SIAM J. APPL. MATH
Vol. 20, No. 1, January 1971

ON MINIMIZING THE NUMBER OF MULTIPLICATIONS NECESSARY FOR MATRIX MULTIPLICATION*

J. E. HOPCROFT† AND L. R. KERR‡

Abstract. This paper develops an algorithm to multiply a $p \times 2$ matrix by a $2 \times n$ matrix in $\lceil (3pn + \max(n, p))/2 \rceil$ multiplications without use of commutativity of matrix elements. The algorithm minimizes the number of multiplications for matrix multiplication without commutativity for the special cases $p = 1$ or 2, $n = 1, 2, \cdots$ and $p = 3$, $n = 3$. It is shown that commutativity actually reduces the number of multiplications required.

1. Introduction. Computational complexity is concerned with the amount of time, space, or number of operations necessary for a computation regardless of the algorithm used. In this paper we consider matrix multiplication and attempt to determine the minimum number of multiplications necessary without commutativity of matrix elements. We succeed only for certain special cases. Our reason for considering the case where the matrix elements do not commute is that we have sufficient tools to get a handle on the problem. As noted later, the number of multiplications required in the commutative case differs by at most a factor of two. Thus if we could establish a growth rate of n^k for some k in the noncommutative case, we would be off by at most a constant factor in the commutative case. We do not try to minimize arithmetic operations for similar reasons. Furthermore, our goal is not to find practical algorithms but to establish lower bounds on such algorithms.

The obvious method for matrix multiplication (computing each element in turn by multiplying appropriate row times column) requires n^3 multiplications. A method due to Strassen [6] requires $n^{\log_2 7}$ or approximately $n^{2.81}$ multiplications. The method is to compute the product of two 2×2 matrices in 7 multiplications not making use of commutativity. Thus it does not matter if the elements of the matrix are themselves matrices. By letting the elements of the matrix be 2×2 matrices, one can multiply two 4×4 matrices in 49 multiplications, and so on. Thus the algorithm uses $7^{\log_2 n}$ or $n^{\log_2 7}$ multiplications.

In the first part of this paper, a modification of Strassen's algorithm will be given for computing a $(p \times 2) \times (2 \times n)$ matrix in $\lceil (3pn + \max(n, p))/2 \rceil$ multiplications without using commutativity. The remainder of the paper is devoted to proving that for $p = 1$ or $2, n = 1, 2, \cdots$ and $p = 3, n = 3$, the algorithm minimizes the number of multiplications necessary for matrix multiplication without using commutativity. It is then shown that commutativity of the matrix elements reduces the number of multiplications required.

2. Algorithm for $(p \times 2) \times (2 \times n)$ matrix multiplication. Let A be a $p \times 2$ matrix, X a $2 \times n$ matrix, and let $Y = AX$. We denote the elements of A, X and Y

* Received by the editors October 16, 1969, and in revised form March 24, 1970. This research was supported in part by the National Science Foundation under Grant GJ-96.

† Department of Computer Science, Cornell University, Ithaca, New York 14850. Now at Department of Computer Science, Stanford University, Stanford, California 94305.

‡ Boeing Scientific Research Laboratories, Seattle, Washington 98124.

by indeterminates a_{ij}, x_{ij} and y_{ij} respectively. Define

$$
\begin{aligned}
A &= a_{12}(x_{11} + x_{21}), & B &= (a_{11} - a_{12})x_{11}, \\
C &= (a_{21} - a_{22})x_{22}, & D &= a_{21}(x_{12} + x_{22}), \\
E &= (a_{12} + a_{22})(x_{21} + x_{22}), & F &= (a_{11} + a_{21})(x_{11} + x_{12}), \\
G &= (a_{12} + a_{21})(x_{11} - x_{22}).
\end{aligned} \tag{1}
$$

Then Strassen's algorithm computes

$$
\begin{aligned}
y_{11} &= A + B, & y_{12} &= -B - D + F - G, \\
y_{21} &= -A + C + E + G, & y_{22} &= -C + D.
\end{aligned} \tag{2}
$$

Let $\mathcal{T}$ be the group of transformations generated by the set of transformations which

(i) interchange two rows of A, two columns of X, or the two columns of A and the two rows of X;

(ii) either add (mod 2) row i of A to row j of A, column i of X to column j of X, or add (mod 2) column i of A to column j of A and simultaneously add (mod 2) row i of X to row j of X.

