МГТУ им. Н.Э.Баумана, кафедра «Системы автоматизированного проектирования» (РК-6)

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет имени
Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехники и комплексной автоматизации»
КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

# ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине «Вычислительная математика»

| | |
|---|---|
| Студент: | Очиров Бадма Юрьевич |
| Группа: | РК6-55Б |
| Тип задания: | лабораторная работа |
| Тема: | Спектральное и сингулярное разложения |

Студент _____ (подпись, дата) Очиров Б. Ю. (Фамилия, И.О.)

Преподаватель _____ (подпись, дата) Соколов А. П. (Фамилия, И.О.)

Москва, 2021

# Содержание

# Спектральное и сингулярное разложения

## 1 Задание

Спектральное разложение (разложение на собственные числа и вектора) и сингулярное разложение, то есть обобщение первого на прямоугольные матрицы, играют настолько важную роль в прикладной линейной алгебре, что тяжело придумать область, где одновременно используются матрицы и не используются указанные разложения в том или ином контексте. В базовой части лабораторной работы мы рассмотрим метод главных компонент (англ. Principal Component Analysis, PCA), без преувеличения самый популярный метод для понижения размерности данных, основой которого является сингулярное разложение. В продвинутой части мы рассмотрим куда менее очевидное применение разложений, а именно одну из классических задач спектральной теории графов – задачу разделения графа на сильно связанные компоненты (кластеризация на графе).

*Базовая часть:*

1. Написать функцию **pca(A)**, принимающую на вход прямоугольную матрицу данных $A$ и возвращающую список главных компонент и список соответствующихстандартных отклонений.

2. Скачать набор данных Breast Cancer Wisconsin Dataset: https://archrk6.bmstu.ru/index.php/f/854843
   - Указанный датасет хранит данные 569 пациентов с опухолью, которых обследовали на предмет наличия рака молочной железы. В каждом обследовании опухоль была проклассифицирована экспертами как доброкачественная (benign, 357 пациентов) или злокачественная (malignant, 212 пациентов) на основе детального исследования снимков и анализов. Дополнительно на основе снимков был автоматически выявлен и задокументирован ряд характеристик опухолей: радиус, площадь, фрактальная размерность и так далее (всего 30 характеристик). Постановку диагноза можно автоматизировать, если удастся создать алгоритм, классифицирущий опухоли исключительно на основе этих автоматически получаемых характеристик. Указанный файл является таблицей, где отдельная строка соответствует отдельному пациенту. Первый элемент в строке обозначает ID пациента, второй элемент – диагноз (M = malignant, B = benign), и оставшиеся 30 элемент соответствуют характеристикам опухоли(их детальное описание находится в файле https://archrk6.bmstu.ru/index.php/f/854842).

3. Найти главные компоненты указанного набора данных, используя функцию **pca(A)**.

4. Вывести на экран стандартные отклонения, соответствующие номерам главных компонент.

5. Продемонстрировать, что проекций на первые две главные компоненты достаточно для того, чтобы произвести сепарацию типов опухолей (доброкачественная и

злокачественная) для подавляющего их большинства. Для этого необходимо вывести на экран проекции каждой из точек на экран, используя scatter plot.

6. Опциональное задание №1.Постройте классификатор в полученном пространстве пониженной размерности, используя любой из классических алгоритмов машинного обучения (например, логистическую регрессию или метод опорных векторов) и, при необходимости, кроссвалидацию.

*Продвинутая часть:*

1. Построить лапласианы (матрицы Кирхгофа) $L$ для трех графов:
-полный граф $G_1$, имеющий 10 узлов;
-граф $G_2$, изображённый на рисунке 1;
-граф $G_3$, матрица смежности которого хранится в файле: https://archrk6.bmstu.ru/index.php/f/854844, где лапласианом графа называется матрица $L = D - A$, где $A$ – матрица смежности и $D$ – матрица, на главной диагонали которой расположены степени вершин графа, а остальные элементы равны нулю.

