**Министерство науки и высшего образования РФ**

**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«Казанский (Приволжский) Федеральный Университет»**

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ТЕОРЕТИЧЕСКОЙ КИБЕРНЕТИКИ

Направление: 01.03.02. Прикладная математика и информатика

Профиль: Прикладная математика и информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Математическая модель описания эпидемиологических процессов. Исследование влияния интенсивности нелокальных контактов в популяции на развитие эпидемии.

**Работа завершена:**

Студент гр.09-712 ИВМиИТ

«\_\_»\_\_\_\_\_\_\_20\_\_г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С.Анцырев

**Работа допущена к защите:**

Научный руководитель,

доцент, к.н., КТК, ИВМиИТ

«\_\_»\_\_\_\_\_\_\_20\_\_г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.Н. Нурмеев

Заведующий кафедрой,

д.ф.м.н., профессор

«\_\_»\_\_\_\_\_\_\_20\_\_г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.М.Аблаев

**Казань – 2021**

Оглавление

[1. Введение 3](#_Toc74976269)

[2. Цель и задачи работы 5](#_Toc74976270)

[3. Теоретический материал 6](#_Toc74976271)

[3.1 Теория перколяции 6](#_Toc74976272)

[3.2 Теория клеточных автоматов 9](#_Toc74976273)

[3.3 Линейная алгебра. 12](#_Toc74976274)

[3.4 Теория вероятностей. 13](#_Toc74976275)

[3.5 Другие методы моделирования. 13](#_Toc74976276)

[4. Реализация программы. 14](#_Toc74976277)

[4.1 Описание программы с условной конструкцией. 14](#_Toc74976278)

[4.2 Нелокальные контакты. 18](#_Toc74976279)

[4.3 Реализация векторизованной программы. 19](#_Toc74976280)

[4.4 Визуализация. 20](#_Toc74976281)

[5. Проведение и анализ экспериментов. 23](#_Toc74976282)

[5.1 Эксперимент 1. 23](#_Toc74976283)

[5.2 Эксперимент 2. 24](#_Toc74976284)

[5.3 Эксперимент 3. 26](#_Toc74976285)

[6. Выводы и рекомендации. 28](#_Toc74976286)

[7. Заключение 30](#_Toc74976287)

[8. Литература 35](#_Toc74976288)

[9. Приложения 37](#_Toc74976289)

# Введение

Существует огромное количество заболеваний, но одной из самых критических для населения являются инфекции и болезни, которые переносятся от человека к человеку воздушно-капельными путями (через капли, выделяющиеся из дыхательных путей человека, к примеру, кашель, чихание или даже общение) из-за их скорости распространения, масштаба, большого числа больных и погибших и т.д.

Со временем появляются всё более прогрессирующие в пространстве и времени инфекционные заболевания, симптомы которых менее заметны, вирулентность выше, с большим спектром способов распространения. Такие заболевания тяжело контролировать, порою они даже переходят в чрезвычайные положения. Их называют эпидемиями (от греческого «повальная болезнь»). У эпидемии же есть более тяжелая стадия, которая распространяется на территориях стран и даже континентов, этот термин называется пандемией (от греческого «весь народ»). Более известными примерами эпидемий и пандемий являются: чёрная оспа, чума, тиф, холера, малярия, туберкулёз, ВИЧ и многие другие. На сегодняшний день актуальной проблемой человечества стала коронавирусная инфекция Covid-19. Первое время не придавалось большого значения этой болезни, но со временем масштаб проблемы увеличился до уровня всей планеты: количество смертей достигло сотен тысяч, а затем и миллионов; был нанесён большой ущерб по экономике многих государств, из-за карантинных мер и т.д.

Но коронавирус является лишь частным случаем множества инфекционных болезней, которые могут появится в мире в любую минуту. Поэтому нужно выработать методы моделирования, а позднее и анализа распространения, а также борьбы с такими проблема. На данный момент не выработано общепризнанного метода по описанию инфекционных заболеваний, поскольку необходимые условия для их создания появились только 2-3 десятка лет назад, а именно:

* уровень человеческого развития в нужных областях математики, для того чтобы правильно описать модель, которая будет аппроксимировать реальную ситуацию.
* уровень человеческого развития в программировании (алгоритмах и структурах данных), для оптимизированной и быстрой работы программ.
* уровень технологического развития, для быстрого выполнения программ при помощи распараллеливания и векторизации определённых моментов в коде.

При изучении методов, при помощи которых можно решать проблему моделирования меня заинтересовала теория перколяции – наука молодая и не очень популярная. Она описывает системы, которые могут находиться в различных состояниях, причём переход из одного состояние в другой происходит при каких-то условиях, т.е. в критический момент, который и называется перколяцией.

В нашем случае средой для протекания инфекций являются места взаимодействия людей, люди – это элементы системы, а контакт между ними является способом переноса.

Применив теорию перколяции в курсовой работе[[3]](#_Литература) и проанализировав результаты, я получил следующий вывод – данная теория может смоделировать очень упрощённую версию реальной ситуации, не учитывая при этом слишком много факторов и параметров, например, наличие у человека вакцины или иммунитета, длительность латентного периода или самого заболевания и т.д.

Поэтому пришлось дополнить модель при помощи теории клеточных автоматов (КА) – которая поможет закрыть недостатки теории перколяции и приблизить её результаты к реальным. Клеточные автоматы – это простые модели вычислений, способные моделировать биологически, физически или экологически сложные явления.

# Цель и задачи работы

**Цель работы:** необходимо изучить влияния числа нелокальных контактов на протекание инфекционных заболеваний.

**Задачи:**

1. Применить теории перколяции и КА и построить математическую модель для описания протекания инфекционных заболеваний.
2. Применить теорию вероятностей, линейную алгебру (векторизацию) для оптимизации и лучшей аппроксимации модели.
3. Создать визуализацию процесса распространения инфекций.
4. Провести эксперименты и их анализ, для выявления закономерностей.

# Теоретический материал

Для начала разберёмся с теоретическими аспектами, которые позволят нам построить модель, поймём почему нам нужна та или иная область знаний. А позднее, опираясь на эти знания построим модель.

