



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:	Горелкина Елизавета Егоровна
Группа:	РК6-52Б
Тип задания:	лабораторная работа
Тема:	Спектральное и сингулярное разложение

Студент

подпись, дата

Горелкина Е.Е.  
Фамилия, И.О.

Преподаватель

подпись, дата

Фамилия, И.О.

Москва, 2021

# Содержание

<b>Спектральное и сингулярное разложения</b>	<b>3</b>
1 Задание . . . . .	3
2 Цель выполнения лабораторной работы . . . . .	5
3 Выполненные задачи . . . . .	5
4 Разработка функции rca(A) . . . . .	5
5 Получение и подготовка исходных данных и использование в rca(A) . . . . .	8
6 Выход стандартных отклонений, соответствующих номерам главных компонент . . . . .	8
7 Проецирование стандартизованных данных на первые две главные компоненты . . . . .	9
8 Построение лапласианов для каждого графа . . . . .	10
9 Доказательство утверждения для лапласиана графа . . . . .	13
10 Поиск спектров каждого из указанных графов . . . . .	15
11 Определение количества кластеров в графе $G_3$ . . . . .	19
12 Заключение . . . . .	23

# 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).

### Задача 25 (спектральное и сингулярное разложения)

Требуется (базовая часть):

1. Написать функцию  $\text{pca}(A)$ , принимающую на вход прямоугольную матрицу данных  $A$  и возвращающую список главных компонент и список соответствующих стандартных отклонений.
2. Скачать набор данных Breast Cancer Wisconsin Dataset: <https://archrk6.bmstu.ru/index.php/f/85484391>. Указанный датасет хранит данные 569 пациентов с опухолью, которых обследовали на предмет наличия рака молочной железы. В каждом обследовании опухоль была проklassифицирована 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, классифицирующий опухоли исключительно на основе этих автоматически получаемых характеристик. Указанный файл является таблицей, где отдельная строка соответствуетциальному пациенту. Первый элемент в строке обозначает ID пациента, второй элемент – диагноз ( $M = \text{malignant}$ ,  $B = \text{benign}$ ), и оставшиеся 30 элементов соответствуют характеристикам опухоли.
3. Найти главные компоненты указанного набора данных, используя функцию  $\text{pca}(A)$ .
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

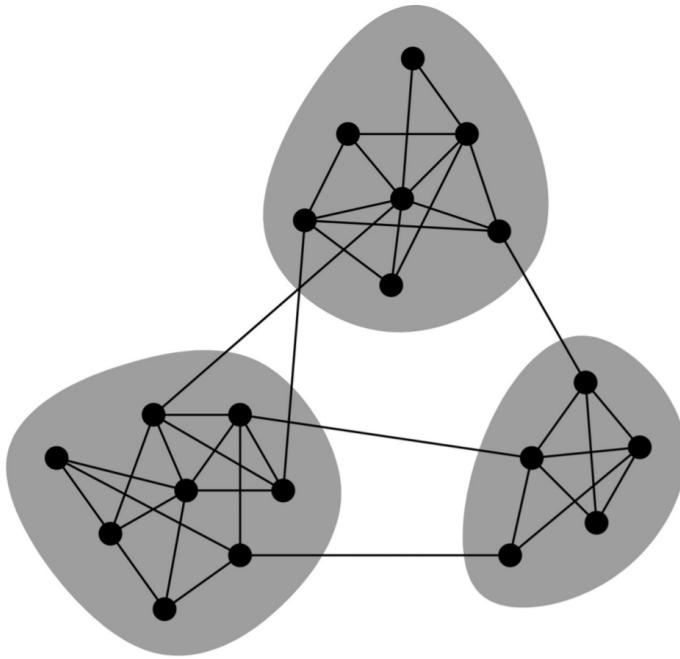


Рис. 1. Граф, содержащий три кластера

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

Требуется (продвинутая часть):

1. Построить лапласианы (матрицы Кирхгофа)  $\mathbf{L}$  для трех графов:
  - полный граф  $G_1$ , имеющий 10 узлов;
  - граф  $G_2$ , изображенный 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  $\mathbf{L} = \mathbf{D} - \mathbf{A}$ , где  $\mathbf{A}$  – матрица смежности и  $\mathbf{D}$  – матрица, на главной диагонали которой расположены степени вершин графа, а остальные элементы равны нулю
2. Доказать, что лапласиан неориентированного невзвешенного графа с  $n$  вершинами является положительно полуопределенной матрицей, имеющей  $n$  неотрицательных собственных чисел, хотя бы 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 ее отсортированную версию.

## 2 Цель выполнения лабораторной работы

Целью данной работы является изучение спектрального разложения квадратных матриц и сингулярного разложения – обобщения спектрального на случай прямоугольных матриц, а также знакомство с базовым алгоритмом – методом главных компонент, который позволяет уменьшить размерность исходных данных без потери важной информации, которую из них можно извлечь.

## 3 Выполненные задачи

1. На языке программирования Python была разработана функция `rca(A)`, принимающая на вход подготовленные данные и возвращающая главные компоненты и соответствующие им стандартные отклонения.
2. Были получены, подготовлены и использованы в функции `rca(A)` исходные данные, скачанные отсюда: <https://archrk6.bmstu.ru/index.php/f/85484391>. В результате были найдены главные компоненты и соответствующие им стандартные отклонения скачанных данных.
3. Были выведены на экран стандартные отклонения, соответствующие номерам главных компонент.
4. Было продемонстрировано, что проекций стандартизованных (подготовленных) данных на первые две главные компоненты достаточно для сепарации данных.
5. На языке программирования Python была реализована функция `laplac(G)`, вычисляющая лапласиан по матрице смежности графа. И также были вычислены лапласианы трех указанных графов.
6. Было доказано утверждение о том, что лапласиан графа с  $n$  вершинами – положительно полуопределенная матрица с  $n$  неотрицательными собственными числами, хотя бы одно из которых равно нулю.
7. Были найдены спектры каждого графа и выявлена их связь с числом кластеров.
8. Было найдено количество кластеров в графе  $G_3$ . И также была продемонстрирована исходная матрица смежности графа  $G_3$  и ее отсортированная версия.

## 4 Разработка функции `rca(A)`

Для реализации функции `rca(A)`, необходимо, прежде всего, дать определение матрице данных, опираясь на лекции [2] и [3].

