is true by showing that an algorithm to multiply matrices can be used to obtain a matrix vector product. (The two problems are the same, of course, if \( q = 1 \).) No known matrix multiplication algorithm does only \( mn \) *'s. However, there are algorithms that, for large matrices, do significantly fewer multiplications and \( \pm \)'s than Winograd's.

### 7.3.4 Strassen's Matrix Multiplication

For the remainder of this section we assume that the matrices to be multiplied are \( n \times n \) square matrices, \( A \) and \( B \). The key to Strassen's algorithm is a method to multiply \( 2 \times 2 \) matrices using seven multiplications instead of the usual eight. (Winograd's algorithm also uses eight.) For \( n = 2 \), first compute the following seven quantities, each of which requires exactly one multiplication:

\[
\begin{align*}
x_1 &= (a_{11} + a_{22}) \cdot (b_{11} + b_{22}) \\
x_2 &= (a_{21} + a_{22}) \cdot b_{11} \\
x_3 &= a_{11} \cdot (b_{12} - b_{22}) \\
x_4 &= a_{22} \cdot (b_{21} - b_{11}) \\
x_5 &= (a_{11} + a_{12}) \cdot b_{22} \\
x_6 &= (a_{21} - a_{11}) \cdot (b_{11} + b_{12}) \\
x_7 &= (a_{11} - a_{22}) \cdot (b_{21} + b_{22})
\end{align*}
\]

(7.2)

Let \( C = AB \). The entries of \( C \) are:

\[
\begin{align*}
c_{11} &= a_{11} \cdot b_{11} + a_{12} \cdot b_{21} \\
c_{12} &= a_{11} \cdot b_{12} + a_{12} \cdot b_{22} \\
c_{21} &= a_{21} \cdot b_{11} + a_{22} \cdot b_{21} \\
c_{22} &= a_{21} \cdot b_{12} + a_{22} \cdot b_{22}
\end{align*}
\]

They are computed as follows:

\[
\begin{align*}
c_{11} &= x_1 + x_4 - x_5 + x_7 \\
c_{12} &= x_3 + x_5 \\
c_{21} &= x_2 + x_4 \\
c_{22} &= x_1 + x_3 - x_5 + x_6
\end{align*}
\]

(7.3)

Thus \( 2 \times 2 \) matrices can be multiplied using seven multiplications and 18 additions. It is critical to Strassen's algorithm that commutativity of multiplication is not used in the formulas in Eq. 7.2, so that they can be applied to matrices whose components are also matrices. Let \( n \) be a power of two. Strassen's method consists of partitioning \( A \) and \( B \) each into four \( n/2 \times n/2 \) matrices as shown in Fig. 7.4 and multiplying them using the formulas in Eqs. 7.2 and 7.3; the formulas are used recursively to multiply the component matrices. Before considering extensions for the case when \( n \) is not a power of 2, we compute the number of multiplications and \( \pm \)'s done.

Suppose that \( n = 2^k \) for some \( k \geq 0 \). Let \( M(k) \) be the number of multiplications (of the underlying matrix components, i.e., real numbers) done by Strassen's method for \( n \times n \) matrices. Then, since the formulas of Eq. 7.2 do seven multiplications of \( 2^{k-1} \times 2^{k-1} \) matrices,

\[
M(0) = 1 \\
M(k) = 7M(k-1) \quad \text{for } k > 0.
\]

This recurrence relation is very easy to solve. \( M(k) = 7^k \), and \( 7^k = \tilde{n}^{2.81} = n^{\log_7 7} \approx n^{2.81} \). Thus the number of multiplications is in \( o(n^3) \).

Let \( P(k) \) be the number of \( \pm \)'s done. Clearly \( P(0) = 0 \). There are 18 \( \pm \)'s in the formulas of Eqs. 7.2 and 7.3, so \( P(1) = 18 \). For \( k \geq 1 \), multiplying \( 2^k \times 2^k \) matrices involves 18 additions of \( 2^{k-1} \times 2^{k-1} \) matrices, plus all the \( \pm \)'s done by the seven matrix multiplications in Eqs. 7.2. So

\[
P(0) = 0 \\
P(k) = 18(2^{k-1})^2 + 7P(k-1) \quad \text{for } k > 0.
\]

Expanding the recurrence relation to see what the terms look like gives

\[
P(k) = 18(2^{k-1})^2 + 7P(k-1) = 18(2^{k-1})^2 + 7 \cdot 18(2^{k-2})^2 + 7^2P(k-2)
\]

\[
= 18(2^k)^2 + 7 \cdot 18(2^{k-2})^2 + 7^2 \cdot 18(2^{k-3})^2 + 7^3P(k-3).
\]

