On Minimizing the Number of Multiplications Necessary for Matrix Multiplication

J. E. Hopcroft

L. R. Kerr

Technical Report

No. 69-44

September 1969

Department of Computer Science
Cornell University
Ithaca, New York 14850

On Minimizing the Number of Multiplications Necessary for Matrix Multiplication*

J.E. Hopcroft
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. 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 with commutativity fewer multiplications are required.

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.

---

* This research has been supported in part by National Science Foundation Grant GJ-96.

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 [3] 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 $p \times 2$ by $2 \times n$ 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 with commutativity fewer multiplications are required.

Algorithm for $p \times 2$ by $2 \times n$ matrix multiplication

Let A be a $p \times 2$ matrix with elements a_{ij}, $1 \le i \le p$, $1 \le j \le 2$ and let X be a $2 \times n$ matrix with elements x_{ij}, $1 \le i \le 2$, $1 \le j \le n$. We shall make use of the following notation. If $A(a_i, x_k) = a_{i2}(x_{1k}+x_{2k})$, then we write $A(a_{i+j}, x_k)$ for $a_{i+j,2}(x_{1k}+x_{2k})$ and $A(a_i+a_j, x_k)$ for $(a_{i2}+a_{j2})(x_{1k}+x_{2k})$.

Let

$A(a_i, x_k) = a_{i2}\ (x_{1k}+x_{2k})$

$B(a_i, x_k) = (a_{i1} - a_{i2})\ x_{1k}$

$C(a_j, x_\ell) = (a_{j1} - a_{j2})\ x_{2\ell}$

$D(a_j, x_\ell) = a_{j1}\ (x_{1\ell} + x_{2\ell})$

$E(a_i + a_j, x_k + x_\ell) = (a_{i2}+a_{j2})\ (x_{2k}+x_{2\ell})$

$F(a_i+a_j, x_k+x_\ell) = (a_{i1}+a_{j1})\ (x_{1k}+x_{1\ell})$

$G(a_i, a_j, x_k, x_\ell) = (a_{j1}+a_{i2})\ (x_{1k} - x_{2\ell})$

Strassen's algorithm computes

$y_{ik} = a_{i1}x_{1k} + a_{i2}x_{2k}$, $\qquad y_{i\ell} = a_{i1}x_{1\ell} + a_{i2}x_{2\ell}$,

$y_{jk} = a_{j1}x_{1k} + a_{j2}x_{2k}$ and $y_{j\ell} = a_{j1}x_{1\ell} + a_{j2}x_{2\ell}$ as

$y_{ik} = A(a_i, x_k) + B(a_i, x_k)$

$y_{i\ell} = -B(a_i, x_k) - D(a_j, x_\ell) + F(a_i+a_j, x_k+x_\ell) - G(a_i, a_j, x_k, x_\ell)$

$y_{jk} = -A(a_i, x_k) + C(a_j, x_\ell) + E(a_i+a_j, x_k+x_\ell) + G(a_i, a_j, x_k, x_\ell)$

$y_{j\ell} = -C(a_j, x_\ell) + D(a_j, x_\ell)$

Note that $y_{ik} + y_{i\ell}$, $y_{jk} + y_{j\ell}$, $y_{ik} + y_{jk}$, $y_{i\ell} + y_{j\ell}$

and $y_{ik} + y_{i\ell} + y_{jk} + y_{j\ell}$ can be computed with no additional multiplications. The formulae are summarized below:

$y_{ik} + y_{i\ell} = A(a_i,x_k) - D(a_j,x_\ell) + F(a_i+a_j,x_k+x_\ell) - G(a_i,a_j,x_k,x_\ell)$

$y_{jk} + y_{j\ell} = -A(a_i,x_k) + D(a_j,x_\ell) + E(a_i+a_j,x_k+x_\ell) + G(a_i,a_j,x_k,x_\ell)$

$y_{ik} + y_{jk} = B(a_i,x_k) + C(a_j,x_\ell) + E(a_i+a_j,x_k+x_\ell) + G(a_i,a_j,x_k,x_\ell)$

$y_{i\ell} + y_{j\ell} = -B(a_i,x_k) - C(a_j,x_\ell) + F(a_i+a_j,x_k+x_\ell) - G(a_i,a_j,x_k,x_\ell)$

$y_{ik}+y_{i\ell}+y_{jk}+y_{j\ell} = E(a_i+a_j,x_k+x_\ell) + F(a_i+a_j,x_k+x_\ell)$

The essence of the algorithm to be given is to compute groups of four elements using Strassen's method and to share as many multiplications between groups as possible. For example, the 3 × 2 by 2 × 3 case would be computed as:

$y_{11} = A(a_1,x_1) + B(a_1,x_1)$

$y_{12} = -B(a_1,x_1) - D(a_2,x_2) + F(a_1+a_2,x_1+x_2) - G(a_1,a_2,x_1,x_2)$

$y_{13} = A(a_1,x_1) - D(-a_1+a_3,-x_1+x_3) + F(a_3,x_3) -$
$\quad G(a_1,-a_1+a_3,x_1,-x_1+x_3)$

$y_{21} = -A(a_1,x_1) + C(a_2,x_2) + E(a_1+a_2,x_1+x_2) + G(a_1,a_2,x_1,x_2)$

$y_{22} = -C(a_2,x_2) + D(a_2,x_2)$

$y_{23} = -A(-a_2+a_3,-x_2+x_3) + D(a_2,x_2) + E(a_3,x_3) +$
$\quad G(-a_2+a_3,a_2,-x_2+x_3,x_2)$

