N-64-CR
217892
19P

Bidiagonalization and Symmetric Tridiagonalization by Systolic Arrays

Robert Schreiber

October, 1988

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 88.30

NASA Cooperative Agreement Number NCC 2-387

RIACS

Research Institute for Advanced Computer Science

Bidiagonalization and Symmetric Tridiagonalization by Systolic Arrays

Robert Schreiber

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS Technical Report 88.30
October, 1988

Abstract

We give a systolic algorithm and array for bidiagonalization of an $n \times n$ matrix in $O(n \log_2 n)$ time, using $O(n^2)$ cells. Bandedness of the input matrix may be effectively exploited. If the matrix is banded, with p nonzero subdiagonals and q nonzero superdiagonals, then $4n \ln(p+q) + O(n)$ clocks and $2n(p+q) + O((p+q)^2 + n)$ cells are needed. This is faster than the best previously reported result by the factor $\log_2 e = 1.44\cdots$. Moreover, in contrast to earlier systolic designs, which require the matrix to be preloaded into the array and the result matrix extracted after bidiagonalization, the present arrays are pipelined.

Work reported herein was supported in part by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA) and partially supported by Office of Naval Research (ONR) under Contract N00014-86-K-0610.

Bidiagonalization and Symmetric Tridiagonalization by Systolic Arrays*

Robert Schreiber[†]

November 29, 1988

Abstract

We give a systolic algorithm and array for bidiagonalization of an $n \times n$ matrix in $O(n \log_2 n)$ time, using $O(n^2)$ cells. Bandedness of the input matrix may be effectively exploited. If the matrix is banded, with p nonzero subdiagonals and q nonzero superdiagonals, then $4n \ln(p+q) + O(n)$ clocks and $2n(p+q) + O((p+q)^2 + n)$ cells are needed. This is faster than the best previously reported result by the factor $\log_2 e = 1.44\cdots$. Moreover, in contrast to earlier systolic designs, which require the matrix to be preloaded into the array and the result matrix extracted after bidiagonalization, the present arrays are pipelined.

1 Introduction

In this paper we present a group of new algorithms, and their implementation by systolic arrays, for computing the factorization

$$B = PAQ \tag{1}$$

[†] Research Institute for Advanced Computer Science, NASA Ames Res. Ctr., Moffett Field, CA 94035. On leave from Rensselaer Polytechnic Institute, Troy, New York 12180-3590. This research was partially supported by Office of Naval Research under Contract N00014-86-K-0610.
* Submitted to: Journal of VLSI Signal Processing

where A is a given $m \times n$ matrix, B is an $n \times n$ upper bidiagonal matrix, Q is orthogonal and P has orthonormal rows. The method has a straightforward analog for the factorization

$$T = PAP^T \tag{2}$$

where A is a given symmetric $n \times n$ matrix, T is a symmetric tridiagonal matrix, and P is orthogonal. These factorizations are the essential first steps of important algorithms for computing the singular value decomposition and the symmetric eigendecomposition [6].

To compute these factorizations with systolic arrays is a long-standing problem. Johnsson [7] developed an implementation of the Householder method that requires $O(n^2)$ time. All later work has examined algorithms that use plane rotations. Kung and Gal-Ezar gave a tridiagonalization method that requires $O(n^2)$ time [8]. It was felt by some that this was the optimal time complexity for systolic tridiagonalization; and it was later proved by Carlsson, et.al., that any algorithm that zeros the off-tridiagonal matrix elements in a column-by-column fashion and does not introduce a nonzero into a position already zeroed must take $O(n^2)$ time on a systolic array [3]. Other algorithms, which do some extra work by allowing fill-in of positions already zeroed, have better systolic possibilities. Schreiber [12] gave a pair of arrays for tridiagonalization in $O(n^{3/2})$ time using $O(n^{3/2})$ processors. Finally, Bojanczyk and Brent [1] gave a systolic tridiagonalization method using $4n \log_2 n + O(n)$ time and n^2 processors.