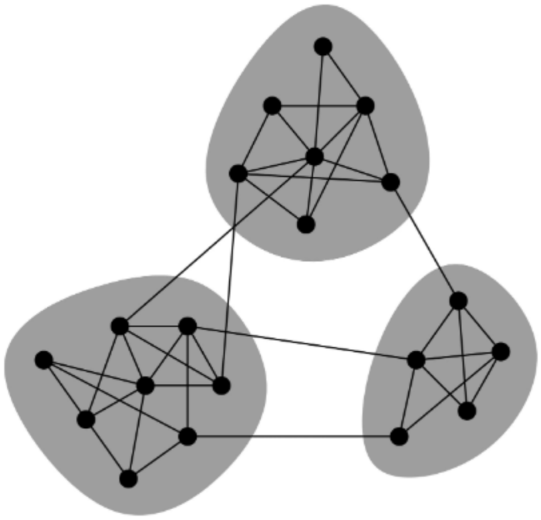

Рис. 1. Граф, содержащий три кластера

2. Доказать, что лапласиан неориентированного невзвешенного графа с $n$ вершинами является положительно полуопределенной матрицей, имеющей $n$ неотрицательных собственных чисел, одно из которых равно нулю.

3. Найти спектр каждого из указанных графов, т.е. найти собственные числа и вектора их лапласианов. Какие особенности спектра каждого из графов вы можете выделить? Какова их связь с количеством кластеров?

4. Найти количество кластеров в графе $G_3$, используя второй собственный вектор лапласиана. Для демонстрации кластеров выведите на графике исходную матрицу смежности и её отсортированную версию.

5. Опциональное задание №2. Реализуйте алгоритм DBSCAN и произведите с помощью него кластеризацию графа $G_2$.

## 2 Цель выполнения лабораторной работы

Цель выполнения лабораторной работы – изучить спектральное и сингулярное разложения, использовать их в реализации метода главных компонент, в анализе данных для сепарации типов опухолей и в задаче разделения графа на сильно связанные компоненты (кластеризация на графе).

## 3 Базовая часть

1. *Разработана функция нахождения главных компонент и соответствующих стандартных отклонений.*

2. *Набор данных Breast Cancer Wisconsin Dataset был стандартизирован и использован для нахождения главных компонент и стандартных отклонений.*

3. *Выведены стандартные отклонения, соответствующие номерам главных компонент.*

4. *Найдены проекции на первые две главные компоненты и произведена сепарация типов опухолей по графику.*

### 1. Разработка функции *pca(A)*.

**Метод главных компонент** или principal component analysis — самый популярный метод для понижения размерности данных с наименьшей потерей информации.

В рамках реализации метода главных компонент первоначально задана некоторая прямоугольная матрица данных $\mathbf{A} \in \mathbb{R}^{m \times n}$:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},$$

где $m$ - количество измерений/строк матрицы, а $n$ - количество показаний/столбцов матрицы.

Но по **теореме о главных компонентах** известно:

Главными компонентами матрицы центрированных данных $\mathbf{A}$ являются её сингулярные вектора, при этом $j$-я главная компонента соответствует $j$-му сингулярному вектору $\mathbf{q}_j$ и стандартному отклонению $\sqrt{\nu}\sigma_j$, где $\sigma_j$ является $j$-м сингулярным числом, а $\nu = \dfrac{1}{m-1}$.

Соответственно, чтобы воспользоваться теоремой и найти искомые данные, необходимо сначала произвести преобразование исходной матрицы в матрицу центрированных данных.

Матрицей центрированных данных называется такая матрица $\mathbf{T}$, в которой зафиксированы отклонения исходных данных от среднего значения в столбце:

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ t_{21} & t_{22} & \cdots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m1} & t_{m2} & \cdots & t_{mn} \end{bmatrix},$$

где $t_{ij} = a_{ij} - \bar{x}_j$, $i \in [1, m]$, $j \in [1, n]$ и $\bar{x}_j = \dfrac{\sum\limits_{i=1}^{m} x_{ij}}{m}$

Для нахождения матрицы центрированных данных была разработана функция *data_centering(A)*. Её реализация представлена в файле basis.ipynb.

После того, как была найдена матрица $\mathbf{T}$, необходимо найти сингулярные числа и вектора этой матрицы по следующим шагам:

1. *Нахождение матрицы Грама*

   Матрицу Грама найдём по формуле (1):

   $$\mathbf{K} = \mathbf{T}^T \cdot \mathbf{T}, \tag{1}$$

   где $\mathbf{K}$ - матрица Грама $n \times n$

2. *Нахождение собственных чисел и векторов матрицы Грама*

   Собственные числа и вектора матрицы Грама $\mathbf{K}$ найдём с помощью функции библиотеки *питру* - linalg.eig(K).

