Московский государственный университет им. М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра автоматизации систем вычислительных комплексов

ДИПЛОМНАЯ РАБОТА

Разработка и реализация алгоритмов представления и операций над кубическими комплексами. Элементы синтеза на кубантах.

Выполнил:
Толстошеев Иван Андреевич,
группа 522

Научный руководитель:
профессор, чл.-корр. РАН
Рябов Геннадий Георгиевич

Москва 2010

Оглавление.

// TBD

Аннотация.

Рассматривается решетчатая модель пространства. Рассматривается введённый в [2] особый объект — кубант. Описываются основные операции над кубантом и над кубическими комплексами. Описывается разработанная система моделирования и визуализации кубантов. Рассматривается архитектурные особенности этой системы, описывается её реализация. Проводиться обзор основных операций над кубантами, а так же реализации этих операций. В раком программной системы реализован алгоритм Хаусдорфа на кубантах. Рассматривается вопрос быстродействия и функциональности программного комплекса.

1. Введение.

Задача представления пространства в виде, удобном для различных расчётов характеристик объектов этого пространства имеет много решений. Решения этой задачи находят себе применения в различных областях наук. Например, очень часто это применяется при медицинских исследованиях, – при магнитно-резонансной томографии. Еще одна область которая тесно связана с этой задачей — это проектирование микросхем. Во всех этих случаях нам нужно каким-то образом представлять структуру пространства и уметь взаимодействовать с объектами этого пространства.

Кроме взаимодействия с объектами пространства необходим инструмент, позволяющий человеку свободно управлять объектами, получать результаты исследований в наглядной форме, легко обрабатывать полученные результаты.

1.1 Структура пространства.

Существует множество способов представления пространства для удобства комбинаторно-топологических операций. Например, С. П. Новиков [1] рассматривает симплициальные комплексы для подобных целей. Модель симплициальных комплексов - гибкая модель для описания объекта в пространстве. Однако, симплициальные комплексы состоят из симплексов, которые могут иметь различные формы и размеры. Поэтому представление пространств и операции с симплициальными комплексами имеют большую вычислительную и алгоритмическую сложность. В связи с этим, реализация на вычислительной машине алгоритмов, работающих с симплициальными комплексами может встретить много препятствий не только со стороны скорости разработки, но и со стороны мощности компьютера. Для того, чтобы моделировать структуру и объекты в пространстве рассматривается более простая, наглядная, и естественная модель, обладающая рядом замечательных свойств.

Рассматривается дискретная решётчатая модель пространства. Для N-мерного пространства это решётка из целочисленных точек (точек с целочисленными координатами) Z^n в евклидовом пространстве R^n с заданным ортонормированным базисом (репером). Эта модель обладает свойствами, инвариантными относительно размерности пространства. Опишем эту модел ь более подробнее. Рассмотрим единичный куб (I^n) в пространстве R^n . Он обладает набором

граней 0,1, … n размерности, где 0-это целая точка (вершина куба), 1 — это одномерное ребро куба, n — это сам I^n. Совокупности таких n-мерных кубов в пространстве R^n образуют кубические комплексы. Кубические комплексы были исследованы С. П. Новиковым [1] . Эти и последующие исследования заложили отношения к кубическим комплексам как к объектам, обладающим широкими возможностями для представления комбинаторно-топологических структур.

В [4] введено понятие кубанта, для этого подробно рассматривается структура n-мерного куба. В I^n можно выделить составные части в виде кубов меньших размерностей — например I^{n-1} — гиперграни n-мерного куба, любая целочисленная вершина куба — 0-мерный куб. Итак, кубант — это конструктивный элемент n- мерного куба, куб меньшей или равной n размерности, принадлежащий исходному кубу.

Комбинируя кубанты в рамках одного n-мерного куба мы можем строить достаточно сложные объекты, а также производить над ними ряд операций.

1.2 Представление кубантов

Введём способ задания кубантов. Пусть есть n-мерный куб, и необходимо выбрать в нём некоторый кубант. Пусть, кроме этого, в R^n задан ортонормированный базис (e_1, e_2, … , e_n), и пусть задано множество всех возможных n-разрядных троичных слов {D}, с разрядами d_i из алфавита {0, 1, 2}. Число всех возможных таких слов равно 3^n. Пусть, также между реперными векторами и номерами разрядов установлено взаимно однозначное соответствие. В некотором слове K из множества D будет содержаться k_2 разрядов со значением 2, k_1 разрядов со значением 1, k_0 разрядов со значением 0, причём $k_0+k_1+k_2=n$. Всем разрядам, принимающим значение 2 ставиться в соответствие k_2 — мерный куб, построенный на соответствующих этим разрядам векторах. Остальные разряды (0 или 1) показывают наличие (1) или отсутствие (0) параллельного переноса на соответствующие им векторы. Проще говоря, трёхразрядный код обозначает два действия — формирования подкуба исходного куба и смещение его внутри исходного куба. В нашей кодировке для трёхмерного куба слову 222 соответствует он сам, а слову 001 — точка с координатами (0,0,1). Отметим, что в данном кодировании кубантов все слова без 2 в их составе в точности соответствуют точкам с теми же координатами. В трёхмерном кубе существует всего 1 кубант размерности 3, 6 кубантов размерности 2, 12 кубантов размерности 1 и 8 кубантов размерности 0. Эту кубанты соответствуют самому кубу,

его двумерным граням, его рёбрам и вершинам. Всего различных кубантов в трёхмерном кубе 27, что в точности равно 3^3.