$y_{31} = B(a_1,x_1) + D(-a_1+a_3,-x_1+x_3) + E(a_3,x_3) +$
$\quad G(a_1,-a_1+a_3,x_1,-x_1+x_3)$

$y_{32} = -B(-a_2+a_3,-x_2+x_3) - C(a_2,x_2) + F(a_3,x_3) -$
$\quad G(-a_2+a_3,a_2,-x_2+x_3,x_2)$

$y_{33} = E(a_3,x_3) + F(a_3,x_3)$

Here we have computed $y_{11} y_{12}, y_{21}$ and y_{22} as a group. Similarly y_{22}, y_{23}, y_{32} and y_{33} and y_{11}, y_{13}, y_{31}, and y_{33}. Note that two multiplications are common to the first and second groups, two multiplications are common to the second and third groups and two multiplications are common to the first and third groups. All told 15 multiplications are used.

Before giving the general construction we proceed with several technical lemmas.

Lemma 1: Let $a_1, a_2, \ldots, a_p$ be indeterminants. For $n \geq p$ there exist linear functionals $\ell_1, \ell_2, \ldots, \ell_{n-p}$ such that for any k, $a_k, a_{k+1}, \ldots, a_{k+p-1}$ (1) are nondependent where $a_{p+i} = \ell_i(a_1, a_2, \ldots, a_p)$, $1 \leq i \leq n-p$.

Proof: We can interpret the sequence $a_1, a_2, \ldots, a_n$ as an $n \times p$ matrix M where the i^{th} row represents a linear functional expressing a_i in terms of $a_1, a_2, \ldots, a_p$. Thus we need only show how to construct an $n \times p$ matrix where

1) the first p rows form the identity matrix, and

2) each submatrix consisting of p consecutive rows of M

---

(1) For notational convenience we interpret $a_k, a_{k+1}, \ldots, a_{k+p}$ for $k + p > n$ as $a_k, a_{k+1}, \ldots, a_n, a_1, a_2, \ldots a_{k+p-n}$. We shall adhere to this convention throughout the paper. The indeterminants $a_1, a_2, \ldots, a_n$ are nondependent if for scalars c_i, $1 \leq i \leq k$, $\sum_{i=1}^{k} \ell_i a_i = 0$ implies $c_i = 0$, $1 \leq i \leq k$.

has a non-zero determinant. (Here rows n and 1 are also considered to be consecutive). This is equivalent to constructing an $n \times p$ matrix such that

1) the first p rows form the identity matrix

2)
$$\begin{vmatrix} a_{i-p+1,1} & \cdots & a_{i-p+1,p} \\ \vdots & & \vdots \\ a_{i1} & \cdots & a_{ip} \end{vmatrix} \neq o, p \leq i \leq n$$

3)
$$\begin{vmatrix} a_{n-i,p-i} & \cdots & a_{n-i,p} \\ \vdots & & \vdots \\ a_{n,p-i} & \cdots & a_{np} \end{vmatrix} \neq o, o \leq i \leq p-1$$

Clearly for $n=p$, the identity matrix satisfies the above three conditions. Assume we have an $n \times p$ matrix satisfying above conditions. Then we can construct an $(n+1) \times p$ matrix satisfying the above conditions since for each i the determinants

$$D(i) = \begin{vmatrix} a_{n+1-i,p-i} & \cdots & a_{n+1-i,p} \\ a_{n+1,p-i} & \cdots & a_{n+1,p} \end{vmatrix} \qquad o \leq i \leq p-1$$

depend on the value of $a_{n+1,p-i}$. Thus starting with $i = o$ for each i, $o \leq i \leq p-1$, select $a_{n+1,p-i}$ so that $D(i) \neq o$.

Lemma 2 If y_{ii} and y_{jj} are each computed by one of the following three methods

1) $y_{\ell\ell} = A(a_\ell, x_\ell) + B(a_\ell, x_\ell)$

2) $y_{\ell\ell} = -C(a_\ell, x_\ell) + D(a_\ell, x_\ell)$

3) $y_{\ell\ell} = E(a_\ell, x_\ell) + F(a_\ell, x_\ell)$

but not the same method, then y_{ij} and y_{ji} can be computed with three additional multiplications.

__Proof.__ If y_{ii} is computed by (1) and y_{jj} by (2), then $y_{ij} = -B(a_i,x_i) - D(a_j,x_j) + F(a_i+a_j,x_i+x_j) - G(a_i,a_j,x_i,x_j)$ and $y_{ji} = -A(a_i,x_i) + C(a_j,x_j) + E(a_i+a_j,x_i+x_j) + G(a_i,a_j,x_i,x_j)$.

If y_{ii} is computed by (1) and y_{jj} by (3), then $y_{ij} = A(a_i,x_i) - D(-a_i+a_j,-x_i+x_j)+F(a_j,x_j) - G(a_i,-a_i+a_j,x_i,-x_i+x_j)$ and $y_{ji} = B(a_i,x_i) + C(-a_i+a_j,-x_i+x_j) + E(a_j,x_j) + G(a_i,-a_i+a_j,x_i,-x_i+x_j)$.

