of the sequential rank-1 UQRD method when executed on the PRAM is given for the cases of unlimited and limited parallelism, respectively, by

$$T_{\mathrm{ULseq}}(g) = 2(2^g - 1)$$

and

$$T_{\mathrm{Lseq}}(g) = \sum_{i=1}^{2^{g-1}} ((2^g + 2 + i) + (2^{g+1} + 1 - i)) = 3(2^{2g} - 1).$$

In the unlimited parallelism case the sequential algorithm requires $2^{g+1} + 1$ processors in order to apply in parallel the rotations to all affected pairs of elements. Omitting additive constants it follows that

$$T_{\mathrm{ULgr}}(g)/T_{\mathrm{ULseq}}(g) \simeq 5/8.$$

Thus, the Greedy algorithm requires approximately 5/8 of the number of steps performed by the serial Givens rank-1 algorithm under the assumption of unlimited parallelism.

Consider now the complexity of the Greedy algorithm in the case of limited parallelism. The first g CDGRs which reduce z to ζe_1 and partly compute (12a) require $g(2^{g+1}+1)$ steps. The ith $(i=g+1,\ldots,2^{g-2}+2)$ CDGR of the Greedy algorithm (see line 15 of Algorithm 1) requires $(2^{g+1}-\sigma_{i-2}+1)$ steps. Therefore, the remaining $(2^{g-2}-g+2)$ CDGRs which complete the factorization (12a) have complexity

$$\sum_{i=g-1}^{2^{g-2}}(2^{g+1}-\sigma_i+1)=2^{g+1}(2^{g-2}-g+2)-S_g,$$

where

$$\begin{aligned}S_g&=\sum_{i=g-1}^{2^{g-2}}(\sigma_i-1)=\sum_{i=1}^{2^{g-2}}\sigma_i-\sum_{i=1}^{g-2}\sigma_i-(2^{g-2}-g+2)\\&=1+\sum_{i=1}^{g-2}\sum_{j=1}^{2^{i-1}}j-\sum_{i=1}^{g-2}i-(2^{g-2}-g+2)\\&=2^{g-3}(2^{g-2}-3)/3-g(g-5)/2-8/3.\end{aligned}$$

Now, the ith $(i=1,\ldots,2^{g-1})$ CDGR which annihilates the corresponding superdiagonal of $\hat{R}^{(g-1)}$ in (12b) requires $(2^{g+1}+1-i)$ steps. The remaining $2^{g-1}-1$ CDGRs of the Greedy algorithm complete the triangularization of $\hat{B}_*^{(g-1)}$ with the ith $(i=1,\ldots,2^{g-1}-1)$ CDGR having complexity $(3\times 2^{g-1}+1-i)$. From this it follows that the number of steps required to compute (12b) is given by

$$\sum_{i=1}^{2^{g-1}}(2^{g+1}+1-i)=2^{g-1}(3\times 2^g-1)-1.$$

Thus, the total complexity of the Greedy algorithm is given by

$$T_{\mathrm{Lgr}}(g)=2^{g-1}(7+2^{g+2})-2^{g-3}(2^{g-2}-3)/3+g(g-3)/2+5/3.$$

For large g it follows that

$$T_{\mathrm{Lgr}}(g)/T_{\mathrm{Lseq}}(g)=(3\times 2^{2g+1}-2^{2g-5})/(9\times 2^{2g})\simeq 2/3.$$

It is assumed that 2^{g-1} processors are available to apply simultaneously the disjoint rotations of the Greedy algorithm. Consider now the execution of the sequential Givens algorithm on the PRAM when these processors are used to apply in parallel a rotation on 2^{g-1} pairs of elements. In this case the complexity of the sequential Givens algorithm becomes:

$$\begin{aligned}\tilde{T}_{\mathrm{Lseq}}(g)&=\sum_{i=1}^{2^{g-1}}\left(\lceil(2^g+2+i)/2^{g-1}\rceil+\lceil(2^{g+1}+1-i)/2^{g-1}\rceil\right)\\&=\left(2(2^g-1)+(2^{g-1}-2)+2^g+3\right)+\left(2(2^g-1)+2^g+2^{g-1}-1\right)\\&=7\times 2^g-4\end{aligned}$$

and

$$T_{\mathrm{Lgr}}(g)/\tilde{T}_{\mathrm{Lseq}}(g) \cong 2^{(g+1)}/7.$$

Thus, in this case, the sequential algorithm outperforms the Greedy one.

