МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)
КАФЕДРА МО ЭВМ

ОТЧЕТ

по лабораторной работе

по дисциплине «Компьютерная математика»

Тема: Повороты графа в пространстве

Афанасьев Н. С.

Студент гр. 0382 _____ Крючков А. М.

Преподаватели _____ Коптелов Я. Ю.

Санкт-Петербург

2022

Оглавление

Задание. 3
Выбранный метод выполнения задачи. 3
Построение графа. 5
Выбор «правильных» рёбер. 5
Матрицы и координатные векторы. 5
Поворот графа. 6
Решение обратной задачи. 6
Алгоритм Кабша. 7
Сингулярное разложение. 9
Вывод угла из матрицы поворота. 11
Вывод и визуализация результатов. 11
Проверка корректности восстановления графа. 15
Выводы. 17
Список использованных источников. 18

Задание.

Пусть дана информация о структуре молекулы в формате JSON из базы данных PubChem, например, молекула аспирина.

Программа должна распарсить JSON (только атомы с координатами и связи с кратностями, игнорируя заряды и прочую дополнительную информацию), построить в памяти граф молекулы (вершины = атомы, рёбра = связи) и определить рёбра, которые одновременно:

- одинарные (т. е. соответствуют одинарным, а не двойным и не тройным связям)
- не принадлежат циклам
- не являются "висячими", т. е. продолжаются с обоих концов

В вышеупомянутой молекуле аспирина таких рёбер четыре (если смотреть на двумерную картинку, в которой не показаны водороды); в трёхмерной картинке с водородами таких рёбер можно насчитать пять.

Далее, программа должна повернуть молекулу в трёхмерном пространстве вокруг каждого из найденных рёбер на случайный угол так, чтобы вращалась не вся молекула, а лишь "половина", лежащая по одну сторону от ребра. Координаты атомов при этом, конечно, изменятся, изменится форма молекулы.

Далее, программа должна забыть случайные углы, проанализировать получившиеся координаты атомов и найти такие углы вращения вокруг найденных связей, чтобы атомы встали на исходные места.

Выбранный метод выполнения задачи.

Существуют несколько возможных способов решения поставленной задачи, был выбран способ, включающий в себя алгоритм Кабша.

При решении прямой задачи (для получения входных данных), после получения графа и нахождения “правильных” рёбер, случайным образом выберем одну вершину. От этой вершины произведём обход графа для того, чтобы установить направление “правильных” рёбер от взятой вершины.

После того, как мы выбрали направление рёбер, перемешиваем список этих рёбер, а затем устанавливаем случайные углы от -π до π и поворачиваем граф. Значение углов забываем, однако запоминаем выбранную вершину. В итоге мы получаем набор изменённых вершин, где, как минимум, одна вершина не изменит свои координаты.

Для решения обратной задачи начнём обход графа в ширину, начиная с запомненной вершины. Таким образом, на каждом шаге обхода мы имеем возможность сравнивать тройки вершин исходного и изменённого графа (по одной вершине на 3-х последовательных уровнях BFS), причём координаты двух вершин совпадают с изначальными, а координаты третьей вершины отличаются от оригинала. Если мы сможем восстановить поворот проходимого ребра так, чтобы третья вершина встала на исходное положение, то мы сможем произвести обратный поворот графа вокруг этой вершины и продолжить обход уже изменённого графа. По окончании обхода, мы должны получить приближение к исходному графу.

Для определения угла поворота был использован алгоритм Кабша, для получения однозначного ответа, которому необходимо как раз минимум три точки. Сам алгоритм будет описан ниже.

Так как в данном решении на каждом шаге рассматриваются три точки, две из которых находятся на изначальном положении, то избавиться от условия как минимум одной стационарной точки не получиться. Если такой точки не будет, то получить её уже после поворота не получится, так как для восстановления координат случайной точки графа уже понадобится искать нужный поворот для конкретных рёбер. Если же восстанавливать координаты этой точки не через вращение вокруг ребра, а через трансляцию, то мы нарушим структуру графа (длина рёбер изменится). Восстанавливать координаты точки, используя равенство длин всех рёбер, не получиться, так как этот факт неверен (длина химической связи обратно пропорциональна энергии связи).

Построение графа.

Структура молекулы получается из сервиса PubChem путём отправки HTTP GET запроса. Полученный JSON файл парсится и из него достаются необходимые данные. На основе этих данных в памяти строится граф: список координат вершин, словарь смежности вершин, словарь кратностей рёбер.

Выбор «правильных» рёбер.

Для выбора “правильных” рёбер ребро удовлетворяет следующим свойствам:

1) Вершины на концах не являются висячими
2) Порядок ребра равен 1
3) Ребро не принадлежит циклам

Для этого мы итерируемся по графу, представленному в виде списка смежности. Для проверки первого условия мы смотрим количество элементов в списке смежности для каждой из вершины на концах. Порядок ребра (1 или более) берётся из списка смежности *order,* который получен из JSON файла. Для того, чтобы проверить нахождения ребра в цикле ребро удаляется из графа. Если путь между вершинами можно найти, то ребро принадлежит какому-то циклу. Затем ребро возвращается на место.

Такие рёбра заносятся в список и записываются в поле графа для дальнейшего условия.

Матрицы и координатные векторы.

Для выполнения работы были реализованы простые классы Matrix и CVector для работы с матрицами и координатными векторами. Для обоих классов определены операции сложения и умножения. У матриц есть такие необходимые для работы методы, как транспонирование и поиск определителя, а у векторов – получение нормы вектора.

Поворот графа.

Один из способов описания поворотов в пространстве – матрицы поворота, которые представляют из себя ортогональные матрицы с единичным определителем. При умножении вектора на такую матрицу изменяются координаты, однако длина вектора остаётся прежней.

