# Quantum Information and Computing 2020/21
# Week 5 report

Rocco Ardino
(Dated: Monday 9th November, 2020)

In this work we deal with eigenvalue problems using Lapack library functions. In particular, we focus on large enough random hermitian matrices and we analyze the normalized spacings between eigenvalues. Then, after a proper handling, we compare the extracted data with a theoretical distribution and find out if the expectation is met.

## 1 Theory

Given a random hermitian matrix $A$ of dimensions $n \times n$, it follows from the spectral theorem that $A$ can be diagonalized by a unitary matrix, and that the resulting diagonal matrix has only real entries. This implies that all eigenvalues of a $A$ are real, and that $A$ has $n$ linearly independent eigenvectors.

Denoting with $\lambda_i$ the eigenvalues of $A$, with $i = 1, \dots, n$ and $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, the normalized spacings between them can be computed as:

$$s_i = \frac{\lambda_{i+1} - \lambda_i}{\langle \Delta\lambda \rangle} \quad , \tag{1}$$

where $\langle \Delta\lambda \rangle$ is the average of $\Delta\lambda_i = \lambda_{i+1} - \lambda_i$. In particular, the $s_i$ can be either computed:

- globally, by considering as normalization the average of all spacings;
- locally, by considering as normalization for the $i^{\text{th}}$ spacing the averange of only the $s_j$ such that $j = i - \ell, \dots, i + \ell$, with $\ell$ a parameter called level.

We expect that the distribution of the normalized spacings will be the so-called Wigner surmise, which reads:

$$P(s) = \frac{32}{\pi^2} s^2 e^{-\frac{4}{\pi}s^2} \quad . \tag{2}$$

A good strategy is to sample a set of normalized spacings from multiple matrices like $A$, bin the sampled data and normalize them, then fit with the following function:

$$P(s) = a s^{\alpha} e^{-b s^{\beta}} \quad . \tag{3}$$

Lastly, an interesting quantity can be computed from the normalized spacings. We start from the following quatities

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

From this set, computed for $i = 1, \dots, n-2$, we can get the average $\langle r \rangle$.

## 2 Code Development

For this work, the new module **hmat** is implemented, containing several functions and subroutines for multiple purposes:

- **random_normal**, for generating random normal distributed numbers;
- **hmat_init(dim)**, for creating a generic hermitian matrix of dimensions $n \times n$ with random normal entries;
- **hmat_diag_init(dim)**, for creating a diagonal hermitian matrix of dimensions $n \times n$ with random normal entries;
- **hmat_eigs(a)**, which returns the eigenvalues of the hermitian matrix `a`;
- **hmat_s(eigs)**, which returns the globally normalized spacings $s_i$, given the eigenvalues `eigs`;
- **hmat_s_loc(eigs, level)**, which works like `hmat_s`, but it reurns the locally normalized spacings, computed with the `level` parameter;
- **hmat_r_avg(eigs, filename, unit)**, for calculating the quantity $\langle r \rangle$ and storing it in a file;
- **hist2pdf(data, min, max, nbins, filename, unit)**, for binning the given `data` in a certain number of bins in an interval $[x_{\min}, x_{\max}]$, and for normalizing to a PDF the result, storing the normalized bin centers and heights in a file.

In particular, we explain the core functionalities in the following subsections.

## 2.1 Eigenvalues and spacings computation

For the calculations of the eigenvalues of a complex hermitian matrix, the Lapack function **zheev** is employed. Therefore, the function **hmat_eigs** is built as a wrapper of the Lapack function, as it is showed in Listing 1.

```
function hmat_eigs(a) result(eigs)
    complex(8), dimension(:,:) :: a

    real(8), dimension(lbound(a,dim=1):ubound(a,dim=1)) :: eigs
    complex(8), dimension(:), allocatable :: work(:)
    real(8),    dimension(:), allocatable :: rwork(:)
    integer(4) :: N
    integer(4) :: lwork, info

    N     = size(a, 1)
    lwork = max(1,2*N-1)

    allocate(work(lwork))
    allocate(rwork(max(1, 3*N-2)))
    call zheev('N', 'U', N, a, N, eigs, work, lwork, rwork, info)
    deallocate(work)
    deallocate(rwork)
end function hmat_eigs
```

Listing 1. Implementation of the wrapper of `zheev`.

