Лабораторная работа 4 по курсу «Нелинейная динамика и её приложения». Отчёт.

Владислав Соврасов
381503м4

1 Построение гистограммы распределения собственных чисел случайных матриц

Рассмотрим матрицы $A \in \mathbf{R}^{N\times N}$ такие, что $A = A^T$ и

$$a_{i,j} = \begin{cases} 2\mathcal{N}(0,1), i = j \\ \mathcal{N}(0,1), i \neq j \end{cases}$$

Известно, что такие матрицы (Gaussian Orthogonal Ensemble) имеют N действительных собственных чисел, которые распределены по закону $\rho(\lambda/\sqrt{N}) = c\sqrt{4-\lambda^2/N}$, где c — нормировочная константа.

Для построения гистограммы распределения нормированных на $\sqrt{N}$ собственных чисел были сгенерированы 100 реализаций случайной матрицы при $N = 1000$. На рис. **??** показана нормированная гистограмма и теоретическое распределение (пунктирный график), константа в котором найдена из условия совпадения максимального значения функции распределения с максимальным значением гистограммы.

Пусть теперь матрицы $B \in \mathbf{C}^{N\times N}$ такие, что $B = B^\dagger$ и $b_{i,j} = \mathcal{N}(0,1) + i\mathcal{N}(0,1)$ (Gaussian Unitary Ensemble). Их собственные числа действительные и распределены по тому же закону, что и собственные числа матриц A. На рис. **??** можно увидеть гистограмму нормированных собственных чисел, полученную, как и для матриц A, по сотне реализаций при $N = 1000$. Из рис. **??** также можно видеть, что гистограмма достаточно точно совпадает с теоретическим распределением.

2 Построение гистограммы расщепления уровней энергии случайных матриц

Пусть имеется упорядоченный набор собственных чисел матрицы GOE или GUE:

$$\lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_{N-1} \leqslant \lambda_N$$

Расщепение уровней энергии определяется следующим образом: $s_i = \lambda_{i+1} - \lambda_i, i = \overline{1, N-1}$. Для того, чтобы распределение расщеплений не зависело от размера матрицы, как и для собственных чисел, вводится нормировка: $\bar{s_i} = s_i/ < s >$, где $< s >$ — средняя величина расщепления.