## 3.1 Теория перколяции

Впервые про теорию перколяции было описано в работах Стокмайера и Флори в 40-х годах, однако более детально о теории протекания было рассказано в публикации работы Хаммерсли и Броадбента в 1957 году[[1]](#_Литература)  
С каждым годов в Западных изданиях появляются все больше и больше публикаций, связанных с этим «молодым» направлением науки. Они связаны с изучением новых теоретических вопросов теории перколяции, а также продолжением и новыми направлениями ее применения.  
Теория перколяции описывает образование связанных между собой объектов в неупорядоченной среде. Можно ее рассматривать с точки зрения математики – теория вероятности в графах, или с точки зрения физики – геометрический фазовый переход. Теория перколяции (от англ. percolation processes, просачивание или фильтрация) описывает возникновение бесконечных связных структур (кластеров), которые состоят из отдельных элементов[[2]](#_Литература).  
 Теория получила широкое применение в электрических свойствах неупорядоченных систем, таких как аморфные или кристаллические полупроводники с примесями, а также металлы и диэлектрики. Но также задача оказалась актуальной и с точки зрения чистой математики.

Теория протекания важна именно в момент перехода и определяется средой, в которой наблюдается это явление, внешним источником, обеспечивающим протекание в данной среде, а также его способом протекания.

В бесконечной квадратной сетке или бесконечном регулярном графе точки пересечения называются **узлами** (вершинами), а сами линии **связями** (ребрами).

Можно случайным образом закрашивать узлы решетки, считая долю крашенных узлов независимым параметром, а крашенные узлы будут принадлежать одному кластеру, если их можно соединить непрерывной цепочкой соседних закрашенных узлов. В теории протекания есть такой термин, как «кластер» (от англ. гроздь) – это цепочка связанных объектов. Кластер, который соединяет противоположные стороны системы, называется бесконечным.

Задача узлов изучает такие вопросы, как среднее количество узлов в кластере, появление бесконечного кластера, распределение по размерам кластера, доля входящих в него закрашенных узлов. Когда закрыты все связи или узлы, решетка является моделью изолятора. При определенном критическом значении произойдет переход перколяции.

Имеется в виду «критическая точка», при которой определенные свойства сразу меняются. Представление всех критических явлений своеобразна, но имеет общие свойства, самое важное из которых состоит в том, что ближе к критической точке система распадается на блоки, у которых также свои критерии.

Момент появления такого состояния решетки, при котором существует хотя бы один непрерывный путь через соседние проводящие узлы от одного до противоположного края называется перколяцией. Отсюда получается, что с ростом числа проводящих узлов, этот момент наступит раньше, чем вся поверхность решетки будет состоять исключительно из проводящих узлов.

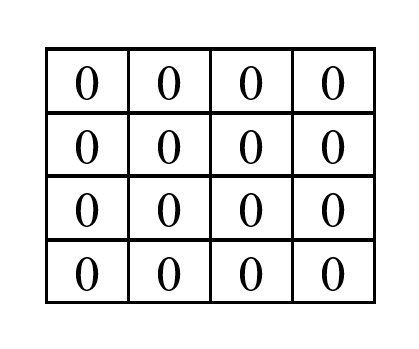
Обозначим непроводящие состояния узлов нулями, а проводящие единицами. Вместе они складываются в двумерную матрицу. При замене нулей полученной на единицы мы получим источник протекания. Изначально матрица из непроводящих элементов будет выглядеть так - Рисунок 1.

Рисунок 1

Когда на среду воздействует внешний источник в матрице появятся проводящие элементы, но их еще недостаточно для протекания. Можно наглядно посмотреть на Рисунке 2.

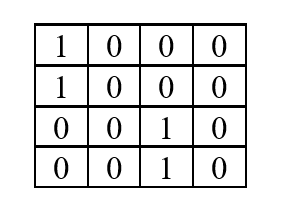
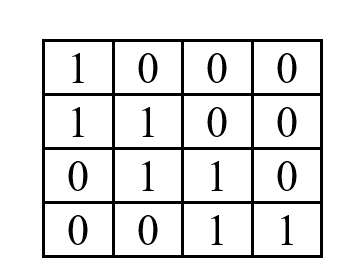
С увеличением числа проводящих узлов получается критический момент, при котором происходит перколяция. На рисунке 3 можно наглядно видеть, как от левой границы к правой имеется цепочка элементов, она обеспечивает перколяцию тока по проводящим узлам (по единицам), которые непрерывно соединены друг за другом.

Рисунок 2

Протекание можно наблюдать в

прохождении жидкости через объемный пористый предмет. Ярким примером можно сказать про губку из пенообразующего материала. В ней при постепенном надувании пузырьков и достижении ими определенного размера жидкость просачивается от одного края губки до другого.

Рисунок 3

В связном случайным графом, в зависимости от конкретной реализации кластер может иметь различную форму. Принято характеризовать его общим размером. Количество элементов перколяционного кластера, отнесенное к общему количеству элементов рассматриваемой среды называется **порогом протекания**.

Основы методов теории перколяции имеют актуальное значение для многих наук и направлений: инженерия, медицина, территориальное распространение каких-либо явлений. Для нас, конечно же, актуальным является распространение инфекций.

