Spectral Decomposition via Systolic Triarray Based on QR Iteration

K.J.R. Liu and K. Yao

Electrical Engineering Department
University of California, Los Angeles
Los Angeles, CA 90024-1594

ABSTRACT

In this paper, we propose a multi-phase systolic algorithm to solve the spectral decomposition problem based on QR algorithm. The spectral decomposition constitutes one of the most computationally intensive needs of modern signal processing applications. While the QR algorithm is well known to be an effective method to solve the eigenvalue problem, there is still no single systolic array architecture that can compute the unitary Q matrix readily and perform the QR algorithm efficiently. Previous methods based on the Jacobi-like approach required global communication or broadcast in computing the eigenvector, and methods using the QR algorithm had communication problems among different architectures. In this paper, we show that the Q matrix can be computed easily by using a multi-phase systolic algorithm and thus the eigenvectors can also be computed without any global communication in the array. Details on these multi-phase operations of the QR algorithm as well as architectural consequences are discussed in the paper.

1 Introduction

Computing the spectral decomposition of a matrix is an important issue in many modern signal processing and system applications. The feasibility of real-time processing for sophisticated modern signal processing systems, depends crucially on efficient implementation of parallel processing of the algorithms and associated architectures needed to perform these operations. While many variations exist in the literatures for solving these matrix problems, the heart of all these iterative methods are based either on the Jacobi-Hestennes method or the QR algorithm [6, 15]. Since present VLSI technology is capable of building a multiprocessor system on a chip, many researchers have proposed different parallel processing architectures to solve eigenvalue and singular value decomposition (SVD) problems.

Luk [9], Brent [1], and Gao and Thomas [3], have used effectively the Jacobi-like method to solve these problems for either a multiprocessor system or systolic array. While the Jacobi-like method, as considered in [9], is currently known as one of the most effective parallel SVD algorithm for full dense matrices, the computations required to obtain the rotational matrices needed in this approach to obtain the singular vectors are not simple and can not be obtained without broadcast [9].

On the other hand, other researchers [8, 13] have used QR algorithm to solve the eigenvalue problems. These methods required the computation of the unitary matrix Q. However, problems exist in the concurrent computation of Q and the pipeline operation of the QR iteration. Torralba and Navarro [14] further purposed a size-independent linear array for QR iteration and Hessenberg reduction. While this approach can provide an efficient computation of one iteration of the QR iteration, it is not obvious how to pipeline the iteration.

For some system applications, the efficient computation of singular values is sufficient, while in other applications such as antenna beamformation, spectral estimation, direct finding, etc., the eigenvectors are crucially needed. This makes practical implementation of systolic arrays discussed above difficult for many applications since they either cannot compute the eigenvector or cannot obtain the eigenvector without broadcast. A system which consists of several systolic modules to compute the MUSIC algorithm has been proposed in [12]. However, communication problems among the modules and the difficulty of matching the pipeline rates and timings among different modules may posse difficulties for practical implementation.

Presently, there is no known simple efficient systolic array approach for the generation of eigenvectors. The main reason is that there is no single architecture that is capable of handling all the steps required in the algorithm such that we can pipeline the successive iterations readily. The communication cost among different architectures is high and the interface problem for an efficient data flow is demanding. In this paper, we propose a multi-phase systolic algorithm to solve the spectral decomposition problem based on the QR algorithm. A triangular systolic array is designed based on the multi-phase concept. A key feature in our method for the successfully application of the QR algorithm is that the Q matrix of the QR decomposition can be computed explicitly by multiphase operations. With the proper feedback of this Q matrix, the QR algorithm can be computed and pipelined effectively in a single systolic array. From the accumulation of those Q matrices in another array, eigenvectors and singular vectors can be computed without needing global communication inside the array.

2 Systolic Array Matrix Processing

In this section, we consider some preliminary matrix and associated systolic array operations needed in the multi-phase systolic algorithm for spectral decompositions.