Для выполнения задачи необходимо уметь поворачивать точки в графе относительно одного из рёбер. Для этого необходимо рассчитать единичный вектор из вектора ребра и вычислить матрицу поворота, затем необходимо переместить вершины так, чтобы конечная точка ребра лежала в начале системы координат, после чего необходимо применить к вершинам матрицу поворота и переместить вершины обратно:

$\overline{x'} = M(\overline{x} - \overline{t}) + \overline{t}$, где $\overline{x'}$ и $\overline{x}$ – конечные и начальные координаты точки, $\overline{t}$ – трансляция, а M – матрица поворота.

Сама матрица поворота вокруг произвольной оси составляется следующим образом:

$$M(\overline{v}, \theta) = \begin{pmatrix} cos\theta + versin\theta x^2 & versin\theta yx - sin\theta z & versin\theta zx + sin\theta y \\ versin\theta yx + sin\theta z & cos\theta + versin\theta y^2 & versin\theta yz - sin\theta x \\ versin\theta zx - sin\theta y & versin\theta yz + sin\theta x & cos\theta + versin\theta z^2 \end{pmatrix}$$

Где $\overline{v}(x, y, z)$ – это произвольный единичный вектор, а θ – угол поворота.

Для выполнения работы были реализованы две функции: функция поворота конкретной точки вокруг конкретного ребра и функция поворота графа, в которую передаются список рёбер, вокруг которых необходимо произвести вращение, список углов, список вершин, и словарь, содержащий информацию о рёбрах.

Решение обратной задачи.

В начале решения обратной задачи нам известна структура графа, начальные и изменённые в результате поворота координаты вершин, а также вершина, с которой будут начинаться обход графа (и, следовательно,

направление рёбер, вокруг которых могло происходить вращение). Изначально нам неизвестно, вокруг каких именно рёбер и на какой угол происходило вращение, а также порядок вращений.

Всё решение основано на обходе графа в ширину, о котором велась речь выше, в пункте «Выбранный метод выполнения задачи». На каждом этапе рассматриваются три точки, координаты, одной из которых не совпадают с оригинальными координатами. Для этих трёх точек применяется алгоритм Кабша, результатом которого является матрица поворота. Зная единичный вектор для поворота (вычисляется из координат ребра) и матрицу поворота, можно вычислить угол вращения. Угол и ребро запоминаются, а сам граф вращается вокруг этого ребра на обратный угол, чтобы рассматриваемая точка встала на своё место, а остальные вершины изменили своё положение.

Нужно заметить, что, если в тройке вершин все вершины стоят на своих местах, то алгоритм Кабша и поворот не выполняются. Это важно, например, когда у одной из вершин несколько дочерних, и нет необходимости проворачивать все описанные действия ля каждой из них, так как поворот относительно ребра для них осуществлялся одинаковый.

Алгоритм Кабша.

Данный алгоритм предназначен для расчёта оптимальной матрицы поворота с минимизацией среднеквадратического отклонения для двух наборов точек. Сам алгоритм состоит из следующих этапов:

1. Получение входных матриц P и Q, которые представляют из себя координаты набора точек. Так как количество точек и размерность должны совпадать, то алгоритм должен проверять равенство размеров матриц. Так как в рассматриваемой задаче точки находятся в трёхмерном пространстве, то матрицы должны быть размера 3xN. Таким образом матрица должна иметь вид:

$$Q = \begin{pmatrix} x_1 & x_2 & x_n \\ y_1 & y_2 & y_n \\ z_1 & z_2 & z_n \end{pmatrix}$$

2. Трансляция координат. Для поворота обычно высчитываются центроиды координат точек и вычитаются из исходных координат, чтобы в результате центроиды обоих наборов совпадали с началом координат. Однако в случае данного решения задачи можно заранее передавать необходимый сдвиг, который будет совпадать с координатами второй точки. Данное суждение можно описать в виде графика (см. рис. 1).

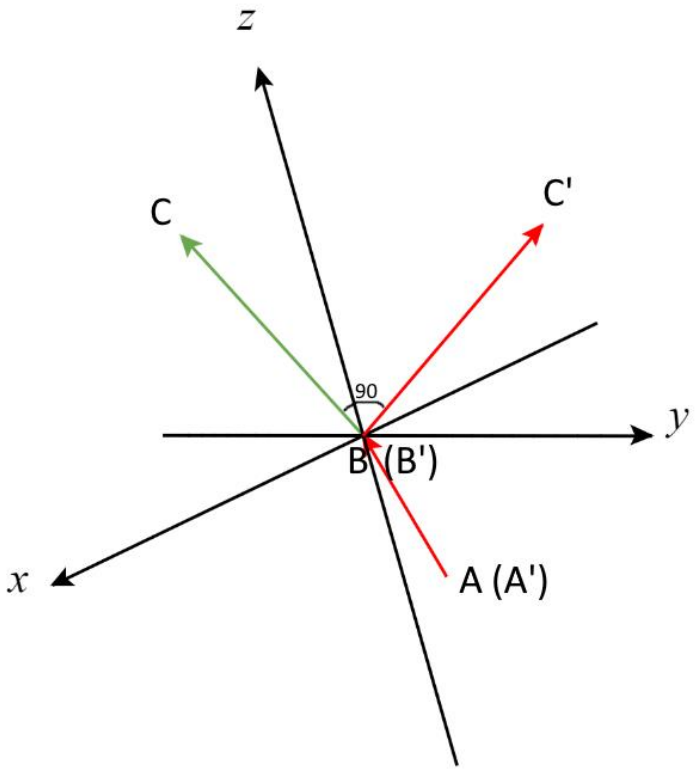


Рисунок 1 – Расположение рассматриваемой тройки точек

3. Вычисление ковариационной матрицы: $H = P^T Q$

4. Вычисление матрицы поворота. Для начала высчитывается сингулярное разложение ковариационной матрицы: $H = U\Sigma V^T$ (описание сингулярного разложения приведено в следующем пункте). Тогда матрица поворота представима как $R = VU^T$.

