Assignment 3

https://github.com/jchryssanthacopoulos/quantum_information/tree/main/assignment_3

Quantum Information and Computing
AA 2022–23

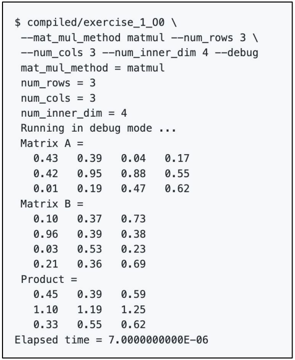
James Chryssanthacopoulos

22 November 2022

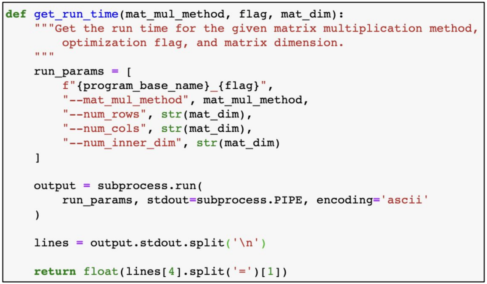
UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Exercise 1: Matrix Multiplication Scaling

- Matrix multiplication program extended to parse command-line arguments using `get_command_argument`
- Python script written in Jupyter notebook to make `subprocess` calls to Fortran program to get runtimes



```
$ compiled/exercise_1_O0 \
 --mat_mul_method matmul --num_rows 3 \
 --num_cols 3 --num_inner_dim 4 --debug
 mat_mul_method = matmul
 num_rows = 3
 num_cols = 3
 num_inner_dim = 4
 Running in debug mode ...
 Matrix A =
   0.43   0.39   0.04   0.17
   0.42   0.95   0.88   0.55
   0.01   0.19   0.47   0.62
 Matrix B =
   0.10   0.37   0.73
   0.96   0.39   0.38
   0.03   0.53   0.23
   0.21   0.36   0.69
 Product =
   0.45   0.39   0.59
   1.10   1.19   1.25
   0.33   0.55   0.62
Elapsed time = 7.0000000000E-06
```



```python
def get_run_time(mat_mul_method, flag, mat_dim):
    """Get the run time for the given matrix multiplication method,
        optimization flag, and matrix dimension.
    """
    run_params = [
        f"{program_base_name}_{flag}",
        "--mat_mul_method", mat_mul_method,
        "--num_rows", str(mat_dim),
        "--num_cols", str(mat_dim),
        "--num_inner_dim", str(mat_dim)
    ]

    output = subprocess.run(
        run_params, stdout=subprocess.PIPE, encoding='ascii'
    )

    lines = output.stdout.split('\n')

    return float(lines[4].split('=')[1])
```

Exercise 1: Matrix Multiplication Scaling

Method	O0	O1	O2	O3	Ofast
`row-col`	3.01	3.01	3.00	3.00	3.00
`col-row`	3.02	3.01	3.02	3.00	3.01
`matmul`	2.34	2.64	2.57	2.66	2.61

Table: Polynomial time complexity, given by coefficient a in fit $\log T = a \log N + b$