If y_{ii} is computed by (2) and y_{jj} by (3), then $y_{ij} = -A(-a_i+a_j,-x_i+x_j) + D(a_i,x_i) + E(a_j,x_j) + G(-a_i+a_j,a_i,-x_i+x_j,x_i)$ and $y_{ji} = -B(-a_i+a_j,-x_i+x_j) - C(a_i,x_i) + F(a_j,x_j) - G(-a_i+a_j,a_i,-x_i+x_j,x_i)$.

The other three cases are obtained by interchanging the roles of i and j.

To simplify the next lemma, we define two constants ℓ and m. Let n and k be positive integers where $2k+2 \leq n < 4k+4$. If $2k+2 \leq n \leq 3k+3$, then $\ell = 2k+3$ and $m = n - (2k+2)$ or $n - (2k+3)$, whichever is even. If $3k+3 < n < 4k+4$, then $\ell = n - k$ and $m = 4k+4 - n$ or $4k+3 - n$, whichever is even.

__Lemma 3:__

A sequence $i_1, i_2, \ldots, i_n$ where $i_j = 1,2,$ or 3 for $1 \leq j \leq n$ can be constructed satisfying the properties:

(1) $i_j \neq i_{j+1}$ $\quad 1 \leq j \leq n$ where i_{n+1} is interpreted to be i_1.

(2) $\left.\begin{array}{l} i_j \neq i_{j-k} \\ i_{j+1} \neq i_{j+k+1} \end{array}\right\}$ $j = \ell, \ell+2, \ldots, \ell+m-2$

(3) either $i_j \neq i_{j+k+1}$, or

$i_{j+1} \neq i_{j-k}$ for $j = \ell, \ell+2, \ldots, \ell+m-2$

Proof:

If $m<2$ the lemma is trivial. Thus assume $m \geq 2$.
Let $i_\ell = i_{\ell+2} = \ldots = i_{\ell+m-2} = 1$,
$i_{\ell+1} = i_{\ell+3} = \ldots = i_{\ell+m-1} = 2$, $i_{\ell-k} = i_{\ell+2-k} = \ldots$
$= i_{\ell+m-2-k} = 2$ and $i_{\ell+1+k} = i_{\ell+3+k} = \ldots = i_{\ell+m-1+k} = 3$.
For $m \geq 2$ $\ell-k \leq \ell+m-2-k < \ell < \ell+m-1 < \ell+1+k \leq \ell+m-1+k$
and $\ell+m-1+k-n < \ell-k$. Thus, so far no conditions are violated. The remaining entries in the sequence are filled in so as to satisfy condition 1.

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

Proof: Without loss of generality assume $p\leq n$. (Otherwise let $A' = X^T$ and $X' = A^T$ and compute $A'X'$). Without loss of generality assume $n < 2p$. (Otherwise write $n = kp + n'$, $k\geq 1, p\leq n' < 2p$ and decompose A into k $n\times 2$ matrices and an $n'\times 2$ matrix. Computing the product of X with each submatrix requires $k\ (\lceil (3p^2 + p)/2\rceil) + \lceil (3pn'+n')/2\rceil =$ $\lceil (3pn + n)/2\rceil$[(2)]).

Let $a_1, a_2, \ldots, a_p$ be indeterminants. Let $\ell_1, \ell_2, \ldots, \ell_{n-p}$ be linear functionals of $a_1, a_2, \ldots, a_p$ such that the sequence $a_1, a_2, \ldots, a_p, a_{p+1}, \ldots, a_n$ where $a_{p+i} = \ell_i\ (a_1, a_2, \ldots, a_p)$ $1 \le i \le n-p$ satisfies Lemma 1. Augment the matrix A with n-p rows where $a_{ij} = \ell_{i-p}(a_{1j}, a_{2j}, \ldots, a_{pj})$, $p+1 \le i \le n$, $1 \le j \le 2$. Call the augmented matrix $\bar{A}$. It suffices to compute p consecutive elements in each column of AX. We consider two cases. Both cases will make use of the following basic construction.

Basic Construction: Let k be an integer between 0 and (n-1)/2. The elements y_{ij}, $1 \le j \le n$, $j \le k \le i \le j+k$ can be computed in 3kn+2n multiplications as follows. For each i compute y_{ii} according to one of the following three methods.

(1) $y_{ii} = A(a_i, x_i) + B(a_i, x_i)$

(2) $y_{ii} = -C(a_i, x_i) + D(a_i, x_i)$

(3) $y_{ii} = E(a_i, x_i) + F(a_i, x_i)$

subject to the restriction that if y_{ii} is computed according to method j, then $y_{i+1,i+1}$ is not computed by method j. Next, each needed y_{ij} and y_{ji}, such that y_{ii} and y_{jj} are computed by distinct methods,