3. *Получение сингулярных чисел и векторов из собственных чисел и векторов*

   Сингулярные числа - это преобразованные собственные числа матрицы Грама:

   $$\sigma_i = \sqrt{\lambda_i}, \tag{2}$$

   где $\lambda_i$ - собственное число, $i \in [1, r]$, $r$ - ранг матрицы Грама.

   Так как матрица Грама $\mathbf{K}$ положительно полуопределенная, то все значения собственных чисел неотрицательны. Но сингулярные числа есть аргументы, полученные по формуле (2) и которые будут ненулевыми, а упорядочены они будут в невозрастающем порядке:

   $$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0,$$

   причём также примем нулевым сингулярное число, которое будут меньше, чем $10^{-14}$.

А сингулярные вектора есть вектора, ассоциированные с собственными числами, из которых и получены соответствующие сингулярные числа.

Получение сингулярных чисел и векторов представлено в функции *from_eig_to_sing(eig_val, eig_vec)*. Она представлена в файле basic.ipynb.

Таким образом, были получены главные компоненты - сингулярные числа матрицы центрированных данных, а стандартные отклонения были получены домножением сингулярных чисел на $\sqrt{\nu}$.

Функция *pca(A)*, реализующая все описанные выше действия в рамках первого пункта базовой части, представлена в Листинге 1.

Листинг 1. Реализация функции *pca(A)*

```
def pca(A):
    A = data_centering(A)

    gram_matrix = A.T @ A
    eig_val, eig_vec = np.linalg.eig(gram_matrix)

    sing_val, sing_vec = from_eig_to_sing(eig_val, eig_vec)

    nu = 1 / (len(A) - 1)
    for i in range(len(sing_val)):
        sing_val[i] = np.sqrt(nu) * sing_val[i]

    return sing_vec.T, sing_val
```

**2. Получение данных из датасета, стандартизация и применение функции pca(A).**

Для работы с данными необходимо было скачать файл "wdbc.data" , в котором 529 строк и 32 столбца. Первые два столбца обозначают ID пациента и его диагноз, а остальные - характеристики опухоли, которые и необходимо проанализировать. Строки хранят данные о 529 пациентах.

Для считывания данных используем функцию библиотеки *pandas - read.csv()*. Полученные данные преобразуем в двумерный массив данных. Отдельно выделим массив диагнозов и отдельно получим матрицу $\mathbf{A}$ размером $529 \times 30$.

Сначала все значения матрицы $\mathbf{A}$ требуется стандартизировать, то есть ко всем значениям применить следующую формулу:

$$a_{ij} = \frac{a_{ij} - \bar{a}_j}{\sigma_j^*}, \tag{3}$$

где

$$\bar{a}_j = \frac{\sum\limits_{i=1}^{m} a_{ij}}{m} \tag{4}$$

- выборочное среднее,

$$\sigma_j^* = \sqrt{\frac{\sum_{i=1}^{m}(a_{ij} - \bar{a}_j)^2}{m-1}} \tag{5}$$

- выборочное стандартное отклонение.

Выборочное среднее и выборочное стандартное отклонение можно найти по формулам (4) и (5), но в текущей реализации найдём их с помощью функций библиотеки *numpy* - *mean(A, axis = 0)*, *std(A, ddof = 1, axis = 0)*.

Для стандартизации по формуле (3) реализована функция *data_scaling(A)*. Она представлена в Листинге 2

Листинг 2. Реализация функции *data_scaling(A)*

```
def data_scaling(A):
    m, n = np.shape(A)

    x_bar = np.mean(A, axis = 0)
    sigma = np.std(A, ddof = 1, axis = 0)
    for i in range(m):
        for j in range(n):
            A[i][j] = (A[i][j] - x_bar[j]) / sigma[j]

    return A
```

Найдём главные компоненты и стандартные отклонения стандартизированного набора данных с помощью функции *pca(A)*. Реализация пунктов 2 и 3 базовой части представлена в Листинге 3.

Листинг 3. Применение стандартизации и функции *pca(A)*

```
if __name__ == '__main__':
    df = pd.read_csv('wdbc.data', names = [i for i in range(32)])
    data = np.array(df)

    diagnosis = data[:, 1]
    A = np.array(data[:, 2:32], dtype=np.float64)

    A = data_scaling(A)
    principal_components, std = pca(A)
```

**3. Выведение стандартных отклонений, соответствующих номерам главных компонент.**

В рамках предыдущего пункта был получен массив *std*, в котором хранятся стандартные отклонения, соответствующие главным компонентам от 1 до $r$. Выведем на графике порядковое число главного компонента и соответствующее значение стандартного отклонения. Реализация функции *plotParagraph4(std)* по построению графика представлена в Листинге 4.