Ipsen [9] presented some systolic implementations of a new bidiagonalization algorithm, without obtaining an $O(n \log n)$ implementation. This paper is largely a refinement of her ideas. We shall present new systolic designs that, like those of [1], require $O(n \log n)$ time and $O(n^2)$ hardware. But the approach is quite different from that of [1]. It is faster. Our array can reduce an upper triangular matrix with m nonzero superdiagonals to bidiagonal form in $4n \ln m$ clocks, while the Bojanczyk/Brent array requires $4n \log_2 m$, which is more by a factor of $\log_2 e \approx 1.44\cdots$. It is a pipelined design in which the matrix flows into the array and the resulting bidiagonal matrix flows out in a uniform manner, while for the Bojanczyk/Brent design the matrix must be preloaded and the result matrix extracted. Finally, an important advantage of the technique described here is that it is fairly simple to understand and describe.

After computing either of the factorizations (1) or (2), it is still necessary to use an iterative technique to find the singular values of B. Schreiber has described a systolic array method for the latter problem that provides all the singular values and requires $O(n)$ time and $O(n)$ processors [12].

In Section 2 we introduce an array for band matrix QR factorization that we use as a building block to construct arrays. The bidiagonalization array is described in Section 3.

1.1 Notation

We use upper case letters for matrices and the matching lower case letters for their elements. We say that A is (p,q)-banded if

$$j < i - p \quad \Rightarrow \quad a_{ij} = 0$$

and

$$j > i + q \quad \Rightarrow \quad a_{ij} = 0.$$

We call the pair (p,q) the half-bandwidth of A and write $\text{hbw}(A) = (p,q)$.

2 The Band-QR Array

In this section we describe a simple systolic array that we shall use as a tool, an array for QR factorization of a banded, square matrix, that was originally introduced by Heller and Ipsen [10]. Consider the processor array of Figure 2.

The array is synchronous; all processors share a common clock. One clock period ("clock") is the time needed by a cell to apply a plane rotation.

Matrix elements not explicitly shown are all zero, and plane rotations not explicitly shown are all identity rotations. Elements of the (p,q)-banded matrix A enter at the bottom of the array; element a_{ij} enters cell $j-i+p+1$ at time $i+j-2$. The array applies plane rotations to the rows of A in order to zero all elements entering cell 1 — the leftmost subdiagonal of A. Thus, the array computes a $(p-1, q+1)$ banded matrix A_1 satisfying $A_1 = PA$, where P is orthogonal. The $(ij)^{\text{th}}$ element of A_1 leaves cell $j-i+p$ at time $i+j$; thus the transit time through the array is two clocks. Note that the rightmost $(q+1)^{\text{st}}$ diagonal, which leaves from the cell at the right end of the array, must have at least $p-1$ leading zeros.

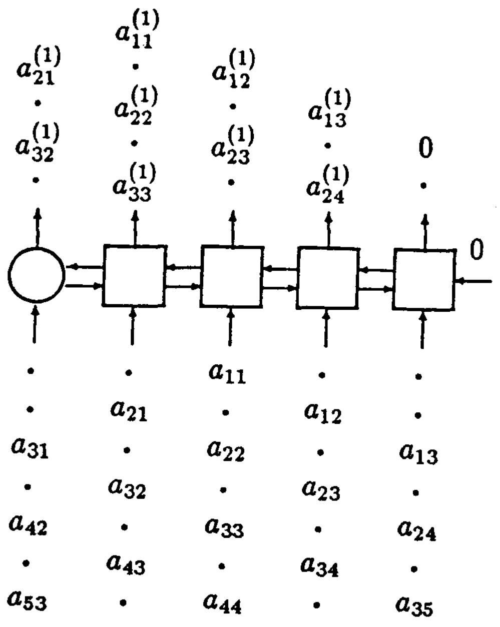


Figure 1: The band-QR array

Suppose we cascade r such arrays, where $r \le p$. The output of the last array is a matrix

$$A_r = PA \tag{3}$$

such that $\text{hbw}(A_r) = (p - r, q + r)$ and P is orthogonal. Furthermore, the diagonals that leave from the rightmost r cells all have at least $p - r$ leading zeros.

