Exercise 5: *Eigenvalue Spectral Analysis of Random Matrices*

Iriarte Delfina (1231682)

Quantum Information

November 10, 2020

Abstract

In this report, the eigenvalue spectra of Hermitian complex matrices are study. In particular this is done for a fully complex matrix and a diagonal one where the normalized spacing between the eigenvalues are study globally and locally. The distribution $P(s)$ of normalized spacing between neighbouring eigenvalues are fitted using GNUPLOT and compared with the *Wigner surmise* distribution.

1 Introduction

Random matrix ensembles are extensively used as models for describing the **statistical properties** of levels of complex systems such as heavy nuclei and many electron atoms. The matrix elements are independent **random variables** chosen by a probability distribution and the matrices define a statistical matrix ensemble. In particular, we restring ourselves only for the case of **Hermitian operators** which corresponds to complex square symmetric matrices with entries $a_{ij} = \overline{a_{ji}}$.

The **statistical properties of eigenvalues** of many-body quantum systems has been study for many years. Furthermore, it has been shown that for fully random matrices, the distribution $P(s)$ of normalized spacing between neighbouring eigenvalues is approximately described by the well-known **Wigner surmise distribution**:

$$P(s) = 2\left(\frac{4s}{\pi}\right)^2 e^{-\frac{4s^2}{\pi}} \tag{1}$$

which is an exact expression for 2×2 Hermitian matrices.

In this project, we are going to study the **spacing distribution of eigenvalues of random Hermitian matrices** of size 2500×2500. In particular for a fully complex random matrix and a diagonal one. The normalized spacing is defined as:

$$s_i = \frac{\lambda_{i+1} - \lambda_i}{\Delta\lambda} \tag{2}$$

where $\Delta\lambda$ is the average of $\Delta\lambda_i$ which can be computed **globally**, considering all eigenvalues at once, or **locally**, over a different number of levels around λ_i. The distribution $P(S)$ is study for 150 sampled matrices, and, inspired by the Wigner surmise Equation 1 the data is fitted by using:

$$P(s) = as^{\alpha} e^{-bx^{\beta}} \tag{3}$$

Finally, the computation of the average of

$$r_i = \frac{\min(\Delta\lambda_i, \Delta\lambda_{i+1})}{\max(\Delta\lambda_i, \Delta\lambda_{i+1})} \tag{4}$$

is reported.

2 Code development

The initialization of a double complex random Hermitian matrix is done by implementing a **Box-Muller algorithm** which generates Gaussian distributed random numbers as shown in Listing 1. Notice that in the case of the diagonal matrix the procedure is the same but this time setting the diagonal entries with real entries only whereas the off diagonal entries are zero.

```
do ii  = 1, nn
    do jj= 1, nn
        call random_number(xx)
        call random_number(yy)
        r = sqrt(2*(-log(xx)))
        theta = 2*(2*asin(1._pr)) * yy
        xx = r*cos(theta)
        yy = r*sin(theta)
        mat(ii, jj ) = cmplx (xx, yy)
        mat(jj, ii ) = conjg (mat (ii,jj))
        if (ii == jj ) then
            mat(ii, jj )= cmplx(xx, 0)
        end if
        end do
end do
```

Listing 1. FORTRAN SUBROUTINE for generating a double complex matrix with entries given by the **Box-Muller algorithm**

The computation of the eigenvalue of a $n \times n$ complex Hermitian matrix `A` is done by implementing the `zheev(JOBZ, UPLO, N, A, LDA, W, WORK, LWORK, RWOEK, INFO)` function provided by LAPACK. In particular, `JOBZ` is a character that tells LAPACK to compute only the eigenvalues if N is set. If `UPLO` is set to `U` then the upper triangle of the double complex matrix `A` is stored. `N` is the order of the matrix A. `LDA` is the leading dimension of the array A, `W` is where the eigenvalues are stored and `RWORK` is a double precision array.

