NORTH-HOLLAND

Fast Rectangular Matrix Multiplication and QR Decomposition

Philip A. Knight*

Department of Mathematics
University of Strathclyde
Richmond Street
Glasgow
G1 1XH, U.K.

Submitted by Richard A. Brualdi

ABSTRACT

In the last twenty-five years there has been much research into "fast" matrix multiplication methods: ones that have an asymptotically smaller operation count than conventional multiplication. Most fast methods are derived for square matrices, but they can be applied to rectangular matrices by a blocking technique. We obtain an expression for the order of the operation count for this blocked multiplication of rectangular matrices. We derive an exact operation count for Strassen's method with rectangular matrices and determine the recursion threshold that minimizes the operation count. We also show that when Strassen's method is used to multiply rectangular matrices it is more efficient to use the method on the whole product than to apply the method to square submatrices. Fast multiplication methods can be exploited in calculating a QR decomposition of an $m \times n$ matrix. We show that the operation count can be reduced from $O(mn^2)$ to $O(mn^{1+(1/(4-\alpha))})$ by using a fast multiplication method with exponent α in conjunction with Bischof and Van Loan's WY representation of a product of Householder transformations.

*E-mail: `P.A.Knight@strath.ac.uk`. Supported by an SERC research studentship.

0024-3795/95/$9.50
SSDI 0024-3795(93)00230-W

1. INTRODUCTION

In the past twenty-five years a number of algorithms have been developed that theoretically improve on the standard time needed for matrix multiplication, usually by reducing the exponent of the number of operations needed. (Throughout this paper an operation represents any one of the scalar operations $+, -, \times$, and $\div$.) One of the first published achievements in the field, and the most significant result in fast matrix multiplication, was the discovery by Strassen in 1969 [19] of a method for generating the product of two $n \times n$ matrices in $O(n^{\omega})$ operations, where $\omega = \log_2 7 \approx 2.807$. This is the most widely used method, in current applications, of those that reduce the exponent of the number of operations in matrix multiplication and is generally regarded as the only one that gives any useful improvements in practical computation [2, 11], but recent work by Laderman, Pan, and Sha [13] provides a new approach that could produce further speedups for modestly sized matrices. We devote a later section to an analysis of Strassen's algorithm.

Further improvements to the exponent have been achieved using bilinear and trilinear algorithms. With these one can show (see, e.g., [14]) that if there is a method for computing AB (where A is an $m \times n$ matrix and B is $n \times p$) in K multiplications, then for all choices of r there is an algorithm for computing the product of two $r \times r$ matrices in fewer than Cr^{ω} multiplications, where C is a constant independent of r and $\omega = 3 \log_{mnp} K$. Strassen's method is an example of this result when $m = n = p = 2$ and $K = 7$, as we shall see later. Pan was the first person to employ this method successfully, and in 1978 he reduced the exponent to 2.795. This was achieved by analyzing a method that can multiply together two $n \times n$ matrices using $\frac{1}{3}n^3 + 6n^2 - \frac{4}{3}n$ multiplications. In particular, one can multiply together two 70×70 matrices in 143,640 multiplications, and this gives us the exponent for Pan's method.

Other techniques that can be used to reduce the exponent include bilinear λ-algorithms, where methods that evaluate part of the product of rectangular matrices can be used to calculate the product of square matrices efficiently, as well as generalized tensor products [14]. The minimum exponent so far discovered is 2.376 by Coppersmith and Winograd [9], although the authors express optimism that the theoretical minimum of 2 can be attained.

Results on the computational complexity of fast matrix multiplication methods are usually given only for square matrices, but in this paper we give a result which shows the connection between the order of a method for square and rectangular matrices. We also derive exact expressions for the operation counts for rectangular matrix multiplications with Strassen's method.

In 1973, Schönhage [15, 16] gave algorithms that exploit fast matrix multiplication methods to reduce the asymptotic order of the QR decomposition of a matrix, but as these require explicit formation of the orthogonal matrix Q, any possible benefits from reducing the exponent are swamped by the number of extra operations that must be performed [12]. In this paper we consider a blocked form of QR decomposition developed by Bischof and Van Loan [4] in which fast matrix multiplication can be exploited. Using our results for rectangular multiplication, we show that the order of operations for the method can be reduced through careful choice of the block size.