---

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

are computed according to Lemma 2 at a cost of three additional multiplications per pair of elements. Finally those y_{ij} and y_{ji}, such that y_{ii} and y_{jj} are computed by the same method, are obtained as follows. Several cases exist. We treat only the case where y_{ii} is computed as $A(a_i,x_i)+B(a_i,x_i)$, y_{jj} is computed as $A(a_j,x_j) + B(a_j,x_j)$ and $y_{i+1,i+1}$ is computed as $-C(a_{i+1},x_{i+1}) + D(a_{i+1},x_{i+1})$. In this case the multiplications $E(a_i+a_{i+1},x_i+x_{i+1})$ and $F(a_i+a_{i+1},x_i+x_{i+1})$ are already used in the computation of $y_{i,i+1}$ and $y_{i+1,i}$. Thus we can compute

$$y_{ij}+y_{i+1,j} = B(a_j,x_j) + C(-a_j+a_i+a_{i+1},-x_j+x_i+x_{i+1})$$
$$+ E(a_i+a_{i+1},x_i+x_{i+1}) + G(a_j,-a_j+a_i+a_{i+1},x_j,-x_j+x_i+x_{i+1})$$
$$y_{ji}+y_{j,i+1} = A(a_j,x_j) - D(-a_j+a_i+a_{i+1},-x_j+x_i+x_{i+1}) +$$
$$F(a_i+a_{i+1},x_i+x_{i+1}) - G(a_j,-a_j+a_i+a_{i+1},x_j,-x_j+x_i+x_{i+1})$$

requiring only the three additional multiplications C,D and G. Now $y_{i+1,j}$ and $y_{j,i+1}$ are already computed since y_{ii} and $y_{i+1,i+1}$ are computed by distinct methods and hence y_{jj} and $y_{i+1,i+1}$ are computed by distinct methods. (If $y_{i+1,j}$ and $y_{j,i+1}$ are not computed because i is not in the range $j-k\le i\le j+k$, then reverse i and j.) Thus we obtain $y_{ij} = y_{ij} + y_{i+1,j} - y_{i+1,j}$ and $y_{ji} = y_{ji}+y_{j,i+1} - y_{j,i+1}$. The total cost is $2n + 3kn$ as was to be shown.

We now return to computing p consecutive elements in each column of $\bar{A}X$. We consider two cases.

<u>Case 1:</u> $p = 2k+1$, $k=0,1,2,\ldots$

Let $k = (p-1)/2$. For each i we compute

$y_{i-k,i}, y_{i-k+1,i}, \ldots, y_{i+k,i}$ by the basic construction computing the diagonal elements by

$$y_{ii} = \begin{cases} A(a_i,x_i) + B(a_i,x_i) & i \text{ odd } i \neq n \\ -C(a_i,x_i) + D(a_i x_i) & i \text{ even} \\ E(a_i,x_i) + F(a_i,x_i) & i \text{ odd } i = n \end{cases}$$

This requires $(3pn+n)/2$ multiplications which is always an integer and hence equal to $\lceil (3pn+n)/2 \rceil$.

<u>Case 2:</u> $p = 2k+2$, $k = 0,1,2, \ldots$

Let $j_1, j_2, \ldots, j_n$ be a sequence satisfying lemma 3.

<u>Step 1</u> For each i we compute $y_{i-k,i}, y_{i-k+1,i}, \ldots,$ $y_{i+k,i}$ by the basic construction computing the diagonal element y_{ii} by

$$\begin{array}{ll} A(a_i,x_i) + B(a_i,x_i) & \text{if } j_i = 1 \\ -C(a_i,x_i) + D(a_i,x_i) & \text{if } j_i = 2 \\ E(a_i,x_i) + F(a_i,x_i) & \text{if } j_i = 3 \end{array}$$

This requires $(3pn - 2n)/2$ multiplications.

<u>Step 2</u> Compute the pairs of elements $y_{k+2,1}$ and $y_{1,k+2}$, $y_{k+3,2}$ and $y_{2,k+3}, \ldots,$ $y_{2k+2,k+1}$ and $y_{k+1,2k+2}$. Also, if $3k+4 \leq n$, compute $y_{3k+4,2k+3}$ and $y_{2k+3,3k+4}, y_{3k+5,2k+4}$ and $y_{2k+4,3k+5}, \ldots, y_{n,n-k-1}$ and $y_{n-k-1,n}$. This computation is done by the same method that was used for computing pairs of off-diagonal elements in the basic construction (i.e. using Lemma 2 if the corresponding diagonal elements are

computed by distinct methods, and computing the adjacent elements otherwise) at a cost of three multiplications per pair.

Step 3 If $2k+2 \le n \le 3k+3$, let $\ell = 2k+3$ and $m' = n - (2k+2)$. If $3k+3 < n < 4k+4$, then $\ell = n-k$ and $m' = 4k+4-n$. Now all columns have p consecutive elements computed except for columns $\ell, \ell+1, \ldots, \ell+m'-1$ which have only p-1 consecutive elements computed. If m' is odd compute $y_{\ell+m'+k, \ell+m'-1}$ as $A(a_{\ell+m'+k}, x_{\ell+m'-1}) + B(a_{\ell+m'+k}, x_{\ell+m'-1})$ and let $m = m' - 1$. If m' is even, let $m = m'$. We now must compute one more independent element in columns $\ell, \ell+1, \ldots, \ell+m-1$. We will compute the pairs of elements $y_{\ell+k+1+2i, \ell+2i}$ and $y_{\ell-k+2i, \ell+1+2i}$ for $0 \le i \le (m-2)/2$.