Applying members of $\mathcal{T}$ to Strassen's algorithm yields other methods of computing y_{11}, y_{12}, y_{21} and y_{22} or other sets of elements of Y depending on the transformation selected. For example, let T be the transformation which interchanges a_{11} and a_{31}, a_{12} and a_{32}, x_{11} and x_{13}, and x_{21} and x_{23}, leaving the remaining variables unchanged. Applying T to the algorithm yields a method of computing y_{22}, y_{23}, y_{32} and y_{33}. The transformation which replaces a_{i1} by $a_{i1} + a_{i2}$ and x_{2j} by $-x_{1j} + x_{2j}, 1 \leqq i \leqq p, 1 \leqq j \leqq n$, leaving the remaining variables unchanged yields a new method of computing y_{11}, y_{12}, y_{21} and y_{22}. In general, transformations on rows of A or columns of X change the y_{ij} computed, and transformations on columns of A and the corresponding rows of X yield new algorithms for computing the same y_{ij}.

In fact, if y_{ii} and y_{jj} are each computed by one of the following three formulas:

$$y_{ll} = a_{l2}(x_{1l} + x_{2l}) + (a_{l1} - a_{l2})x_{1l}, \tag{3a}$$

$$y_{ll} = -(a_{l1} - a_{l2})x_{2l} + a_{l1}(x_{1l} + x_{2l}), \tag{3b}$$

$$y_{ll} = a_{l2}x_{2l} + a_{l1}x_{1l}, \tag{3c}$$

but not the same formulas, then there exists a transformation T in $\mathcal{T}$ which when applied to Strassen's algorithm yields an algorithm for y_{ij} and y_{ji} with three additional multiplications.[1] This suggests that diagonal elements of Y can each be computed with two multiplications, and pairs of off-diagonal elements by three multiplications.

THEOREM 1. *Let A be a $p \times 2$ matrix whose elements are indeterminates a_{ij}, $1 \leqq i \leqq p, 1 \leqq j \leqq 2$, and let X be a $2 \times n$ matrix whose elements are indeterminates x_{jk}, $1 \leqq j \leqq 2, 1 \leqq k \leqq n$. Then $\lceil (3pn + \max(n, p))/2 \rceil$ multiplications are sufficient to compute AX without making use of commutativity.*

[1] The details are tedious and of little interest to the development and thus are omitted. A proof of this and other claims can be found in [2].

Proof. We give only a sketch. Technical details are omitted but can be found in [2]. Without loss of generality assume $p \leqq n$ (otherwise let $A' = X^T$ and $X' = A^T$ and compute $A'X'$). Without loss of generality assume $n < 2p$ (otherwise write $n = kp + n'$, $k \geqq 1$, $p \leqq n' < 2p$ and decompose A into k $p \times 2$ matrices and an $n' \times 2$ matrix). Computing the product of X with each submatrix requires[2] $k\lceil(3p^2 + p)/2\rceil + \lceil(3pn' + n')/2\rceil = \lceil(3pn + n)/2\rceil$.