2. FAST MATRIX MULTIPLICATION

2.1. Rectangular Matrices

The "speed" of a fast technique is usually measured by the order of the operation count for the multiplication of two $n \times n$ matrices (theoretically, a method with a smaller order will be faster asymptotically, i.e., for sufficiently large dimension n). It is straightforward to evaluate the algebraic complexity of rectangular matrix multiplication from this value, but this does not seem to have been noted before in the literature. Suppose that the method can perform the multiplication of two $n \times n$ matrices using $O(n^\alpha)$ operations, where $2 < \alpha < 3$. If A and B are $m \times n$ and $n \times p$ matrices respectively, then the product AB can be formed in $O(n_1^{\alpha-2} n_2 n_3)$ operations, where $n_1 = \min(m, n, p)$ and n_2, n_3 are the other two dimensions.

To see this, consider the case when m is the smallest dimension of A and B, and suppose $n = jm$ and $p = km$ for some integers j and k. Then the multiplication can be split into $m \times m$ blocks:

$$AB = (A_1 \quad A_2 \quad \cdots \quad A_j) \begin{pmatrix} B_{11} & \cdots & B_{1k} \\ \vdots & & \vdots \\ B_{j1} & \cdots & B_{jk} \end{pmatrix},$$

which involves a total of jk multiplications of $m \times m$ matrices, each involving $O(m^\alpha)$ operations. Thus the total number of operations is $O(jkm^\alpha)$ or $O(m^{\alpha-2}np)$, as required, and we can show similar results for the cases when n and p are the smallest dimensions. The figure of $O(n_1^{\alpha-2} n_2 n_3)$ is optimal for this scheme of splitting the multiplication, and it appears to be optimal over all splittings.

2.2. Strassen's Method

Strassen originally derived his algorithm for square matrix multiplication, but it is not difficult to generalize to the rectangular case, and this was first done by Brent [7]. Consider the product C of two matrices A and B that have dimensions $2^m \times 2^n$ and $2^n \times 2^p$ respectively. We can partition A, B, and C into four equally sized blocks

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix},$$

and then Strassen's method can be written accordingly:

$$\begin{aligned} P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}), & P_2 &= (A_{21} + A_{22})B_{11}, \\ P_3 &= A_{11}(B_{12} - B_{22}), & P_4 &= A_{22}(B_{21} - B_{11}), \\ P_5 &= (A_{11} + A_{12})B_{22}, & P_6 &= (A_{21} - A_{11})(B_{11} + B_{12}), \end{aligned}$$

$$P_7 = (A_{12} - A_{22})(B_{21} + B_{22}),$$

$$\begin{aligned} C_{11} &= P_1 + P_4 - P_5 + P_7, & C_{12} &= P_3 + P_5, \\ C_{21} &= P_2 + P_4, & C_{22} &= P_1 + P_3 - P_2 + P_6. \end{aligned}$$

Since the blocks A_{ij}, B_{ij} are matrices whose dimensions are powers of 2, we can compute the products $P_1, \dots, P_7$ using the same algorithm, and we can carry on the recursion until one of the dimensions of the blocks to be multiplied is 1. Alternatively, we can carry on the recursion to a certain level and then form the remaining products using conventional multiplication. We are not restricted to using Strassen's algorithm for matrices whose dimensions are a power of 2, and we can modify the method in a number of ways to cope with any odd dimensions that are encountered [11].

Winograd derived a variation of the formula that uses 15 matrix additions at each level of recursion rather than 18 [6]. However, this method has a weaker error bound [3], and so we shall not consider this variant any further.

A question remains as to the technique that should be used when multiplying rectangular matrices. For example, when we use Strassen's method we can divide the problem into square matrix multiplications, as we illustrated in Section 2.1, or we can use the algorithm on the whole system and recur with rectangular matrices until the smallest dimension involved reaches the threshold. Before we address this question, we derive expressions for the operation count for Strassen's algorithm with rectangular matrices.

2.3. Operation Counts for Strassen's Method

From Section 2.1 we know the order of the operation count for multiplying an $m \times n$ and $n \times p$ matrix together by Strassen's method. It is also known [11] that to multiply two square matrices of dimension 2^k using Strassen's method requires

$$7^{k-r}(2 \times 8^r + 5 \times 4^r) - 6 \times 4^k$$

operations, where 2^r is the threshold such that conventional multiplication is used when matrices of dimension $\leq 2^r$ are involved (the cutoff point).

