Library

123138

MET O 11 TECHNICAL NOTE NO. 72

A Plus-one Algorithm for Matrix Multiplication

R DIXON

1. Introduction

Almost everybody knows how to multiply two matrices together to get the ordinary matrix product, and on the face of it there appears to be little scope for innovation in the general case. Nevertheless, the demands made on computer science have led to quite a variety of different ways of carrying out matrix multiplication. The variant described in this note is aimed at avoiding the housekeeping arithmetic which has to be done when using the customary approach. This housekeeping arithmetic either has to be explicitly written into the program in the customary method or else the compiler arranges for it to be done. Either way it has to be paid for.

The Plus-one method is worth examination because it may well be advantageous in a low-level language on the IBM 195, and it may have some relevance to a vector processor machine since it does effect a kind of vectorization of the process of matrix multiplication.

FX3B

2. Basic Theory

It is first necessary to set forth the connections between the handling of arrays considered as matrices and dealt with according to the usual matrix symbolism and rules for matrix manipulations and the handling of arrays considered as dyadics and dealt with using the symbols and formalism of the standard Gibbs notation for vectors.

Using 2 x 2 arrays to keep down the amount of writing, let us start with two matrices A and X

$$A = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}, X = \begin{pmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{pmatrix} \qquad (1)$$

The ordinary matrix product Y, usually denoted in texts by AX is

$$Y = AX = \begin{bmatrix} (a_{00}\, x_{00} + a_{01}\, x_{10}) & (a_{00}\, x_{01} + a_{01}\, x_{11}) \\ (a_{10}\, x_{10} + a_{11}\, x_{10}) & (a_{10}\, x_{01} + a_{11}\, x_{11}) \end{bmatrix} \qquad (2)$$

by the usual rule for ordinary matrix multiplication.

Now express the arrays A and X as dyadics, using two orthogonal unit vectors $\underline{i}$ and $\underline{j}$. We then have

$$A = \begin{pmatrix} a_{00}\underline{ii} & a_{01}\underline{ij} \\ a_{10}\underline{ji} & a_{11}\underline{jj} \end{pmatrix}, X = \begin{pmatrix} x_{00}\underline{ij} & x_{01}\underline{ij} \\ x_{10}\underline{ji} & x_{11}\underline{jj} \end{pmatrix} \qquad (3)$$

Note that the insertion of the unit dyads $\underline{ii}$, $\underline{ij}$, $\underline{ji}$ and $\underline{jj}$ does of itself completely specify the kind of geometrical entity involved. It is not, as with matrix notation, actually necessary to write the dyadics down as rectangular arrays, nor is it necessary to stick to any conventional order. A can just as well be written as

$$A = a_{00}\underline{ii} + a_{01}\underline{ij} + a_{10}\underline{ji} + a_{11}\underline{jj} \qquad (4)$$

or

$$A = a_{01}\underline{ij} + a_{11}\underline{jj} + a_{00}\underline{ii} + a_{01}\underline{ij} \qquad (5)$$

or

$$A = \begin{pmatrix} a_{11}\underline{jj} & a_{10}\underline{ji} \\ a_{00}\underline{ii} & a_{01}\underline{ij} \end{pmatrix} \qquad (6)$$

It makes no difference. A in (3), (4), (5) and (6) is the same dyadic. The product of the dyadics A and X which corresponds to the matrix product (2) is the scalar product Y = A•B where • is the Gibbs symbol for the scalar product. Thus, using the form (4) for both A and X

$$Y = (a_{00}\underline{ii} + a_{01}\underline{ij} + a_{10}\underline{ji} + a_{11}\underline{jj})\bullet(x_{00}\underline{ii} + x_{01}\underline{ij} + x_{10}\underline{ji} + x_{11}\underline{jj}) \qquad (7)$$

from which, taking into account the simple rules

$$\underline{i}\bullet\underline{i} = \underline{j}\bullet\underline{j} = 1 \qquad (8)$$

$$\underline{i}\bullet\underline{j} = \underline{j}\bullet\underline{i} = 0 \qquad (9)$$

there results

$$Y = (a_{00}x_{00} + a_{01}x_{10})\underline{ii} + (a_{00}x_{01} + a_{01}x_{11})\underline{ij} + (a_{10}x_{10} + a_{11}x_{10})\underline{ji} + (a_{10}x_{01} + a_{11}x_{11})\underline{jj} \qquad (10)$$

and this is the same as (2) since (2) is simply (10) written as an array in a conventional order with the unit dyads suppressed.

