Random Matrix Theory

Abstract

In this assignment, some of the properties of hermitian random matrices are explored: namely, the distribution of the spacings between adjacent eigenvalues of random hermitian matrices and random, real diagonal matrices. The spacings are normalized with respect to the global mean, and also with respect to a local average. A fit is then performed on these distributions: the fitting function is also known as *Wigner surmise*. Lastly, the average ratio between the minimum and the maximum of adjacent spacings is computed for each case.

Theory

Random Matrix Theory, as its name suggests, studies the properties of random matrices. Among the most studied objects are hermitian random matrices; when their elements are drawn from a normal distribution with null mean and unitary variance, these matrices are said to belong in the *Gaussian Unitary Ensemble (GUE)*. Once diagonalized, one can sort the eigenvalues λ_i of these matrices in increasing order, and then compute the spacing $\Delta\lambda_i$ between them as

$$\Delta\lambda_i = \lambda_{i+1} - \lambda_i, \tag{1}$$

and then normalize them by their average $\bar{\Delta\lambda}$. The distribution, then, is expected to follow this equation:

$$p(s) = \frac{32}{\pi^2} s^2 \exp\left(-\frac{4}{\pi} s^2\right) \tag{2}$$

If, instead, one considers a real, diagonal matrix with gaussian distributed entries, no such analytical result is available. Intuitively, however, since all the values in the diagonal are drawn from the same normal distribution, most of the values will fall relatively close to 0, and so most of the differences should also be close to 0. In the general formula for $p(s)$,

$$p(s) = a s^\alpha \exp\left(-b s^\beta\right), \tag{3}$$

the parameter α is linked to the presence of a peak in the distribution. This will prove useful when comparing the results of the two cases, hermitian and diagonal.

A useful tool for this assignment is the Box-Muller transform, which allows to draw two independent, random numbers from a gaussian distribution $\mathcal{N}(0,1)$, by drawing two random numbers from a uniform distribution $\mathcal{U}(0,1)$. The most simple implementation is the following: denoting the gaussian variables by z_1, z_2 and the uniform ones as u_1, u_2,

$$z_1 = \sqrt{-2\ln u_1}\cos(2\pi u_2) \tag{4}$$

$$z_2 = \sqrt{-2\ln u_1}\sin(2\pi u_2) \tag{5}$$

Lastly, the ratio

$$r_i = \frac{\min(\Delta\lambda_i, \Delta\lambda_{i+1})}{\max(\Delta\lambda_i, \Delta\lambda_{i+1})}, \tag{6}$$

can be computed for different locality values, that is, by computing $\Delta\lambda$ as the average over a certain number of spacing values.

Code development

All the references functions and subroutines can be found, together with their documentation, in the modules that come with the source files; these modules are `PROB_DIST`, which contains the utilities for creating the distributions, and `RANDOM_MATRIX`, which is used for the handling of random matrices and the related quantities.

Ex. 1: The first task was to generate a random hermitian matrix, of which to calculate the eigenvalues. In order to draw normal distributed random variables for both the real and imaginary part of each entry, the Box-Muller transformation was employed.

In order to get the eigenvalues of the matrix, the `LAPACK` subroutine `cheev` was used, as all the involved floating point numbers were single precision (both `REAL` and `COMPLEX`). Since using `cheev` requires many arguments, many of which bear no usefulness after the completion of the subroutine, `cheev` was wrapped inside a subroutine, inside of which all required variables are allocated, used and deallocated. `cheev` can actually fail to produce the eigenvalues of a given matrix (either because of illegal values in the passed arguments, or because of lack of convergence in intermediate passages): in this case, the `INFO` variable takes on non-zero values. This was used to repeat the matrix generation and the calculation of its eigenvalues, as long as the `INFO` flag proved it was successful.

The eigenvalues produced by `cheev` are, by default, sorted in ascending order. As a consequence, the spacing could be directly computed, without any sorting step.

Together with the normalized spacings, the various levels of average spacing were computed, and everything was printed on screen. A Python script then redirects this output on an appropriate file in a separate folder.

Ex. 2: For the second exercise, the focus is on creating the distribution $p(s)$ of the spacings for both a hermitian matrix the likes of which have already been described in the Ex. 1, and for a diagonal matrix with real, random entries.

Both of these were actually created in Ex. 1, as the subroutine which initializes the random matrix takes an optional `LOGICAL` parameter, and, depending on its value, creates either a diagonal or a hermitian matrix. Then, since a real diagonal matrix is actually hermitian, the same code that was implemented for the hermitian matrix works for the diagonal, too.