We now extend this result to obtain the number of operations involved in the multiplication of rectangular matrices. Upper bounds have been given for this previously [18], but here we give an exact figure. Suppose we have matrices of dimensions $2^m \times 2^n$ and $2^n \times 2^p$. Let $2^m = a2^j$, $2^n = b2^j, 2^p = c2^j$, where $j = \min(m, n, p)$ (so $a, b, c \geq 1$ and at least one of a, b, c is equal to 1). Let $S_R(M, N, P)$ denote the number of operations involved in using Strassen's method to multiply matrices of dimensions $M \times N$ and $N \times P$, where R is the cutoff point [i.e., we stop recursion when $\min(M, N, P) \leq R$]. Also, let $A(M, N)$ be the number of operations involved in adding together two $M \times N$ matrices [therefore $A(M, N) = MN$]. Then, by considering the formulae for Strassen's method in Section 2.2 we get the recurrence relation

$$\begin{aligned} &S_{2^r}(a2^{j+1}, b2^{j+1}, c2^{j+1}) \\ &\quad = 7S_{2^r}(a2^j, b2^j, c2^j) + 5A(a2^j, b2^j) + 5A(b2^j, c2^j) + 8A(a2^j, c2^j) \\ &\quad = 7S_{2^r}(a2^j, b2^j, c2^j) + (5ab + 5bc + 8ac)4^j. \end{aligned}$$

This implies that we can write $S_{2^r}(a2^j, b2^j, c2^j)$ in the form $\alpha 7^j + \beta 4^j$. Substituting this into the above expression, we have

$$\alpha 7^{j+1} + \beta 4^{j+1} = 7\alpha 7^j + 7\beta 4^j + (5ab + 5bc + 8ac)4^j,$$

from which we find that

$$\beta = -\tfrac{1}{3}(5ab + 5bc + 8ac).$$

We also know that

$$S_{2^r}(a2^r, b2^r, c2^r) = 2abc8^r - ac4^r,$$

because at this stage we use conventional multiplication, and from this we deduce

$$\alpha = 2abc\left(\tfrac{8}{7}\right)^r + \tfrac{5}{3}(ab + bc + ac)\left(\tfrac{4}{7}\right)^r.$$

TABLE 1. VALUE OF α VERSUS RECURSION THRESHOLD

r	Case 1: $b > 1, c > 1$	Case 2: $b = 1, c > 1$
0	$\frac{1}{3}[11bc + 5(b+c)]$	$\frac{1}{3}(16c + 5)$
1	$\frac{1}{21}[68bc + 20(b+c)]$	$\frac{1}{21}(88c + 20)$
2	$\frac{1}{147}[464bc + 80(b+c)]$	$\frac{1}{147}(544c + 80)$
3	$\frac{1}{1029}[3392bc + 320(b+c)]$	$\frac{1}{1029}(3712c + 320)$
4	$\frac{1}{7203}[25856bc + 1280(b+c)]$	$\frac{1}{7203}(27136c + 1280)$

Note that α is symmetric in a, b, c. To minimize the number of operations involved with Strassen's method we should minimize α by choosing the appropriate positive integer value for r (assuming j is large enough to make the β term negligible). We know that at least one of a, b, c equals 1, and since α is symmetric, we can assume $a = 1$ without loss of generality. We also see that there is no reason why we should assume that b and c are powers of 2. If we stop recursion when the smallest dimension reaches a certain threshold, then the only condition that b and c need satisfy is that they are positive integers. There are now three cases to consider.

Case 1: $b > 1, c > 1$. Since b and c are greater than 1 and are integers, we know that $b, c \geq 2$. We see from Table 1 that we shall minimize α by choosing either $r = 2$ or $r = 3$. If $r = 2$ then

$$\alpha_2 = \tfrac{464}{147}bc + \tfrac{80}{147}(b+c) \approx 3.16bc + 0.54(b+c),$$

whereas if we choose $r = 3$ then

$$\alpha_3 = \tfrac{3392}{1029}bc + \tfrac{320}{1029}(b+c) \approx 3.30bc + 0.31(b+c).$$

Clearly, as b and c get larger, the term multiplying bc will dominate, but we can find small values for b and c for which $\alpha_3 < \alpha_2$. This will occur if

$$\tfrac{3392}{1029}bc + \tfrac{320}{1029}(b+c) < \tfrac{464}{147}bc + \tfrac{80}{147}(b+c),$$

which simplifies to

$$b < \frac{5c}{3c-5},$$

and this inequality is true if and only if $c = 2$, $b \leq 9$; $c = 3$, $b \leq 3$; or $4 \leq c \leq 9$, $b = 2$.