## Теория клеточных автоматов

Принципы клеточных автоматов впервые были озвучены в работах Джона фон Неймана и Конрада Цузе примерно в 40-х годах прошлого столетия[[4]](#_Литература). Учёные представляли их как универсальную среду вычисления для построения алгоритмов, не уступающими по возможностям и машине Тьюринга. Эта идея породила многочисленные теоретические и прикладные исследования, в основном по созданию формальных алгоритмов и моделей на основе локальных взаимодействий, нейрокомпьютеров, а также клеточных процессов. И по сей день регулярно проводятся международные саммиты по обработке информации с помощью клеточных автоматов. Интерес к данному направлению только усиливается из-за существования возможности реализации на СБИС с большой степенью интеграции, а также обработки информации на молекулярном уровне.  
 Клеточные автоматы являются дискретными динамическими системами, поведение которых полностью определяется в терминах локальных зависимостей, в значительной степени так же обстоит дело для большого класса непрерывных динамических систем, определенных уравнениями в частных производных. В этом смысле клеточные автоматы в информатике являются аналогом физического понятия "поля". Клеточный автомат может мыслиться как стилизованный мир. Пространство представлено равномерной сеткой, каждая ячейка которой, или клетка, содержит несколько битов данных; время идет вперед дискретными шагами, а законы мира выражаются единственным набором правил, скажем небольшой справочной таблицей, по которой любая клетка на каждом шаге вычисляет свое новое состояние по состояниям ее близких соседей. Таким образом, законы системы являются локальными и повсюду одинаковыми. Если задан подходящий набор правил (рецепт), то такой простой операционный механизм достаточен для поддержания целой иерархии структур и явлений. Клеточные автоматы дают полезные модели для многих исследований в естественных и вычислительных науках и комбинаторной математике; они, в частности, представляют естественный путь изучения эволюции больших физических систем. Клеточные автоматы к тому же образуют общую парадигму параллельных вычислений, подобно тому как это делают машины Тьюринга для последовательных вычислений. Клеточные автоматы применимы в различных сферах изучения, в физике, химии, биологии, информатики и так далее[[5]](#_Литература). Одномерные клеточных автоматы могут моделировать процесс горения, распространение тепловых потоков, динамику развития доменных зон, даже макет движения толпы и многого другого. Если есть задача алгоритмической разрешимости какого-либо процесса, клеточные автоматы помогут в ее решении.   
 Итак, КА описывают системы состоящую из клеток. Задана **окрестность** – соседи, которые оказывают влияние на клетку т.е. от них зависит смена состояния в клетке. **Правила** – условия, которые трактуют смену состояний в клетке. Правила бывают разные: простые – создаются на базе условий, например, сравнений суммы клеток в состоянии 1 в окрестности с заданным числом; сложные – которые имеют более сложный вид. Такие правила могут привязываться ко времени и описывать будущее не только на основе настоящего, как это делают простые правила, но и на основе прошлого. Все клетки меняют состояния одновременно, чтобы не было неопределённости. Время и пространство – дискретны. Количество состояний у клетки всегда конечно, иначе мы не сможем добраться до состояния за конечное число шагов. На клетку влияют только клетки из её окрестности, то есть все процессы в КА локальны. Клеточные автоматы очень сильно привязаны к начальным условиям, которые определяют в каком состоянии находится каждая клетка на старте, эксперименты показывают, что это один из самых больших недостатков. Для клеточных автоматов создано специальное аппаратное оборудование, продаваемое на рынке, которое позволяет быстро моделировать какие-либо процессы, за счёт архитектуры самих устройств и распараллеливания процессов, но такие устройства не являются дешёвыми, и их цена растёт с ростом числа соседей, которых мы хотим учитывать при формировании правил.

У клеточных автоматов есть огромное количество разновидностей: одномерные, двумерные и автоматы больших размерностей, вероятностные, детерминированные, однородные, неоднородные, подвижные или неподвижные, синхронные или асинхронные и так далее. Подвижные автоматы могут изменять положения клетки в решетке во время развития системы, неподвижные же не могут. В детерминированных каждое новое состояние определяется состоянием ее данной ячейки и ее окрестных соседей в предыдущий момент времени. Каждому значению параметра в данном автомате каждому параметру соответствует конкретное число (целое, вещественное, комплексное) или соответствующая функция. В вероятностных КА одна и та же ситуация может привести к нескольким различным результатам с соответствующей вероятностью каждого из них. В синхронных автоматах все клетки единовременно переходят в новое состояние по сигналу глобального отсчета времени. То есть в качестве входных данных используются старые состояния соседних клеток. Асинхронные клетки переходят в новое состояние в случайном порядке, но новое состояние клетки сразу может использоваться ее соседями как входное.

Идеальным примером работы клеточного автомата является игра "Жизнь", опубликованная математиком Джоном Конвеем в 1970 году[[7]](#_Литература). Она описывает популяцию стилизованных организмов, где существуют этапы размножения и вымирания. Индивидуум данной игры представлен клеткой с состоянием 1, в то время как 0 представляется как пустое пространство. То есть «живые» и «мертвые» клетки. На каждом шаге каждая клетка реагирует на состояние ее окружения, которое состоит из 8 ближайших соседей.

В «Жизни» определены следующие правила. Клетка остается живой, только если окружена 2 или 3 живыми соседями, в противном случае наступает «смерть». Процесс «рождения» произойдет (мертвая клетка станет живой), когда ячейка будет окружена тремя живыми соседями. КА похожие на игру «Жизнь» находят применение в экологических и биологических задачах.

С помощью клеточных автоматов можно моделировать и исследовать физические, биологические и химические процессы. Примером можно назвать моделирование роста кристаллов либо поведения идеального газа. Но в нашем случае КА помогут в решении эпидемиологической задачи.

## Линейная алгебра.

Довольно популярной практикой на сегодняшний день является векторизация, которая позволяет во много раз повысить производительность и скорость выполнения программ. В нашей программе это будет достигаться за счет того, что все переходы из одного состояния в другое, будут описаны на языки линейной алгебры, а именно с помощью замены условных конструкций на матричные произведения и их аналоги. Позже будет описаны программы: основанная на условных конструкциях и с большим числом циклов, и программа в которой не будет почти циклов, за счет особого представления системы и её компонентов.

## Теория вероятностей.

Теория вероятностей имеет очень широкое применение и тут она тоже нашла своё. Дело в том, что инфекция передаётся от носителя с какой-то случайно, то есть при контакте с больным можно как заразиться, так и нет. Также больной человек с какой-то вероятностью может умереть. В нашей программе количество дней вакцинации, приобретенного иммунитета после болезни, времени болезни и латентного периода являются константными значениями, то есть мы предполагаем, что человек ровно столько дней находился в определённом состоянии, но можно при помощи теории вероятностей для каждого из состояний определить своё распределение, задать параметры и смоделировать модели, возможно, что такая модель даже будет более приближенной к реальности.

## Другие методы моделирования.

Как уже упоминалось возможно моделирование только с помощью теории перколяции, но такая модель является слишком упрощённой и отдалённой от реальных ситуаций.

Возможно применение дифференциальных вычислений, но как показывает практика они плохо справляются с этой задачей, могут работать с упрощёнными моделями, хорошо даже с этим справляются, но люди пока не научились решать реальную многопараметровую задачу с помощью аппарата дифференциальных уравнений.

Модели, основанные на графах и различных обходах по ним, методы, основанные на многомерных случайных величинах с различными распределениями и на случайных процессах и т.д

# 4. Реализация программы.

Поскольку мы уже разобрались с теоретической базой, которая нам понадобится для построения модели, то перейдём к её созданию. Для начала разберём основные моменты программы – её ядро, дальше опишем программу, написанную через условную и векторизованную программу, поймём в чём различия и как они реализованы программно.

## 4.1 Описание программы с условной конструкцией

Наша система является двумерной, состоящая из клеток, каждая клетка может находится в одном из состояний:

* Умер
* Здоров, отсутствует какой-либо иммунитет, может заболеть.
* Болен
* Переболел, т.е. имеет временный иммунитет
* Вакцинирован, т.е. также имеет временный иммунитет
* Имеет абсолютный иммунитет, т.е. никогда не заболеет.