Note too that if $r \le q$, then by introducing A^T into the array we may compute

$$A_r = AQ \tag{4}$$

where $\text{hbw}(A_r) = (p + r, q - r)$, Q is orthogonal, and the leftmost r diagonals of A_r have at least $q - r$ leading zeros. It may be inconvenient to transpose a matrix; fortunately, it is equivalent to send A into an array that is the mirror image of the one in Figure 2.

Finally, suppose that each of the leftmost r subdiagonals of A has s leading zeros. Then the r new superdiagonals created in the factorization (3) must have at least $s_r \equiv s + p - r$ leading zeros. The corresponding relation holds for the factorization (4). The algorithm described in the next section is based on the use of a sequence of factorizations of types (3) and (4) that in this way introduce more and more zeros at the top of a group of diagonals.

3 Bidiagonalization by Systolic Array

We shall first describe the new bidiagonalization algorithm, then its systolic implementation. The algorithm we describe works for upper triangular, square matrices. If A is $m \times n$, then it would first be necessary to upper-triangularize A by an orthogonal transformation. After factoring

$$A = P_0^T R \tag{5}$$

where R is upper triangular and P_0 has orthonormal rows, we would bidiagonalize R, giving the factorization

$$B = P_R R Q. \tag{6}$$

Then

$$\begin{aligned} B &= P_R P_0 A Q \\ &= PAQ \end{aligned}$$

is the desired factorization (1). Systolic arrays for (5) are well-known [2,4, 10,13].

The algorithm proceeds by reducing the bandwidth of R by stages until R becomes bidiagonal. At each stage, as few as one and as many as half of the excess superdiagonals to the right of the first superdiagonal may be eliminated. Suppose $\mathrm{hbw}(R) = (0, m)$. Let

$$m = r_1 + r_2 + \cdots + r_N + 1 \tag{7}$$

where each r_k is a positive integer such that, for $k = 1, 2, \ldots, N$,

$$\begin{aligned} r_k &\leq r_{k+1} + \cdots + r_N + 1 \\ &\equiv s_k. \end{aligned}$$

At the k^{th} stage, r_k superdiagonals are eliminated and s_k remain. In all, N stages are needed. The algorithm generates a sequence $\{R_k\}$ of upper triangular matrices such that