5. Обеспечение правосторонней систему координат. Если определитель полученной матрицы меньше нуля, то необходимо поменять знак у последнего столбца матрицы V и посчитать матрицу поворота сначала:

$$R = V\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}U^T$$

Сингулярное разложение.

Сингулярное разложение (SVD) – это разложение вида $A = U\Sigma V^T$, где A_{m*n} – входная матрица, U_{m*k} – матрица левых сингулярных векторов, V_{n*k} – матрица правых сингулярных векторов, а Σ_{k*k} – матрица сингулярных чисел. Геометрический смысл разложения представлен на рис. 2.

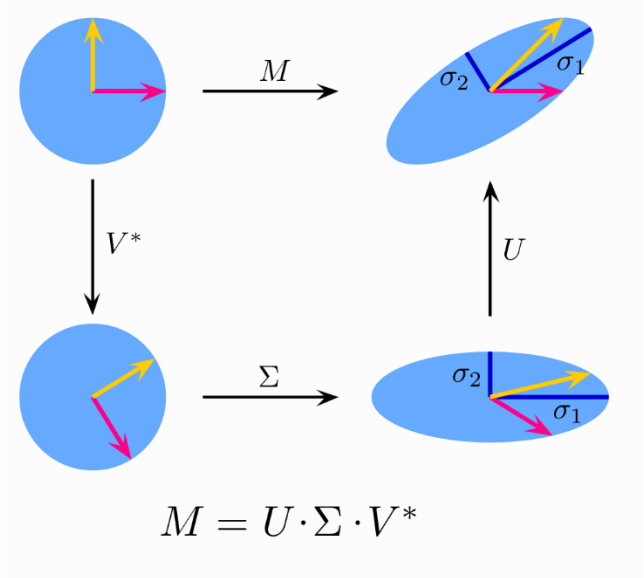


Рисунок 2 – Сингулярное разложение

Алгоритм поиска разложения состоит из следующих этапов:

1. Найти собственные числа матрицы A. Так как эта матрица может быть треугольной нужно рассматривать матрицу A^TA. Необходимо получить характеристический многочлен: $det(A^TA - \lambda I)$, корни которого и будут являться собственными числами.

2. Составить матрицу Σ. На диагоналях этой матрица расположены сингулярные числа в убывающем порядке. Сингулярные числа представляют из себя корни собственных чисел.

3. Найти собственные вектора. Получаются вектора путём решения системы вида $(A^TA - \lambda I)x = 0$, где λ – рассматриваемое собственное число.

4. Составить матрицу V из собственных векторов.

5. Найти матрицу U. Из начальной формулы можно вывести формулу для матрицы U: $U = AV\Sigma^{-1}$.

6. Транспонировать матрицу V, и разложение найдено.

Реализация такого алгоритма весьма затруднительна из-за расчёта собственных чисел и векторов, поэтому существуют другие реализации поиска. В рамках курса упоминался алгоритм простой итерации для поиска сингулярного разложения (power iteration method). Этот метод основан на итерации вида $b_{k+1} = \frac{Ab_k}{\|Ab_k\|}$, где A – исходная матрица, b_0 – случайный вектор. Итерация продолжается пока результат не сойдётся (разница меньше эпсилона; существует вариация алгоритма, которая рассчитывает необходимое число итераций). Найденный вектор будет являться доминирующим собственным вектором, из которого можно получить собственное и, следовательно, сингулярное число. Для получения следующего вектора мы должны удалить доминирующее направление из матрицы и повторить поиск наиболее доминирующего сингулярного значения: $A = A - \sigma_i U_i V_i^T$.

Этот метод проще для реализации, однако на практике он обычно не используется, так как имеет ряд ограничений. Во-первых, исходная матрица должна быть диагонализируемой. Во-вторых, каждый следующий собственный вектор для доминирующего собственного числа должен быть строго меньше предыдущего. В-третьих, выбранный случайным методом вектор должен иметь ненулевую компоненту в направлении собственного вектора, связанного с доминирующим собственным числом. Если эти ограничения не выполняются, то метод может не сойтись. В виду возможной недетерминированности и численной нестабильности, в работе использовалась готовая реализация сингулярного разложения (хотя описанный метод был также реализован).

Вывод угла из матрицы поворота.

Зная матрицу поворота и координаты границ ребра, можно высчитать угол поворота. Для начала приведём ребро в вид нормированного вектора: $\overline{v} = c_2 - c_1; \overline{v} = \frac{\overline{v}}{\|\overline{v}\|}$. Далее, из формулы матрицы вокруг произвольной оси, можно высчитать косинус и синус: $cos\theta = \frac{R_{1,1} - x^2}{1 - x^2}; sin\theta = \frac{yzversin\theta - R_{1,2}}{x}$. Стоит заметить, что знаменатели могут оказаться нулями, поэтому для корректного вычисления нужно будет видоизменять формулу (брать другие коэффициенты R и, соответственно, другие переменные). Значение угла можно получить через арккосинус с учётом знака полученного синуса.

Вывод и визуализация результатов.

Вывод осуществляется в консоль и файл. Сначала записывается таблица координат: начальный, изменённые и восстановленные. Затем приводится исходный список рёбер, вокруг которых происходил поворот, и углов, а затем приводится полученный список рёбер и углов. Выводятся только те найденные рёбра, значение угла поворота которых не меньше 0.01°. Пример вывод можно увидеть в следующем пункте.