Let $i_1 = \ell+k+1+2i$, $i_2 = \ell+2i$, $i_3 = \ell-k+2i$ and $i_4 = \ell+1+2i$. For $0 \le i \le (m-2)/2$ we compute $y_{i_1 i_2}$ and $y_{i_3 i_4}$ by computing the four elements

$$Z(i_2+i_3, i_2+i_3) = y_{i_2 i_2} + y_{i_2 i_3} + y_{i_3 i_2} + y_{i_3 i_3}$$

$$Z(i_2 + i_3, i_1+i_4) = y_{i_2 i_1} + y_{i_2 i_4} + y_{i_3 i_1} + y_{i_3 i_4}$$

$$Z(i_1+i_4, i_2+i_3) = y_{i_1 i_2} + y_{i_1 i_3} + y_{i_4 i_2} + y_{i_4 i_3}$$

$$Z(i_1+i_4, i_1+i_4) = y_{i_1 i_1} + y_{i_1 i_4} + y_{i_4 i_1} + y_{i_4 i_4}$$

Note that for $0 \le i \le (m-2)/2$, $0 < i_3 < \ell$, $\ell+m-1 < i_1$ and $i_1 - n < \ell$. Thus every element in columns i_1 and i_3 is a linear combination of already computed elements and since $y_{i_2 i_4}$ and $y_{i_4 i_2}$ were computed in step 1, we can obtain $y_{i_1 i_2}$ from $Z(i_1+i_4, i_2+i_3)$ and $y_{i_3 i_4}$ from $Z(i_2+i_3, i_1+i_4)$. (The case k=0 is vacuously true.)

The elements $Z(i_2+i_3, i_2+i_3)$, $Z(i_2+i_3, i_1+i_4)$, $Z(i_1+i_4, i_2+i_3)$ and $Z(i_1+i_4, i_1+i_4)$ are computed using seven multiplications by Lemma 2. The element $Z(i_2+i_3, i_2+i_3)$ is computed using two multiplications according to method j ($j = 1, 2,$ or 3) of Lemma 2 where j is selected so that $j \neq j_2$ and $j \neq j_3$ where j_2 and j_3 are the methods of Lemma 2 used to compute the diagonal elements $y_{i_2 i_2}$ and $y_{i_3 i_3}$, respectively. Similarly $Z(i_1+i_4, i_1+i_4)$ is computed by two multiplications by method j' where j' is selected so that $j' \neq j_1$ and $j' \neq j_4$ where j_1 and j_4 are the methods of Lemma 2 used to compute $y_{i_1 i_1}$ and $y_{i_4 i_4}$. By (3) of Lemma 3 $j \neq j'$.

The claim is made that the two multiplications used to compute $Z(i_2+i_3, i_2+i_3)$ are already present since they were used to compute $y_{i_2 i_3}$ and $y_{i_3 i_2}$ in step 1. Similarly, the two multiplications used to compute $Z(i_1+i_4, i_1+i_4)$ are already present. Thus only three additional multiplications rather than seven are needed to compute the Z's.

This completes the construction. Two multiplications were used for each of the n diagonal elements and three multiplications for each of $\lfloor (np-n)/2 \rfloor$ pairs of elements.

For p even and n odd two additional multiplications were required for the left over element. Thus a total of $\lceil(3np+n)/2\rceil$ multiplications were used.

The algorithm for the $n\times 2$ by $2\times p$ case has an obvious extension to the $kn\times 2^k$ by $2^k \times kp$ case (i.e. uses Strassen's method for computing the product of a $2^k \times 2^k$ matrix with a $2^k \times 2^k$ matrix in 7^k multiplications and overlap these). However, there seems to be no simple formula for the more general case. Usually to compute the product of an $m\times p$ matrix and a $p \times n$, writing p as $\Sigma k_i 2^i$, $k_i=0$ or 1, and then computing products of $m\times 2^i$ matrices with $2^i \times n$ matrices for each i such that $k_i=1$ results in a small number of multiplications. However, in the 8×15 by 15×8 case such an algorithm requires $64+100+182+343 = 689$ multiplications, whereas simply computing the 8×16 by 16×8 case directly requires only 686. Thus a savings can sometimes be achieved by embedding in larger matrices.

Even computing the $m \times 2^k$ by $2^k \times n$ case presents difficulties. For m and n even, one would substitute 2×2 matrices into an $\frac{m}{2} \times 2^{k-1}$ and $2^{k-1}\times \frac{n}{2}$ matrix. If either of m or n is odd one would compute the extra row or column separately. But even this does not result in the smallest number of multiplications. For example, to compute the 8×8 by 8×7 case would require $7 \times 40 + 64 = 344$ multiplications whereas embedding in the 8×8 by 8×8 case requires only 343 multiplications.

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 o 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 (i.e. $j<i$) or an input variable. Certain f_i will be considered to be output variables.

In proving that certain matrix multiplication algorithms are optimal with respect to the number of multiplications, we will assume that the elements of the matrix are binary numbers and addition and multiplication are modulo two. We can do this since, given an algorithm that works for integers, by taking all constants modulo two we have an algorithm for binary numbers with no more multiplications. Thus the optimal algorithm for matrix multiplication with integer elements has at least as many multiplications as for binary. The converse may not be true since $2a \cdot b$ requires one multiplication in the integer case but is identically zero in the binary case.

We are also able to assume that algorithms are of a certain form. Namely, the only multiplications that occur are multiplications of two linear sums of the input variables. The reason for this is as follows. First assume a constant appears in a multiplication such as $(c + f_i)f_j$. The result can be written as $f_i \cdot f_j + f_j$. Thus without loss of generality we can assume that constant terms do not appear in any multiplicand.

