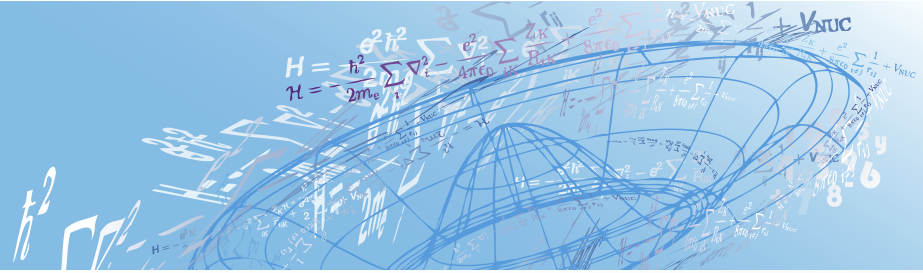


QcMatrix: An Abstract Matrix Library

Version 0.1.0

Bin Gao

Centre for Theoretical and Computational Chemistry (CTCC)
Department of Chemistry
University of Tromsø—The Arctic University of Norway
N–9037 Tromsø, Norway

March 14, 2015

Outline

Outline

QcMatrix

1. An “abstract” matrix library written in C language (with C++ and Fortran interface).
2. Provides a special square block complex matrix (all the blocks are square matrix with the same dimension) and corresponding functions to C, C++ and Fortran codes.
3. Being “abstract”, QcMatrix should be in general built on top of external matrix library.