Визуализация осуществляется в файл html. Была необходима библиотека, которая умеет создавать простейшую картинку в 3d. Была выбрана библиотека *plotly* за удобство просмотра результата. Сначала в 3d пространства заносятся координаты молекул (красные точки). Затем рисуются рёбра между(серые линии).

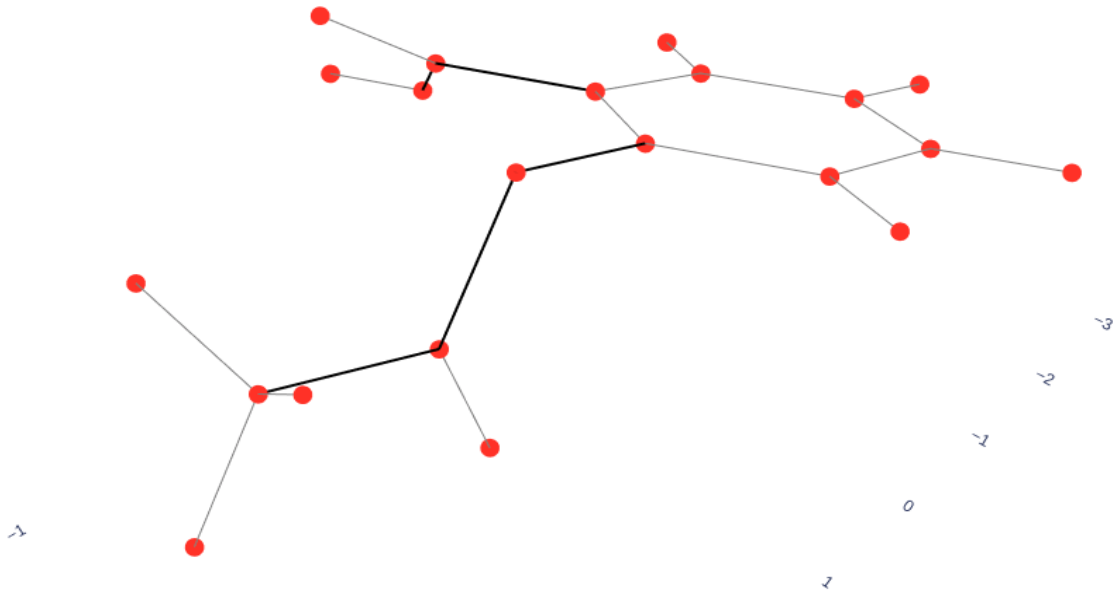


Рисунок 3 - Начальное состояние молекулы Аспирина

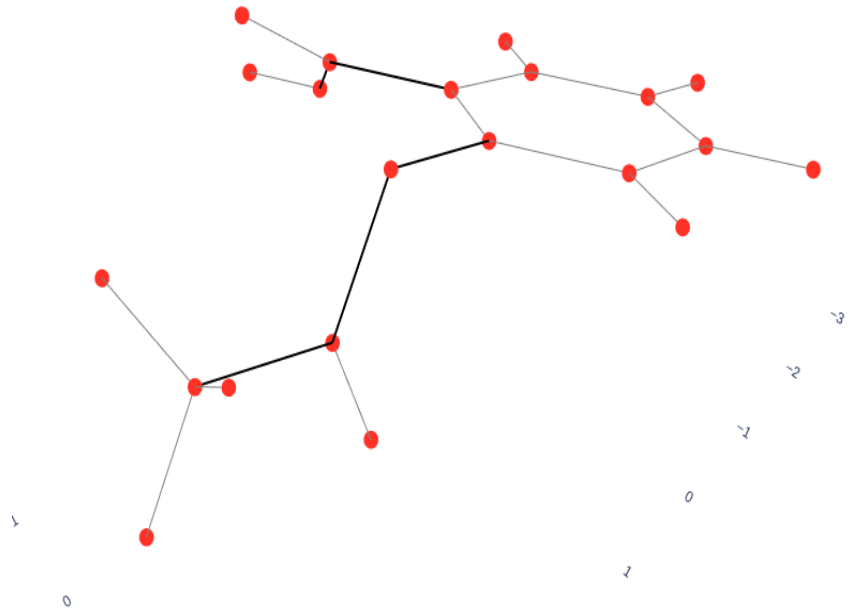


Рисунок 4 – восстановленное состояние молекулы аспирина

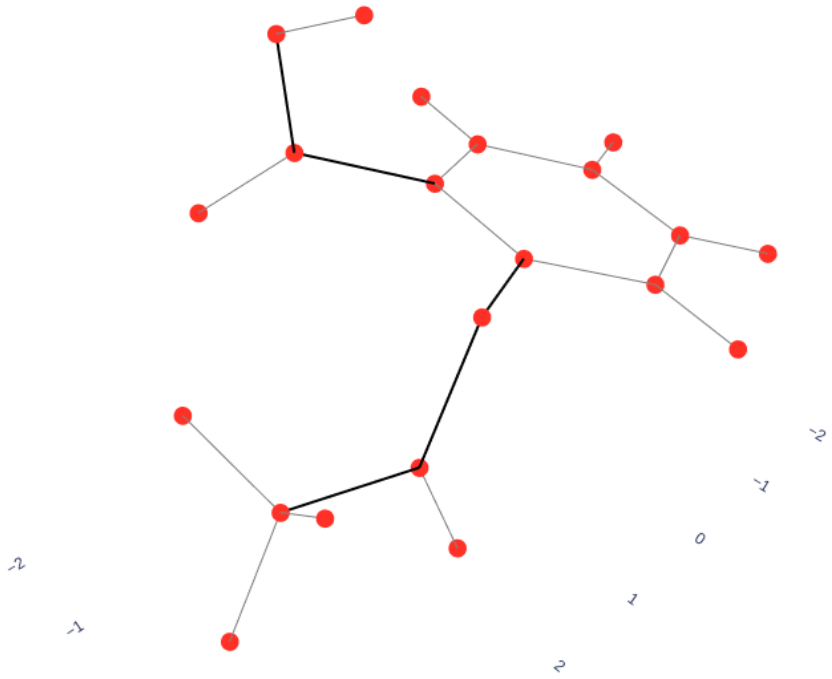


Рисунок 5 – Повернутое состояние молекулы аспирина

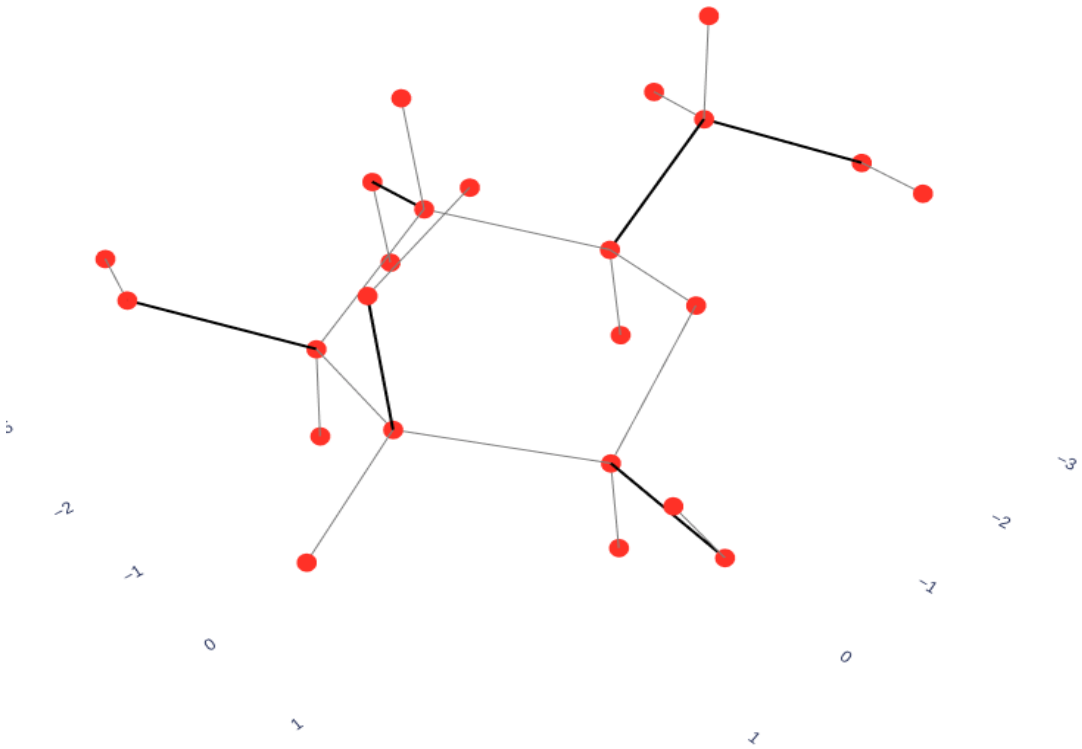


