```
return
END
```

```
SUBROUTINE rotate(r,qt,n,np,i,a,b)
INTEGER n,np,i
REAL a,b,r(np,np),qt(np,np)
```

Given n×n matrices r and qt of physical dimension np, carry out a Jacobi rotation on rows i and $i+1$ of each matrix. a and b are the parameters of the rotation: $\cos\theta = a/\sqrt{a^2+b^2}$, $\sin\theta = b/\sqrt{a^2+b^2}$.

```
INTEGER j
REAL c,fact,s,w,y
if(a.eq.0.)then                  Avoid unnecessary overflow or underflow.
    c=0.
    s=sign(1.,b)
else if(abs(a).gt.abs(b))then
    fact=b/a
    c=sign(1./sqrt(1.+fact**2),a)
    s=fact*c
else
    fact=a/b
    s=sign(1./sqrt(1.+fact**2),b)
    c=fact*s
endif
do 11 j=i,n                      Premultiply r by Jacobi rotation.
    y=r(i,j)
    w=r(i+1,j)
    r(i,j)=c*y-s*w
    r(i+1,j)=s*y+c*w
enddo 11
do 12 j=1,n                      Premultiply qt by Jacobi rotation.
    y=qt(i,j)
    w=qt(i+1,j)
    qt(i,j)=c*y-s*w
    qt(i+1,j)=s*y+c*w
enddo 12
return
END
```

We will make use of QR decomposition, and its updating, in §9.7.

CITED REFERENCES AND FURTHER READING:

Wilkinson, J.H., and Reinsch, C. 1971, *Linear Algebra*, vol. II of *Handbook for Automatic Computation* (New York: Springer-Verlag), Chapter I/8. [1]

Golub, G.H., and Van Loan, C.F. 1989, *Matrix Computations*, 2nd ed. (Baltimore: Johns Hopkins University Press), §§5.2, 5.3, 12.6. [2]

2.11 Is Matrix Inversion an N^3 Process?

We close this chapter with a little entertainment, a bit of algorithmic prestidigitation which probes more deeply into the subject of matrix inversion. We start with a seemingly simple question:

How many individual multiplications does it take to perform the matrix multiplication of two 2×2 matrices,

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \tag{2.11.1}$$

Eight, right? Here they are written explicitly:

$$\begin{aligned} c_{11} &= a_{11} \times b_{11} + a_{12} \times b_{21} \\ c_{12} &= a_{11} \times b_{12} + a_{12} \times b_{22} \\ c_{21} &= a_{21} \times b_{11} + a_{22} \times b_{21} \\ c_{22} &= a_{21} \times b_{12} + a_{22} \times b_{22} \end{aligned} \tag{2.11.2}$$

Do you think that one can write formulas for the c's that involve only *seven* multiplications? (Try it yourself, before reading on.)

Such a set of formulas was, in fact, discovered by Strassen [1]. The formulas are:

$$\begin{aligned} Q_1 &\equiv (a_{11} + a_{22}) \times (b_{11} + b_{22}) \\ Q_2 &\equiv (a_{21} + a_{22}) \times b_{11} \\ Q_3 &\equiv a_{11} \times (b_{12} - b_{22}) \\ Q_4 &\equiv a_{22} \times (-b_{11} + b_{21}) \\ Q_5 &\equiv (a_{11} + a_{12}) \times b_{22} \\ Q_6 &\equiv (-a_{11} + a_{21}) \times (b_{11} + b_{12}) \\ Q_7 &\equiv (a_{12} - a_{22}) \times (b_{21} + b_{22}) \end{aligned} \tag{2.11.3}$$

in terms of which

$$\begin{aligned} c_{11} &= Q_1 + Q_4 - Q_5 + Q_7 \\ c_{21} &= Q_2 + Q_4 \\ c_{12} &= Q_3 + Q_5 \\ c_{22} &= Q_1 + Q_3 - Q_2 + Q_6 \end{aligned} \tag{2.11.4}$$

