QR FACTORIZATIONS USING A RESTRICTED SET OF ROTATIONS

DIANNE P. O'LEARY[*] AND STEPHEN S. BULLOCK[†]

Dedicated to Alan George on the occasion of his 60th birthday

Abstract. Any matrix A of dimension $m \times n$ $(m \geq n)$ can be reduced to upper triangular form by multiplying by a sequence of $mn - n(n+1)/2$ appropriately chosen rotation matrices. In this work, we address the question of whether such a factorization exists when the set of allowed rotation planes is restricted. We introduce the rotation graph as a tool to devise elimination orderings in QR factorizations. Properties of this graph characterize sets of rotation planes that are sufficient (or sufficient under permutation) and identify rotation planes to add to complete a deficient set. We also devise a constructive way to determine all feasible rotation sequences for performing the QR factorization using a restricted set of rotation planes. We present applications to quantum circuit design and parallel QR factorization.

Key Words: QR decomposition, Givens rotations, plane rotations, parallel QR decomposition, quantum circuit design, qudits.

AMS(MOS) subject classifications: 65F25, 81R05, 65F05, 65F20, 81P68.

1. Introduction.

The QR factorization of a matrix A of dimension $m \times n$ $(m \geq n)$ is a key tool in many matrix computations, including finding a basis for the range or null space of a matrix and solving linear least squares problems. The unitary or real orthogonal matrix Q is usually computed in one of three ways: Givens rotations, Householder reflections, or Gram-Schmidt orthogonalization [7]. We focus in this paper on Givens rotations.

The Givens-based decomposition is also an essential tool in quantum computing. A quantum computation applies a unitary matrix to a vector [4], and quantum (logic) circuits are a notational shorthand for factorizations of such unitary matrices into more basic ones, corresponding to *gates*. These gates are either tensors of 2×2 unitary matrices with identity matrices (one-qubit operators) or tensors of 4×4 unitary matrices with identity matrices (two-qubit operators.) It is also common to implement Givens rotations, using circuit blocks that include up to $p-1$ controls $(m = n = 2^p)$, with the number of gates proportional to the number of controls.

A (real) Givens rotation G_{ij} is a matrix that is equal to the identity except that four entries are modified: $g_{ii} = g_{jj} = \cos\theta$ and $g_{ij} = -g_{ji} = \sin\theta$ for some angle $0 \leq \theta \leq 2\pi$. Forming $G_{ij}A$ modifies only rows i and j of a matrix, and an appropriate choice of θ forces some entry in the jth row of the product to be zero. The extension to the complex case is straightforward.

The usual Givens QR algorithm reduces A to upper triangular form by applying

[*]Mathematical and Computational Sciences Division, National Institute of Standards and Technology, Gaithersburg, MD 20899-8910; and Dept. of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (oleary@cs.umd.edu) The work of this author was supported in part by the National Science Foundation under Grant CCR-0204084.

[†]Mathematical and Computational Sciences Division, National Institute of Standards and Technology, Gaithersburg, MD 20899-8910 (stephen.bullock@nist.gov)

$mn - n(n+1)/2$ Givens rotations in the order

$$G_{12}^{(1)}, \dots G_{1m}^{(1)},\ G_{23}^{(2)}, \dots G_{2m}^{(2)}, \dots,\ G_{n,n+1}^{(n)}, \dots G_{n,m}^{(n)},$$

where $G_{ij}^{(k)}$ is used to zero out the entry in row j and column k of the product.

There are many different rotation sequences that accomplish this same goal, however. For example, for a 4×3 matrix, we could construct Q by applying the rotations indicated above

$$G_{12}^{(1)}, G_{13}^{(1)}, G_{14}^{(1)}, G_{23}^{(2)}, G_{24}^{(2)}, G_{34}^{(3)}$$

or by applying this sequence of rotations:

$$G_{34}^{(1)}, G_{23}^{(1)}, G_{12}^{(1)}, G_{34}^{(2)}, G_{23}^{(2)}, G_{34}^{(3)}.$$

Both sequences produce an upper triangular R using the minimal number of rotations.

