Правительство Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Национальный исследовательский университет»
«Высшая школа экономики»
Нижегородский кампус

Факультет математики, информатики и компьютерных наук
Кафедра фундаментальной математики

КУРСОВАЯ РАБОТА

НАЗВАНИЕ КУРСОВОЙ

Выполнил:
Студент 2 курса группы 22 ФМ
Турсунов Данил Вячеславович

Научный руководитель:
Баринова Марина Константиновна

Нижний Новгород
Май 2024 г.

Содержание

Hello everyone!

План работы

1 Вступление

1.1 Цели и задачи работы

Основной целью работы является создание приложения на языках C++ и Python с применением библиотеки Manim. Суть приложения заключается в генерации 3D-изображений дискретных динамических систем, заданных на сфере, из соответствующих им трёхцветных графов, заранее проверенных программой на корректность. Алгоритмическая часть приложения написана на C++, так как этот язык в разы быстрее чем язык Python, а на языке Python написана визуальная составляющая программы, так как язык содержит множество удобных для этого библиотек. Помимо этого поставлены следующие подзадачи: 1) Разобраться в связи динамических систем (КАКИХ?) и трёхцветных графов. 2) Научиться писать Unit-тесты на языке C++, необходимые для стабильной работы программы при её изменении в дальнейшем 3) Придумать алгоритм генерации трёхцветных графов с заданными параметрами: число Эйлера и число сёдел динамической системы.

1.2 Актуальность работы

Программа, написанная в результате работы, будет полезна начинающим научным сотрудникам или обычным студентам, желающим разобраться в динамических системах на сфере, так как поможет им быстрее визуализировать эти динамические системы. Помимо этого, отдельные функции из алгоритмический части могут быть полезны другим исследователям трёхцветных графов в написании рабочих программ.

2 Теоретическая часть

2.1 Описание

В этой главе будет представлено построение трёхцветного графа по градиентно-подобному каскаду на поверхности. Стоит отметить, что на языке трёхцветных графов получена полная топологическая классификация градиентно-подобных каскадов на поверхностях.

2.2 Трёхцветный граф как полный топологический инвариант диффеоморфизма на поверхности

Определение 1. *Диффеоморфизм $f : M^n \to M^n$, заданный на гладком замкнутом n-многообразии, называется диффеоморфизмом Морса-Смейла, если:*

1) неблуждающее множество Ω_f гиперболично и конечно (т.е. состоит из конечного чила

периодических точек, для которых модули собственных значений матрицы Якоби не равны единице);
2) для любых периодических точек p, q устойчивое многообразие W_p^s и неустойчивое многообразие W_q^u либо не пересекаются, либо трансверсальны в каждой точке пересечения.

Пусть $f : M^n \to M^n$ - диффеоморфизм Морса-Смейла, тогда периодические точки называются источниками, если неустойчивое многообразие W_q^u имеет размерность n, стоками, если 0, и сёдлами при остальных.
Далее скажем, что для любой периодической точки p диффеоморфизма f компоненты связности $W_p^s \backslash p$ ($W_p^u \backslash p$) называются её устойчивыми (неустойчивыми) сепаратрисами.
Рассмотрим класс диффеоморфизмов на поверхности M^2, тогда диффеоморфизм Морса-Смейла называется градиентно-подобным, если $W_p^s \cap W_p^u = \emptyset$ для любых различных седловых точек p, q.
В дальнейшем в работе будут рассматриваться исключительно граентно-подобные диффеоморфизмы, заданные на поверхности M^2.
Удалим из поверхности M^2 замыкание объединения устойчивых и неустойчивых многообразий седловых точек f и получим множество $M' = M^2 \backslash (W^u_{\Omega_f^0} \cup W^u_{\Omega_f^1} \cup W^s_{\Omega_f^1} \cup W^s_{\Omega_f^2})$.
M' является объединением ячеек, гомеоморфных открытому двумерному диску, граница которых имеет один из 3-х следующих видов: (КАРТИНКА С ЯЧЕЙКАМИ)
Пусть A - ячейка из M', α и ω - источник и сток, входящие в её границу. Кривую $\tau \in A$, началом и концом которой являеются α и ω, будем называть $t-$кривой. Через T обозначим множество $t-$кривых, взятых по одной из каждой ячейки.
Разобьём каждую ячейку этой кривой на 2 области, компоненты связности $M_\Delta = M' \backslash T$ назовём треугольными областями. Через Δ_f обозначим множество всех областей диффеоморфизма f.
В границу каждой треугольной области входят 3 периодические точки: источник, сток и седло, а также устойчивая сепаратриса, неусточивая сепаратриса и кривая τ. В дальнейшем будем называть их $s-$кривой, $u-$кривой и $t-$кривой соответственно. Замыкание каждой из этих кривых будем называть стороной треугольной области. Скажем, что сторона является общей стороной для двух треугольных областей, если она принадлежит замыканиям этих треугольных областей.
Для дальнейшего введения в теорию, потребуется ввести несколько определений из теории графов.