**Определение 1.** Матрицей данных называется прямоугольная матрица  $\mathbf{X}$ , состоящая из  $m$  строк и  $n$  столбцов. Где столбцы содержат какие-либо показания, а строки – измерения по каждому из этих показаний.

$$\mathbf{X} = \begin{bmatrix} x_1^1 & \dots & x_1^n \\ \vdots & \dots & \vdots \\ x_m^1 & \dots & x_m^n \end{bmatrix}. \quad (1)$$

Стоит отметить, что матрица данных обычно не является квадратной ( $m > n$ ). Для нахождения главных компонент, данные необходимо сначала подготовить, а именно, произвести их стандартизацию. По каждому показанию по всем измерениям вычисляется вектор выборочных средних показаний измерений:

$$\bar{\mathbf{x}} = \frac{1}{m} \mathbf{e}^T \mathbf{X}, \quad (2)$$

где  $\mathbf{e}$  – единичный вектор.

После чего можно получить *матрицу центрированных данных*:

$$\mathbf{A}_c = \mathbf{X} - \mathbf{e}\bar{\mathbf{x}}. \quad (3)$$

Затем по каждому показанию по всем измерениям вычисляется вектор стандартных отклонений  $\sqrt{\mathbf{S}_n^2}$ . После чего можно получить *матрицу стандартизованных данных*:

$$\mathbf{A} = \mathbf{A}_c \sqrt{\mathbf{S}_n^2}, \quad (4)$$

где  $S_{n_i}^2$  является несмещенной выборочной дисперсией:

$$S_{n_i}^2 = \frac{1}{m-1} \sum_{j=1}^m (x_{ji} - \bar{x}_i)^2. \quad (5)$$

Далее для нахождения главных компонент стоит воспользоваться теоремой из лекции [2].

**Теорема 1.** Главными компонентами матрицы стандартизованных данных  $\mathbf{A}$  являются ее сингулярные векторы, при этом  $j$ -ая главная компонента соответствует  $j$ -му сингулярному вектору  $\mathbf{q}_j$  и стандартному отклонению  $\sqrt{\nu}\sigma_j$ , где  $\sigma_j$  –  $j$ -ое сингулярное число.

Значение  $\nu = \frac{1}{m-1}$  – коэффициент для построения несмещенной оценки выборочной дисперсии (и, как следствие, стандартного отклонения). Число  $m$  равно размеру выборки – числу измерений в матрице данных. Так же необходимо привести некоторые определения и свойства из указанных выше лекций, чтобы можно было воспользоваться теоремой 1.

**Определение 2.** Сингулярными числами матрицы  $A \in \mathbb{R}^{m \times n}$  называются неотрицательные вещественные числа  $\sigma_i = \sqrt{\lambda_i}$ , где  $\lambda_i$  – ненулевые собственные числа матрицы Грама  $K = A^T A$ . Собственные векторы матрицы Грама  $K$  называют *сингулярными векторами*.

**Свойство 2.1.** Сингулярные числа сортируются в невозрастающую последовательность  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$ . Где  $r$  – ранг определенных выше матриц  $A$  и  $K$ .

Теперь есть все необходимые данные для реализации функции `pca(A)`. Ее реализация представлена в **листеинге 1**.

**Входные данные** для функции `pca(A)` – двумерный список  $A$ , значения которого равны значениям матрицы стандартизованных данных.

**Алгоритм**, реализованный в данной функции, следующий. Сначала вычисляется количество измерений в матрице данных – количество списков в списке  $A$ . Затем вычисляется матрица Грама, исходя из **определения 2**, данного выше, – список  $K$ . После чего вычисляются собственные значения и собственные числа матрицы Грама путем использования функции `eig(K)` из модуля `numpy.linalg` модуль языка программирования Python. Так как [4] данная функция возвращает неотсортированные по невозрастанию собственные значения, то далее в алгоритме происходит их сортировка. После чего вычисляется список `sigma`, который содержит значения стандартных отклонений, соответствующих главным компонентам.

**Выходными данными** функции `pca(A)` являются списки со значениями координат главных компонент и список со значениями стандартных отклонений, соответствующих главным компонентам.

**Листинг 1:** Определение функции `pca(A)`, которая возвращает главные компоненты и стандартные отклонения.

---

```
def pca(A):
    m = len(A)
    K = A.T @ A

    w, v = np.linalg.eig(K)

    idx = w.argsort()[:-1]
    w = w[idx]
    v = v[:,idx]

    sigma = sqrt(w) / sqrt(m - 1)
    return(v.T, sigma)
```

---

## 5 Получение и подготовка исходных данных и использование в pca(A)

Для удобства последующей работы использован модуль `pandas` языка программирования Python. С помощью функции `read_csv` были считаны данные из скаченного файла. Затем с помощью функций `mean()` и `std()` были вычислены средние значения и стандартные отклонения для матрицы данных (1 - 5), как было описано в предыдущем пункте данной работы. После чего данные были стандартизованы и переданы в функцию `pca(A)`. В переменные `v` и `w` были занесены, соответственно, списки со значениями координат главных компонент и список со значениями стандартных отклонений, ассоциированных с главными компонентами. Все это реализовано в **листеинге 2**.

**Листинг 2:** Получение, подготовка и использование данных в функции `pca(A)`

---

```
col = [i + 1 for i in range(30)]                                # названия столбцов признаков
df = pd.read_csv('wdbc.data',
                  names = ['ID', 'D'] + col)                         # получение данных из файла
x = np.array([df[i + 1].mean() for i in range(30)])           # вычисление средних
                                                               # значений для каждого признака
ex = np.array([x for i in range(569)])                          # формирование матрицы
                                                               # со средними значениями
sigma = np.diag(np.array([1 / df[i + 1].std()
                           for i in range(30)]), 0)                 # вычисление
                                                               # стандартных отклонений
X = df.loc[:, col].values                                     # получение матрицы количественных признаков
A = (X - ex) @ sigma                                         # стандартизация данных
v, w = pca(A)                                                 # применение функции pca(A) к модифицированным данным
```

---

## 6 Выход стандартных отклонений, соответствующих номерам главных компонент

По **свойству 2.1** сингулярные числа сортируются в невозрастающую последовательность. Поэтому, как видно из рисунка 2, функция зависимости стандартного отклонения от номера главной компоненты является невозрастающей. Вспомогательные функции для вывода графиков хранятся в файле `cm_lab4.ipynb`.

