Quantum Information and Computing (QIaC)

Prof. Simone Montangero

Tommaso Faorlin

Università degli Studi di Padova - Physics of Data

tommaso.faorlin@studenti.unipd.it

November 22, 2021

Theory

- **Exercise 1**. Fit of the scaling of matrix-matrix multiplication CPU Time vs N

$$\forall N \in [N_{min}, N_{min} + 20, \ldots, N_{max}] \Rightarrow \sum_{kk=1}^{N} A_{ii,kk} B_{kk,jj}$$

  where $N_{min} = 20$ and $N_{max} = 2020$ with 3 methods:
  - `matmul_byrow`: *ii* (row of mat_A) ,*kk* (col of mat_B), *jj* (col of mat_A)
  - `matmul_bycol`: *jj* (**col** of mat_A) ,*kk* (**col** of mat_B), *ii* (row of mat_A)
  - `MATMUL`: FORTRAN intrinsic subroutine (the fastest!)

  Since we use 3 for loops on N values we **expect a scaling** of $\mathcal{O}(N^3)$

- **Exercise 2**. The **spacing between eigenvalues** stored in **crescent order** s of a **generic** hermitian matrix (both of size $N = 2500$ in our case) follows the *Wigner surmise* distribution:

$$P(s) = as^{\alpha} e^{-bs^{\beta}} \approx \frac{32s^2}{\pi^2} e^{-\frac{4s^2}{\pi}} \qquad \text{Exact for } N = 2. \tag{1}$$

Code development (Ex.1)

- Automatized the launch of the program `w3_ex1.f90` via a Python script `script.py` and saved the results in a `.txt` file for the three methods.

```
!part of FORTRAN code to read from command line
CHARACTER(10) :: command_line_arg
CALL GET_COMMAND_ARGUMENT(1, command_line_arg)
READ(command_line_arg, FMT="(I8)") rows_mat_A
```

```
#Python code to read from output of FORTRAN code
sp.run(['gfortran', src , 'my_mat_mul.f90','error_handling.f90','-o', exe])
for nn, N in enumerate(N_vector):
    output = sp.Popen( [('./' + exe), str(N)], stdout=sp.PIPE ).communicate()[0]
    times = output.decode('utf-8').split()
```

- Fitted a polynomial function and determined the scaling of CPU time vs size of the matrices (using `np.polyfit` from `numpy` and `mean_squared_error` from `sklearn.metrics`).

Code development (Ex.2)

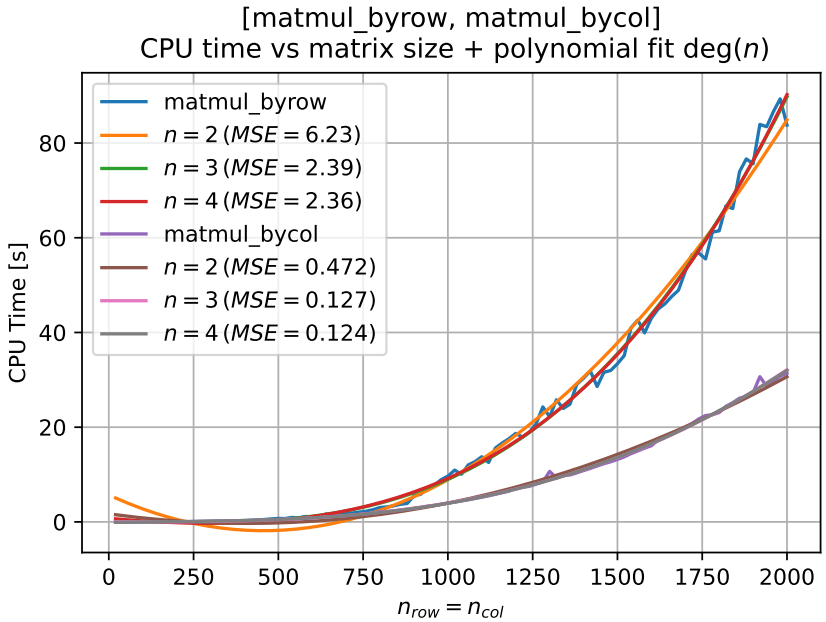
- Expanded the module dc_matrix defined for Assignment 2 including: `hermitian_init`, `diagonal_init`, `eigenvalues`, `norm_eigenvalues_spacing`.