The Square Block Complex Matrix in QcMatrix

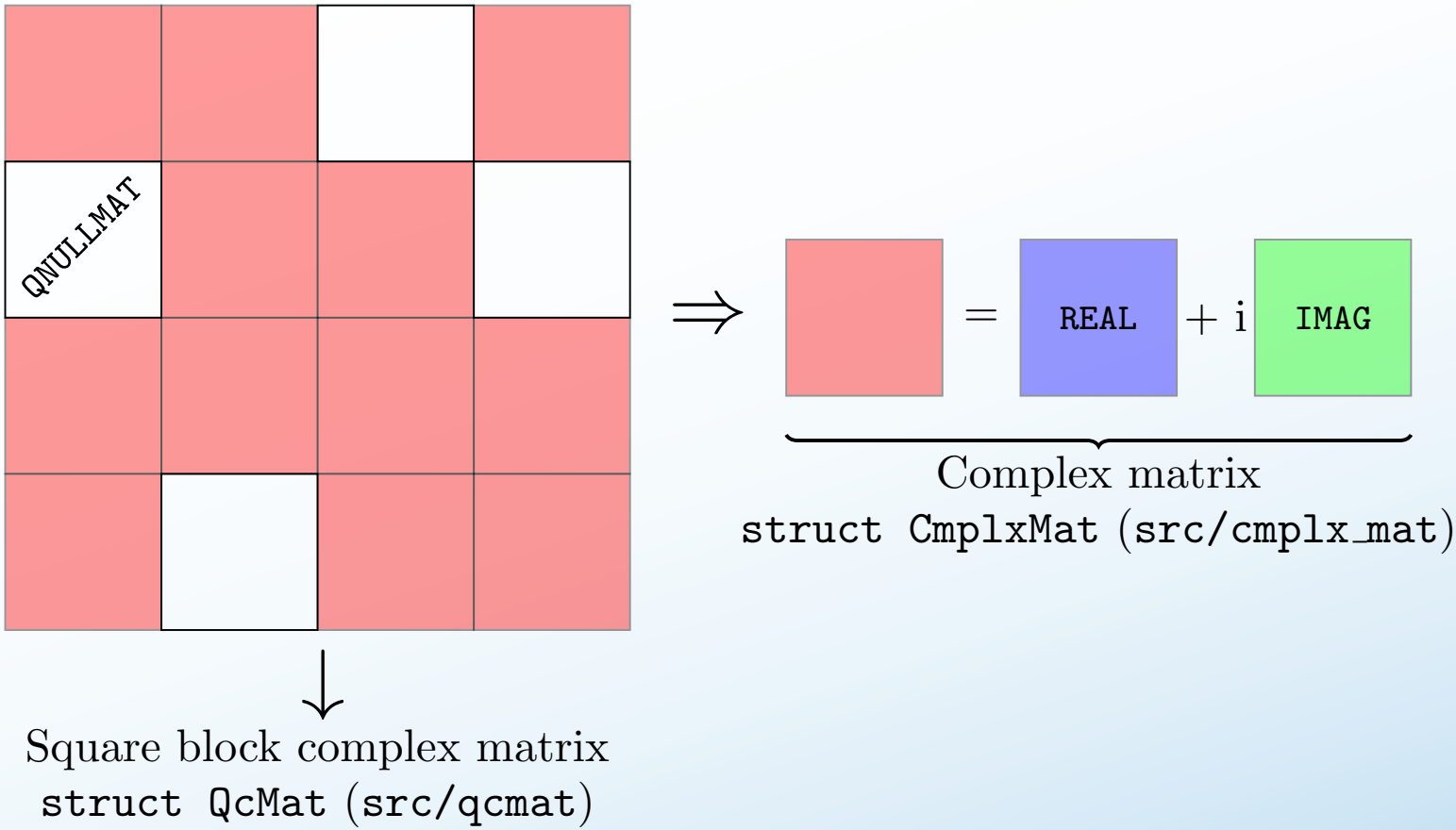


Figure : Illustration of the square block complex matrix implemented in QcMatrix — 4×4 blocks, with red blocks as the non-zero complex matrices. Each block is square matrix with the same dimension.