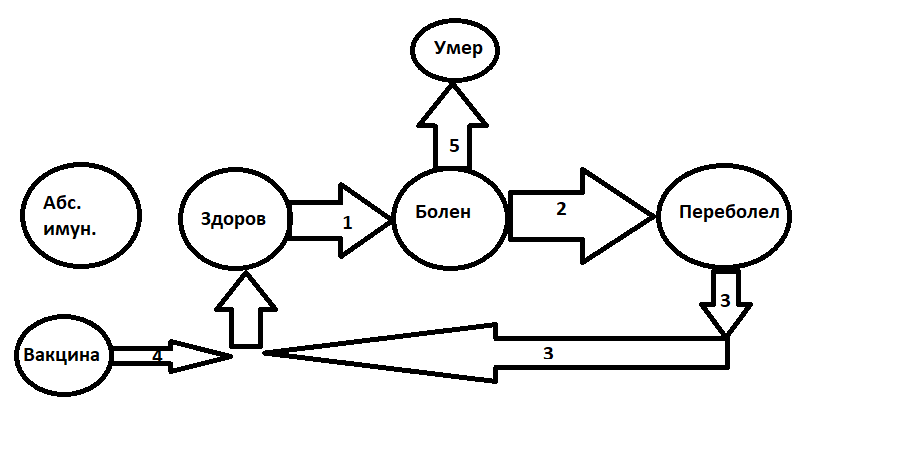


Рисунок 1

Схематично состояния и переходы между ними можно увидеть на рис.1

Опишем условия перехода из одного состояния в другое:

1: Человек является здоровым, мы просматриваем в его окрестности больных людей, и каждый больной человек может заразить нас с какой-то вероятностью p2. Мы генерируем случайную величину с равномерным случайным распределением из (0,1), для каждого больного она будет принимать своё значение. В будущем, для простоты будем генерировать случайную величину с такими параметрами, конечно, их можно заменить и тем самым усложнить модель, но пока что неизвестны распределения Сравниваем значения с p2, если оно меньше – значит человек заражён и теперь он болен, иначе он здоров.

2,5: Человек болен, но он не сразу находит болезнь и ограничивает контакты с другими людьми, а какое-то время ходит и заражает людей, т.е. существует латентный период = t4, до которого человек является носителем вируса. Если человек более больше чем латентный период, то он не является носителем вируса, но он всё ещё заражен – он может умереть. Человек болеет всего t1 дней – если спустя этот период он не умер – он переходит в состояние переболел и приобретает временный иммунитет. Во время болезни у человека каждый день есть вероятность умереть = p3. Каждый день, для каждого больного во время его болезни генерируем случайную величину и сравниваем с p3 – если значение меньше его, то человек умер, если больше – то остался живой.

3: У человека после болезни появляется иммунитет, на период времени t3. Каждый день сравниваем количество дней у человека, который переболел с этим числом, и если оно больше, то человек становится просто здоровым, без всякого иммунитета, если меньше, то увеличиваем число его дней после болезни.

4: В самом начале программы мы вакцинируем какую-то часть людей, у них также появляется временный иммунитет, но предел у него другой, не такой как у людей, которые переболели, для него существует своя переменная t2. По аналогии с прошлыми счётчиками, каждый день увеличиваем на 1, а затем сравниваем количество дней с вакциной с пределом. И осуществляем переход в состояние здоров, если он вакцинирован больше предела.

В итоге в программе есть следующие гиперпараметры:

**p1 –** вероятность, что человек обладает абсолютным иммунитетом.

**p2 –** вероятность заражения при контакте с больным.

**p3** – вероятность смерти человека, во время его болезни

**p4** – вероятность того, что человек на границе болен.

**t1** – предел времени болезни.

**t2**– предел времени временного иммунитета, после болезни

**t3** – предел времени временного иммунитета, после вакцинации.

**epocha** – длительности дней или количество итераций работы функции

**there\_is\_leak ()**

Перейдем к реализации поставленной задачи в программном коде.

Как мы уже поняли, в теории перколяции мы вычисляем какое-то критическое значение, в качестве него у нас будет выступать число людей с абсолютным иммунитетом, т.е. если у всех будет иммунитет, то никто не будет болеть, если ни у кого не будет иммунитета, то рано или поздно все заболеют, нам нужно будет определить при каком значении осуществляется переход системы из одного состояние в другое.

В качестве языка программирования был выбран Python[[8]](#_Приложения) из-за следующих преимуществ:

- Одной из задач данной работы является визуализация, в Python есть библиотека PyGame[[9]](#_Литература), которая позволит нам быстро и без особых проблем её реализовать.

- Быстрота написания кода, его простота чтения и популярность. Из-за популярности языка, в своё время я начал на нём специализироваться, что тоже сыграло свою роль в выборе языка.

- Библиотеки для работы с массивами, визуализации и векторизации:

Numpy[[10]](#_Литература), matplotlib и numba[[11]](#_Литература), которые опять же ускорили процесс написания кода, а также перекрыли самый главный минус языка – его производительность.

Для работы программы реализованы следующие функции:

**there\_is\_leak ()** – основная функция, с помощью которой мы определяем – есть протекание или нет. Для начала все клетки находятся в состоянии здоров. Далее мы генерируем население с абсолютным иммунитетом и вакциной в зависимости от соответствующих переменных.

Далее мы начинаем генерировать больных на границе – на первой строчке массива. Каждую итерацию мы будем генерировать новых больных. Считаем, что человек на границе болен с вероятностью **p4.**

Далее: на каждой итерации проходимся по каждой клетке и смотрим на её соседей в окрестности. И в зависимости от условий перехода обновляем состояние клетки. Причем создаётся временная копия массива, в которую будут записаны новые состояния, и в конце каждой итерации будет происходить обновление массива состояний.

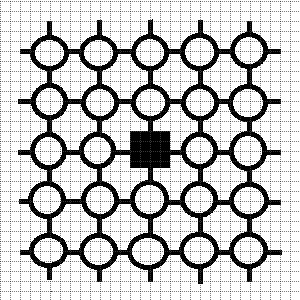


Рисунок 2

В качестве окрестности была выбрана решётка, изображённая на рисунке 2. Она имеет степень связи – 24. Т.е. на неё напрямую на каждой итерации влияют только эти клетки.

