EXERCISE 3
MATRIX – MATRIX MULTIPLICATION & RANDOM MATRIX THEORY

QUANTUM INFORMATION AND COMPUTING COURSE 2021/2022

ALESSANDRO MARCOMINI (2024286)

PROF. SIMONE MONTANGERO

11/23/21

EXERCISE GOALS

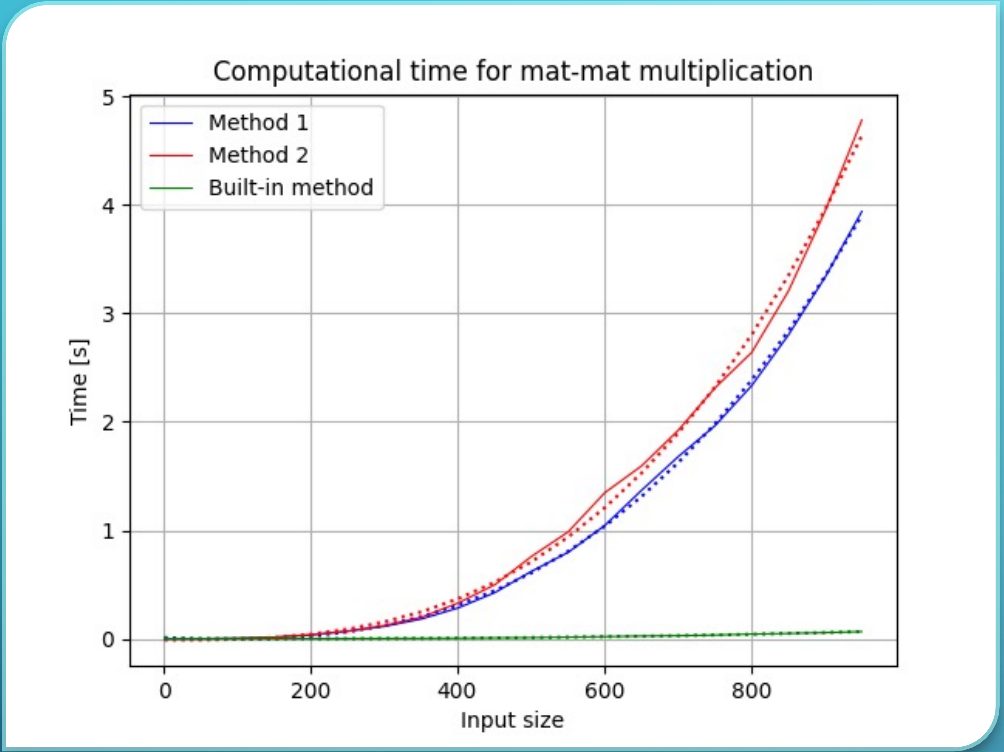
PART I

- Investigate the scaling of different matrix – matrix multiplication subroutines in FORTRAN via PYTHON interface

PART II

- Generate large random Hermitian matrices and diagonal real matrices
- Investigate the statistical distribution of eigenvalues spacings of these matrices

PART I – MATRIX MATRIX MULTIPLICATION




```
! Iterative procedure - method 1
do I = 1,size(1)
    do J = 1,size(3)
        do K = 1, size(2)
            C(I,J) = C(I,J) + A(I,K)*B(K,J)
        end do
    end do
end do
```

```
! Iterative procedure - method 2
do J = 1,size(3)
    do I = 1,size(1)
        do K = 1, size(2)
            C(I,J) = C(I,J) + A(I,K)*B(K,J)
        end do
    end do
end do
```

- Three methods under investigation: most external loop on output's rows, on output's columns, built-in (optimized) multiplication method MATMUL
- Verification of compatibility among methods (sum up to 10^{-6})
- Interfaced via F2PY
    - Does not work with allocatable variables…