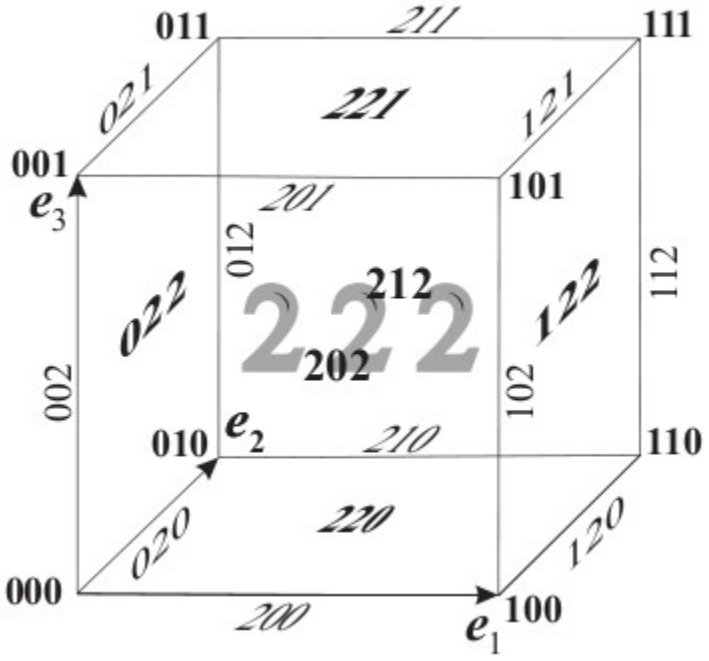


Рис.1 (кодирование кубантов в трёхмерном кубе)

На рисунке 1 изображён трёхмерный куб с отмеченными на нём кубантами.

1.3 Операции над кубантами.

Любые два кубанта в составе n-мерного куба могут находиться в следующих отношениях : не пересекаться, пересекаться или совпадать. Причем, областью пересечения двух кубантов является еще один кубант меньшей размерности, являющийся общей частью двух кубантов. Для простого нахождения пересечения кубантов введём операцию пересечения — поразрядную операцию на множестве слов из {D}.

c1/c2	0	1	2
0	0	∅	0
1	∅	1	1
2	0	1	2

Если при вычислении пересечения кубантов мы получили в одном из разрядов ∅, то это означает, что данные кубанты не имеют пересечения. Например, если мы возьмём кубанты

002212 и 011002, и применим к ним операцию пересечения, то получим результат:

0	0	2	2	1	2
0	1	1	0	0	2
0	∅	1	0	∅	2

Итак мы получили кубант 0∅10∅2, в кодированной записи которого содержится символ ∅, а значит, исходные кубанты не пересекаются.
Получается, что к символам алфавита {0,1,2} добавляется еще один символ - ∅. Наш итоговый алфавит для представления закодированных кубантов будет иметь вид {∅, 0, 1 ,2}. Кубант, содержащий в своей поразрядной записи хотя бы один символ ∅ мы будет называть псевдокубантом.

Так же представляет интерес операция выпуклая оболочка. Пусть даны несколько кубантов в одном I^n. Тогда, для того чтобы найти их наименьший общий кубант (наименьший кубант, содержащий в себе все кубанты из данного набора) мы введём новую поразрядную операцию над словом из множества {D}, описывающего кубант. Эта поразрядная операция описана в следующей таблице:

c1/c2	0	1	2
0	0	2	2
1	2	1	2
2	2	2	2

Например, если мы возьмём кубанты 002212 и 011002, и применим к ним операцию «выпуклая оболочка» то получим результат:

0	0	2	2	1	2
0	1	1	0	0	2
0	2	2	2	2	2

Таким образом, минимальный кубант, который содержит два исходных кубанта — это 022222.

Рассмотрим вычисления расстояния (в виде манхеттеновой метрики, или реберной метрики). Для этого просто вычислим пересечение кубантов, и рассмотрим количество ∅ в поразрядной записи пересечения. Пресечение кубантов обладает замечательным свойством - если это псевдокубант, то количество ∅ в его записи и будет расстояние в терминах манхеттеновой метрики.

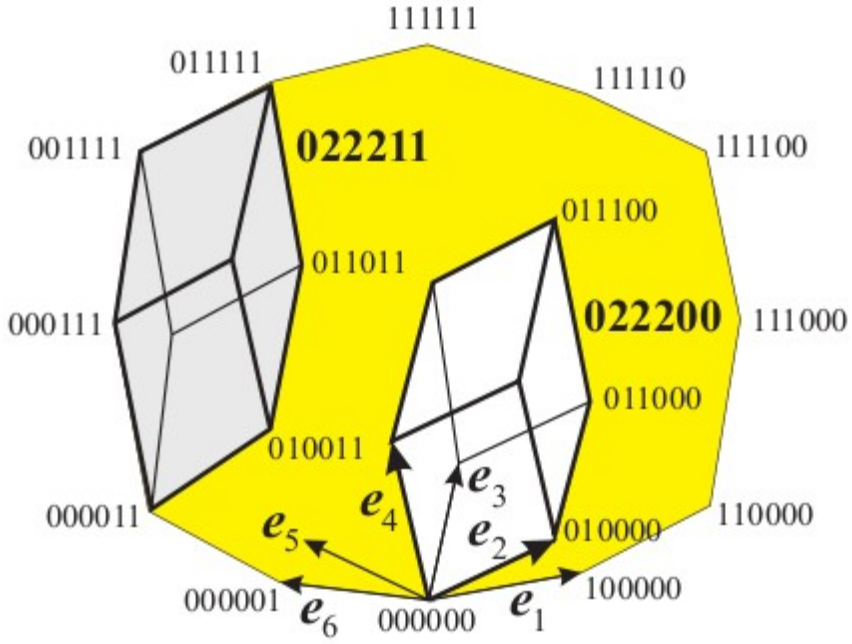


Рис 2. Манхеттеновое расстояние между кубантами 022211 и 022200 равно 2.

Так, на рисунке 2 изображены два кубанта, представимые в закодированном виде как 022211 и 022200. Вычисляя пересечение между ними, мы получим псевдокубант 0222∅∅. В записи этого псевдокубанта содержится 2 символа ∅, поэтому манхеттеновое расстояние между этими кубантами — 2.

Заметим, что все операции, введённые на множестве кубантов одной размерности поразрядные, а значит, их можно быстро вычислить на современных компьютерах. Получается, мы можем представлять сложные топологические объекты в виде совокупности кубантов, а затем быстро рассчитывать необходимые нам характеристики. Так же задача вычисления топологических свойств кубантов достаточно легко распараллеливается, так как объект легко разделяется на

множество маленьких подобъектов .

1.4 Цель работы.