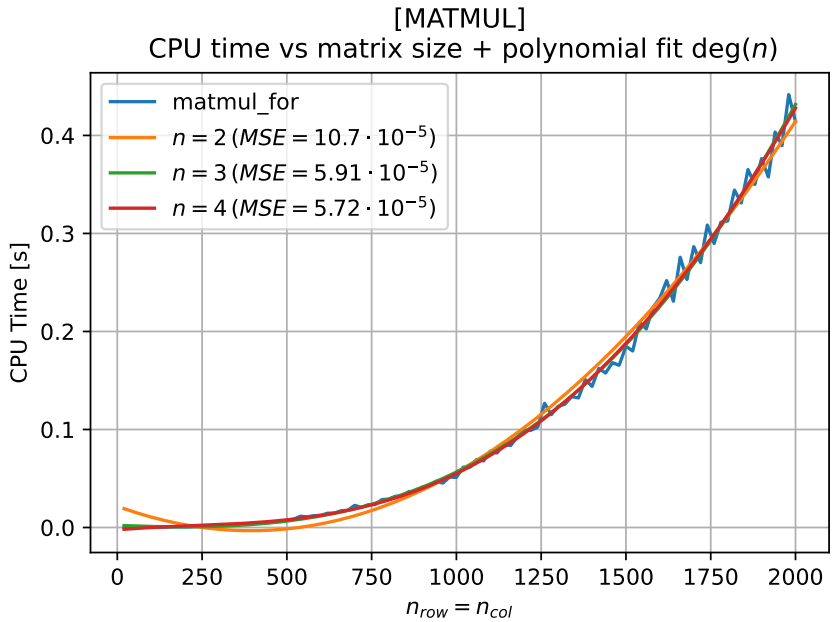
```
FUNCTION hermitian_init(input_dims) RESULT(matrix_out)
    !  [...]
    DO jj = 1, input_dims(1)
        DO ii = 1, input_dims(2)
            IF (ii.EQ.jj) THEN
                matrix_out%m_el(ii, jj) = COMPLEX(-1+2*RAND(),0d0)
            ELSE IF (jj.GT.ii) THEN
                matrix_out%m_el(ii, jj) = COMPLEX(-1+2*RAND(),-1+2*RAND())
                matrix_out%m_el(jj, ii) = CONJG(matrix_out%m_el(ii, jj))
            ELSE
```

```
FUNCTION eigenvalues(matrix_in) RESULT(vector_output)
    !  [...]
    IF (matrix_in%m_dims(1).EQ.matrix_in%m_dims(2)) THEN
        N = matrix_in%m_dims(1)
        ALLOCATE(vector_output(N))
        !ZHEEV computes the eigenvalues and eigenvectors for HE matrices
        CALL ZHEEV('N','U', N, matrix_in%m_el, N, vector_output, work, ...)
        !DLASRT sorts numbers in increasing or decreasing order
        CALL DLASRT('I', N, vector_output, INFO)
    !  [...]
```

CPU Time vs matrix size for three methods (Ex.1)

We report in the following graphs the **results** obtained with the **three methods**, considering the **polynomial** function (order n) for the fit: $y = \sum_{i=1}^{n} a_i x^i + a_0$



