

Assignment 1

https://github.com/jchryssanthacopoulos/quantum_information/tree/main/assignment_1

Quantum Information and Computing AA 2022–23

James Chryssanthacopoulos
1 November 2022



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Exercise 1: Setup (1)



- First program to test basic features
- Contains module and function to compute square root

```
module first_module
  implicit none

  real*8 var1, var2

  contains
    function mysqrt(x) result(sx)
      real*8 x
      real*8 sx
      sx = sqrt(x)
    end function

end module first_module
```

- Program output

The square root of 5.0 is 2.2361

Exercise 1: CloudVeneto (2)



- Key-pair and virtual machine created on CloudVeneto
- Private key copied onto gateway machine

```
scp /path/to/private/key [username]@gate.cloudveneto.it:~
```

- SSHed into gateway machine, then VM

```
ssh [username]@gate.cloudveneto.it
```

```
ssh -i /path/to/private/key ubuntu@[VM_IP_address]
```

- Installed gfortran
- git cloned my repository
- All code compiled and executed

Exercise 2: Number precision



- Program to test limits of integers and real numbers

- **Part (a)**

- Add 2.000.000 and 1 using `INTEGER*2` and `INTEGER*4`
- Since `INTEGER*2` only has range of $\approx 10^4$, storing 2.000.000 causes overflow

The sum of -31616 and 1 using `INTEGER*2` is -31615

The sum of 2000000 and 1 using `INTEGER*4` is 2000001

- **Part (b)**

- Sum $\pi \cdot 10^{32}$ and $\sqrt{2} \cdot 10^{21}$ with single and double precision
- Since single has 8 digits of precision, summing has no effect

The sum of 3.14159278E+32 and 1.41421360E+21 using `REAL*4` is 3.14159278E+32

The sum of 3.1415926535897933E+32 and 1.4142135623730950E+21 using `REAL*8` is 3.1415926536039354E+32

Exercise 3: Performance testing (1)



- Program to implement matrix multiplication and time it
- Matrices multiplied using three for loops, resulting in $\mathcal{O}(n^3)$ time complexity

```
do i = 1, n_1
  do j = 1, n_2
    do k = 1, n_3
      matrix3(i, j) = matrix3(i, j) + matrix1(i, k) * matrix2(k, j)
    end do
  end do
end do
```

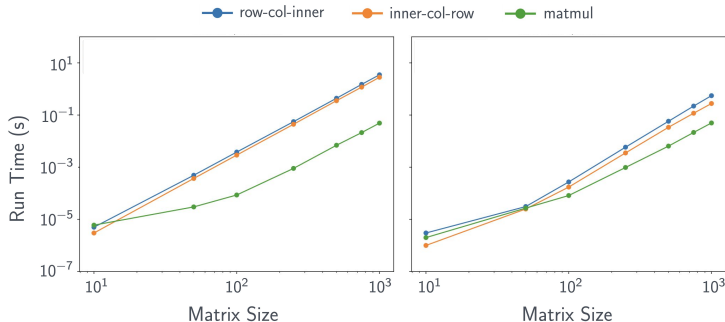
- Two different loop orders used, row-col-inner and inner-col-row
- Methods timed against builtin matmul method, for example

```
Elapsed time for matmul = 2.1500000000E-04
Max abs error for row-col-inner = 3.1974423109E-14
Elapsed time for row-col-inner = 8.0370000000E-03
Max abs error for inner-col-row = 3.1974423109E-14
Elapsed time for inner-col-row = 6.2030000000E-03
```

Exercise 3: Performance testing (2)



- Different optimizations were used: 01–03, 0s, and 0fast
- 0fast reduced performance gap between `matmul` and custom methods the most, $\sim \mathcal{O}(n^{2.8})_{\text{custom}}$ to $\sim \mathcal{O}(n^{2.2})_{\text{matmul}}$



(a) No optimization

(b) 0fast optimization

Exercise 3: Performance testing (3)



Performance of other optimization methods (run on 10-core M1 Pro 32 GB)

