МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ

«АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных технологий

Кафедра математического анализа

**АЛЬТЕРНАТИВНЫЕ ПОДХОДЫ К ВИЗУАЛИЗАЦИИ СТАТИСТИЧЕСКИХ ДАННЫХ**

(выпускная квалификационная работа)

Выполнил:

студент 405 группы,

Калинкин Александр Александрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

Научный руководитель:

к.ф.-м.н., доцент

Дронов Сергей Вадимович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

Допустить к защите:

Зав. кафедрой, к.ф.-м.н., доцент

Саженков Александр Николаевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

*(подпись)*

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Работа защищена:

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель ГЭК:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2024

Оглавление

[Введение 3](#_Toc170229021)

[Глава 1. Основы визуализации многомерных данных 5](#_Toc170229022)

[1.1. Многомерные данные и сложность их визуализации 5](#_Toc170229023)

[1.2. Линейные методы визуализации 6](#_Toc170229024)

[1.3. Нелинейные методы визуализации 10](#_Toc170229025)

[Глава 2. Построение неискаженного изображения 12](#_Toc170229026)

[2.1. Постановка задачи 12](#_Toc170229027)

[2.2. Теоретические основы неискаженной визуализации 14](#_Toc170229028)

[2.3. Методика снижения искажений 23](#_Toc170229029)

[2.4. Основной алгоритм 25](#_Toc170229030)

[Глава 3. Практическое применение алгоритма 27](#_Toc170229031)

[3.1. Описание компьютерной программы 27](#_Toc170229032)

[3.2. Пример применения к данным 27](#_Toc170229033)

[3.3. Сравнение с классическими линейными алгоритмами 30](#_Toc170229034)

[Заключение 34](#_Toc170229035)

[Библиографический список 35](#_Toc170229036)

[Приложение 37](#_Toc170229037)

# Введение

В связи с появившейся сегодня возможностью обрабатывать большие объемы данных, особое значение приобретает задача первоначального грубого анализа этих данных с целью сформулировать предварительные направления исследования и сделать прикидочное заключение о возможных его результатах. Обычно подобный анализ проводится путём изучения некоторых изображений, но если данные имеют достаточно большую размерность, то построение изображений, адекватно отображающих структуру этих данных, представляет собой серьезную математическую задачу.

Существующие на сегодня методы решения такой задачи практически всегда искажают истинную картину данных. И, хотя эти искажения незначительны, но, при возможности избежать их, это стоит сделать.

Цель работы: разработка нового алгоритма построения неискаженного изображения многомерных данных в случае, когда подобное изображение возможно, а также построения изображения с минимальными искажениями в тех случаях, когда не представляется возможным получить неискаженное изображение.

Для достижения этой цели поставлены задачи:

* систематизировать и единообразно изложить основные применяющиеся сегодня методы визуализации многомерных данных;
* найти и обосновать необходимые и достаточные условия для возможности неискаженного изображения данных;
* на основе найденных условий предложить неитеративный алгоритм построения неискаженного изображения данных при наличии такой возможности;
* разработать метод построения изображения с минимальными искажениями в тех случаях, когда не удаётся достичь неискаженного изображения;
* реализовать оба алгоритма в виде компьютерных программ;
* продемонстрировать работу программ на реальных массивах данных;
* сравнить полученные результаты с классическими алгоритмами визуализации.

Выпускная квалификационная работа оформлена на 38 страницах, содержит введение, 3 главы, заключение, библиографический список из 15 наименований и приложение. В приложении описан доступ к расположенному в сети исходному коду программ.

# Глава 1. Основы визуализации многомерных данных

## 1.1. Многомерные данные и сложность их визуализации

Визуализация (изображение) данных – представление в наглядной форме данных эксперимента или результатов теоретического исследования. Создание визуализаций востребовано в любом исследовании, в силу чего этой задаче посвящены многочисленные научные статьи и монографии, см, например, [1 – 3] и библиографию там. Обычно каждый объект наблюдения задан набором числовых данных (вариант 1 ниже) – значениями каких-то своих показателей. Количество показателей, значения которых предполагаются известными у каждого из объектов, называется размерностью задачи. Данные размерности 2 или 3 изобразить просто. Но, если данные имеют более высокую размерность, задача становится нетривиальной. В последние годы были разработаны различные методы для визуализации многомерных данных, что отражено в современных исследованиях, таких как [4]. Также проблемы возникают в случае, когда сами объекты никак не заданы, но известны величины их попарных отличий друг от друга (вариант 2 ниже, обычная ситуация, например, в социологии, см. [5]). Рассмотрим эти два основных варианта задания объектов подробнее и точнее:

1) Через набор  их числовых показателей. Здесь точки-объекты заведомо можно изобразить неискаженным образом, но размерность пространства изображений *p* может оказаться слишком высокой. При таком подходе мы получаем классическую задачу сокращения размерности, именно в той форме, в какой она была поставлена в [6].

2) Через попарные расстояния  . По сути, получается таблица, где на пересечении -й строки и -го столбца расположено число, которые оценивает различие между объектами .

Перейти от варианта 1 к варианту 2 несложно. Если каждая точка задана координатами, то можно вычислить все расстояния между точками. Задача же перевода данных из 2) в 1) однозначно не решается. Это одна из причин, почему в основной части работы, с мы будем рассматривать исключительно второй способ исходного задания объектов.

## 1.2. Линейные методы визуализации

Простейшими методами визуализации являются те, для реализации которых приходится применять линейные преобразования и решать линейные уравнения. Такие методы будем называть линейными. Кратко рассмотрим два основных линейных метода визуализации многомерных данных.

Метод главных компонент – один из методов решения задачи визуализации многомерных данных путём сокращения размерности, потеряв при этом как можно меньшее количество информации. Этот метод дает возможность по числу *p* исходных показателей выделить заранее заданное и обычно небольшое количество *q* главных компонент или обобщенных признаков. Эти обобщенные признаки подменяют собой первоначальные, сокращая тем самым размерность задачи.

Метод главных компонент часто используется для представления многомерной выборки данных на двумерном чертеже. Главными компонентами называют направления наибольшей протяженности облака данных в исходном *p*-мерном пространстве. Для визуализации выбирают две первые главные компоненты и проецируют данные на двумерную плоскость, натянутую на них. Проекция на главные компоненты является наименее искаженной из всех линейных проекций многомерной выборки на какую-либо пару осей. Этот метод обоснован, например, в [7, 8]. Там же приводятся и доказываются его свойства. Как правило, в осях главных компонент удаётся увидеть наиболее существенные особенности исходных данных, даже несмотря на неизбежные искажения. Перейдем к подробностям.

Пусть – исходные данные, каждый из этих векторов  имеет размерность . Опишем алгоритм главных компонент c двумя главными компонентами.

1) Стандартизуем набор данных. Цель первого шага –стандартизировать диапазоны исходных переменных, чтобы избавиться от большого разброса значений. Для каждого вектора рассчитываются новые значения его координат:



где и  – среднее значение и стандартное отклонение *j*-го признака соответственно.

После этого составляется новая матрица :

.

2) Вычислим ковариационную матрицу строк новой матрицы:

.

3) Найдём её собственные значения и собственные векторы единичной длины. Затем отсортируем собственные значения по убыванию. Выберем два наибольших собственных значения и введем матрицу , в которой по столбцам размещены координаты соответствующих собственных векторов:

.

4) Преобразование исходных данных. На этом шаге, который является последним, цель состоит в том, чтобы использовать вектор признаков, сформированный с использованием собственных векторов ковариационной матрицы, чтобы переориентировать данные с исходных осей на те, которые представлены главными компонентами:



Строки полученной матрицы размерами  используются в качестве координат изображений объектов на плоскости.

Другой метод, называемый методом многомерного шкалирования – это схожий с главными компонентами способ визуализации данных, по сути, являющийся усовершенствованным методом главных компонент. Изображения строятся так, чтобы большему расстоянию между исходными объектами соответствовало большее расстояние между их изображениями, т.е. сохранился порядок первоначальных расстояний.