A. QR Decomposition

A non-degenerate $m \times n$ rectangular matrix A can be factorized into two matrices Q and R such that $A = QR$, where Q is an $m \times m$ unitary matrix and R is an $m \times n$ upper triangular matrix. The matrix Q can be computed using sequences of Given rotations. An elementary Givens transformation has the form of

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

CH2847-2/90/0000-1017 $1.00 © 1990 IEEE

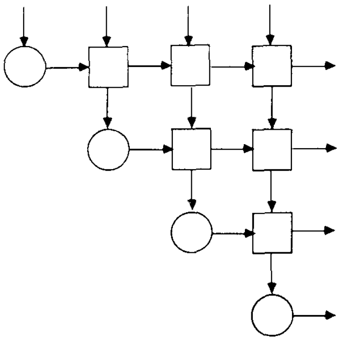


Fig.1 Triangular systolic array for QR decomposition.

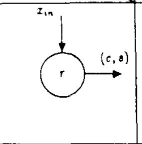
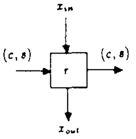
Cell \ Phase	Phase 1	Phase 2	Phase 3
	If $x_{in} = 0$ then $c \leftarrow 1$; $s \leftarrow 0$; otherwise $r' = \sqrt{\lambda^2 r^2 + x_{in}^2}$; $c \leftarrow \lambda r/r'$; $s \leftarrow x_{in}/r'$ $r \leftarrow r'$; end	$s \leftarrow x_{in}/r$	$s \leftarrow x_{in}r$
	$x_{out} \leftarrow cx_{in} - s\lambda r$ $r \leftarrow sx_{in} + c\lambda r$	$x_{out} \leftarrow x_{in} - sr$	$s_{out} \leftarrow s_{in} + x_{in}r$

Table 1 Operations of the processing cells for different phases.

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} 0\cdots0 & r_i & r_{i+1} & \cdots \\ 0\cdots0 & x_i & x_{i+1} & \cdots \end{bmatrix} = \begin{bmatrix} 0\cdots0 & r_i' & r_{i+1}' & \cdots \\ 0\cdots0 & 0 & x_{i+1}' & \cdots \end{bmatrix} \tag{1}$$

where $c = r_i/\sqrt{r_i^2 + x_i^2}$ and $s = x_i/\sqrt{r_i^2 + x_i^2}$. Several different QR arrays have been considered by Gentleman and Kung [5], Heller and Ipsen [8], and Luk [10]. In particular, the computation of the Q matrix without broadcast is difficult for the array considered in [10]. On the other hand, [5] has shown that a triangular systolic array can be used to obtain the upper triangular matrix R based on sequences of Given rotations. This systolic array is shown in Fig.1 and the operations of the cells are described in the first column of Table 1. While the rotation parameters are propagated to the right, the Q matrix will not appear directly at the right as originally suggested by [13].

B. Computation of $R^{-T}\underline{x}$

In [7], Comon and Robert presented a rectangular systolic array for the computation of $B^{-1}A$, where B and A are square and rectangular matrices respectively. For the special case where B is an upper triangular matrix denoted by R, instead of a full dense matrix, McWhirter and Shepherd [11] used the property that a triangular array can compute $R^{-T}\underline{x}$ in one phase with the matrix R prestored in the triarray. The corresponding systolic array to implement the above algorithm is the same as the one shown in Fig.1. The operations of the cells are shown in the second column of Table 1.

C. Triangular-Matrix Multiplication

The multiplication of a triangular matrix R and an rectangular full dense matrix B can be done by using the same array as in Fig.1, with R prestored in the triarray and the operations shown in the third column of Table 1, this multiplication can be easily obtained if B is inputted row by row.

3 QR Algorithm

In this section we review briefly the basic operation of the QR algorithm, in order to present our new results in Section 4. For a complex-valued nxn matrix A, it states that there is a unitary transform U such that $R = UAU^H$ is a upper triangular matrix with diagonal eigenvalues of descending order. This follows from the QR Algorithm [6, 15] where by setting $A_1 = A$, we have $A_k = Q_kR_k$ and $A_{k+1} = R_kQ_k = Q_k^HA_kQ_k$, $k = 1, \cdots$, with unitary Q_k and upper triangular R_k. Furthermore, A_k converges to the upper triangular matrix with diagonal eigenvalue elements. However, it is not obvious how to compute the eigenvectors from those Q_k and R_k we have calculated. With similar derivations as in [15], here we shows how to obtain the eigenvector associated with the largest eigenvalue from cumulative multiplications of Q_k. From the above discussions, we have