Определение 2. *Конечным графом называется упорядоченная пара (B,E), для которой выполнены следующие условия:*
1) B - непустое конечное множество вершин;
2) E - множество пар вершин, называемых рёбрами.

Определение 3. *Если граф содержит ребро e = (a,b), то каждую из вершин a, b называют инцидентной ребру e и говорят, что вершины a и b соединены ребром e.*

Определение 4. *Путём в графе называют конечную последовательность его вершин и рёбер вида:* $b_0, (b_0, b_1), b_1, \ldots, b_{i-1}, (b_{i-1}, b_i), b_i, \ldots, b_{k-1}, (b_{k-1}, b_k), b_k, k >= 1$. *Число k называется длиной*

пути, оно совпадает с числом входящих в него рёбер.

Определение 5. *Граф называют связным, если любые две его вершины можно соединить путём.*

Определение 6. *Циклом длины $k \in \mathbb{N}$ в графе называют конечное подмножество его вершин и рёбер вида $\{b_0, (b_0, b_1), b_1, \ldots, b_{i-1}, (b_{i-1}, b_i), b_i, \ldots, b_{k-1}, (b_{k-1}, b_0)\}$. Простым циклом называют цикл, у которого все вершины и рёбра попарно различны.*

С учётом имеющихся у нас вводных, введём определение трёхцветного графа, а также сформулируем некоторые теоремы.

Определение 7. *Граф T называется трёхцветным графом, если:*
1) множество рёбер графа T является объединением трёх подмножеств, каждое из которых состоит из трёх рёбер одного и того же определенного цвета (цвета рёбер из разных подмножеств не совпадают, будем обозначать эти цвета буквами s, t, u, а рёбра для краткости будем называть s-, t-, u-рёбрами);
2) каждая вершина графа T инцидентна в точности трём рёбрам различных цветов;
3) граф не содержит циклов длины 1.

Определение 8. *Простой цикл трёхцветного графа T назовём двухцветным su-, tu- или st-циклом, если он содержит рёбра в точности двух цветов s и u, t и u, s и t соответственно.*

Непосредственно из определения трёхцветного графа следует, что длина любого двухцветного цикла является чётным числом (так как цвета рёбер строго чередуются), а отношение на множестве вершин, состоящее в принадлежности двухцветному циклу определённого типа, является отношением эквивалентности, то есть каждая отдельно взятая вершина лежит в точности в одном $su-$, одном $tu-$ и одном $st-$цикле.

Определение 9. *Построим трёхцветный граф T_f, соответствующий диффеоморфизму $f \in G$, следующим образом:*
1) вершины графа T_f взаимно однозначно соответствуют треугольным областям множества Δ;
2) две вершины графа инцидентны ребру цвета s, t, u, если соответствующие этим вершинам треугольные области имеют общую s-, t- или u-кривую .

Граф T_f полностью удовлетворяет определению трёхцветного графа.

Теорема 1. *Теорема 1. Для того чтобы диффеоморфизмы f, f' из класса G были топологически сопряжены, необходимо и достаточно, чтобы их графы (T_f, P_f) и $(T_{f'}, P_{f'})$ были изоморфны.*

Определение 10. *Определение 2. Трёхцветный граф (T,P) назовём допустимым, если он обладает следующими свойствами:*
1) граф T связен;
2) длина любого su-цикла графа T равна 4;
3) автоморфизм P является периодическим.

Лемма 1. *Пусть $f \in G$. Тогда трёхцветный граф (T_f, P_f) является допустимым.*

Теорема 2. *Пусть (T,P) - допустимый трёхцветный граф. Тогда существует диффеоморфизм $f : M^2 \to M^2$ из класса G, граф (T_f, P_f) которого изоморфен графу (T,P). При этом:*
1) эйлерова характеристика поверхности M^2 вычисляется по формуле $X(M^2) = v_0 - v_1 + v_2$, где v_0, v_1, v_2 - число всех tu-, su-, st-циклов графа T соответственно;
2) поверхность M^2 ориентируема тогда и только тогда, когда все циклы графа T имеют чётную длину.

3 Алгоритмическая часть

3.1 Структура программы

Программа состоит из 2 частей: алгоритмической и графической, каждая из частей запускается отдельно и независимо друг от друга. Изначально запускается алгоритмическая часть на языке C++, туда вводится корректный трёхцветный граф, программа его обрабатывает и выдаёт в отдельном файле координаты сепаратрис. Далее запускается графическая часть программы, написанная на языке Python. Она генерирует по заданным координатам конца и начала сепаратрис 3D-изображение, а затем, после рендеринга в библиотеке Manim, показывает её пользователю.