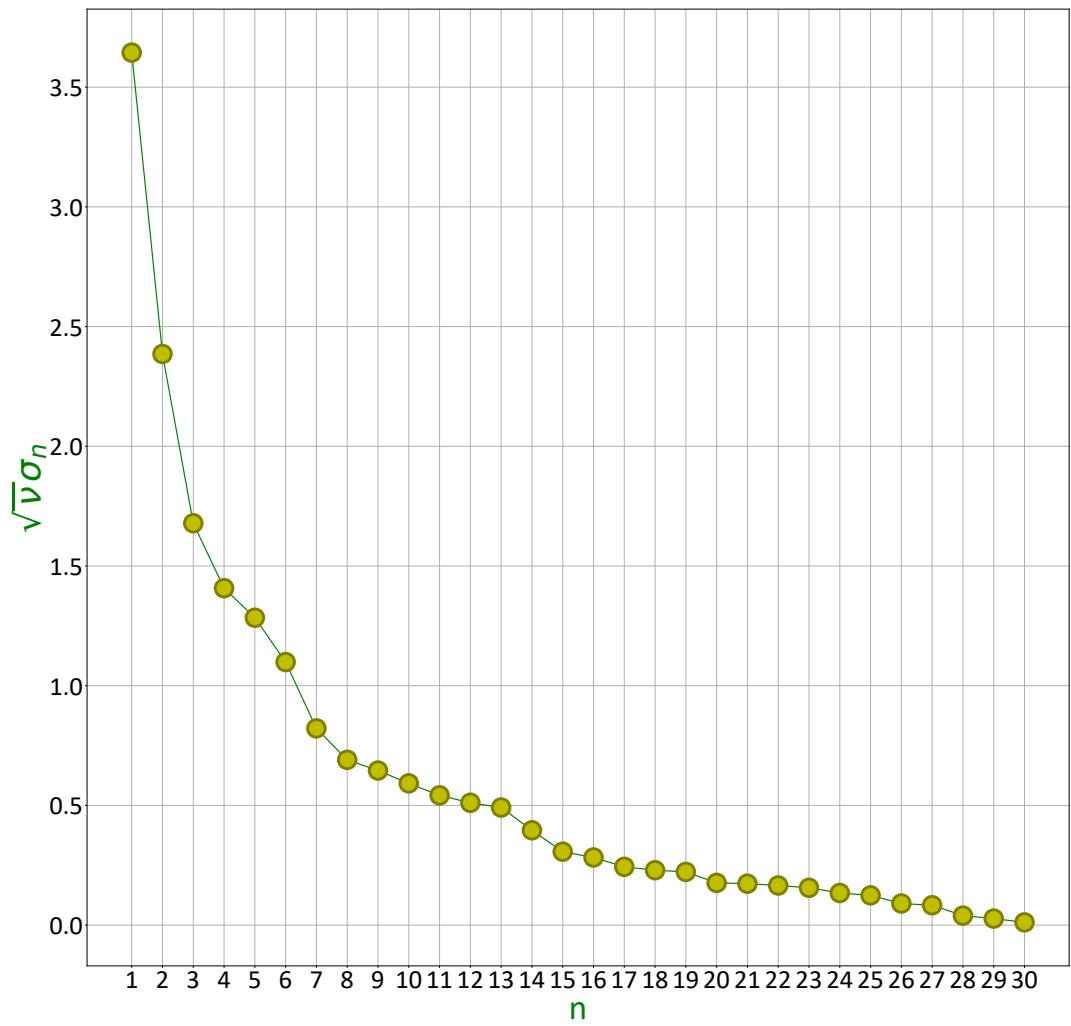


Рис. 2. Стандартные отклонения, соответствующие номерам главных компонент,  $n$  — номер главной компоненты,  $\sqrt{\nu}\sigma_n$  — соответствующее ей стандартное отклонение

## 7 Проецирование стандартизованных данных на первые две главные компоненты

Для того, чтобы построить проекции стандартизованных данных на первые две главные компоненты, необходимо построить матрицу, столбцами которой будут являться две первые главные компоненты. Пусть эта матрица равна  $P_c = [q_1 \ q_2]$ , где

$\mathbf{q}_1$  и  $\mathbf{q}_2$  – первые две главные компоненты. Тогда матрица, содержащая проекции центрированных данных на первые две главные компоненты будет равна  $\mathbf{A}_p = \mathbf{A}\mathbf{P}_c$ . Чтобы построить список со значениями матрицы  $\mathbf{P}_c$ , была использована функция `column_stack([v[0], v[1]])` из модуля `numpy` языка программирования Python. Все необходимые вычисления приведены в **листеинге 3**. Функция для вывода графика находится в файле `cm_lab4.ipynb`.

**Листинг 3:** Проецирование данных на первые две главные компоненты

---

```
pc_vec = np.column_stack([v[0], v[1]])
A_proj = A @ pc_vec
```

---

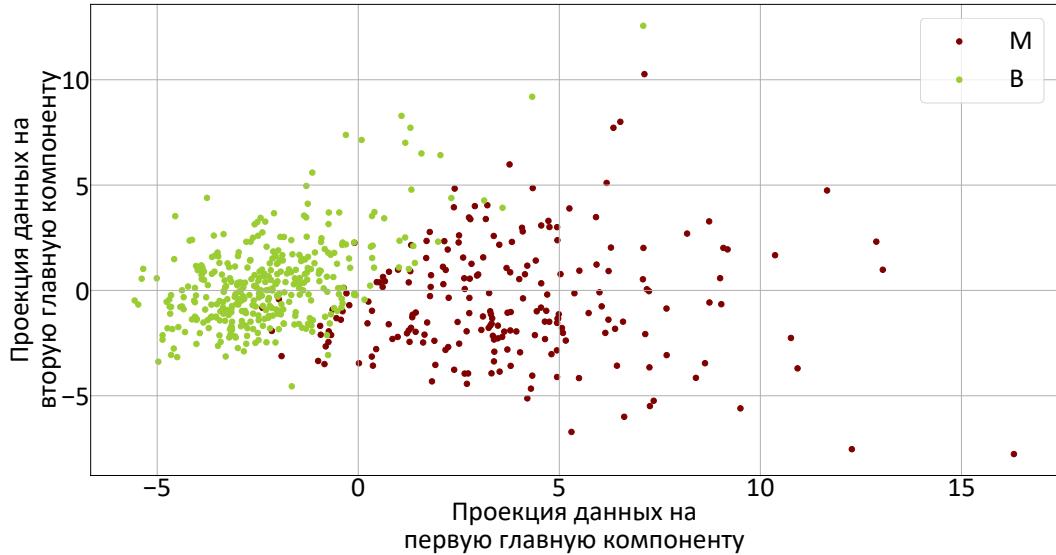


Рис. 3. Проекции матрицы стандартизованных данных на первую и вторую главные компоненты

Как видно из рисунка 3, проекций на первые две главные компоненты действительно достаточно, чтобы провести сепарацию типов опухолей. Зеленые точки на графике – данные, имеющие в столбце диагноза значение 'B' (добропачественная опухоль), красные точки – данные, имеющие в столбце диагноза значение 'M' (злокачественная опухоль).