Case 2: $b = 1$, $c > 1$. Here only c remains, and to minimize α we

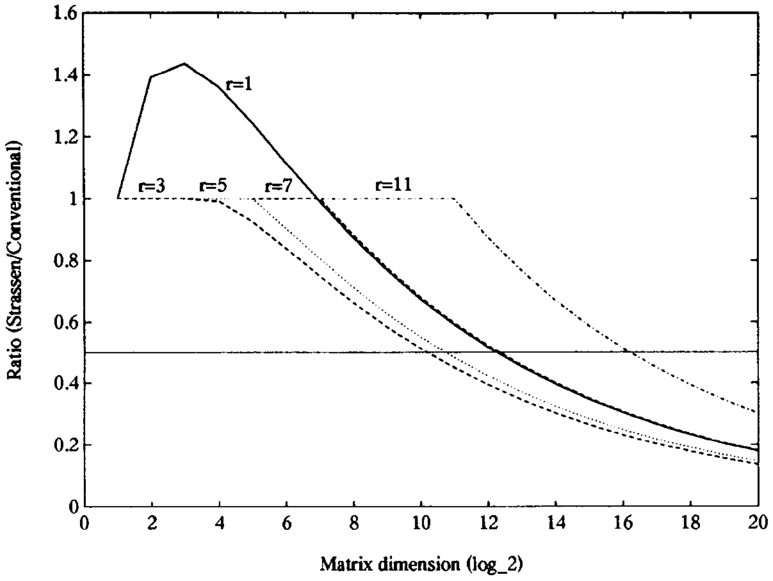

FIG. 1. A comparision of operation counts for matrix multiplication.

see from Table 1 that we need to pick $r = 3$, for which

$$\alpha = \tfrac{3712}{1029}c + \tfrac{320}{1029} \approx 3.61c + 0.31.$$

Case 3: $b = 1, c = 1$. Here we have the special case of multiplication of square matrices, and we can use

$$S_{2^r}(2^k, 2^k, 2^k) = 7^{k-r}(2 \times 8^r + 5 \times 4^r) - 6 \times 4^k.$$

By examining this formula we find that it is minimized by letting $r = 3$ (agreeing with [11]), which gives $\alpha = 192/49 \approx 3.92$. Figure 1 compares operation counts for $n \times n$ matrices using conventional multiplication and using Strassen's method for a variety of cutoff points 2^r.

In summary, for the multiplication of matrices whose dimensions are $m \times n$ and $n \times p$ with $m = a2^j$, $n = b2^j$, $p = c2^j$ we have

$$\begin{aligned} S_{2^r}(m,n,p) &= \alpha 7^j + \beta 4^j, \\ \alpha &= 2abc\left(\tfrac{8}{7}\right)^r + \tfrac{5}{3}(ab + bc + ac)\left(\tfrac{4}{7}\right)^r, \\ \beta &= -\tfrac{1}{3}(5ab + 5bc + 8ac). \end{aligned}$$

The values of r that minimize α are given in the three cases above. If we let $r = 0$ and let $a = 1$, we find that

$$\begin{aligned} \alpha 7^j &= \left[2bc + \tfrac{5}{3}(b + bc + c)\right] 7^j \\ &= O(m^{\omega-2}np), \end{aligned}$$

where $\omega = \log_2 7$, so we recover the result from Section 2.1.

In practice we may have to perform extra operations that we have not considered here. For example, when coding Strassen's algorithm in MATLAB we found we used more operations when calculating the indices of the submatrices during recursion. These extra operations can be incorporated into our recurrence relation, but we have not done this here, as they depend on the programming language used.