For the calculation of the spacings, `hmat_eigs` is called in both global and local implementations to get firstly the eigenvalues. Then, some other algebraic operations are performed in order to get the spacings, as it can be seen in Listing 2 and 3 respectively for the global and local version.

```
function hmat_s(eigs) result(s)
    real(8), dimension(:)  :: eigs

    real(8), dimension(lbound(eigs,dim=1):(ubound(eigs,dim=1)-1)) :: s
    real(8) :: s_m
    integer(4) :: ii

    do ii=1,size(eigs,1)-1
        s(ii) = eigs(ii+1) - eigs(ii)
    end do

    s_m = (eigs(size(eigs,1)) - eigs(1)) / size(s,1)

    do ii=1,size(s,1)
        s(ii) = s(ii) / s_m
    end do
end function hmat_s
```

Listing 2. Function for the calculation of the globally normalized spacings.

```
function hmat_s_loc(eigs, level) result(s)
    real(8), dimension(:)  :: eigs

    integer(4) :: level
    real(8), dimension(lbound(eigs,dim=1):(ubound(eigs,dim=1)-1)) :: s
    integer(4) :: ii, ll, uu


    do ii=1,size(s,1)
        ll = MAX(1, ii-level)
        uu = MIN(size(s,1), ii+level)
        s(ii) = (eigs(ii+1) - eigs(ii)) / ((eigs(uu) - eigs(ll)) / (uu-ll+1))
    end do
end function hmat_s_loc
```

Listing 3. Function for the calculation of the locally normalized spacings.

## 2.2 $P(s)$ PDF CONSTRUCTION

After having computed the normalized spacings, the subroutine **hist2pdf** takes the list of spacings and starts to fill a histogram with a given number of bins in a defined interval. Then, it normalizes every bin height with the total number of entries and the bin width. It finally creates a file and fills it with the bin centers and normalized heights. The whole simplified operation is showed in Listing 4.

```
subroutine hist2pdf(data, min, max, nbins, filename, unit)
    ! input arguments
    real(8), dimension(:) :: data
    real(8) :: min, max
    integer(4) :: nbins
    character(*) :: filename
    integer(4) :: unit

    real(8), dimension(nbins) :: binc, binh
    real(8) :: binw
    integer(4) :: ii, jj
    logical :: bin_inf, bin_sup, x_inf, x_sup

    ! set to zero bin centers and heights
```

```
    binc = 0
    binh = 0

    ! define bin centers
    binw = (max - min) / nbins
    do ii=1,size(binc)
        binc(ii) = min + (ii-0.5)*binw
    end do

    ! fill bins (~define bin heights)
    do ii=1,size(data,1)
        do jj=1,nbins
            bin_inf = data(ii) > (min + (jj-1)*binw)
            bin_sup = data(ii) < (min +  jj   *binw)
            x_inf   = data(ii) > min
            x_sup   = data(ii) < max
            if (bin_inf .AND. bin_sup .AND. x_inf .AND. x_sup) then
                binh(jj) = binh(jj) + 1
            end if
        end do
    end do

    ! normalize bin heights to get a pdf
    do ii=1,size(binh,1)
        binh(ii) = binh(ii) / (size(data,1)*binw)
    end do

    ! write results on file to use for plotting
    open(unit, file=filename, status='replace')
    do ii=1,size(binc,1)
        write(unit,'(g0ag0)') binc(ii), char(9), binh(ii)
    end do
    close(unit)
end subroutine hist2pdf
```

Listing 4. Subroutine for the creation of a PDF from histogram data.

## 2.3 Test program and automatization

Lastly, all the functions described before are employed in a test program on multiple samples, i.e. hermitian matrices, for both glabal and local case and for both random hermitian and random diagonal matrices. The program takes multiple command-line input arguments, listed in Table 1.

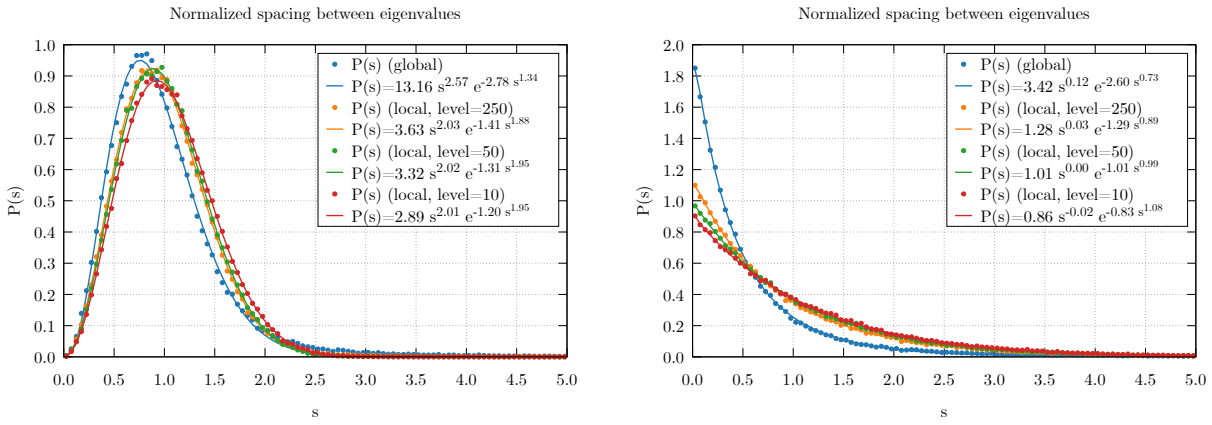
| Arg name | Arg number | Arg meaning |
|---|---|---|
| `h_type` | 1 | Type of hermitian matrix (herm=generic, diag=diagonal) |
| `samples` | 2 | Number of matrices from which spacings are sampled |
| `N` | 3 | Dimension of sampled matrices |
| `nbins` | 4 | Number of bins for the PDF |
| `xmin` | 5 | Lower bound for the histogram |
| `xmax` | 6 | Upper bound for the histogram |
| `local` | 7 | Enable local average (1) or disable (otherwise) |
| `level` | 8 | Level parameter if local average is enabled |