## 8 Построение лапласианов для каждого графа

Как известно из условия задания, лапласиан графа – матрица  $\mathbf{L} = \mathbf{D} - \mathbf{A}$ , где  $\mathbf{A}$  – матрица смежности графа и  $\mathbf{D}$  – матрица, на главной диагонали которой расположены степени вершин графа, а остальные элементы равны нулю. Для вычисления

лапласиана графа была реализована функция `laplac(G)`. Определение данной функции приведено в **листиинге 4**.

**Входными данными** функции `laplac(G)` является двумерный список `G`, в котором хранятся значения матрицы смежности графа  $G$ .

**Алгоритм**, реализованный в данной функции, следующий. Создается пустой список `D` для хранения степени каждой вершины. Далее в цикле этот список заполняется по матрице смежности: вычисляется сумма каждого элемента списка `G` и добавляется в список `D`. Затем из списка `D` формируется диагональная матрица, на диагонали которой стоят степени вершин графа  $G$ .

**Выходными данными** функции `laplac(G)` является двумерный список, в котором хранятся значения лапласиана данного графа.

**Листинг 4:** Определение функции `laplac(G)`, осуществляющей вычисление лапласиана графа

---

```
def laplac(G):
    D = []
    for g in G:
        D.append(sum(g))
    return np.diag(D, 0) - G
```

---

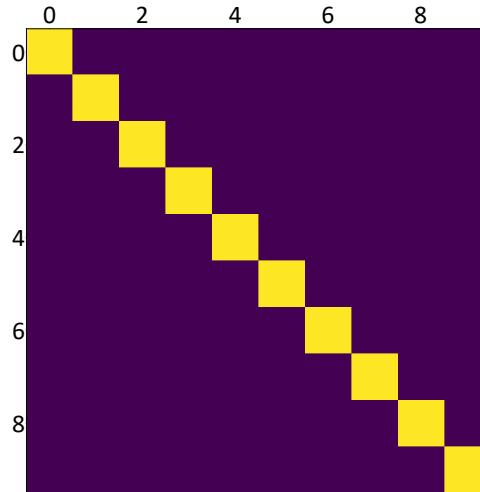


Рис. 4. Лапласиан графа  $G_1$

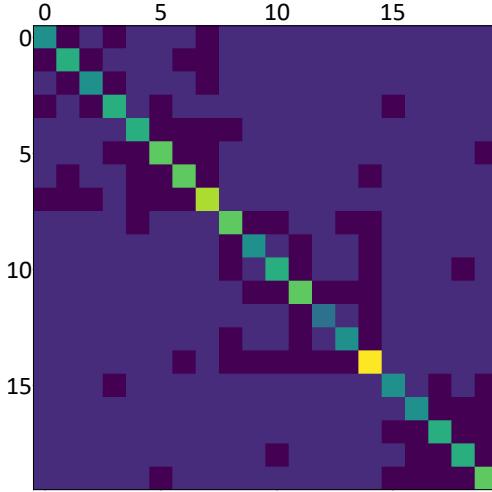


Рис. 5. Лапласиан графа  $G_2$

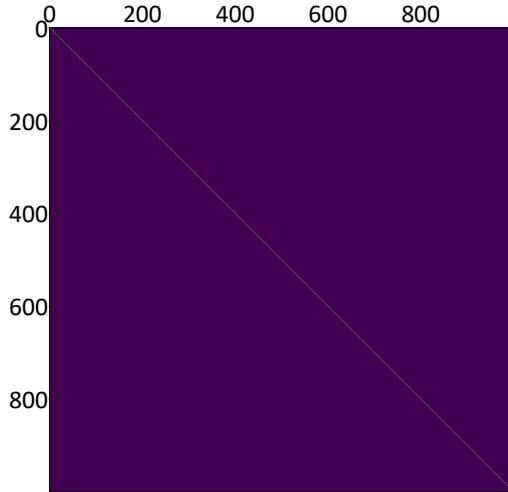


Рис. 6. Лапласиан графа  $G_3$

На рисунках 4, 5 и 6 приведены лапласианы графов  $G_1$ ,  $G_2$  и  $G_3$ , соответственно. Графики были выведены с помощью функции `matshow(M)` из модуля `matplotlib.pyplot` языка программирования Python. Можно заметить, что чем меньше значения в лапласиане графа, тем более темной является ячейка для этого значения на графике (на диагонали стоят положительные значения, недиагональные элементы равны 0 или -1).

## 9 Доказательство утверждения для лапласиана графа

Для доказательства того, что лапласиан неориентированного невзвешенного графа с  $n$  вершинами является положительно полуопределенной матрицей, имеющей  $n$  неотрицательных собственных чисел, одно из которых равно нулю, сначала будет доказано, что лапласиан неориентированного невзвешенного графа является матрицей Грама, а затем будет показано наличие нулевого собственного числа.

◀ По определению 2 матрицу Грама можно представить в виде  $\mathbf{K} = \mathbf{P}^T \cdot \mathbf{P}$ , где  $\mathbf{P} \in \mathbb{R}^{m \times n}$ , так как ее элементами являются скалярные произведения  $g_{ij} = (p_i, p_j)$ , где  $p_i$  – столбец матрицы  $\mathbf{P}$ . В таком случае необходимо доказать, что лапласиан можно представить как произведение какой-либо матрицы  $\mathbf{P}$  на ее транспонированную.

Пусть имеется неориентированный невзвешенный граф  $G = (V_G, E_G)$ . Пусть также есть отображение неориентированного графа  $G$  в множество  $\mathbf{H}$  ориентированных графов  $f : G \mapsto \mathbf{H}$ , такое что:

$$\forall H = (V_H, E_H) \in \mathbf{H} \Leftrightarrow V_H = V_G \wedge (\forall e_G = \{v_i, v_j\} \in E_G \exists! e_H \in E_H : e_H = (v_i, v_j) \vee e_H = (v_j, v_i)),$$

где  $v_i, v_j \in V_G$ . Мощность множества  $\mathbf{H}$  равна  $|\mathbf{H}| = 2^{|E_G|}$ , так как для каждого ребра  $e_G = \{v_i, v_j\}$  в неориентированном графе  $G$  можно построить одно из двух возможных ребер в ориентированном графе  $H$  (либо  $e_H = (v_i, v_j)$ , либо  $e_H = (v_j, v_i)$ ).