Therefore

\[
P(k) = \sum_{i=0}^{k-1} 7^i 18(2^{k-i-1})^2 = 18(2^k)^2 \sum_{i=0}^{k-1} \frac{7^i}{4} = \frac{9}{2} (2^k)^2 \sum_{i=0}^{k-1} \left( \frac{7}{4} \right)^i
\]

\[
= \frac{9}{2} (2^k)^2 \left( 1 - \left( \frac{7}{4} \right)^{k-1} \right) = \frac{9}{2} (2^k)^2 \left( \frac{7^k}{4^k} - 1 \right) = 6 \cdot 7^k - 6 \cdot 4^k
\]

\[
= 6n^{2.81} - 6n^2,
\]

which is also in \( o(n^3) \).

If \( n \) is not a power of 2, some extension of Strassen's algorithm must be used and more work will be done. There are two simple approaches, both of which can be very slow. The first possibility is to add extra rows and columns of zeros to make the dimension a power of 2. The second is to use Strassen's formulas as long as the dimension of the matrices is even and then use the usual algorithm when the dimension is odd. Another, more complicated possibility is to modify the algorithm so that at each level of the recursion, if the matrices to be multiplied have odd dimension, one extra row and one extra column are added. Strassen described a fourth strategy, one that combines the advantages of the first two. The matrices are embedded in larger ones with dimension \( 2^m m \), where \( k = \lceil \log m - 4 \rceil \) and \( m = \lfloor n/2^k \rfloor + 1 \). Strassen's formulas are used recursively until the matrices to be multiplied are \( m \times m \); then the usual method is applied. The total number of arithmetic operations done on the matrix entries will be less than \( 4.7n^{1.87} \).

**Figure 7.4** Partitioning for Strassen's matrix multiplication.
Table 7.1
Comparison of matrix multiplication methods for $n\times n$ matrices.

<table>
<thead>
<tr>
<th></th>
<th>The usual algorithm</th>
<th>Vinograd’s algorithm</th>
<th>Strassen’s algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplications</td>
<td>$n^3$</td>
<td>$\frac{1}{3}n^3 + n^2$</td>
<td>$7^2 = n^{2.81}$, where $n = 2^k$</td>
</tr>
<tr>
<td>Additions/subtractions</td>
<td>$n^4 - n^2$</td>
<td>$\frac{3}{2}n^3 + 2n^2 - 2n$</td>
<td>$6 \cdot 7^4 - 6 \cdot 4^4 = 6n^{2.81} - 6n^2$, where $n = 2^k$</td>
</tr>
<tr>
<td>Total</td>
<td>$2n^3 - n^2$</td>
<td>$2n^3 + 3n^2 - 2n$</td>
<td>$4.7n^{2.81} = 4.7n^{2.81}$ (n need not be a power of 2)</td>
</tr>
</tbody>
</table>

Table 7.1 compares the numbers of arithmetic operations done by the three matrix multiplication methods for $n\times n$ matrices. For large $n$, Strassen’s algorithm does fewer multiplications and fewer $+$’s than either of the other methods. In practice, however, it is not a very good algorithm. Because of its recursive nature, implementing this algorithm would require a lot of bookkeeping that would be very slow and/or complicated. The other, much simpler algorithms will be more efficient for moderate-size $n$.

The primary importance of Strassen’s algorithm is that it broke the $\Theta(n^3)$ barrier for matrix multiplication and the $\Theta(n^2)$ barrier for a number of other matrix problems. These problems, which include matrix inversion, computing determinants, and solving systems of simultaneous linear equations, have well-known $\Theta(n^3)$ solutions, but since they can be reduced to matrix multiplication, they too can be solved in $O(n^{10})$ time. Strassen’s result has been improved upon several times in recent years. There is now a matrix multiplication algorithm with running time in $O(n^{2.376})$. It still remains for very practical algorithms with complexity in $o(n^2)$ to be developed. The lower bound of $n^2$ multiplications has not been increased; whether or not matrix multiplication can be done in $\Theta(n^2)$ steps is still an open question.

*7.4
The Fast Fourier Transform
and Convolution

7.4.1 Introduction

Let $U$ and $V$ be $n$-vectors with components indexed from 0 to $n-1$. The convolution of $U$ and $V$, denoted $U \ast V$, is, by definition, an $n$-vector $W$ with components $w_i = \sum_{j=0}^{n-1} u_j v_{i-j}$, where $0 \leq i \leq n-1$ and the indexes on the right-hand side are taken modulo $n$. For example, for $n = 5$, ...