Let $a_1, a_2, \cdots, a_p$ be indeterminates. Let $l_1, l_2, \cdots, l_{n-p}$ be linear operators of $a_1, a_2, \cdots, a_p$ such that for any k, $a_k, a_{k+1}, \cdots, a_{k+p-1}$ are nondependent,[3] where $a_{p+i} = l_i(a_1, a_2, \cdots, a_p)$, $1 \leqq i \leqq n - p$. Augment the matrix A with $n - p$ rows, where $a_{ij} = l_{i-p}(a_{ij}, a_{2j}, \cdots, a_{pj})$, $p < i \leqq n$, $1 \leqq j \leqq 2$. Call the augmented matrix $\bar{A}$. It suffices to compute p consecutive elements in each column of $\bar{A}X$. The diagonal elements are computed by two multiplications each using one of the three formulas for y_{ll} given previously. For p odd, let $k = (p - 1)/2$, and for p even, let $k = (p - 2)/2$. In each column of Y, the k elements immediately above and immediately below the diagonal element are computed. These elements are computed by selecting symmetrically located pairs and using three additional multiplications to compute each pair. If y_{ii} and y_{jj} are computed by different formulas from (3), then the formulas for y_{ij} and y_{ji} are obtained by applying the appropriate transformation T from $\mathscr{T}$ to formulas (2). If y_{ii} and y_{jj} are computed by the same formula from (3), then we compute $y_{ij} + y_{i+1,j}$ and $y_{ji} + y_{j,i+1}$ instead and subtract $y_{i+1,j}$ and $y_{j,i+1}$. (If $y_{i+1,j}$ and $y_{j,i+1}$ are not computed because i is not in the range $j - k \leqq i \leqq j + k$, then reverse i and j.) For p odd, the algorithm is complete and the number of multiplications is $2n + 3kn$ as was to be shown. For p even, we must compute one additional element in each column of Y. Again the details are omitted but the elements are computed in pairs, three multiplications per pair, with one element left over if n is odd. Two multiplications have been used for each of the n diagonal elements and three multiplications for each of $\lceil(np - n)/2\rceil$ pairs of elements. For p even and n odd, two additional multiplications are required for the leftover element. Thus a total of $\lceil(3np + n)/2\rceil$ multiplications are required.

3. Minimality of the number of multiplications. In order to prove minimality we must first specify the class of algorithms under consideration. We consider algorithms consisting of a sequence of instructions of the form $f_i = g_i \circ h_i$, where $\circ$ stands for one of the binary operations—multiplication, addition or subtraction. Each f_i is a new variable and each g_i and h_i is either a fixed integer, a previously computed f_j or an input variable. Certain f_i will be considered to be output variables.

In proving minimality we shall assume that all numbers are binary and addition and multiplication are modulo two. This can be done since an algorithm which works for integers must also work modulo 2. Without loss of generality we can assume that for at least one minimal program the only multiplications are between linear functionals of the elements of A and X. Since we do not have

[2] Note $\lceil(3p^2 + p)/2\rceil = (3p^2 + p)/2$.

[3] For notational convenience we interpret $a_k, a_{k+1}, \cdots, a_{k+p}$ for $k + p > n$ as $a_k, a_{k+1}, \cdots, a_n, a_1, a_2, \cdots, a_{k+p-n}$. We shall adhere to this convention throughout the paper. The indeterminates $a_1, a_2, \cdots, a_n$ are nondependent if for scalars c_i, $1 \leqq i \leqq k$, $\sum_{i=1}^{k} c_i a_i = 0$ implies $c_i = 0$, $1 \leqq i \leqq k$.

commutativity, we can assume without loss of generality that an optimal algorithm computes each y_{ij} as a linear sum of the m_i, where each m_i is of the form $\alpha \cdot \beta$, α a linear functional of the a_{ij} and β a linear functional of the x_{ij}. One should note from these canonical forms that commutativity can reduce the number of multiplications by at most one half.

The technique which we shall use is similar to that in Pan [3]. Initially we wish to determine the minimum number of multiplications needed to compute AX for arbitrary matrices A and X. If an algorithm with a minimal number of multiplications has as one multiplication $a_{11}x_{11}$, then setting $a_{11} = 0$ we can conclude that the original number of multiplications is at least one more than the minimal number of multiplications needed for the restricted problem where the upper left-hand element of A is always zero. By repeatedly restricting the input space, we achieve a lower bound on the number of multiplications.