QcMatrix — An Adapter between Different Languages

1. The external matrix library (implemented in the host program) can be written either in C, C++ or Fortran language.

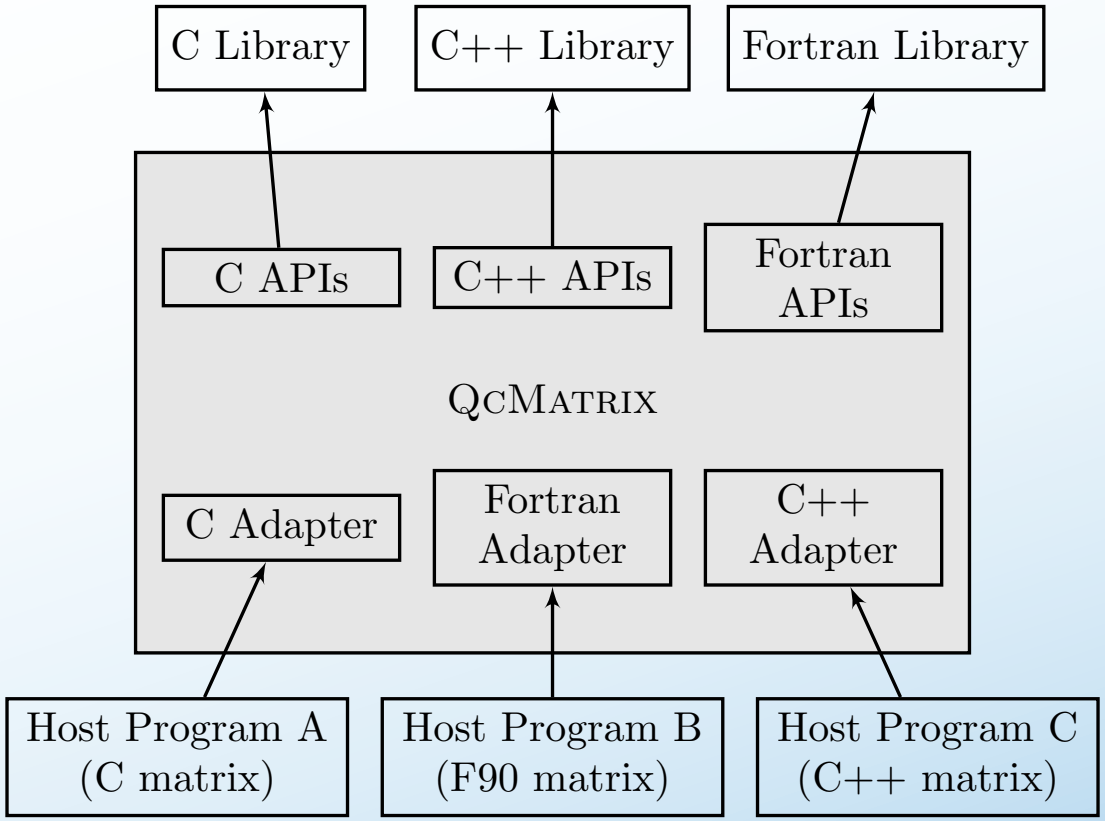


Figure : QcMatrix as an adapter between different languages.

QcMatrix — An Adapter between Different Matrices

1. The external matrix library can implement either real matrix, complex matrix, square block real matrix or square block complex matrix.

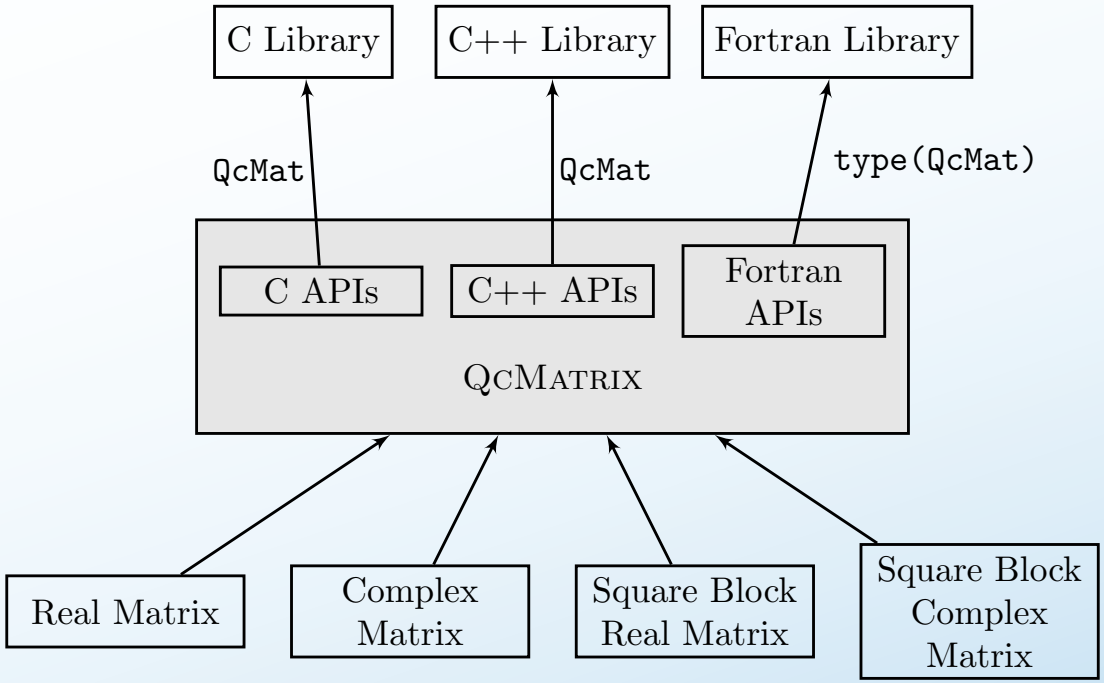


Figure : QcMatrix as an adapter between different matrices.