На вход алгоритма многомерного шкалирования подаётся матрица различий *D*. На месте *(i, j)* в ней стоит оценка меры различия *i*-го и *j*-го объектов. Если бы задачу удалось решить, и мы построили бы матрицу *S*, то было бы справедливо представление:

Рассмотрим один из вариантов реализации метода многомерного шкалирования – алгоритм Торгерсона, см. [9, 10].

Примем предположение о том, что задача имеет решение, т.е. для элементов заданной на входе алгоритма матрицы *D* имеется нужное представление (координаты визуализаций объектов в *q*-мерном пространстве могут быть найдены).

Алгоритм предлагает перейти к новой вспомогательной матрице  с двойным центрированием, в которой элементами служат следующие числа:

, *i, j = 1,… , n,*

где , , 

Основная теорема Торгерсона [9] утверждает, что для такой матрицы координаты точек-визуализаций с условием (1) всегда могут быть найдены. Координаты этих точек, решающие задачу многомерного шкалирования, являются элементами строк матрицы *X*, которая определяется из матричного уравнения



Теорема Торгерсона утверждает, что решение этого уравнения всегда существует.

Для приближенного решения выписанного уравнения из матрицы Δ\* выделяют две главные компоненты и получают первую итерацию изображения. Последующие итерации двигают полученные точки таким образом, чтобы порядок расстояний между этими точками стал более похож на порядок исходных расстояний. Подобное движение может быть организовано по-разному и носит название метрического этапа алгоритма.

## 1.3. Нелинейные методы визуализации

С широким распространением компьютерной техники связано возникновение и участившееся использование на практике нелинейных методов визуализации данных. Эти методы требуют существенно большего объема вычислений, но и дают в итоге менее искаженные результаты, чем уже описанные линейные методы. Приведем здесь краткое описание двух из них.

Самоорганизующаяся карта Кохонена (Self-organizing map – SOM, [11]) состоит из компонентов, называемых узлами или нейронами. Каждый из узлов описывается двумя векторами. Первый — т. н. вектор веса *m*, имеющий такую же размерность, что и входные данные. Второй — вектор *r*, меньшей размерности, представляющий собой координаты узла на карте. Карта Кохонена визуально отображается с помощью ячеек прямоугольной или шестиугольной формы; последняя применяется чаще, поскольку в этом случае расстояния между центрами смежных ячеек одинаковы, что повышает корректность визуализации карты. Суть метода заключается в том, что он позволяет отобразить многомерные данные на двумерную сетку таким образом, чтобы близкие по значению данные оказались близко друг к другу на карте. Нелинейный алгоритм SOM сегодня успешно применяется во многих практических и прикладных задачах, см., например, [12].

Алгоритм быстрого нелинейного отображения (БНО) (его описание можно найти в [6]) основан на том, что в -мерном пространстве координаты точки  можно однозначно определить набором евклидовых расстояний  между  и  соответствующим образом выбранными опорными точками . Для определения  используется система из  линейных уравнений , где -я строка матрицы  есть  , а -я компонента вектора  равна

.

Если опорные точки зафиксированы, то в качестве функционала качества отображения может быть использована следующая величина:



где  – расстояние между объектом  и опорной точкой  в исходном пространстве, а  – аналогичные расстояния, но в пространстве образов.

Итерационная процедура уточнения координат точки  в пространстве образов имеет вид:

, где

Пусть  - объекты из набора данных, выбранные в качестве опорных. Чтобы найти образы опорных :

1. Вычисляется матрица попарных скалярных произведений размера 



1. Вычисляются собственные числа и векторы матрицы *V*. Компоненты собственных векторов и являются координатами опорных точек в пространстве образов.

# Глава 2. Построение неискаженного изображения

## 2.1. Постановка задачи

Пусть в результате наблюдений или опросов у нас имеются оценки попарных расстояний между какими-либо объектами. Будем считать, что эти оценки представляют собой точные значения соответствующих расстояний. В результате наблюдений или опросов у нас имеются оценки попарных расстояний между какими-либо объектами. Будем считать, что эти оценки представляют собой точные значения соответствующих расстояний.

Задача состоит в том, чтобы по заданной таблице попарных расстояний  между некоторым количеством объектов  выяснить, возможно ли построить плоское изображение без искажений этих расстояний и, если это возможно, предложить алгоритм этого построения.

Иногда неискаженное изображение расстояний бывает невозможно организовать, например, потому что они слишком велики или слишком малы. Попробуем, к примеру, изобразить без искажений карту России по данным расстояний между городами. Чтобы это действительно можно было построить карту, придется пропорционально уменьшить, т.е. исказить все расстояния. Следовательно, для корректной формулировки задачи нужно договориться одновременное пропорциональное уменьшение или увеличение всех расстояний не считать искажениями. Именно поэтому задачу на построение наименее искаженных расстояний принято формулировать как задачу соответствия расстояний между реальными объектами и расстояний между их изображениями.

Условимся далее считать, что все наши расстояния одновременно масштабированы таким образом, чтобы каждое из них в отдельности можно было изобразить на доступном нам листе бумаги.

Поскольку после масштабирования свойства расстояния не нарушаются, следовательно, для них, в частности, выполнено неравенство треугольника:



Перед нами поставлена задача наглядно представить имеющиеся данные. Конкретно – объекты должны быть изображены точками в евклидовом пространстве, так, что степень различия между ними отображалась в расстояниях между этими точками, что, после масштабирования превращается в условие точного равенства расстояний между объектами и их изображениями.

Все изложенные в главе 1 методы искажают в изображениях исходно заданные расстояния. Но, если данных немного, их иногда можно отобразить точно. Но при этом неискаженное изображение не всегда возможно. Следовательно, вначале нужно убедиться в существовании такой возможности.

Наряду с этим нужно разработать метод по построению наименее искаженных изображений в случаях, когда неискаженное изображение построить невозможно.

Мы собираемся строить изображения не в произвольном евклидовом пространстве, а на плоскости, что позволит визуализировать данные, видимо, наиболее удобно.

Итак, задача выглядит следующим образом: по заданной таблице попарных расстояний  между объектами  выяснить, возможно ли построить плоское изображение без искажений этих расстояний и, если это возможно, предложить алгоритм этого построения. Для объектов, которые невозможно представить на изображении без искажений, предлагается разработать алгоритм, направленный на минимизацию возможных искажений.

Поскольку решающим обстоятельством является не конкретное положение точек, изображающих наши объекты, а их взаимное расположение, то условимся считать, что изображения, совмещаемые при помощи движений плоскости или осевых симметрий, совпадают. Таким образом, если любые два из получаемых решений могут быть совмещены описанным образом, то будем говорить, что задача визуализации имеет единственное решение.

## 2.2. Теоретические основы неискаженной визуализации

Начнем с алгоритма для двух точек и будем постепенно увеличивать их количество. Для двух точек: возьмем точку  в любом месте плоскости, в произвольном направлении от нее построим отрезок длины , на конце которого размещаем точку . Это построение возможно при произвольном неотрицательном значении . Заметим, что, как бы мы не строили отрезок заданной длины, любой такой способ можно перевести в любой другой параллельным переносом или осевой симметрией, при необходимости меняющей концы отрезка местами. Значит, решение здесь будет единственным.

Для трёх точек: сначала с помощью метода, описанного выше, построим отрезок, концы которого изображают первую и вторую точки  и . Далее строим окружности с центрами в этих точках радиусов  и  соответственно. Поскольку мы считаем, что известные нам различия между объектами даны безошибочно, то эти две окружности обязательно пересекаются в силу справедливости неравенства треугольника. Их пересечение обычно представляет собой две точки. Эти точки симметричны относительно прямой  Поэтому, хотя в качестве изображения третьего объекта можно выбрать любую из них, решение будет также одно, и, как следует из описанного построения, такое решение всегда существует.

Для четырёх точек: для начала нужно проверить, возможно ли такое построение. Известно, что тетраэдр однозначно определяется заданными длинами своих ребер ([13]), следовательно, наши 4 точки обязательно будут являться вершинами этого тетраэдра. Но если все вершины этого тетраэдра не лежат в одной плоскости, то неискаженное построение невозможно.

