Exercise 2
Complex Matrix Derived Type

Michele Guadagnini - Mt. 1230663

October 20, 2020

Abstract

This exercise is about the usage of derived types in Fortran. A complex matrix type containing the matrix elements, dimensions, trace, determinant and adjoint matrix is defined. The derived type include also the functions and interface operators for the trace and adjoint matrix computation.

1 Code Development

1.1 Derived Type Module

This module is inside the file *ComplexMatrixType-CODE.f90*.

The derived type contains an integer vector of length 2 to store the *matrix size*, two double complex variables for *Trace* and *Determinant* and two double complex allocatable arrays to store the matrix and its adjoint one. Then the functions computing trace and adjoint are defined with their correspondent operators. The trace function also checks that the matrix is square. The adjoint matrix function uses the intrinsic Fortran functions **CONJG** and **TRANSPOSE**.

To randomly initialize the matrix, a subroutine is created making use of the intrinsic function **RANDOM_NUMBER**; *Real* and *Imaginary* parts are scaled and shifted to be in the range [-1,1].

Finally also the subroutine for writing the object on a file is implemented. To write complex numbers in readable format a particular string is passed to the format specifier in the **WRITE** function. This descriptor is "(*('('sf6.3xspf6.3x'i)':x))" and it is made the following specifiers:

- *s* and *sp* to control the sign printing;
- *x* to add a blank space;
- *f6.3* to plot real numbers with 6 characters and 3 decimal digits;
- *:* to terminate format scanning immediately if there is no more data to print (to avoid the opening of another bracket at the end of the line);
- *** at the beginning is needed to print the elements in rows and columns.

A problem faced when writing this module was that the arguments passed to a function as input must be declared *INTENT(IN)* inside it in order to instruct the compiler to treat it as input, as suggested by the compiler.

1.2 Test Program

This program is written in the file *ComplexMatrixTest-CODE.f90*.

In order to test the new module a program is created, which accepts the output file name and the matrix size as arguments in this order. In case no arguments are provided it uses default values. Then it allocates the memory to store the matrix elements, calls the initialization subroutine, computes and stores trace and adjoint matrix and print all on a file. Then it deallocates memory and stops. The provided *README* file describes the compilation and usage of the program.

One encountered issue was that sometimes the name of the output file was truncated when saved because the length of the character variable used to store it was too short. Changing from *CHARACTER(20)* to *CHARACTER(30)* solved the issue.

2 Results

The default output file is called *TestOutput.txt.* The program works for different matrix shapes, also for non-square matrices skipping the trace computation. Two other example output files are provided (*TestOutputBigger.txt* and *TestOutputNonSquare.txt*).

3 Self-evaluation

New things learned with this exercise are:

- creating a derived type object;
- using functions and subroutines, passing values to and retrieving results from;
- generating random numbers in any range;
- using format specifiers with *WRITE* function;
- passing and using user arguments when running a program.

Something that can be done next is to implement in the module a function that computes the determinant of the matrix.

The difficulties encountered in this exercise were mostly syntax errors, usually fast solved with the help of the compiler.