ulm university universität
uulm

Institut of Numerical Mathematics
Dr. Andreas F. Borchert and Dr. Michael C. Lehn
Constantin Greif

18 November 2019
Quiz 3

High Performance Computing I (WS 2019/2020)

Deadline: 25 November 2019, 2pm

We want to propose our cache optimized GEMM implementation to the real word (i. e. to people outside the math department). However, we learn that in the real world many people use Fortran code. We reject the idea to rewrite our code in Fortran and instead provide a Fortran interface (have a look at the skeleton on the last page).

Look at interface *ulm_dgemm*: In the Fortran world matrices are always stored col-major. Because of that, the interface only requires the increment for columns (which is called *leading dimension*).

The interfaces and functionality should be compatible with the reference implementation from netlib.org. The functionality for dgemm is there described as

DGEMM performs one of the matrix−matrix operations

$$C := alpha*op(A)*op(B) + beta*C,$$

where op(X) is one of

$$op(X) = X \textbf{ or } op(X) = X**T,$$

alpha **and** *beta are scalars*, **and** *A*, *B* **and** *C are matrices*, *with op(A)*
an m by k matrix, *op(B) a k by n matrix* **and** *C an m by n matrix*.

We learn from the real world that a character is used in the interface to specify whether a matrix is supposed to be transposed or not, i.e. *'n'* or *'N'* means *not-transposed* and *'t'* or *'T'* means *transposed*.

We further learn that function parameters in Fortran are passed by reference. For our C interface that means that all parameters are passed as pointers.

Implement the Fortran interface for the GEMM implementation developed in session 07. Just include the functions needed for your GEMM implementation (functions for packing, GEMM micro kernel, GEMM macro kernel, GEMM frame algorithm, any utilities used by these fucntions) and the Fortran interface.

Your submitted file "quiz03.cpp" should compile on Theon without warnings to an object file with:

```
theon$ g++ -Wall -std=c++11 -m64 -c quiz03.cpp
```

Please submit your program “quiz03.cpp” as follows:

```
theon$ submit hpc quiz03 quiz03.cpp
```

You can extend the following skeleton with your implementation:

Listing 1: Skeleton for “quiz03.cpp”

```
extern "C" {

void
ulm_dgemm_(const char *transA, const char *transB,
           const int *m, const int *n, const int *k, const double *alpha,
           const double *A, const int *ldA, const double *B, const int *ldB,
           const double *beta, double *C, const int *ldC)
{
    // Call your C++ implementation.
    // Do *not* allocate any memory.
}

} // extern "C"
```