The computation of the eigenvalues is shown in Listing 2.If `INFO = 0` then it was successfully implemented and it returns the eigenvalues in the ascending form.Moreover, we have to tweak the internal parameter of the subroutine by calling with `lwork=-1`.

```
lwork=-1
allocate(work(1))
allocate(rwork(max(1, 3*n-2)))
if(.not.allocated(eig))allocate(eig(n))
call zheev('N','U',n,matr,n,eig,work,lwork,rwork,info
lwork = int(real(work(1)))
deallocate(work)
deallocate(rwork)
allocate(work(max(1,lwork)))
allocate(rwork(max(1, 3*n-2)))
call zheev('N','U',n,matr,n,eig,work,lwork,rwork,info)
deallocate(work)
deallocate(rwork)
```

Listing 2. FORTRAN SUBROUTINE for computing eigenvalues of a complex matrix of size $N \times N$

Finally, the average ratio given in Equation 4 is computed in a FORTRAN SUBROTUINE where an array r with the computed value is returned. The local spacing are computed using the SUBROUTINE shown in Listing 3. It takes as input the vector of different interval spacing, meaning $N/2, N/10, \ldots$, the eigenvalues and returns the local spacing already computed for the several spacings by taking as input the index of the spacing vector.

```
temp_s = spac(i_space)
min_index = MAX(1, i_eigen - (temp_s/2))
max_index = min ( i_eigen + (temp_s/2), num_eigen-1)
delta = eigenvalues(i_eigen+1) - eigenvalues(i_eigen)
summ = SUM(eigenvalues((min_index +1):(max_index +1)) - eigenvalues(min_index:max_index)  )
t_size = size(eigenvalues((min_index +1) :(max_index +1)) - eigenvalues((min_index) :(
    max_index))  )
resuls = delta/(summ/t_size)
```

Listing 3. Local spacing computation

In Listing 4 the main code is shown. Recall that this code is done by looping over the sampled matrices where the values are saved into an output file. In particular a logical element `diag` was implemented in order to initialized either a diagonal or a full matrix.

```
call eigenvalues(matrix, n, work, eig, rwork,  lwork, info) !Compute eigenvalues
do i = 2 , n!Compute spacing
    space(i-1) =  eig(i) - eig(i-1)
end do
rat = ratio(space, n-1)        !Compute ratio
avg_space = sum(space)/(n-1)!Normalize average spacing
space =  space/avg_space
do i = 1, size(spacings)!Calculate local spacing
    do j = 1, n-1
        call calculate_spacing(spacings, i, eig, j, n, resuls)
        s_i(i,j) = resuls
    end do
end do
```

Listing 4. Main code

The histograms are computed using NUMPY in PYTHON as show in Listing 5. The bins and counts are saved into an output file where then they are fitted using GNUPLOT. Again this is automated using PYTHON where GNUPLOT is called using the `subprocess` library.