$$R_0 = R,$$

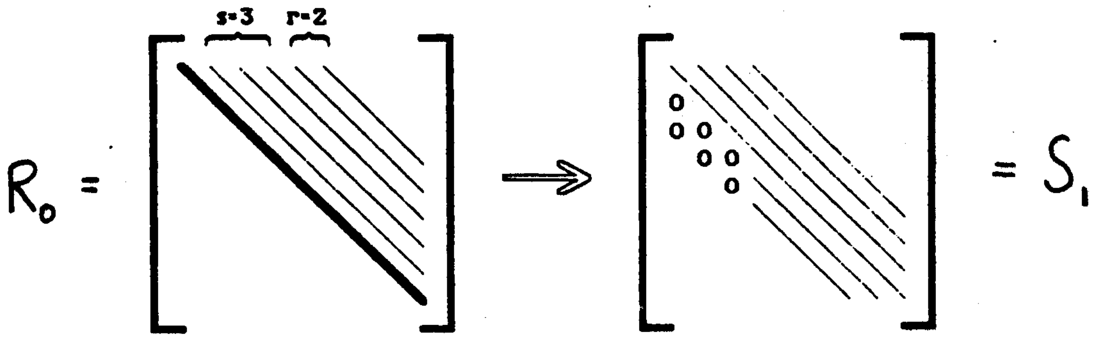
and

$$R_k = P_k R_{k-1} Q_k, \tag{8}$$

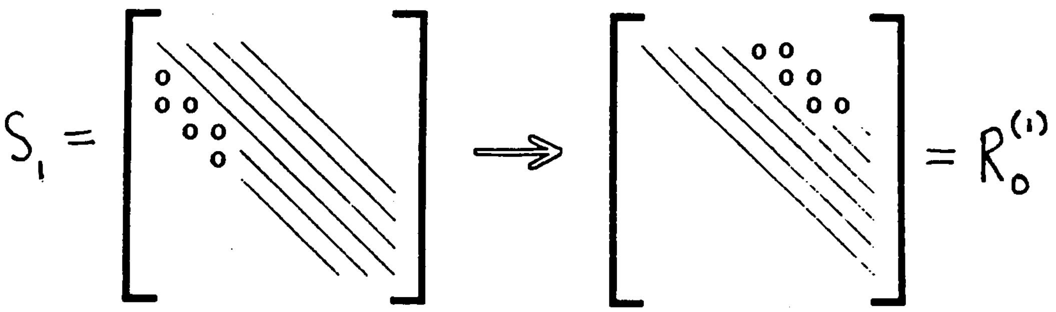
where P_k and Q_k are orthogonal and $\text{hbw}(R_k) = (0, s_k)$. Thus, $B = R_N$ is upper bidiagonal ($s_N = 1$) and satisfies (6) with $P_R = \prod_{k=N}^{1} P_k$ and $Q = \prod_{k=1}^{N} Q_k$.

We now show how diagonals of R are removed and its bandwidth reduced. The technique used was first employed by Ipsen [9]. It is easier to look at a particular case first. Suppose that $R = R_0$ has $\text{hbw}(R) = (0, 5)$, so there are four diagonals to be removed. Let $r_1 = 2$; thus $s_1 = 3$. Therefore, in the first stage, we want to remove the two outermost superdiagonals of R_0 and leave three superdiagonals, thus creating R_1 such that $\text{hbw}(R_1) = (0, 3)$. This is done as follows.

First remove the two rightmost diagonals of R_0 with a 2×6 band-QR array. Thus:



where $S_1 = R_0 V_1$. Clearly, $\text{hbw}(S_1) = (2, 3)$ and the two subdiagonals of S_1 have three leading zeros. Next, remove the two subdiagonals from S_1 with a 2×6 band-QR array. Thus:



where

$$R_0^{(1)} = U_1 S_1 = U_1 R V_1 .$$

Clearly, $\mathrm{hbw}(R_0^{(1)}) = (0,5)$, and its two outer superdiagonals have three leading zeros. Continuing, we have

$$R_0^{(j)} = U_j \cdots U_1 R V_1 \cdots V_j , \qquad j = 0, 1, \ldots$$

where $\mathrm{hbw}(R_0^{(j)}) = (0,5)$ and the outer two superdiagonals have $3j$ leading zeros. Let $R_1 = R_0^{(J)}$, where $J = \lceil (n-4)/3 \rceil$. Clearly R_1 is upper triangular and the fourth and fifth superdiagonals of R_1 have at least $3J >= n-4$ leading zeros. Thus they are entirely zero, and $\mathrm{hbw}(R_1) = (0,3)$ as we wished.

In general,

$$\begin{aligned} R_k &= R_{k-1}^{(J(k))} \\ &= U_J \cdots U_1 R_{k-1} V_1 \cdots V_J \\ &= P_k R_{k-1} Q_k \end{aligned}$$

where $J = J(k) \equiv \lceil (n-(s_k+1))/s_k \rceil$. Each matrix V_j zeros r_k superdiagonals of $R_{k-1}^{(j-1)}$ creating

$$S_j = R_{k-1}^{(j-1)} V_j, \tag{9}$$

which has r_k subdiagonals having $j s_k$ leading zeros. Next, U_j removes the r_k subdiagonals from S_j, giving

$$R_{k-1}^{(j)} = U_j S_j, \tag{10}$$

which has $j s_k$ leading zeros in its r_k outermost superdiagonals. Thus, $J(k)$ steps suffice to remove all of these superdiagonals.