Рисунок 7 - d-mannose до поворота

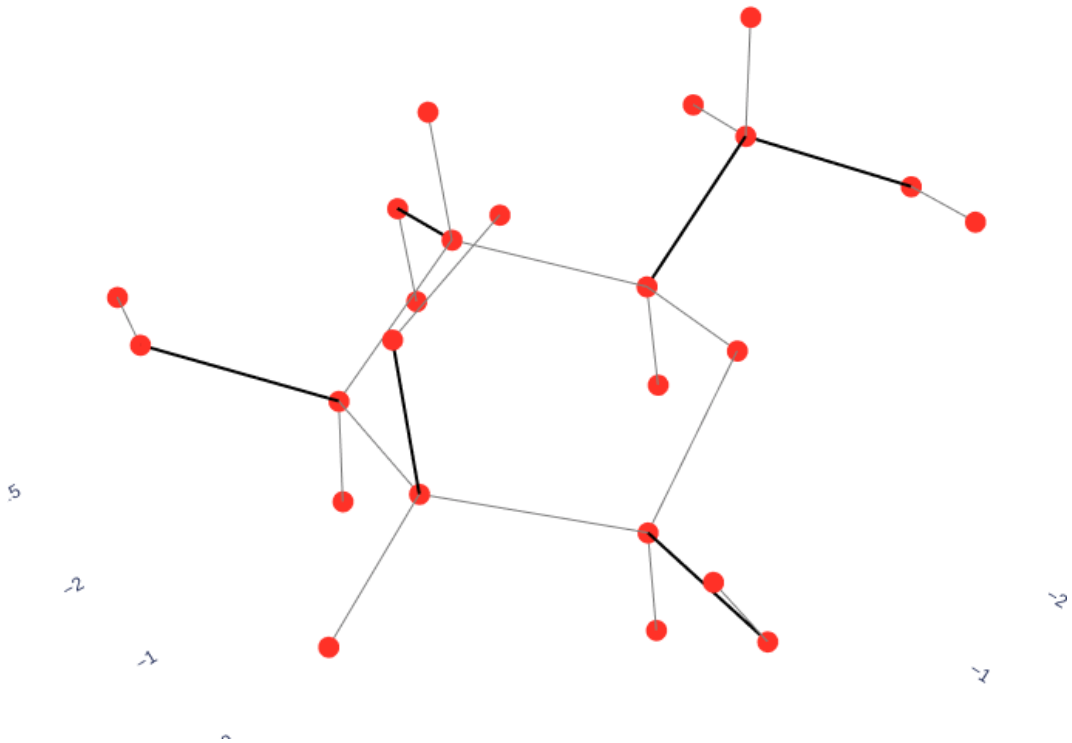


Рисунок 9 — d-mannose после восстановления

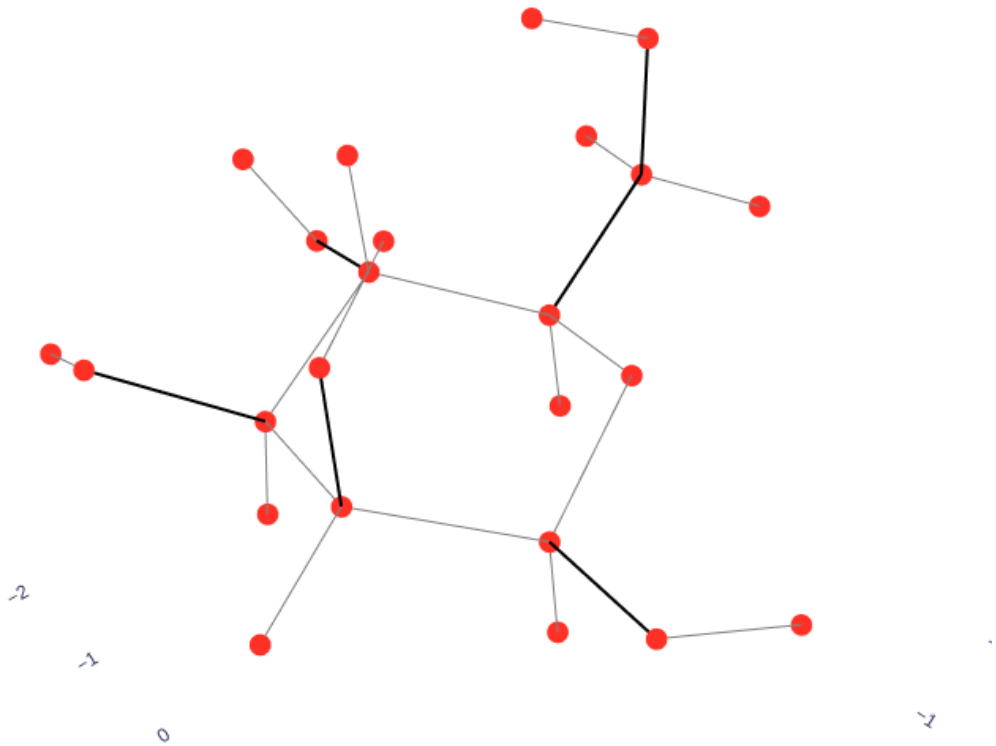
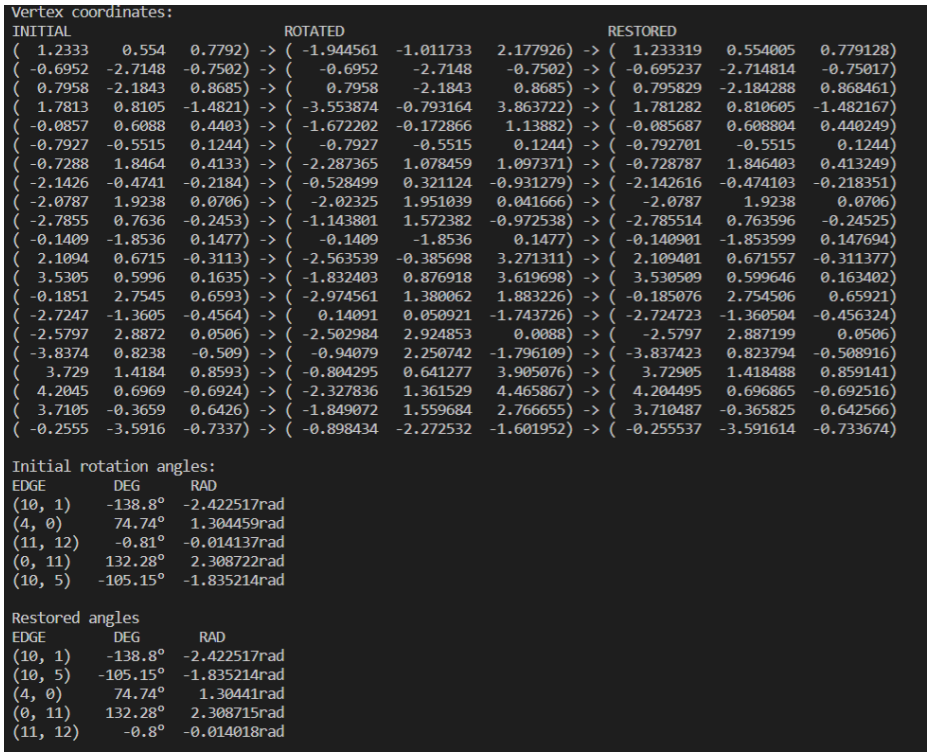


Рисунок 10 — d-mannose повернутое состояние

Проверка корректности восстановления графа.