Цель этой дипломной работы — исследовать возможности представления объектов в пространстве, используя решетчатую модель пространства. Для этого необходимо разработать программный продукт, который мог бы работать с кубантами, кубическими комплексами. Используя аппарат кубических комплексов мы можем описать и исследовать комбинаторно-топологические свойства достаточно сложных пространственных объектов в многомерных пространствах, а также пополнить программное обеспечение современных компьютерных систем.

Постановка задачи.

В рамках инструментальной системы и подсистемы «Алгебра кубантов» разработать приложения , позволяющие визуализировать кубанты и кубические комплексы, производить операции над кубантами и кубическими комплексами. Реализовать алгоритмы основных операций, описанных в [4]. Сделать возможность визуализации как в 2D режиме, так и в 3D. Предусмотреть в процессе разработке возможность исполнения части программного комплекса на супер-ЭВМ.

Обзор существующих решений поставленной задачи.

В [1] предлагается несколько способов представления структуры пространства. Для того, чтобы наиболее полно представить пространство используются симплексы (геометрические фигуры, являющиеся N-мерным обобщением треугольника). Из совокупности симплексов можно построить симплициальный комплекс, таким образом задать сколь угодно сложный объект в N-мерном пространстве.

Так же в [1] предлагается использовать N-мерную дискретную сетку, и использовать ячейки этой сетки для представления некоторых дискретных структур. Используя такое представление, можно построить сколь угодно сложные структуры, однако с определённой погрешностью.

В области программного обеспечения есть достаточно много реализованных проектов, которые позволяют моделировать пространство, использую симплициальную и решетчатую модель. Однако все эти программные продукты узкоспециализированы. Например Mathlab с установленным модулем поддержки симплициальных вычислений наиболее ориентирован на вычисления, а любая из программ 3-мерного моделирования (как пример, Blender) чаще всего используется не для математических рассчётов, а для отображения исходных данных.

Поэтому и была создана программная система, как отображать модель, так и вычислять характеристики определённых конфигураций в трёхмерном пространстве.

Описание программного комплекса для моделирования кубантов.

Предисловие.
Для того, чтобы моделировать кубанты или кубические комплексы (комплексы кубантов) было создано специальное программное обеспечение. Оно позволяет, с одной стороны производить вычисления и получать некоторые результаты над кубантами, и, с другой стороны — визуализировать полученные результаты.

Для того, чтобы эта многофункциональность было доступна, была разработана архитектура, состоящая из трёх практически независимых частей:

«Ядро»
«Интерпретатор»
«Визуализатор»

Опишу каждую из этих частей:

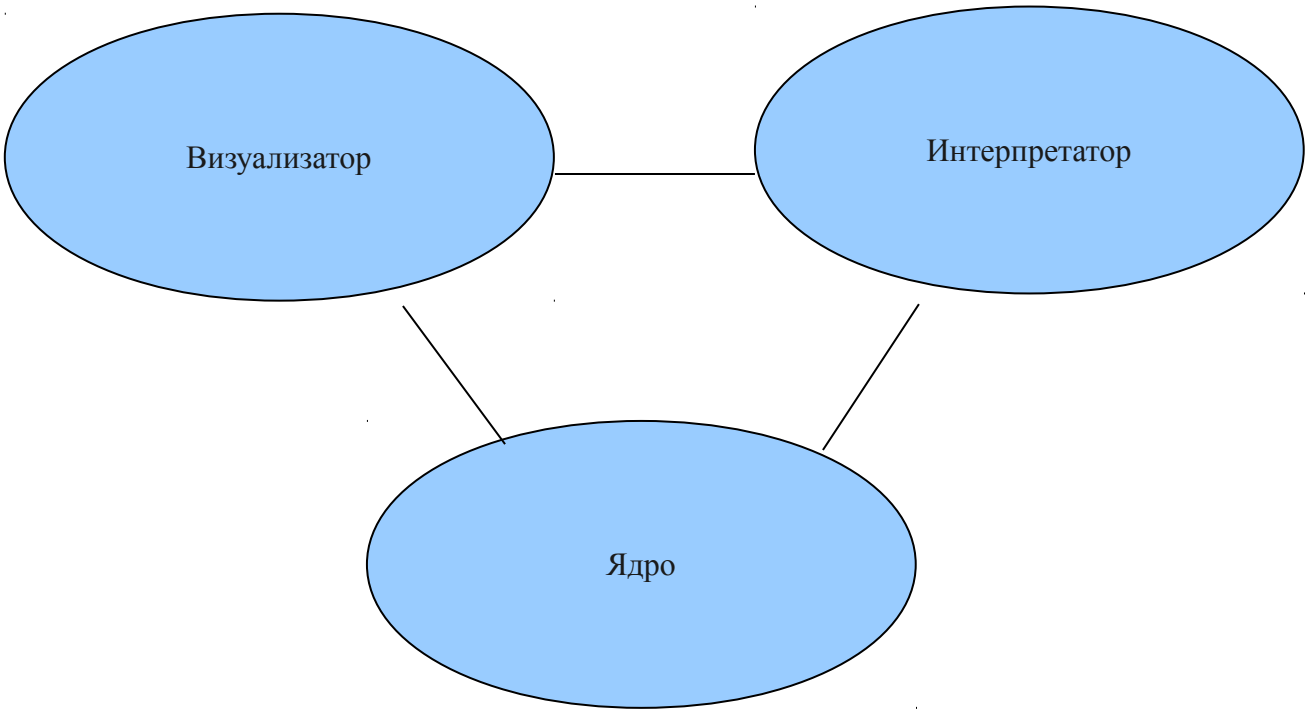
Ядро.
Ядро — это основная часть программы, которая предоставляет возможность создавать кубанты, производить с ними различные операции, получать результат. Ядро системы написано на языку программирования C++, для того чтобы обеспечить кроссплатформенность. За счёт этого, все программы, написанные с использованием библиотеки-ядра могут работать как и на обычных компьютерах, так и на суперкомпьютерах.