The problem of bidiagonalizing a banded, upper triangular matrix has, of course, been studied before. The original work is Rutishauser's [11]. An analysis of several schemes was given by Golub, Luk, and Overton [5]. The scheme employed by Bojanczyk and Brent is a hybrid of the schemes Band Givens I and Band Givens II of [5]. An operation count for the present algorithm, summarized in Table 3, shows that it uses about twice as many operations as the best schemes discussed by Golub, Luk, and Overton. The first column of the table gives operation counts for algorithms that compute only B. The second column is for algorithms that accumulate the rotations in order to compute both P_R and Q.

Algorithm	Reduction without vectors	Reduction with vectors
systolic (present paper)	$\sim 8mn^2$	$\sim 8n^3 \ln m$
Band Givens I [5]	$\sim 4mn^2$	$\sim 4n^3$
Band Givens II [5]	$\sim 4mn^2$	$\sim 4n^3 \ln m$
Band Givens III [5]	$\sim 4mn^2$	$\sim \frac{8n^3}{3}$
Bojanczyk/Brent [1]	$\sim 4mn^2$	$\sim 3.88n^3 \ln m$

Table 1: Asymptotic multiplication counts for bidiagonalization of a matrix of order n with half-bandwidth $(0, m)$

k	r_k	s_k	$J(k)$	$t(k)$
1	4	6	3	24
2	3	3	6	36
3	1	2	9	18
4	1	1	18	36

$t(k)$ is the number of rows in the processor array; $t(k) \equiv 2r_kJ(k)$.

Table 2: Bandwidth reduction; $n = 20$, $m = 10$, first partition

3.1 The systolic bidiagonalization array

The systolic array that carries out this algorithm will now be described. The process (9) can be accomplished by an $r_k \times (1 + s_k + r_k)$ band-QR array for removing superdiagonals by column rotations, as described in Section 2. The process (10) can be accomplished by a matching $r_k \times (1 + s_k + r_k)$ band-QR array for removing subdiagonals by row rotations; the two arrays form a single $2r_k \times (1 + s_k + r_k)$ array, the output of one array becoming the input to the other. Thus, a sequence of $J(k)$ pairs of band-QR arrays can carry out one full stage in the reduction of bandwidth (8); see Figure 2.

The whole bidiagonalization process is accomplished by a sequence of such arrays. Again, to make it clear, we will look at a particular case. Suppose that $R = R_0$ has half bandwidth (0,10). The parameters of the algorithm are given in Table 2 and the array is illustrated in Figure 3.

For the same problem, a different choice of partition (7) leads to a different bidiagonalization process. An alternate choice for the problem above leads