Без потери общности (что будет показано далее) можно взять любой граф  $H$  из множества  $\mathbf{H}$  и построить для него матрицу инцидентности  $\mathbf{S}$ . Данная матрица будет иметь размер  $|V_H| \times |E_H|$ . Элементами данной матрицы являются значения из множества  $\{-1, 0, 1\}$ , где  $s_{ij} = -1$ , если существует ребро из  $v_j$  в  $v_i$ ,  $s_{ij} = 1$ , если существует ребро из  $v_i$  в  $v_j$  и  $s_{ij} = 0$ , если между  $v_i$  и  $v_j$  ребра не существует.

Теперь несложно заметить, что при перемножении  $\mathbf{S}^T$  и  $\mathbf{S}$  в результирующей матрице на диагонали будет стоять степень  $i$ -ой вершины, так как скалярное произведение  $(s_i, s_i)$  вектора самого на себя равно квадрату его нормы  $\|s_i\|_2^2 = \sum_j^{|V_H|} s_{ij}$  ( $i$ -я строка из  $\mathbf{S}^T$  и  $i$ -й столбец из  $\mathbf{S}$  содержат одни и те же элементы). Недиагональные элементы результирующей матрицы при наличии между  $i$ -ой и  $j$ -ой вершинами ребра будут равны  $-1$ , так как граф не имеет петель и кратных ребер, а значит, различные строки в матрице инцидентности на одной и той же позиции могут иметь значения  $-1$  в одной строке и  $1$  в другой только в случае, если между  $i$ -ой и  $j$ -ой вершинами есть ребро, если ребра между ними нет, то в какой-то из строк (или в двух сразу) на одной определенной позиции будет стоять  $0$  (и  $-1, 0$  или  $1$  на этой же позиции в другой строке), поэтому в произведении строки из  $\mathbf{S}^T$  и столбца из  $\mathbf{S}$  будет только один ненулевой элемент, равный  $-1$ , если  $i \neq j$  и между  $i$  и  $j$  есть ребро.

Как видно из рассуждений выше – неважно, идет ли ребро в графе  $H$ , образе графа  $G$ , из вершины  $i$  в вершину  $j$  или наоборот. В любом случае в результирующей

матрице появляется  $-1$  при наличии ребра и  $0$  при его отсутствии в графе  $G$ .

Построенная таким образом матрица  $\mathbf{S}^T \cdot \mathbf{S}$  является матрицей Грама и при этом лапласианом графа  $G$ , потому что диагональные элементы равны степеням вершин графа  $G$ , а недиагональные равны  $0$  или  $-1$ . Если вычесть  $\mathbf{S}^T \cdot \mathbf{S}$  из  $\mathbf{D}$ , то получится матрица смежности графа  $G$ :  $\mathbf{D} - \mathbf{S}^T \cdot \mathbf{S} = \mathbf{A}$ . В итоге найдено разложение лапласиана в произведение двух матриц:  $\mathbf{L} = \mathbf{S}^T \cdot \mathbf{S}$ .

Как показано выше, лапласиан – матрица Грама, поэтому легко показать его положительную полуопределенность, воспользовавшись линейностью скалярного произведения:

$$x^T \mathbf{L} x = \sum_{i,j} x_i^T x_j (l_i, l_j) = \sum_{i,j} (x_i l_i, x_j l_j) = \left( \sum_i x_i l_i, \sum_j x_j l_j \right) = \left\| \sum_i x_i l_i \right\|_2^2 \geq 0.$$

Легко заметить, что сумма элементов каждого столбца в лапласиане равна нулю: в силу симметричности матрицы смежности (и, как следствие, лапласиана) и того факта, что диагональный элемент  $l_{ii}$  – степень  $i$ -ой вершины графа, равная сумме  $i$ -ой строки в матрице смежности, то есть  $l_{ii} = \sum_{j, j \neq i} l_{ij}$  и  $l_{ii} - \sum_{j, j \neq i} l_{ij} = 0$  (аналогично для столбца в силу симметричности).

Раз сумма строк лапласиана равна нулю, то имеется нетривиальная линейная комбинация векторов (строк/столбцов лапласиана) равная нулю, а значит, они являются линейно зависимыми. Отсюда следует, что определитель лапласиана равен нулю.

Для нахождения собственных чисел используется следующее характеристическое уравнение:  $r(\lambda) = \det(\mathbf{A} - \lambda \mathbf{E}) = 0$  ( $r(\lambda)$  – характеристический многочлен). Пусть  $\lambda = 0$ , тогда  $r(0) = \det(\mathbf{A}) = 0$ , но матрица  $\mathbf{A}$  вырожденная, поэтому  $\det(\mathbf{A}) = 0$  всегда выполняется, а значит  $\lambda = 0$  является корнем характеристического уравнения. Поэтому имеется хотя бы одно нулевое собственное число.

Остальные собственные числа будут неотрицательными, так как:

$$x^T \mathbf{L} x = x^T \mathbf{S}^T \mathbf{S} x = (\mathbf{S} x)^T (\mathbf{S} x) = (\mathbf{S} x, \mathbf{S} x) = \|\mathbf{S} x\|_2^2,$$

и если взять собственный вектор  $v$ , то:

$$\lambda \|v\|_2^2 = (\mathbf{L} v, v) = (\mathbf{S}^T \mathbf{S} v, v) = \|\mathbf{S} v\|_2^2 \geq 0.$$

А значит и собственные числа неотрицательны.

Наконец, собственных чисел будет  $n = |V_G|$  (матрица смежности имеет размер  $n \times n$  и диагональная матрица  $\mathbf{D}$  тоже), то есть лапласиан графа на  $n$  вершинах будет иметь  $n$  собственных чисел.

В итоге, как и требовалось доказать, лапласиан действительно является положительно полуопределенной матрицей с  $n$  неотрицательными собственными числами, одно из которых равно нулю. ■

## 10 Поиск спектров каждого из указанных графов

Для нахождения спектров каждого из графов необходимо найти собственные числа и собственные векторы лапласианов этих графов. Лапласианы были вычислены в пункте 8 данной работы. Поэтому остается лишь найти их собственные числа и собственные векторы. Для этих целей была реализована функция `spectrum(G)`. Ее определение приведено в **листе 5**.

**Входными данными** для функции `spectrum(G)` является двумерный список  $G$ , в котором хранятся значения лапласиана соответствующего графа.

**Алгоритм** реализованный в данной функции содержит почти все пункты алгоритма из функции `rsca(A)`, за исключением лишь того, что не осуществляется поиск матрицы Грама и не происходит вычисление квадратных корней из собственных чисел с домножением на  $\sqrt{\nu}$ .

**Выходными данными** функции `spectrum(G)` являются списки со значениями собственных векторов лапласиана ассоциированного с ним графа и список с собственными числами лапласиана.