$$A_{k+1} = Q_k^HQ_{k-1}^H \cdots Q_1^HA_1Q_1Q_2\cdots Q_k. \tag{2}$$

Define

$$\begin{aligned} \tilde{Q}_k &= \coprod_{i=1}^{k} Q_i = Q_1Q_2\cdots Q_k, \\ \tilde{R}_k &= \coprod_{i=k}^{1} R_i = R_kR_{k-1}\cdots R_1. \end{aligned} \tag{3}$$

Then we have

$$\tilde{Q}_kA_{k+1} = A_1\tilde{Q}_k. \tag{4}$$

Thus the multiplication of $\tilde{Q}_k\tilde{R}_k$ can be expressed as

$$\begin{aligned} \tilde{Q}_k\tilde{R}_k &= Q_1Q_2\cdots Q_kR_kR_{k-1}\cdots R_1 \\ &= \tilde{Q}_{k-1}A_k\tilde{R}_{k-1} = A_1\tilde{Q}_{k-1}\tilde{R}_{k-1} = A_1^k = A^k. \end{aligned} \tag{5}$$

Let the eigenvalues of A satisfy, $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. Denote the matrix eigenvectors and eigenvalues of A by X and Λ respectively. Then A^k is given by $A^k = X\Lambda^kX^{-1}$. Let the QR decomposition of X be $X = QR$ and the LU decomposition of X^{-1} be $X^{-1} = LU$, where L is an unit-lower triangular matrix. Then

$$A^k = QR\Lambda^kLU = QR(\Lambda^kL\Lambda^{-k})\Lambda^kU, \tag{6}$$

where

$$\Lambda^kL\Lambda^{-k} = I + E_k, \tag{7}$$

and

$$(E_k)_{ij} = \begin{cases} l_{ij}(\lambda_i/\lambda_j)^k, & i > j, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Therefore, when k is large enough, we have $\lim_{k\to\infty} E_k = 0$ and thus $\Lambda^kL\Lambda^{-k}$ approachs the identity matrix. Then (6) can be rewritten as $A^k \to QR\Lambda^kU$. Since the term $R\Lambda^kU$ is an upper triangular matrix, comparing to (5) we can see that $\tilde{Q}_k \to Q$ when k is large. That is, the Q matrix of the QR decomposition of A^k approachs to that of the Q matrix of the QR decomposition of the matrix of eigenvector X. Define $\tilde{Q}_k = [\underline{\tilde{q}}_1, \underline{\tilde{q}}_2, \cdots, \underline{\tilde{q}}_n]$, $X = [\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_n]$, and r_{ij} as the (i,j) element of R. From $\tilde{Q}_k \to X$, we find $r_{11}\underline{\tilde{q}}_1 \to \underline{x}_1$ when k is large. Since $\underline{x}_1$ is the eigenvector associated with the largest eigenvalue, we conclude that the first column of the matrix $\tilde{Q}_k$ approach the eigenvector associated

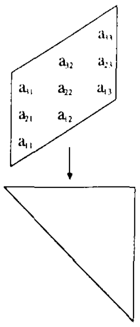


Fig.2 Phase 1: The QR decomposition.

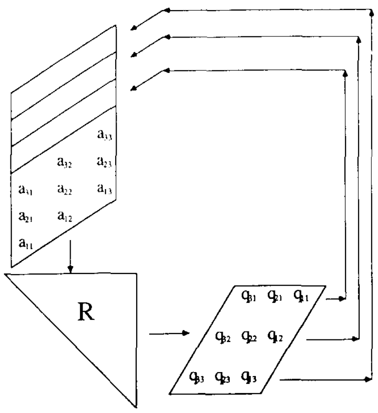


Fig.3 Phase 2: Computing the Q matrix.