**Лемма 1.** ([13], с. 3). *Пусть известны все попарные расстояния*  *между четырьмя точками. Тогда объем тетраэдра с вершинами в этих точках независимо от конкретного его расположения в трехмерном пространстве, может быть найден из формулы:*



Из этой леммы вытекает, что неискаженное изображение четырех точек на плоскости возможно тогда и только тогда, когда 

Если такое изображение возможно, то строим первые три точки из наших четырех так, как это было описано выше. Затем построим возможные четвёртые точки ,  пересечением окружностей в центрах  и  с радиусами . Обозначим расстояния от точек  и  до точки , как  и  соответственно. Аналогично для точки . Нетрудно заметить, что  и . Таких точек , для которых выполняются оба равенства, всего две. Вывод: если решение есть, то это одна из точек  или (см. рис. 1).

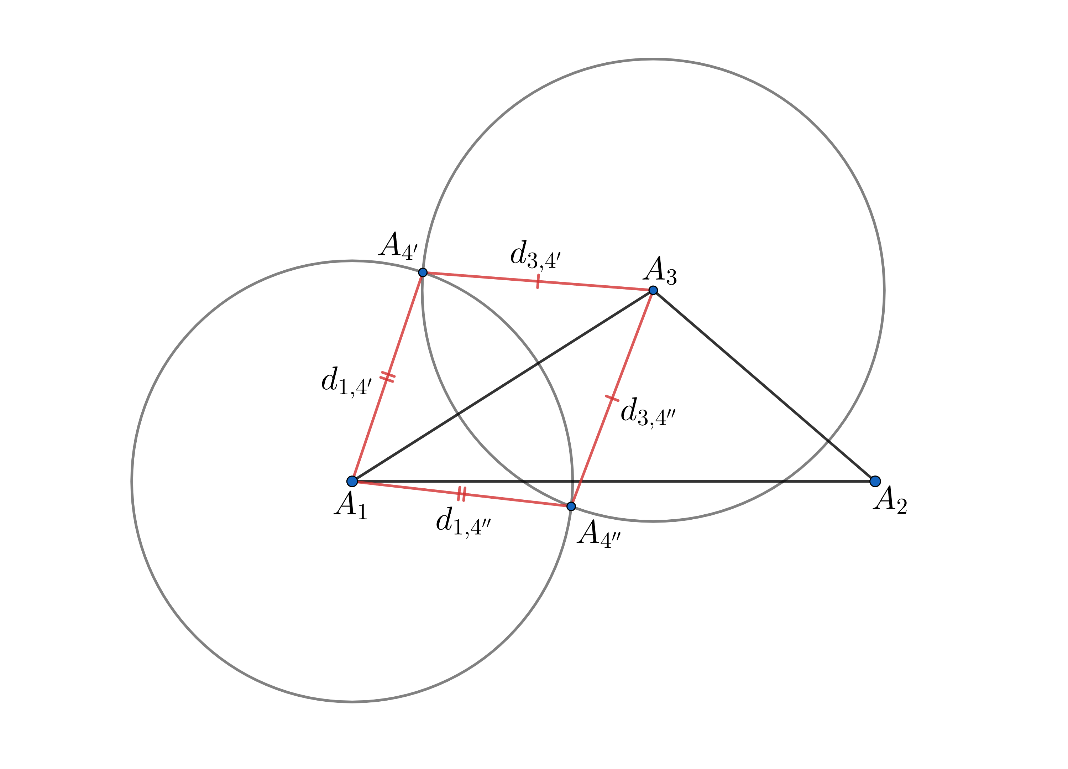
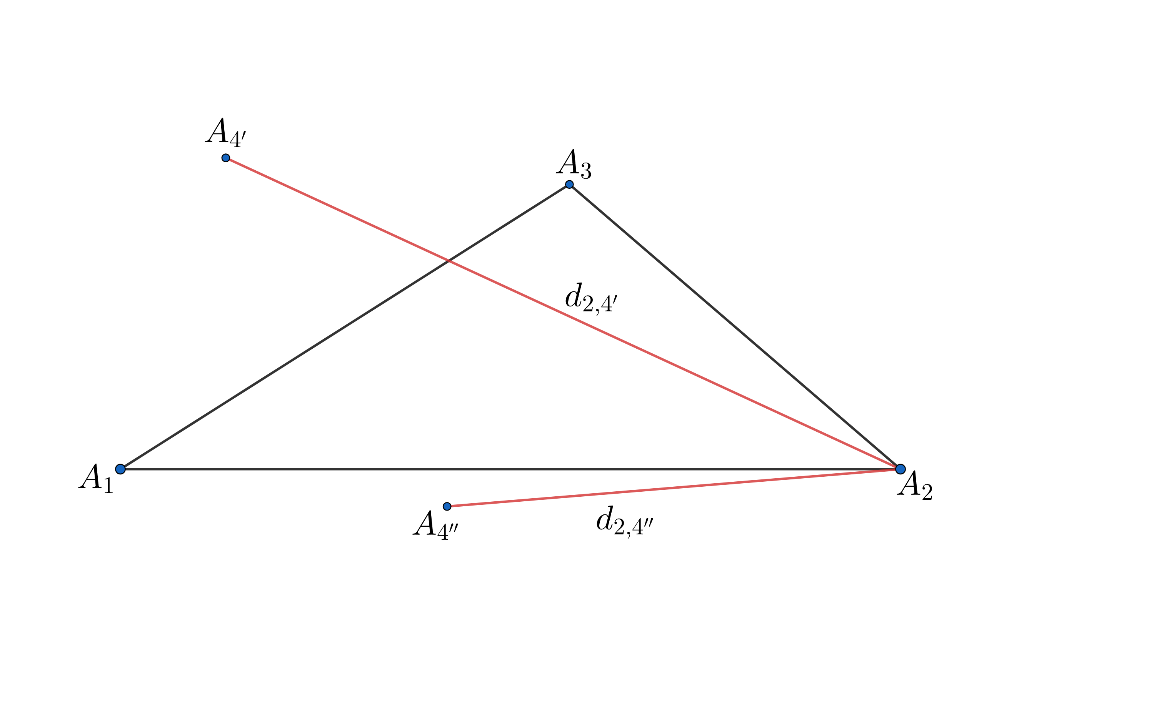


Рис. 1. Варианты построения четвёртых точек.

Далее измеряем расстояния от точки  до  и до  (). Точка, для которой полученное расстояние совпало с расстоянием , и будет подходящей, т.е.  или  (см. рис. 2).

  
Рис. 2. Выбор четвёртой точки.

**Теорема 1.** *Для двух и трех точек неискаженное изображение на плоскости всегда существует и единственно. Изображение четырех точек также единственно, если только оно существует, то есть выполнено условие V* = 0.

**Доказательство.** Возможность построения и его единственность были продемонстрированы при изложении вариантов алгоритма выше. Рассмотрим 4 точки. Условие  обеспечивает возможность неискаженного построения (лемма 1), а то, что оно единственно, сразу же вытекает из того, что, согласно результату из [13], при произвольном неискаженном построении всегда получаются равные тетраэдры, а любые два равных тетраэдра можно перевести один в другой с помощью сдвигов и симметрий трехмерного пространства.

Можно порассуждать и иначе. Рассматривая алгоритм построения для четырех точек, видим, что два решения возникают лишь при условии, что подойдут обе построенные на последнем шаге точки, т.е. . Отсюда следует, что точка  лежит на срединном перпендикуляре к , а это прямая  То есть точка  расположена либо на отрезке  , либо за его пределами. Но  перпендикулярна прямой . Следовательно, эти точки симметричны ей, то есть решение одно, поскольку одно из них переходит в другое при симметрии относительно этой прямой. Теорема доказана.