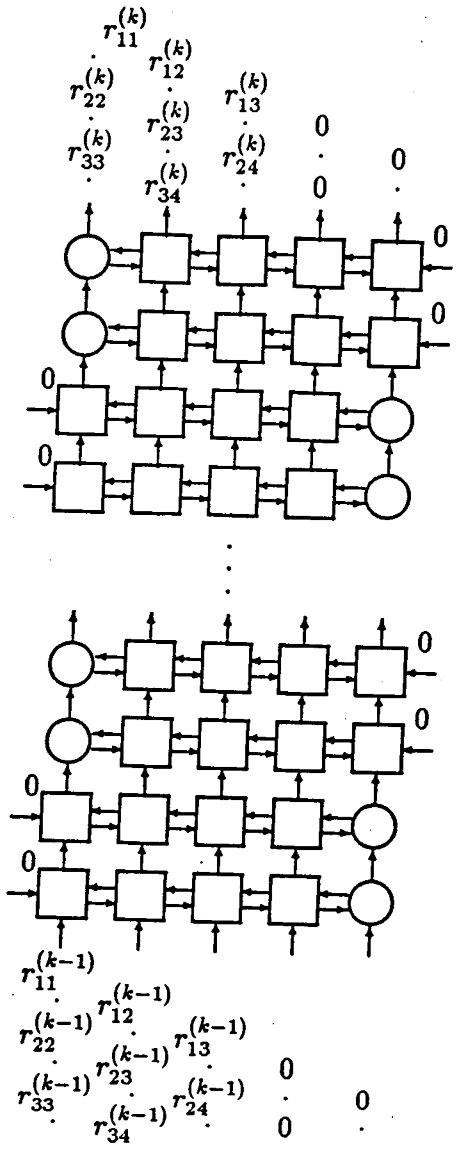


Figure 2: The bandwidth reduction array; r=2, s=3, n=10, J=2

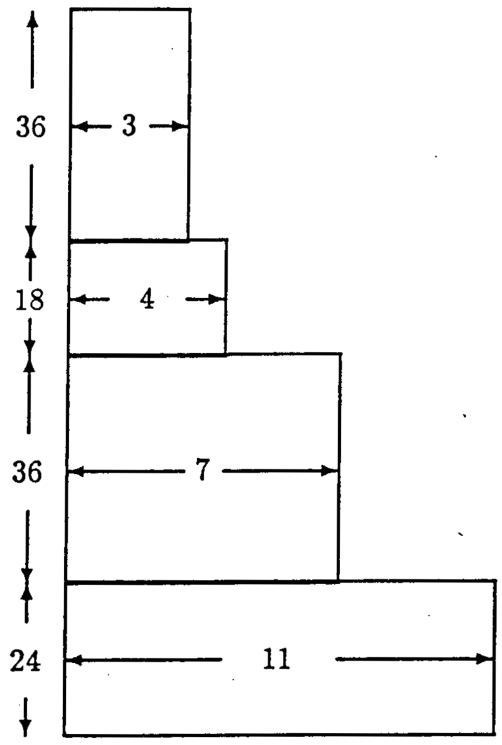


Figure 3: Bidiagonalization array for first partition

k	r_k	s_k	$J(k)$	$t(k)$
1	1	9	2	4
2	1	8	2	4
3	1	7	2	4
4	1	6	3	6
5	1	5	3	6
6	1	4	4	8
7	1	3	6	12
8	1	2	9	18
9	1	1	18	36

$t(k)$ is the number of rows in the processor array; $t(k) \equiv 2r_kJ(k)$.

Table 3: Bandwidth reduction; $n = 20$, $m = 10$, second partition

to a somewhat more efficient scheme; the parameters are given in Table 3 and an illustration in Figure 4 While 696 processing elements are needed for the array of Figure 3, only 498 processing elements are needed for the array of Figure 4. Moreover, the time required drops from 228 to 196 clocks.

Since the transit time through a band-QR array having r rows is $2r$ clocks, it follows that $4r_k$ clocks are needed for a matrix to traverse a single stage in Figure 2, that $4r_kJ(k)$ clocks are needed to traverse the $J(k)$ stages that carry out the k^{th} reduction stage (8), and therefore that the total time required to traverse the array is

$$\begin{aligned} T(n, m, \{r_k\}_{k=1}^N) &= \sum_{k=1}^N 4r_kJ(k) \\ &= \sum_{k=1}^N 4r_k \left\lceil \frac{n - s_k - 1}{s_k} \right\rceil \end{aligned}$$

To simplify the analysis we consider instead

$$\begin{aligned} T_1 = T_1(n, m, \{r_k\}_{k=1}^N) &\equiv \sum_{k=1}^N r_k \left(\frac{n - s_k - 1}{s_k} \right) \\ &\approx \frac{1}{4}T \end{aligned}$$