Интерпретатор.
Для того, чтобы произвести простой расчёт или просто произвести вычисления, не требующие больших вычислительных мощностей, можно использовать интерпретатор. Его использование позволяет быстро написать программу, не компилируя её. В качестве интерпретируемого языка в системе моделирования кубантов используется JavaScript. Интерпретатор может использоваться для предварительных расчётов, для поиска вариантов решения. После того, как подходящий алгоритм найден, мы можем реализовать его, используя непосредственно ядро, и прогонять его как на обычном настольном компьютере, так и на кластере.

Визуализатор.

Чтобы более полно представлять картину происходящего, в систему был добавлен визуализатор. Он используется для непосредственного отображения кубантов. С помощью визуализатора мы можем отобразить репер, кубант или несколько кубантов, задать различные цвет для кадого кубанта.

Все три компонента этой системы являются легко заменяемыми на другие, с аналогичным интерфейсом. За счёт этого достигнута гибкость — интерпретатор не обязательно должен быть интерпретатором языка Javascript, а может быть интерпретатором любого скриптового языка, например Lua или ActionScript. Практически это свойство заменяемости использовалось для построения двух различных визуализаторов: один 2D, а другой 3D, между которыми мы можем свободно переключаться. Так, рисуя кубант или кубический комплекс, мы получаем два (в текущей реализации) варианта его представления. Первый — это двухмерный вариант с плоским репером (для плоского варианта в текущей реализации репер мы можем задавать самостоятельно) и трёхмерный вариант, который отображает кубант, используя конусообразный репер.



Текущая схема Ядро — Интерпретатор — Визуализатор является реализацией известного паттерна проектирования MVC — Model — Controller — View , где модель — это ядро, контроллер — это интепретатор, а View — это визуализатор.

Поддерживаемые операции

Так как система моделирования кубантов является хорошо расширяемой системой, то количество операций в Ядре постоянно увеличивается. Поэтому я опишу некоторые из них, которые уже давно реализованы и стабильны.

Операция «Оболочка кубантов из гиперграней»
Данная операция по одному кубанту размерности больше 0 (т. е. не точке) создаёт множество кубантов, являющимися его гипергранями. То есть, для кубанта /2,0,1/ результат выполнения этой операции будет множество кубантов (/0,0,1/, /1,0,1/). С точки зрения языка Javascript (компонент интерпретатор) эта операция выглядит так:

```
var cubant=createCubant("/2,0,1/");
System.print(cubantFacet(cubant));
```

В первой строке создаётся кубант /2,0,1/, и сохраняется в переменной cubant. Во второй строке к переменной cubant применяется операция «Оболочка кубантов из гипергранней», и и сразу после выполнения этой операции выводится результат. Полученный результат — это Javascript-массив:

```
[ “/0,0,1/” ,“/1,0,1/”]
```

Таким образом с помощью интерпретатора мы можем достаточно быстро необходимую нам информацию.

Операция «Расстояние по рёбрам между кубантами»

В связи с тем, что мы рассматриваем решетчатые структуры, то и подсчёт расстояния между объектами в них может быть выполнен по-разному. Сейчас мы рассмотрим расстояние, вычисляемое на основе манхетенновской метрики. Мы будем считать расстояние между кубантами (только кубантами, а не кубическими комплексами) по рёбрам решетки. То есть, фактически расстояние по рёбрам в этом случае — это минимальное расстояние между какими -либо частями кубантов.

Предположим, что у нас имеется 2 кубанта в 9-мерном пространстве. Пусть они будут заданы кодами /1,2,1,0,0,1,2,2,1/ и /0,1,1,1,1,1,2,2,2,2/ . Выпишем их в виде таблицы и посчитаем хэмминогово расстояние.

1	2	1	0	0	1	2	2	1
0	1	1	1	1	2	2	2	2
∅	1	1	∅	∅	1	2	2	1

В первых двух строках данной таблицы описаны исходные кубанты, а в третьей строке — результат выполнения операции «расстояние по рёбрам». Как видно, оно вычисляется с помощью простых поразрядных операций. (Это свойство можно использовать для представления кубанта в виде некоторого кода в памяти вычислительной машины, для увеличения быстродействия и экономии оперативной памяти.)

В результате операция нахождения расстояния по рёбрам зависит линейно (O(n)) от размерности пространства. Разумеется, здесь говориться об нахождении расстояния в рамках единичного куба.

Заметим, что в получившемся результате нахождения расстояния (мы будем называть эту операцию операцией умножения) информации больше, чем просто число — символов ∅. В результате этой операции мы получаем кубант (или псевдокубант, если количество в его записи есть хотя бы один символ ∅).

Если символов ∅ в записи кубанта — результата операции умножение нет, то получившийся кубант — это просто пересечение данных кубантов.

Таким образом, операция умножения носит комбинаторный характер.

В реализации интерпретатора эта операция представлена методом intersect().

Пример её применения:

```
var cubant1=createCubant("/2,0,1/");
var cubant2=createCubant("/2,1,1/");
var result=intersect(cubant1, cubant2);
System.print(cubant);
```

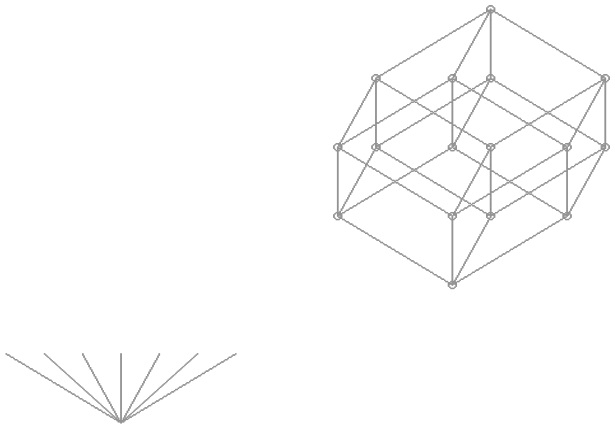
На выходе их консоли мы получим результат — кубант со значением "/2,Z,1/", что эквивалентно /2,∅,1/. Таким образом, получается, что расстояние по рёбрам расстояние между двумя данными кубантами — 1.