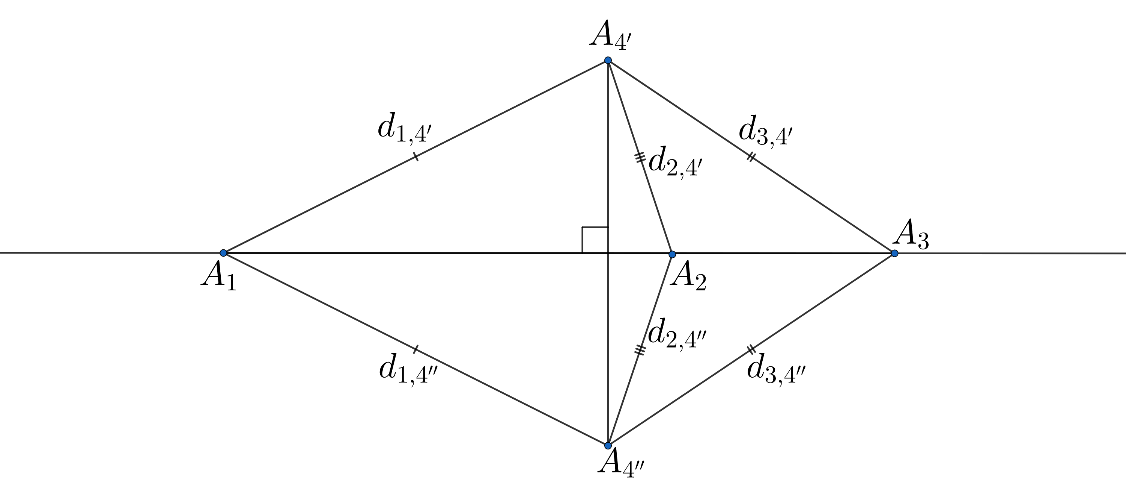


Рис. 3. Возможные два решения для четвёртой точки

Оказывается, если неискаженное построение для пяти точек возможно, то оно также единственно. Проверим это.

Пусть четыре точки уже построены. Аналогично построению четвёртой точки построим потенциальные пятые точки  пересечением окружностей в центрах  и . Обозначим расстояния от точек  и  до точки , как  и  соответственно. Аналогично для точки . Так же, как и для четырёх точек, если решение есть, то это одна из точек  или Но здесь нужно проверить совпадение расстояния  с исходным  и  с Точка, для которой расстояния совпали, и будет искомой.

Два решения могут возникнуть в следующих условиях:

1) Подошли обе построенные на последнем шаге точки для  и , т.е.  и . Отсюда следует, что точки  и  лежат на срединном перпендикуляре к , а это прямая То есть точки  и  расположены либо на отрезке , либо за его пределами. Но  перпендикулярна прямой . Следовательно, эти точки симметричны ей, то есть решение одно, поскольку одно из них переходит в другое при симметрии относительно этой прямой.

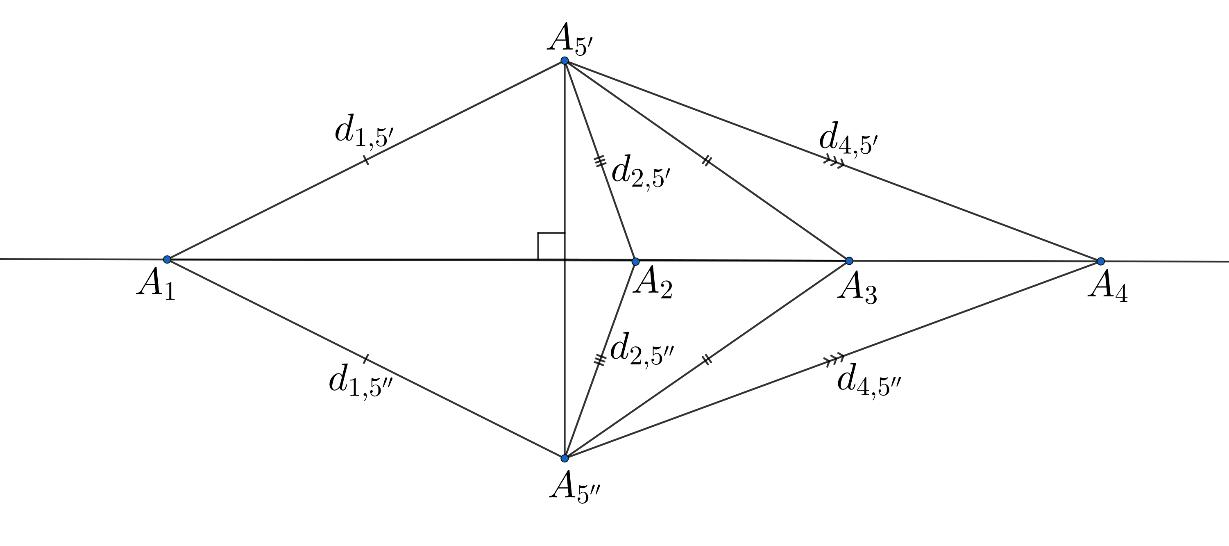


Рис. 4. Возможные два решения для пятой точки – 1 вариант

2) Для одной точки подошли обе построенные на последнем шаге пятые точки, а для другой нет. Пусть обе точки подошли только для , т.е. . Значит, измеряем расстояния от точки  до  и до  . Точка, для которой полученное расстояние совпало с расстоянием , и будет подходящей, т.е.  или  (см. рис. 5). То есть, вновь решение одно.

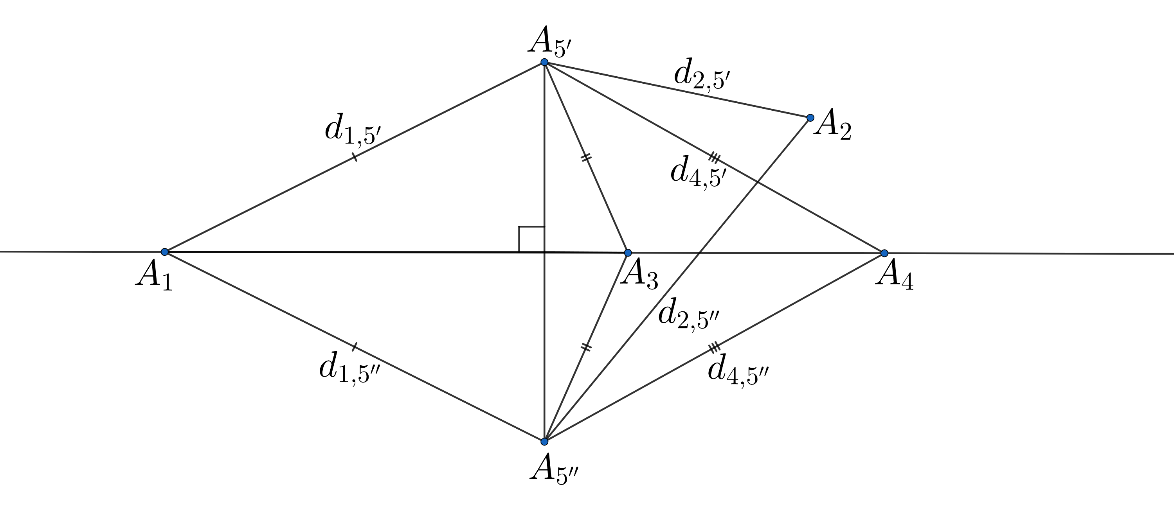
**

Рис. 5. Возможные два решения для пятой точки – 2 вариант

**Теорема 2.** *Пусть заданы попарные расстояния для множества точек, состоящего не менее, чем из 5 элементов. Если любые 4 из имеющихся точек можно изобразить на плоскости без искажения расстояний между ними, то можно изобразить с этим условием все точки множества. При этом неискаженное изображение этих точек единственно.*

Заметим, что если для некоторого количества (четырех или более) точек неискаженное изображение возможно, то из единственности этого изображения следует, что можно считать наши точки лежащими в одной плоскости. Очевидно, верно и обратное.

Теперь покажем, что все наши точки лежат в одной плоскости. Пусть сначала нашлись 3 точки из них, не лежащие на одной прямой. Пусть это . Условие принадлежности одной прямой через соотношение попарных расстояний - , следовательно, его нетрудно проверить без построения чертежа. Тогда, взяв произвольную четвертую точку, изобразим  в плоскости . Покажем, что в этой плоскости лежат и все остальные точки. Действительно, при выборе любой пятой точки по условию существует плоскость  значит, в частности, 

Но  тоже проходит через  поэтому можно утверждать, что  поскольку  не лежат на одной прямой. Значит, и  лежит в плоскости .