However, performance can vary based on the choice of ordering of Givens rotations. Moreover, some rotation planes might not be available.

- Commercial computers have memory hierarchies (involving registers, cache levels, main memory, disks, etc.), so some pairs of rows are faster to access than others. Thus, in order to obtain optimal performance, the choice of rotations must depend on the layout of the matrix in memory.
- For parallel or grid computing with memory distributed among processors, the choice of rotations should minimize the amount of processor communication, and this again depends on how the matrix is distributed among processors.
- In quantum circuit design, applying a rotation for which the binary representations of $i-1$ and $j-1$ differ in a single bit can be accomplished by a single fully-controlled one-qubit rotation (a particular Givens rotation) and hence costs a small number of gates. All other rotations require a permutation of data before the rotation is applied and thus should be avoided.
- Since many atoms and ions have more than two hyperfine energy levels for the electron into which quantum data might be encoded, there is also interest in quantum multi-level logics (qudits.) However, selection rules apply to these hyperfine levels, meaning that some but not all choices of Givens rotation planes can be accessed using laser pulses.

Therefore, it is important to determine whether a given set of rotation planes $\{(i,j)\}$ can be used to reduce a matrix to upper triangular form using a minimal number of rotations. We will call such a set of rotation planes *complete*. In this paper we provide a constructive answer to the following question:

> Given a set of allowed rotation planes $\{(i,j)\}$, is there an algorithm to reduce any $m \times n$ matrix A to upper triangular form using $mn - n(n+1)/2$ rotations in those planes?

In other words, is the set of allowed rotation planes complete?

In the next section we answer this question, and then in Section 3 we illustrate the application of our result.

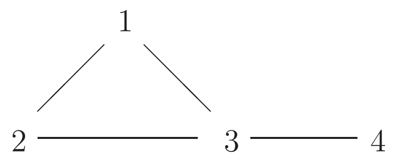

FIG. 2.1. *A sample rotation graph.*

2. Necessary and Sufficient Conditions for Completeness of a Set of Rotations. The key to the solution of our problem is the *rotation graph*, an undirected graph of m nodes with a connection between node i and node j if a rotation is allowed in plane (i, j). A sample rotation graph is given in Figure 2.1. There are 4 edges and therefore 4 allowed rotation planes.

A *feasible rotation sequence* for step k of a triangularization will be a sequence of $m - k$ allowed rotation planes that can be used to reduce the elements in rows $k + 1$ through m of column k to zero.

One other definition is useful. Consider deleting the nodes of a tree one-by-one, with the root deleted last, in any order that leaves a connected tree at each stage. From such an *admissible node ordering*, the list of edges (i, j) in the order in which node j was deleted defines an *admissible edge ordering*. For example, form a tree from the graph in Figure 2.1 by removing the edge $(2, 3)$ and rooting the graph at node 1. Then the three admissible edge orderings are

$$(3,4), (1,3), (1,2);$$
$$(3,4), (1,2), (1,3);$$
$$(1,2), (3,4), (1,3).$$

Notice that these are exactly the three feasible rotation sequences ($k = 1$) for the rotation planes corresponding to the three edges that we kept.

One interesting rotation graph is the *star graph*, with node m connected by an edge to each other node. There are $(m - 2)!$ admissible edge orderings on the resulting tree rooted at m, corresponding to feasible rotation sequences that put $m - 1$ zeros in column 1 of a matrix using rotations (m, j), $j = 2, \dots m - 1$ in any order, followed by the rotation $(1, m)$. It is easy to see that this set of $m - 1$ rotation planes is complete, since it can be used to reduce any $m \times n$ $(m \geq n)$ matrix to upper triangular form.