Листинг 4. Реализация функции *plotParagraph4(std)*

```
def plotParagraph4(std):
    fig, ax = plt.subplots(figsize = (16, 6))

    ax.plot(1 + np.arange(len(std)), std, 'o--', color = 'purple')

    ax.set_xlabel(r'$i$', fontsize = 16)
    ax.set_ylabel(r'$\sqrt{\nu} \sigma_i$', fontsize = 16)
    ax.grid()

    plt.savefig("std.pdf")
```

Построенный график представлен на рисунке 2.

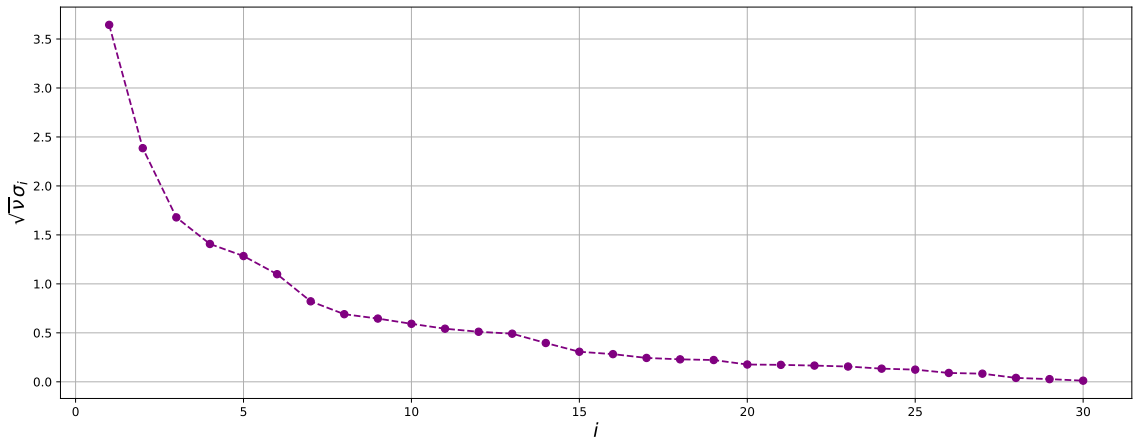


Рис. 2. Стандартные отклонения, соответствующие номерам главных компонент

**4. Нахождение проекций на первые две главные компоненты и сепарация типов опухолей по графику.**

Спроецируем матрицу **A** размером $529 \times 30$ на первые две главные компоненты, представленные матрицей размером $2 \times 30$ в переменной *principal_components*. Для перемножения транспонируем матрицу главных компонент и в результате получим матрицу $529 \times 2$, данные из которой мы используем для сепарации типов опухолей.

Выведем все точки полученной матрицы, используя *scatter plot*, а разделение на доброкачественные и злокачественные опухоли выполним с помощью массива *diagnosis*, хранящий последовательно все диагнозы пациентов. То есть проходимся по всем пациентам от 0 до 528, выводя для каждого проекции на главные компоненты и окрашивая их в зелёный или красный в соответствии с диагнозом.

Реализация функции построения *plotParagraph5()* представлена в Листинге 5.

Листинг 5. Реализация функции *plotParagraph5(A, principal_components, diagnosis)*

```
def plotParagraph5(A, principal_components, diagnosis):
    principal_components = principal_components[:2] #две главные компоненты

    proj = A @ principal_components.T
    fig, ax = plt.subplots(figsize=(16, 6))

    for i in range(len(diagnosis)):
        if diagnosis[i] == 'B':
            green = ax.scatter(proj[i][0], proj[i][1], color='lime')
        else:
            red = ax.scatter(proj[i][0], proj[i][1], color='red')

    ax.set_xlabel('Проекция на1 главнуюкомпоненету', fontsize=16)
    ax.set_ylabel('Проекция на2 главнуюкомпоненту', fontsize=16)
    plt.title('Обследование напредметналичияракамолочнойжелезы', fontsize=16)
    ax.legend([green, red], ['Доброкачественная', 'Злокачественная'], loc='upper left',
        fontsize=15)
    ax.grid()
    plt.savefig("cancer.pdf")
```

Выполненные построения точек представлены на рисунке 3.

