Quantum Information and Computing

Prof. Simone Montangero

a.a. 2021/2022

Tommaso Faorlin (2021857)

Assignment 1

1 Exercise 1

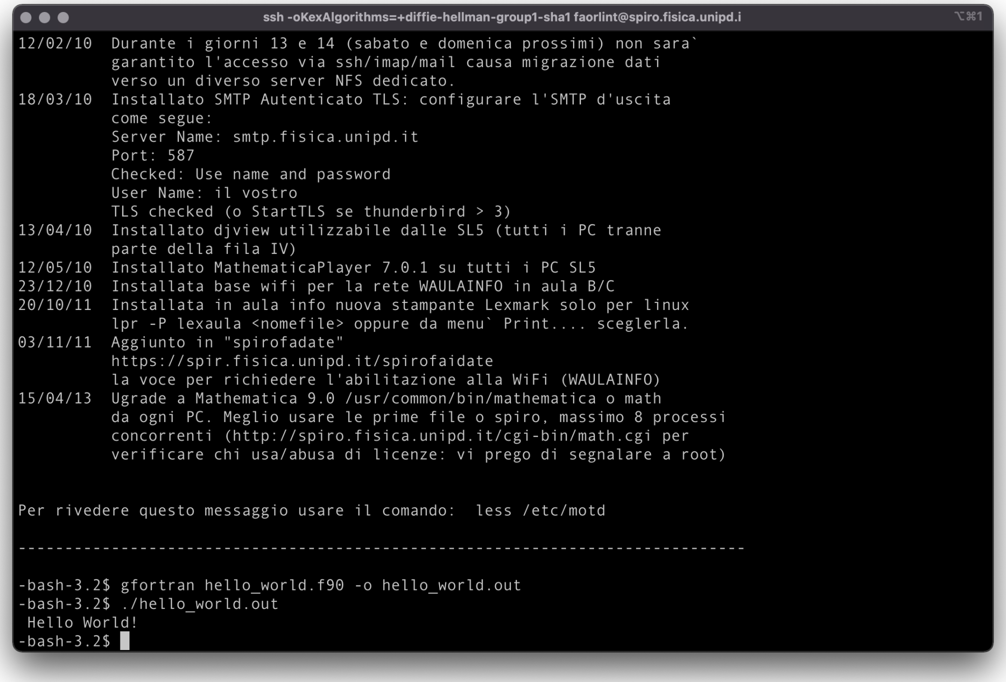
(a) Created a working directory `/Users/tommasofaorlin/Documents/QuantumInfo`.

(b) The following program is written.

```
!Week 1 - Exercise 1, Faorlin Tommaso
program hello_world
    implicit none
    print *, "Hello␣World!"
end program hello_world
```

(c) The program is compiled and run: we obtain *Hello World!* on the standard output.

(d) We compile and run the Hello World program on the `spiro.fisica.unipd.it` gate, obtaining the same result as in (c).


ssh -oKexAlgorithms=+diffie-hellman-group1-sha1 faorlint@spiro.fisica.unipd.i
⌥⌘1

```
12/02/10  Durante i giorni 13 e 14 (sabato e domenica prossimi) non sara`
          garantito l'accesso via ssh/imap/mail causa migrazione dati
          verso un diverso server NFS dedicato.
18/03/10  Installato SMTP Autenticato TLS: configurare l'SMTP d'uscita
          come segue:
          Server Name: smtp.fisica.unipd.it
          Port: 587
          Checked: Use name and password
          User Name: il vostro
          TLS checked (o StartTLS se thunderbird > 3)
13/04/10  Installato djview utilizzabile dalle SL5 (tutti i PC tranne
          parte della fila IV)
12/05/10  Installato MathematicaPlayer 7.0.1 su tutti i PC SL5
23/12/10  Installata base wifi per la rete WAULAINFO in aula B/C
20/10/11  Installata in aula info nuova stampante Lexmark solo per linux
          lpr -P lexaula <nomefile> oppure da menu` Print.... sceglerla.
03/11/11  Aggiunto in "spirofadate"
          https://spir.fisica.unipd.it/spirofaidate
          la voce per richiedere l'abilitazione alla WiFi (WAULAINFO)
15/04/13  Ugrade a Mathematica 9.0 /usr/common/bin/mathematica o math
          da ogni PC. Meglio usare le prime file o spiro, massimo 8 processi
          concorrenti (http://spiro.fisica.unipd.it/cgi-bin/math.cgi per
          verificare chi usa/abusa di licenze: vi prego di segnalare a root)


Per rivedere questo messaggio usare il comando:  less /etc/motd

------------------------------------------------------------------------------

-bash-3.2$ gfortran hello_world.f90 -o hello_world.out
-bash-3.2$ ./hello_world.out
 Hello World!
-bash-3.2$ █
```

2 Exercise 2

(a) The sum of 2000000 and 1, after they are stored as integers in two bytes `INTEGER*2` returns a wrong result: -31615. In fact, in two bytes we can store only signed integers in the range $[-32768, 32767]$ and 2e6 is outside of it. Fortran converts the first number in binary digits, i.e. 111101000010010000000, save the first 16 digits (2 bytes) from the LSB in memory 1000010010000000, sum 0000000000000001, and the corresponding decimal after 2's complement is exactly -31615.

What happened in the previous exercise does not happen in the case we use 4 bytes to store the two numbers. With INTEGER*4 we can store signed integers in the range $[-2147483648, 2147483647]$ and so with no problems we obtain in the end a correct result: 2000001.