In matrix notation another type of product which has found increasing use over recent years is the direct product, also called the Kronecker product. It is denoted by $\otimes$ and it is defined by

$$A \otimes X = a_{ij}X \qquad i,j = 0,1 \tag{11}$$

Thus for the two matrices in (1) the direct product is, specifically

$$A \otimes X = \begin{pmatrix} a_{00}X & a_{01}X \\ a_{10}X & a_{11}X \end{pmatrix} \tag{12}$$

i.e.

$$A \otimes X = \begin{pmatrix} a_{00}x_{00} & a_{00}x_{01} & a_{01}x_{00} & a_{01}x_{01} \\ a_{00}x_{10} & a_{00}x_{11} & a_{01}x_{10} & a_{01}x_{11} \\ a_{10}x_{00} & a_{10}x_{01} & a_{11}x_{00} & a_{11}x_{01} \\ a_{10}x_{10} & a_{10}x_{11} & a_{11}x_{10} & a_{11}x_{11} \end{pmatrix} \tag{13}$$

Notice that $A \otimes X$ is a 4 x 4 matrix, and in general if A is m x n and X is s x t then $A \otimes X$ is ms x nt. This stepping-up of the size of the matrix has a significance which is not always appreciated if an exclusively matricial view of the matter is taken.

Viewing A and X now as dyadics, as in (3), the product corresponding to the direct matrix product (11) is usually denoted by **AX** in bold Clarendon type and is given by

$$\mathbf{AX} = (a_{00}\underline{ii} + a_{01}\underline{ij} + a_{10}\underline{ji} + a_{11}\underline{jj})(x_{00}\underline{ii} + x_{01}\underline{ij} + x_{10}\underline{ji} + x_{11}\underline{jj}) \tag{14}$$

(14) is (7) without the scalar product $\bullet$ between the brackets, and it is evaluated to give

$$\begin{aligned} \mathbf{AX} = {} & a_{00}x_{00}\underline{iiii} + a_{00}x_{01}\underline{iiij} + a_{00}x_{10}\underline{iiji} + a_{00}x_{11}\underline{iijj} \\ & + a_{01}x_{00}\underline{ijii} + a_{01}x_{01}\underline{ijij} + a_{01}x_{10}\underline{ijji} + a_{01}x_{11}\underline{ijjj} \\ & + a_{10}x_{00}\underline{jiii} + a_{10}x_{01}\underline{jiij} + a_{10}x_{10}\underline{jiji} + a_{10}x_{11}\underline{jijj} \\ & + a_{11}x_{00}\underline{jjii} + a_{11}x_{01}\underline{jjij} + a_{11}x_{10}\underline{jjji} + a_{11}x_{11}\underline{jjjj} \end{aligned} \tag{15}$$

The expression (15) could, were it possible, have been typed out in one long line with the terms in any order. No confusion can arise. The individual tetrads unequivocally determine what is meant by the collection of terms in (15) however they are set out. In particular the terms in (15) can be set out so that the subscripts occur in the same pattern as in (13). We then have, with the plus signs understood.

$$\mathbf{AX} = \begin{pmatrix} a_{00}x_{00}\underline{iiii} & a_{00}x_{01}\underline{iiij} & a_{01}x_{00}\underline{ijii} & a_{01}x_{01}\underline{ijij} \\ a_{00}x_{10}\underline{iiji} & a_{00}x_{11}\underline{iijj} & a_{01}x_{10}\underline{ijji} & a_{01}x_{11}\underline{ijjj} \\ a_{10}x_{00}\underline{jiii} & a_{10}x_{01}\underline{jiij} & a_{11}x_{00}\underline{jjii} & a_{11}x_{01}\underline{jjij} \\ a_{10}x_{10}\underline{jiji} & a_{10}x_{11}\underline{jijj} & a_{11}x_{10}\underline{jjji} & a_{11}x_{11}\underline{jjjj} \end{pmatrix} \tag{16}$$