In order to create a distribution, a `HISTOGRAM` subroutine was implemented. It takes a vector as input and produces two vectors, one with the counts per bin, and one with the edges of the bins. Both the range of the histogram and the number of bins can be manually inserted by the user, and are otherwise set to a default value (which, in the case of the range, is from the minimum to the maximum value in the input data). This subroutine is used inside a `NORM_HISTOGRAM` subroutine (which, as the name implies, simply normalizes the output of the `HISTOGRAM` one), and is in turn used by the `PROB_DISTRIBUTION`. All of these share the optional parameters described for `HISTOGRAM`. A probability distribution is obtained by setting an appropriately small bin width, and then dividing the per-bin value provided by the `NORM_HISTOGRAM` by the width of the bin itself. This is, of course, a rather sloppy terminology. From the theoretical point of view, simply taking the limit of the bin width going to 0 is sufficient; however, in practice, this may result in the distribution being very noisy, despite the number of matrices one uses, because of statistical fluctuations.

To avoid this, the number of bins and the range that allow a somewhat smooth dis-

tribution were chosen. This choice will be better explained in the Results section.

In addition to this, a *naive* Kernel Density Estimation subroutine, `KDE`, was implemented. This is an alternate way to estimate the probability distribution of the data. A grid of points is created, $\{x_j^{(g)}\}_{j=1}^{M}$ and for each point in the input file, $\{x_j^{(i)}\}_{j=1}^{N}$, the values of a gaussian centered on that point and with fixed variance σ are calculated on the grid, and then summed together. If $y^{(g)}$ represent the values of the estimated distribution in $\{x_j^{(i)}\}$:

$$y_j^{(g)} = \sum_{k=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_j^{(g)} - x_k^{(i)})^2}{2\sigma^2}\right) \tag{7}$$

and these are then normalized by the number of points that lie in the specified range (which may or may not be N, depending on the choice of the user). The tails of data points outside the specified range are not considered. The result of this procedure is greatly dependent on the value of σ, as it specifies the 'width' that each data point carries. This arbitrariety is analogous to that of the choice of the number of bins in an histogram. This method, however, is capable of producing less noisy results.

Both strategies were employed in the creation of the distribution and in the subsequent fitting procedure.

Finally, the calculation of r was performed for the various locality parameters, by using a dedicated function, `R_CALC`.

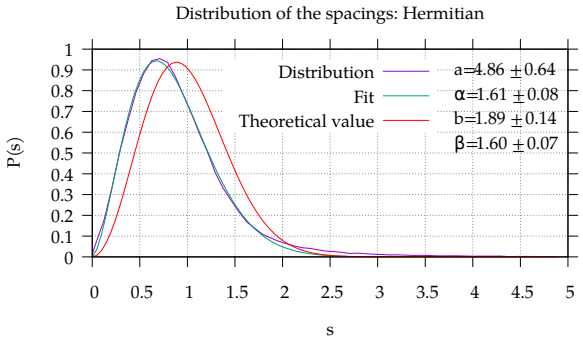
Fitting The fitting of the distributions is carried out by using Gnuplot. The distributions for both the hermitian and the diagonal matrices have the same functional form; however, for the diagonal case, numerical reasons require that the value of α be exceedingly close to 0, so much in fact that convergence from the standard values (i.e., $\alpha = 1$) is not at all guaranteed. Furthermore, setting α to a small value (in the range of 1×10^{-5} and below) results in convergence being attained, but the value of α not being actually modified by the algorithm. As a consequence, and also to ease convergence, the fitting function was chosen on the basis of the type of data to fit: in particular, in the diagonal case, it was set to be

$$p(s) = a \exp(-bs^{\beta}) \tag{8}$$

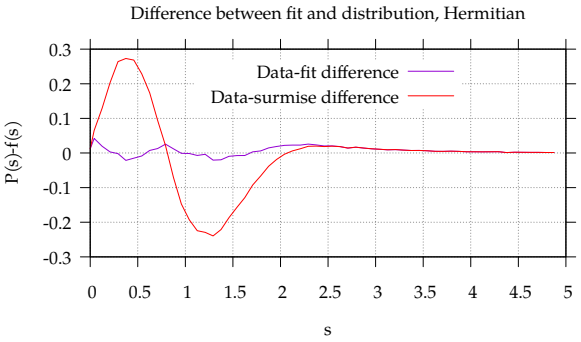
and so, implicitly, $\alpha = 0$.