3.2 Проверка введённого трёхцветного графа на корректность

Проверяет граф на корректность функция $is\_acceptable$, которая принимает на вход заданный граф и возвращает булевое значение: True, если граф является корректным (допустимым) трёхцветным графом, и False, если граф таковым не является. Согласно определению 10 граф называется корректым трёхцветным графом, если:
1) граф является трёхцветным, то есть попадает под определение трёхцветности;
2) граф является связным;
3) все SU-циклы в графе имеют длину равную четырём.
Функция $is\_acceptable$ для проверки пункта 1 вызывает функцию $is\_3\_colored\_and\_non\_oriented$, которая действует следующим образом: функция циклом проходит по вершинам графа, для каждой вершины проверяет, действительно ли из неё выходит только 3 ребра, причем эти рёбра должны быть разных цветов: u, s и t. Параллельно с этим в этом же цикле проверяется то, что граф является неориентируемым, а также то, что граф не содержит петель, то есть циклов длины 1. Если хотя бы одно из условий не выполняется, функция возвращает False, в противном случае она возращает True. (ФОТО АЛГОРИТМА)
Для проверки пункта 2 функция вызывает функцию $is\_connected$, которая считает расстояния от вершины с порядковым номером 0 до остальных вершин при помощи функции bfs. Здесь под расстоянием имеется в виду длина кратчайшего пути между 2 вершинами. Если не существует пути, соединяющего 2 вершины, расстояние считается равным inf. Если хотя бы одно расстояние окажется равным inf, то граф не связен и функция вернёт значение False, а в противном

случае граф является связным и функция возвращает значение True.
Функция bfs принимает на вход начальную вершину (в данном случае это верши с номером 0) и сам граф и работает по алгоритму $breadth - first\ search$, что можно перевести как "поиск в ширину". Алгоритм работает с применением структуры данных "очередь". Принцип работы этой структуры данных объясняется фразой: "Первый зашёл - последний вышел.". Алгоритм работает следующим образом: создаётся вектор, стуктура данных для хранения данных в C++, содержащий известные расстояния от начальной вершины до остальных, изначально заполнен inf, кроме начальной, так как расстояние от начальной до начальной вершины равно 0, и очередь, состоящая только из начальной вершины, далее запускается цикл, он работает до тех пор, пока очередь не опустеет. В каждой итерации цикл делает первую вершину из очереди текущей и проходит по всем соседним вершинам текущей (то есть тем, кто соединён с ней ребром), и, если известное расстояние до соседа больше суммы расстояния до текущей и единицы (ребро, которое их соединяет), то минимальное известное расстояние обновляется, а вершина-сосед кладётся в очередь. Функция возвращает вектор расстояний до каждой из вершин. (ФОТО АЛГОРИТМА)
Для проверки пункта 3 функция вызывает функцию $find\_cycles$ с переданными в неё графом, литералами 's' и 'u', которые отвечают за то, какого цвета циклы надо найти. Функция $find\_cycles$ возвращает вектор, состоящий из циклов, каждый цикл представляет собой последовательность номеров вершин цикла, так же последовательно соединённых между собой в самом цикле. Эта функция в процессе своего исполнения использует факт, который вытекает из определений 7 и 8 про то, что каждая отдельно взятая вершина лежит только в одном $su-$, одном $tu-$ и одном $st-$цикле, поэтому каждый цикл по отдельности ищется функцией $find\_cycle$ достаточно тривиальным алгоритмом, который просто идёт по циклу из начальной вершины, пока снова не встретит начальную вершину. Далее функция $is\_acceptable$ проверяет, имеют ли все SU-циклы в графе длину 4. (ФОТО АЛГОРИТМА)
Если все условия выполнены, функция $is\_acceptable$ возвращает True, в противном случае возвращает False.(ФОТО АЛГОРИТМА)

3.3 Проверка поверхности на ориентируемость

Для проверки графа на ориентируемость используется теория о базе циклов (ССЫЛКА НА ТЕОРИЮ). Базой циклов неориентированного графа является такой набор циклов, путём соединения или вычитания которых могут получиться все остальные циклы. Для нахождения базы циклов необходимо построить из графа дерево, а далее путём соединения дерева и рёбер графа, которые в дерево не попали, по одному найти все циклы из базы. По (ОТКУДА?) поверхность ориентируема тогда и только тогда, когда все циклы графа имеют четную длину. Очевидно, что при вычитании или сложении циклов четной длины получится цикл чётной длины, то есть на чётность достаточно проверить всего лишь циклы из базы циклов. Этот алгоритм реализован в функции $is\_oriented\_surface$, которая возвращает True, если поверхность ориентируема, и False, если поверхность неориентируема.

