Eigen Problems and Random Matrix Theory

1 Abstract

In this exercise i wrote a program able to initialize an hermitian or real diagonal square matrices and it computes their eigenvalues in crescent order.

Related to the same matrix, using eigenvalues array,it computes the difference between two adjacent eigenvalues $\lambda_{i+1} - \lambda_i$ and normalize these differences using $\overline{\Delta\lambda}$ that it computes on all matrix's eigenvalue list (*global average*) or using part of it (*local average*).

Collecting all of these ratios by different matrices samples and by different $\overline{\Delta\lambda}$,the code writes on different files two columns that will be used by gnuplot in order to plot different P(s) distributions.

Finally,it computes an another ratio , $\frac{min(\lambda_{i+1},\lambda_i)}{max(\lambda_{i+1},\lambda_i)}$,for different cases.

2 Theory

————Hermitian matrix————

By definition , an hermitian matrix (or self-adjoint matrix) is a complex square matrix that is equal to its own conjugate transpose.

The element in the i-th row and j-th column,$a_{i,j}$ is equal to the complex conjugate of the element in the j-th row and i-th column $\overline{a_{j,i}}$:

$$A \text{ Hermitian} \iff a_{ij} = \overline{a}_{ji}$$

for all indices i and j and ,in particular, each symmetric real matrix is hermitian.

————————eigenvalues and eigenvectors————————

Be T a linear transformation from a vector space V over a field F into itself and $\vec{v}$ is a non-zero vector in V.

Then $\vec{v}$ is an eigenvector of T if for T$\vec{v}$ exists λ scalar value such that : $\mathbf{T}\vec{v} = \lambda\vec{v}$ **.**

3 Code development

My code is divided in three program units : two modules and one main program.

- MODULE ***EX1*** :

  here i define ***punto1*** and ***punto2*** functions.

  Using ZHEEV function the first one computes eigenvalues in crescent order and using *info* parameter it checks if diagonalization procedure works properly and returns an eigenvalues list W in crescent order.

  The second one takes W list and an integer number called "suddivisioni" as inputs in order to compute ratios $\frac{\lambda_{i+1}-\lambda_i}{\overline{\Delta\lambda}}$ for different "levels",that is, $\overline{\Delta\lambda}$ is computed using all elements in W list or a part of it.

  So it returns an S^1 matrix with ratios (before defined) in first column and in second one a specific "level" ($\frac{N}{150}, \frac{N}{100}$ ...) where N is matrix size.

- MODULE ***EX2*** :

  here i define ***punto3*** ,***punto4*** and ***punto5*** functions.
  The first one takes two inputs : a tridimensional matrix S that contains ,in order,matrices samples number and S^1 's elements, and a character called "input2" that specify which $\overline{\Delta\lambda}$ will be computed : *local or global*.

During function execution two lists are created : the first one with interval's centroid and the second one with *normalized* number of ratio fallen into that interval.

A some intermediate checks are executed : for example if some data is lost during binning procedure or how many entries are not zero.

The last is useful in order to have an idea of histogram that will be done afterwards.
Finally one txt file is written depending by which character is passed as input.
Returns a bool variable to check if everything is gone well.

The second one, essentially, does the same stuffs as previous one,but i preferred to create an another function to better evaluate bin size in order to create as better as possible histogram.

Indeed previous function is reserved for hermitian matrices and this one for real diagonal ones.

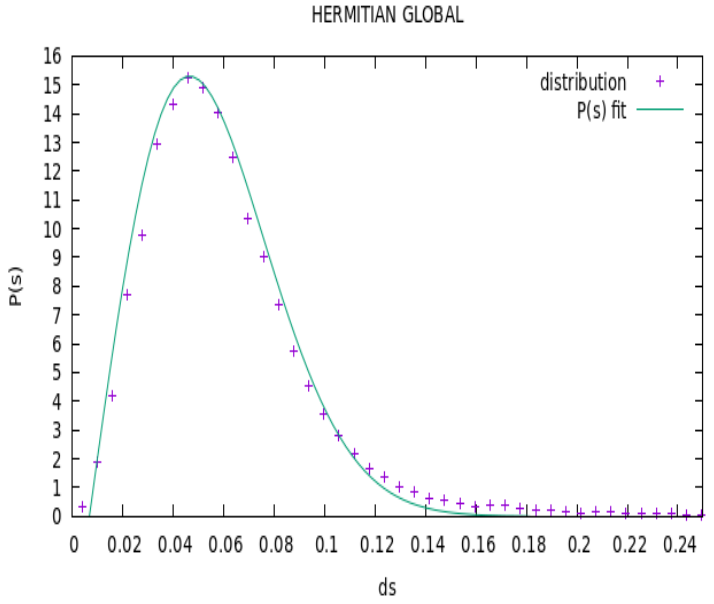
The third one takes the same inputs as previous ones but,this time,returns a double precision variable called "R average" that is the average of all ratios $\frac{min(\lambda_{i+1},\lambda_i)}{max(\lambda_{i+1},\lambda_i)}$ computed for all matrices sample.