It is now clear that (13) is simply the product (14) evaluated and written out in the order (16) but with the unit tetrads suppressed. Then in any subsequent manipulations (13) is treated as a larger dyadic, with the unit dyads suppressed. This brings out an important difference between the matricial and Gibbsian formalisms. The Gibbs notation makes it explicitly clear that <u>the result of directly multiplying two dyadics is a different geometrical entity, namely a tetradic</u>. This aspect of the matter is easily lost in matrix notation. At any ordinary level of derivation and manipulation this does not matter at all. As long as the matrix rules are stuck to the answer will come out right. But there are situations where the triadic or tetradic nature of the algebra is very relevant to the physical circumstances of the problem and in such cases the Gibbsian formalism is advantageous.

3. <u>A recursive form of the scalar product Y = A X B</u>

It will be noticed that if the scalar product indicating $\bullet$ is placed between the second and third unit vectors in each tetrad in (16) or (15) and the indicated operation is carried out according to (8) and (9) then (15) and (16) reduce to (10). Thus the usual scalar product of two matrices is a particular function of the tetradic quantity which results from taking the direct product. We can thus invent a bit of notation which makes this explicitly clear, and instead of writing the scalar product Y of A and X as $Y = A \bullet X$ we can write it as

$$Y = \underset{23}{\bullet}(A \otimes X) \tag{17}$$

where $\underset{23}{\bullet}$ means that the scalar product $\bullet$ is to be placed between the second and third unit vectors of each tetrad in $(A \otimes X)$ and the indicated operation carried out. In the parlance of yet another formalism, namely tensor analysis, this is said to be contraction on the second and third indices.

If now a third matrix (dyadic) B is introduced where

$$B = \begin{pmatrix} b_{00}\underline{ii} & b_{01}\underline{ij} \\ b_{10}\underline{ji} & b_{11}\underline{jj} \end{pmatrix} \tag{18}$$

then the matrix product

$$Y = \quad AXB \tag{19}$$

is the same thing as

$$Y = \quad A \bullet X \bullet B \tag{20}$$

To evaluate this in Gibbsian style, note that (20) is the same as

$$Y = \quad (A \bullet X) \bullet B \tag{21}$$

and that $(A \bullet X)$ is already given by (10). All that remains to do is to take

$$\bullet\,(b_{00}\underline{ii} + b_{01}\underline{ij} + b_{10}\underline{ji} + b_{11}\underline{jj}) \tag{22}$$

through (10) from the right, whereupon taking (8) and (9) into account again there results

$$Y = [b_{00}(a_{00}x_{00} + a_{01}x_{10}) + b_{10}(a_{00}x_{01} + a_{01}x_{11})]\underline{ii}[b_{01}(a_{00}x_{00} + a_{01}x_{10}) + b_{11}(a_{00}x_{01} + a_{01}x_{11})]\underline{i}$$

$$[b_{00}(a_{10}x_{01} + a_{11}x_{10}) + b_{10}(a_{10}x_{01} + a_{11}x_{11})]\underline{ji}[b_{01}(a_{10}x_{01} + a_{11}x_{11}) + b_{11}(a_{10}x_{01} + a_{11}x_{11})]\underline{j}$$

(23)

and of course this is exactly what is obtained (with the unit dyads suppressed) if (19) is worked out purely matricially.

Just as the scalar product of two matrices or dyadics can be represented in the style (17) as a function of a tetradic, so the scalar product of three matrices or dyadics, (19) or (20), can be represented as a function of a sexadic. The direct product of A, X, and B is $A \otimes X \otimes B$ and it is the sum of 64 sexads of which a typical one is, for example, $a_{01}x_{10}b_{11}\underline{ijjijj}$. It is then found that (20) is the same as

$$Y = \underset{23}{\bullet}\,\underset{45}{\bullet}\,(A \otimes X \otimes B) \qquad (24)$$

which means - in each of the 64 sexads insert a scalar product • between the second and third unit vectors and between the fourth and fifth unit vectors and apply (8) and (9). Doing this reduces the sexads to dyads, most of them vanish because of (9) and in fact the only terms which survive are the four dyads which appear in (23).