**Листинг 5:** Определение функции `spectrum(G)`, осуществляющей вычисление спектра графа

---

```
def spectrum(G):
    eigenValues, eigenVectors = np.linalg.eig(G)

    idx = eigenValues.argsort() [::-1]
    eigenValues = eigenValues[idx]
    eigenVectors = eigenVectors[:,idx]

    return(eigenVectors.T, eigenValues)
```

---

Код вызова реализованной выше функции и код для вывода графиков приведены в файле `sm_lan4.ipynb`. Теперь можно подробнее рассмотреть спектры каждого из трех графов.

Интересно рассмотреть собственные значения лапласианов каждого графа. Для большей наглядности они были отсортированы по невозрастанию.

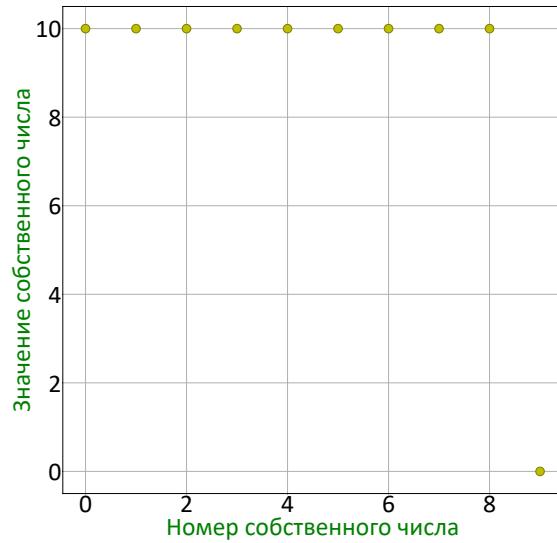


Рис. 7. Отсортированные по невозрастанию собственные числа лапласиана графа  $G_1$

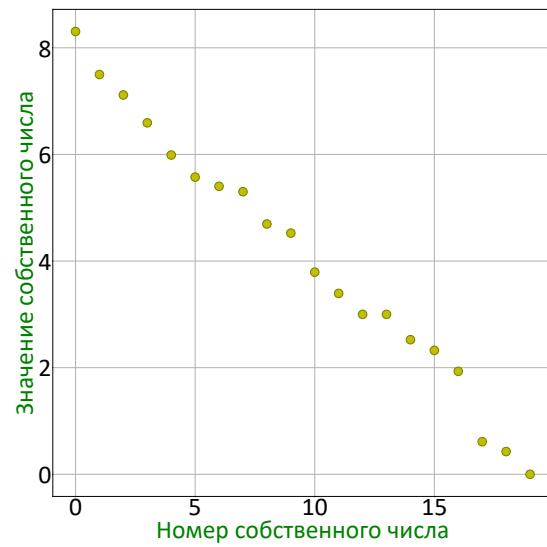


Рис. 8. Отсортированные по невозрастанию собственные числа лапласиана графа  $G_2$

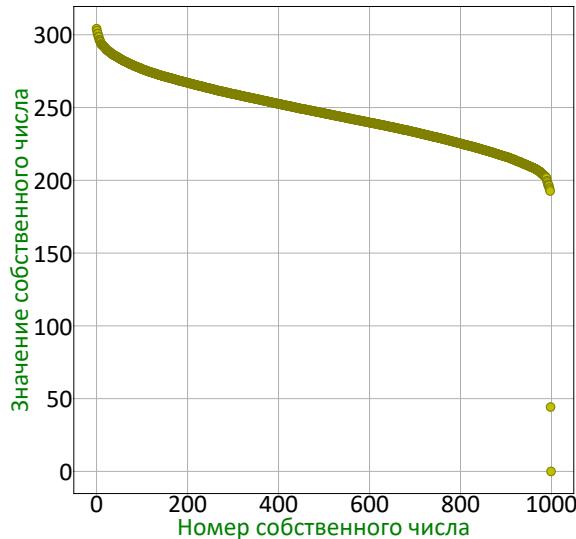


Рис. 9. Отсортированные по невозрастанию собственные числа лапласиана графа  $G_3$

Как видно из рисунков выше, каждый лапласиан имеет нулевое собственное значение. Интерес представляет второе по величине собственное значение.

Так, для графа  $G_1$ , как видно из рисунка 7, второе по величине собственное значение не отличается от остальных. Напротив, для графа  $G_2$  это значение, судя по рисунку 8 достаточно небольшое, относительно остальных собственных значений. На данном этапе можно сделать некоторый эмпирический вывод, что чем плотнее связан граф, тем больше первое ненулевое собственное число его лапласиана.

**Таблица 1:** значения вторых по величине собственных чисел лапласианов каждого из трех графов

Граф	$G_1$	$G_2$	$G_3$
Второе по величине собственное число	10.00	0.43	44.25

Судя по таблице 1 и рисунку 9, третий граф имеет также относительно небольшое второе по величине собственное число. Причем после него уже третье по величине собственное число достаточно велико, относительно второго.

Теперь можно посмотреть на собственные векторы лапласианов графов, которые соответствуют второму по величине собственному числу.

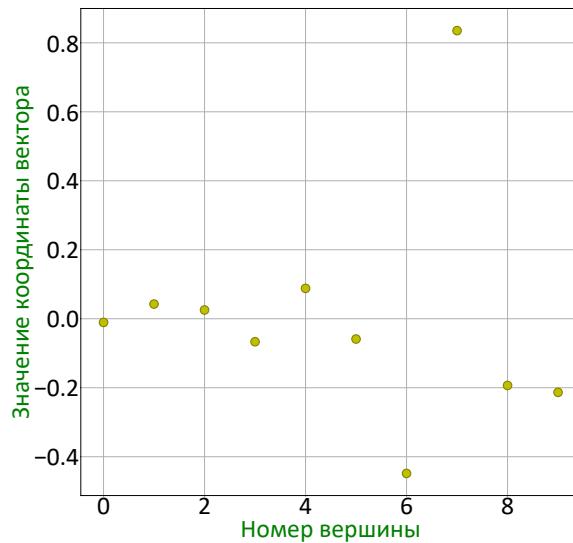


Рис. 10. Отсортированный собственный вектор лапласиана графа  $G_1$ , ассоциированный со вторым по величине собственным числом лапласиана