Для проверки достаточно посмотреть на результаты выполнения программы и сравнить полученные координаты и углы.



```
Vertex coordinates:
INITIAL                          ROTATED                                  RESTORED
(  1.2333    0.554   0.7792) -> ( -1.944561  -1.011733   2.177926) -> (  1.233319   0.554005   0.779128)
( -0.6952  -2.7148  -0.7502) -> (   -0.6952    -2.7148    -0.7502) -> ( -0.695237  -2.714814   -0.75017)
(  0.7958  -2.1843   0.8685) -> (    0.7958    -2.1843     0.8685) -> (  0.795829  -2.184288   0.868461)
(  1.7813   0.8105  -1.4821) -> ( -3.553874  -0.793164   3.863722) -> (  1.781282   0.810605  -1.482167)
( -0.0857   0.6088   0.4403) -> ( -1.672202  -0.172866    1.13882) -> ( -0.085687   0.608804   0.440249)
( -0.7927  -0.5515   0.1244) -> (   -0.7927    -0.5515     0.1244) -> ( -0.792701    -0.5515     0.1244)
( -0.7288   1.8464   0.4133) -> ( -2.287365   1.078459   1.097371) -> ( -0.728787   1.846403   0.413249)
( -2.1426  -0.4741  -0.2184) -> ( -0.528499   0.321124  -0.931279) -> ( -2.142616  -0.474103  -0.218351)
( -2.0787   1.9238   0.0706) -> (  -2.02325   1.951039   0.041666) -> (   -2.0787    1.9238    0.0706)
( -2.7855   0.7636  -0.2453) -> ( -1.143801   1.572382  -0.972538) -> ( -2.785514   0.763596  -0.24525)
( -0.1409  -1.8536   0.1477) -> (   -0.1409    -1.8536     0.1477) -> ( -0.140901  -1.853599   0.147694)
(  2.1094   0.6715  -0.3113) -> ( -2.563539  -0.385698   3.271311) -> (  2.109401   0.671557  -0.311377)
(  3.5305   0.5996   0.1635) -> ( -1.832403   0.876918   3.619698) -> (  3.530509   0.599646   0.163402)
( -0.1851   2.7545   0.6593) -> ( -2.974561   1.380062   1.883226) -> ( -0.185076   2.754506    0.65921)
( -2.7247  -1.3605  -0.4564) -> (   0.14091   0.050921  -1.743726) -> ( -2.724723  -1.360504  -0.456324)
( -2.5797   2.8872   0.0506) -> ( -2.502984   2.924853     0.0088) -> (   -2.5797   2.887199    0.0506)
( -3.8374   0.8238   -0.509) -> (  -0.94079   2.250742  -1.796109) -> ( -3.837423   0.823794  -0.508916)
(   3.729   1.4184   0.8593) -> ( -0.804295   0.641277   3.905076) -> (   3.72905   1.418488   0.859141)
(  4.2045   0.6969  -0.6924) -> ( -2.327836   1.361529   4.465867) -> (  4.204495   0.696865  -0.692516)
(  3.7105  -0.3659   0.6426) -> ( -1.849072   1.559684   2.766655) -> (  3.710487  -0.365825   0.642566)
( -0.2555  -3.5916  -0.7337) -> ( -0.898434  -2.272532  -1.601952) -> ( -0.255537  -3.591614  -0.733674)

Initial rotation angles:
EDGE        DEG      RAD
(10, 1)    -138.8°  -2.422517rad
(4, 0)       74.74°   1.304459rad
(11, 12)     -0.81°  -0.014137rad
(0, 11)    132.28°   2.308722rad
(10, 5)    -105.15°  -1.835214rad

Restored angles
EDGE        DEG       RAD
(10, 1)    -138.8°  -2.422517rad
(10, 5)    -105.15°  -1.835214rad
(4, 0)       74.74°    1.30441rad
(0, 11)    132.28°   2.308715rad
(11, 12)      -0.8°  -0.014018rad
```

