Random Matrix Eigenvalues Spacings

Davide Maniscalco

November 12, 2019

Abstract

This paper analyzes the distribution of the spacings of the eigenvalues of a random Hermitian matrix and of a real diagonal matrix.

COMPILATION INSTRUCTIONS:

- The compilation instruction for the code (that handles properly lapack, output filename, external modules) is the following: `gfortran Ex5-Maniscalco-CODE.f90 module_hermit.o -llapack -o Ex5-Maniscalco.out`
- The compilation instruction for the python script is: `python3 Ex5-Maniscalco-SCRIPT.py`
- Gnuplot can be run automatically through the python script. All the graphs are performed in the `FITS` folder, where the gnuplot code is placed.

1 Theory

Let's consider a random Hermitian matrix of size N and let's call $\lambda_{i=1,\dots,N}$ its eigenvalues. We define the normalized spacings between eigenvalues as

$$s_i = \frac{\Delta\lambda_i}{\bar{\Delta\lambda}} \qquad \text{with} \qquad \Delta\lambda_i = \lambda_{i+1} - \lambda_i \, .$$

Our purpose is to study the distribution of the normalized spacings for the eigenvalues of a random Hermitian matrix and of a real diagonal matrix, and try to fit them using the distribution function

$$P(s) = as^{\alpha} \exp(-bs^{\beta}) \, .$$

The random matrices are generated with fortran picking the random numbers from a uniform distribution $[0, 1]$, and diagonalized using a lapack subroutine.

2 Code development

The fortran code `Ex5-Maniscalco-CODE.f90` fulfills the purpose of generating tre matrices and making the histograms. The matrices dimensions are taken as input from the file `inputdim.dat` and then the random numbers are picked with the command `CALL RANDOM_NUMBER`. Once initialized, the matrices are diagonalized calling the `ZHEEV` subroutine provided by lapack, and then a subroutine calculates and store in two vectors the spacings. At this point, the bins and the normalized frequencies for the histogram are calculated:

```
threshold_index = int(size(spaced)*98/100+1)
threshold = spaced(threshold_index)
step = 0.01
```

```
sps = pack(spaced,spaced < threshold)
number_bins = int(threshold/step)+1

print*, threshold

ALLOCATE(hist(number_bins))
ALLOCATE(half_points(number_bins))

hist = 0d0

do ii=1,number_bins
  do jj=1,size(sps)
     if(step*(ii-1) <= sps(jj) .and. sps(jj)<step*ii)  then
     hist(ii) = hist(ii) + 1
  end if
 end do
half_points(ii) = step*(ii-1) + step/2
end do

ALLOCATE(to_print(number_bins,2))

hist = hist/sum(hist)
hist = hist/step

to_print(:,1) = half_points
to_print(:,2) = hist
```

In order to leave out the less significant data, the last 2% (4% for the real diagonal matrix case) of them are left. Then the bin width is selected (variable `step`) and two loops increment the right places of the `hist` vector. These elements are finally divided for the total number of elements and for the bin width in order to obtain a proper normalization. The bin-centred column and the one with the normalized values are stored in a matrix to be printed. The printing occurs in a file whose name recalls the input matrix dimension; the file is placed into the `FITS` folder, where the `gnuplot` script is.
Let's move to the python script. Its purpose is just the automation: it can create the file containing the matrix dimensions, run the program, run the script for the graphs.

3 Results

A matrix dimension of 3000 was retained enough in order to have a sufficient statistic. The fits made with the following start parameters:

	Hermitian	Diagonal
a	3	3
b	2	2
α	1	1
β	1	1

Table 1: Starting fit parameters

and produced the results shown in Tab.2, Figg.1,2.

In spite the figures may look meaningful, the fit has completely failed in the parameter estimation; furthermore, with different initial parameters the fit leads to a couple of order of magnitude different parameters for a and b. The estimation can be refused without any other analysis.

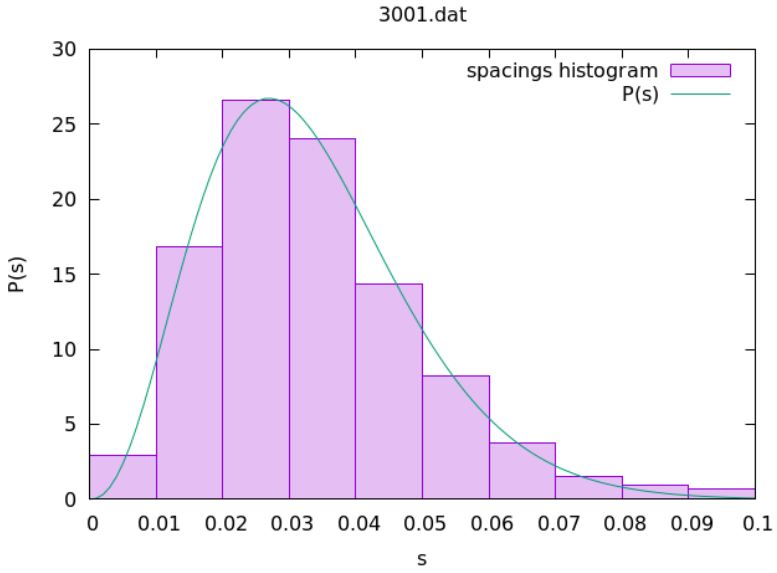

Figure 1: Histogram and fit for spacings of eigenvalues from Hermitian matrix

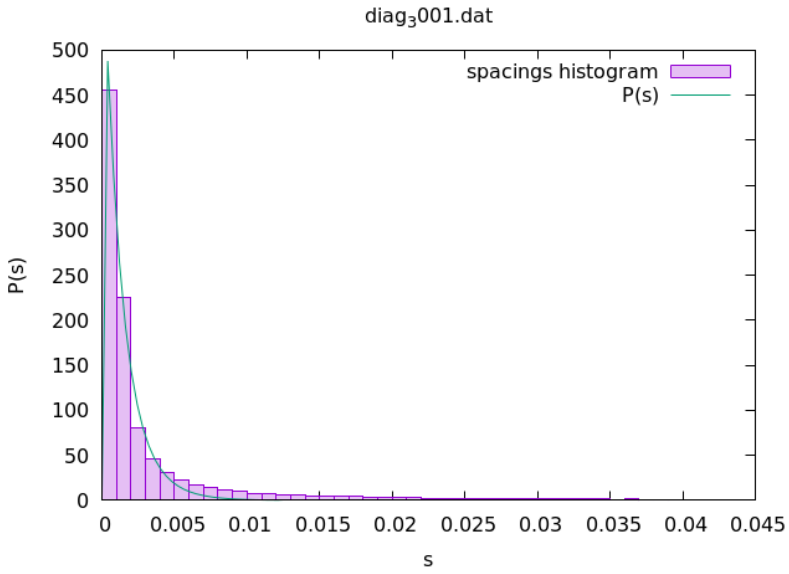

Figure 2: Histogram and fit for spacings of eigenvalues from real diagonal matrix

	Hermitian	Error(%)	Diagonal	Error(%)
a	613715	192	4361	902
b	276	16	275	232
α	2.325	47	0.202	459
β	1.419	15	0.781	76

Table 2: Estimated parameters

4 Self evaluation

The mandatory tasks have been achieved; it has been learned how to make a histogram with fortran and many skills concerning automation (like handling different folders, different output names). Unfortunately the results are poor and not satisfactory. The big waste of time due to attacking the exercise in the wrong way during the first days made impossible doing better.