За это обновление состояний и счётчиков отвечает функция

**update\_cell()**. Основная разница между реализациями программы с условными конструкциями и векторизованной лишь в реализации этой функции. Эта разница будет описана в следующем подразделе.

В конце каждой итерации на последней строчке ищется больной человек, если он есть – значит протекание есть, если нету, цикл продолжается. Цикл заканчивается, когда проходит **epocha,** в конце которой возвращается False – протекания нету.

**calculate\_critical\_value ()** - считает критическое значение = процент населения с абсолютным иммунитетом. Осуществляется это всё с помощью метода дихотомии. Интервал (0, 1) делится пополам – подаётся в функцию

**there\_is\_leak ()** – определяется протекание, если его нету - то берётся левый подотрезок, если есть - то правый. Так продолжается до тех пор, пока мы не достигнем необходимой точности, в нашем случае eps = 0.001.

Очень много случайности в наших экспериментах, поэтому чтобы уменьшить дисперсию критического значения есть функция усреднения. В решении нашей задачи возьмём n = 10. Функция **calculate\_average\_critical\_value ()** будет считает среднее критическое значения, т.е. запускать функцию **calculate\_critical\_value ()** n раз, а затем суммировать результат работы каждой функции и делить на n.

## 4.2 Нелокальные контакты.

У каждого человека есть круг лиц, с которыми он находится в постоянном контакте – соседи, семья, друзья, коллеги и т.д. Но также у него есть непостоянные взаимодействия с людьми – в метро, в автобусе, в магазине и других общественных местах. Такие взаимодействия будем называть - **нелокальными контактами**. Одной из задач этой работы является определить влияние нелокальных контактов на протекание инфекционных заболевания.

Как это реализовано в программе из условных конструкций:

Мы генерируем случайные координаты в пределах двумерного массива столько раз, сколько нелокальных контактов, а затем уже действуем по описанным выше правилам. Стоить заметить, что нелокальные контакты влияют только на переход из здоров в болен.

## 4.3 Реализация векторизованной программы.

Как уже было написано выше, основные различия данной программы в реализации функции **update\_cell()**, а также в реализации нелокальных контактов, которое также будет описываться в ней.

В программе с условными конструкциями мы каждый раз проходим по всему двумерному массиву и смотрим на соседей, смотрим какое состояние в самой клетке, а уже потом действуем. Этих циклов и условий можно избежать, если правильно представить наши переменные.

Для каждого состояния заведём массив numpy, который будет иметь следующий смысл: 1 – состояние есть, 0 - состояния нет; для каждого из счётчиков: время болезни, время вакцинирования и время после болезни. Заметим, что человек может находиться только в одном из состояний, этим мы будем пользоваться, при помощи операций булевой алгебры. Поскольку мы работаем с массивами numpy, а не с обычными списками, у нас уже есть выигрыш в скорости. Все проходы по циклам заменены матричными умножениями, соответствующих состояний друг на друга. Например, работа с состоянием болен: обновляем время у тех, кто уже болен, это можно сделать так: у всех, у кого состояние 1 в массиве болен, добавляем время в массив со временем болезни, благодаря матричному сложению, счётчики времени болезни будут обновлены только у больных людей. Далее мы сравниваем действуем по правилу: берём тех, у кого время болезни меньше латентного срока пересекаем со множеством случайных величин, значения которых меньше вероятности смерти – это те люди, которые погибли от болезни. Далее пересекаем со множеством больных людей. Получили больных людей, которые умерли. Обновляем у этих людей состояния. Аналогично были переписаны другие переходы между состояниями. С помощью матричных перемножений и операций булевой алгебры.

Также была применена библиотека Numba, которая ещё позволила ускорить программу. Numba была задумана для работы с числовыми значениями и массивами numpy. Библиотека Numba генерирует оптимизированный машинный код с использованием определённого компилятора, таким образом, что он становится по производительности, сравним по скорости с кодом, написанным на С\С++[[12]](#_Литература).

В итоге разница в скорости выполнения между 2 программами минимум в 20 раз. Чем больше итераций и повторений одной и той же функции, тем больше разница в скорости выполнения между ними.

У векторизации также есть и свои требования: нужно точно понимать программу, которую нужно реализовать, сформулировать правила перехода между состояниями, уметь работать с массивами numpy и понимать к чему приводят те или иные операции, иначе просто не получится реализовать нормально работающую программу. В процессе перехода от 1 программы ко второй была реализована визуализация процесса протекания болезни, которая помогла совершить переход между программами, и позволила убедиться в адекватности работы обеих программ.

## 4.4 Визуализация.

Библиотека PyGame помогла реализовать визуализацию без особых проблем, позволив избежать принципов визуализации и других низкоуровневых знаний в этой сфере.

Создана функция **visualization()**, которая при необходимости вызывается в функции **there\_is\_leak ().** На каждой итерации работы этой функции происходит перерисовка пикселей в зависимости от состояния клетки:

- здоров – зелёный

- мёртв – чёрный

- болен – красный

- временный иммунитет - голубой

- абсолютный иммунитет – жёлтый

Библиотека является очень высокоуровневой, поэтому почти никаких вычислений не нужно, необходимо вызвать лишь встроенные функцию по прорисовке прямоугольника – в которую будут поданы границы фигуры и её цвет.

Примеры визуализации изображены ниже на рисунках 3-5.

