Multi-run script and Automated fits

Quantum Information and Computing - Homework #4

Beatrice Segalini

Università degli Studi di Padova

November 3rd, 2020

Requests

Exercise 1: Multi-run script

Consider the program developed in Exercise 3 of Week 1.

a) Define the matrix dimension N as an input value to be read from file.

b) Write a `Python` script that changes N between two values N_{min} and N_{max}, and launches the program. Store the results in different files depending on the multiplication method used.

c) Plot (using `gnuplot`) the results for the different multiplication methods.

Exercise 2: **Automated fits**

Consider the program of the previous exercise.

a) Fit the scaling of the time needed for different methods as a function of the input size. Consider the biggest possible difference between N_{min} and N_{max}.

b) Save the `gnuplot` file you used in part 1 and exploit it to write a `Python` script that performs automatically the previous fits.

Adapt input and Python script

1a) Read input from file

```
open(1, file=inputfile)

do

read(1, *, end=2) N

[......]

end do

2 print *, "Execution complete."

close(1)
```

1b) Run script from Python

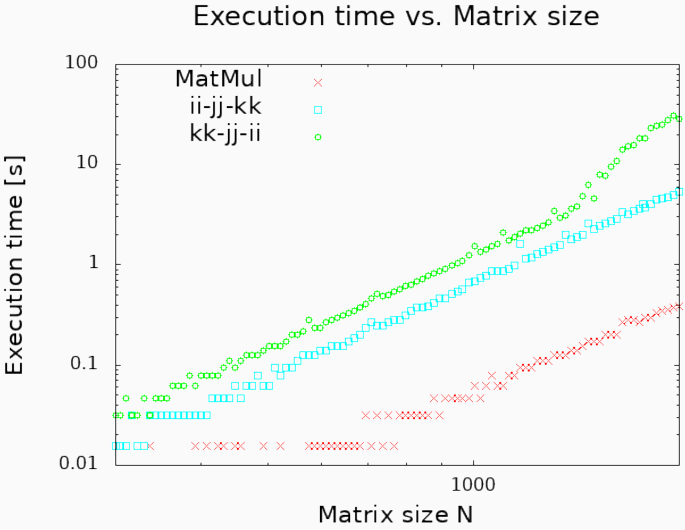
```
Ns = np.logspace(np.log10(Nmin), np.log10(Nmax+1), num=points,
dtype=int)

np.savetxt("./N.dat", Ns, fmt="%d")

subprocess.run(["gfortran", "Ex4-Segalini-CODE.f90", "-o", "runo2.exe",
"-O2"])

subprocess.run(["./runo2.exe"], stdout=subprocess.PIPE)
```

Plots of collected data



1c) Plot of execution times as a function of the size of the matrix *N*. Plotted with `gnuplot`.

Fitting

Fitting function:

$$f(x) = kx^h \implies f_{log}(x) = a + bx$$

Parameters found:

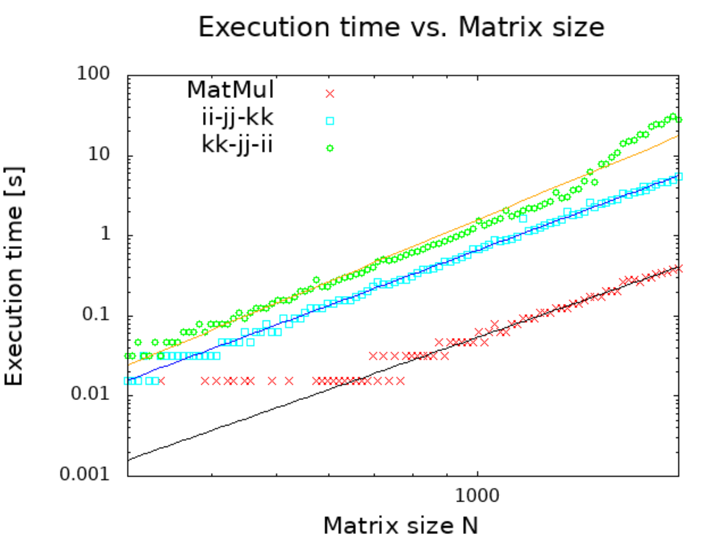
Product	$a \pm \sigma_a$	$b \pm \sigma_b$
MatMul	-10.0352 ± 0.1927	2.92229 ± 0.06258
ii-jj-kk	-9.47615 ± 0.06157	3.09851 ± 0.02121
kk-jj-ii	-10.1999 ± 0.1365	3.4666 ± 0.04708

Fit parameters and associated errors computed by `gnuplot`.

Python automated fits

```
plot = subprocess.run(["gnuplot", "fit.gnu"])
```

Fit comparison



2b) Plot of execution times as a function of the size of the matrix *N*, with correspondent fitted lines.

Conclusions

Further improvements:

- test with different optimisation flags;
- try also other loop permutations;
- use `Python` to create the `.gnu` files;
- implement a more clever fit mode that chooses better fit ranges.

What I learned

- manage I/O from files;
- use `Python` to run `Fortran` programs;
- adjust graphic settings of `gnuplot`;
- quantify the time scaling behaviour of matrix-matrix multiplication with different methods.