3.4 Генератор графов по заданному числу Эйлера и числу сёдел

Тут надо рисовать

3.5 Поиск соседних неподвижных точек

Нам дан корректный трёхцветный граф, для дальнейшей визуализации динамической системы для каждой неподвижной точки необходимо найти соседние, то есть соединённые с данной неподвижной точкой сепаратрисой, неподвижные точки. Для этого реализуем функцию $find\_neighbors$, которая принимает на вход корректный трёхцветный граф, а выдаёт вектор, состящий из стоков, сёдел и источников, а также их соседей в правильной последовательности. Для начала найдём все ST-, UT-, SU-циклы в исходном графе. Напомним, что каждый ST-цикл соответствует источнику, UT-цикл - стоку, SU-цикл - седлу. Из построения трёхцветного графа следует, что циклы имеют общее красное или синее ребро тогда и только тогда, когда неподвижные точки, представляющие эти циклы, соединены сепаратрисой, причём порядок обхода сепаратрис вокруг неподвижной точки соответствует порядку обхода рёбер графа, а также что одно ребро лежит ровно в двух двухцветных циклах, поэтому, найдя все двухцветные циклы, будем идти по ним в порядке обхода и для красных и синих рёбер будем смотреть, в каком ещё двухцветном цикле они лежат, далее сопоставляем новому двухцветному циклу для ребра неподвижную точку, соответствующую этому циклу. Проделаем это для всех неподвижных точек, получим искомый вектор.

3.6 Нахождение сепаратрис

Представим сферу как прямоугольник -90, 90 x 0, 360 фи пси, где все точки из отрезка -90 x 0, 360 и из отрезка 90 x 0, 360 отождествлены между собой. Впоследствии при визуализации этот прямоугольник будем отображать на сферу по формуле (КАКОЙ?). Сепаратрисы будем представлять как пары, состоящие из цвета сепаратрисы, красная или синяя, и вектора координат, содержащего $a, a0, b, b0$ и при этом заданной формулой (КАКОЙ?).

3.7 Unit-тестирование

4 Графическая часть

4.1 Работа с библиотекой Manim

Графическая часть программы, написанная в файле draw.py, отвечает за генерацию 3D-изображения дискретной динамической системы на сфере.

В файле определён класс DynamicalSystemSphere, который в дальнейшем будет указываться при запуске графической части.

Графическая часть работает по следующему алгоритму:

1) Считывается информация о сепаратрисах, полученная в результате запуска алгоритмиче-

ской части;
2) Объявляется функция $func\_sphere$, задающая параметрически поверхность сферы:
x = r * cos(u) * sin(v) - x0
y = r * sin(u) * sin(v) - y0
z = r * cos(v) - z0
3) Объявляется функция *construct*, которая отвечает непосредственно за генерацию 3D-изображения. При помощи библиотеки Manim создаются поверхность сферы и оси OX, OY и OZ. Далее каждая сепаратриса, разбитая на 3 равные части, отличающиеся по цвету: более яркий красный и синий цвет соответствуют близости к стокам и источникам соответственно, а бледные оттенки этих цветов соответствуют близости к седлу, по отдельности добавляется на поверхность сферы следующим образом:
сепаратриса, заданная параметрически на прямоугольнике значениями $a, a0, b, b0$, отображается на поверхность сферы по правилу:
x = cos(pi * (t * b + b0) / 180) * sin(pi * (t * a + a0) / 180)
y = cos(pi * (t * b + b0) / 180) * cos(pi * (t * a + a0) / 180)
z = sin(pi * (t * b + b0) / 180)
где t принадлежит [0, 1].
Для получения 3D-изображения объявляется полный оборот камеры вокруг сферы.

4.2 Запуск и результат работы программы

Для запуска генерации 3D-изображения необходимо:
1) При помощи терминала установить библиотеку Manim на компьютер, если эта библиотека ещё не установлена.
2) Предварительно запустить алгоритмическую часть с введённым в неё корректным трёхцветным графом.
3) При помощи терминала перейти в каталог с файлом draw.py.
4) Запустить графическую часть при помощи команды:
manim -pqh draw.py DynamicalSystemSphere - для генерации изображения высокого качества;
manim -pql draw.py DynamicalSystemSphere - для генерации изображения низкого качества.

5 Ссылка на репозиторий в Github

Ссылка: $https://github.com/dan1lka257/graphs\_and\_algorithms/tree/main$

6 Список литературы

капкаева, починка какая-нибудь ссылка на базу циклов