Presentation of Assignment 1

Quantum Information and Computing, A.Y. 2022/2023

Due date: 01/11/2022

Paolo Zinesi
paolo.zinesi@studenti.unipd.it

Setup

Run “EX1a_Zinesi_CODE.f90” in local

```
paolozinesi@MBP-di-Paolo Assignment1 % cd EX1a
paolozinesi@MBP-di-Paolo EX1a % gfortran EX1a_Zinesi_CODE.f
paolozinesi@MBP-di-Paolo EX1a % ./a.out
 Hello, world!
```

Run the same file in CloudVeneto

```
paolozinesi@MBP-di-Paolo EX1a % ssh -J pzinesi@gate.cloudveneto.it \
> -i [redacted]i.pem ubuntu@10.67.22.158
Warning: Identity file [redacted].pem not accessible: No su
ch file or directory.
pzinesi@gate.cloudveneto.it's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-128-generic x86_64)
```

```
ubuntu@qic-zinesi:~$ cd QIC_project/Assignment1/EX1a/
ubuntu@qic-zinesi:~/QIC_project/Assignment1/EX1a$ gfortran EX1a_Zinesi_CODE.f
ubuntu@qic-zinesi:~/QIC_project/Assignment1/EX1a$ ./a.out
 Hello, world!
```

Number Precision

When compiling "EX1b_Zinesi_CODE.f90" the following error arises

```
Error: Arithmetic overflow converting INTEGER(4) to INTEGER(2) at (1). This chec
k can be disabled with the option '-fno-range-check'
```

By forcing the compilation with "**-fno-range-check**" the result is

```
paolozinesi@MBP-di-Paolo EX1b % ./a.out
 Result of the sum with INTEGER*2: -31615
 Result of the sum with INTEGER*4:      2000001
 Result of the sum with REAL*4:    3.14159259E+32
 Result of the sum with REAL*8:    3.14159265360393 54E+032
```

An overflow is clearly occurring since **INTEGER*2** variables cannot store positive numbers greater than *32767*.

If "**-Wconversion**" flag is set, a warning is issued when assigning to a **REAL*4** variable a number with greater precision.

Performance Testing - Theory

Given two matrices A, B, the product $C = AB$ can be computed with two different methods:

- Naive method: $$C_{ik} = \sum_j A_{ij} B_{jk}$$
- Optimized method: $$C_{ik} = \sum_j \left(A^T\right)_{ji} B_{jk}$$

These two methods are mathematically equivalent, but since FORTRAN stores matrices by columns the first method is significantly slower than the second one for large matrices.

In fact, the transposition of a matrix scales only quadratically with the size and this additional computation is rewarded by an overall execution speedup.

Performance Testing - Strategy

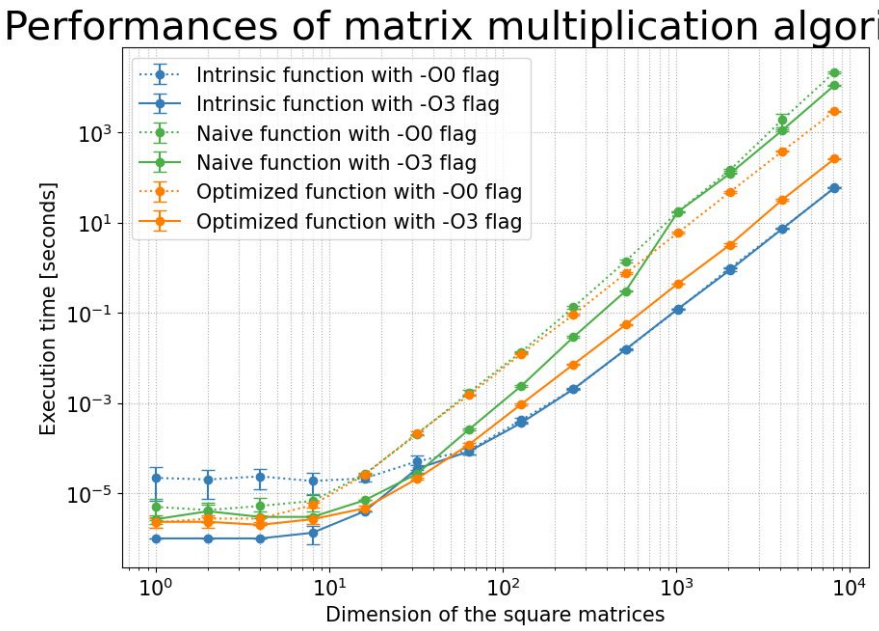
Two subroutines, "**my_MatMul_naive**" and "**my_MatMul_opt**", are developed to perform explicitly matrix-matrix multiplication. They expect three matrices as inputs, two input matrices and one output matrix in which results are stored. Compatibility of matrices' dimensions is checked beforehand.

The execution times of these two subroutines are then compared with FORTRAN "**MATMUL**" intrinsic function.

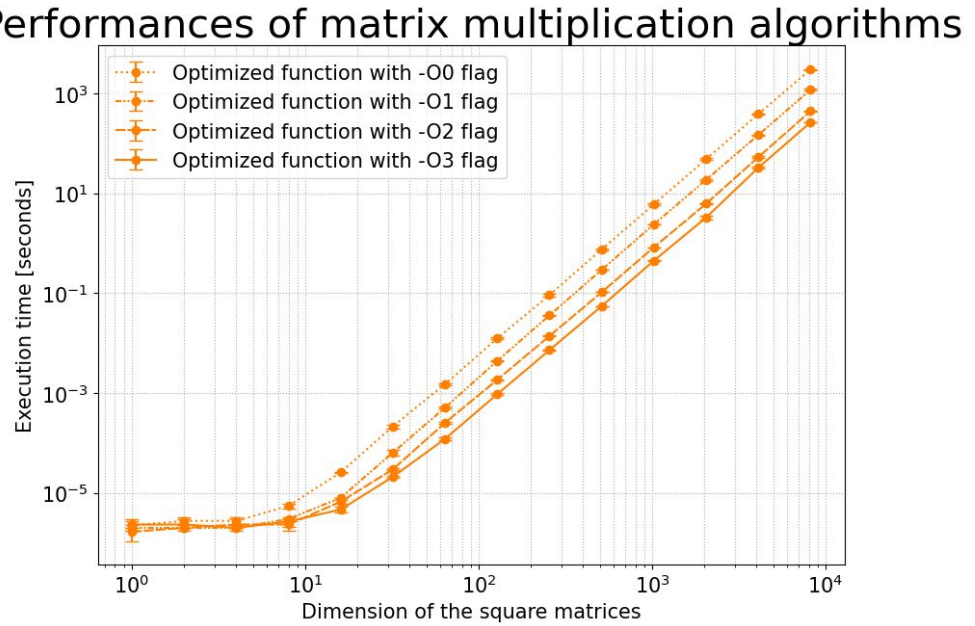
Tests are performed on a CloudVeneto VM with:

- 4 VCPUs
- 25 GB Disk
- 8 GB RAM

Performance Testing - Results



Optimization flags have an impact only on the optimized algorithm (orange curves).



- -O0 = no optimization (default)
- -Ox = optimization of execution time at level x