Notice that the fill-in of R in (4) using the Greedy and conventional sequential rank-1 algorithms is $\sum_{i=1}^{g} 2^{2(g-i)} \simeq 2^{2g}/3$ and $(2^g - 1)$, respectively. Thus, the Greedy algorithm requires more memory locations than the sequential one.

2.2. Recursive derivation of the QRD of $\tilde{H}^{(g)}$ for $g > 3$

The recursive computation of (12a) and (12b) is considered in the context of unlimited parallelism. Substituting $g-1$ for g in $\tilde{H}^{(g)}$ gives

$$\tilde{H}^{(g)} = \left(\begin{array}{c|c} \widetilde{H}^{(g-2)}\tilde{B}^{(g-2)} & \tilde{B}_1^{(g-1)} \\ R^{(g-2)}\hat{B}^{(g-2)} & \tilde{B}_2^{(g-1)} \\ \hline R^{(g-1)} & \hat{B}^{(g-1)} \end{array}\right) \begin{array}{l} 2^{g-2} \\ 2^{g-2} \\ 2^{g-1} \end{array} \tag{14a}$$

where

$$\tilde{B}^{(g-1)} = \begin{pmatrix} \tilde{B}^{(g-1)} \\ \tilde{B}^{(g-1)} \end{pmatrix}. \tag{14b}$$

Initially the orthogonal factorizations

$$\tilde{H}^{(g-2)} = \bar{G}_A \bar{R}^{(g-2)} \tag{15}$$

and

$$\bar{G}_B^{\mathrm{T}} \left(\begin{array}{c|c|c} R^{(g-2)} & \hat{B}^{(g-2)} & \tilde{B}_2^{(g-1)} \\ \hline R^{(g-1)} & & \hat{B}^{(g-1)} \end{array}\right) = \left(\begin{array}{c|c|c} 0 & \tilde{W}^{(g-2)} & \bar{B}_2^{(g-1)} \\ \hline \hat{R}^{(g-1)} & & \check{B}_2^{(g-1)} \end{array}\right) \begin{array}{l} 2^{g-2} \\ 2^{g-1} \end{array} \tag{16}$$

are computed, where $\bar{R}^{(g-2)}$, $\tilde{W}^{(g-2)}$ and $\hat{R}^{(g-1)}$ are upper triangular. For

$$\check{B}_1^{(g-1)} = \begin{pmatrix} \bar{G}_A^{\mathrm{T}} \tilde{B}_1^{(g-1)} \\ \tilde{B}_2^{(g-1)} \end{pmatrix}$$

and

$$W^{(g-1)} = \begin{pmatrix} \bar{R}^{(g-2)} & \bar{G}_A^{\mathrm{T}} \tilde{B}^{(g-2)} \\ 0 & \tilde{W}^{(g-2)} \end{pmatrix},$$

the upper triangular factor of the QRD of $\tilde{H}^{(g)}$ is derived after computing the orthogonal factorization

$$\check{G}^{\mathrm{T}} \begin{pmatrix} W^{(g-1)} & \check{B}_1^{(g-1)} \\ \hat{R}^{(g-1)} & \check{B}_2^{(g-1)} \end{pmatrix} = \tilde{R}. \tag{17}$$

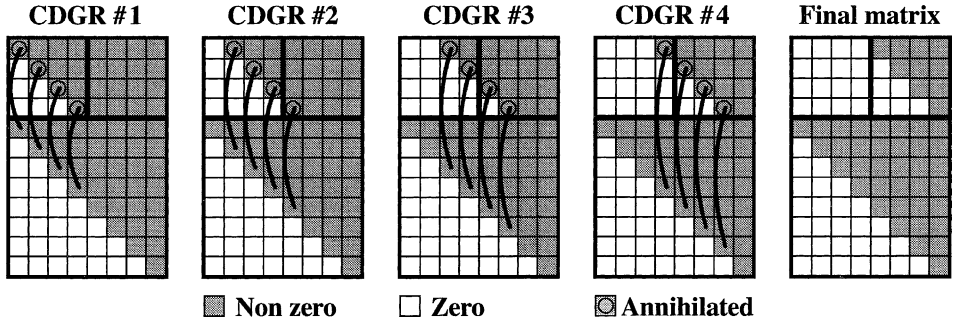

Fig. 5. Factorization of (16) using CDGRs, where $g = 4$.