We are now in a position to answer the question we posed at the end of the last section, namely: What technique should we use when multiplying together two rectangular matrices? Consider the case when we have matrices with dimensions 2^m, 2^n, and 2^p, and suppose $m < n, p$. If we choose to split the problem into square matrix multiplications, then the total number of operations (to highest order terms) will be at least

$$3.92 \times 7^m 2^{n+p-2m},$$

since we must perform 2^{n+p-2m} multiplications of $2^m \times 2^m$ matrices, whilst if we choose to carry out our recursion using rectangular blocks, we can carry out our task in

$$3.16 \times 2^{n+p-2m} 7^m + 0.54(2^{n-m} + 2^{p-m})7^m$$

operations (by picking $r = 2$ in case 1). As $n-m$ and $p-m$ increase, the first term of this expression becomes dominant and we see that asymptotically we can reduce the number of operations in the multiplication by $[100(3.92-3.16)/3.92]\% \approx 20\%$ if we work with rectangular blocks.

To conclude this section we review some of the performance results that have been published for Strassen's method. Bailey et al. [2] mention that for some modern workstations (e.g., Sun-4 and Silicon Graphics IRIS 4D) Strassen's method is faster than conventional multiplication for 16×16 matrices, which we see from Figure 1 is the smallest possible size that can offer improvements in the operation count; but for most machines it is necessary to use larger matrices before performance improvements can be seen. For example, Brent [7] managed to reduce the time involved for conventional multiplication of 110×110 matrices by using an Algol-W version of Strassen's method with just one level of recursion on an IBM 360/67, but he achieved bigger speedups for matrices whose dimensions were less than 300 by using a method developed by Winograd [20]. Bailey [1] has achieved speedups of 45% for 128×128 matrices using a recursion threshold of 127 on a Cray-2, whereas the optimum operation count (using a threshold of 8) is only 25% less than that for conventional multiplication—35% of this speedup comes from Cray-specific techniques. Bjørstad et al. [5] have parallelized Strassen's method on the MasPar MP-1 and have achieved impressive speedups over conventional parallel block techniques using a recursion threshold of 128. These last three examples illustrate the fact that on most machines overheads from sources such as data transfer mean that we are unlikely to optimize the time needed to multiply matrices by minimizing the number of operations involved.

An important application of matrix multiplication is in the level 3 BLAS, a set of matrix-matrix linear algebra operations. Higham [11] has shown how Strassen's method can be used to produce level 3 BLAS routines whose operation counts have smaller exponents than for the conventional methods. In the rest of this paper we consider ways of exploiting fast matrix multiplication techniques to develop fast methods for orthogonal decomposition.

3. FAST BLOCK *QR* DECOMPOSITION

3.1. Block QR Decomposition

Bischof and Van Loan [4] have developed a method for computing the QR decomposition of a matrix based on a new way to represent products of Householder matrices. This representation leads to an algorithm that is rich in level 3 BLAS operations, which makes it ideal to use in combination with fast matrix multiplication techniques. We will show that by choosing the appropriate block size for the particular multiplication method used, a new algorithm can be produced that has a lower order operation count than standard QR decomposition techniques.

The key to Bischof and Van Loan's method is the so-called WY representation for the product of Householder matrices. This gives an expression for the product of k Householder matrices of the form $I - 2u_iu_i^T/(u_i^Tu_i)$ ($u_i \in \mathbf{R}^m$ is a nonzero vector) that can be written as $Q_k = I + W_kY_k^T$, where W_k and Y_k are both $m \times k$ matrices of rank k. The WY form clearly exists for the case $k = 1$, since we can choose

$$W_1 = -\frac{2u_1}{u_1^Tu_1}, \qquad Y_1 = u_1,$$

and we can show by induction that

$$W_k = [W_{k-1} \quad Q_{k-1}u_k], \qquad Y_k = [Y_{k-1} \quad u_k].$$

Thus the WY block QR algorithm for decomposing an $m \times n$ matrix in n_0 stages with block size p can be written as follows (assuming n_0, $p \in \mathbf{Z}$ and $n_0 p = n$):

```
for $k = 1 : n_0$
    $s = (k-1)p + 1$
    Use standard $QR$ decomposition to upper-triangularize
    $A(s:m, s:s+p-1)$
    and to generate Householder vectors $u_s, \dots, u_{s+p-1}$
    $Y = u_s$, $W = -2u_s/(u_s^T u_s)$
    for $j = s+1 : s+p-1$
        $z = -2(I + WY^T)u_j/(u_j^T u_j)$, $Y = [Y \ u_j]$, $W = [W \ z]$
    end
    $A(s:m, s+p:n) = (I + WY^T)^T A(s:m, s+p:n)$ (#)
end
```

It is with this final stage (#) that fast matrix multiplication techniques can be exploited.