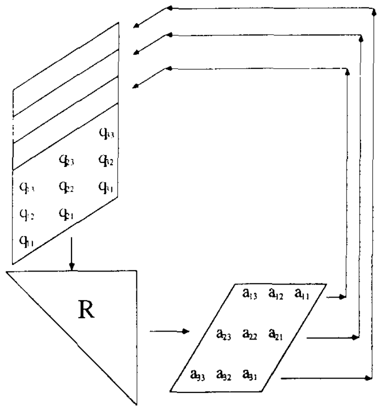


Fig.4 Phase 3: Computing the matrix product RQ.

with the largest eigenvalue of matrix A when k is large. If the matrix A is symmetric, which is often the case for many signal processing applications, the similar transformation $A_{k+1} = \tilde{Q}_k^H A \tilde{Q}_k$ is also symmetric. Since A_{k+1} approaches the upper triangular matrix by the QR algorithm, A_{k+1} approaches a diagonal matrix. That is $A_k \rightarrow \Lambda = R$ and $\tilde{Q}_k \Lambda \rightarrow X$. In this case, for large k, the columns of $\tilde{Q}_k$ become proportional to the columns of eigenvector in X.

If A is real, then A_k will converge to a real block upper triangular matrix with 1×1 and 2×2 main diagonal blocks. The complex conjugate pairs of eigenvectors of the 2×2 blocks can be solved easily using the quadratic formula. When A is not a square matrix, the singular values and vectors are of interest. For a $m \times n$ matrix B, where $m > n$, the SVD of B shows $B = U \Sigma V^T$, where U is a $m \times n$ matrix of orthogonal columns, V is a $n \times n$ unitary matrix, and Σ is a $n \times n$ diagonal matrix with diagonal elements called singular values given in descending order. For most situations where high condition numbers are not encountered, a simple symmetric $n \times n$ matrix $C = B^T B$ can be formed and the matrix V can be found by direct use of the QR algorithm. Similarly, U can be found by using $D = BB^T$.

4 Multi-phase Triangular Systolic Array

In this section, we introduced the multi-phase systolic algorithm to compute the QR algorithm. We shall show that our methods compute the Q matrix explicitly without requiring any global communication. Before we consider the multi-phase algorithm, a communication switch is first discussed. A *circular multiplexer* is a device which takes its inputs and distributes them in different output positions. We use a skewed row to represent the circular multiplexier. The computation of a QR algorithm consists of two basic steps. Initially, set $A_1 = A$. (1) for $k = 1, 2, \cdots$, compute $A_k = Q_k R_k$; (2) compute $A_{k+1} = R_k Q_k$, stop if converge, otherwise go back to step (*1*).

The QR Decomposition triarray proposed by Gentleman and Kung [5] is used in our approach. The R matrix is stored in the triarray after the computation. To compute the matrix A_{k+1} in step (2), the Q_k matrix has to be computed first. Let us call the computations in step (1) and step (2) an iteration. Several iterations are required for A_k to converge. For each iteration, we propose a three phase operation on a triarray as follows:

- **Phase 1:** QR decomposition for A_k
  Compute the QR decomposition of the matrix $A_k = Q_k R_k$, with the upper triangular matrix R_k being stored in the triarray [5]. The data in A_k is inputted row by row skewed in time as shown in Fig.2.

- **Phase 2:** Computing the Q_k matrix
  From the QR decomposition, we have $R_k^{-T} A_k^T = Q_k^T$. Let the i^{th} column of matrices A_k^T and Q_k^T be denoted by $\underline{a}_i$ and $\underline{q}_i$ respectively. Then

$$R_k^{-T}[\underline{a}_1, \underline{a}_2, \cdots, \underline{a}_n] = [\underline{q}_1, \underline{q}_2, \cdots, \underline{q}_n]. \qquad (9)$$

  Section 2 showed that $R_k^{-T}\underline{x}$ can be computed in a triarray same as the one used in Phase 1. Since the i^{th} column of A_k^T is the i^{th} row of A_k, then with A_k inputted row by row skewed in time as shown in Fig.3, the operations of the processing cells are given in the second column of Table 1. The triarray computes the Q_k matrix of A_k. The matrix Q_k is then outputted row by row as shown in Fig.3. In order to start Phase 3, the matrix Q_k has to be in the form of Fig.4. Observe that the output Q_k of phase 2 shares the same snap-shot order as the desired arrangement of Q_k in Phase 3 after a transpose operation. A circular multiplexer is used to distribute each column output of Q_k into row input as indicated in Fig.3.