We now state several results which will be needed later. Let $\mathcal{F}$ be a field and $\{a_1, a_2, \cdots, a_n\}$ a set of indeterminates. Let $\hat{\mathcal{F}}$ be the field obtained by extending $\mathcal{F}$ by rational expressions of the a_i. A set of vectors $x_1, x_2, \cdots, x_p$ with elements from $\hat{\mathcal{F}}$ are *nondependent* if and only if $\sum_{i=1}^{p} a_i x_i = 0$, each a_i in $\mathcal{F}$, implies each $a_i = 0$. Nondependence should not be confused with linear independence where the a_i are not restricted to $\mathcal{F}$ but can be arbitrary elements of $\hat{\mathcal{F}}$.

LEMMA 1 (Winograd). *Let A be an $m \times n$ matrix whose elements are from $\hat{\mathcal{F}}$, and let $\mathbf{x}$ be an arbitrary n-vector. If A has p nondependent columns, then any algorithm computing $A\mathbf{x}$ requires at least p multiplications or divisions.*

LEMMA 2. *Let $F = \{f_1, \cdots, f_k, \cdots, f_n\}$ be a set of expressions, where $f_1, \cdots, f_k$ are independent and each can be expressed as a single product. If F can be computed with p multiplications, then there exists an algorithm for F with p multiplications in which k of the multiplications are $f_1, \cdots, f_k$.*

LEMMA 3. *Any program for*

$$a_{21}x_{2i}, \quad a_{21}x_{1i} + a_{22}x_{2i}, \qquad 1 \leqq i \leqq n,$$

requires $3n$ multiplications.

Proof. We can assume by the above lemma that n of the multiplications are $a_{21}x_{2i}$, $1 \leqq i \leqq n$. If we remove these multiplications from the algorithm, it will compute $a_{21}x_{1i} + a_{22}x_{2i} + [a_{21}x_{21}] + \cdots + [a_{21}x_{2n}], 1 \leqq i \leqq n$, where $[\alpha]$ means that α may or may not be present.

In matrix form this can be represented as

$$\begin{bmatrix} a_{21} & & 0 & a_{22} + [a_{21}] & & [a_{21}] \\ & \ddots & & & \ddots & \\ & & \ddots & & & \ddots \\ 0 & & a_{21} & [a_{21}] & & a_{22} + [a_{21}] \end{bmatrix} \begin{bmatrix} x_{11} \\ \vdots \\ x_{1n} \\ x_{21} \\ \vdots \\ x_{2n} \end{bmatrix}.$$

The first matrix clearly has $2n$ nondependent columns; therefore, by Lemma 1, $2n$ multiplications are required for this computation. Since n multiplications were removed from the original algorithm, it must have had $3n$ multiplications.

We now make use of the high degree of symmetry in matrix multiplication to establish the technical lemmas needed to show that multiplying a 2×2 matrix A by a $2 \times n$ matrix X requires exactly $\lceil 7n/2 \rceil$ multiplications.

LEMMA 4. *If an algorithm for $A \times X$ has k multiplications of forms $a_{11}\alpha$, $(a_{12} + a_{21})\beta$, and $(a_{11} + a_{12} + a_{21})\gamma$, then the algorithm requires at least $3n + k$ multiplications.*

Proof. Set $a_{11} = 0$ and $a_{12} = a_{21}$. This eliminates k multiplications and the algorithm now computes:

$$a_{21}x_{2i}, \quad a_{21}x_{1i} + a_{22}x_{2i}, \qquad 1 \leqq i \leqq n,$$

which requires $3n$ multiplications by Lemma 3.