As for the factorization (12a) the factorization (16) can be computed in 2^{g-2} steps. The orthogonal matrix $\bar{G}_B^{\mathrm{T}}$ is defined as the product of the CDGRs $C^{(2^{g-2},g)} \cdots C^{(1,g)}$, where $C^{(i,g)}$ $(i = 1, \ldots, 2^{g-2})$ is defined by

$$C^{(i,g)} = \prod_{j=1}^{2^{(g-2)}} G^{(p)}_{j+2^{(g-2)},p+2^{(g-1)}} \quad \text{for } p = i + j - 1.$$

Fig. 5 illustrates the factorization (16), where $g = 4$. The two full matrices $\tilde{B}_2^{(g-1)}$ and $\hat{B}^{(g-1)}$ are not shown.

The QRD (15) is derived in $T_{\mathrm{ULgr}}(g-2) = 5 \times 2^{(g-4)} + 1$ steps and the factorization (17) is computed in $2^g - 1$ steps. Factorizations (15) and (16) can start, respectively, after the third and second CDGR has been applied from the left of z. Hence, the total number of steps required to triangularize $\tilde{H}^{(g)}$ using the above method is given by

$$\begin{aligned}\tilde{T}_{\mathrm{ULrg}}(g) &= \max\left(1 + T_{\mathrm{ULgr}}(g-2), 2 + 2^{(g-2)}\right) + 2^g - 1 \\ &= 21 \times 2^{(g-4)} + 1 \quad \text{for } g > 3,\end{aligned}$$

with

$$\tilde{T}_{\mathrm{ULrg}}(g)/T_{\mathrm{ULgr}}(g) \simeq 21/20.$$

This indicates that the recursive method requires more CDGRs for solving the rank-1 UQRD problem than does Algorithm 1. The same conclusions are anticipated if the above method is used with g replaced by $g - 2^i$ $(i = 0, \ldots, \log_2 g - 1)$ in (14a) and (14b). Observe that at least $2^{(g-2)}$ steps will be required to compute simultaneously the factorizations equivalent to (15) and (16), and $2^g - 1$ steps are needed to find the final factorization which corresponds to (17). That is, the recursive triangularization $\tilde{H}^{(g)}$ will require at least the same number of steps as $T_{\mathrm{ULgr}}(g)$.

3. The greedy rank-k algorithm

The Greedy Givens annihilation scheme described in [3,10] can be used to compute the QRD of $z \in \mathbb{R}^{n \times k}$ using $g + (k-1)\log_2 g$ CDGRs, where $n = 2^g \gg k$. The case $k < g$ is considered only within the context of unlimited parallelism. The em-

ployment of the Greedy algorithm will result in $\tilde{H}$ having an almost full structure that is difficult to exploit. This can be overcome by using a variant of the Greedy algorithm called log-Greedy. At each step the log-Greedy algorithm annihilates $2^{\lfloor \log_2 \mu_i \rfloor}$ elements of the ith column by preserving previously zeroed elements, where μ_i is the maximum number of elements in column i that can possibly be annihilated. When $\lfloor \log_2 \mu_i \rfloor < \log_2 \mu_i$, the rotations are chosen so that less fill-in occurs in R. For $k = 1$ and $n = 2^g$ both algorithms are equivalent to the Greedy rank-1 algorithm. Obviously, for computing the QRD of z the log-Greedy algorithm requires more steps than the Greedy algorithm when $k > 1$, but the difference in the number of step between the two algorithms is negligible when $k < g$.