Assume that a previously computed product appears in another product. Then a cubic term will appear in the product. Since no cubic term appears in the result, if we delete all terms other than quadratic, we will not change the result. Since no constant term appears in any element of a product, deleting all non-linear terms in the multiplicands leaves all and only the quadratic terms in the result.

Since we do not have commutativity we can assume a more restricted general form for the optimal program. The general form of the optimal algorithm computes each y_{ij} as a linear sum of m_i's where each m_i is of the form $\alpha \cdot \beta$, where α is a linear sum of a_{ij}'s and β is a linear sum of x_{ij}'s. We denote by $\mathcal{L}\{a_{ij}\}$ the set of all linear functionals of a_{ij}'s.

We now state three results which will be needed later.

Let $\mathcal{F}$ be a field and $\{a_1, a_2, \ldots, a_n\}$ a set of indeterminants. Let $\hat{\mathcal{F}}$ be the field obtained by extending $\mathcal{F}$ by rational expressions of the a_i's.

A set of vectors $x_1, x_2, \ldots, x_p$ with elements from $\hat{\mathcal{F}}$ are non-dependent 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's are not restricted to $\mathcal{F}$ but can be arbitrary elements of $\hat{\mathcal{F}}$.

Theorem 2: (Winograd) Let A be an m×n matrix whose elements are from $\hat{\mathcal{F}}$, and let $\underline{x}$ be an arbitrary n-vector. If A has p nondependent columns, then any algorithm computing A$\underline{x}$ requires at least p multiplications or divisions.

Lemma 4: Let $F = \{f_1, \ldots, f_k, \ldots, f_n\}$ be a set of expressions, where $f_1, \ldots, f_k$ 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 $f_1, \ldots, f_k$.

Proof Let $m_1, \ldots, m_p$ be the multiplications of some algorithm for F. Since $f_1, \ldots, f_k$ are independent we can solve for k of the m_i's in terms of $f_1, \ldots, f_k$ and the rest of the m_i's.

Lemma 5 Any program for

$$\left.\begin{array}{l} a_{21}x_{2i} \\ \\ a_{21}x_{1i} + a_{22}x_{2i} \end{array}\right\} \quad 1 \leq i \leq n$$

requires 3n multiplications.

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

In matrix form this can be represented as

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

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

In what follows we will make use of the fact that after computing a set of expressions, we can with no additional multiplications compute all linear sums of the expressions.

<u>Theorem 3:</u> Let A and B be two sets of expressions where each expression in B is some linear combination of the expressions in A. If the computation of B requires p multiplications, then the computation of A requires p multiplications.

Proof Suppose there is an algorithm for A containing fewer than p multiplications. Then this algorithm can be used to compute B with fewer than p multiplications simply by appending the necessary additions to compute the linear combinations.

We now develop the concepts necessary to take advantage of the high degree of symmetry in the matrix multiplication problem. A transformation T is a mapping T: $\{a_{ij}\} \cup \{x_{k\ell}\} \to \mathcal{L}[\{a_{ij}\} \cup \{x_{k\ell}\}]$. $A \times X$ is said to be invariant under a transformation T if $T(\mathcal{L}\{A \times X\}) = \mathcal{L}\{A \times X\}$. An example of such a transformation is

$$
\begin{array}{ll}
a_{1j} \to a_{1j} + a_{2j} & 1 \le j \le 2 \\
a_{ij} \to a_{ij} & 2 \le i \le n,\ 1 \le j \le 2 \\
x_{ij} \to x_{ij} & 1 \le i \le 2,\ \ 1 \le j \le n
\end{array}
$$

The set $\mathcal{T}$ of all transformations which leave $A \times X$ invariant forms a group. Thus, we can use $\mathcal{T}$ to define an equivalence relation on multiplications as follows: If $m_1 = \alpha.\beta$ and $m_2 = \gamma.\delta$ are two multiplications and there exists some T in $\mathcal{T}$ such that $T(\alpha) = \gamma$, then $m_1 \equiv m_2$. Since $\mathcal{T}$ is a group, $\equiv$ as defined above is, in fact, an equivalence relation. Under this equivalence relation, the multiplications used in the computation of $A \times X$ form n equivalence classes where n is the dimension of the largest square matrix contained in A. Representative members of these classes are $a_{11}\ \beta$, $(a_{11}+a_{22})\beta,\ldots,(a_{11}+a_{22}+\ldots+a_{nn})\beta$, for any β in $\mathcal{L}\{x_{ij}\}$.

The next technique, similar to one used by Pan [1], is to reduce the number of variables by selecting a linear sum of variables $x_1+x_2+\ldots+x_n$ which appears as a multiplicand and setting the sum equal to zero. This can be done by solving the resulting linear equation for one of the variables, say x_1, and substituting the solution $x_1 = x_2 + \ldots+x_n$ for that variable whenever it occurs in the algorithm. The result is a new algorithm P' which has at least one less multiplication than the original algorithm P. If P computed $f_i(x_1,x_2,\ldots,x_n)$, $1\le i\le m$, then P' computes $f'_i(x_2,\ldots,x_n) = f_i(x_2+\ldots+x_n,x_2,\ldots,x_n)$, $1\le i\le m$. We conclude that the optimal algorithm for the f's has at least one more multiplication than the optimal algorithm for the f' 's.