Рисунок 11 - Пример 1: молекула аспирина

```
Vertex coordinates:
INITIAL                          ROTATED                          RESTORED
( -0.2837   0.2194  -0.0834) -> ( -1.188275   0.891339   -2.46852) -> (  -0.28356   0.218946  -0.082967)
( -3.4068   0.4826     -0.6) -> (   -3.4068     0.4826      -0.6) -> ( -3.406705   0.481527  -0.599606)
( -2.5385   2.4043   0.2557) -> (   -2.5385     2.4043    0.2557) -> ( -2.538627   2.403709   0.255235)
( -1.0133   0.9937  -1.0473) -> (   -1.0133     0.9937   -1.0473) -> (  -1.01328   0.992733  -1.047188)
( -2.3432     1.32   -0.451) -> (   -2.3432       1.32    -0.451) -> ( -2.343207   1.319126  -0.450997)
(  0.5559   0.9181   0.8971) -> ( -1.327033  -0.439964  -3.070903) -> (  0.555975    0.91818   0.897207)
( -4.5437   1.0037    0.097) -> (   -4.5437     1.0037     0.097) -> ( -4.543654   1.002788   0.097195)
(  -0.344  -1.1853  -0.0652) -> ( -1.216458   2.029501  -3.293712) -> ( -0.343682  -1.185754  -0.064153)
( -3.9801   2.3232   0.6456) -> (   -3.9801     2.3232    0.6456) -> ( -3.980207   2.322598   0.645206)
(  2.0019   0.9458   0.4789) -> ( -1.901824  -0.369719  -4.460601) -> (  2.001965   0.945878   0.478962)
( -0.8012  -1.8532   1.0734) -> ( -1.086143   1.895724  -4.678144) -> ( -0.800773  -1.853215    1.07475)
(  0.0519  -1.9194  -1.1859) -> ( -1.376015   3.299105  -2.733127) -> (  0.052282  -1.920292  -1.184544)
(  2.8467  -0.0833   0.8708) -> ( -2.553019  -1.476225  -4.987773) -> (  2.846897  -0.082961   0.871266)
(  2.4702   2.0004  -0.2924) -> ( -1.772103   0.800527  -5.195502) -> (  2.470111   2.000185  -0.292831)
(  -0.862  -3.2466   1.0913) -> ( -1.115044   3.024559  -5.496971) -> ( -0.861403  -3.246616   1.093256)
(  -0.009  -3.3128  -1.1681) -> (  -1.40505   4.427965  -3.551919) -> ( -0.008449  -3.313693  -1.166138)
(  4.1865  -0.0575   0.4838) -> ( -3.085174  -1.411413  -6.275449) -> (  4.186673  -0.057195   0.484173)
(     3.81   2.0262  -0.6796) -> ( -2.304392   0.865487  -6.483176) -> (  3.809886   2.025951  -0.680124)
( -0.4659  -3.9764  -0.0294) -> ( -1.274475   4.290655  -4.933894) -> ( -0.465241  -3.976855  -0.027138)
(  4.6682   0.9973  -0.2914) -> ( -2.960836  -0.240534  -7.023213) -> (  4.668217   0.997311  -0.291521)
( -0.4942   1.9282  -1.2881) -> ( -0.654005   0.051826   -0.61774) -> ( -0.494303   1.927194  -1.288409)
( -1.1618   0.4594  -1.9924) -> ( -0.290787   1.767992  -0.765894) -> ( -1.161736   0.458002  -1.992051)
(  0.4647   0.4673   1.8937) -> ( -1.943336  -1.098592  -2.445501) -> (  0.464854   0.467804   1.894005)
(  0.2024   1.9489   1.0333) -> ( -0.350937  -0.941593  -3.111717) -> (  0.202349   1.948995   1.032966)
( -4.8543   0.3353   0.9054) -> (   -4.8543     0.3353    0.9054) -> (  -4.85415   0.334701   0.905894)
( -5.3877   1.1752  -0.5771) -> (   -5.3877     1.1752   -0.5771) -> ( -5.387692   1.173887  -0.576959)
( -4.5165   3.1817   0.2277) -> (   -4.5165     3.1817    0.2277) -> ( -4.516726   3.180848   0.226945)
(  -4.067   2.3627   1.7365) -> (    -4.067     2.3627    1.7365) -> ( -4.067086   2.362562   1.736091)
( -3.4068  -0.4007  -1.0951) -> (   -3.4068    -0.4007   -1.0951) -> ( -3.406606  -0.401989  -1.094321)
( -1.1259  -1.3015   1.9518) -> ( -0.973556   0.916596  -5.136427) -> (  -1.12553  -1.301172   1.952915)
(  0.4193   -1.419  -2.0783) -> ( -1.469846   3.425331  -1.657477) -> (  0.419618  -1.420235  -2.077164)
(  2.4835   -0.909   1.4761) -> (  -2.65439  -2.394133  -4.415717) -> (  2.483783  -0.908508   1.476824)
(  1.8119    2.808  -0.5996) -> ( -1.261849    1.66778  -4.786282) -> (  1.811771    2.80772  -0.600118)
( -1.2204  -3.7633   1.9768) -> ( -1.016238   2.917547  -6.573226) -> ( -1.219719  -3.762975   1.978989)
(  0.3014  -3.8812  -2.0399) -> ( -1.527052   5.414006  -3.113397) -> (     0.302  -3.882435  -2.037699)
(  4.8548  -0.8585   0.7862) -> ( -3.596133  -2.272624  -6.696054) -> (  4.855075  -0.857992   0.786887)
(  4.1852   2.8476   -1.283) -> ( -2.207406   1.777066  -7.065567) -> (  4.184965   2.847122  -1.283909)
( -0.5133  -5.0614  -0.0156) -> (   -0.5133    -5.0614   -0.0156) -> (   -0.5133    -5.0614   -0.0156)
(  5.7114   1.0176  -0.5928) -> (    5.7114     1.0176   -0.5928) -> (    5.7114     1.0176   -0.5928)

Initial rotation angles:
EDGE        DEG      RAD
(3, 0)      -9.59°  -0.167377rad
(0, 7)     -73.76°  -1.287355rad
(5, 9)     -66.33°  -1.157677rad
(4, 3)     149.09°   2.602111rad
(0, 5)      -99.1°  -1.729621rad

Restored angles
EDGE        DEG       RAD
(4, 3)     149.09°   2.602111rad
(3, 0)      -9.59°  -0.167378rad
(0, 5)      -99.1°  -1.729619rad
(0, 7)     -73.76°  -1.287353rad
(5, 9)     -66.33°  -1.157668rad
(13, 17)     0.02°   0.000292rad
```

Рисунок 12 - Пример 2: молекула антазолина

Как видно из результатов, рёбра находятся правильно. Координаты восстанавливаются с очень маленькой погрешностью, соответственно углы тоже.

Выводы.

Был изучен и применён алгоритм Кабша. Было изучено применение сингулярного разложения на практике. Придумано и реализовано решение обратного поворота молекулы. Была осуществлена работа с парсингом json файла. Были применены алгоритмы на графах. Были реализованы функции поворота векторов в пространстве с помощью матриц поворота.

Список использованных источников.

1. https://www.bsuir.by/m/12_113415_1_70397.pdf - описание матриц поворота
2. https://ru.wikipedia.org/wiki/Матрица_поворота - ещё одно описание матриц поворота
3. https://drive.google.com/drive/folders/1MhRtUpfdauweNcWX5NtfEnyewFd1ofcP - конспекты лекций по алгоритму Кабша
4. https://en.wikipedia.org/wiki/Kabsch_algorithm - алгоритм Кабша
5. https://en.wikipedia.org/wiki/Singular_value_decomposition - SVD
6. https://towardsdatascience.com/simple-svd-algorithms-13291ad2eef2 - SVD Power iteration method
7. https://jeremykun.com/2016/05/16/singular-value-decomposition-part-2-theorem-proof-algorithm/ - SVD
8. https://en.wikipedia.org/wiki/Power_iteration - power iteration method
9. https://c3d.libretexts.org/CalcPlot3D/index.html - отрисовка графов в 3D