Results

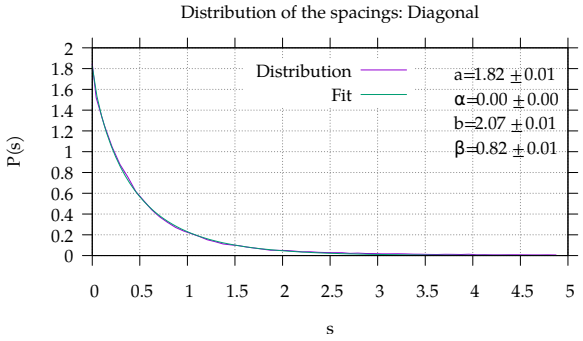
The distributions, calculated via the histogram method, are shown, together with the fitting function. The upper range was limited to 5, as the number of spacings higher than this threshold is negligible (less than 1% of the points are ignored). Also, the number of bins for the distribution is set to 80, which guarantees a smooth result but is still representative of eventual trends. The same was done by using the 'KDE' like method; the results are shown below. Where a theoretical result was available, it was shown together with the distributions. Differences between the functions are also shown.



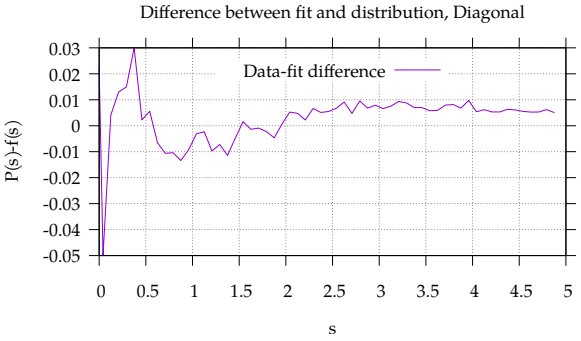
(a) Distribution of the spacings (hermitian).



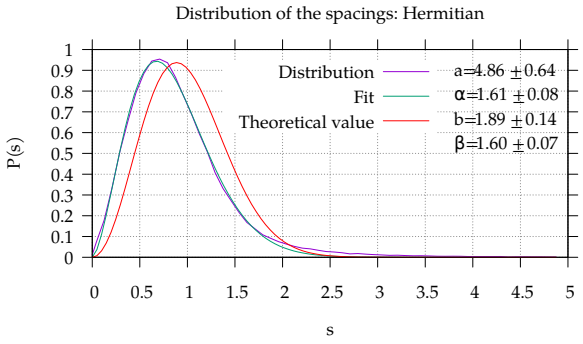
(b) Distribution-fitting function difference.



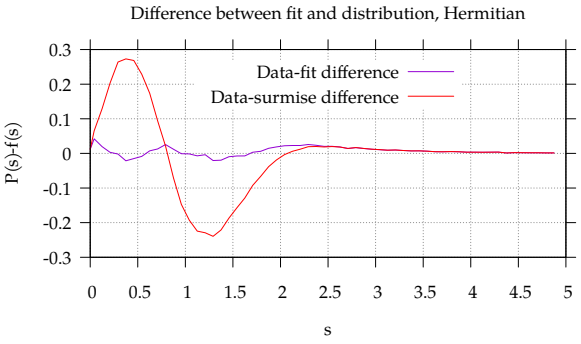
(c) Distribution of the spacings (diagonal).



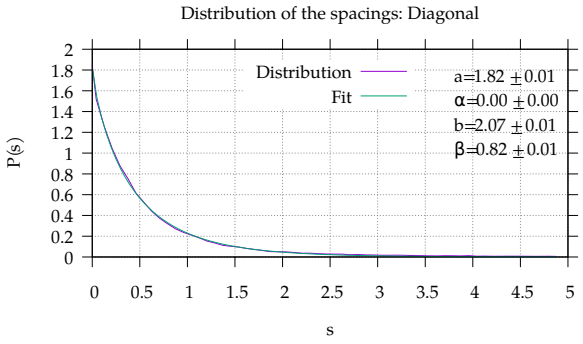
(d) Distribution-fitting function difference.



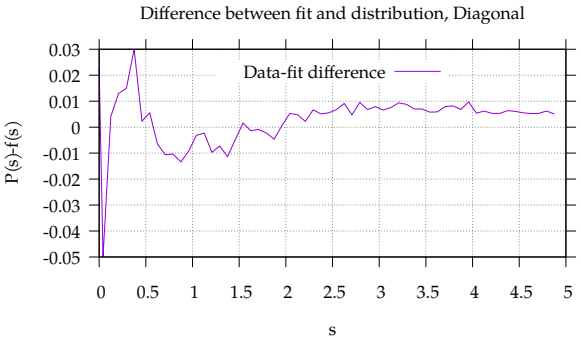
(e) Distribution of the spacings (hermitian, KDE).



(f) Distribution-fitting function difference.