Outline

An Matrix Application Library

1. To greatly use an matrix application library LIBAPP, we normally want to implement it into different host programs, like

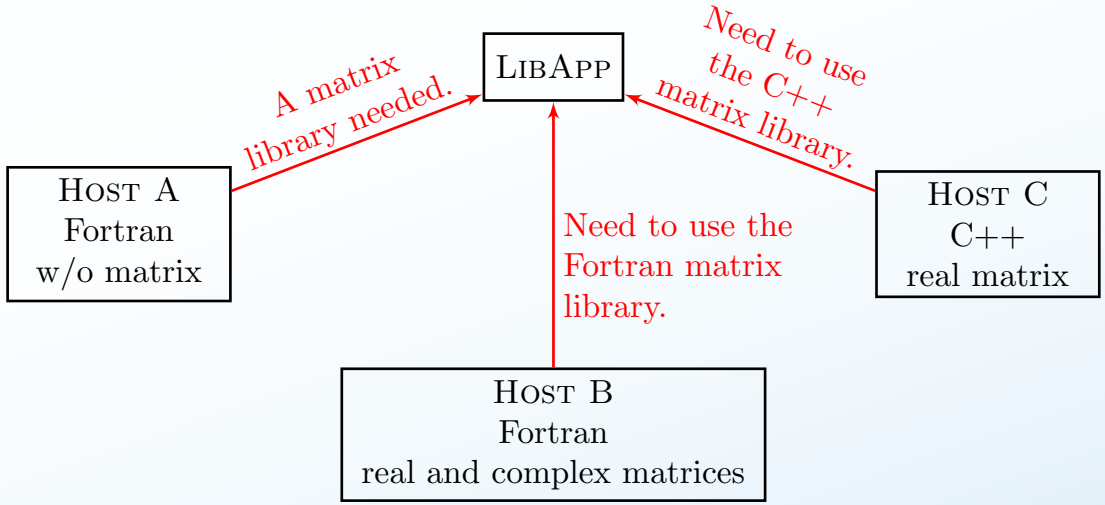
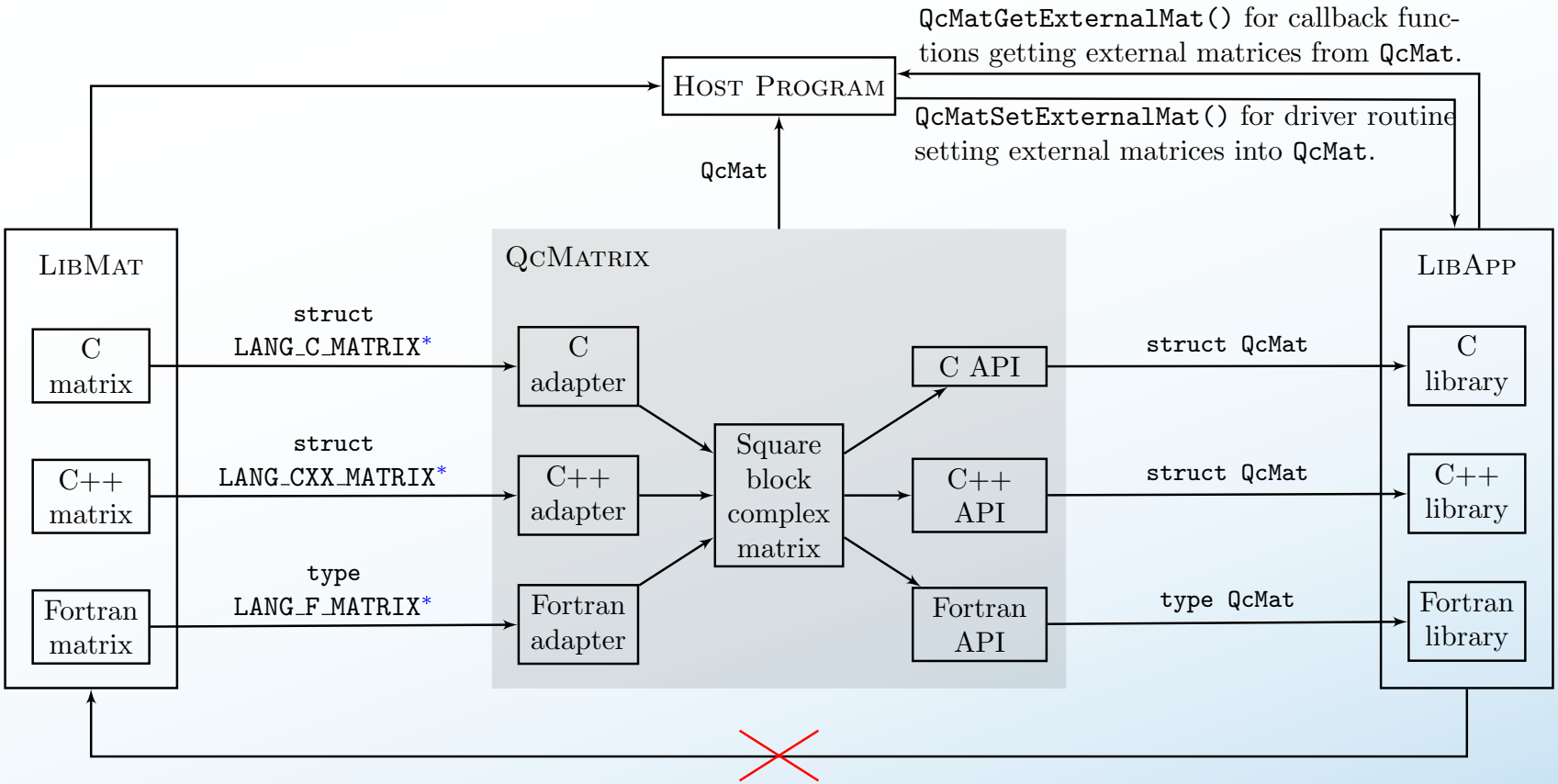