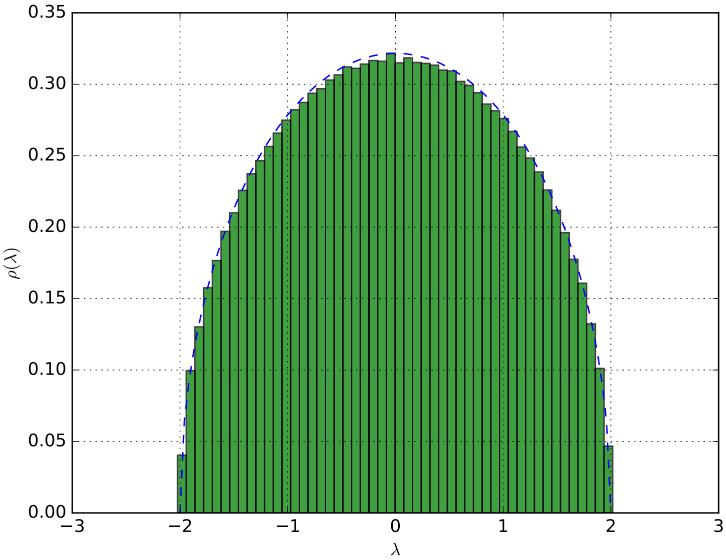

Рис. 1: Распределение нормированных собственных чисел для GOE

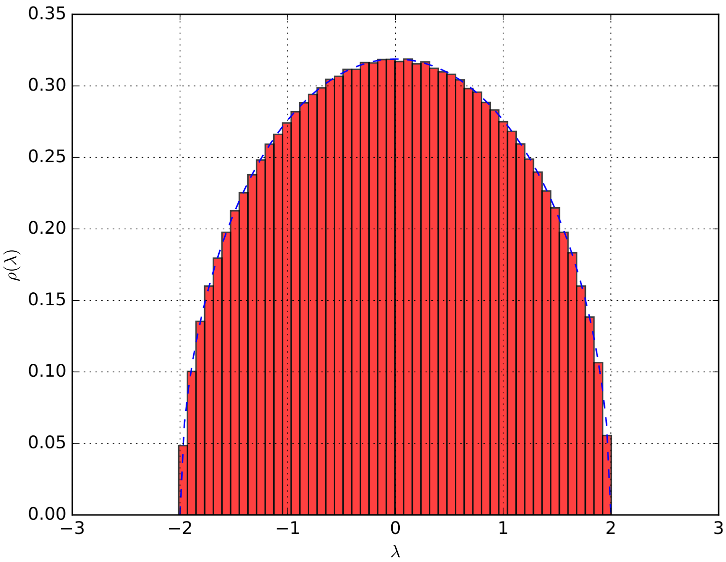

Рис. 2: Распределение нормированных собственных чисел для GUE

Теоретическое распределений расщеплений для матриц GOE имеет вид: $p_{GOE}(\bar{s}) = c_1\bar{s}\exp(-\frac{\pi}{4}\bar{s}^2)$. На рис. **??** изображена нормированная гистограмма, построенная по полученным в предудущем пункте реализациям. Для сравнения на том же графике также построена функция $p_{GOE}(\bar{s})$ с таким значением c_1, что её максимальное значение совпадает с максимумом гистограммы. Из графика видно, что гистограмма затухает с увеличением $\bar{s}$ не так быстро, как $p_{GOE}(\bar{s})$, но тем не менее общий качественный вид зависимости совпадает. В окрестности нуля значения гистограммы растут линейно.

Для матриц GUE распределений расщеплений имеет вид: $p_{GUE}(\bar{s}) = c_2\bar{s}^2\exp(-\frac{4}{\pi}\bar{s}^2)$. На рис. **??** показана нормированная гистограмма расщеплений и график функии $p_{GUE}(\bar{s})$ с вычисленной ранее описанным способом константой c_2. Как и для матриц GOE, гистограмма затухает медленнее, чем теоретическое распределение, но качественно похожа на график реального распределения. В частности, виден квадратичный порядок роста значений гистограммы в окрестности нуля.

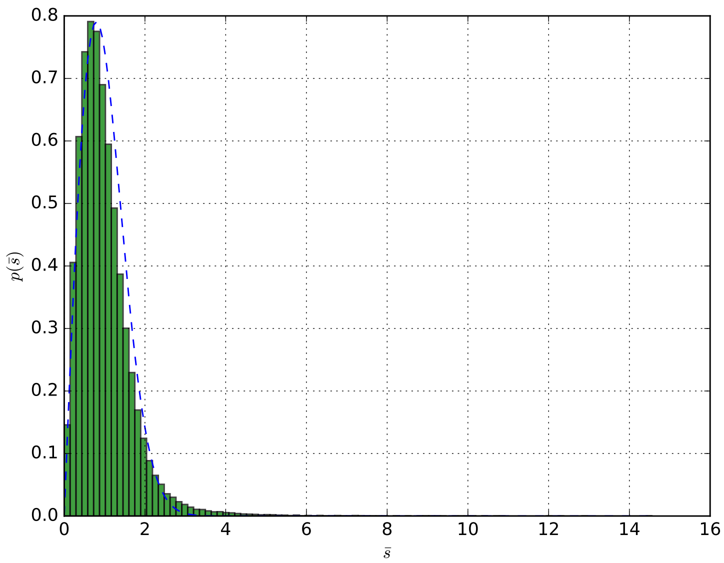

Рис. 3: Распределение нормированных расщеплений уровней для GOE

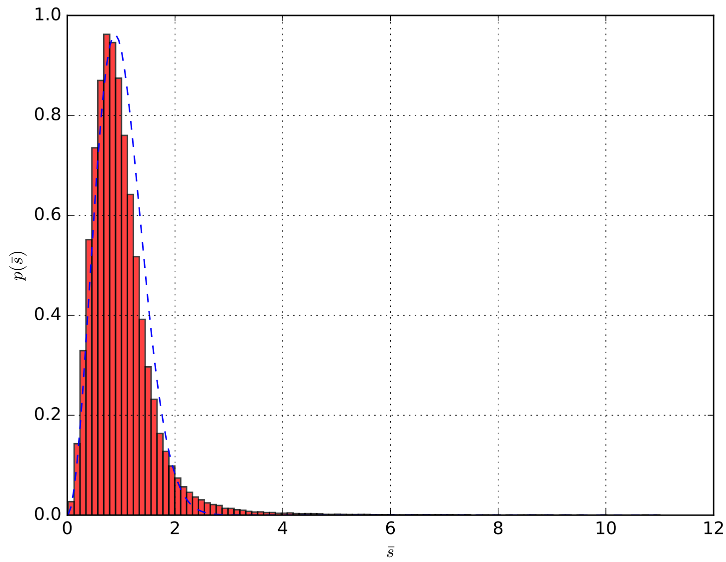

Рис. 4: Распределение нормированных расщеплений уровней для GUE

3 Исходный код

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt

def getGOE(size):
    a = np.triu(np.matrix(np.random.normal(0., 1., (size, size))))
    return a + np.transpose(a)