(g) Distribution of the spacings (diagonal, KDE).



(h) Distribution-fitting function difference.

All the coefficients are collected in the following Table.

Parameter	Hermitian			Diagonal		
	Histogram	KDE	Theory	Histogram	KDE	Theory
a	4.9 ± 0.6	8.9 ± 0.4	$\frac{32}{\pi^2}$	1.82 ± 0.01	1.41 ± 0.01	None
α	1.61 ± 0.08	2.32 ± 0.03	2	0^*	0^*	None
b	1.9 ± 0.1	2.39 ± 0.05	$\frac{4}{\pi}$	2.07 ± 0.01	1.68 ± 0.02	None
β	1.60 ± 0.07	1.43 ± 0.01	2	0.82 ± 0.01	1.5 ± 0.02	None

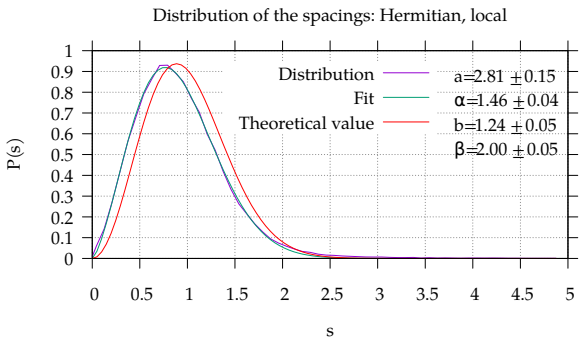
Table 1: A comparison of the coefficients obtained by fitting the distributions and the ones found in the literature (if present). The starred values are assumed, and so bear no error.

The KDE method is only really useful when dealing with the hermitian distribution; in the diagonal case, instead, it produces a peak. This is because the chosen kernel was gaussian.

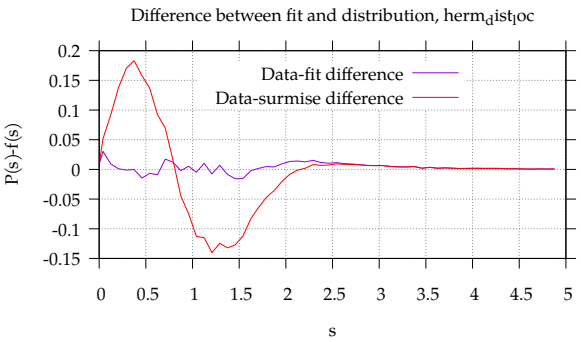
From a simple visual inspection, the distribution in the hermitian case are slightly shifted with respect to the theoretical prediction; while this may be caused by statistical fluctuations, the size (2000) and the number (60) of the matrices seem to be enough to rule out this hypothesis.

The diagonal distribution, while not having a theoretical result to be compared to, seem to validate the intuitions explained above: there is no peak outside the origin. This can be justified by the following reasoning: all the eigenvalues are drawn from the same, normal distribution. The distribution of the differences of two normal variables is itself normal, with zero mean and double the mean. However, the distribution we are dealing with takes order into account, and so it must be even more peaked in the origin.

An interesting phenomenon happens when, instead of the 'globally normalized' spacings s, one considers the distribution of the 'locally normalized' spacings; each spacing is normalized by the mean of the surrounding 400 spacings (border effects are caused, since not every spacing has 400 neighbors on each side). This produced a curve which better resembles the theoretical one. The result is shown below.



(i) Distribution of the spacings (hermitian).



(j) Distribution-fitting function difference.

This suggests that, by changing the localized average, one could better approach the theoretical result.

Finally, the values of $\langle r \rangle$ are reported; the choice of the locality level does not seem to influence these values. These values are those for the last of the 60 matrices, and are truncated at the third figure.

Locality level	**Hermitian**	**Diagonal**
N	0.836	0.652
N/5	0.839	0.596
N/10	0.838	0.658
N/50	0.835	0.631
N/100	0.835	0.647
1	0.837	0.639

Table 2: The values of $\langle r \rangle$ for the various cases.

Comments and self evaluation

I found this assignment to be very interesting; approaching an unknown topic such as Random Matrix Theory by means of direct computation was also very rewarding, if slightly frustrating at times. Among the possible improvements, a more in depth statistical analysis of the quantities of interest could be carried out.

This assignment proved to be rather challenging for me. I initially tried to make my scripts as general as possible, but this ended up in a crowded work environment, with many folders and a general lack of order. Furthermore, I had some difficulties in understanding the assignment, and this only increased my confusion. In the end, I had to start over from the beginning, trying to keep everything as simple as possible. The end result, however, was satisfying.