CONSIDERATIONS IN MANAGING ARRAYS IN FORTRAN

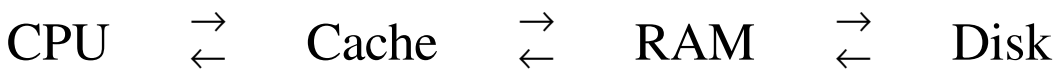
Fortran stores multidimensional arrays in column major order. I.e., the most rapidly varying index is the innermost one. Hence, the elements a_{ij} of an array **A** dimensioned as **A**(N,M) are stored as:

$$a_{11}a_{21}a_{31}\cdots a_{N1}a_{12}a_{22}\cdots a_{N2}\cdots a_{1M}a_{2M}\cdots a_{NM} \tag{1}$$

Note: This is the opposite of the convention used in languages such as C/C++ where arrays are stored in row major order. I.e.,

$$a_{11}a_{12}a_{13}\cdots a_{1M}a_{21}a_{22}\cdots a_{2M}\cdots a_{N1}a_{N2}\cdots a_{NM} \tag{2}$$

Beyond their internal registers, modern computer systems store data in one of 3 types of memory locations: a small cache, typically a few dozen to a few hundred Kb, which is located on the CPU and can be accessed very quickly; RAM, typically several Mb, which is located on the motherboard and can be accessed quickly (about 10 to 100 slower than cache), and disk which is many thousands to times slower than RAM. Data is exchanged between the various types of memory in the units of a page (which is typically about 4Kb).



The hardware and the compiler are both designed to try to keep data being worked on or to be worked on in the near future in the internal registers and cache in order to maximize performance. The hardware and compiler algorithms assume that the data needed for a future operation is likely to be stored adjacent to the memory location being used in the current operation. Clearly then, it is to our advantage to keep the data being operated on close together when array calculations are performed. Hence, if the dimensions of the array **A** are sufficiently large, then the organization of the array becomes important issue in deciding how to best code a matrix multiplication.

Consider multiplying a $M \times L$ dimensional array **B** by $N \times M$ dimensional array **A** to obtain the $N \times L$ dimensional matrix $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$. In general, this operation can be written in Fortran as:

```
do i=1,N
  do j=1,L
    c(i,j)=0.0
  end do
end do
do ...
  do ...
    do ...
      c(i,j)=c(i,j)+a(i,k)*b(k,j)
    end do
  end do
end do
```

where in the last 3 loops we can loop over the indices i,j, and k in any of 6 possible orderings.

Let's see how the arrays are accessed in each ordering:

Method	Type	outer loop	middle loop	inner loop	Access Method		
					A	B	C
1	dot product	i	j	k	row	column	row
2	row oriented	i	k	j	row	row	row
3	dot product	j	i	k	row	column	column
4	matrix sum	k	i	j	column	row	row
5	column oriented	j	k	i	column	column	column
6	matrix sum	k	j	i	column	row	column

Methods 1 and 3 only differ in how the result array, **C**, is accessed, otherwise they correspond in linear algebra to the standard picture of matrix multiplication as a dot product of the rows of **A** with the columns of **B**:

$$c_{ij} = \mathbf{a}_i \cdot \mathbf{b}_j \tag{3}$$

where $\mathbf{a}_i$ is the ith row vector of **A** and $\mathbf{b}_j$ is the jth column vector of **B**.

Method 2 corresponds in linear algebra to the construction of the row vectors of **C** by taking linear combinations of the row vectors of **B** as in:

$$\mathbf{c}_i = \sum_k a_{ik} \mathbf{b}_k \tag{4}$$

where $\mathbf{c}_i$ is the ith row vector of **C** and $\mathbf{b}_k$ is the kth row vector of **B**. *Based upon our previous discussion, this should be the most efficient method for languages such as C/C++ that store arrays in row order.*

Method 5 corresponds in linear algebra to the construction of the column vectors of **C** by taking linear combinations of the column vectors of **A** as in:

$$\mathbf{c}_i = \sum_k \mathbf{a}_k b_{ki} \tag{5}$$

where $\mathbf{c}_i$ is the ith column vector of **C** and $\mathbf{a}_k$ is the kth column vector of **A**. *This should be the most efficient method for languages like Fortran that store arrays in column order.*

Methods 4 and 6 correspond to accumulating the product matrix as a sum of matrices $\mathbf{D}_k$ whose elements are $[\mathbf{D}_k]_{ij} = a_{ik} b_{kj}$, i.e. $\mathbf{D}_k = \mathbf{a}_k \mathbf{b}_k$ where the $\mathbf{a}_k$ are the column vectors of **A** and $\mathbf{b}_k$ are the row vectors of **B** and

$$\mathbf{C} = \sum_k \mathbf{D}_k \tag{6}$$