Пусть  лежат на одной прямой, добавим к ним точку. Через любые две точки проходит прямая и притом единственная, поэтому прямые , ,  лежат на . Значит, если любые 3 из наших точек лежат на одной прямой, то и все они лежат на этой прямой. Но это автоматически означает, что все они лежат на одной плоскости.

Единственность изображения вытекает, например, из следующей леммы, заимствованной из [14].

**Лемма 2.** *Даны четыре точки на плоскости: A, B, C и D; A, B и C не лежат на одной прямой. Пусть E такая точка, что AE=AD, BE=BD и CE=CD. Тогда E обязательно совпадает с D.*

**Доказательство.** *AE=AD* и *BE=BD*; значит, если *E* и *D* не совпадают, то они симметричны относительно *AB*. Аналогично, если *E* и *D* не совпадают, то они симметричны относительно *AC* (и относительно *BC*). Так как две точки не могут быть одновременно симметричны и относительно *AB*, и относительно *AC*, то, значит, *E=D.*

Для создания компьютерной программы понадобятся аналитические выражения для координат строящихся точек. Без ограничения общности допустим, что точка  помещается в начало координат, то есть имеет координаты (0, 0). Так можно считать поскольку иначе ее можно просто передвинуть в эту точку (и все последующую конструкцию вместе с ней) с помощью параллельного переноса. При этом в силу сделанных ранее замечаний решение задачи от этого не изменится.

Точка  может быть расположена на оси абсцисс на расстоянии  от точки с координатами (0, ). Для нахождения третьей точки нужно найти точку пересечения окружностей c центрами в  и . Нужные координаты получаем, решая систему из двух уравнений относительно переменных :



Раскрыв скобки во втором уравнении, вычитаем первое уравнение из второго.



Выражаем  из второго уравнения:

Подставляя в первое уравнение, найдём :



Поскольку мы уже знаем, что возможно лишь единственное решение, выберем только положительное значение для *y*.



Таким образом, мы получим координаты точки .

Для нахождения координат четвёртой точки решим систему относительно :



Раскроем скобки во втором уравнении и вычтем первое уравнение из второго:





Обозначим правую часть равенства за . Выразим  и подставим его в первое уравнение:





Получается квадратное уравнение:







Получаем две точки. Для выбора нужной, сверяем расстояние  c исходным. Выбираем ту из точек, для которой они одинаковы.

Итак, располагая на входе алгоритм попарными расстояниями, следует проверить условие . Если оно не выполнено, то отказываемся от построения изображения. Иначе можно вычислить координаты точек, используя (4 – 7) и по ним изображение построить.

## 2.3. Методика снижения искажений

В случаях, когда неискаженное изображение построить невозможно, то есть  предлагается в качестве первого (предварительного) шага перейти от второго способа задания объектов к первому, т.е. определить их координаты в некотором, пространстве, возможно, имеющем высокую размерность. Итак, если изображаемые объекты с самого начало были заданы вторым способом (через расстояния между ними), то выполним предварительный шаг, решая матричное уравнение (2). Если же первоначально наши объекты были заданы своими координатами, то предварительный шаг опускаем, используя уже известные координаты.

**Лемма 3.** Пусть объекты  заданы в *q*-мерном пространстве. Если произвольные четыре объекта из них можно изобразить без искажений на плоскости, то все объекты лежат в одной плоскости в исходном *q*-мерном пространстве.

**Доказательство.** Выберем произвольные четыре объекта  Если мы можем изобразить на плоскости без искажений эти объекты, тогда тетраэдр  плоский.

Теперь возьмём три точки  и произвольную четвёртую точку  из набора  который потенциально может быть изображен на плоскости без искажения. Тогда тетраэдр  тоже плоский. Таким образом, плоскость, содержащая точки  является общей для обоих плоских тетраэдров  и . Поскольку точка  была выбрана произвольно из набора  это означает, что все точки  лежат в одной плоскости.

Теперь путем последовательного перебора выберем наибольшее количество объектов  из заданных, лежащих в одной плоскости. Построим проекции остальных объектов на эту плоскость. Получившийся плоский рисунок объявляем результатом работы нашего алгоритма.

Дадим детальное описание построения проекций на языке формул.

Из набора точек  выберем два неколлинеарных вектора . Они определяют плоскость  Обозначим их как  Будем считать, что  - начало новой системы координат на  Направим первый орт по . Тогда



Нужно найти второй орт  который будет перпендикулярен  и лежать в плоскости  Для этого ортогонализуем систему векторов  Орт  параллелен вектору  Чтобы найти проекцию вектора  на вектор  воспользуемся  где  - угол между  и  Тогда вектор  Полученный вектор обозначим



Таким образом, второй орт



Теперь, чтобы найти координаты в новой системе координат для всех точек, можно использовать базис  Пусть  — это координаты точки  в этом базисе. Тогда



При этом формулы (11) верны и в том случае, когда выбирается точка не из числа лежащих на плоскости. Просто в этом случае *x*, *y* будут координатами проекции этой точки на нашу плоскость.

## 2.4. Основной алгоритм

Используя результаты предыдущих трёх разделов настоящей главы, можно теперь сформулировать основной алгоритм работы.

На входе – число объектов и матрица попарных расстояний между ними.

Шаг 1. Проверка числа объектов. Если будет введены данные только одного объекта, выведется «Число объектов должно быть не менее 2», и повторяем просьбу о вводе. Если два или три к шагу 3, иначе к шагу 2.

Шаг 2. Проверка возможности построения. Перебираем все возможные четвёрки точек и проверяем по формуле (2), что *V = 0.* Если хотя бы один раз это не выполняется, то находим наибольшее число объектов, для которых выполняется это условие, запоминая эти объекты, переходим к шагу 3 с сообщением «Неискаженное изображение построить невозможно». Иначе к шагу 4.

Шаг 3. Из объектов, которые запомнили выбираем точки вычисляем векторы  По формулам (8 - 11) находим окончательные координаты точек и выходим из алгоритма.

Шаг 4. Строим две точки с координатами (0, 0) и (0, ). Все ли точки построены? Если да – выход из алгоритма, иначе к шагу 5.

Шаг 5. Строим третью точку, определяя её координаты по формулам (3) и (4). Все ли точки построены? Нет – запоминаем основную 3-конструкцию, т.е. координаты первых трех построенных точек, и к шагу 6.

Шаг 6. Берем очередную точку и строим её относительно основной 3-конструкции. Вычисляем координаты по формулам (5) и (6). Получаем две потенциально возможные точки  и . Чтобы выбрать нужную подсчитаем расстояния между  и . Аналогично для . Нужная точка – та, у которой расстояние совпало с исходным . Все ли точки построены? Если да – выход из алгоритма, иначе – повторяем шаг 6.

# Глава 3. Практическое применение алгоритма

## 3.1. Описание компьютерной программы

Алгоритм из предыдущей главы был реализован в виде компьютерной программы visualization.py. Программа написана на языке Python 3.12.2. Для корректной работы программы необходимо наличие интерпретатора Python на компьютере пользователя, а также библиотек numpy и matplotlib.

Для работы программы матрицу попарных расстояний нужно представить в виде текстового файла с названием data.txt. Разделителем целой и дробной части служит точка. Значения расстояний в строке должны быть отделены табуляцией. Файл нужно разместить в одной папке с программой.

