Derived Types in Fortran90

Abstract

The aim of this exercise is to calculate the adjoint and the trace of a given matrix, with complex entries. This is implemented in Fortran90, through the definition of a *double complex matrix derived TYPE*, that contains the matrix elements, the dimensions, the trace and the determinant. Also functions to perform the calculations are developed and the results are printed on text files. The focus is on the usage of the `INTERFACES` to built new operators to solve our tasks.

Theory

The *Hermitian adjoint matrix*, of a given (rectangular or square) matrix A, over the field $\mathbb{C}$ of complex numbers, is its complex-coniugate transpose matrix. The latter definition holds only for a finite dimensional Hilbert space H.

The adjoint matrix of A is denoted as $A^\dagger$, where

$$A^\dagger = A^{\intercal *} \tag{1}$$

The symbol $*$ denotes the complex coniugate.

The trace of a given square matrix A , $n \times n$, is defined as

$$Tr(A) = \sum_{i=1}^{n} a_{ii} \tag{2}$$

Note that the trace is *only* defined for a square matrix.

The following properties hold:

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

$$det(A^\dagger) = detA^* \tag{4}$$

Code development

Firstly a `MODULE` called `matrix` is written and inside it a derived data `TYPE`, called `cmatrix`, has the following content:

- A vector, named `RC`, that contains two integers, that are the dimensions of our matrix.
- A matrix, named `m`, whose size is not yet specified (`ALLOCATABLE`) , that will contains double precision complex numbers.
- Two double precision complex numbers, the trace and the determinant of the matrix. The determinant is not calculated in this exercise (maybe in future).

After the `TYPE`, there are the interfaces for the initialization of the type, the adjoint matrix and trace computation. The corresponding functions follow, below the keyword `CONTAINS`.
Let's see the main points of them.

1) The function `f_init` takes in input a two dimensional integer vector, named `m_size`, that represents the matrix's size, and initializes the type defined above to zero. The vector `m_size` is defined in the function and allows to check if the dimensions of the matrix inserted by the user are positive. If not so, a warning message is printed on screen.

To calculate the adjoint and the trace the whole `cmatrix` data type is passed to the functions `adj_cmatrix` and `trace_cmatrix`.

2) The function `trace_cmatrix` returns a double precision complex number, that is the trace of the given matrix. When the matrix is not square a warning message is printed, setting the trace of this matrix to zero. As stated in the *Theory* section, the trace is defined only for a square matrix.

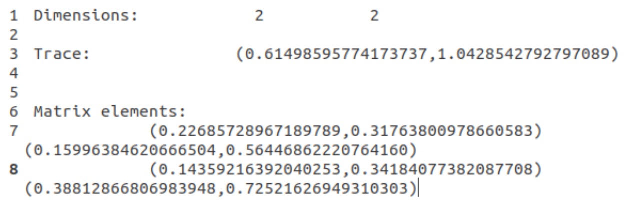
3) The function `adj_cmatrix` returns all the quantities defined in `cmatrix`, properly calculated. In particular it calculates the adjoint through the following FORTRAN instrinsic way:

$$TRANSPOSE(CONJG(matrix)) \tag{5}$$

Then a subroutine called `writetxt` writes on a file the aforementioned informations. In the end the main program `EX2` is implemented. The program first asks the user the number of rows and columns he wants for the matrix. Then it allocates the matrix with the decided dimensions and choose at random the values for the real and the imaginary part of the entries. Then the matrix is printed in the file `"Amat.txt"` and its adjoint is calculated and printed in `"Aadj.txt"`. The trace for both is avaible. The calculations are done through the operators defined in the interfaces, `.INIT.` `.ADJ.` and `.TRACE.`.