Однако, рассматривая внутри единичного N-мерного куба не только простые кубанты, но и кубические комплексы понятно, что они могут быть и не выпуклыми. Наиболее естественной метрикой в данном случае будет являться хаусдорфова метрика

Предположим у нас имеется два кубических комплекса в пространстве R3: комплекс A={/0,0,0/,/2,0,0/}, и комплекс B={/1,1,1/,/0,0,2/}. Найдём расстояние (в в смысле расстояния хаусдорфа между двумя этими комплексами)

// Здесь описание алгоритма.

Графическое представление.



4-мерный кубант в 7-мерном пространстве.

Для представления кубантов была разработана специльная утилита.

Использование интерпретируемого языка Javascript.

Перед тем, как писать программу для исполнения на суперкомпьютере, желательно иметь возможность выполнить предварительные вычисления. Эти вычисления нужны для того, чтобы иметь иметь возможность заранее просчитать что-то перед написанием и отладкой высокопроизводительной программы . Желательно также в процессе этих вычислений строить наглядное визуальное представление данных. Для этих целей был выбран Javascript как быстрый, легко встраиваемый скриптовый язык. Для встраивания был использован модуль QtScript из платформы Qt. Благодаря Javascript возможно выполнение программы для расчёта кубантов без перекомпиляции, построение визуального отображения в реальном времени.

В программной системе есть две независиых реализации встраивания Javascript — первая с использованием библиотеки SpiderMonkey. Данная библиотека используется в браузере Firefox.

В этой реализации имеется возможность только реализовать программы использующие вычислительные аспекты кубантов.

Вторая реализация выполнена с помощью библиотеки QtScript. В ней, кроме возможности реализаций вычислительных операций имеется возможность вывода графического представления. Графическое представление может выводиться в двух различных вариантах — это двумерный вариант представления кубанта и трехмерный вариант представления. В двумерном варианте кубант изображается как система точек, соединённых рёбрами.

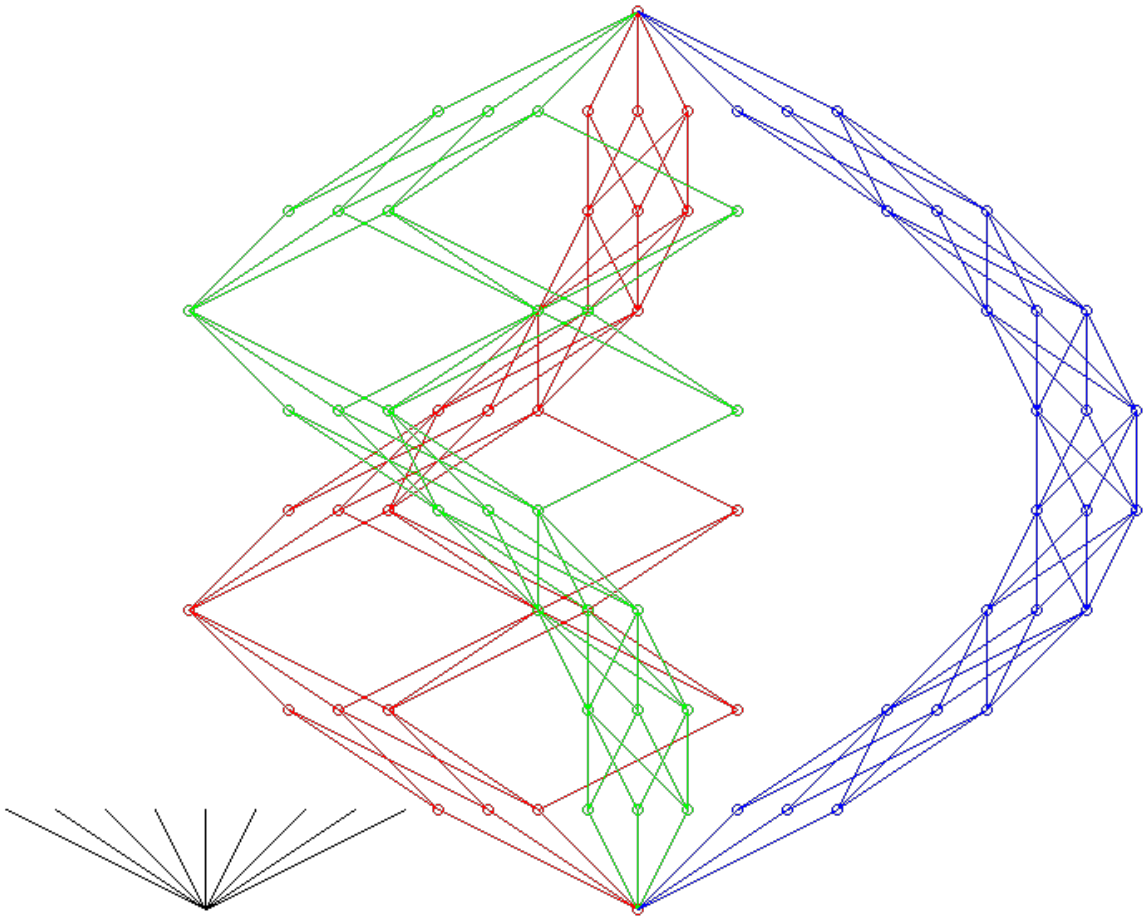


Иллюстрация 1: Три трёхмерных кубанта в 9-мерном кубе.

Базисные векторы кубанта на изображении задаются репером. Репер может быть инициализирован в Javascript-интерпретаторе нужными значениями.

Так же эта же конфигурация кубантов может быть изображена в трёхмерном варианте.