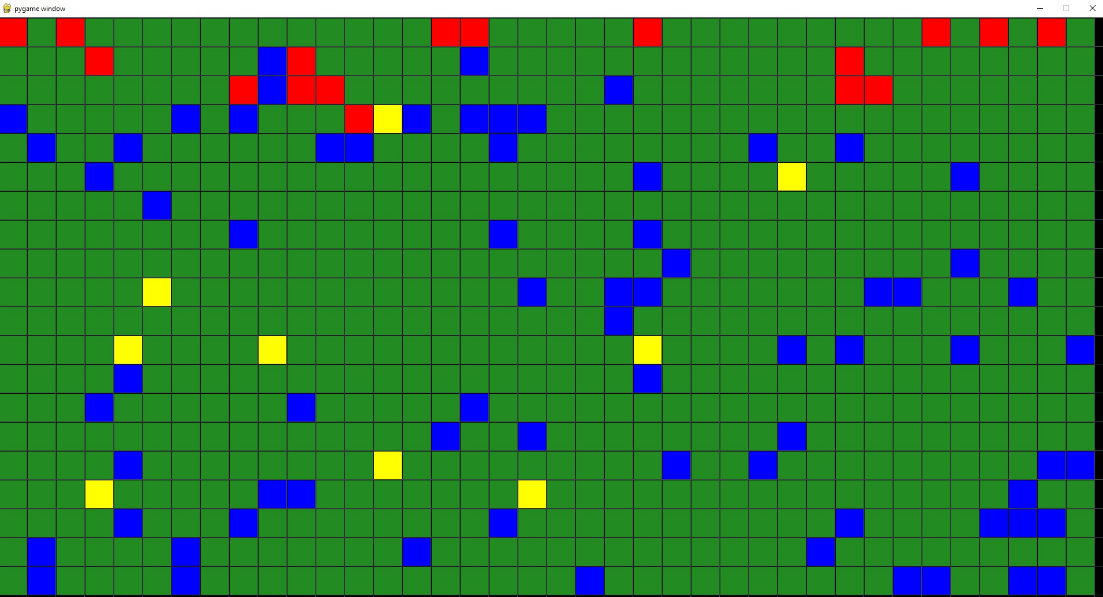


Рисунок 3

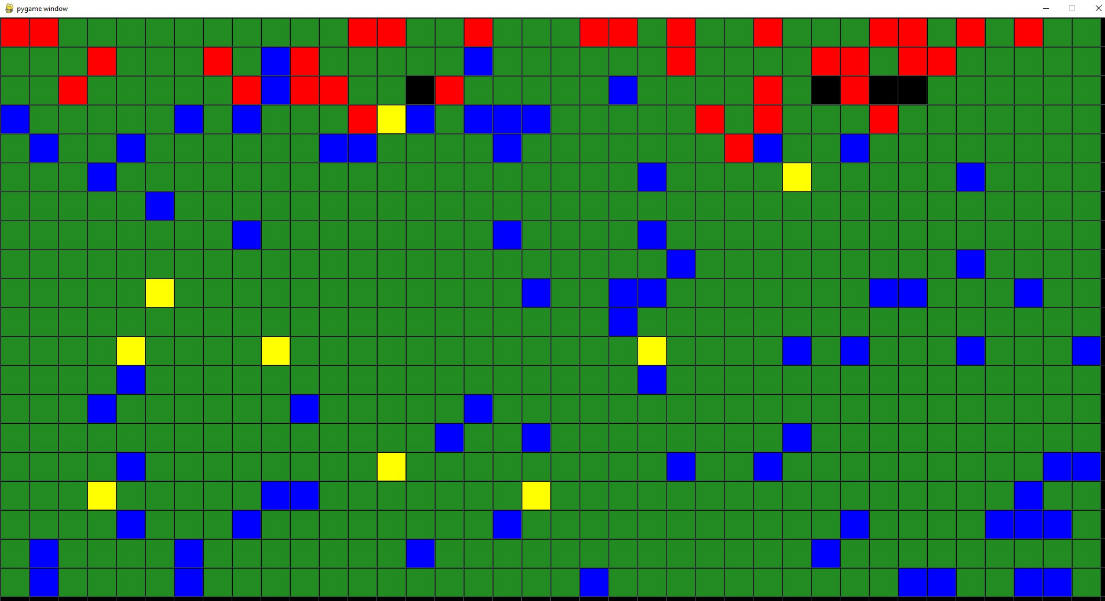


Рисунок 4

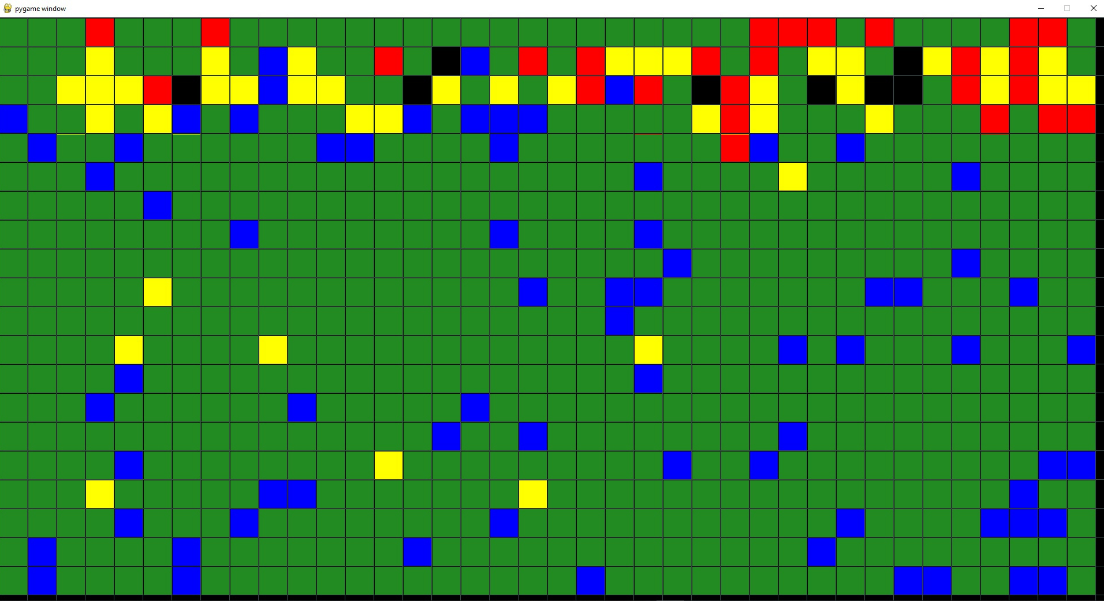


Рисунок 5

# Проведение и анализ экспериментов.

## 5.1 Эксперимент 1.

Проведём следующий эксперимент: при фиксированных параметрах, меняя лишь размерность задачи, посчитаем критические значения, а затем построим графики, изображённые на Рисунок 6-8.

Заметим, что точки расположены равномерно, с небольшим количеством аномальных результатов. Разброс точек постепенно уменьшается, что говорит, о том, что если провести эксперименты много раз, а затем усреднить результаты, то точки будут лежать ещё плотнее – и рост размерности будет мало отражаться на росте критического значения. Теория перколяции утверждает, что на больших размерностях процесс стабилизируется и значения не сильно друг от друга отличаются. И я думаю графики это отражают.

