Quantum Information and Computing

Week 1 assignment

Ardino Rocco
Mat. 1231629
rocco.ardino@studenti.unipd.it

Monday 12th October, 2020

Exercise 1

The setup of the workspace was accomplished without any trouble. After, creating a working directory, inside this the following code for a test job is written:

```fortran
! 01_ex1.f90

program setup

    implicit none

    print*, "Hello World!"

end program
```

Listing 1: Test job submitted.

After accessing to spiro.fisica.unipd.it through `ssh` protocol, the test job is compiled through gfortran compiler and correctly submitted and executed.

Exercise 2

The first sum of the numbers 1 and 2000000 is performed using integers of 2 bytes and then of 4 bytes. In the first case, there is evidence of overflow since the result is not correct (-31615). In the second case, the operation returns the correct result.
The second sum of the numbers $\pi \cdot 10^{32}$ and $\sqrt{2} \cdot 10^{21}$ is performed using floating points of 4 bytes and then of 8 bytes. The first operation returns the correct result up to the seventh significant digit in comparison to the second operation, which is considered much more accurate for the greater number of bytes involved.

Exercise 3

Several implementations of matrix matrix multiplication are possible, depending on the order of the loops needed for the purpose. The two implementations for this report are given in Listings 2 and 3.

```fortran
! matrix matrix multiplication by column
function matmul_col(mat1, mat2) result(mat3)
    integer*2 :: ii, jj, kk
    integer*2 :: N, M, L
    real*4, dimension(:,:) :: mat1, mat2
    real*4, dimension(size(mat1, 1),size(mat2, 2)) :: mat3

    N = size(mat1, 1)
    M = size(mat1, 2)
    L = size(mat2, 2)

    do jj=1,M
        do kk=1,L
```

```fortran
            do ii=1,N
                mat3(ii,jj) = mat3(ii,jj) + mat1(ii,kk)*mat2(kk,jj)
            end do
        end do
    end do
end function
```

Listing 2: First implementation of matrix matrix multiplication.

```fortran
! matrix matrix multiplication by row
function matmul_row(mat1, mat2) result(mat3)
    integer*2 :: ii, jj, kk
    integer*2 :: N, M, L
    real*4, dimension(:,:) :: mat1, mat2
    real*4, dimension(size(mat1, 1),size(mat2, 2)) :: mat3

    N = size(mat1, 1)
    M = size(mat1, 2)
    L = size(mat2, 2)

    do ii=1,N
        do kk=1,L
            do jj=1,M
                mat3(ii,jj) = mat3(ii,jj) + mat1(ii,kk)*mat2(kk,jj)
            end do
        end do
    end do
end function
```

Listing 3: Second implementation of matrix matrix multiplication.

Their performances are monitored for $n \times n$ matrices through the function `CPU_TIME` and for different values of n. The results are showed in Figure 1. It is possible to see how the first implementation is slightly faster than the second one and how the two user implementations are much slower than the intrinsic one. The first fact is due to the way fortran stores variables in memory when dealing with arrays. In fact, this is done by columns, so the first implementation is faster since the inner loop is over consecutive elements in memory. The second fact is due to a different implementation highly optimized using state of the art algorithms.

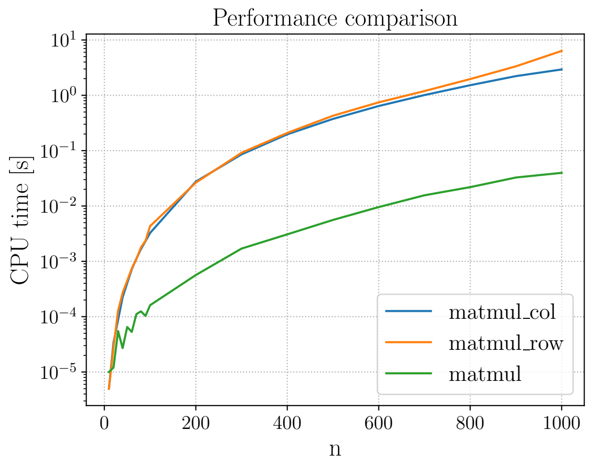

Figure 1: Performances of the two implementations and of the intrinsic function depending on the matrix size n.

Another test on the execution speed is done by applying different optimization flags, listed in Table 1. The same analysis done before is performed for every implementation and for every optimization flag. Again, the results are plotted in order to compare how the different flags affect the execution time of the matrix matrix multiplication. The plots are showed in Figure 2.

Flag	Exec. time	Code size	Mem. usage	Compile time
`-O0`	+	+	−	−
`-O1`	−	−	+	+
`-O2`	−−		+	++
`-O3`	− − −		+	+ + +
`-Ofast`	− − −		+	+ + +

Table 1: Optimization flags effects.

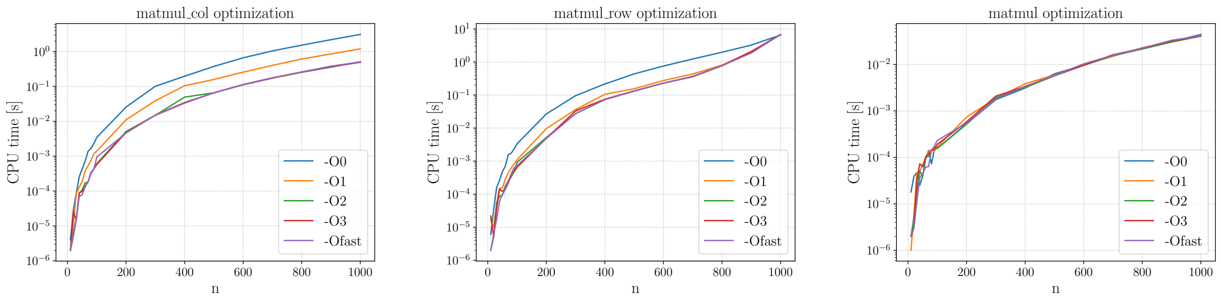

Figure 2: Left: execution time for the first implementation for several optimization flags. Center: execution time for the second implementation for several optimization flags. Right: execution time for the intrinsic function for several optimization flags.