The WY method has been developed further through a compact representation [17] that reduces the amount of storage space needed for the Householder factors. We do not consider this algorithm here, because our aim is to try to minimize operation counts for the QR decomposition. Studying the effect of the different factors on the operation count, we find that the algebraic complexity of the compact method will be the same as for the noncompact one, but the number of operations involved will be bigger.

3.2. A Fast Method

Suppose we wish to compute the decomposition of an $m \times n$ matrix A ($m \geq n$), using the block method in n_0 stages with block size $p = n/n_0 \in \mathbf{Z}$, and suppose also that we can multiply two $N \times N$ matrices in $O(N^\alpha)$ operations. We assume for convenience that both m/p and n/p are integers. This makes the calculations neater, and it does not affect the final result, because if p is not a factor of m or n, we can simply pad the matrix with zeros until it is, and this will not change the order of the operation count of the algorithm. At each stage $k = 1, 2, \dots, n_0 - 1$ we must compute the factors W and Y and apply them to a block $A^{\#}$ of the current decomposition of A. This is achieved by forming the product

$$(I + WY^T)^T A^{\#} = A^{\#} + YW^T A^{\#}.$$

At stage k, W and Y are $[m-(k-1)\,n/n_0]\times n/n_0$ matrices, while $A^{\#}$ is an $[m-(k-1)\,n/n_0]\times(n-kn/n_0)$ matrix. Thus, using our fast technique, the first multiplication, $M_1 = W^T A^{\#}$, requires $O(n_1^{\alpha-2}n_2n_3)$ operations, where $n_1 = n/n_0$, $n_2 = n - kn/n_0$, and $n_3 = m-(k-1)n/n_0$ (see Section 2.1), as does the second multiplication $M_2 = YM_1$. This means that at each stage $k = 1, 2, \ldots, n_0 - 1$ we have approximately

$$c\Big(\frac{n}{n_0}\Big)^{\alpha}\Big(\frac{mn_0}{n}+1-k\Big)(n_0-k)$$

operations, for some constant c.

Summing over k and considering only the highest order terms, we obtain a figure of

$$\frac{c}{2}n^{\alpha-1}n_0^{3-\alpha}\Big(m-\frac{n}{3}\Big)$$

operations (a detailed derivation is given in [12]).

The generation of W and Y needs a total of $(4mn^2-2n^3)/n_0$ scalar operations [4]. Thus, in total, we have approximately

$$Z = m\Big(\frac{c}{2}n^{\alpha-1}n_0^{3-\alpha}+4n^2n_0^{-1}\Big)-n\Big(\frac{c}{6}n^{\alpha-1}n_0^{3-\alpha}+2\frac{n^2}{n_0}\Big)$$

operations.

Now suppose $n_0 = n^{\beta}$ for some $\beta \in [0,1]$; then

$$Z = m\Big(\frac{c}{2}n^{\alpha-1+3\beta-\alpha\beta}+4n^{2-\beta}\Big)-n\Big(\frac{c}{6}n^{\alpha-1+3\beta-\alpha\beta}+2n^{2-\beta}\Big).$$

To minimize the order of Z we require β^* such that

$$\alpha-1+3\beta^*-\alpha\beta^* = 2-\beta^*,$$

giving us

$$\beta^* = 1-\frac{1}{4-\alpha}.$$

For example, when $n = 1000$ and $\alpha = \log_2 7$ (i.e., for Strassen's method), $\beta^* \approx 0.162$, and therefore in this particular case the number of stages that minimizes the order of the method is $n_0 = n^{\beta^*} \approx 3$, and the optimum block size $p \approx 330$. Now,

$$2-\beta^* = 1+\frac{1}{4-\alpha},$$

so the number of operations involved in the computation is, at best,

$$c_1n^{1+(1/(4-\alpha))}(m-c_2n) \tag{3.1}$$

for certain constants c_1 and c_2. For example, by replacing conventional multiplication with Strassen's method we can reduce the complexity of the calculation from $O(mn^2)$ to $O(mn^{1.838})$.