Let $\tilde{G}^{(i,q)}$ $(i \geqslant q)$ denote the CDGR applied at step i in column q of z. Let $F^{(i,q)}$ denote the fill-ins of R resulting from the application of $\tilde{G}^{(i,q)}$. All of the fill-ins of R resulting from the application of $\tilde{G}^{(1,q)}, \tilde{G}^{(2,q)}, \ldots$, are denoted by $F^{(:,q)}$. A fill-in is shaped like a parallelogram with the maximum height of the parallelograms corresponding to $F^{(:,q)}$ being $2^{(g-q)}$. Fig. 6 illustrates the log-Greedy scheme for computing the QRD of z in (7) and the structure of H, where $g = 5$ and $k = 4$. The numeral i in z and H denotes the annihilated entries and fill-ins, respectively, which have resulted from the application of the corresponding CDGR. The fill-ins $F^{(:,q)}$ $(q = 1, 2, 3, 4)$ are shown using different shades.

Notice that H and $\tilde{H} = H + R_z y^{\mathrm{T}}$ have the same structure. The retriangularization of $\tilde{H}$ can be obtained in k multi-step stages. In stage j the algorithm annihilates simultaneously the fill-ins resulting from the application of the CDGRs in column $(k - j + 1)$ of z—i.e. the fill-ins $F^{(:,k+1-j)}$. Using the Givens strategy for computing the factorization (12b) a parallelogram of height 2^p can be annihilated in $2^{(p+1)} - 1$ steps. Thus, the jth $(j = 1, \ldots, k)$ stage is completed in $2^{(g-k+j)} - 1$ steps which corresponds to the number of steps required to annihilate the biggest fill-in of $F^{(:,k+1-j)}$. After the first $k - 1$ stages the matrix $\tilde{H}$ will have the same recursive structure as $H^{(g)}$ in (10). However, the computation of factorization (12a) can start before $F^{(2,2)}$ is completely annihilated. The number of steps (i.e. CDGRs) required by the log-Greedy algorithm to solve the rank-k UQRD problem $(1 < k < g$ and $n = 2^g)$ is given approximately by

$$T_{\mathrm{logr}}(k, g) \simeq \sum_{j=1}^{k} 2^{(g-k+j)} = (1 - 2^{-k}) 2^{(g+1)},$$

where the computation of factorization (12a), the QRD of z and small constants have been ignored.

The complexity of the (unlimited parallelism) SK algorithm in [9] is given by

$$T_{\mathrm{ULsk}}(g) = 2(2^g + k - 2) \approx 2^{g+1} \quad \text{for } k \ll 2^g$$

and it follows that

$$T_{\mathrm{logr}}(k, g)/T_{\mathrm{ULsk}}(g) \simeq 1 - 2^{-k}.$$

Thus, as k increases, the efficiency of the log-Greedy algorithm decreases in relation to the SK algorithm. For $k > g$ the triangularization of z using the Greedy algorithms is inefficient since they result in $\tilde{H}$ being a full matrix that requires $O(2^{(g+1)})$