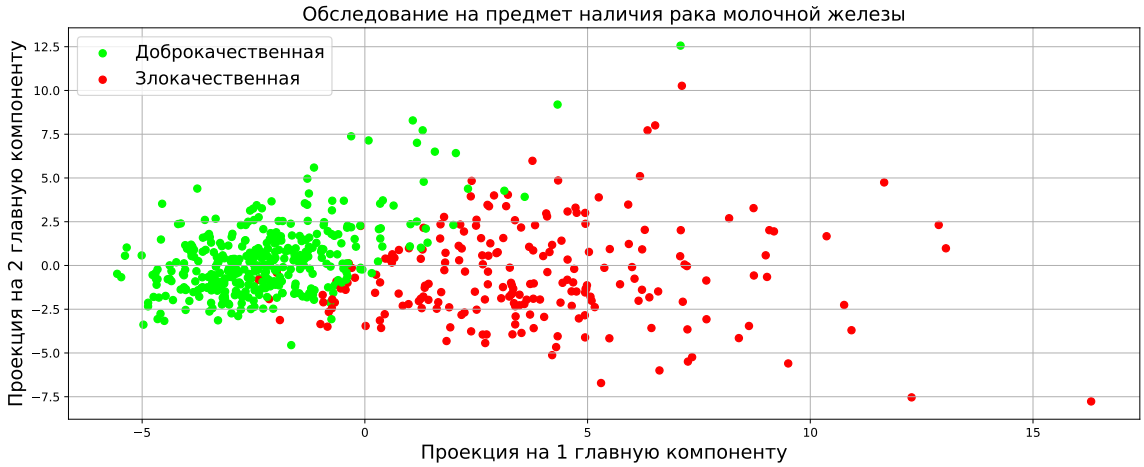


Рис. 3. Проекция матрицы **A** на две главные компоненты

По этому рисунку можно чётко увидеть, что с помощью метода главных компонент получается проанализировать исходное большое количество данных, спроецировать на две главные компоненты и определить разницу между доброкачественными и злокачественными опухолями так же, как они явно разделены на рисунке 3.

# 4 Продвинутая часть

1. *Построены лапласианы* **L** *для трёх графов.*

2. *Найдены спектры каждого графа и проанализированы их особенности.*

**1. Построение лапласиан (матриц Кирхгофа) для трёх графов.**
В рамках продвинутой части лабораторной работы нам заданы три графа:

- Полный граф $G_1$ с 10 вершинами - его матрица смежности есть единичная матрица $10 \times 10$ с нулями на главной диагонали;
- Граф $G_2$, представленный на рисунке 1 - его матрица смежности выведена вручную;
- Граф $G_3$ - его матрица смежности была дана в условии.

Соответствующие матрицы смежности зададим через файлы: *G1.txt, G2.txt* и *adjacency_matrix.txt*. Все они представлены в папке ресурсных файлов и использованы в дальнейшем.

Лапласианы (матрицы Кирхгофа) **L** находятся по следующей формуле:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{6}$$

где **D** - матрица, у которой на главной диагонали степени вершин, а на остальных позициях нули. **A** - известная матрица смежности.

Матрица смежности отражает факт того, является ли пара вершин смежными. А степенью вершины называется число ребер графа, которым принадлежит эта вершина. Таким образом, очевидно, что с помощью матрицы смежности **A** можно получить матрицу **D**, просуммировав для каждой вершины элементы, которые смежны с ним:

$$d_{ii} = \sum_{j=1}^{n} a_{ij}, \tag{7}$$

где $d_{ii}$ - элементы главной диагонали матрицы **D**, $a_{ij}$ - элементы матрицы **A**, $i \in [1, n]$, а $n$ - размерность матрицы смежности **A** и матрицы **D**.

Реализация функции нахождения матрицы **D** представлена в Листинге 6.