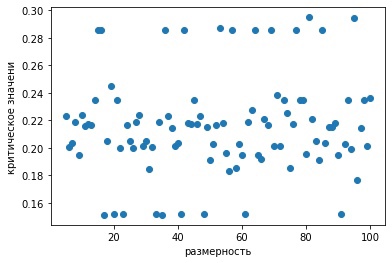


Рисунок 6

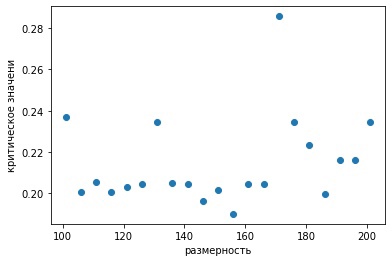


Рисунок 7

## 5.2 Эксперимент 2.

Теперь, определим, как количество контактов влияет на такой показатель, как смертность населения.

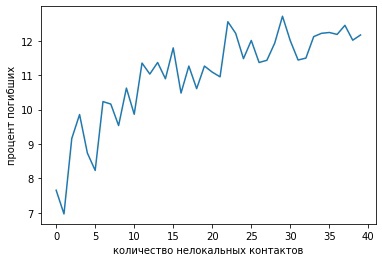


Рисунок 8

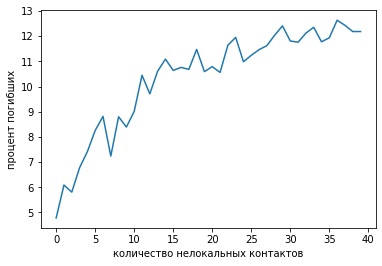


Рисунок 9



Рисунок 10

В диапазоне (0, 40) числа нелокальных контактов при заданных параметрах определяем существует ли протекание при уровне населения с абсолютным иммунитетом = 0.05. При этом подсчитывая количество погибших на самой последней итерации функции, проводим такой эксперимент 10 раз, а затем усредняем число погибших для каждого нелокального контакта и получаем рисунки 6-8.

Анализ графиков показывает, что с ростом числа нелокальных контактов – растёт смертность. Это может означать, что слабые карантинные ограничения, плотность населения, развитость индустрии – оказывают влияние на смертность. Чтобы её понизить мы можем только увеличить карантинные ограничения, тем самым снизить контакты людей между собой, которые носят не постоянный характер. Очевидно, что так мы не особо влияем на матрицу связей, которая описывает взаимодействия с людьми, которыми вы видитесь постоянно. На неё мы можем повлиять только при введении очень жёстких ограничений, которые зачастую экономически не выгодны для государства.

## 5.3 Эксперимент 3.

Посмотрим теперь, как число нелокальных контактов влияет на критическое значение, т.е. на процент людей с абсолютным иммунитетом.

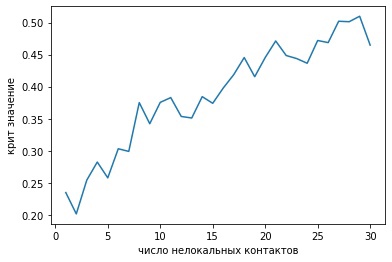


Рисунок 11



Рисунок 12

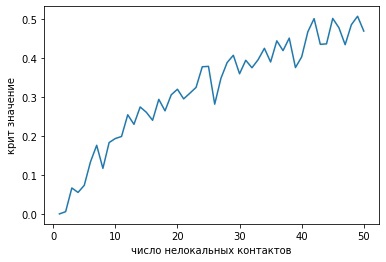


Рисунок 13

Анализ графиков показывает, что с ростом числа нелокальных контактов – критическое значение увеличивается, также можно заметить, что скорость роста функции падает, и в какой-то момент её значение становится почти константным. Это может означать, что если число нелокальных контактов очень большое, то есть люди часто контактируют друг с другом, то слабые ограничения почти никакого эффекта не дадут, поскольку критическое значение уже практически стабилизировалось, поэтому придётся вводить более жёсткие ограничения.

# Выводы и рекомендации.

При проведении экспериментов и создании программы были выявлены следующие недостатки в математической модели:

* Программа привязана к параметрам очень сильно, чтобы можно было построить графики зависимостей нужно правильно подбирать параметры, нужно уметь их интерпретировать и понимать, на что влияет изменение того или иного параметра. Например, если население умирает за 10 дней, то нет смысла даже смотреть на влияние числа нелокальных контактов, и таких примеров может быть очень много.
* При больших размерностях решётки вычисления занимают много времени, даже с учётом векторизации. К сожалению, я не смог провести вычисления при больших размерностях, в силу отсутствия больших машинных мощностей. Могу только предположить, опираясь на теорию перколяции, что процесс будет становится более устойчивым и показательным. Это проблему можно решить с помощью, созданных специальных устройств, которые уже существуют, но в более упрощенном виде. Но тут может возникнуть другая проблема – большие затраты на создание такого оборудования. Поскольку даже небольшое улучшение этого оборудования или модернизация, требует больших затрат, как финансовых, так и интеллектуальных.
* В модели много параметров нахождение, которых является затруднительным, вследствие чего может получится, что затраты несоизмеримо велики. Например, очень трудно посчитать количество людей с абсолютным иммунитетом, вероятности можно получить на основе математической статистики, но на это может понадобится очень большое количество времени и сил и др.

Какие были выявлены достоинства:

* Простота в освоении и не слишком большой багаж знаний (если не брать в счёт векторизацию программы), необходимый для построения или разбора уже существующей подобной модели. Все параметры и условия задачи легко переносятся и интерпретируются на реальную жизнь.
* Выводы из экспериментов сходятся с логическими и реальными представлениями о инфекционные заболевания, что может говорить о том, что модель способна в какой-то мере отображать реальную ситуацию.

Рекомендации при работе с данной моделью или её аналогами:

* Чем больше у вас мощностей, тем больше информации можно извлечь из анализа модели, так как все функции можно усложнить, хотя бы за счёт увеличения точности нахождения критического значения, количество проведения того или иного эксперимента или запуска функции, для усреднения.
* Нужно понимать, как параметры связаны между собой, как они могут влиять друг на друга, тогда будет понятно, для чего нужен подбор параметров, и почему он отражает закономерности, а не лишь частные случаи.
* Желательно иметь реальные значения параметров, для того, чтобы понять на сколько точно модель отражает реальную ситуацию и при необходимости подкорректировать её работу, если это возможно. Иначе постараться понять и обосновать эти различия.
* Ещё существует большое количество разнообразных экспериментов, которые можно провести, всё зависит от вашего понимания данной темы и вашей фантазии. При желании их можно провести и установить какие-то более интересные закономерности, которые являются менее очевидными.

