A Reprint from the

PROCEEDINGS

Of SPIE-The International Society for Optical Engineering

Volume 298

Real-Time Signal Processing IV

August 25-28, 1981
San Diego, California

Matrix triangularization by systolic arrays

W. M. Gentleman
Department of Computer Science, University of Waterloo
Waterloo, Ontario N2L 3G1, Canada

H. T. Kung
Department of Computer Science, Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Matrix triangularization by systolic arrays

W. M. Gentleman
Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
H. T. Kung
Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213

Introduction

Given an $n \times p$ matrix X with $p \leq n$, *matrix triangularization*, or *triangularization* in short, is to determine an $n \times n$ nonsingular matrix M such that

$$MX = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

where R is $p \times p$ upper triangular, and furthermore to compute the entries in R. By triangularization, many matrix problems are reduced to the simpler problem of solving triangular linear systems (see for example, Stewart[12]). When X is a square matrix, triangularization is the major step in almost all direct methods for solving general linear systems. When M is restricted to be an orthogonal matrix Q, triangularization is also the key step in computing least squares solutions by the QR decomposition, and in computing eigenvalues by the QR algorithm. Triangularization is computationally expensive, however. Algorithms for performing it typically require n^3 operations on general $n \times n$ matrices. As a result, triangularization has become a bottleneck in some real-time applications.[11] This paper sketches unified concepts of using systolic arrays to perform real-time triangularization for both general and band matrices. (Examples and general discussions of systolic architectures can be found in other papers.[6,7]) Under the same framework systolic triangularization arrays are derived for the solution of linear systems with pivoting and for least squares computations. More detailed descriptions of the suggested systolic arrays will appear in the final version of the paper.

Triangularization for General Matrices

Basic Ideas

Consider a partially triangularized matrix as shown in Figure 1 (a). The triangularization can be carried out a step further by replacing the fourth row successively with some linear combination of itself with the first row, itself with the second row, and itself with the third row so that the resulting fourth row will have zeros in its first three entries, as shown in Figure 1 (b). Clearly this process can continue to the fifth row, the six row, and so on, until the triangularization is complete.

A triangular systolic array as depicted in Figure 2 is well suited for the execution of the triangularization process described above. The systolic array consists of two types of cells, *internal cells* (represented by squares) and *boundary cells* (represented by circles). Internal cells basically perform multiplies and adds, whereas boundary cells perform divisions or reciprocals plus possibly other operations. During the computation, X enters the systolic array from its top boundary one row after another row in a skewed order. Current entries in the i-th

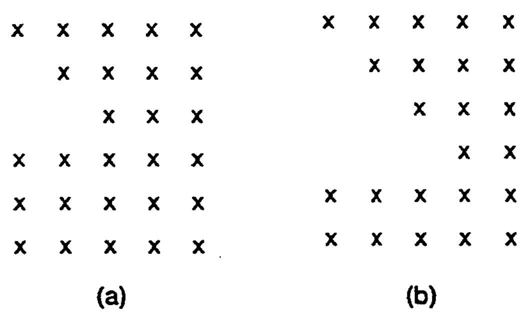


Figure 1. A step in triangularization for general matrices.