A second interesting rotation graph is the chain. There are constraints on the node numbering for the chain, though. If the rotation graph is the chain 1–2–3, then we can introduce zeros in the first column using the rotations (2,3) and (1,2), and then in the second column using the rotation (2,3), so these rotation planes are complete. If the rotation graph is the chain 3–1–2, then we can introduce zeros in the first column using the rotations (1,3) and (1,2) in either order. But then there are no rotations that can be used to zero the element in row 3, column 2 of the matrix, so this set of rotation planes is not complete and is not sufficient for triangularization.

These examples lead us to understand that there is a correspondence between feasible rotation sequences and admissible edge orderings of trees derived from the rotation graph. We summarize this relation in the following result.

Lemma 2.1. *Suppose that an m-node rotation graph is connected after removal of nodes 1 through $k-1$.*

a. Then there exists an admissible node ordering on the remaining $m-k$ nodes.

b. An ordering of the edges of a spanning tree of the $(m-k)$-node graph, rooted at node k, is admissible if and only if the corresponding rotation sequence at step k of the reduction of a matrix to upper triangular form is feasible.

Proof. a. Create any spanning tree rooted at k. An admissible node ordering can be created by a depth first search, deleting children before parents.

b. Suppose we have an admissible ordering of the edges of a spanning tree rooted at node k, and consider the sequence of rotations corresponding to this ordering, where we use rotation (i,j) to eliminate the element in row j of the matrix when node j of the graph is deleted. Consider the first edge (i,j) in the list. One of its nodes, say j, the one further from k, does not appear later in the list, since it is being deleted. In the matrix, we can zero the element in row j, knowing that none of our later rotations will use this row and that either $i=k$ or we will have a later opportunity to zero the element in row i, when its turn for node deletion comes. We repeat this argument on each of the $m-k$ edges in our list through the last edge, which uses rotation (k,j) to eliminate the last node other than k. Thus an admissible edge ordering induces a feasible rotation sequence.

Conversely, if we have a feasible rotation sequence, then it defines an elimination order for the nodes of the rotation graph, with k eliminated last. These rotations form a spanning tree rooted at k by directing each edge toward the node that is not eliminated. $\square$

Now we can state the criterion for determining completeness of a set of allowed rotation planes, related to the notion of reachable sets [6, p.97].

Theorem 2.2. *A set of allowed rotation planes $\{(i,j)\}$ with $1 \le i,j \le m$ is complete if and only if for every node j there exists a path in the rotation graph from node m to node j that passes through no node numbered lower than j.*

Proof. Suppose we have a node j for which such a path does not exist. When nodes $1,\ldots,j-1$ are eliminated, the remaining rotation graph will be disconnected. If we try to do elimination in column j, we will be able at best to eliminate all but one of the nonzeros in the disconnected piece, but there is no rotation that will eliminate that last nonzero. Therefore, this set of allowed rotation planes is not complete.

Suppose these paths do exist. Then at each stage j ($j=1,\ldots,n$) of the elimination, the rotation graph is connected and thus by Lemma 2.1, an admissible edge ordering and a feasible rotation sequence exists. Therefore the set of rotation planes is complete. $\square$

Corollary 2.3. *A set of allowed rotation planes $\{(i,j)\}$ with $1 \le i,j \le m$ is complete if and only if the rotation graph as well as each of the graphs formed by deleting nodes 1 through k, for $k=1,\ldots,n-1$, are connected.*

Proof. This connectedness property is necessary and sufficient for the existence of a path in the rotation graph from node m to node j that passes through no node numbered lower than j. $\square$

One obvious result is perhaps worth stating.

Corollary 2.4. *The minimal number of allowed rotation planes that can be used to reduce a general $m \times n$ matrix to upper triangular form is $m-1$.*

Proof. Since a connected graph on m nodes must have at least $m-1$ edges, the statement follows from Corollary 2.3. □

Thus the star graph and the chain graph with node numbers decreasing along each path from m give minimal sets of allowable rotation planes.