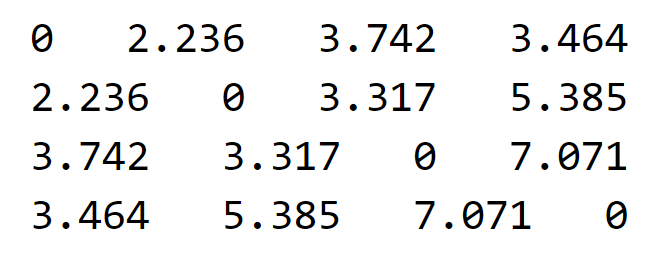


Рис. 6. Пример файла в формате .txt

## 3.2. Пример применения к данным

Для иллюстрации работы программы в случае неискаженного изображения в плоскости трехмерного пространства, заданной уравнением  возьмём 6 точек, тогда неискаженное их изображение будет заведомо возможно.

|  |  |  |  |
| --- | --- | --- | --- |
|  | *x* | *y* | *z* |
|  | 0 | 0 | 1 |
|  | 1 | 0 | 3 |
|  | 2 | 3 | 2 |
|  | -2 | -2 | -1 |
|  | 2 | 0 | 5 |
|  | 1 | 2 | 1 |

Табл. 1. Координаты точек в примере 1

По этим данным вычислим таблицу попарных расстояний и запустим программу. На выходе получаем координаты точек и конечное изображение:

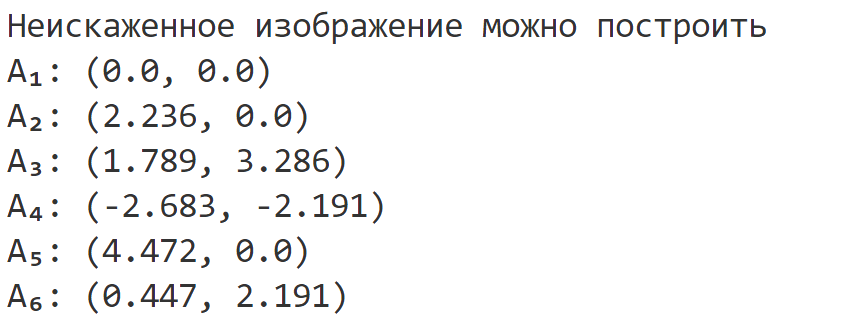


Рис. 7. Вывод в консоль результатов работы программы для первого примера

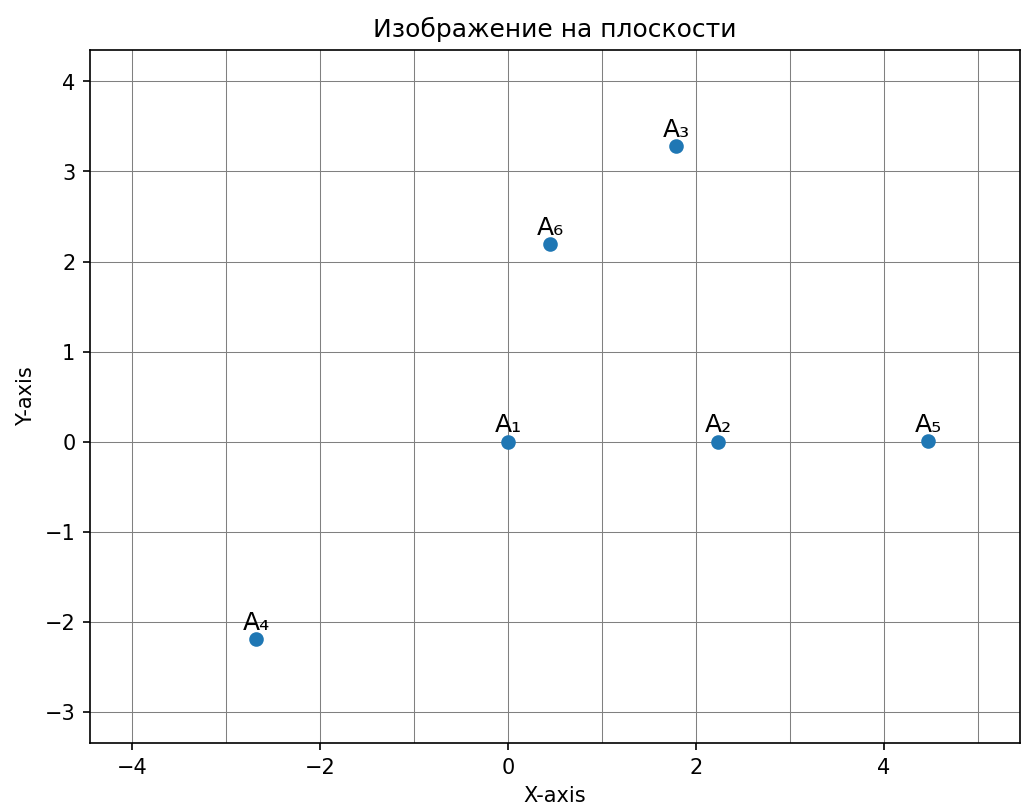


Рис 8. Вывод итогового изображения для первого примера

Чтобы продемонстрировать работу алгоритма в случае, когда неискаженное изображение невозможно, изменим координаты исходных точек так, чтобы не все точки лежали в одной плоскости. Для этого изменим координаты точек 

|  |  |  |  |
| --- | --- | --- | --- |
|  | *x* | *y* | *z* |
|  | 0 | 0 | 1 |
|  | 1 | 0 | 3 |
|  | 2 | 3 | 2 |
|  | -2 | -2 | -1 |
|  | 2 | 0 | 4 |
|  | 1 | 2 | 0 |

Табл. 2. Координаты точек в примере 2

Также по этим данным вычислим таблицу попарных расстояний и запустим программу

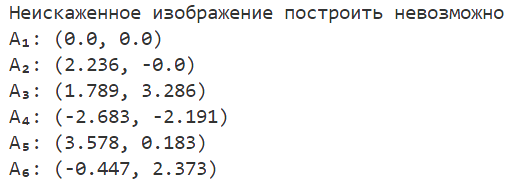


Рис. 9. Вывод в консоль результатов работы программы для второго примера

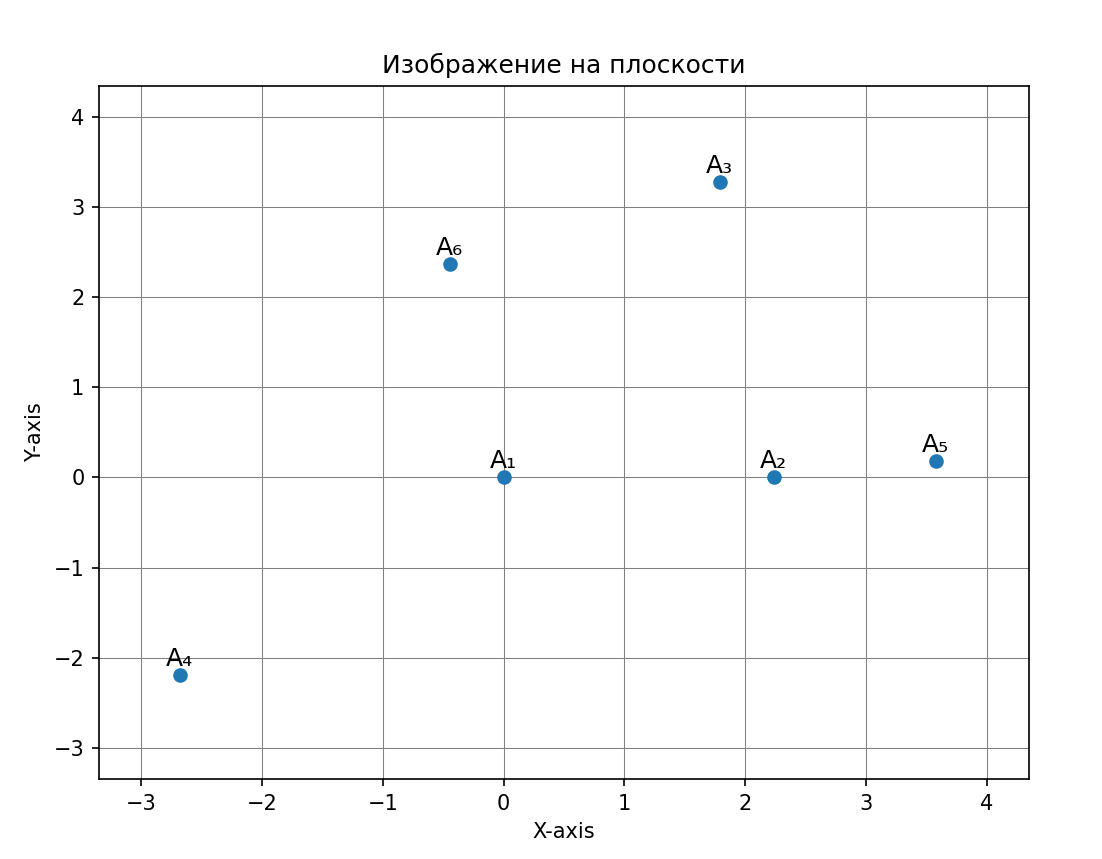


Рис 10. Вывод итогового изображения для второго примера

## 3.3. Сравнение с классическими линейными алгоритмами

К данным из предыдущего пункта применим метод многомерного шкалирования:

|  |  |  |
| --- | --- | --- |
|  | *x* | *y* |
|  | 0.826 | -0.836 |
|  | -1.282 | -0.143 |
|  | 0.033 | 2.843 |
|  | 2.816 | -3.667 |
|  | -3.448 | 0.383 |
|  | 1.054 | 1.419 |