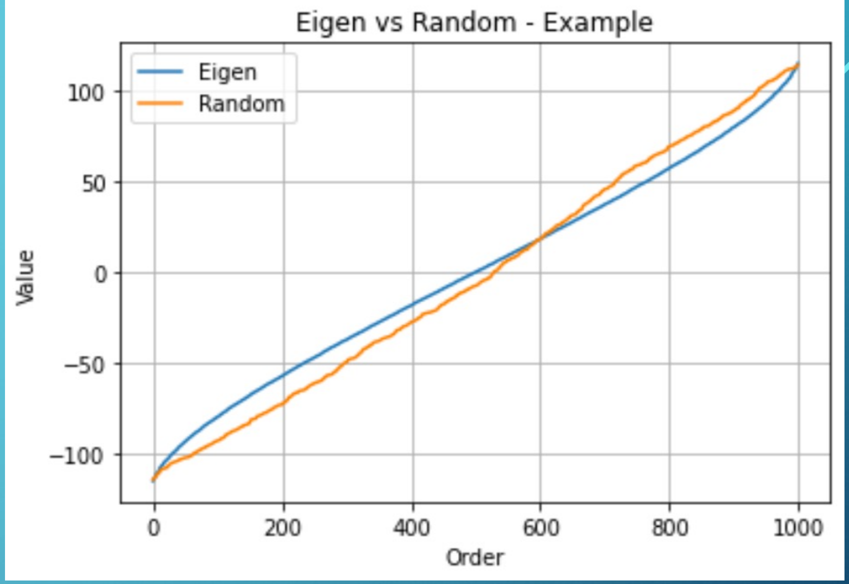
```
#Loop over data size
for N in range(1,maxsize,50):
    print(N)
    output_time = matmat_mult.random_mat_mult([N,N,N])
    output_time[0] = N
    data.append(output_time)
```

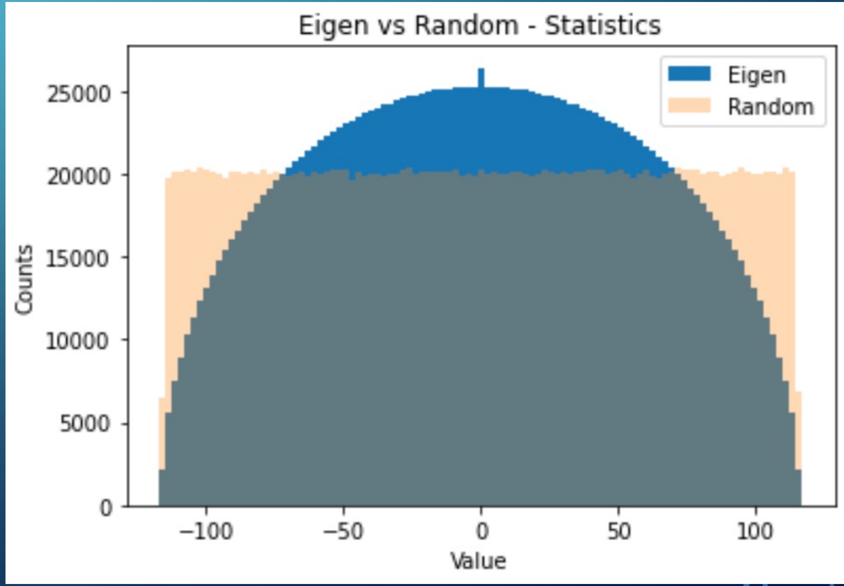
- Fit – scipy.polyfit(order = 3)

PART II – RANDOM MATRIX THEORY

- Implementation of Hermitian case in the user defined type
- Generation of 2k random Hermitian matrices of size (1k x 1k) with Re, Im uniformly in $(-1,1)$ and random real diagonal matrices
- Eigenvals calculation via ZHEEV
- Visualization and test (compatibility with trace up to 10^{-6})

```
call new_random_cmat(H,SIZE,SIZE,hermit=.TRUE.)
call zheev( 'Numbers', 'Lower', N, H%mel, LDA, W, WORK, LWORK, RWORK, INFO )
write(89,'(*(G0.6,:,","))') W
```