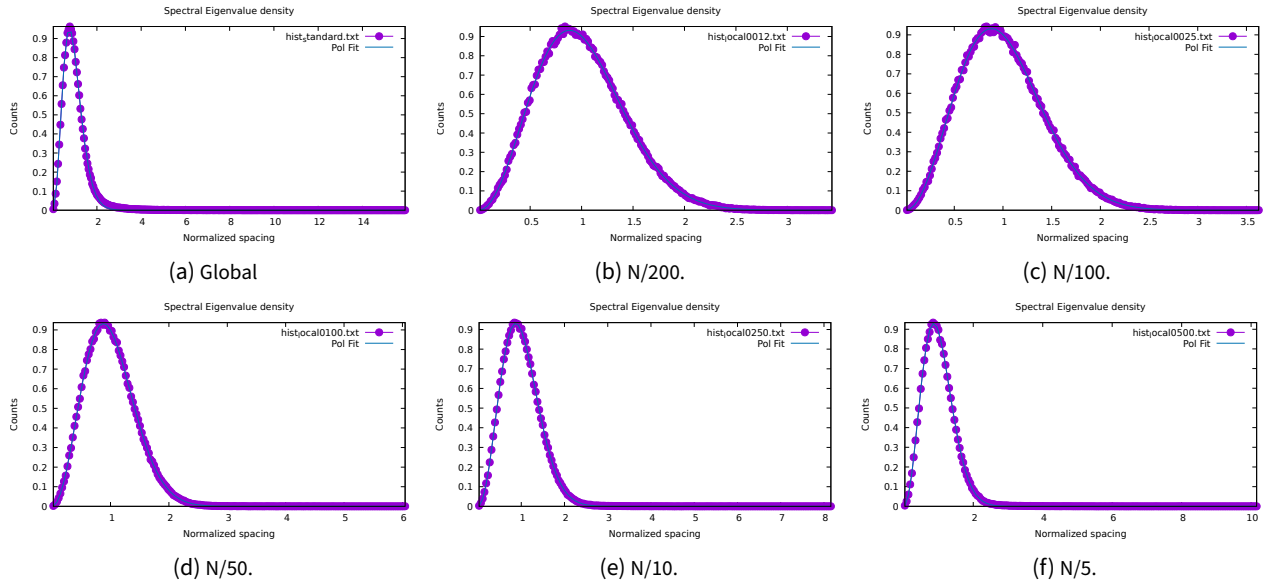
```
data = np.loadtxt("eigenvalues_spacing.txt")
hist, bin_edges =np.histogram(data, density = 1, bins  =30)
bin_centres = (bin_edges[1:]+bin_edges[:-1])/2
graph_ = pd.DataFrame([bin_centres, hist]).T
graph_.to_csv(str("hist_standard.txt"), header=None, index=None, sep="\t")
```

Listing 5. Histogram generation

3 Results

Figure 1 shows the spectral eigenvalue density for the global and some locally ones of a 2500×2500 Hermitian matrix with 150 sampled matrices. Notice that more information can be found when the spacing interval is localized into some region. In fact, one can compare the size of the $x-$axis and indeed can be seen that for smaller interval the data obtain is more meaningful, whereas in the others more zero-values can be found.The parameters obtained of the fitting in GNUPLOT are shown in Table 1.

Furthermore, it can be seen that the **Wigner surmise** Equation 1 fit with the data properly as seen in Figure 2. Moreover, the **Wigner** fits better for the data founds locally which makes sense because it has a more "detail" data than the global one, which holds general information.



(a) Global (b) N/200. (c) N/100.

(d) N/50. (e) N/10. (f) N/5.

Figure 1. Spectral normalized eigenvalue for a fully double complezx matrix of size 2500 × 2500.

Table 1. Parameters obtain of fitting a double complezx matrix of size 2500 × 2500.

Spacing λ_i	**a**	**alpha**	**b**	**beta**
Avg Spacing	14.0553340404903	2.6001583494858	2.8449055443227	1.32010821103149
N/200	3.35740073082372	2.01460344907661	1.31479514532526	1.95590367100242
N/100	3.35530047662223	2.01013323860852	1.31449916396975	1.9566492536886
N/50	3.36810042819539	2.01107642354898	1.31854748012557	1.95507633864006
N/25	3.44553859309805	2.02274989744634	1.34304030649198	1.93706697294175
N/10	3.59194433241422	2.03950077548869	1.38953962994789	1.90415886679865
N/5	3.89563086382039	2.07763994026452	1.47866488056657	1.84496887627572
N/2	5.10754473648372	2.21254536830112	1.77407313671586	1.67360016750083