COROLLARY. *By applying transformations in $\mathcal{T}$, we also have similar theorems for*

(a) $(a_{11} + a_{21})\alpha, (a_{12} + a_{21} + a_{22})\beta, (a_{11} + a_{12} + a_{22})\gamma$,
(b) $(a_{11} + a_{12})\alpha, (a_{12} + a_{21} + a_{22})\beta, (a_{11} + a_{21} + a_{22})\gamma$,
(c) $(a_{11} + a_{12} + a_{21} + a_{22})\alpha, (a_{12} + a_{21})\beta, (a_{11} + a_{22})\gamma$,
(d) $a_{21}\alpha, (a_{11} + a_{22})\beta, (a_{11} + a_{21} + a_{22})\gamma$,
(e) $(a_{21} + a_{22})\alpha, (a_{11} + a_{12} + a_{22})\beta, (a_{11} + a_{12} + a_{21})\gamma$,
(f) $a_{12}\alpha, (a_{11} + a_{22})\beta, (a_{11} + a_{12} + a_{22})\gamma$,
(g) $(a_{12} + a_{22})\alpha, (a_{11} + a_{21} + a_{22})\beta, (a_{11} + a_{12} + a_{21})\gamma$,
(h) $a_{22}\alpha, (a_{12} + a_{21})\beta, (a_{12} + a_{21} + a_{22})\gamma$.

LEMMA 5. *Any algorithm P for $A \times X$ which has k multiplications of types $a_{11}\alpha$, $a_{12}\beta$, and $(a_{11} + a_{12})\gamma$ has at least $3n + k/2$ multiplications.*

Proof. Let M be the vector space of linear combinations of the k multiplications which are assumed to be present. We first prove the theorem for the case where the dimension of M is k.

Consider the set of expressions $S' = \{a_{11}x_{1i} + a_{12}x_{2i} | 1 \leqq i \leqq n\}$, each of which is computed by P. Let S be the vector space of linear combinations of expressions in S'. Clearly S has dimension n. The subspace $S \cap M$ has dimension less than or equal to $k/2$, since any $l/2$ linearly independent expressions in $S \cap M$ can be represented as the product of a matrix with l nondependent columns and an arbitrary vector and thus requires l multiplications by Lemma 1. Hence, the vector space spanned by $S \cup M$ has dimension of at least $n + k/2$. We can therefore solve for $n + k/2$ of the multiplications in the algorithm P in terms of the expressions in S' and the k multiplications. If we now set $a_{11} = a_{12} = 0$, then we lose $n + k/2$ multiplications from P.

The new algorithm computes $a_{21}x_{1i} + a_{22}x_{2i}$, $1 \leqq i \leqq n$, which requires $2n$ multiplications by Lemma 1. Hence, P must have had at least $3n + k/2$ multiplications.

In the case where the dimension of M is k', $k' < k$, then solve for $k - k'$ multiplications in terms of the remaining multiplications and replace them by their solution wherever they appear in the program. This results in an algorithm with at least $3n + k'/2$ multiplications, as we have just proved. Thus P must have had at least $3n + k'/2 + k - k' > 3n + k/2$ multiplications.

COROLLARY. *By transformations we also have similar theorems for $a_{21}\alpha$, $a_{22}\beta$, $(a_{21} + a_{22})\gamma$ and $(a_{11} + a_{21})\alpha, (a_{12} + a_{22})\beta, (a_{11} + a_{12} + a_{21} + a_{22})\gamma$.*

THEOREM 2. *Let A be a 2×2 matrix and X a $2 \times n$ matrix. Any algorithm P for $A \times X$ not making use of commutativity requires $\lceil 7n/2 \rceil$ multiplications.*

Proof. Divide the multiplications in P into two disjoint sets S_A and S_B, where the multiplications in S_A begin with $a_{11}, a_{12}, a_{21}, a_{22}, a_{11} + a_{12}, a_{11} + a_{21}$, $a_{12} + a_{22}$, $a_{21} + a_{22}$, or $a_{11} + a_{12} + a_{21} + a_{22}$, and the multiplications in S_B begin with $a_{11} + a_{22}$, $a_{12} + a_{21}$, $a_{11} + a_{12} + a_{21}$, $a_{11} + a_{12} + a_{22}$, $a_{11} + a_{21} + a_{22}$ or $a_{12} + a_{21} + a_{22}$. Note that each multiplication in S_A belongs to one group of Lemma 4 or its corollary and one group of Lemma 5 or its corollary, while each multiplication in S_B belongs to three groups of Lemma 4 or its corollary.