Figure : Implementation of an matrix application library (LIBAPP) in different host programs.

2. But we **only want one version** LIBAPP, which could be used in **different host programs**.
3. We may also want LIBAPP to **use the matrix library in the host programs** which could be well optimized.
4. These two requirements however may not be easy to satisfy for the library LIBAPP itself.

Make It Easier by using QcMatrix



*Accept square block complex matrix, square block real matrix, complex matrix and real matrix.

1. No communication between LibApp and different matrix libraries.
2. QcMatrix APIs `QcMatSetExternalMat` and `QcMatGetExternalMat` further facilicitate the driver routine and callback functions between the host program and LibApp, which will be discussed in the part "How to use QcMatrix".

Outline

Where to get QcMatrix?

1. Public at `https://gitlab.com/bingao/qcmatrix`.
2. Read-only access version
   `git clone git@gitlab.com:bingao/qcmatrix.git`.
3. Interface of LSDalton (`http://daltonprogram.org/`) implemented (`qcmatrix` branch).
4. Maybe more interfaces, for instance, PSI4 (`http://www.psicode.org/`), ErgoSCF (`http://ergoscf.org/`), ...

Outline

Outline

Before Compilation

Make sure the following programs are installed on your computer:

1. Git[1],
2. CMake ($\geq$ 2.8),
3. HDF 5 ($\geq$ 1.8) if matrix I/O is enabled (if HDF 5 is not available, ordinary binary format file will be used which is not portable),
4. C, C++ (if C++ adapter and APIs required) and/or Fortran (if Fortran adapter and APIs needed) compilers,
5. BLAS and LAPACK libraries for test suite and/or QcMatrix internal real matrix library.

[1] If you use QcMatrix in a host program, you may not need git. Please check the manual of the host program.