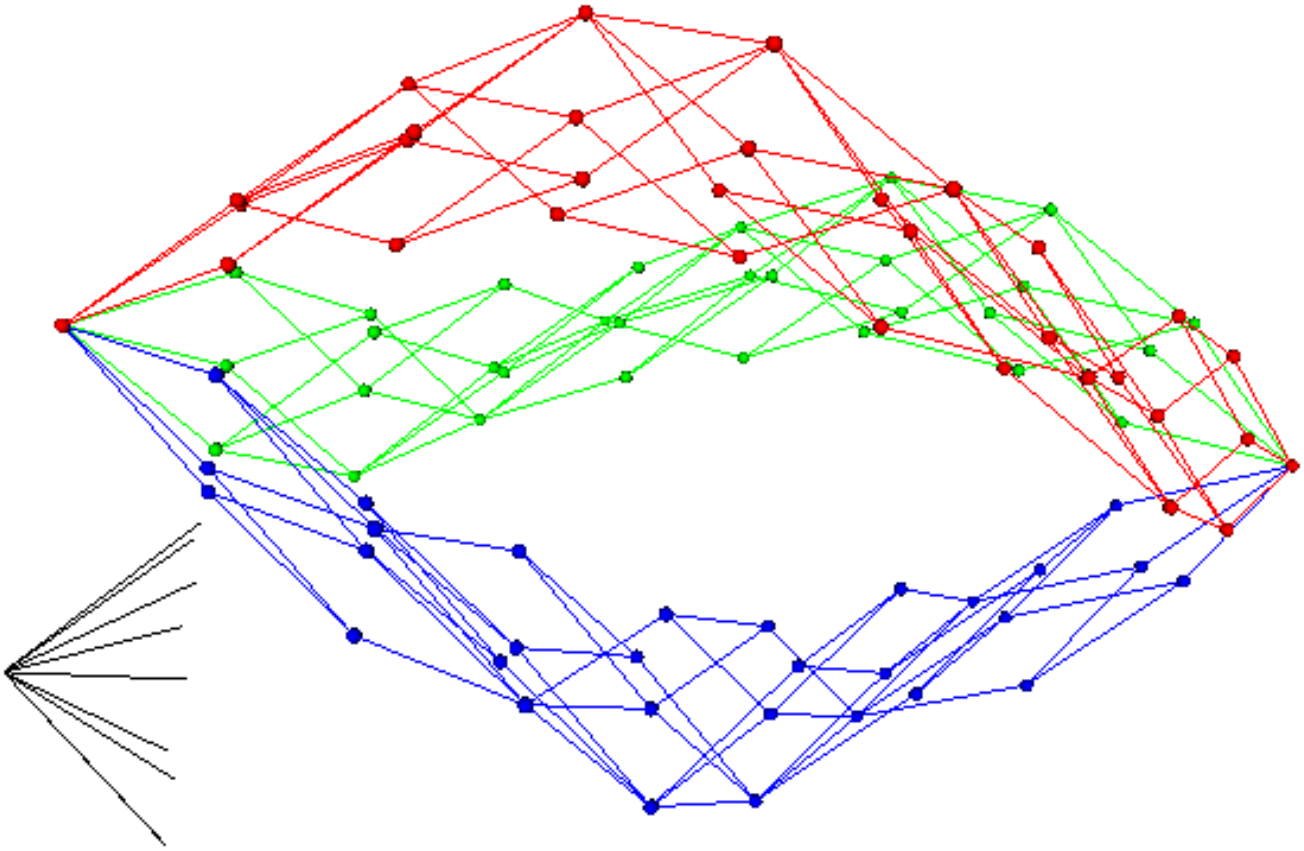


Иллюстрация 2: 3 кубанта в 9-мерном кубе

В трёхмерном случае можно также задать репер.

Синтез кубантов. Задача и решение.

Нужно решить следующую задачу: имеется 9-мерный кубант, и нам нужно построить пути из точки (0,0,0,0,0,0,0,0,0) в точку (1,1,1,1,1,1,1,1,1). Эти пути должны состоять из трёхмерных кубантов. Всего будет три пути, каждый кубант в пути будет соприкасаться с другим кубантами из этого пути двумерными гранями.

Для решения этой задачи была написана программа на языке Javascript. На вход этой программе подаётся начальное расположение исходных кубантов в точке (0,0,0,0,0,0,0,0,0). После этого программа рассчитывает пути из начальный точки в конечную, учитывая требование непересечения путей. После работы программы мы получаем решение, одно из них представлено на рисунке.

Для вычисления пути был реализован алгоритм для нахождения пути между кубантами. В случае с путём из кубантом он немного изменён (для построения именно пути, а не нахождения расстояния.)

2	2	2	0	0	0	0	0	0
1	2	2	2	0	0	0	0	0
1	1	2	2	2	0	0	0	0
1	1	1	2	2	2	0	0	0
1	1	1	1	2	2	2	0	0
1	1	1	1	1	2	2	2	0
1	1	1	1	1	1	2	2	2

Итак, пусть у нас имеется 2 кубанта A :/0,0,0,0,0,0,2,2,2/ B :/2,2,2,1,1,1,1,1,1/ , и между ними необходимо построить путь. Построение пути — это итерационная операция. Для соблюдения условий (каждый кубант в пути имеет общую двумерную грань со своим соседом (соседями)) мы можем производить следующаю операцию — менять в кубанте A одну двойку на единицу, и одновременно менять один ноль на двойку. Получившийся после этого кубант и будет удовлетворять всем свойствам соседа в пути. На основе этого и создан алгоритм на Javascript, который сторит путь от одного кубанта к другому.

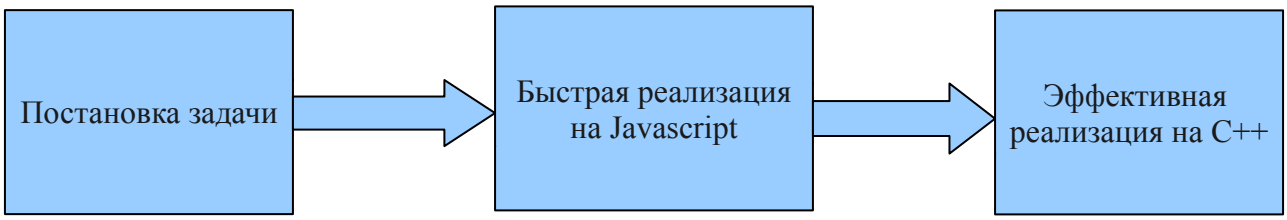
Ниже приводиться схема его выполнения.

Для изображения 3-мерной картинки была применена технология VRML. VRML быд выбран потому, что это платформонезависимый язык, предназначенный для 3D-моделирования. Таким образом, получается, что одной языковой конструкцией в интерпретаторе Javascript мы достигаем сразу двух целей – одновременно изображаем двухмерный и трёхмерный вариант расположения кубантов.

Вычислительный аспект.