What's the use of this? There is one fewer multiplication than in equation (2.11.2), but *many more* additions and subtractions. It is not clear that anything has been gained. But notice that in (2.11.3) the a's and b's are never commuted. Therefore (2.11.3) and (2.11.4) are valid when the a's and b's are themselves matrices. The problem of multiplying two very large matrices (of order $N = 2^m$ for some integer *m*) can now be broken down recursively by partitioning the matrices into quarters, sixteenths, etc. And note the key point: The savings is not just a factor "7/8"; it is that factor at *each* hierarchical level of the recursion. In total it reduces the process of matrix multiplication to order $N^{\log_2 7}$ instead of N^3.

What about all the extra additions in (2.11.3)–(2.11.4)? Don't they outweigh the advantage of the fewer multiplications? For large N, it turns out that there are six times as many additions as multiplications implied by (2.11.3)–(2.11.4). But, if N is very large, this constant factor is no match for the change in the *exponent* from N^3 to $N^{\log_2 7}$.

With this "fast" matrix multiplication, Strassen also obtained a surprising result for matrix inversion [1]. Suppose that the matrices

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \tag{2.11.5}$$

are inverses of each other. Then the c's can be obtained from the a's by the following operations (compare equations 2.7.22 and 2.7.25):

$$\begin{aligned}
R_1 &= \text{Inverse}(a_{11}) \\
R_2 &= a_{21} \times R_1 \\
R_3 &= R_1 \times a_{12} \\
R_4 &= a_{21} \times R_3 \\
R_5 &= R_4 - a_{22} \\
R_6 &= \text{Inverse}(R_5) \\
c_{12} &= R_3 \times R_6 \\
c_{21} &= R_6 \times R_2 \\
R_7 &= R_3 \times c_{21} \\
c_{11} &= R_1 - R_7 \\
c_{22} &= -R_6
\end{aligned} \tag{2.11.6}$$

In (2.11.6) the "inverse" operator occurs just twice. It is to be interpreted as the reciprocal if the a's and c's are scalars, but as matrix inversion if the a's and c's are themselves submatrices. Imagine doing the inversion of a very large matrix, of order $N = 2^m$, recursively by partitions in half. At each step, halving the order *doubles* the number of inverse operations. But this means that there are only N divisions in all! So divisions don't dominate in the recursive use of (2.11.6). Equation (2.11.6) is dominated, in fact, by its 6 multiplications. Since these can be done by an $N^{\log_2 7}$ algorithm, so can the matrix inversion!

This is fun, but let's look at practicalities: If you estimate how large N has to be before the difference between exponent 3 and exponent $\log_2 7 = 2.807$ is substantial enough to outweigh the bookkeeping overhead, arising from the complicated nature of the recursive Strassen algorithm, you will find that LU decomposition is in no immediate danger of becoming obsolete.

If, on the other hand, you like this kind of fun, then try these: (1) Can you multiply the complex numbers $(a+ib)$ and $(c+id)$ in only *three* real multiplications? [Answer: see §5.4.] (2) Can you evaluate a general fourth-degree polynomial in

x for many different values of x with only *three* multiplications per evaluation? [Answer: see §5.3.]

CITED REFERENCES AND FURTHER READING:

Strassen, V. 1969, *Numerische Mathematik*, vol. 13, pp. 354–356. [1]

Kronsjö, L. 1987, *Algorithms: Their Complexity and Efficiency*, 2nd ed. (New York: Wiley).

Winograd, S. 1971, *Linear Algebra and Its Applications*, vol. 4, pp. 381–388.

Pan, V. Ya. 1980, *SIAM Journal on Computing*, vol. 9, pp. 321–342.

Pan, V. 1984, *How to Multiply Matrices Faster*, Lecture Notes in Computer Science, vol. 179 (New York: Springer-Verlag)

Pan, V. 1984, *SIAM Review*, vol. 26, pp. 393–415. [More recent results that show that an exponent of 2.496 can be achieved — theoretically!]