Combing this technique with the symmetry argument results in a powerful tool. For example, in computing A×X where A and X are both 2 × 2 matrices, we can assume by symmetry that either a_{11} or $a_{11} + a_{22}$ appears as a multiplicand. Thus, by substituting first $a_{11}=0$ and then $a_{11} = a_{22}$, we deduce that the minimal number of multiplications for the above computation is more than the minimal number for one of

$$\begin{array}{llll} & \text{a)} & a_{12}x_{21} & a_{12}x_{22} \\ & & a_{21}x_{11}+a_{22}x_{21} & a_{21}x_{12}+a_{22}x_{22} \\ \text{or} & \text{b)} & a_{22}x_{11}+a_{12}x_{21} & a_{22}x_{12}+a_{12}x_{22} \\ & & a_{21}x_{11}+a_{22}x_{21} & a_{21}x_{12}+a_{22}x_{22} \end{array}$$

We now make use of these techniques to develop several technical lemmas which will be used to show that multiplying a 2×2 matrix A by a $2 \times n$ matrix X requires exactly $\left\lceil \frac{7n}{2} \right\rceil$ multiplications.

<u>Lemma 6:</u> 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.

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

$$\left.\begin{array}{l} a_{21}x_{2i} \\ a_{21}x_{1i} + a_{22}x_{2i} \end{array}\right\} \quad 1 \leq i \leq n$$

which requires 3n multiplications by Lemma 5.

<u>Corollary:</u>

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 7: Any algorithm P for $A \times X$ which has k multiplications of type $a_{11}\alpha$, $a_{12}\beta$, and $(a_{11}+a_{12})\gamma$ has at least $3n + \frac{k}{2}$ 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 $S' = \{a_{11}x_{1i} + a_{12}x_{2i} | 1 \le i \le n\}$ 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 $\frac{k}{2}$, since any $\frac{\ell}{2}$ linearly independent expressions in $S \cap M$ can be represented as the product of a matrix with ℓ non-dependent columns and an arbitrary vector and thus requires ℓ multiplications by Theorem 2. Hence, the vector space spanned by $S \cup M$ has dimension of at least $n + \frac{k}{2}$. We can therefore solve for $n + \frac{k}{2}$ 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 $n + \frac{k}{2}$ multiplications from P.

The new algorithm computes $a_{21}x_{1i}+a_{22}x_{2i}$ $1\leq i\leq n$ which requires 2n multiplications by Theorem 2. Hence P must have had at least $3n + \frac{k}{2}$ 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 $3n + \frac{k'}{2}$ multiplications as we have just proved. Thus P must have had at least $3n + \frac{k'}{2} + k - k' > 3n+\frac{k}{2}$ 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 4. 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 $\left\lceil\frac{7n}{2}\right\rceil$ 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 6 and one group of Lemma 7, while each multiplication is S_B belongs to three groups of Lemma 6.

Suppose there are m_A multiplications in S_A and m_B multiplications in S_B. Then there must be at least $(m_A + 3m_B)/9$ multiplications in some group of Lemma 6 and at least $m_A/3$ multiplications in some group of Lemma 7.

Suppose P has at least m multiplications. Since $m = m_A+m_B$, then either $m_A \geq 6m/7$ or $m_B \geq m/7$. If $m_A \geq 6m/7$, then there must be some group of Lemma 7 with at least $m_A/3 \geq 2m/7$ multiplications, and hence by Lemma 7 $m \geq 3n + m/7$. If $m_B \geq m/7$, then there must be some group of Lemma 6 with at least $(m_A+3m_B)/9 = ((m-m_B)+3m_B)/9 \geq m/7$ multiplications and hence by Lemma 6 $m \geq 3n + \frac{m}{7}$. In either case, therefore, $m \geq 3n+\frac{m}{7}$. Solving this inequality, we find that $m \geq \frac{7n}{2}$.

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.

<u>Theorem 5:</u> The optimal algorithm for matrix multiplication with non-commutative elements has $\lceil \frac{7n}{2} \rceil$ multiplications for the 2×2 times $2 \times n$ case (and similarly for the $n \times 2$ times 2×2 case).

We now proceed with a series of lemmas which will provide the basis for proving that any algorithm for the 3×2 times 2×3 case requires at least 15 multiplications, and hence the algorithm of Theorem 1 is optimal for this case.

<u>Lemma 8:</u> Any algorithm for the 3×2 times 2×3 case with a multiplication equivalent (by a transformation in $\mathcal{T}$) to $a_{31}\beta$ has at least 15 multiplications.

<u>Proof:</u> Assume that the algorithm has only 14 multiplications. Assume without loss of generality that $a_{31}\beta$ is present. Set $a_{31}=0$.

Then
$$\left.\begin{array}{c} a_{11}x_{1j}+a_{12}x_{2j} \\ a_{21}x_{1j}+a_{22}x_{2j} \\ a_{32}x_{2j} \end{array}\right\} \quad 1\leq j\leq 3$$
must be computed in 13 multiplications. By Lemma 4 we can assume that the multiplications $a_{32}x_{2j}$, $1\leq j\leq 3$, are present. Set $a_{32} = 0$ and we have an algorithm computing $a_{i1}x_{1j}+a_{i2}x_{2j}$, $1\leq i\leq 2$, $1\leq j\leq 3$, in 10 multiplications contradicting Theorem 4.

<u>Lemma 9:</u> If an algorithm for the 3×2 by 2×3 case has two multiplications of type $(a_{11}+a_{22})\alpha$ and $(a_{12}+a_{21})\beta$, then the algorithm has at least 15 multiplications.