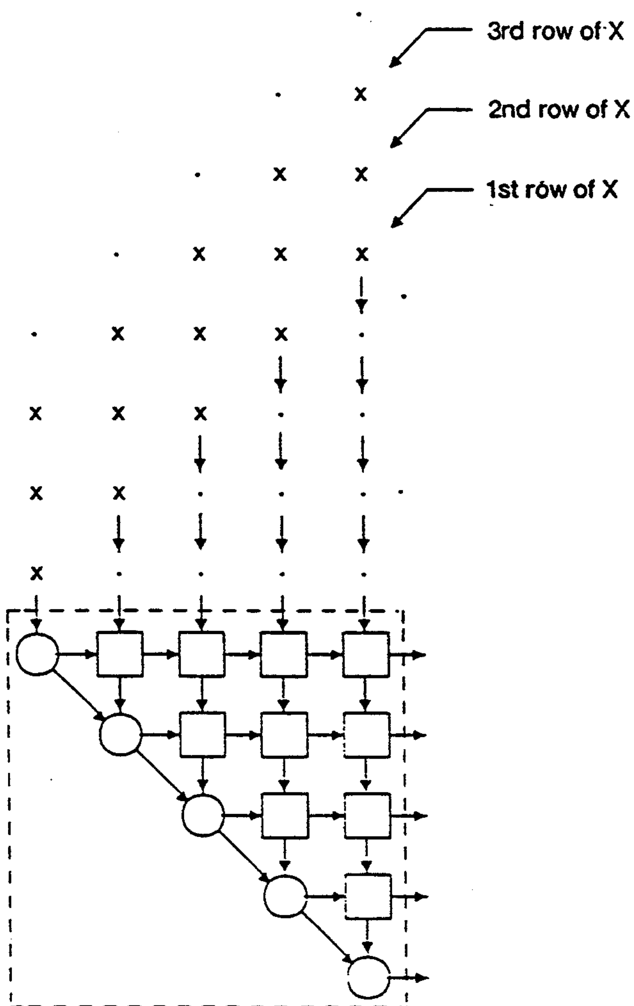


Figure 2. Abstraction of a triangular systolic array for triangularizing a general matrix.

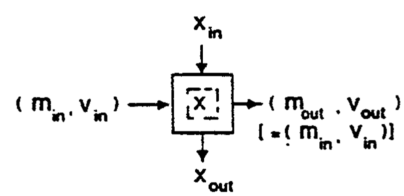
row of R are kept in the i-th row of the systolic array, one entry at each cell. Initially zeroes are stored in all cells, and when computation is complete, entries in R will be readily read out, one from each cell.

The first row of the systolic array, i.e., the top row of cells in the systolic array, turns every arriving row of X into a row with zero in its first entry, and outputs results to the second row of the systolic array. Similarly, the second row of the systolic array turns every row of X it receives into a row with zero in its second entry, and outputs results to the third row of the systolic array, and so on. While updating a row of X, a row of the systolic array may also update current entries of R that are stored in its cells. The boundary cell at the left end determines parameters needed for both updates, and they are sent to the right to be used in actual updatings taking place at internal cells. Because rows of X pass through the systolic array in the skewed order, parameters determined at the boundary cell will rearch internal cells at right times.

Triangularization with Neighbor Pivoting

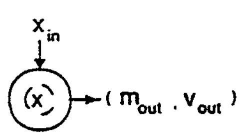
Triangularization needed in solving linear systems is classically performed by Gaussian elimination. For numerical stability, Gaussian elimination in general requires pivoting, and the usual partial or complete pivoting strategy is not suited to a systolic array since it may require global communication for pivot selection. The triangularization process outlined above suggests another pivoting strategy. This technique, called ***neighbor pivoting*** here, introduces a zero to a row by subtracting a multiple of an *adjacent* row from it, interchanging the two rows when necessary to prevent the multiple from exceeding unity. The fractional multiple suggests that triangularization with neighbor pivoting is numerically stable. Indeed numerical experiments have confirmed this.[9,10] Figure 3 specifies cells in a triangular systolic array that can perform triangularization with neighbor pivoting. At every cell cycle a boundary cell generates a multiplier m as well as a Boolean variable v, which signals a row interchange when having value one.

INTERNAL CELL:



if v_{in} **then**
  begin
    $x_{out} \leftarrow x + m_{in} \cdot x_{in}$;
    $x \leftarrow x_{in}$
  end
else
  $x_{out} \leftarrow x_{in} + m_{in} \cdot x$

BOUNDARY CELL:



if $|x_{in}| \geq |x|$ **then**
  begin
    $v_{out} \leftarrow 1$;
    $m_{out} \leftarrow$ **if** $x_{in} \neq 0$ **then** $-x/x_{in}$
      else 0;
    $x \leftarrow x_{in}$
  end
else
    $v_{out} \leftarrow 0$;
    $m_{out} \leftarrow -x_{in}/x$

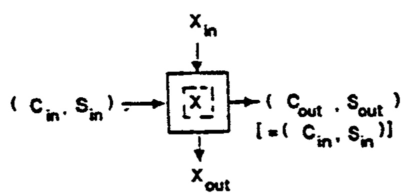
Figure 3. Cell definitions for a triangular systolic array for performing triangularization with neighbor pivoting.