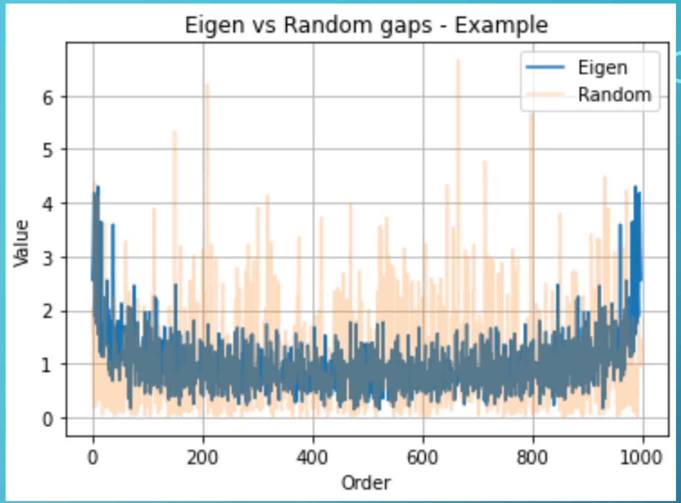


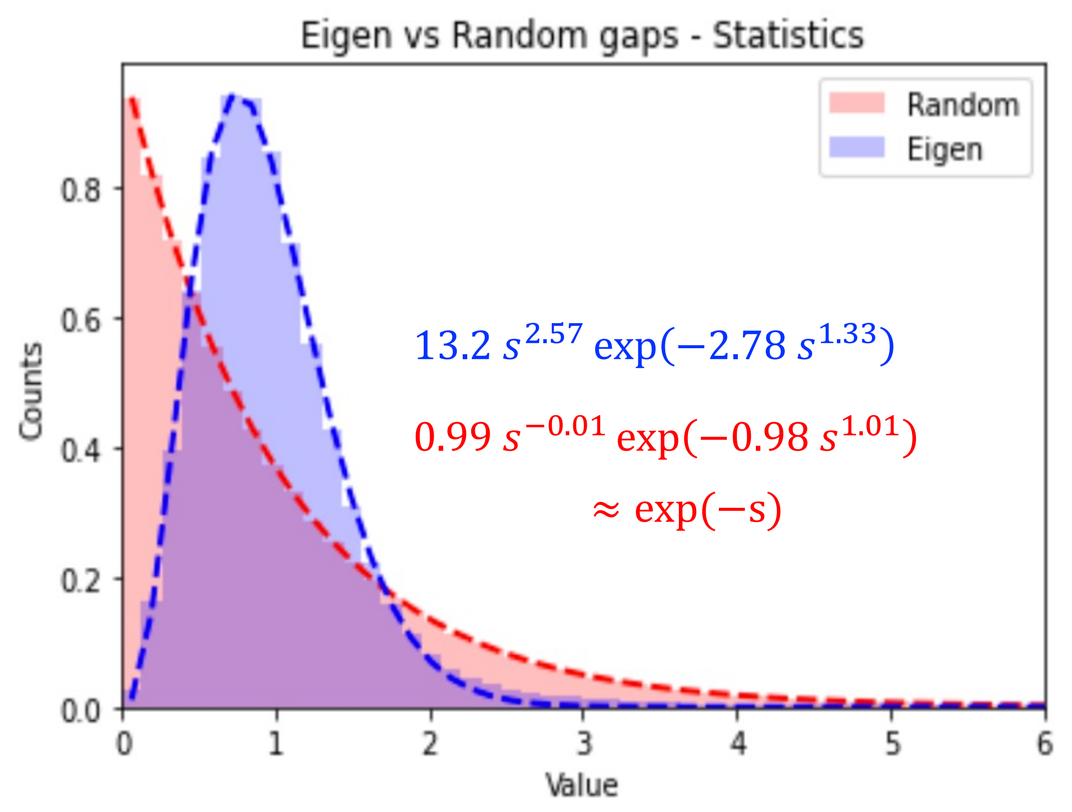
PART II – RANDOM MATRIX THEORY

- Calculation of spacings and global average normalization: $s_i = \Delta\lambda_i / \overline{\Delta\lambda}$

```
# Calculate normalized spectral gaps
gaps_e = np.diff(eigendata, axis = 1)
means_e = np.mean(gaps_e, axis = 1)
for i in range(len(gaps_e)):
    gaps_e[i,:] /= means_e[i]
```

- Histograms and interpolations
  - Overflow encountered
  - Rebinning
  - Scipy problems in fitting negative numbers with real exponent





WHAT SHOULD I HAVE EXPECTED?

MATRIX MATRIX MULTIPLICATION

- Expected scaling order: $O(n^3)$
- Fortran stores matrix elements columnwise: impact on computational time
- Built-in method optimized

RANDOM MATRIX THEORY

- Wigner theory:
  - Semicircle law
  - Wigner’s surmise
- Random real values:
  - Uniformly distributed values
  - Exponential (Poissonian) distribution of spacings