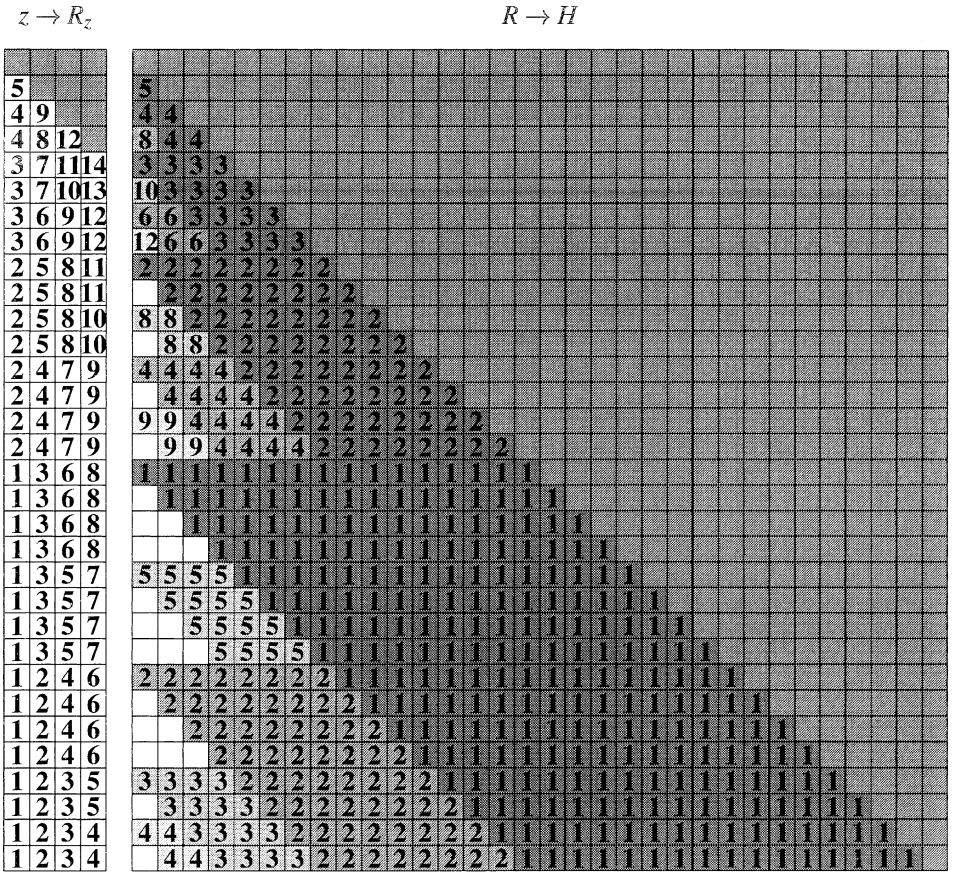

Fig. 6. The log-Greedy scheme for computing the QRD of z and the fill-ins of R, where $g = 5$ and $k = 4$.

CDGRs for triangularization. The k-times ($1 < k < g$) repetition of rank-1 Givens algorithm to solve the rank-k UQRD problem will perform approximately $5k/8$ more steps than the log-Greedy method.

Fig. 7 shows the first stage in triangularizing $\tilde{H}$ and the whole process for solving the rank-2 UQRD problem, where $g = 4$. The non-empty $F^{(i,2)}$ $(i = 2, 4, 6)$ is distinguished by bold frames and a numeral i denotes the entries annhiliated by the ith CDGR. Note that the triangularization of $\tilde{H}$ starts before the QRD of z is completed and also that the computation which corresponds to the factorization (12a) starts at step 9.

4. Conclusions

Givens strategies have been proposed for solving the rank-k UQRD problem. The analysis of the algorithms is based on an EREW PRAM computational model. Ini-

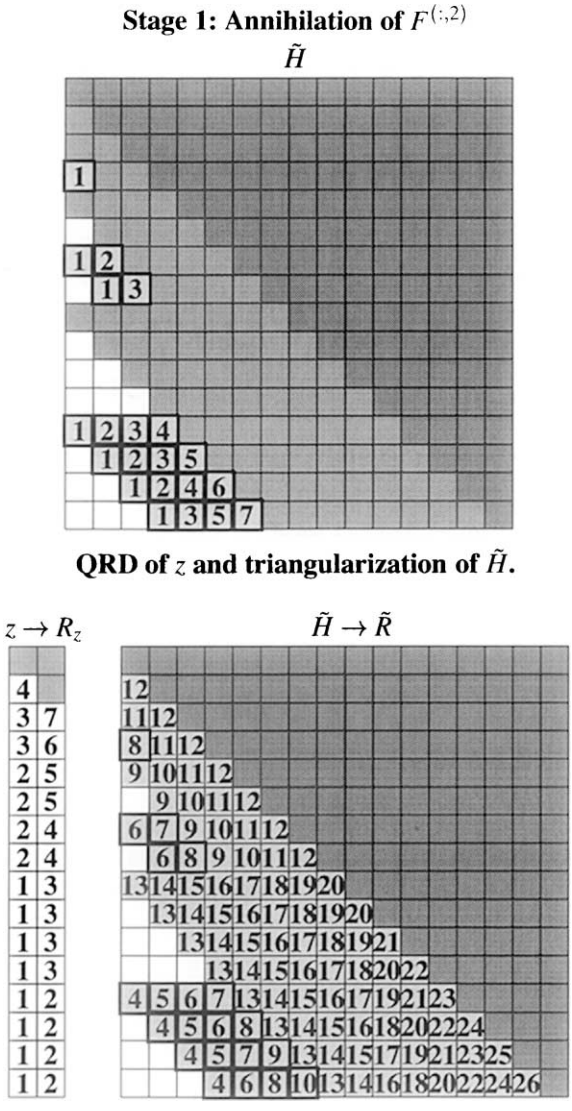

Fig. 7. Solving the rank-2 UQRD problem using the log-Greedy method, where $g = 4$.