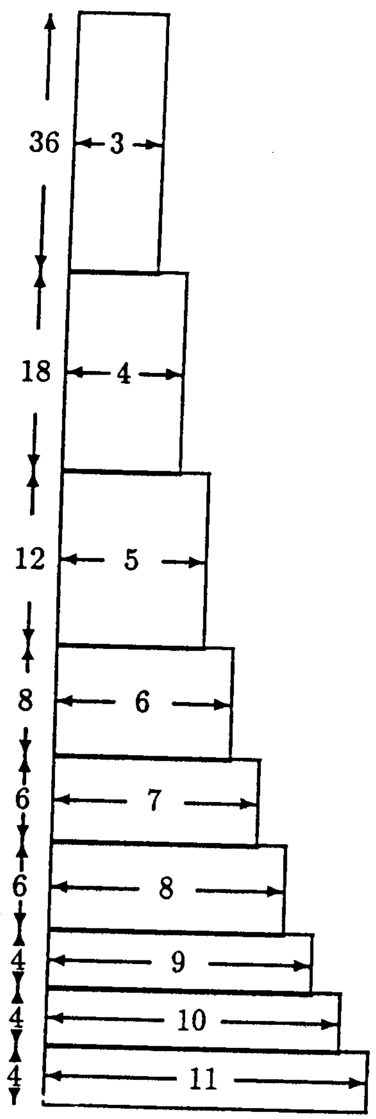


Figure 4: Bidiagonalization array for second partition

Lemma 1 *T_1 is minimized by choosing $N = m - 1$ and $r_k = 1$ for $k = 1, \cdots, N$.*

Proof: Choose any partition (7) of m. Suppose $r_k > 1$. Replace r_k by as many ones. This changes the term

$$t_k = r_k \left(\frac{n - s_k - 1}{s_k} \right) = r_k \left(\frac{n-1}{s_k} - 1 \right)$$

into the sum

$$\begin{aligned} t'_k &= \sum_{j=1}^{r_k} \left(\frac{n-1}{s_k + j - 1} - 1 \right) \\ &< t_k \end{aligned}$$

This process can continue until all $r_k = 1$.

QED

With this choice we have that

$$T \approx 4T_1 = 4\left((n-1)H_{m-1} - (m-1)\right)$$

where H_{m-1}, the $m - 1^{st}$ harmonic number, satisfies

$$H_{m-1} \approx \gamma + \ln(m-1) + \frac{1}{2(m-1)} + O(m^{-2})$$

and γ, Euler's constant, is $.577\cdots$. Thus,

$$T \approx 4n \ln m + 2.31n - 4m + o(n+m).$$

An alternative, which maximizes T, is to choose r to be about half the remaining bandwidth at each step. In that case, $N \approx \log_2 m$, and $T \approx 4(n \log_2 m - m)$. Thus the difference between the optimal and the worst choice of partition is, asymptotically, only a factor of $\log_2 e = 1.44\cdots$. By comparison, the Bojanczyk/Brent array requires $4n \log_2 m$ clocks; thus it is slower by the factor $\log_2 e$ than our best array.

The space requirement of our method is

$$P = P(n, m, \{r_k\}_{k=1}^{N}) \equiv \sum_{k=1}^{N} t(k)(s_k + r_k + 1)$$

cells. For the choice $r_k = 1, N = m - 1$ we have

$$\begin{aligned} P &\approx \sum_{k=1}^{m-1} 2\left(\frac{n - s_k - 1}{s_k}\right)(s_k + 2) \\ &= 2nm + O(n \ln m + m^2). \end{aligned}$$

Thus, about twice as many processors are needed as for the Bojanczyk/Brent array.

If the matrices P and Q are needed, then additional computation must be done to accumulate the product of all the row rotations (which move left to right in the array) that is P, and the product of the column rotation (which move right to left) that is Q. A $T \times n$ array can accumulate the row rotations. The rotations enter at the left edge of the array in the format created by a band-QR array. The $n \times n$ identity matrix enters at the bottom, one column per processor. The array applies the rotations to this matrix, and their product leaves from the top. Obviously, there is also an array of this type for accumulating column rotations.

3.2 Systolic tridiagonalization

We will briefly describe the generalization of our bidiagonalization method to the symmetric tridiagonalization problem. Let A by a symmetric $n \times n$ matrix with $\text{hbw}(A) = (m, m)$. Using an $r \times 2m + 1$ band-QR array, we compute the factorization

$$B_1 = U_1 A, \tag{11}$$

where $\text{hbw}(B_1) = (m-r, m+r)$ and the r newly created outer superdiagonals of B_1 each has $s \equiv m - r$ leading zeros. Next form the symmetric matrix

$$A^{(1)} = B_1 U_1^t. \tag{12}$$

It is straightforward to show that $A^{(1)}$ has only m nonzero subdiagonals, so by its symmetry, $\text{hbw}(A^{(1)}) = (m, m)$; furthermore, the outermost r sub-