- **Phase 3:** Computing $R_k Q_k$
  With the operations of the processing cell as shown in the third column of Table 1 and the Q_k obtained in Phase 2, Fig.4 shows the computation of $A_{k+1} = R_k Q_k$ in the triarray. Then the matrix A_{k+1} comes out column by column from the right side of the triarray. Again, we observe that A_{k+1} shares the same snap-shot order as the desired arrangement of A_k in Phase 1 after a transpose operation. If not convergent, a new iteration is repeated by feeding back A_{k+1} into the triarray after using a circular multiplexer as shown in Fig.4. Then Phase 1 operation begins as in Fig.2.

An attractive property of this multi-phase operation is that the feedback requirement of the matrices in different phases are identical. Thus, a circular multiplexer is enough for each row outside the array. Observe that the first column of the matrix input in Phase 2 and Phase 3 finishes at time n and the next phase can be started at time $n+1$ for this column. We find the first data output at the right hand side of the triarray is at time $n+1$, after passing through the circular multiplexer, can be piped into the array for the next phase computation without suffering any delay. If we assume the multiplexer is ideal such the delay in the multiplexer can be ignored, it takes $3n+(2n-1)=5n-1$ system clocks for one iteration. The $(2n-1)$ term represents the initial time to feed the data into the array. Suppose the number of iteration required for convergence is S, then the total number of system clocks needed is $3Sn+(2n-1)$. Thus, the converge rate of this algorithm is of the order of $O((3S+2)n)$. After the convergence of the A_k matrix, those values on the boundary cell are the eigenvalues of the A matrix.

To compute an eigenvector, a matrix multiplication systolic array can be incorporated with the multi-phase array such that those matrices $Q_1,\cdots,Q_k$ are cumulated to form the $\tilde{Q}_k$ matrix. Noted that $\tilde{Q}_k=\tilde{Q}_{k-1}Q_k$ and the matrix $\tilde{Q}_{k-1}$ is available at the start of the k^{th} iteration, while the matrix Q_k coming out at Phase 2 operation of the k^{th} iteration. Then $\tilde{Q}_k$ is obtained by multiplying $\tilde{Q}_{k-1}$ and Q_k as shown in Fig.4. As discussed in section 3, for a symmetric A matrix, when A_k converged, $\tilde{Q}_k$ yields the matrix of eigenvectors. For a non-symmetric A matrix, the first column of $\tilde{Q}_k$ yields the eigenvector associated with the largest eigenvalue.

5 Performance Efficiency

Although there are three phases of operations, the arithematical operations in Phase 2 and 3 form a subset of the operations executed in Phase 1. Therefore we do not increase the cell complexity in the multi-phase array. Due to the computation of R^{-T} in Phase 2 of the operations, it may be numerical unstable for certain highly ill-conditioning data.

Similar to Luk in [9], by convergence we mean that the parameter $off(A_k)$ defined as

$$off(A_k)=\frac{\sum_{i<j}a^2_{ij(k)}}{N}, \tag{10}$$

where N is the number of off-diagonal elements, has fallen below some prechosen tolerance value. As indicated in [9], it is difficult to monitor $off(A_k)$ in the parallel computation. Luk then proposed that the iteration be stopped after a sufficiently large number S of iterations. In the studies of Brent and Luk [1, 9], they found that $S\leq 9$ for random symmetric matrices of order $n\leq 230$ and $S\leq 6$ for $n\leq 24$. Therefore, they choosed $S=10$ for $n\leq 100$ for Jacobi-like method. Similar to their approach, we apply the QR algorithm to random nxn symmetric matrices (a_{ij}), where the elements a_{ij} for $1\leq i\leq j\leq n$ were uniformly and independently distributed in $[-1,1]$. The tolerance to meet the stopping condition is $off(A_k)\leq 10^{-10}$. From our simulations, the number of iterations for a QR algorithm to converge is in the order of 10 for matrix size smaller than 20×20. Even though we can reduce the matrix to Hessenberg form for full dense matrix or tridiagonal form for symmetric matrix, and the QR iteration with origin shift can accelerated the convergent rate, the number of iteration is still in the order of 10.