The idea of performing neighbor pivoting in Gaussian elimination was known at least as early as 1960 for a totally different reason[8] -- it was used for the purpose of minimizing storage requirements rather than for avoiding global communications.

Orthogonal Triangularization

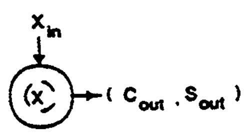
The classical procedure of performing a sequence of plane rotations (known also as Givens rotations) for orthogonal triangularization is a special case of the general triangularization process described earlier. Consequently a triangular systolic array with cells defined in Figure 4 can perform orthogonal triangularization. It is of interest to note that the same parallelism as used in the present systolic array was actually assumed in a previous error analysis of QR decompositions by Givens rotations.[4]

INTERNAL CELL:



$$x_{out} \leftarrow -s_{in}x + c_{in} \cdot x_{in}$$
$$x \leftarrow c_{in}x + s_{in} \cdot x_{in}$$

BOUNDARY CELL:



if $x_{in} = 0$ **then**
  begin
    $c_{out} \leftarrow 1$;
    $s_{out} \leftarrow 0$
  end
else
  begin
    $c_{out} \leftarrow x/\sqrt{x^2 + x_{in}^2}$;
    $s_{out} \leftarrow x_{in}/\sqrt{x^2 + x_{in}^2}$;
    $x \leftarrow \sqrt{x^2 + x_{in}^2}$
  end

Figure 4. Cell definitions for a triangular systolic array for performing orthogonal triangularization with Givens rotations.

Observe that in this case boundary cells are considerably more complex than internal cells. Boundary cells compute square roots as well as reciprocals, whereas internal cells perform only additions and multiplications. Since all the cells in the systolic array must operate at the same throughput rate, boundary cells could form a bottleneck for the overall performance. Thus it is desirable to reduce the complexity of boundary cells so that it can be close to that of internal cells. In Gentleman[2] and Hammarling[5] methods are described for performing Givens rotations without square roots. Using similar techniques, systolic arrays for performing orthogonal triangularization that involve no square roots have been devised. For example, in one of these designs a boundary cell performs five multiplications and one reciprocal at each cycle. To carry out the scaling needed for the square root free scheme, boundary cells on the diagonal of the systolic array are now required to be linearly connected, as depicted in Figure 2. Detailed cell definitions of this square root free systolic array will be given in the final version of the paper.

On-the-fly Linear Least Squares Computations

Let X be an n x p matrix X with $p \leq n$, and y an n-vector, the *linear least squares* problem is to determine a p-vector b such that $\|y - Xb\|^2$ is minimized, where the norm is the usual Euclidean norm. The use of the QR decomposition to solve the least squares problem has proven to be a successful method. Assuming X has full rank, the method consists of the following steps.

Step 1 (*Orthogonal Triangularization*). Find an n x n orthogonal matrix Q such that

$$QX = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

where R is p x p upper triangular.

Step 2 (*Solution of Triangular Linear System*). Solve

$$Rb = Q_1 y$$

for b, where Q_1 is the matrix consisting of the first p rows of Q.

Step 1 can be carried out by a triangular systolic array as described earlier. Note that $Q_1 y$ can be formed at the same time as X being orthogonally triangularized by treating y as an additional column of X in its right-hand side. By a result in Kung and Leiserson[7], step 2 can be realized with a linear systolic array. Chaining the two systolic arrays together, as shown in Figure 5, forms a powerful system capable of producing on-the-fly the least squares fit to all the data that have arrived up to any given moment. Since Givens transformations without square roots actually solve the *weighted* linear least square problem,[2,3] exponential decays or other appropriate weights are readily incorporated in the system.

It should be obvious that a similar system can be formed for solving general linear systems using neighbor pivoting or orthogonal triangularization. For this case the former costs about half as much as the latter in terms of the required hardware.