(b) The sum of $\pi \cdot 10^{32}$ and $\sqrt{2} \cdot 10^{21}$ is performed with single and double precision. The two results are both correct up to different significant digits. With single precision I obtain $3.14158278 \cdot 10^{32}$, while with double precision $3.1415927578462317 \cdot 10^{32}$.

3 Exercise 3

(a)(b) The matrix multiplication has been implemented in two different ways (matmul_byrow and matmul_bycol), by changing the order of the loops needed for the computation. The state-of-art MATMUL subroutine has been used as well.

(c)(d) We perform a benchmark by testing the three different methods on squared matrices with sizes $[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]$. A first result (where the compilation includes -Oflag) is the following:

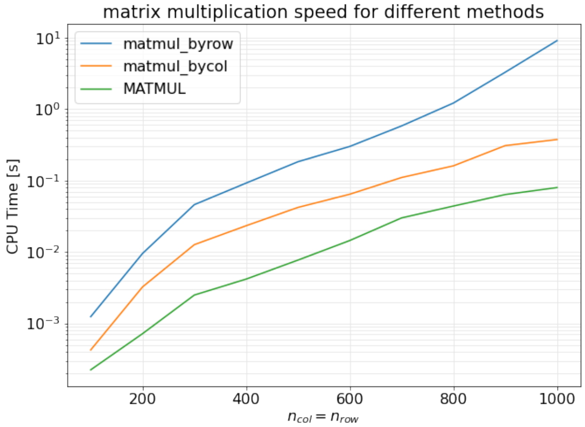


Figure 1: Single runs for different input sizes of matrix multiplication with three distinct methods and -Ofast flag in compilation. matmul_bycol is slightly faster than matmul_byrow, and this is related to the fact that Fortran stores matrices in memory by columns, and having the innermost for loop running on consecutive elements in memory is recommended.

Also, we show in the following graphs the results with some optimization flags (taken from https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html) in compilation for the three different methods .

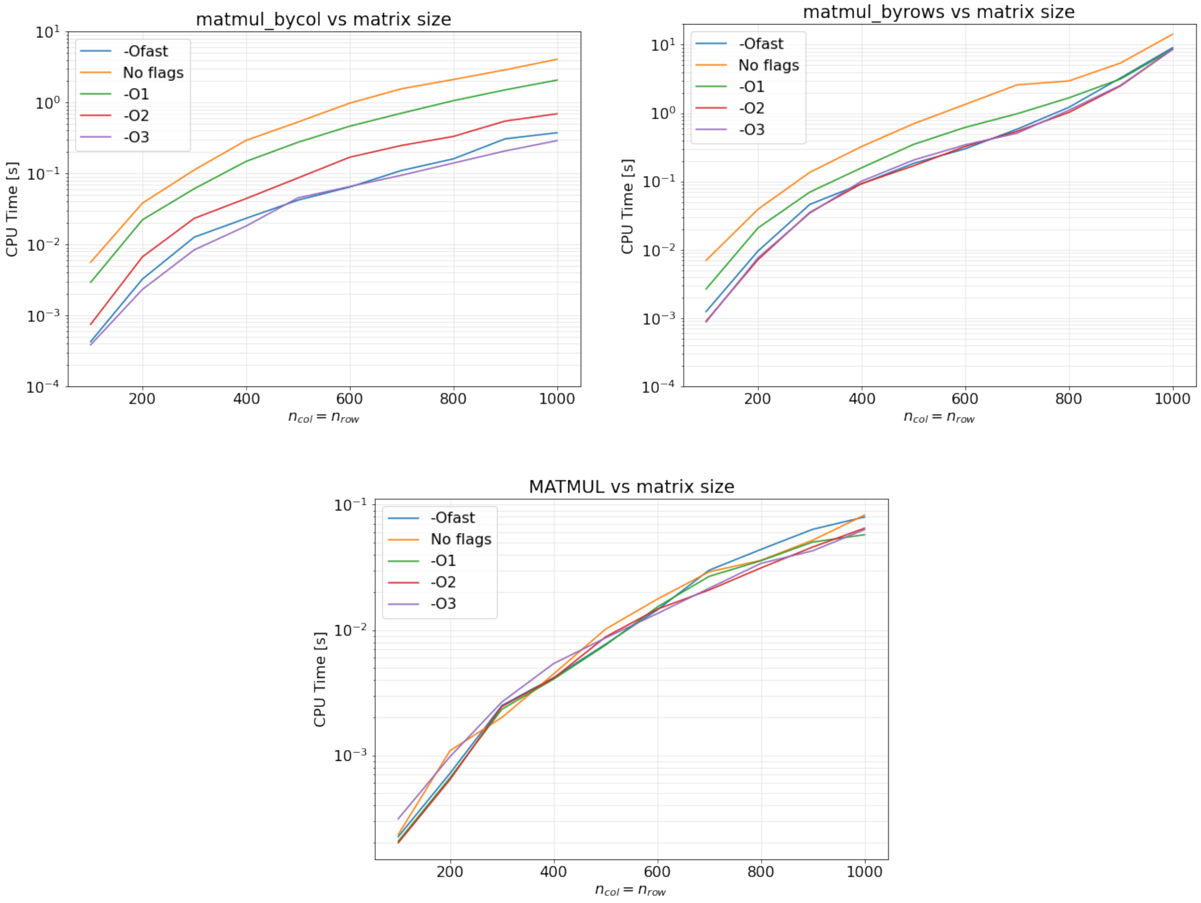


Figure 2: CPU time for the three methods, for different input sizes and different flags.