Information Theory and Computation
Exercise 2

Vincenzo Maria Schimmenti - 1204565

October 22, 2019

Theory

The exercise asked to implement a *derived type* handling a double precision complex matrix with an arbitrary number of rows and columns and containing two variables for storing the trace and the determinant; also it is required to implement the adjoint and trace functions and operators. If we denoted the matrix elements as A_{ij} (assuming A is an $N \times N$ matrix) the adjoint is the matrix $A^\dagger$ for which:

$$A^\dagger_{ij} = A^*_{ji} \tag{1}$$

where the symbol $*$ denotes the complex conjugation. The trace is the scalar quantity:

$$\mathrm{tr}(A) = \sum_{i=1}^{n} A_{ii} \tag{2}$$

One can verify that:

$$\mathrm{tr}(A) = \mathrm{tr}^*(A^\dagger) \tag{3}$$

Code Development

First, since the derived type is based on a double complex bidimensional array, we implemented the trace and adjoint functions for this type of variable and then extended to our type. In principle the adjoint operation is defined only for square matrices but can be generalized without any ambiguity to rectangular ones; one can not say the same for trace operation. A crucial part of the code was deciding how to handle exceptions which in this case consist basically of negative or zero rows and columns sizes: one can either print an error message and close the program (using, for example, the *STOP* directive) or return a *null* output (which can be a zero for the trace or a zero by zero matrix in case of matrix initialization); we opted for the second solution.

Results

We tested our implementation on a random generated matrix (which, along with the adjoint, we wrote inside two output files); we also validated our result on the basis of equation 3.

Self Evaluation

The code written can be further improved by implementing a LU decomposition for our custom matrix in order to make an efficient determinant computation. Our code, as it is, suffers from a problem which is the marix elements update: since we are storing the trace and the determinant as variable we are assuming (since no otherwise stated) that these two quantities will remain coherent with the matrix elements; in principle one can either change the matrix or this values and the subsequent computation will be wrong; to circumvent this problem one has to create a flag that signals whether a value has changed but this would imply a fully wrapped matrix (i.e. custom access and writing of matrix elements).