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

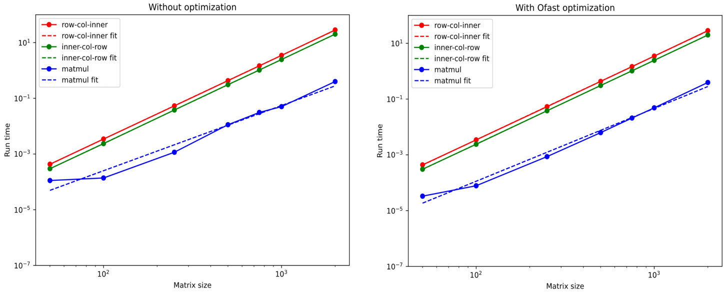
- Matrix multiplication program extended to parse command-line arguments using `get_command_argument`
- Python script written in Jupyter notebook to make `subprocess` calls to Fortran program to get runtimes

```
$ 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

Linear fit of log run time to log matrix size performed with numpy's `polyfit`



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 $\log T = a \log N + b$

Exercise 2: Eigenproblem

Università degli Studi di Padova

- Hermitian matrix created with random number generator
- Eigenvalues computed using `zheev`

```fortran
function rand_hermitian_matrix(n) result(H)
    integer n
    integer ii, jj
    complex*16, dimension(n, n) :: H

    do ii = 1, n
        do jj = 1, ii
            if (ii /= jj) then
                ! sample random complex numbers off the diagonal
                H(ii, jj) = cmplx(RAND(0)*2 - 1, RAND(0)*2 - 1)
                H(jj, ii) = conjg(h(ii, jj))
            else
                ! sample real numbers on the diagonal
                H(ii, ii) = RAND(0)*2 - 1
            end if
        end do
    end do
end function
```

```fortran
! compute eigenvalues in ascending order
call zheev('N', 'U', ndim, H, ndim, eigvals, work, lwork, rwork, info)

! compute eigenvalue spacings and average
ave_delta_eigvals = (eigvals(ndim) - eigvals(1)) / (ndim - 1)
do ii = 1, ndim - 1
    norm_eigval_spacings(ii) = (eigvals(ii + 1) - eigvals(ii)) / ave_delta_eigvals
end do
```

```
$ compiled/exercise_2 --ndim 2 -d
 mat_type = hermitian
 output_filename = histogram.csv
 ndim = 2
 nsamples = 1
 nbins = 100
 min_val = 0.000
 max_val = 5.000
 Sampled Hermitian matrix =
-1.0000 +0.0000i   -0.7369 -0.5112i
-0.7369 +0.5112i   -0.0827 +0.0000i
 Eigenvalues =  -1.5486887628051993       0.46600424589846079
 Normalized eigenvalue spacings =   1.0000000000000000
 Average eigenvalue spacing =   2.0146930087036603
```

Exercise 3: Random Matrix Theory

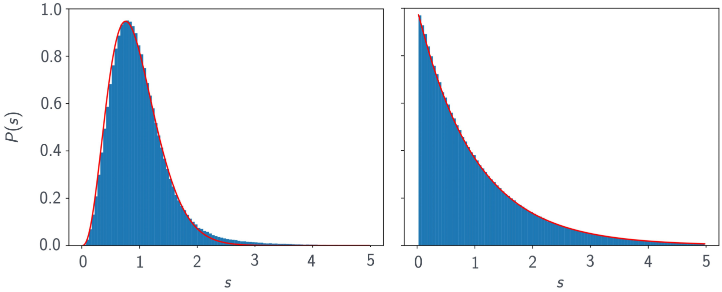
- Distribution of normalized spacings $P(s)$ produced for Hermitian and diagonal matrix
- Fit to $P(s) = as^{\alpha} \exp\left(bs^{\beta}\right)$ performed using `scipy`'s `curve_fit` function

```
def func(x, a, b, alpha, beta):
    return a * x ** alpha * np.exp(b * x ** beta)
popt, pcov = curve_fit(func, bin_centers, norm_count)
popt
```

```
array([13.79272512, -2.82466922,  2.59653888,  1.32155578])
```

Matrix	a	b	α	β
hermitian	13.792 ± 2.266	-2.825 ± 0.166	2.597 ± 0.091	1.322 ± 0.043
diagonal	1.017 ± 0.007	-1.018 ± 0.008	0.005 ± 0.002	0.987 ± 0.005

- Histograms produced with matrix size $N = 1000$ and number of bins $n_{\text{bins}} = 100$
- Results agree with random matrix theory (cf. Edelman and Rao, *Random Matrix Theory*, *Acta Numerica*, 2005, pp. 41)



(a) Hermitian

(b) Diagonal