A Remark

In Bojanczk, Brent and Kung[1] a different systolic array for performing the orthogonal triangularization for square matrices is described. That scheme was designed for providing a numerically stable solution for solving linear systems and the convenience of performing the QR algorithm in finding eigenvalues, as opposed to solving the least squares problem. With that scheme, when computation is complete factors of Q rather than entries of R are stored inside the systolic array.

Triangularization for Band Matrices

When triangularization is to be done on a band matrix, it is possible to organize the systolic array so that its size depends on the band width of the matrix rather than on the order of the matrix. Figure 6 illustrates the general idea of how to construct such a systolic array. As in the preceding section, by defining the two kinds of cells appropriately the systolic array can perform triangularization with neighbor pivoting or orthogonal triangularization.

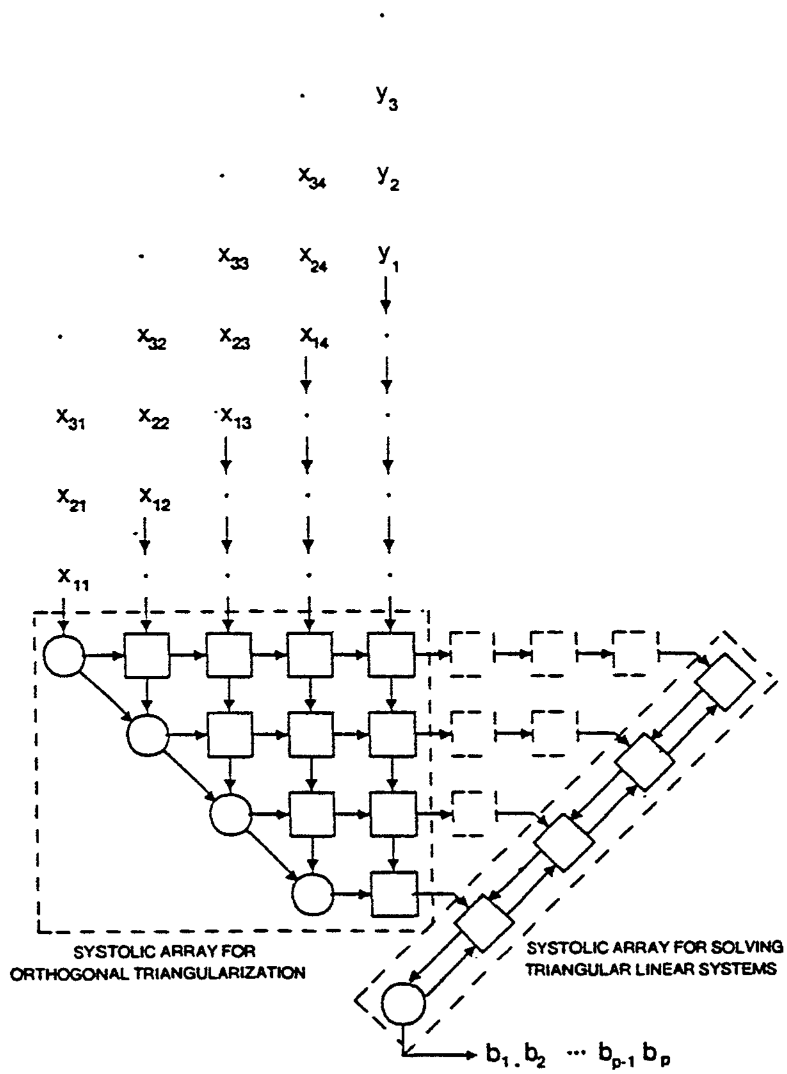


Figure 5. On-the-fly least squares solutions using systolic arrays.

Concluding Remarks

Being able to perform neighbor pivoting and Givens rotations by systolic arrays appears to be an important realization. Because of this matrix triangularization or other similar computations such as reduction of a general matrix to its Hessenberg form can all be handled with relatively simple special-purpose modules. Results of this paper and other papers[1,7] suggest that a few types of simple VLSI chips can used to configure a large variety of systolic arrays for real-time matrix computations. These linear algebra chips should be built in the near future.