Кроме визуализации кубантов данный программный комплекс предоставляет возможность

выполнить некоторые предварительные вычисления, которые можно впоследствии реализовать на низком уровне для увеличения быстродействия. Таким образом, работа над каким-то вычислением обычно реализовна таким способом:



В рамках разработки вычислительного комплекса для моделирования кубантов была рассчитана попарная для каждого из кубантов из размерности метрика хаусдорфа по рёбрам. Для этого алгоритм рассчёта был сначала реализован на Javascript для доказательства его применимости, а после вычисления задачи в трёхмерном пространстве с использованием интепретируемой программы на Javascript этот алгоритм был переписан на C для увеличения производительности. В результате расчетов, используя встроенные функции печати в Javascript мы получили следующий результат:

dummy	/0,0,0/	/0,0,1/	/0,1,0/	/0,1,1/	/1,0,0/	/1,0,1/	/1,1,0/	/1,1,1/	/0,0,2/	/0,1,2/	/0,2,0/	/0,2,1/	/1,0,2/	/1,1,2/	/1,2,0/	/1,2,1/	/2,0,0/	/2,0,1/	/2,1,0/	/2,1,1/	/0,2,2/	/1,2,2/	/2,0,2/	/2,1,2/	/2,2,0/	/2,2,1/	/2,2,2/
/0,0,0/	0	1	1	2	1	2	2	3	1	2	1	2	2	3	2	3	1	2	2	3	2	3	2	3	2	3	3
/0,0,1/	1	0	2	1	2	1	3	2	1	2	2	1	2	3	3	2	2	1	3	2	2	3	2	3	3	2	3
/0,1,0/	1	2	0	1	2	3	1	2	2	1	1	2	3	2	2	3	2	3	1	2	2	3	3	2	2	3	3
/0,1,1/	2	1	1	0	3	2	2	1	2	1	2	1	3	2	3	2	3	2	2	1	2	3	3	2	3	2	3
/1,0,0/	1	2	2	3	0	1	1	2	2	3	2	3	1	2	1	2	1	2	2	3	3	2	2	3	2	3	3
/1,0,1/	2	1	3	2	1	0	2	1	2	3	3	2	1	2	2	1	2	1	3	2	3	2	2	3	3	2	3
/1,1,0/	2	3	1	2	1	2	0	1	3	2	2	3	2	1	1	2	2	3	1	2	3	2	3	2	2	3	3
/1,1,1/	3	2	2	1	2	1	1	0	3	2	3	2	2	1	2	1	3	2	2	1	3	2	3	2	3	2	3
/0,0,2/	1	1	2	2	2	2	3	3	0	1	1	1	1	2	2	2	1	1	2	2	1	2	1	2	2	2	2
/0,1,2/	2	2	1	1	3	3	2	2	1	0	1	1	2	1	2	2	2	2	1	1	1	2	2	1	2	2	2
/0,2,0/	1	2	1	2	2	3	2	3	1	1	0	1	2	2	1	2	1	2	1	2	1	2	2	2	1	2	2
/0,2,1/	2	1	2	1	3	2	3	2	1	1	1	0	2	2	2	1	2	1	2	1	1	2	2	2	2	1	2
/1,0,2/	2	2	3	3	1	1	2	2	1	2	2	2	0	1	1	1	1	1	2	2	2	1	1	2	2	2	2
/1,1,2/	3	3	2	2	2	2	1	1	2	1	2	2	1	0	1	1	2	2	1	1	2	1	2	1	2	2	2
/1,2,0/	2	3	2	3	1	2	1	2	2	2	1	2	1	1	0	1	1	2	1	2	2	1	2	2	1	2	2
/1,2,1/	3	2	3	2	2	1	2	1	2	2	2	1	1	1	1	0	2	1	2	1	2	1	2	2	2	1	2
/2,0,0/	1	2	2	3	1	2	2	3	1	2	1	2	1	2	1	2	0	1	1	2	2	2	1	2	1	2	2
/2,0,1/	2	1	3	2	2	1	3	2	1	2	2	1	1	2	2	1	1	0	2	1	2	2	1	2	2	1	2
/2,1,0/	2	3	1	2	2	3	1	2	2	1	1	2	2	1	1	2	1	2	0	1	2	2	2	1	1	2	2
/2,1,1/	3	2	2	1	3	2	2	1	2	1	2	1	2	1	2	1	2	1	1	0	2	2	2	1	2	1	2
/0,2,2/	2	2	2	2	3	3	3	3	1	1	1	1	2	2	2	2	2	2	2	2	0	1	1	1	1	1	1
/1,2,2/	3	3	3	3	2	2	2	2	2	2	2	2	1	1	1	1	2	2	2	2	1	0	1	1	1	1	1
/2,0,2/	2	2	3	3	2	2	3	3	1	2	2	2	1	2	2	2	1	1	2	2	1	1	0	1	1	1	1
/2,1,2/	3	3	2	2	3	3	2	2	2	1	2	2	2	1	2	2	2	2	1	1	1	1	1	0	1	1	1
/2,2,0/	2	3	2	3	2	3	2	3	2	2	1	2	2	2	1	2	1	2	1	2	1	1	1	1	0	1	1
/2,2,1/	3	2	3	2	3	2	3	2	2	2	2	1	2	2	2	1	2	1	2	1	1	1	1	1	1	0	1
/2,2,2/	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	0

Данная таблица была быстро получена с помощью системы моделирования кубантов, так как в рамках систем была реализована встроенная в Javascript поддержка стаднартного формата таблиц — CVS .

Реализация метрики Хаусдорфа на кубантах.

В [4] описан алгоритм для эффективного нахождения хаусдорфовой метрики на кубантах. Хаусдорфова метрика определяется слудющим образом:

$$d_H(X,Y)=max\left(\sup_{x\in X}\left(inf_{y\in Y}\left(\left(xy\right)\right)\right),\sup_{y\in Y}\left(inf_{x\in X}\left(\left(xy\right)\right)\right)\right)$$

Поэтому вычисление метрики Хаусдорфа по определению — это трудоёмкая вычислительная задача (O(n+m), где m и n – это количество точек в множествах X и Y). Для кубантов мы можем сохранить ту же сложность, однако вместо точек здесь выступают кубанты. Поэтому мы можем строить более сложные объекты и вычислять метрику Хаусдорфа со сложностью O(n+m), где n и

m — это количство кубантов в множествах X и Y.