def getGUE(size):
    a = np.matrix(np.random.normal(0., 1., (size, size)) + \
        1j*np.random.normal(0., 1., (size, size)))
    return 0.5*(a + a.getH())

def plotRhoHist(data, xLabel, yLabel, fileName, color):
    plt.xlabel(xLabel)
    plt.ylabel(yLabel)
    n, bins, patches = plt.hist(data, 50, normed = True,
        facecolor = color, alpha = 0.75)

    c = np.amax(n) / 2.
    f = np.vectorize(lambda x: c*np.sqrt(4. - x**2))
    lGrid = np.linspace(-2., 2., 100)
```

```
    plt.plot(lGrid, f(lGrid), 'b' + '--')

    plt.grid()
    plt.savefig(fileName, format = 'pdf')
    plt.clf()

def plotLevelSpacingsHist(data, xLabel, yLabel, testDistr,
        distrArgMax, fileName, color):
    plt.xlabel(xLabel)
    plt.ylabel(yLabel)
    n, bins, _ = plt.hist(data, 100, normed = True,
        facecolor = color, alpha = 0.75)

    c = np.amax(n) / testDistr(distrArgMax)
    f = np.vectorize(lambda x: c*testDistr(x))
    sGrid = np.linspace(np.amin(bins), np.amax(bins), 200)
    plt.plot(sGrid, f(sGrid), 'b' + '--')

    plt.grid()
    plt.savefig(fileName, format = 'pdf')
    plt.clf()

def main():

    np.random.seed(10)
    size = 1000
    nImpls = 100

    goeEigens = []
    gueEigens = []

    for i in range(nImpls):
        goeEigens.append(np.linalg.eigvalsh(getGOE(size)))
        gueEigens.append(np.real(np.linalg.eigvalsh(getGUE(size))))

    goeAllEigens = np.array(goeEigens).reshape(nImpls*size)
    gueAllEigens = np.array(gueEigens).reshape(nImpls*size)

    plotRhoHist(goeAllEigens / np.sqrt(size), r'$\lambda$', r'$\rho(\lambda)$',
        '../pictures/lab4_goe_eig_hist.pdf', 'green')
    plotRhoHist(gueAllEigens / np.sqrt(size), r'$\lambda$', r'$\rho(\lambda)$',
        '../pictures/lab4_gue_eig_hist.pdf', 'red')

    goeSplits = []
    for eigenSet in goeEigens:
        goeSplits.append( \
            [eigenSet[i+1] - eigenSet[i] for i in range(len(eigenSet) - 1)])
        goeSplits[-1] = np.array(goeSplits[-1]) / np.mean(goeSplits[-1])

```

```
    gueSplits = []
    for eigenSet in gueEigens:
        gueSplits.append( \
            [eigenSet[i+1] - eigenSet[i] for i in range(len(eigenSet) - 1)])
        gueSplits[-1] = np.array(gueSplits[-1]) / np.mean(gueSplits[-1])

    goeSplits = np.array(goeSplits).reshape(nImpls*len(goeSplits[0]))
    gueSplits = np.array(gueSplits).reshape(nImpls*len(gueSplits[0]))

    plotLevelSpacingsHist(goeSplits, r'$\bar{s}$', r'$p(\bar{s})$',
        lambda x: x*np.exp(-np.pi * 0.25 * x**2), np.sqrt(2. / np.pi),
        '../pictures/lab4_goe_spacing_hist.pdf', 'green')

    plotLevelSpacingsHist(gueSplits, r'$\bar{s}$', r'$p(\bar{s})$',
        lambda x: x**2*np.exp(-4. / np.pi * x**2), np.sqrt(np.pi) / 2.,
        '../pictures/lab4_gue_spacing_hist.pdf', 'red')

if __name__ == '__main__':
    main()
```