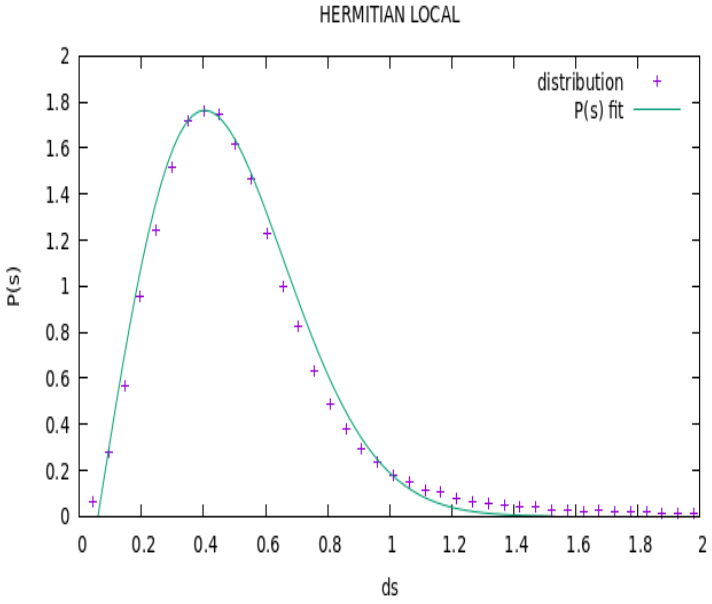
- PROGRAM ***MAIN*** :
  here the code takes as inputs by the user two character variables called "input1" and "input2" and decides,using their values ,which matrices will be initialized (hermitian or diagonal ones) and decides how recall functions defined in modules before.

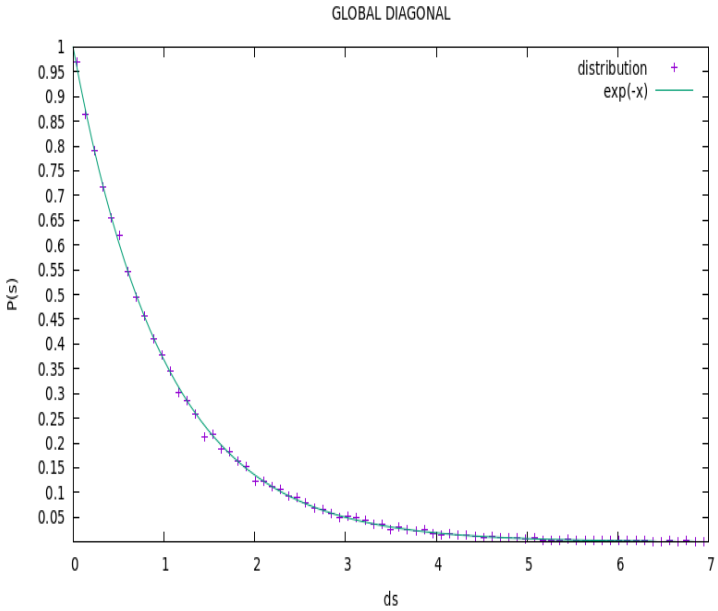
  Finally it checks if bool variable that returns from ***punto 3*** function or ***punto 4*** function is TRUE and in this case program ends successfully.

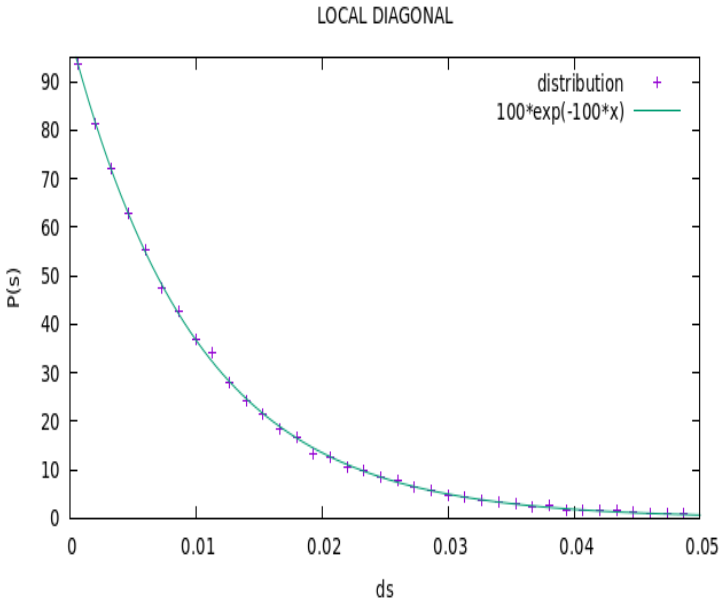
4 Results

In the following i'll show my results achieved both using fortran code and GNUPLOT software.









In order to plot a possible distribution functions P(s) i tried to use the expression $as^{\alpha}exp(-bs^{\beta})$ checking each time if it is correctly normalized to 1.
So i report best parameter value used to reach this aim.

	a	α	b	β
HERMITIAN GLOBAL	$\frac{2*1000}{\pi}$	1	$\frac{1000}{\pi}$	2
HERMITIAN LOCAL	$\frac{2*13.3}{\pi}$	1	$\frac{13.3}{\pi}$	2
DIAGONAL GLOBAL	1	0	1	1
DIAGONAL LOCAL	100	0	100	1

Local graphs shown above, are plotted using a $\overline{\Delta\lambda}$ computed as an average done over an interval width equal to $\frac{N}{2}$ centered in each s_i

Finally i collect the averages of ratio r computed , in different cases, averaging between 150 matrices each one (1000,1000) .

	$< r >$
HERMITIAN GLOBAL	0.5964880835
HERMITIAN LOCAL	0.596488835
DIAGONAL GLOBAL	0.3907737337
DIAGONAL LOCAL	0.38513915139

5 Self-evaluation

During my tests i tried only for matrices 1000*1000 because above these dimensions my computer's kernel died .

On the other hand i tested my code so many times and each time i used from 100 to 150 matrices samples an so i can consider graphs and $< r >$ values robust enough: possible variations fall into 10^{-2} respect to value found.