Для этого можно определить метрику Хаусдорфа как

$$d_H(X,Y)=max\left(Lmin\left(D_{1,}D_2^{i}/D_1\right),Lmin\left(D_{2,}D_1^{i}/D_2\right)\right)$$

где «сжатие кубантов» D1*/D2 определяются согласно поразрядным действиям, заданным в таблицах.

D2*/D1	0	1	2
0	0	1	1
1	0	1	0
2	0	1	2

D2*/D1	0	1	2
0	0	1	1
1	0	1	0
2	0	1	2

Такое сжатие соответствует выделению в одном кубанте удалённой части, которая в свою очередь, тоже является кубантом.

Этот алгоритм полностью реализован в системе моделирования кубантов, с помощью него были посчитана метрика Хаусдорфа для любых 2 различных кубантов в 3-ехмерном пространстве. Так же он может быть расширен для подсчёта метрики Хаусдорфа среди кубических комплексов.

Оценка программного системы для моделирования кубантов.

Для создания программного средства — системы моделирования кубантов был применён объектно-ориентированный подход. Благодаря этому, приложение легко расширяемо и гибко в разработке. За представление кубанта и его операций отвечает шаблонный класс Cubant.

```
template <class Implementation> class Cubant;
```

Благодаря этому мы имеем возможность использовать различные реализации, легко их заменяя. На сегодняшний день существуют две реализации — там где разряд кубанта представлен в виде битов - `BitImpl` (быстрые операции, малое потребление памяти, чуть медленнее извлечение данных), и там где разряд кубанта представлен в виде числа `Int32mpl`. Последняя реализация была сделана больше для отладочных целей. Однако, на другой архитектуре, отличной от архитектуры процессоров intel, существует возможность переопределить реализацию для достижения оптимальной производительности.

Оценка производительности приложения.

В большинстве операций над кубантами лежат в основе битовые операции над числами, который выполняются за время, сопоставимое с 1 тактом на современных машинах с архитектурой x86 (x86_64). Поэтому, скажем, операция нахождения реберного расстояния между кубантами выполниться за O(k), где k-это размерность пространства. Заметим, что если размеры пространства меньше либо равно, чем 16, то данная операция выполнится за ~1 такт вычислительной машины. Для вычисления метрики Хаусдорфа между двумя кубантами оценка также O(k) .

В случае, если мы используем интерпретатор Javascript для предварительных вычислений, то производительность приложения будет заметно ниже. Это происходит из-за того, что Javascript — это интерпретируемый язык. Поэтому на любую быструю операцию, реализованную в системе моделирования кубантов накладываются избыточные расходы, которые порождает Javascript. По эмпирическим оценкам, скорость выполнения приложения падает в 3-100 раз при использовании Javascript.

Однако, это не является серьёзным недостатком так как Javascript в системе моделирования кубантов предназначен для предварительных «грубых» вычислений. Учитывая это, падение производительности примерно на 1-2 порядка можно считать приемлимым.

Потребление памяти.

Для того, чтобы моделировать большие количества кубантов, требуется некоторые объемы памяти ЭВМ. Оценим требуемые объемы.

Положим, мы используем реализацию кубантов `BitImpl`. В этом случае каждый кубант занимает O(k) байт памяти. Например, кубанты с размерностями от 1 до 16 занимают 4 байта, кубанты с размерностями от 17 до 32 занимают 8 байт и так далее. Так как объёмы памяти зависят линейно от размерности кубантов, то мы можем использовать систему моделирования кубантов в расчётах для кубантов больших размерностей.

В случае с использованием реализации `Int32Impl` система для моделирования кубантов использует примерно в 16 раз больше памяти ЭВМ. Таким образом, использования реализации операций кубантов BitImpl оправдано потреблением памяти.

В случае использования интерпретатора Javascript потребляемая память значительно увеличивается за счёт внутренних структур Javascript.

Производительность приложения: выводы.

Производительность приложения с точки зрения скорости выполнения команд и потребляемой памяти зависит от количества кубантов и размерности пространства кубантов. Причём, зависимость от размерности кубантов - линейная. Это позволяет использовать достаточно большие размерности для вычислений.

Заключение.

Был разработан программный комплекс для синтеза и моделирования кубантов. С помощью него можно производить вычислительные операции над кубантами, так и визуализировать полученные результаты, в том числе и промежуточные. Библиотеку, разработанную в рамках данного проекта можно использовать для вычисления на суперкомпьютерах. Благодаря Javascript-интерпретатору, встроенному в программный комплекс, возможно проведение предварительных отладочных вычислений.

Список литературы:

1. Новиков С.П. Топология. Москва-Ижевск: РХД, 2002.
2. Рябов Г. Г. О путевом кодировании k-граней в n-кубе. // Вычислительные методы и программирование. 2008. 9, №1. 20-22
3. Понтрягин Л. С. Основы комбинаторной топологии. - 3-е изд. - М.: Наука. Гл. ред. физ. мат. лит. 1986.
4. Рябов Г. Г. О четверичном кодировании кубических структур. // Вычислительные методы и программирование. 2009. Т. 10. 340-347
5. V. Desoutter, N. Destainville. Flip dynamics in three-dimensional random tilings // arXiv:cond-mat/0406728v3 [cond-mat.stat-mech]
6. Н. П. Долбилин, М. А. Штанько, М. И. Штогрин, “Кубические многообразия в решётках”, Изв. РАН. Сер. матем., 58:2 (1994), 93–107
7. Manin Yu. I. Classical computing, quantum computing and Shor's factoring algorithm. // arXiv : quant-ph/9903008v1, 2 March, 1999.
8. Деза М., Шторгин М. Вложение графов в гиперкубы и кубические решетки // Успехи матем. Наук. 1997. 52, № 6, 155-156
9. Деза М., Шторгин М. Мозаики и их изометрические вложения. // Изв. РАН Сер. Матем. 2002. 66, №3, 3-22.