This kind of property is not desirable for parallel processing implementation. It is known that Jacobi-like method may require more flops as compared to the symmetric QR algorithm. However, due to parallel implementation, many rotations may take place at the same time. The computations involved in QR algorithm and Jacobi-like method are generally of the same complexity. From these discussions, the one which requires less number of iterations is more attractive from the parallel implementational point of view. Furthermore, the convergence rate of an QR iteration depends on the ratio of the eigenvalues. In our simulations, in more than 10% of the cases, the randomly generated symmetric matrices required significantly larger number of iterations to converge. This is also an undesirable intrinsic property of the QR algorithm for parallel implementation as compared to that of the Jacobi-like method.

6 Conclusions

The multi-phase systolic algorithm proposed in this paper can be used efficiently to solve the eigenvalue and SVD problems based on the QR algorithm. In particular, the eigenvectors can be obtained without global communication using the multi-phase operations. We showed that the QR algorithm can achieve a parallel implementation on a single architecture. We will show a numerically stable multi-phase rectangular array for the QR algorithm in a future publication. Since the operations in each phase belongs to the same types of computation, the cell complexity is thus not increased by multi-phase operations. Each iteration takes $O(n)$ time unit while the time required for convergence is $O(Sn)$, where S is the number of iterations. While we have demonstrated the advantage of the QR algorithm that can yield two multi-phase systolic algorithm implementable on single architectures without requiring global connections, the intrinsic convergence rate for the QR algorithm make it less attractive as compared to the Jacobi-like method in parallel implementation.

References

[1] R.P. Brent and F.T. Luk, "The solution of of singular-value singular-value and symmetric eigenvalue problems problems problems on multiprocessor array," SIAM J. Sci. Stat. Comput., Vol 6, pp. 69-84, Jan. 1985.

[2] P. Comon and Y. Robert, "A systolic array for computing BA^{-1}", IEEE Trans. ASSP, ASSP-35, pp.717-723, June, 1987.

[3] G.R. Gao and S.J. Thomas, "An optimal parallel Jacobi-like solution method for singular value decomposition", Proc. Int'l Conf. Parallel Processing, pp. 47-53, 1988.

[4] G.D. de Villiers, "A Gnetleman-Kung architecture for finding the singular value of a matrix", Proc. Int'l Conf. Systolic Array, pp.545-554, Ireland, 1989.

[5] W.M. Gentleman and H.T. Kung, "Matrix triangularization by systolic array," Proc. SPIE, Vol. 298, pp. 298, 1981.

[6] G.H. Golub and C.F. Van Loan, **Matrix Computation,** Johns Hopkins, 1983.

[7] S. Haykin, **Adaptive Filter Theory**, Prentice-Hall, 1986.

[8] D.E. Heller and I.C.F. Ipsen, "Systolic networks for orthogonal decomposition", SIAM J. Sci. Stat. Comput. pp.261-269, June, 1983.

[9] F.T. Luk, "A triangular processor array for computing singular value," Linear Algebra and Its Applications, pp. 259-273, 1986.

[10] F.T. Luk, "A rotation method for computing the QR-decomposition", SIAM J. Sci. Stat. Comput. pp.452-459, Apr. 1986.

[11] J.G. McWhirter and T.J. Shepherd, "An efficient systolic array for MVDR beamforming," Proc. Int'l Conf. Systolic Array, 1988.

[12] W. Robertson and W. Phillips, "A systolic MUSIC system for VLSI implementation", Proc. IEEE ICASSP, pp.2577-2580, 1989.

[13] R. Schreiber, "Systolic array for eigenvalue computation", Proc. SPIE Vol 341, Real Time Signal Processing V, 1982.

[14] N. Torralba and J.J. Navarro, "Size-independent systolic algorithms for QR iteration and Hessenberg reduction", Proc. Int'l Conf. Systolic Array, pp.166-175, 1989.

[15] J.H. Wilkinson, **Algebraic Eigenvalue Problem**, Oxford University Press, 1965.