Compile QcMatrix as A Standalone Library

Two options to compile QcMatrix as a standalone library:

1. Use the simple C matrix library in `src/real_mat`:
   1. Assume that you want to compile QcMatrix in directory `build`, you could invoke the following commands:

      ```
      mkdir build
      cd build
      ccmake ..
      make
      ```

   2. During the step `ccmake`, you should set `QCMATRIX_BUILD_ADAPTER` as `OFF`, and freely choose other parameters[2]. But it is strongly recommended that you build the test suite excutables by setting `QCMATRIX_TEST_EXECUTABLE=ON`.
2. Use either the simple C++ or Fortran matrix libraries in `tests/cxx/adapter` and `tests/f90/adapter`:
   1. Compile the C++ or Fortran matrix libraries first by following the `README` file therein.
   2. During the step `ccmake`, set `QCMATRIX_BUILD_ADAPTER=ON` and other parameters for the adapter.

[2] Please refer to Table 2.1 in the manual for the meanings of the CMake parameters.

Compile QcMatrix as A Standalone Library (Continue)

1. If no error happened, a library `lib${LIB_QCMATRIX_NAME}.a` will be built (and probably some test suite executables named as `${LIB_QCMATRIX_NAME}_xx_test`).
2. Here, `LIB_QCMATRIX_NAME` is set during the CMake procedure (default is `qcmatrix`).

Compile QcMatrix with Host Program

1. It should follow the same procedure to compile QcMatrix as the second option in last slide by using the matrix library of the host program, which means that the host program needs to provide the matrix type and functions to QcMatrix. We will be discuss this later in Section "How to Prepare the External Library for QcMatrix".
2. To further facilitate the compilation, CMake `ExternalProject` could be used to build QcMatrix automatically in external trees. See for instance, `http://www.kitware.com/media/html/BuildingExternalProjectsWithCMake2.8.html`.
3. Currently, QcMatrix is implemented in the `qcmatrix` branch of Dalton program using the CMake `ExternalProject`. Interested users could check these directories in Dalton: `DALTON/qcmatrix` and `LSDALTON/qcmatrix`.

Outline

Test Suite of QcMatrix

1. All the tests are in the directory `tests`, and will be built as executables if `QCMATRIX_TEST_EXECUTABLE=ON`. More explicitly, `${LIB_QCMATRIX_NAME}_c_test` will always be built, `${LIB_QCMATRIX_NAME}_cxx_test` and `${LIB_QCMATRIX_NAME}_f_test` will be built only if C++ and Fortran APIs are built.
2. If `QCMATRIX_TEST_EXECUTABLE=OFF`, these tests will also be compiled and into the library `lib${LIB_QCMATRIX_NAME}.a` so that they could be invoked from the host program as, `ierr = test_c_QcMatrix()` (C and C++) or `call test_f_QcMatrix(io_log)` (Fortran).
3. In any case, users are strongly recommended to perform the test suite of QcMatrix to make it has been correctly compiled.

Outline

QcMatrix APIs

1. Use QcMatrix library:

Language	Including QcMatrix	Declaration of a matrix
C	`#include "qcmatrix.h"`	`QcMat A;`
C++		
Fortran	`use qcmatrix_f`	`type(QcMat) A`

2. All QcMatrix APIs have been described in Chapter 3 "QcMatrix API Reference" of the manual. Importantly:
   1. All functions should be used as `ierr = QcMat...(...)`. `QErrorCode ierr` contains the error information, and should be `QSUCCESS` if no error happened[3].
   2. All matrices must first be created by `QcMatCreate`, and finally destroyed by `QcMatDestroy`. Futher requirements can be needed for some functions, please see Table 3.6 in the manual.
   3. All functions can be used in the same way in Fortran code, but the types of some arguments are different from those in C/C++ code, please check Table 3.2 of the manual for the types in Fortran.

[3] If `QCMATRIX_AUTO_ERROR_EXIT==ON`, QcMatrix will automatically exit on error, and users do not need to check `QErrorCode ierr`.

Examples

1. A simple Fortran code `examples/simple_qcmat.F90` has been prepared to show the use of QcMatrix. Interested users could follow it, or read the test codes in `tests` to understand the use of QcMatrix.