Table 1. Command-line arguments of the test program.

In order to compile the program and for an easy debugging, `-Wall` and `-ffree-line-length-0` flags are added and the Lapack library is linked. Then, a Python script is implemented to call the Fortran executable for all the possible cases, saving every result in a dedicated folder.

## 3 Results

To sample a sufficient amount of data for the fit, the spacings for a $2500 \times 2500$ random hermitian matrix are sampled for 50 times. Moreover, this procedure is done for both diagonal and generic hermitian matrices and for both global and local average implementations. The number of bins chosen is 100 and the range is the interval $[0, 5]$. Lastly, the quantity $\langle r \rangle$ is computed in every case. The results are showed in Figures 1a and 1b, obtained through a Gnuplot script, and in Table 2, where the fit parameters are listed. It is interesting to observe how the fit parameters converge to the expected values from theory when the local average is employed and the level parameter is sufficiently small. This is probably due to the fact that the spacing average becomes more sharp and significant at a local level.




(a) Generic hermitian matrices

(b) Diagonal hermitian matrices

Figure 1. Results for the normalized spacings between eigenvalues, obtained for both type of matrices and for both local and global implementations of the average spacing. The sampled distributions are showed with points and the corresponding fits with lines of the same color.

| H. matrix type | Level | $a$ | $\alpha$ | $b$ | $\beta$ | $\langle r \rangle$ |
|---|---|---|---|---|---|---|
| generic | $***$ | $13 \pm 3$ | $2.6 \pm 0.1$ | $2.8 \pm 0.2$ | $1.34 \pm 0.05$ | $0.5967 \pm 0.0002$ |
| generic | 250 | $3.6 \pm 0.2$ | $2.03 \pm 0.03$ | $1.41 \pm 0.04$ | $1.88 \pm 0.03$ | $0.6012 \pm 0.0002$ |
| generic | 50 | $3.3 \pm 0.1$ | $2.02 \pm 0.03$ | $1.31 \pm 0.04$ | $1.95 \pm 0.03$ | $0.6025 \pm 0.0002$ |
| **generic** | **10** | $\mathbf{2.9 \pm 0.1}$ | $\mathbf{2.01 \pm 0.03}$ | $\mathbf{1.20 \pm 0.03}$ | $\mathbf{1.95 \pm 0.03}$ | $\mathbf{0.5989 \pm 0.0002}$ |
| diagonal | $***$ | $3.4 \pm 0.2$ | $0.12 \pm 0.01$ | $2.60 \pm 0.07$ | $0.73 \pm 0.02$ | $0.3859 \pm 0.0002$ |
| diagonal | 250 | $1.28 \pm 0.03$ | $0.029 \pm 0.006$ | $1.29 \pm 0.03$ | $0.89 \pm 0.01$ | $0.3919 \pm 0.0002$ |
| diagonal | 50 | $1.01 \pm 0.02$ | $0.005 \pm 0.006$ | $1.01 \pm 0.02$ | $0.99 \pm 0.01$ | $0.3833 \pm 0.0002$ |
| **diagonal** | **10** | $\mathbf{0.86 \pm 0.02}$ | $\mathbf{-0.017 \pm 0.005}$ | $\mathbf{0.83 \pm 0.02}$ | $\mathbf{1.08 \pm 0.02}$ | $\mathbf{0.3875 \pm 0.0002}$ |

Table 2. Fit results and $\langle r \rangle$ for every case analyzed. The best results are highlighted in green.

## 4 Self-evaluation

In this work we managed to work with Lapack library functions and to describe the PDF of the eigenvalue spacings for several cases. The implementation of the code needed for the purpose has led to successful results, in agreement with the theory for the local average case and for a sufficiently small value of the level parameter.