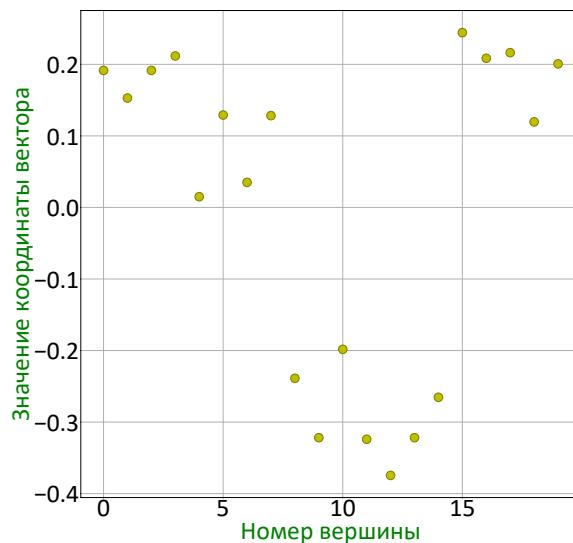


Рис. 11. Отсортированный собственный вектор лапласиана графа  $G_2$ , ассоциированный со вторым по величине собственным числом лапласиана

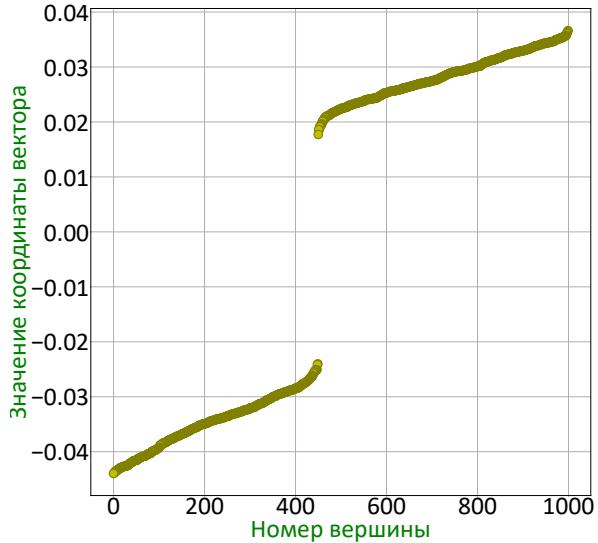


Рис. 12. Отсортированный собственный вектор лапласиана графа  $G_3$ , ассоциированный со вторым по величине собственным числом лапласиана

Из рисунков 10 - 12 видно, что, например, для графа  $G_2$ , про который известно, что он имеет три кластера, сортировка второго собственного вектора, соответствующего второму по величине собственному числу, позволила своеобразно разделить вершины между собой. Для графа  $G_1$  вершины точки расположены достаточно разрозненно. Это происходит из-за его особенностей – граф  $G_1$  является полным, поэтому выделить более одного кластера в нем невозможно. Что касается графа  $G_3$ , в нем, судя по графику на рисунке 12, выделяются два кластера (часть координат принимает отрицательные значения, далее – разрыв, и вторая часть принимает положительные значения). Это предположение будет подтверждено далее.

## 11 Определение количества кластеров в графе $G_3$

Для сортировки матрицы была реализована функция `matsort(vec, M)`. Ее определение представлено в [листе 6](#).

**Входными данными** для функции `matsort(vec, M)` являются список `vec`, по которому будет происходить сортировка и двумерный список `M`, который необходимо отсортировать по строкам и столбцам.

**Алгоритм**, реализованный в данной функции такой же как, например, в части функции `rca(A)`, где происходит сортировка списка собственных чисел. Кроме того в `matsort(vec, M)` по полученным индексам сортируется передаваемый в нее двумерный список.

**Выходными данными** функции `matsort(vec, M)` является отсортированный двумерный список.

**Листинг 6:** Определение функции `matsort(vec, M)`, осуществляющей сортировку матрицы смежности графа

---

```
def matsort(vec, M):
    idx = vec.argsort(axis = 0)
    vec = vec[idx]
    M = M[idx]
    M = M[:, idx]
    return M
```

---

Для определения количества кластеров в графе  $G_3$ , был взят собственный вектор, соответствующий второму по величине собственному числу, лапласиана данного графа. Затем он и матрица смежности графа  $G_3$  были переданы в `matsort(vec, M)`. Исходная матрица смежности изображена на рисунке 13, отсортированная указанным выше способом матрица смежности графа  $G_3$  изображена на рисунке 14.

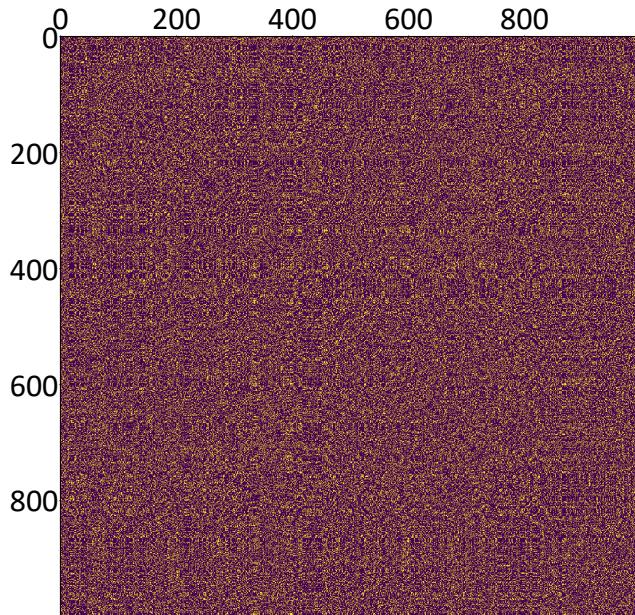


Рис. 13. Исходная матрица смежности графа  $G_3$

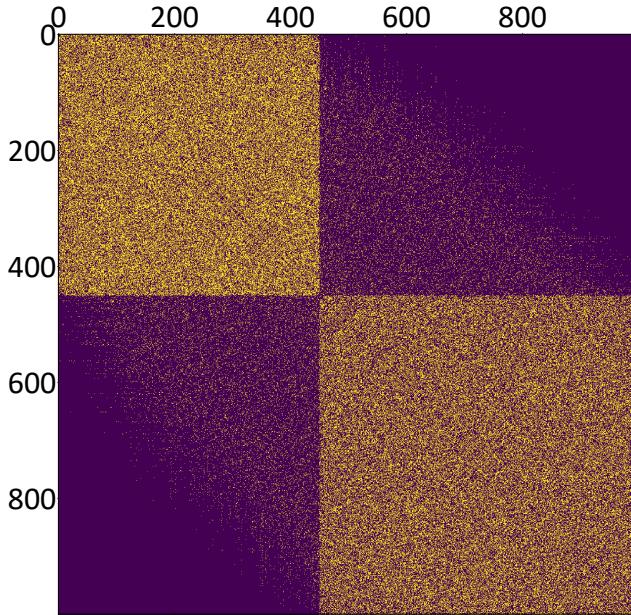


Рис. 14. Матрица смежности графа  $G_3$ , отсортированная по координатам второго собственного вектора лапласиана графа  $G_3$

Как видно по рисунку 14, предположение о наличии в графе  $G_3$ , исходя из анализа в предыдущем пункте, о том, что данный график имеет два кластера, подтвердилось.