Set External Matrices into QcMat

`QcMatSetExternalMat_f()` can set a matrix of host program `type(LANG_F_MATRIX)` into a matrix of `type(QcMat)`, making the driver routine for application libraries easier[4]:

```
subroutine host_program_driver(A_ext, ...)
    use qcmatrix_f
    implicit none
    type(LANG_F_MATRIX), pointer, intent(inout) :: A_ext  !real matrix
    type(QcMat) A
    ierr = QcMatCreate_f(A)
    ! Sets the real part of block(1,1) of the matrix A
    ierr = QcMatSetExternalMat_f(A=A,                    &
                                 idx_block_row=1,        &
                                 idx_block_col=1,        &
                                 data_type=QREALMAT,     &
                                 A_ext=A_ext)
    ! Calls application library using the matrix A
    ... ...
    ierr = QcMatDestroy_f(A)
    return
end subroutine host_program_driver
```

[4]The context of `A_ext` will not be destroyed by `QcMatDestroy_f`. It should be destroyed by the host program after its use.

Get External Matrices from `QcMat`

`QcMatGetExternalMat_f()` can get a matrix of host program `type(LANG_F_MATRIX)` from a matrix of `type(QcMat)`, making the callback functions for application libraries easier[5]:

```
subroutine host_program_callback(A, ...)
    use qcmatrix_f
    implicit none
    type(QcMat), intent(inout) :: A
    type(LANG_F_MATRIX), pointer :: A_ext  !real matrix
    ! Gets the real part of block(1,1) of the matrix A
    ierr = QcMatGetExternalMat_f(A=A,                  &
                                 idx_block_row=1,      &
                                 idx_block_col=1,      &
                                 data_type=QREALMAT,   &
                                 A_ext=A_ext)
    ! Calls other subroutines in the host program using the matrix A_ext
    ... ...
    return
end subroutine host_program_callback
```

[5]The context of `A_ext` could be destroyed by the host program, but it might be better that it is destroyed by `QcMatDestroy_f`.

More about `QcMatGetExternalMat()`

1. It should be noted that users should **not** manipulate the external matrix `A_ext` (got from `QcMatGetExternalMat()`) if it is `QNULLMAT` (or in other words, if it is not assembled).
2. If you need to manipulate a `QNULLMAT` external matrix `A_ext` from `QcMatGetExternalMat()`, please write to the authors.

Outline

External Matrix Library from the Host Program

1. To use QcMatrix, the external library should implement the matrix type and functions/subroutines required by QcMatrix. First, it should prepare the required header files/modules and implemented matrix type[6]:

Language	Header file/Module	Implemented matrix
C	`LANG_C_HEADER`	`struct LANG_C_MATRIX`
C++		
Fortran	`LANG_F_MODULE`	`type LANG_F_MATRIX`

2. The external library should also implement functions/subroutines as described in Table 4.1 of the manual. Alternatively, users can check the files in `tests/cxx/adapter` and `tests/f90/adapter` for an example of external C++ and Fortran matrix libraries.

[6] The names of header files/modules and the matrix type can be specified during CMake. Please see Table 2.1 of the manual for the CMake parameters.

Outline

More about QcMatrix

1. Only one type of adapter can be compiled (C, C++ or Fortran) at once.
2. Strassen method for the square block matrix-matrix multiplication and 3M Method for the complex matrix-matrix multiplication can be used, but users should be aware of their numerical instabilities in some situations, and we as the authors can not always guarantee the use of these two methods.
3. If you have used QcMatrix and found it is useful, please consider to cite QcMatrix as

```
@misc{QcMatrix,
  author = {Bin Gao},
  title = {{QcMatrix Version 0.1.0}},
  year = {2015},
  note = {https://gitlab.com/bingao/qcmatrix}
}
```

Acknowledgments

1. DALTON Forum (`http://daltonprogram.org/forum/`, a good place to ask help for implementing QCMATRIX interface into DALTON and LSDALTON).
2. Stallo (`https://www.notur.no/hardware/stallo`, computation resources).
3. CTCC at UiT (`http://www.ctcc.no/`, an excellent working place).

Thank You for Your Attention!