Finally, we generalize the concept of a complete set of rotations by allowing reorderings of the rows and columns of the matrix to be triangularized. We will call a set of allowed rotation planes *permutation-complete* if there exists a permutation matrix P such that any $m \times m$ matrix A can be factored as $P^T AP = (P^T QP)R$ where R is upper triangular, P is a permutation matrix, and Q is the product of at most $m(m-1)/2$ rotations in the allowed rotation planes. (Equivalently, we could require that an $m \times n$ matrix A $(m \geq n)$ can be factored into a permutation of an upper triangular matrix using $mn - n(n+1)/2$ such rotations.)

Realizing that permutations of Q correspond to renumberings of the nodes of the rotation graph, we characterize permutation-completeness in the next theorem.

Theorem 2.5. *A set of allowed rotation planes is permutation-complete if and only if the rotation graph is connected.*

Proof. Suppose that the rotation graph is connected. As in the proof of Lemma 2.1, choose a spanning tree of the graph and create an admissible node ordering by depth-first search, deleting children before parents. With this numbering of the nodes of the rotation graph, Theorem 2.2 tells us that the set of allowed rotation planes is complete.

Conversely, if the rotation graph is not connected, there is no renumbering that creates a path between nodes in the disjoint pieces, so we conclude from Theorem 2.2 that this set of allowed rotation planes is not permutation-complete. □

3. Three Examples.

3. Three Examples. We apply our algorithms to two problems in constructing the sequence of gates to implement a quantum circuit and to QR factorization of a matrix distributed among parallel processors. For ease of notation, we number the rows and columns of an $n \times n$ matrix from 0 to $n-1$, rather than from 1 to n, since this makes the graph connectivity more clear.

3.1. Designing Quantum Circuits for 3 Qubits. In quantum computing with 3 qubits, we transform a state vector of length $2^3 = 8$ by multiplication by a unitary matrix. Due to the axioms of quantum mechanics, a data-state of a three quantum-bit quantum computer is described by a vector in the Kronecker product space $\mathcal{H}_1 \otimes \mathcal{H}_1 \otimes \mathcal{H}_1$, where $\mathcal{H}_1$ is the one-qubit state space spanned by vectors (or kets) $|0\rangle$ and $|1\rangle$. The practical implication is that the most easily implemented unitary maps, which correspond to physical processes local to a single quantum-bit, are matrices of the form $V \otimes I_2 \otimes I_2$, $I_2 \otimes V \otimes I_2$, and $I_2 \otimes I_2 \otimes V$, where V is a 2×2 unitary matrix and I_ℓ is the $\ell \times \ell$ identity matrix. In order to apply an arbitrary unitary matrix, we would like to design our quantum circuits by factoring the unitary into a product of matrices drawn from this collection of tensor products, but they are

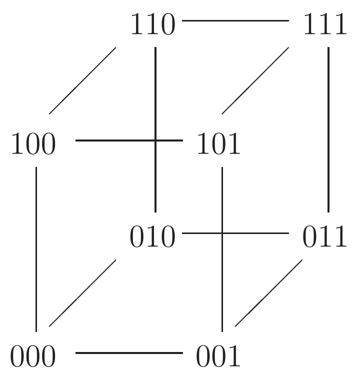

Fig. 3.1. *The 3-qubit rotation graph (Section 3.1). Also, the allowed rotation planes for a hypercube (Section 3.2)*

not sufficient. Indeed, the set of all matrices of the form $U \otimes V \otimes W$ is ten dimensional and cannot contain all 8×8 unitary matrices. However, it is possible to obtain all unitaries by interleaving two-qubit operators, tensors of the form $I_2 \otimes V$ and $V \otimes I_2$ for V varying over 4×4 unitary matrices [5].

Instead of adding two-qubit operators, we augment this collection with *controlled* operations. The simplest of these is $I_6 \oplus V$. All are equivalent to replacing certain copies of V in the tensor product matrices by identity matrices. Physically, V is applied to the target qubit if and only if the control qubits carry a fixed logical state, usually chosen to have all qubits equal to one. Controlled gates may be implemented using a number of two-qubit gates that is linear in the number of controls [4, Fig.6]. Not all Givens rotations of an eight dimensional vector space are doubly-controlled gates, but every $G^{(0)}_{jk}$ where the binary expansions of j and k differ in a single bit is.