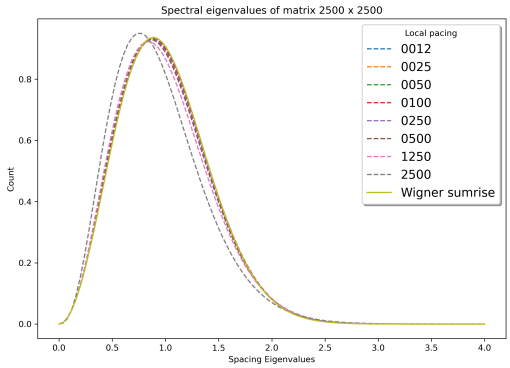
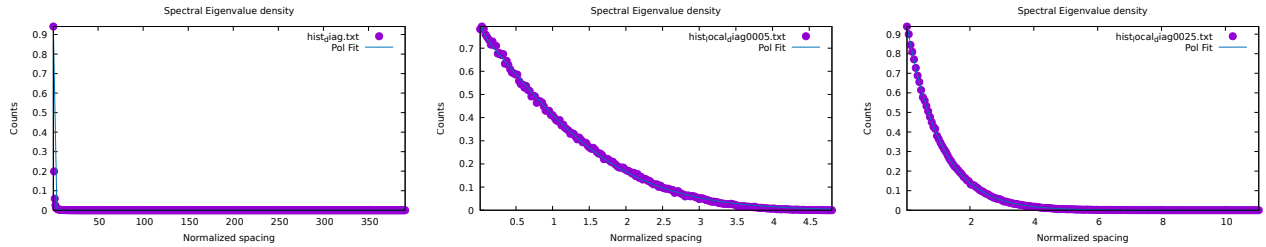
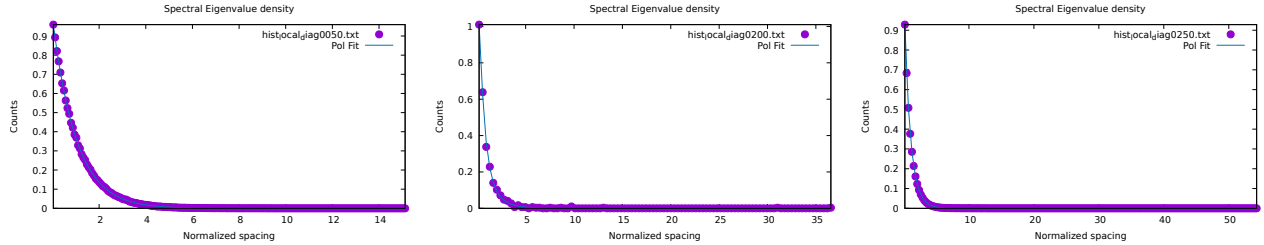


Figure 2. Reproduction of the probability distribution with the parameters obtains of the fitting

Finally, Figure 3 shows the histogram of a 2500 × 2500 square diagonal matrix for global and local. It can be seen, as for the case of the fully complex matrix, that better results are found when the spacing is highly local. Indeed, as we argued before, data is more specialized and not generalized leading to better results.



(a) Global. (b) N/500. (c) N/100.



(d) N/50. (e) N/15. (f) N/10.

Figure 3. Spectral normalized eigenvalue for a **diagonal** double complezx matrix of size 2500×2500.

Nevertheless, we can see that in this case the data is more difficult to analyze, and in fact the **Wigner surmise** does not fit properly. Probably it resembles more to an Exponential distribution rather than a Poisson one. Table 2 shows the parameter obtained of the fitting made by GNUPLOT.

Table 2. Parameters obtain of fitting a double complezx matrix of size 2500×2500.

Spacing λ_i	**a**	**alpha**	**b**	**beta**
Avg Spacing	3.53907266626958	-0.146115301962612	2.5977288311296	0.611932664931136
N/500	0.694955355830172	-0.0372485707504143	0.520386559981891	1.43246271831399
N/100	0.942373772817035	-0.00621495761424018	0.924103018663205	1.04260376007897
N/50	0.977852975273681	-0.00539614210880061	0.973349250197404	1.01419178970136
N/15	1.9021393857489	0.0946381928465638	1.6273841926411	0.635162819305610
N/10	1.10812916076104	0.00365185216359804	1.1231399034276	0.947549468799194

Finally the average of the quantities r are reported in Table 3.

Table 3. N = 2500

	Fully complex Hermitian Matrix	**Diagonal Matrix**
$\langle r \rangle$	0.5995294796286627	0.3864954831952835

4 Conclusion

In this project we study the distribution of the normalized spacings of the eigenvalues globally and locally, and compare it with the Wigner sunrise distribution in FORTRAN. The parameter of the distribution were computed by fitting the spectral eigenvalue distribution using GNUPLOT but automatized efficiently in PYTHON. As expected, the local average leads to more accurate result than the global one.