Листинг 6. Реализация функции *from_adjacency_to_diag(A)*.

```
def from_adjacency_to_diag(A):
    diagonal = []
    for i in range(len(A)):
        sum = 0
        for j in range(len(A)):
            sum += A[i][j]
        diagonal.append(sum)

    return np.diag(diagonal)
```

С помощью функции *read_table()* библиотеки *pandas* считываем матрицы с файлов *G1.txt, G2.txt* и *adjacency_matrix.txt*. Преобразуем их в массивы при помощи библиотеки *numpy* - получим матрицы $\mathbf{A_1}, \mathbf{A_2}$ и $\mathbf{A_3}$. Используем их для нахождения матриц $\mathbf{D_1}, \mathbf{D_2}$ и $\mathbf{D_3}$. Произведём вычитание соответствующих матриц и получим лапласианы $\mathbf{L_1}, \mathbf{L_2}$ и $\mathbf{L_3}$.

Реализация этого алгоритма представлена в Листинге 7

Листинг 7. Построение лапласиан для трёх графов

```
if __name__ == '__main__':
    G1 = pd.read_table('G1.txt', sep='\s+', engine='python', names=[i for i in range(10)])
    G2 = pd.read_table('G2.txt', sep='\s+', engine='python', names=[i for i in range(20)])
    G3 = pd.read_table('adjacency_matrix.txt', sep='\s+', engine='python', names=[i for i
        in range(1000)])

    A1 = np.array(G1)
    A2 = np.array(G2)
    A3 = np.array(G3)

    D1 = from_adjacency_to_diag(A1)
    D2 = from_adjacency_to_diag(A2)
    D3 = from_adjacency_to_diag(A3)

    L1 = np.array(D1 - A1, dtype=np.float64)
    L2 = np.array(D2 - A2, dtype=np.float64)
    L3 = np.array(D3 - A3, dtype=np.float64)
```

**2. Нахождение спектров каждого графа и анализ их особенностей.**

Для каждого лапласиана $\mathbf{L}_1$, $\mathbf{L}_2$ и $\mathbf{L}_3$ найдём собственные числа и вектора, отсортируем собственные числа от наименьшего до наибольшего и так же отсортируем ассоциированные собственные вектора. Приведём все собственные числа и собственные вектора. Реализация всех построений находится в ресурсном файле advanced.ipynb.

Собственные числа приведём на рисунках 4, 5 и 6.