tially the rank-1 UQRD problem was investigated. The first Greedy Givens sequence was found to require 3/8 less time units than the conventional sequential Givens method under unlimited parallelism. However, in the case of limited parallelism the sequential method was found to be the most efficient when the processors are used to perform simultaneously the Givens rotations to all of the relevant two-element vectors. A modification of the Greedy algorithm which recursively computes the QRD of (12a) and (12b) required slightly more CDGRs. A log-Greedy Givens

strategy has been described for solving the rank-k UQRD problem under unlimited parallelism. This algorithm is found to outperform an existing parallel method (the SK algorithm in [9]) for small k. In general, the efficiency of the log-Greedy method decreases as k increases.

In order that the complexity analysis of the algorithms be more realistic it should not be assumed that all of the disjoint Givens rotations can be applied simultaneously. In such circumstances the complexity analysis under limited parallelism should apply to the performance of the algorithms when implemented on a shared memory machine [1]. Furthermore, under this additional assumption, the annihilation schemes presented here will be a special case of a wider class of Givens sequences which needs to be developed [8].

The QR decomposition of $\tilde{H}$ can be obtained by a series of n Householder transformations. The transformations annihilate column-by-column the non-zero elements below the main diagonal of $\tilde{H}$ and without creating any fill-in. The efficiency of this strategy compared to that of using Givens rotations merits investigation. Currently the design of Givens sequences for computing the QRD decomposition on the PRAM under limited parallelism is considered.

Acknowledgements

The author is grateful to Maurice Clint and the two anonymous referees for their constructive comments and suggestions.

References

[1] J. Boleng, M. Misra, Load balanced parallel QR decomposition on shared memory multiprocessors, Parallel Computing 27 (10) (2001) 1321–1345.
[2] M. Cosnard, M. Daoudi, Optimal algorithms for parallel Givens factorization on a coarse-grained PRAM, Journal of the ACM 41 (2) (1994) 399–421.
[3] M. Cosnard, J.-M. Muller, Y. Robert, Parallel QR decomposition of a rectangular matrix, Numerische Mathematik 48 (1986) 239–249.
[4] P.E. Gill, G.H. Golub, W. Murray, M.A. Saunders, Methods for modifying matrix factorizations, Mathematics of Computation 28 (126) (1974) 505–535.
[5] G.H. Golub, C.F. Van Loan, Matrix computations, third ed., Johns Hopkins University Press, Baltimore, Maryland, 1996.
[6] E.J. Kontoghiorghes, Parallel strategies for computing the orthogonal factorizations used in the estimation of econometric models, Algorithmica 25 (1999) 58–74.
[7] E.J. Kontoghiorghes, Parallel algorithms for linear models: numerical methods and estimation problems, in: Advances in Computational Economics, vol. 15, Kluwer Academic Publishers, Boston, MA, 2000.
[8] E.J. Kontoghiorghes, Parallel Givens sequences for solving the general linear model on a EREW PRAM, Parallel Algorithms and Applications 15 (1–2) (2000) 57–75.
[9] E.J. Kontoghiorghes, Parallel strategies for rank-k updating of the QR decomposition, SIAM Journal on Matrix Analysis and Applications 22 (3) (2000) 714–725.
[10] J.J. Modi, M.R.B. Clarke, An alternative Givens ordering, Numerische Mathematik 43 (1984) 83–90.

[11] S.J. Olszanskyj, J.M. Lebak, A.W. Bojanczyk, Rank-k modification methods for recursive least squares problems, Numerical Algorithms 7 (1994) 325–354.
[12] A.H. Sameh, D.J. Kuck, On stable parallel linear system solvers, Journal of the ACM 25 (1) (1978) 81–91.
[13] G.M. Shroff, C.H. Bishof, Adaptive condition estimation for rank-one updates of QR factorizations, SIAM Journal on Matrix Analysis and Applications 13 (4) (1992) 1264–1278.