Acknowledgments

Part of this research was carried out while H.T. Kung was on leave from CMU with ESL Inc., a subsidiary of TRW, during January - August, 1981.

$$\begin{bmatrix}
x_{11} & x_{12} & x_{13} & & & & & & & & \\
x_{21} & x_{22} & x_{23} & x_{24} & & & & & & & \\
x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & & & & & & \\
x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & & & & & \\
 & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} & & & & \\
 & & x_{63} & x_{64} & x_{65} & x_{66} & x_{67} & x_{68} & & & \\
 & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & \\
 & & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
 & & & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{bmatrix}$$

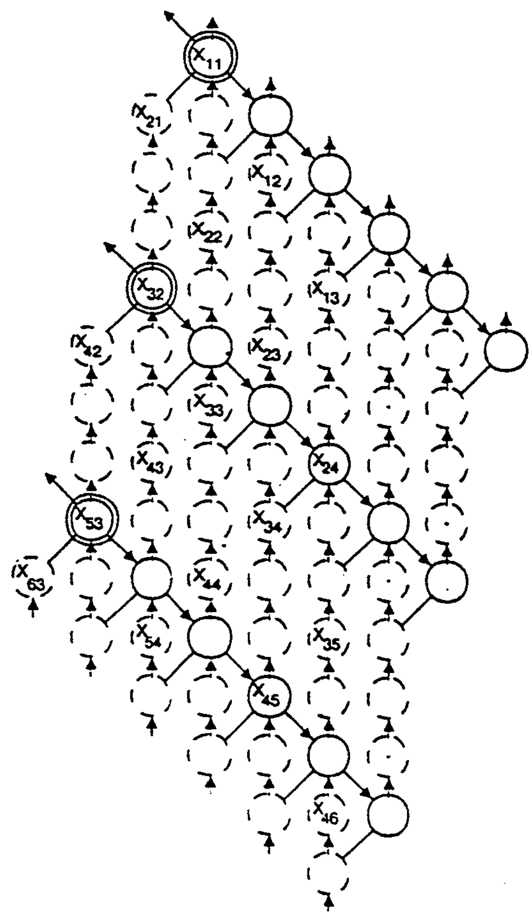


Figure 6. Systolic array for triangularizing a band matrix.

References

1. Bojanczk, A., Brent, R. P. and H. T. Kung, "Numerically Stable Solution of Dense Systems of Linear Equations Using Mesh-Connected Processors," Technical Report, Carnegie-Mellon University, Department of Computer Science. May 1981.

2. Gentleman, W. M., "Least Squares Computations by Givens Transformations Without Square Roots," J. Inst. Maths Applics Vol. 2, pp. 329-336. 1973.

3. Gentleman, W. M., "Basic Procedures for Large, Sparse or Weighted Linear Least Squares Problems (Algorithm AS 75)," Applied Statistics Vol. 3, pp. 448-454. 1974.

4. Gentleman, W. M., "Error Analysis of *QR* Decompositions by Givens Transformations," Linear Algebra and Its Applications Vol. 10, pp. 189-197. 1975.

5. Hammarling, S., "A Note on Modifications to The Givens Plane Rotations," J. Inst. Maths Applics Vol. 13, pp. 215-218. 1974.

6. Kung, H. T., "Why Systolic Architectures," to appear in Computer Magazine.

7. Kung, H. T. and C. L. Leiserson, "Systolic Arrays (for VLSI)," in Sparse Matrix Proceedings 1978, edited by I. S. Duff and G. W. Stewart, SIAM 1979, pp. 256-282. A slightly different version appears in the text, Introduction to VLSI Systems, by C. A. Mead and L. A. Conway, Addison-Wesley 1980, Section 8.3.7.

8. National Physical Laboratory, England, Modern Computing Methods, Her Majesty's Stationery Office 1961.

9. Rogers, L. D., Optimal Paging Strategies and Stability Considerations for Solving Large Linear Systems. Ph.D. thesis, University of Waterloo, Department of Computer Science 1973.

10. Sameh, A. H., private communication.

11. Speiser, J. M. and M. J. Whitehouse, "Architectures for Real-Time Matrix Operations," Prodeedings of Government Microcircuits Applications Conference, 1980.

12. Stewart, G. W., Introduction to Matrix Computations, Academic Press 1973.