Стоит отметить, что собственный вектор лапласиана графа, ассоциированный со вторым по величине собственным числом, называется вектором Фидлера [5]. А второе по величине собственное число лапласиана – числом Фидлера. Судя по оригинальной статье, значение Фидлера содержит информацию о том, каков минимальный срез графа, необходимый для его разделения на две компоненты связности.

Также второе по величине собственное число лапласиана графа называют алгебраической связностью графа [6]. Можно заметить, что второе собственное число положительно тогда и только тогда, когда график является связным. Понятно, что ранг лапласиана графа равен  $n - c$ , где  $n$  – количество вершин,  $c$  – количество компонент связности. Тогда для связного графа, ранг всегда будет  $n - 1$ , а значит всегда будет одно нулевое собственное число. И поэтому второе собственное число по величине будет уже положительным. Однако если компонент связности больше, чем одна, то нулевых чисел будет столько, сколько компонент связности в графике. Если компонент связности, например, две, то первым положительным собственным числом уже будет третье по величине собственное число лапласиана этого графа.

Еще одним интересным наблюдением является то, что компоненты связности можно выделить даже в матрице смежности, просто нужным образом переставив строки

и столбцы. Из-за симметрии получается блоки – в сущности блочная матрица. И если в каждом из блоков сложить строки, то вся матрица обнуляется, в силу симметрии.

Как видно из анализа спектра приведенных в данной работе трех графов, все они являются связными, так как имеют всего одно нулевое собственное число.

Также можно попробовать отобразить исходную матрицу смежности графа  $G_3$  на вектор Фидлера. Код приведен в файле `cm_lab4.ipynb`, а результат можно увидеть на рисунке 15.

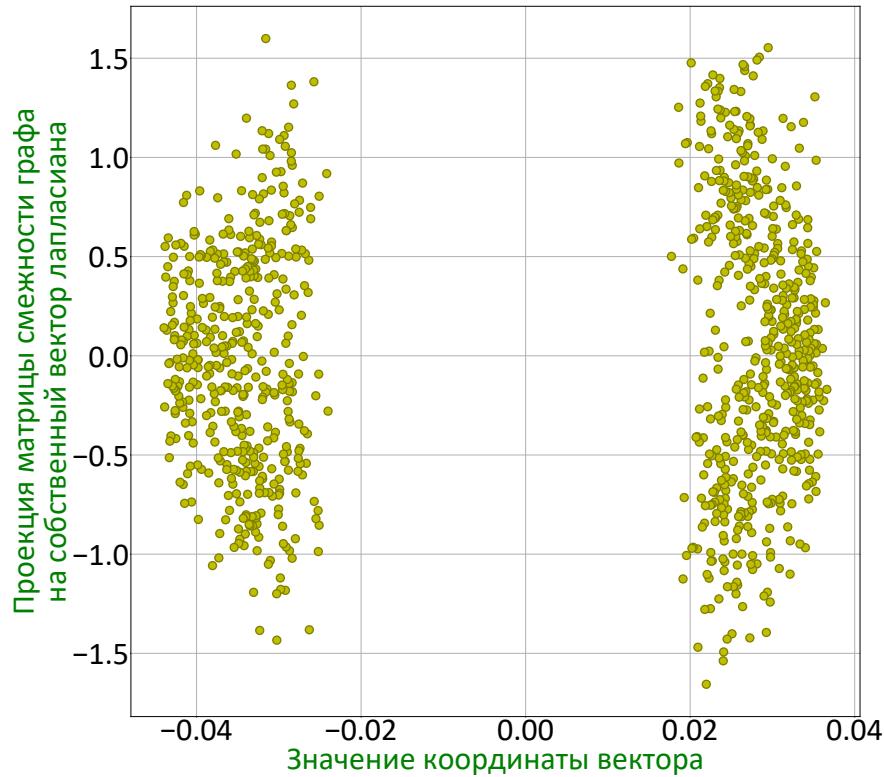


Рис. 15. Проекция матрицы смежности на второй отсортированный собственный вектор для графа  $G_3$

Здесь также видно наличие двух множеств точек на плоскости, которые разделены на две группы. Первая группа соответствует отрицательным значениям вектора Фидлера, вторая – положительным.

Таким образом, кластеризовать граф  $G_3$  удалось путем сортировки матрицы смежности, а также проецированием ее на вектор Фидлера.

## 12 Заключение

1. В результате выполненных задач можно отметить определенную связь между спектральным и сингулярным разложениями матриц. Даже программная реализация алгоритмов достаточно похожая.
2. Сингулярное разложение позволяет уменьшить размерность исходных данных без потери важной информации, извлекаемой из них. Это может быть полезно в случае, когда признаков очень много и выборка достаточно объемная, и при этом есть потребность быстро оценить какие-то качественные характеристики.
3. Спектральный анализ графов позволяет делать выводы как об их связности, так и о наличии в них кластеров. При этом важным объектом в спектральном анализе является вектор Фидлера.
4. Результатом доказанной теоремы служит очень важное свойство лапласиана графа – он является матрицей Грама, и, как следствие, проецирование данных на его собственные векторы позволяет также извлекать информацию, как это происходит при проецировании на главные компоненты.

## Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018–2021. С. 140.
2. Лекция по вычислительной математике номер 14 [https://www.youtube.com/watch?v=isU\\_HPUC0ds](https://www.youtube.com/watch?v=isU_HPUC0ds)
3. Лекция по вычислительной математике номер 15 [https://www.youtube.com/watch?v=PyP9T4\\_uUHM](https://www.youtube.com/watch?v=PyP9T4_uUHM)
4. <https://numpy.org/doc/stable/reference/generated/numpy.linalg.eig.html>
5. Miroslav Fiedler. Algebraic connectivity of graphs. Czechoslovak Mathematical Journal, Vol. 23 (1973), No. 2, 298–305
6. M. Fiedler. Laplacian of graphs and algebraic connectivity. Combinatorics and Graph Theory. — 1989. — №. 23

## Выходные данные

Горелкина Е.Е.. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] – Москва: 2021. – 24 с. URL: <https://sa2systems.ru:88> (система контроля версий кафедры РК6)

Постановка:  ассистент кафедры PK-6, PhD А.Ю. Першин  
Решение и вёрстка:  студент группы PK6-52Б, Горелкина Е.Е.

*2021, осенний семестр*