Older papers on quantum circuit design (e.g., [4]) used the traditional choice of rotations. For example, to reduce the first column of an 8×8 matrix to upper triangular form, we apply the rotations in the order

$$G^{(0)}_{01}, G^{(0)}_{02}, G^{(0)}_{03}, G^{(0)}_{04}, G^{(0)}_{05}, G^{(0)}_{06}, G^{(0)}_{07}.$$

Only the rotations G_{01}, G_{02}, and G_{04} satisfy the constraint that the binary representations of the indices i and j differ in a single bit.

There are 12 allowed rotation planes that satisfy the constraint, corresponding to the 12 edges of the rotation graph of Figure 3.1. Since this is a connected graph on the 8 nodes, and since deleting nodes in numerical order preserves connectivity, this set of rotation planes is complete. We can derive many elimination strategies that use only the allowed rotations. For example, for $k = 1$, forming the spanning tree by breaking the edges (2,6) and (3,7) and processing highest numbered nodes first gives the rotation sequence

$$G^{(0)}_{57}, G^{(0)}_{46}, G^{(0)}_{15}, G^{(0)}_{04}, G^{(0)}_{13}, G^{(0)}_{02}, G^{(0)}_{01},$$

and this construction can be extended to hypercube rotation graphs of any size.

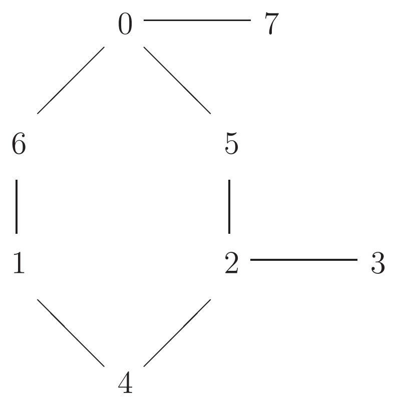

FIG. 3.2. *The qudit rotation graph.*

One such strategy has been described in [9], optimizing the use of the tensor product structure. A construction for qudits, with a state space of dimension d^p $(d > 2)$ instead of 2^p, is given in [2].

3.2. Parallel QR Factorization. If a QR factorization is to be computed on a hypercube multiprocessor, where the matrix is distributed as one row per processor, then the allowed rotation planes, the ones corresponding to communication only among neighboring processors, are exactly those corresponding to the edges of a hypercube, as illustrated in Figure 3.1 for $n = 8$ processors.

If a block of rows is distributed on each processor, then the hypercube rotations are sufficient to add to the rotations local to each processor in order to form a complete set of rotation planes.

One such elimination strategy was developed by Chu and George [3]. In addition to using only the allowed rotations, it is important to minimize the height of the spanning tree and the degree of the nodes within it, since this determines the time needed for the decomposition.

3.3. Designing Quantum Circuits for Qudits. The alkali of $^{87}\texttt{Rb}$ is being investigated for its use in quantum computing. The 8 hyperfine levels of its ground state can be used to encode quantum information as a qudit with $d = 3$ bits. (In the Dirac notation, the state-space of such a qudit is isomorphic to $\mathcal{C}^d$ with spanning kets $|0\rangle, |1\rangle, \ldots, |d-1\rangle$.) For physical reasons, only the 8 rotation planes corresponding to the edges of the graph of Figure 3.2 are allowed [8, 1].

If we are constrained to the ordering given in the graph, this set of rotation planes is not complete; for example, there is no path from 5 to 7 that does not pass through 0. Therefore, in order to do the reduction, we must, for instance, add the ability to do *swaps*, permutations of two rows in order to position them for the allowed rotation planes. These permutations are themselves (trivial) rotations. Counting permutations, 54 rotations are required to reduce a general matrix 8×8 matrix to upper triangular form.