Suppose that P has m multiplications and that there are m_A multiplications in S_A and m_B multiplications in S_B. Since $m = m_A + m_B$, then either $m_A \geqq 6m/7$ or $m_B \geqq m/7$. If $m_A \geqq 6m/7$, then there must be some group of Lemma 5 or its corollary with at least $m_A/3 \geqq 2m/7$ multiplications, and hence by Lemma 5 or its corollary, $m \geqq 3n + m/7$. If $m_B \geqq m/7$, let the average number of multiplications per group of Lemma 4 or its corollary be r. Then $m = 9r - 2m_B$ since there are nine groups and each multiplication in S_A appears in exactly one group while each multiplication in S_B appears in three groups. Thus $r = (m + 2m_B)/9$. Since $m_B \geqq m/7$, $r \geqq m/7$. Hence one group of Lemma 4 or its corollary must have at least $m/7$ multiplications. In either case, therefore, $m \geqq 3n + m/7$. Solving this inequality yields $m \geqq 7n/2$.

The lower bound of Theorem 2 for a 2×2 matrix A times a $2 \times n$ matrix X is realized by the algorithm of Theorem 1. This leads to the following theorem.

THEOREM 3. *The minimal algorithm for matrix multiplication with noncommutative elements has $\lceil 7n/2 \rceil$ multiplications for the $(2 \times 2) \times (2 \times n)$ case (and similarly for the $(n \times 2) \times (2 \times 2)$ case).*

By similar techniques, one can show that 15 multiplications are necessary and sufficient for the $(3 \times 2) \times (2 \times 3)$ case. However, a detailed analysis of the $(3 \times 2) \times (2 \times 4)$ case seems to indicate that 19 multiplications are required rather than the 20 of Theorem 1. However, we conjecture that 20 are required and that the algorithm of Theorem 1 is optimal for all n and p.

We now turn our attention to matrix multiplication in which the multiplication operation is commutative. Methods are known [5] for $(n \times 2) \times (2 \times m)$ matrix multiplication which require $mn + m + n$ multiplications. Setting $m = 2$ and comparing with minimal bounds for the noncommutative case, we see that the commutative case requires fewer multiplications for all $n \geqq 5$. Clearly the commutative case cannot require more multiplications than the noncommutative case. Here we have shown that for matrices of certain dimension, commutativity actually reduces the number of multiplications. As mentioned earlier, however, the number of multiplications cannot be reduced by more than a factor of two.

REFERENCES

[1] V. V. KLYUYEV AND N. I. KOKOVKIN-SHCHERBAK, *On the minimization of the number of arithmetic operations for the solution of linear algebraic systems of equations*, TR CS24, Department of Computer Science, Stanford University, Stanford, Calif., 1965.

[2] J. E. HOPCROFT AND L. R. KERR, *On minimizing the number of multiplications necessary for matrix multiplications*, TR 69-44, Department of Computer Science, Cornell University, Ithaca, N.Y., 1969.

[3] V. YA. PAN, *Methods of computing values of polynomials*, Russian Math. Surveys, 21 (1966), pp. 105–136.
[4] S. WINOGRAD, *On the number of multiplications required to compute certain functions*, Proc. Nat. Acad. Sci. U.S.A., 58 (1967), pp. 1840–1842.
[5] ———, *A new algorithm for inner product*, IEEE Trans. Computers, 17 (1968), pp. 693–694.
[6] VOLKER STRASSEN, *Gaussian elimination is not optimal*, Numer. Math., 13 (1969), pp. 354–356.