Табл. 3. Координаты в двумерном пространстве, полученные методом многомерного шкалирования для первого примера

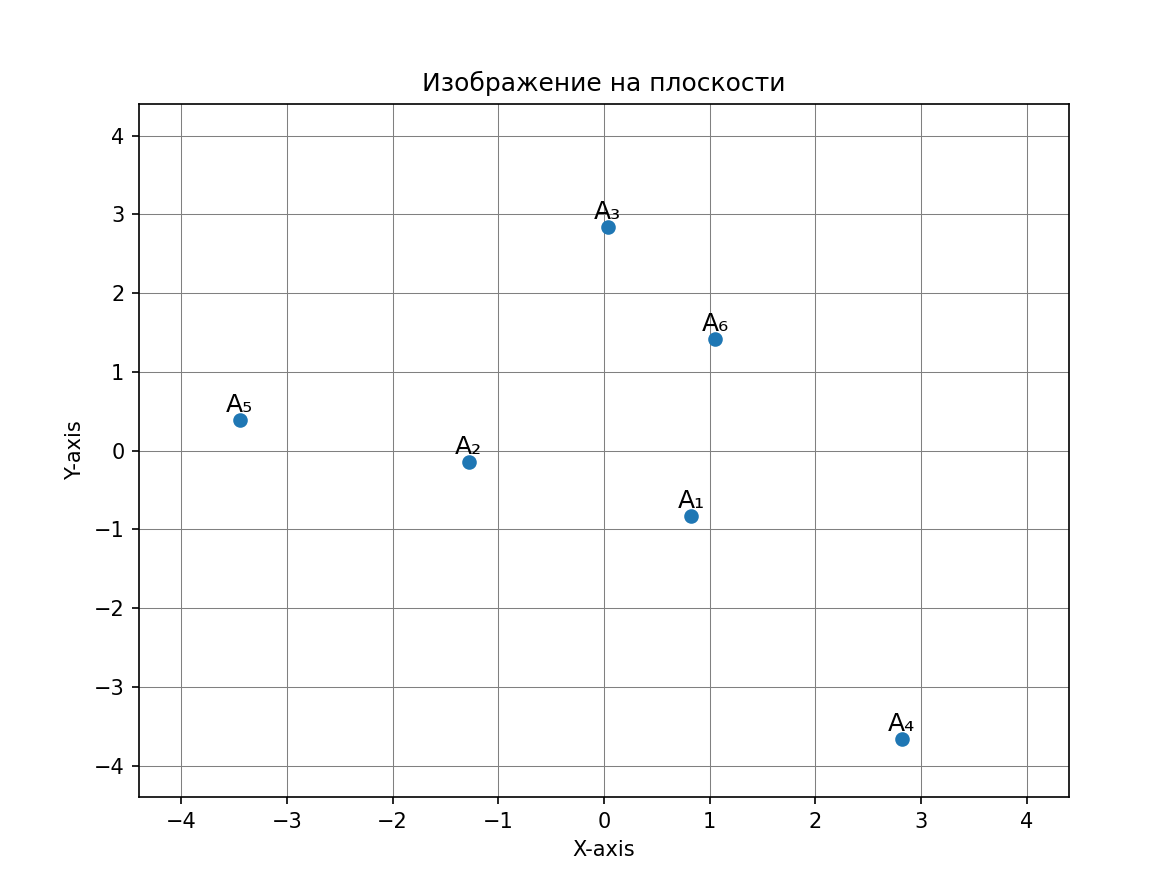


Рис 11. Изображение, полученное методом многомерного шкалирования для первого примера

Можно увидеть, что если наше изображение отобразить симметрично относительно вертикали и немного перевернуть влево, то оно будет очень похоже на изображение, полученное методом многомерного шкалирования.

Применим метод главных компонент к трёхмерным координатам и получим координаты в двумерном пространстве:

|  |  |  |
| --- | --- | --- |
|  | *x* | *y* |
|  | -0.662 | 0.073 |
|  | 0.312 | -0.599 |
|  | 1.361 | 0.978 |
|  | -2.66 | -0.081 |
|  | 1.287 | -1.271 |
|  | 0.362 | 0.9 |

Табл. 4. Координаты в двумерном пространстве, полученные методом главных компонент для первого примера

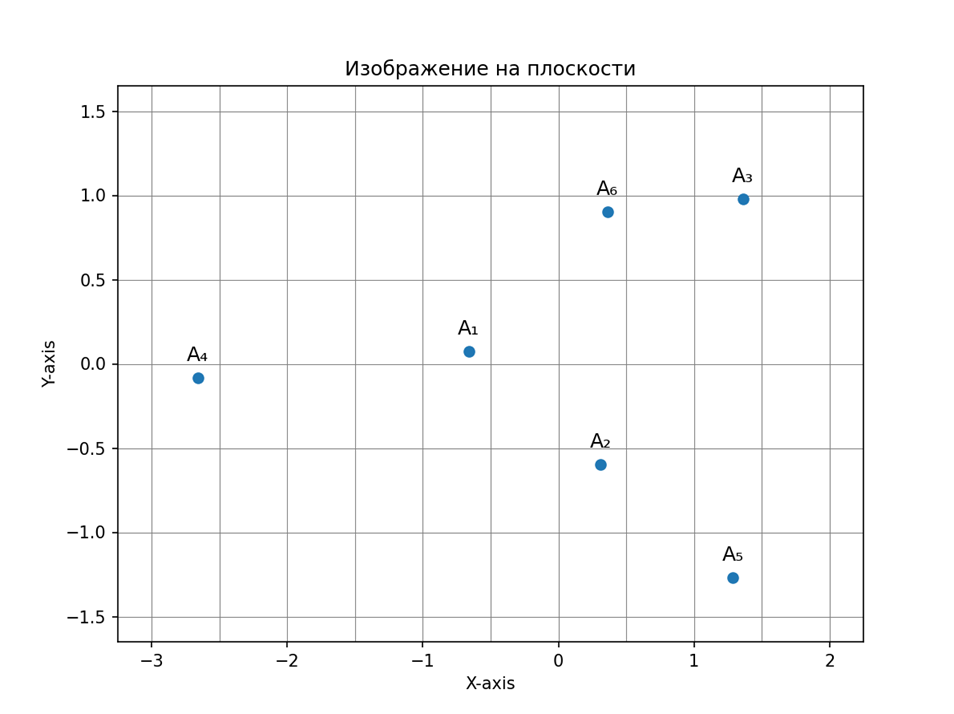


Рис 12. Изображение, полученное методом главных компонент для первого примера

Чтобы сравнить полученные при помощи различных методов результаты, для всех вариантов решения вычислим стресс-критерии по формуле , где  – реальное расстояние между *i*-м и *j*-м объектами, а  – расстояние между их образами. Величина этого критерия позволяет оценить степень искаженности расстояний на изображении. Для рассматриваемого в курсовой работе основного метода этот критерий, очевидно, равен 0.