Conventional QR decomposition, using Householder rotations, requires $2n^2(m - n/3)$ operations. With the fast methods currently available the constant c_1 in (3.1) is always considerably bigger than 2; hence for the new method to be faster than normal (purely in terms of the number of operations), a large matrix must be factorized. This is tempered to some extent by a number of considerations. The optimum block size, in terms of algebraic complexity, does not minimize the number of operations required by block QR decomposition until the matrices involved are enormous (see [12] for details). The consequence of this is that we can improve on the operation count in the case of smaller matrices by choosing a block size according to different criteria. For example, we could look at some specific matrices and from these choose the ratio of block size to matrix dimension that approximately optimizes the operation count emprically. Secondly, the factored form of the orthogonal matrix Q is in block form too. This can be exploited to achieve further potential speedups by using the fast multiplication techniques when forming products of blocks in the solution of least squares problems. A further source of possible improvement is that the block nature of the algorithm could be exploited in parallelizing the method, as it can when conventional multiplication is used [4, 5].

I thank Nick Higham for his guidance during my research and his advice during the production of this document, and the referee for helpful suggestions on improvements.

REFERENCES

1 D. H. Bailey, Extra high speed matrix multiplication on the Cray-2, *SIAM J. Sci. Statist. Comput.* 9:603–607 (1988).

2 D. H. Bailey, K. Lee, and H. D. Simon, Using Strassen's algorithm to accelerate the solution of linear systems, *J. Supercomputing* 4:357–371 (1991).

3 D. Bini and D. Lotti, Stability of fast algorithms for matrix multiplication, *Numer. Math.* 36:63–72 (1980).

4 C. Bischof and C. F. Van Loan, The WY representation for products of Householder matrices, *SIAM J. Sci. Statist. Comput.* 8:s2–s13 (1987).

5 P. Bjørstad, F. Manne, T. Sørevik, and M. Vajteršic, Efficient matrix multiplication on SIMD computers, *SIAM J. Matrix Anal. Appl.* 13:386–401 (1992).

6 G. Brassard and P. Bratley, *Algorithmics: Theory and Practice*, Prentice-Hall, Englewood Cliffs, N.J., 1988.
7 R. P. Brent, Algorithms for Matrix Multiplication, Technical Report CS 157, *Computer Science Dept., Stanford Univ.* 1970.
8 R. P. Brent, Error analysis of algorithms for matrix multiplication and triangular decomposition using Winograd's identity, *Numer. Math.* 16:145–156 (1970).
9 D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progression, in *19th Annual ACM Symposium on Theory of Computing*, pp. 1–6, 1987.
10 G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins U.P., 1989.
11 N. J. Higham, Exploiting fast matrix multiplication within the level 3 BLAS, *ACM Trans. Math. Software*, 16:352–368 (1990).
12 P. A. Knight, Exploiting Fast Matrix Multiplication, M.Sc. Thesis, Univ. of Manchester, 1990.
13 J. Laderman, V. Y. Pan, and Xuan-He Sha, On practical algorithms for accelerated matrix multiplication, *Linear Algebra Appl.* 162–164:557–588 (1992).
14 V. Y. Pan, How can we speed up matrix multiplication? *SIAM Rev.* 26:393–415 (1984).
15 A. Schönhage, Unitäre Transformationen grosser Matrizen, *Numer. Math.* 20 (1973).
16 A. Schönhage, Fast Schmidt orthogonalization and unitary transformations of large matrices, in *Complexity of Sequential and Parallel Numerical Algorithms* (J. F. Traub, Ed.), Academic, 1973, pp. 283–291.
17 R. Schreiber and C. F. Van Loan, A storage-efficient WY representation for products of Householder transformations, *SIAM J. Sci. Statist. Comput.* 10:53–57 (1989).
18 J. Spiess, Untersuchungen des Zeitgewinns durch neue Algorithmen zur Matrix-Multiplikation, *Computing*, 17:23–36 (1976).
19 V. Strassen, Gaussian elimination is not optimal, *Numer. Math.* 13:354–356 (1969).
20 S. Winograd, A new algorithm for inner product, *IEEE Trans. Comput.* C-17:693–694 (1968).

Received 24 August 1992; final manuscript accepted 23 November 1992