Results

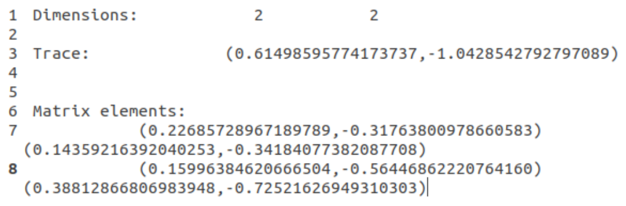
I runned the code several times, varing the dimensions of the matrix. Each time the real and the imaginary part of the entries are choosen at random as expected. The figure 1 shows an example of the output, contained in the file "`Amat.txt`".



```
Dimensions:           2           2

Trace:             (0.61498595774173737,1.0428542792797089)


Matrix elements:
              (0.22685728967189789,0.31763800978660583)
 (0.15996384620666504,0.56446862220764160)
              (0.14359216392040253,0.34184077382087708)
 (0.38812866806983948,0.72521626949310303)
```

Figure 1: Input complex matrix, randomly generated, with the trace.

One can easily check, in this dimensions ,that the trace satisfies the definition (2). In figure 2 we can see the output contained in the file "`Aadj.txt`".

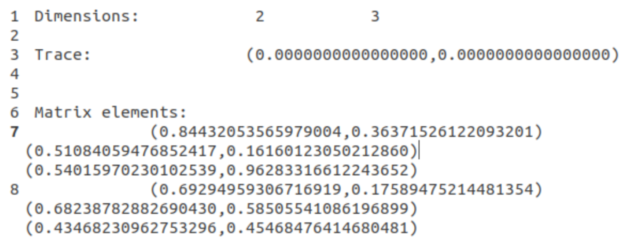


```
Dimensions:           2           2

Trace:            (0.61498595774173737,-1.0428542792797089)


Matrix elements:
             (0.22685728967189789,-0.31763800978660583)
 (0.14359216392040253,-0.34184077382087708)
             (0.15996384620666504,-0.56446862220764160)
 (0.38812866806983948,-0.72521626949310303)
```

Figure 2: The adjoint complex matrix, with the trace.

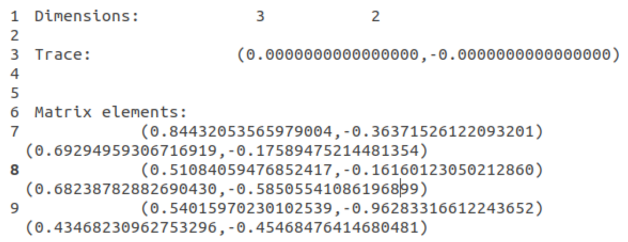
Again one can verify that the trace agrees with the definition (2) and that the matrix is the adjoint of the one in figure 1, according to the definition (1). Finally, comparing the traces in figure 1 and 2, one can see that the property (3) is verified.

Figures 3 and 4 show the outputs, contained respectively in the files "`Amat.txt`" and "`Aadj.txt`". This time the dimensions chosen by the user are not equal i.e. the matrix is not square. One can perform the aforementioned check, the matrix in figure 4 is the adjoint of the one in figure 3. Note that the trace in this case is set to zero, for both the matrices.



```
Dimensions:           2          3

Trace:              (0.0000000000000000,0.0000000000000000)


Matrix elements:
             (0.84432053565979004,0.36371526122093201)
(0.51084059476852417,0.16160123050212860)
(0.54015970230102539,0.96283316612243652)
             (0.69294959306716919,0.17589475214481354)
(0.68238782882690430,0.58505541086196899)
(0.43468230962753296,0.45468476414680481)
```

Figure 3: Input complex matrix, randomly generated, with the trace.



```
Dimensions:           3          2

Trace:              (0.0000000000000000,-0.0000000000000000)


Matrix elements:
             (0.84432053565979004,-0.36371526122093201)
(0.69294959306716919,-0.17589475214481354)
             (0.51084059476852417,-0.16160123050212860)
(0.68238782882690430,-0.58505541086196899)
             (0.54015970230102539,-0.96283316612243652)
(0.43468230962753296,-0.45468476414680481)
```

Figure 4: The adjoint complex matrix, with the trace.

Self-Evaluation

The exercise is instructive since it allows to understand the powerful tool `INTERFACE OPERATOR`, that allows to 'summarize' in one symbol (the operator) one or even more functions. Further modifications of the code could be implementing the calculus of the determinant and adding some more functions that are used by the operators `.ADJ.` `.TRACE.` and `.INIT.`