Для метода главных компонент стресс-критерий равен 23.622, а у метода многомерного шкалирования он составляет 0.005.

Сравнив с реальными попарными расстояниями, можно заметить, что оба метода исказили расстояния между точками. Наш алгоритм эти расстояния изобразил точно, а, следовательно, значение стресс-критерия для точек с рис. 8 равен 0.

Теперь применим метод многомерного шкалирования для второго примера, когда неискаженное изображение будет невозможно:

|  |  |  |
| --- | --- | --- |
|  | *x* | *y* |
|  | 0.228 | 0.683 |
|  | -1.177 | 0.53 |
|  | -0.714 | -2.764 |
|  | 3.619 | 2.892 |
|  | -2.569 | 0.329 |
|  | 0.613 | -1.67 |

Табл. 5. Координаты в двумерном пространстве, полученные методом многомерного шкалирования для второго примера

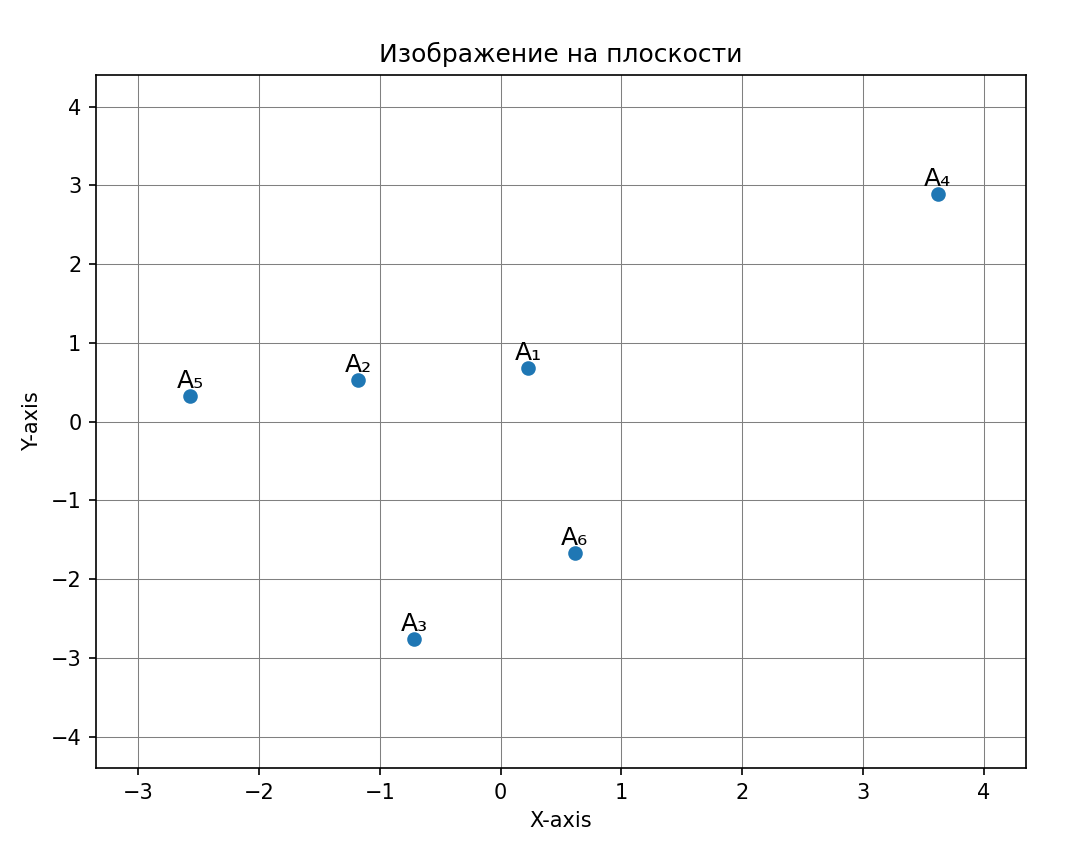


Рис 13. Изображение, полученное методом многомерного шкалирования для второго примера

Сравнив значение стресс-критерия для многомерного шкалирования 0.011 и нашего алгоритма 0.008, можно убедиться, что наш алгоритм расстояния между точками изобразил точнее.

# Заключение

Основной теоретический результат выпускной квалификационной работы – это выявление необходимого и достаточного условия для возможности неискаженного изображения данных. Также дано доказательство единственности неискаженного изображения для всех точек. Предложен метод построения наименее искаженных изображений в случаях, когда неискаженное изображение построить невозможно.

Главным практическим результатом является компьютерная программа на языке Python, реализующая разработанный алгоритм. Произведено сравнение с классическими линейными алгоритмами.

Таким образом, цель работы достигнута, все поставленные во введении задачи решены.

Результаты работы докладывались в 2024 году на студенческой конференции «Мой выбор – наука». Доклад был признан лучшим по секции и удостоен диплома. Частично результаты ВКР были опубликованы в [15].

# Библиографический список

1. Паклин Н. Б., Орешков В. И. Визуализация данных. // Бизнес-аналитика. От данных к знаниям. — 2-е изд. — СПб.: Питер, 2013. — С. 173–210.
2. Mohammed L. T., Al Habshy A. A. and El Dahshan K. A. Big Data Visualization: A Survey, 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). Ankara, Turkey, 2022. – P. 1–12.
3. Healy, Kieran. Data Visualization: A Practical Introduction. Princeton University Press, 2018. – 272 p.
4. Liu, S., Maljovec, D., Wang, B., Bremer, P.-T., & Pascucci, V. Visualizing High-Dimensional Data: Advances in the Past Decade. IEEE Transactions on Visualization and Computer Graphics, 23(3), 2017. – P. 1249–1268.
5. Bandalos D. L., Boehm-Kaufman M.R. Four common misconceptions in exploratory factor analysis. In: Lance C. E., Vandenberg R. J. Statistical and Methodological Myths and Urban Legends: Doctrine, Verity and Fable in the Organizational and Social Sciences. London: Taylor & Francis, 2008. – P. 61–87.
6. Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика: Классификации и снижение размерности. — М.: Финансы и статистика, 1989. – 608 с.
7. Gorban A. N., Kegl B., Wunsch D., Zinovyev A. Y. (Eds.). Principal Manifolds for Data Visualisation and Dimension Reduction (Lecture Notes in Computational Science and Engineering, 58). New York: Springer, 2007. – 364 p.
8. Мокеев В. В., Метод главных компонент в задачах экономического анализа и прогнозирования: монография. Челябинск: Изд-во ЮУрГУ, 2009. – 168 c.
9. Torgerson W. S. Multidimensional scaling: I. Theory and method. // Psychometrika, – v. 17, (1952). – P. 401–419.
10. Толстова Ю. Н. Основы многомерного шкалирования. — М.: КДУ, 2006. – 160 c.
11. Kohonen T. Self-Organizing Maps (Third Extended Edition). New York: Springer, 2001. – 501 p.
12. Журенков О. В. Методика применения карт Кохонена для выделения линии уреза воды по спутниковым данным. //Известия АлтГУ. – 2016. – №1(89). – С. 111 – 116.
13. Сабитов И. Х. Объёмы многогранников. М.: МЦНМО, 2002. – 18 с.
14. Гервер М. Л. Интернет-проект «Задачи» [Электронный ресурс]. Режим доступа: https://www.problems.ru/view\_problem\_details\_new.php?id=73780&amp;into\_basket=73780. Дата обращения 03.03.2024.
15. Калинкин А.А. Неитерационный алгоритм визуализации многомерных данных // Труды семинара по геометрии и математическому моделированию, 2023, – № 9. – С. 105-111.

# Приложение

В удалённом репозитории https://github.com/alexaxax/bachelor\_work расположены следующие файлы:

* ВКР.docx – текст выпускной квалификационной работы;
* visualization.py – файл, содержащий исходный код программы, реализующий алгоритм (запускается при помощи интерпретатора Python);
* data.txt – пример файла ввода данных;
* samples.jpg – пример вывода.

### ПОСЛЕДНИЙ ЛИСТ ВКР

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись) (Ф.И.О.)*