It is also found that (20) can be written as

$$Y = \underset{23}{\bullet}\,(A \otimes (\,\underset{23}{\bullet}\,(X \otimes B))) \qquad (25)$$

(25) is equivalent to (24) and (20). It must be understood to imply the following sequence of operations - first form the tetradic $(X \otimes B)$ of 16 terms, then take the scalar product of the second and third unit vector in each of the 16 tetrads, thus reducing the tetradic to a dyadic. Then form the direct product of A with this dyadic getting another tetradic, and finally carry out $\underset{23}{\bullet}$ to reduce this tetradic to the final required dyadic.

The significant point about (25) as compared with (2) or (24) is its obviously recursive character. The form (25) will extend to any number of factors. Thus

$$Y = ABCDEFG \quad \text{(in matrix notation)} \qquad (26)$$

may be written as

$$Y = \underset{23}{\bullet}\,(A \otimes (\,\underset{23}{\bullet}\,(B \otimes (\underset{23}{\bullet}\,(- - - - - \underset{23}{\bullet}(F \otimes G)))))))))) \qquad (27)$$

As an example the evaluation of

$$Y = ABCD \quad \text{(matrix notation)} \qquad (28)$$

where A is $l \times m$, B is $m \times n$, C is $n \times p$, and D is $p \times q$

can be effected in Fortran by some coding sequence such as is given on the following page. Programmers/scientists studying this sequence should understand that I am not a programmer and that the sequence is not intended to be anything more than a section of pidgin - Fortran, the intention behind the statements being, hopefully, sufficiently clear for the sequence to serve as an epexegesis to the foregoing algebra.

```
                              JH=1
DO      7Ø                    IM=1, N-1
                              ID=1
                               K=1
DO      3Ø                    IP=1, NA(1)
                               J=JH
DO      4Ø                    IQ=1, NA(IM+2)
                               I=ID
                               A=Ø.Ø
DO      5Ø                    IL=1, NA(IM+1)
                               A=A + H(J) * D(I)
                               I=I + 1
                               J=J + 1
5Ø                            CONTINUE
                            S(K)=A
                               K=K+1
4Ø                            CONTINUE
                              ID=I
3Ø                            CONTINUE
DO      6Ø                    IM=1, NA(1) * NA(IM+2)
                            D(IM)=S(IM)
6Ø                            CONTINUE
                              JH=J
7Ø                            CONTINUE
```

Notes

(a) N is the number of matrices involved, so in this case N=4.

(b) NA is a one-dimensional array containing the N+1 independent parameters q, p, n, m, l, in that order.

(c) Initially the array D contains the elements of the matrix D in (28) stored column by column. Intermediate results are also put into this array column by column. I counts through the elements of whatever vectors are in D.

(d) The index K counts serially through the elements of the intermediate results, column wise.

(e) Here the assumption is that the matrix D has been read into the store by columns and that A, B and C have been read in by rows starting with the first row of C and finishing with the last row of A. This area of store constitutes the array H. The index J counts serially right through this array. It will be apparent that a corresponding routine could be written for the situation where the matrix D is stored by rows and A, B and C are stored by columns. Routines can also be written for the cases where all the matrices involved are stored column-wise or row-wise, but in these cases it is not clear whether there is any advantage in this style of routine.

(f) The salient feature of the above routine is that the string of matrix multiplications is accomplished without doing more than add one to any index. It should be possible to take advantage of this in Assembler code on our present machine, and it may be relevant to the vectorized operations of projected machines.

(g) The copying across done in the DO 6Ø loop is an unwelcome overhead in the above version but in many practical situations it could be avoided.

(h) A competent programmer could clearly rewrite the above section of coding as a SUBROUTINE which would deal with the general case of N matrix factors of any conformable dimensions.

R Dixon

R Dixon
Met O 11
December 1976