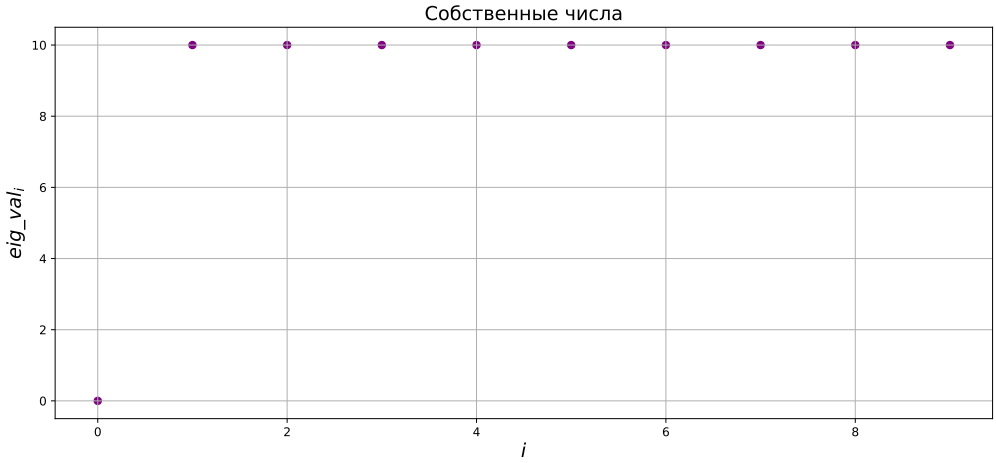


Рис. 4. Собственные числа лапласианы графа $G_1$

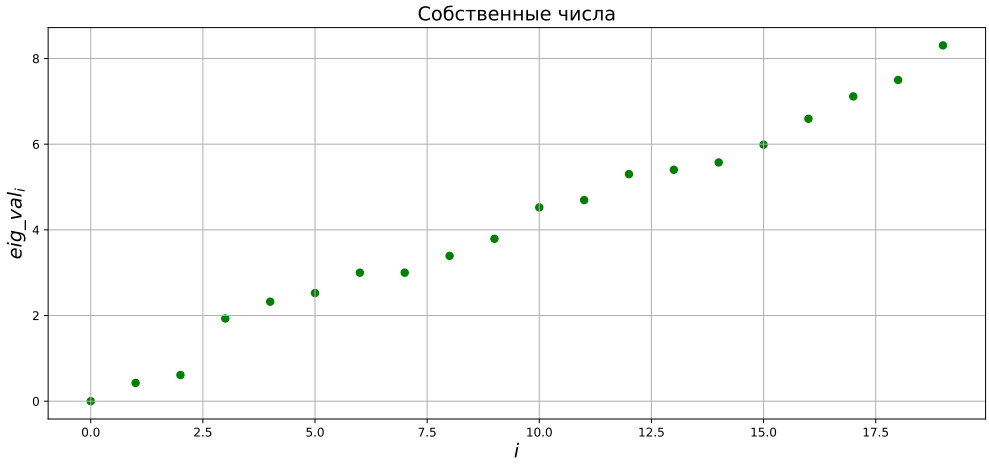


Рис. 5. Собственные числа лапласианы графа $G_2$

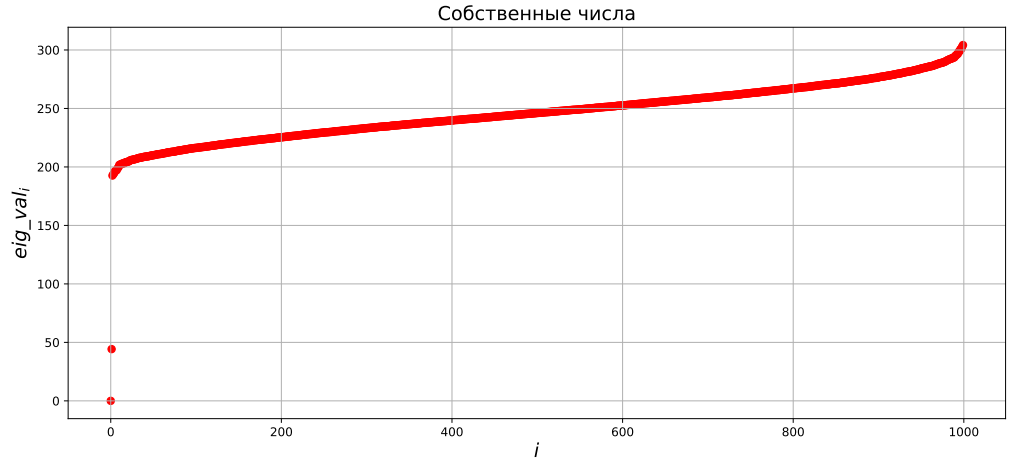


Рис. 6. Собственные числа лапласианы графа $G_3$

Рассмотрим собственные вектора для первых двух ненулевых собственных чисел каждого графа, так как рассматривать все их не имеет смысла - рисунки 7, 8 и 9:

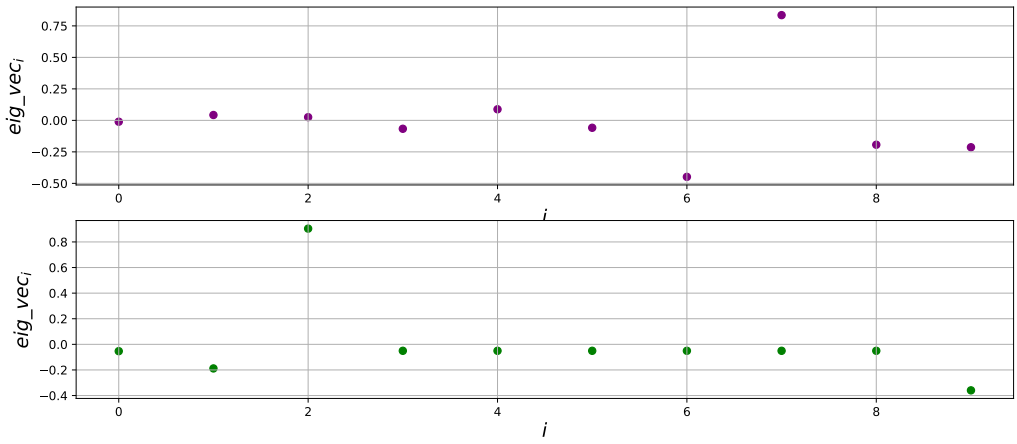


Рис. 7. Первый и второй собственные вектора лапласианы графа $G_1$

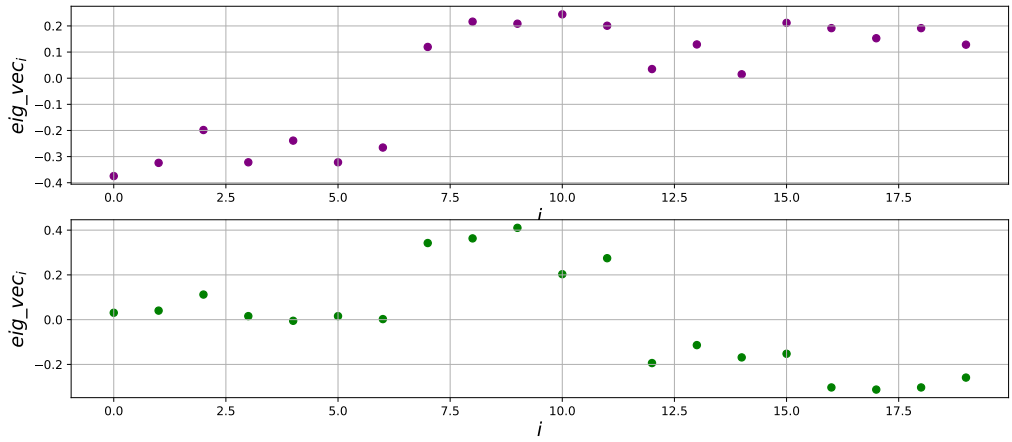


Рис. 8. Первый и второй собственные вектора лапласианы графа $G_2$

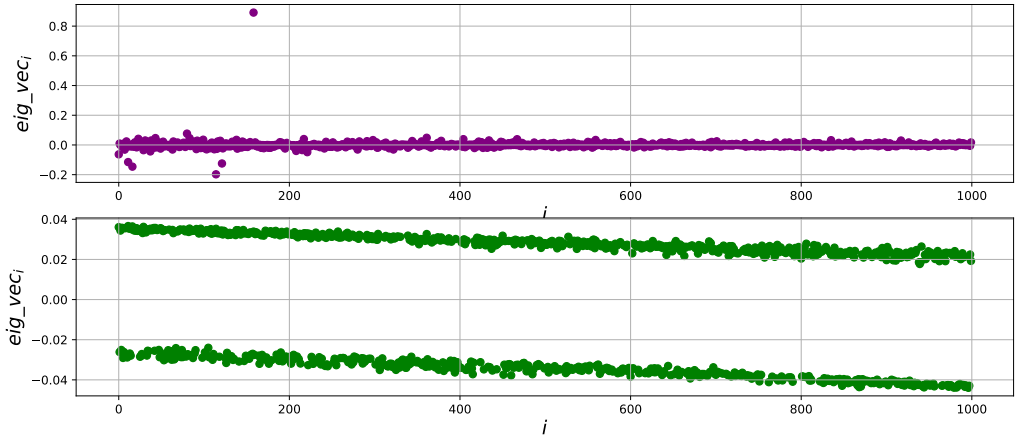


Рис. 9. Первый и второй собственные вектора лапласианы графа $G_3$

Проанализировав эти рисунки, стало ясно, что по первому графу сложно сделать какие-то выводы, но по второму и третьему понятно, что первый ненулевой собственный вектор лапласианы отвечает за спектральный промежуток, а второй за алгебраическую связность, то есть кластеризация графа в подграфы. Как можно заметить, явно выделены 3 кластера во втором графе и 2 кластера в третьем.

# 5 Заключение

1. **По завершении базовой части:**

   - *были изучены спектральное и сингулярное разложения, которые применили в методе главных компонент*
   - *на языке программирования Python реализовали метод стандартизации данных и метод главных компонент*
   - *успешно использовали метод главных компонент для анализа данных о больных раком молочной железы и сепарировали типы опухолей по графику*

2. **По завершении продвинутой части:**

   - *были рассмотрены три графа, найдены их лапласианы*
   - *на основании найденных собственных чисел и векторов лапласиан были сделаны выводы об их особенностях и наличии кластеров в графах*

### Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140.

### Выходные данные

*Очиров Б. Ю.. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2021. — 16 с. URL: https://sa2systems.ru:88 (система контроля версий кафедры РК6)*

Постановка: *ассистент кафедры РК-6, PhD А.Ю. Першин*

Решение и вёрстка: *студент группы РК6-55Б, Очиров Б. Ю.*

*2021, осенний семестр*