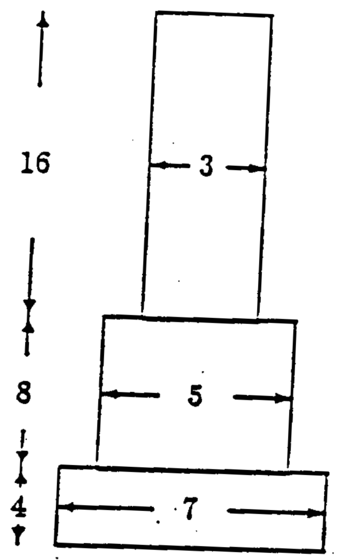


Figure 5: The tridiagonalization-step array, $m = 3$; $r = 1$; $n = 10$.

and superdiagonals have s leading zeros. Thus, we have an analog of the bidiagonalization algorithm, with the same time and space requirements. The only essential difference is that in each pass over the matrix, only one orthogonal matrix is generated (U_1 above). It is applied from both the left and the right to the input matrix. This can be accomplished by a systolic array of $2r$ rows, in which the first r rows carray out the factorization (11) and the following r rows form the product (12). To do this, plane rotations generated by cells at the left edge of the first group of rows are sent, via long connections, to the right edge of the second group of rows. An illustration of the array is given in Figure 3.2.

4 Acknowledgement

I thank Adam Bojanczyk, Ilse Ipsen and Franklin Luk for their comments on this research.

References

[1] A. Bojanczyk and R.P. Brent. Tridiagonalization of a symmetric matrix on a square array of mesh-connected processors. Journal of Parallel and Distributed Computing 2, 261-276, (1983).

[2] A. Bojanczyk, R.P Brent, and H.T. Kung. Numerically stable solution of dense systems of linear equations using mesh-connected processors. SIAM Journal on Scientific and Statistical Computing 5, 95-104, (1984).

[3] G. Carlsson, H. Sexton, M. Shensa, and C. Wright. Algebraic techniques in systolic array design. Naval Ocean Systems Center, San Diego, CA, (1983).

[4] W.M. Gentleman and H.T. Kung. Matrix triangularization by systolic array. In Tian F. Tao, editor, Real-Time Signal Processing IV, SPIE Vol. 298, 1981, pp. 19-26.

[5] Gene H. Golub, Franklin T. Luk, and Michael L. Overton. A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix. ACM Transactions on Mathematical Software 7, 149-169, (1981).

[6] Gene H. Golub and Charles F. Van Loan. Matrix Computations. Johns Hopkins, 1983.

[7] L. Johnsson. A computational array for the QR-method. In P. Penfield, editor, Proc. Conference on Advanced Research in VLSI, Artech House, 1982, pp. 123-129.

[8] S.Y. Kung and R.J. Gal-Ezar. Linear or systolic array for eigenvalue and singular value decompositions. In T. Kailath, S.Y. Kung, and H.J. Whitehouse, editors, VLSI and Modern Signal Processing, Prentice-Hall, 1984.

[9] Ipsen, I.C.F. Singular Value Computations with Systolic Arrays. Proc. SPIE Symp. 549 (Real Time Signal Processing VII), 1984, pp 13-21.

[10] Don E. Heller and Ilse C. F. Ipsen. Systolic networks for orthogonal decompositions. SIAM Journal on Scientific and Statistical Computing 4, 261-269, (1983).

[11] H. Rutishauser. On Jacobi rotation patterns. Proc. Symp. Appl. Math. 15, 219-239, (1963).

[12] Robert Schreiber. Computing generalized inverses and the eigenvalues of symmetric matrices using systolic arrays. In R. Glowinski and J.L. Lions, editors, Computing Methods in Applied Sciences and Engineering, North-Holland, 1984, pp. 285-295.

[13] Robert Schreiber. On systolic array methods for band matrix factorizations. BIT 26, 303-316, (1986).