<u>Proof:</u> Set $a_{11}=a_{22}$ and $a_{12} = a_{21}$, losing two multiplications. Without loss of generality we can assume that $a_{32}\beta$ is present, because any multiplication involving a_{32} is equivalent to $a_{32}\beta$ by some transformation in $\mathcal{T}$. Set $a_{32} = 0$, losing one multiplication. By Lemma 4 setting $a_{31}=0$ costs three multiplications. It can be shown that what remains requires

9 multiplications, and therefore 9+3+1+2 multiplications were required originally.

Corollary: By transformations we obtain similar theorems for the groups $a_{11}+a_{22}$ and $a_{11}+a_{21}+a_{22}$, and $a_{11}+a_{22}$ and $a_{11}+a_{12}+a_{22}$.

Lemma 10: If an algorithm for the 3×2 by 2×3 case has two multiplications of the type $(a_{11}+a_{22})\alpha$, $(a_{11}+a_{12}+a_{21})\beta$ and $(a_{12}+a_{21}+a_{22})\gamma$, then the algorithm has at least 15 multiplications.

Proof: Set $a_{11}=a_{22}$ and $a_{12}=a_{21}+a_{22}$, losing two multiplications. In the algorithm so obtained some multiplication must involve a_{22}; let it be $(a_{22}+\alpha)\beta$ where α is one of eight possible cases. For each case it can be shown that after setting $a_{22}=\alpha$, which costs one multiplication, the remainder requires 12 multiplications. Hence the original algorithm must have had at least 12+1+2 = 15 multiplications.

Lemma 11. If an algorithm for the 3×2 by 2×3 case has two multiplications of type $(a_{11}+a_{22})\alpha$, $(a_{11}+a_{12}+a_{22})\beta$, $(a_{12}+a_{21}+a_{22})\gamma$, $(a_{12}+a_{21})\delta$, $(a_{11}+a_{12}+a_{21})\varepsilon$ and $(a_{11}+a_{21}+a_{22})\zeta$, then the algorithm has at least 15 multiplications.

Proof: Without loss of generality assume the first of the two multiplications is $(a_{11}+a_{22})\alpha$. If the second multiplication is $(a_{11}+a_{22})\beta$, $(a_{11}+a_{12}+a_{21})\beta$

or $(a_{12}+a_{21}+a_{22})\beta$, then the algorithm has at least 15 multiplications by Lemma 10. If the second multiplication is $(a_{12}+a_{21})\beta$, $(a_{11}+a_{21}+a_{22})\beta$ or $(a_{11}+a_{12}+a_{22})\beta$, then the algorithm has at least 15 multiplications by Lemma 9 and its corollary.

Corollary: The Lemma holds for seven groups of six multiplications each, which covers all 42 types of multiplications not equivalent to $a_{31}\beta$.

Lemma 12: If an algorithm for the 3×2 by 2×3 case has eight or more multiplications not equivalent to $a_{31}\beta$, then the algorithm has at least 15 multiplications.

Proof: At least one group of the corollary to Lemma 11 must have two or more multiplications, and the proof follows immediately from Lemma 11.

Theorem 6: Any algorithm for the 3×2 by 2×3 case requires 15 multiplications.

Proof: The algorithm requires at least 11 multiplications to compute the first two rows by Theorem 5. If one of these is equivalent to $a_{31}\beta$, then the theorem is true by Lemma 8. If none of the multiplications is equivalent to $a_{31}\beta$, then the theorem is true by Lemma 12.

A detailed case analysis of the 3×2 by 2×4 case seems to indicate that 19 multiplications are required rather than the 20 of Theorem 1. However, we conjecture that 20 are required and 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. A method due to Robert Wagner [4] will be described which computes $A \times X$, where A is an $m \times 2$ matrix and X is a $2 \times n$ matrix, using only $mn+m+n$ multiplications.

Wagner's method proceeds as follows:

1) Compute the m products $a_{i1}a_{i2}$, $1 \leq i \leq m$.

2) Compute the n products $x_{1j}x_{2j}$, $1 \leq j \leq n$.

3) Compute the mn products $(a_{i1}+x_{2j})(a_{i2}+x_{1j})$, $1 \leq i \leq m, 1 \leq j \leq n$.

4) If $A \times X = Y$, then compute the y_{ij}'s from the above products by $y_{ij} = (a_{i1}+x_{2j})(a_{i2}+x_{1j}) - a_{i1}a_{i2} - x_{1j}x_{2j}$.

Clearly this algorithm requires just $mn+m+n$ multiplications. If we set $m=2$ and compare this with the minimal bound for non-commutative multiplication, we find that Wagner's method requires fewer multiplications for all $n \geq 5$. Thus we have the following theorem:

Theorem 7: The minimum number of multiplications necessary for matrix multiplication is less for the commutative case than for the non-commutative case.

References

[1] V. Ya. Pan, "Methods of Computing Values of Polynomials," Russian Math. Surveys 21, pp. 105-136 (1966).

[2] S. Winograd, "On the Number of Multiplications Required to Compute Certain Functions," Proc. National Academy of Sciences 58:5, pp. 1840-1842 (1967).

[3] Volker Strassen, "Gaussian Elimination is not Optimal," unpublished manuscript, Angewandte Mathematik der Universitat Zurich, December 1968.

[4] Robert Wagner, private correspondence.