Assignment 3

Python scripting, random matrices and eigenproblem

October 29th 2024

Exercise 1: python **scripting**.

Scaling of the matrix-matrix multiplication. Consider the code developed in the Exercise 3 from Assignment 1 (matrix-matrix multiplication):

(a) Write a python script that changes N between two values N_{min} and N_{max} , and launches the program.

(b) Store the results of the execution time in different files depending on the multiplication method used.

(c) Fit the scaling of the execution time for different methods as a function of the input size. Consider the largest possible difference between N_{min} and N_{max} .

(d) Plot results for different multiplication methods.

WHY

Scientific simulations requires creation/storage/process of large amount of data;

Check-convergence;

Exploring range of parameters

Time dependent properties, study local/non local properties;

All these tasks require *many repetition of 'almost equal simulations' and production of many data-files*

HOW

Smart data structure

Scripting for pre- and post-processing: *automatizing repetitive work and avoid human errors!!*

SCRIPTING LANGUAGES

Bash, Python, …

Exercise 1: python **scripting**.

Scaling of the matrix-matrix multiplication. Consider the code developed in the Exercise 3 from Assignment 1 (matrix-matrix multiplication):

(a) Write a python script that changes N between two values N$_{min}$ and N$_{max}$, and launches the program.

(b) Store the results of the execution time in different files depending on the multiplication method used.

(c) Fit the scaling of the execution time for different methods as a function of the input size. Consider the largest possible difference between N $_{min}$ and N $_{max}$.

(d) Plot results for different multiplication methods.

Define differen sizes in [N$_{min}$,N$_{max}$]

Define different multiplication methods (ex. algTypes = [1,2,3])

Define optimization flags (ex. algTypes = [1,2,3])

(If you have already a compiled executable which takes sizes as input, you just loop across this and save data accordingly)

Exercise 2 : compute the spectrum of RANDOM Hermitian matrix of size N

(a) Diagonalize A and store the N eigenvalues λ_i in ascending order.

(b) Compute the normalized spacing between eigenvalues

$$\Lambda_i = \lambda_{i+1} - \lambda_i \text{ (spacing)}$$

$$s_i = \frac{\Lambda_i}{\bar{\Lambda}} \text{ (normalized spacing)}$$

HERMITIAN MATRICES

A is hermitian $\iff a_{i,j} = a^*_{j,i}$

if $a_{i,j} \in \mathscr{R}, A$ is symmetric

- use symmetries: store only the lower (upper triangle)

$$n^2 \rightarrow n(n+1)/2$$

- Generate a random complex vector of a given size
- Compute eigenvalues and spacings (*hint: discard the first eigenvalue*)
- Compute normalized spacings

WHY RANDOM MATRICES?

- Wigner: dealing with the statistics eigenvalues and eigenvectors of complex many body systems

  $H \rightarrow$ ensamble of random $\tilde{H}_i$

  (Application: description of spectral properties of atomic nuclei, …)

- Bohigas conjecture: common features in the spectra of time-reversal invariant Hamiltonian, whose classical analogue are chaotic systems

Exercise 3 : starting from the spectrum of a RANDOM Hermitian, study the distribution of the normalized spacings

(a) Random (complex) Hermitian matrix A

(b) Diagonal matrix with real random entries

Average is taken over multiple realization of random matrices

- Getting the normalized spacings $s_i = \frac{\Lambda_i}{\bar{\Lambda}}$ (nomalized spacing);
- Range of the normalized spacings $\Delta s = \max(s_i) - \min(s_i)$
- Defining binning (N_{bin}) and count how many eigenvalues fall in each bin
- We want a probability distribution not just a histogram

$$\int P(s)ds = 1 \rightarrow \sum_{m=1}^{N_{bin}} P_m(s)ds = 1 \text{ with } P_m(s) = \frac{count_m}{N_{tot}\Delta s}$$

- Fit the distribution with the generic fit
- Two different distributions for the real diagonal vs Hermitian complex