can be alleviated by using a macro to define the storage-mapping function, such as

\[
\text{#define mat_ptr(r,c) } \\
\quad \left(\ast \text{mat} \_\text{ptr} + \left(\ast \text{r} \right) \ast \text{num} \_\text{cols} + \left(\ast \text{c}\right)\right)
\]

With this, the assignment above can be written as

\[
\text{mat} \_\text{ptr} \left(\text{row}, \text{col}\right) = \text{x};
\]

Other languages deal differently with the problem of passing multidimensional arrays. Ada compilers are able to determine the defined size of dimensions of all arrays that are used as parameters at the time subprograms are compiled. In Ada, unconstrained array types can be formal parameters. An unconstrained array type is one in which the index ranges are not given in the array type definition. Definitions of variables of unconstrained array types must include index ranges. The code in a subprogram that is passed an unconstrained array can obtain the index range information of the actual parameter associated with such parameters. For example, consider the following definition:

\[
\text{type Mat} \_\text{Type is array } \left(\text{Integer range }\leq, \right. \text{ } \\
\quad \left.\text{Integer range }\geq \right) \text{ of Float};
\]

\[
\text{Mat} \_1 : \text{Mat} \_\text{Type}(1..100, 1..20);
\]

A function that returns the sum of the elements of arrays of \text{Mat} \_\text{Type} type follows:

\[
\text{function Sumr(Mat : in Mat} \_\text{Type) return Float is}
\]

\[
\text{Sum : Float := 0.0; begin for Row in Mat'range(1) loop}
\quad \text{for Col in Mat'range(2) loop}
\quad \quad \text{Sum := Sum + Mat(Row, Col);}
\quad \text{end loop; -- for Col ...}
\quad \text{end loop; -- for Row ...}
\quad \text{return Sum; end Sumr;}
\]

The \text{range} attribute returns the subscript range of the named subscript of the actual parameter array, so this works regardless of the size or index ranges of the parameter.

In Fortran, the problem is addressed in the following way. Formal parameters that are arrays must have a declaration in the header. For single-dimensioned arrays, the subscripts in declarations are irrelevant. But for multidimensional array subscripts in such declarations allow the compiler to build