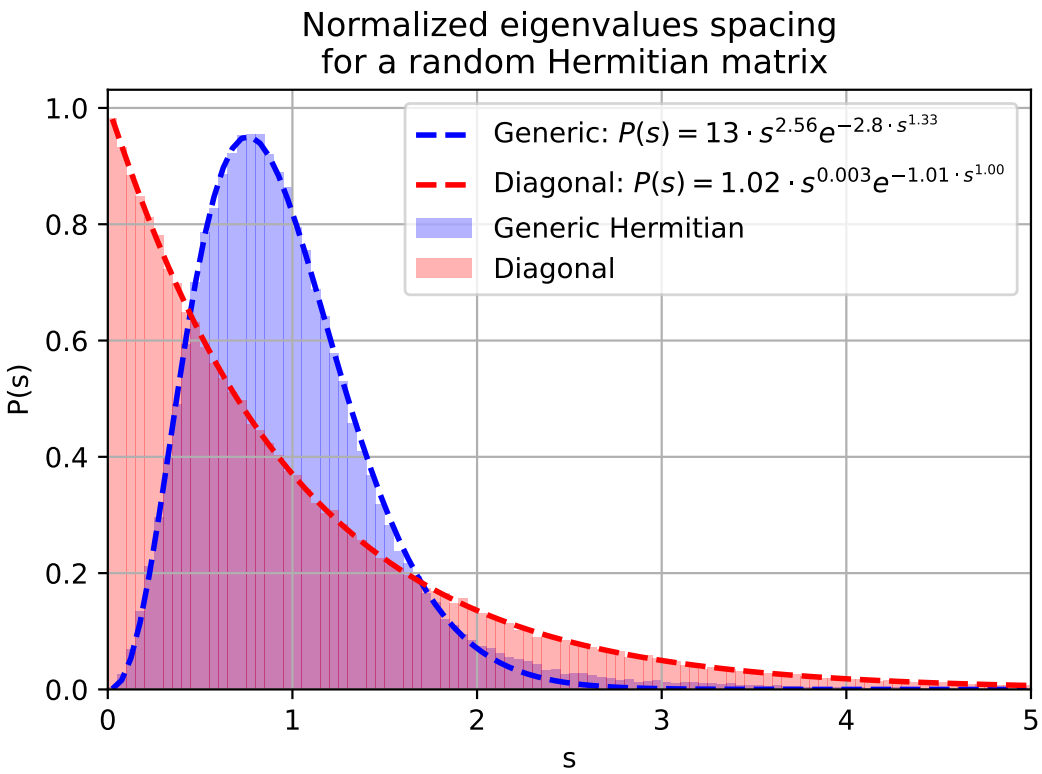


$o(4)$ polynomial fit	a_3	a_4
`matmul_bycol`	6.4e-09	−6.2e-13
`matmul_byrow`	3.6e-09	2.3e-12
`MATMUL`	1.2e-10	−1.8e-14

Table: a_3 and a_4 for the $n = 4$ polynomial fit.

The MSE for $n = 4$ is smaller than the one for $n = 3$ but the coefficients for the fits a_4 and a_3 are more than 3 order of magnitude apart in all cases. For **Occam's razor** we can state that the divergence is $\mathcal{O}(N^3)$.

Normalized eigenvalues spacing (Ex.2)



Sampling is done by running 80 times the code for a $N = 2500$ matrix. The eigenvalues λ_i are stored in **crescent order** and their **normalized spacing** is:

$$s_i = \frac{\lambda_{i+1} - \lambda_i}{\overline{\Delta\lambda}} \tag{2}$$

We use the same fitting function in Eq. 1 for both the Hermitian and the diagonal case. The parameters for the former distribution **differ** from the theoretical ones, but we are able to **distinguish** the two curves.

	a	b	α	β
Generic	13 ± 2	2.8 ± 0.2	2.56 ± 0.09	1.33 ± 0.04
Diagonal	1.02 ± 0.02	1.01 ± 0.02	0.003 ± 0.005	1.00 ± 0.01

Table: Results for the fits on 100 points.