LECTURE 10: Allocation

If the size of a vector or matrix is not known at the beginning of a program, declare it to be allocatable with

- \( \text{dimension}(::) \) for vectors,
- \( \text{dimension}(::,:) \) for matrices.

Then allocate space later in the program (when the dimension for the vector or matrix is known) with

- \( \text{allocate}(v(n)) \) or \( \text{allocate}(v(1:n)) \) for vectors, and
- \( \text{allocate}(a(n,m)) \) for matrices.

For two vectors \( v, w \), \( \text{allocate}(v(n),w(n)) \) allocates \( n \) places for each.

c10.1 The magnitude of a vector \( v = [v_1, v_2, \ldots, v_n] \) if given by \( \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} \). Write a function \( \text{magnitude}(v,n) \) which, given an integer \( n \) and vector \( v \) with \( n \) real numbers, returns the magnitude of \( v \). Then, write a Fortran program which asks for and reads the number \( n \) and allocates space for a vector \( v \). Then it asks for and reads the \( n \) values of vector \( v \) on one line. Finally, it prints: "The magnitude = " followed by the magnitude.

c10.2 Write a real function \( \text{mean}(v,n) \) which, given an integer \( n \) and vector \( v \) with \( n \) real numbers, returns the mean (average) value \( \frac{v_1 + \cdots + v_n}{n} \). Then, write a Fortran program which asks for and reads the number \( n \) and allocates space for a vector \( v \). Then it asks for and reads the \( n \) values of vector \( v \) on one line. Finally, it prints: "The mean = " followed by the mean.

c10.3 Write a real function \( \text{distance}(v,w,n) \) which, given an integer \( n \), and 2 vectors \( v, w \) with \( n \) real numbers, returns the distance between these two vectors given by \( \sqrt{(v_1 - w_1)^2 + \cdots + (v_n - w_n)^2} \). Write a program \( \text{test_distance} \) which uses this function. It asks for and reads the number \( n \) of values. It allocates space for vectors \( v \) and \( w \). It asks for and reads the \( n \) values of vector \( v \) on one line. It asks for and reads the \( n \) values of vector \( w \) in the next. Finally it prints: "The distance = " followed by the distance calculated by \( \text{distance}(v,w,n) \).

c10.4 Write a subroutine \( \text{calculate}(a,b,n) \) which, given an \( n \times n \) matrix \( a \) of integers, extends this to a larger \( (n+1) \times (n+1) \) matrix \( b \) which has an additional column of row totals on the right and an additional row of column totals at the bottom. Write a program which asks for and reads the number \( n \), allocates space for matrices \( a \) and \( b \), initialize \( b \) to the zero matrix, asks for and reads the \( n \) rows of \( a \), one at a time, calls the subroutine \( \text{calculate} \) and prints the resulting matrix \( b \).
Equality of matrices

In fortran, we cannot check if two matrices \(a\) and \(b\) are equal to each other just by typing \(a==b\). The test must be performed element-by-element.

e10.3

The following script uses a function \texttt{matrix\_equal(A,B,n,m)} to verify if 2 \(n \times m\) matrices \(a\) and \(b\), entered by the user, are equal to each other.

```fortran
program diago
  logical::matrix\_equal
  integer, dimension(:,:), allocatable ::a,b
  Print*, 'number of rows?'
  read*, n !user enters number of rows
  Print*, 'number of columns?'
  read*, m !user enters numbers of columns
  allocate(a(n,m),b(n,m));
  Do i=1,n
    Print*, 'enter row',i,'of matrix a'
    read*,a(i,:) !user enters rows of a, one at a time
  enddo
  Do i=1,n
    Print*, 'enter row',i,'of matrix b'
    read*,b(i,:) !user enters rows of b, one at a time
  enddo
  Print*, 'matrix a='
  Do i=1,n
    print*,a(i,:)!prints rows of a, one at a time
  enddo
  Print*, 'matrix b='
  Do i=1,n
    print*,b(i,:)!prints rows of b, one at a time
  enddo
  if (matrix\_equal(a,b,n,m)) then !checks if a equals b
    print*,’a and b are the same’
  else
    print*,’a and b are different’
  endif
endprogram
```
Let $A$ and $B$ be two $n \times n$ matrices. The *Lie bracket* of $A$ and $B$, denoted $[A, B]$, is defined by $[A, B] = AB - BA$. We say that $A$ and $B$ commute if $[A, B]$ is a matrix of zeros ($[A, B] = 0_{n,n}$). Write a function $\text{Liebrack}(a, b, n)$ which, given an integer $n$ and two matrices $a$ and $b$ of dimension $n \times n$, returns "commute" if $A$ and $B$ commute and "do not commute" otherwise.

**PRACTICE SESSION**

We recall that, in linear algebra, the *transpose* of an $n \times m$ matrix $A$, denoted $A^T$, is an $m \times n$ matrix whose rows are the columns of $A$. Moreover, a $n \times n$ matrix $A$ is said to be *symmetric* if $A$ is equal to its transpose (in other words, if $A = A^T$).

Write a subroutine $\text{transp}(a, b, n, m)$ which, given two integers $n$ and $m$, a $n \times m$ matrix $a$ and a $m \times n$ matrix $b$, sets $b$ equal to the transpose of $a$. Write a function $\text{is_symmetric}(a, n)$ which, given an integer $n$ and a $n \times n$ matrix $a$, uses the subroutine $\text{transp}$ to return the logical value .TRUE. if $a$ is symmetric and the logical value .FALSE. otherwise. Finally, write a program which asks for and reads a number $n$, allocates space for a $n \times n$ matrix $a$, asks for and reads the $n$ rows of $a$, one at a time, and uses the function $\text{is_symmetric}$ to print 'a is symmetric' if $a$ is symmetric and 'a is not symmetric' otherwise.

Moreover, a $n \times n$ matrix $A$ is said to be *orthogonal* if the matrix product $A.A^T$ is equal to the identity matrix $I_n$ ($A^T$ being the transpose of $A$, see the previous practice session).

Write a function $\text{is_identity}(a, n)$ which, given an integer $n$ and a $n \times n$ matrix $a$, returns .TRUE. if $a$ is equal to identity matrix $I_n$ and .FALSE. otherwise. Write a function $\text{is_orthogonal}(a, n)$ which, given an integer $n$ and an $n \times n$ matrix $a$, uses the subroutines $\text{transp}$ from the previous practice session and the function $\text{is_identity}$ to return the logical value .TRUE. if $a$ is orthogonal and the logical value .FALSE. otherwise. Finally, write a program which asks for and reads a number $n$, allocates space for an $n \times n$ matrix $a$, asks for and reads the $n$ rows of $a$, one at a time, and uses the function $\text{is_orthogonal}$ to print 'a is orthogonal' if $a$ is orthogonal and 'a is not orthogonal' otherwise.