In this application, however, ordering is not important, so the critical property is permutation-completeness. The rule (Corollary 2.3) is to delete the nodes in an order that ensures that the remaining graph at each stage is connected. So, for example, 7 must be deleted before 0, and 3 must be deleted before 2. Once a node is deleted from the cycle (0-5-2-4-1-6), then the ordering on the remaining nodes must be such that we delete from one or both ends of the resulting chain of 5 nodes.

The minimum number of rotations required is $8*7/2 = 28$. One ordering that requires only 28 allowed rotations (gates) is 7,0,6,5,3,2,4,1, and here is one minimal feasible rotation sequence:

- Step 1: Reduce column 7 to a single nonzero by the rotation sequence
$$G^{(7)}_{4,1}, G^{(7)}_{2,4}, G^{(7)}_{2,3}, G^{(7)}_{5,2}, G^{(7)}_{0,5}, G^{(7)}_{0,6}, G^{(7)}_{7,0}.$$
- Step 2: Reduce column 0 to 2 nonzeros by the rotation sequence
$$G^{(0)}_{41}, G^{(0)}_{24}, G^{(0)}_{23}, G^{(0)}_{52}, G^{(0)}_{05}, G^{(0)}_{06}.$$
- Step 3: Reduce column 6 to 3 nonzeros by the rotation sequence
$$G^{(6)}_{25}, G^{(6)}_{23}, G^{(6)}_{42}, G^{(6)}_{14}, G^{(6)}_{61}.$$
- Step 4: Reduce column 5 to 4 nonzeros by the rotation sequence
$$G^{(5)}_{41}, G^{(5)}_{24}, G^{(5)}_{23}, G^{(5)}_{52}.$$
- Step 5: Reduce column 3 to 5 nonzeros by the rotation sequence
$$G^{(3)}_{41}, G^{(3)}_{24}, G^{(3)}_{32}.$$
- Step 6: Reduce column 2 to 6 nonzeros by the rotation sequence
$$G^{(2)}_{41}, G^{(2)}_{24}.$$
- Step 7: Reduce column 4 to 7 nonzeros by the rotation sequence
$$G^{(4)}_{41}.$$

4. Conclusions.

We have developed the rotation graph as a tool to devise elimination orderings in Givens QR factorizations. Properties of this graph characterize sets of rotation planes that are complete (or permutation-complete) and identify rotation planes to add to complete an incomplete set. Through spanning trees, we also have a constructive way to determine all feasible rotation sequences for performing the QR factorization. This construction could be automated for quantum circuit design.

REFERENCES

[1] G. K. Brennen, D. P. O'Leary, and S. S. Bullock, *Criteria for exact qudit universality*, Physical Review A, (to appear). http://xxx.lanl.gov/abs/quant-ph/0407223.

[2] S. S. Bullock, D. P. O'Leary, and G. K. Brennen, *Unitary evolutions of d-level systems*, tech. report, http://xxx.lanl.gov/abs/quant-ph/0410116, October 2004.

[3] E. Chu and A. George, *QR factorization of a dense matrix on a hypercube multiprocessor*, SIAM J Sci. Stat. Comput,, 11 (1990), pp. 990–1028.

[4] G. Cybenko, *Reducing quantum computations to elementary unitary operations*, Computing in Science and Engineering, 3 (March/April 2001), pp. 27–32.

[5] D. P. DiVincenzo, *Two-bit gates are universal for quantum computation*, Phys. Rev. A, 51 (1995), p. 1015.

[6] A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[7] G. H. Golub and C. F. V. Loan, *Matrix Computations*, Johns Hopkins Univ., Baltimore, Maryland, third ed., 1996.

[8] C. Law and J. Eberly, *Synthesis of arbitrary superposition of Zeeman states in an atom*, Optics Express, 2 (1998), pp. 368–371. http://www.opticsexpress.org/abstract.cfm?URI=OPEX-2-9-368.

[9] J. J. Vartiainen, M. Mottonen, and M. M. Salomaa, *Efficient decomposition of quantum gates*, http://www.arxiv.org/abs/quant-ph/0312218, (2003).