# Заключение

В данной работе мы убедились, что теория клеточных автоматов способна помочь в создании более совершенной модели, в сравнении с теорией перколяции. Данная модель может помочь в решении актуальных проблем, но для этого нужно иметь доступ к реальным данным, необходимо взаимодействие со специалистами в этой области, для юстировки параметров и правильного соотношения реальных данных с параметрами модели. Необходимы большие компьютерные мощности для более точного моделирования и лучшей аппроксимации реальных процессов.

Таким образом, были проделаны все запланированные задачи В таблице 1. указаны следующие приобретенные компетенции:

|  |  |
| --- | --- |
| Компетенция и расшифровка | Освоенные навыки |
| ОК-1 Способность использовать основы философских знаний для формирования мировоззренческой позиции | Были поставлены цели и найдены пути их достижения исходя из основных физических закономерностей при реализации поставленной задачи. |
| ОК-2 Способность анализировать основные этапы и закономерности исторического развития общества для формирования гражданской позиции | Были проанализированы и оценены факторы и механизмы исторических изменений. |
| ОК-3 Способность использовать основы экономических знаний в различных сферах жизнедеятельности | Для достижения нулевой стоимости разработки были использованы только свободно распространяемые программные продукты (Jupiter Notebook, ). |
| ОК-4 Способность использовать основы правовых знаний в различных сферах жизнедеятельности | В проделанной работы был определен круг задач в рамках избранных видов профессиональной деятельности, сформирован ход решения задачи, исходя из имеющихся ресурсов. Были использованы знания, основанные на анализе нормативных актов, регулирующих отношения в различных сферах деятельности |
| ОК-5 Способность к коммуникации в устной и письменной формах на русском и иностранном языках для решения задач межличностного и межкультурного взаимодействия | В составлении текста работы были применены литературная форма государственного языка, основы устной и письменной коммуникации на иностранном языке, функциональные стили родного языка, требования к деловой коммуникации.  В ходе написания работы в научном стиле была применена расшифровка научных слов с помощью словаря для того, чтобы правильно употреблять базовую лексику. |
| ОК-6 Способностью работать в команде, толерантно воспринимая социальные, этнические, конфессиональные и культурные различия | Изучены правила и принципы эффективного взаимодействия в коллективе, проведена работа вместе с научным руководителем, принятие поправок и замечаний. Совместно с другими студентами был изучен материал по Теории перколяции и клеточных автоматов, выявлены основные принципы действия процессов данной теории. |
| ОК-7 Способность к самоорганизации и самообразованию | Был самостоятельно проведен обзор и поиск современной литературы на тему теории перколяции и клеточных автоматов. Были изучены различные методы построения графических решеток и составления клеточных решеток. Были применены современные технологии (Python, Numpy, Numba) для реализации построения математической модели. |
| ОК-8 Способностью использовать методы и средства физической культуры для обеспечения полноценной социальной и профессиональной деятельности | Умение восстанавливать работоспособность головного мозга и применять средства физической культуры для разгрузки нервной системы. |
| ОК-9 Способность использовать приемы первой помощи, методы защиты в условиях чрезвычайных ситуаций | Изучены основы безопасности жизнедеятельности, телефоны служб спасения. Изучены правила оказания помощи при долговременном проведении работы за компьютером. |
| ОПК-1 Способность использовать базовые знания естественных наук, математики и информатики, основные факты, концепции, принципы теорий, связанных с прикладной математикой и информатикой | Были изучены основные положения и концепции в области математических и естественных наук, в большей степени получены углубленные знания в области клеточных автоматов, изучена основная терминология. Произведён анализ изученного материала и интерпретация математических объектов. |
| ОПК-2 Способность приобретать новые знания, используя современные образовательные и информационные технологии | Для успешного изучения материала и написания работы было распланировано рабочее время и время для саморазвития, сформулированы цели личностного и профессионального развития и условия их достижения. Поддерживалась связь с руководителем, были скорректированы и обсуждены цели, задачи, научная литература. |
| ОПК-3 Способность к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей, созданию информационных ресурсов глобальных сетей, образовательного контента, прикладных баз данных, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям | Для реализации поставленной задачи был написан программный код, содержащий несколько модулей. Модель клеточных автоматов была создана с помощью языка программирования Питон, также с помощью этой платформы была произведена оптимизация времен работы программы, из-за применения векторизации. |
| ОПК-4 Способностью решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности | Были использованы информационно-коммуникационные технологии, информационные в решении профессиональных задач. Входе работы были применены знания структур данных и линейной алгебры. Использованы приёмы из документации Numpy, Numba. |
| ПК-1 Способность собирать, обрабатывать и интерпретировать данные современных научных исследований, необходимые для формирования выводов по соответствующим научным исследованиям | Изучение технической документации, научно-исследовательских работ по схожей теме и литературы. Проведен поиск необходимых материалов по теме в Интернете |
| ПК-2 Способность понимать, совершенствовать и применять современный математический аппарат | Были изучены и применены математический аппарат и инструментальные средства для обработки, анализа и систематизации информации по теме исследования.  Реализована вычислительная часть приложения, с последующей проверкой и отладкой верности вычислений и переводов данных. |
| ПК-3 Способность критически переосмысливать накопленный опыт, изменять при необходимости вид и характер своей профессиональной деятельности | Была построена программа для моделирования процесса протекания инфекционных заболеваний, основанная на теории клеточных автоматов. Были проведены эксперименты и получены данные, отображающие различные закономерности. |
| ПК-4 Способность работать в составе научно-исследовательского и производственного коллектива и решать задачи профессиональной деятельности | Во время работы поддерживалась связь с научным руководителем, были определены цели и задачи работы, были рассмотрены различные виды кристаллических решеток, используемых в современных исследованиях. |
| ПК-5 Способность осуществлять целенаправленный поиск информации о новейших научных и технологических достижениях в сети Интернет и из других источников | Был произведен поиск актуальной литературы в сети Интернет для применения на практике работу программы. Был проведен анализ найденной литературы, сделанные выводы были отражены в проделанной работе. |
| ПК-6 Способность формировать суждения о значении и последствиях своей профессиональной деятельности с учетом социальных, профессиональных и этических позиций | В результате поиска современной литературы по статистике распространения инфекционных заболеваний был освоен навык понимания параметров модели и их интерпретирования. В значительной мере была изучена специфика предмета научного познания, а именно анализа эпидемиологических показателей и уменьшения распространения болезни по территории. |
| ПК-7 Способность к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения | Написан программный код, содержащий несколько модулей, была реализована визуализация процесса протекания, а также векторизация программы. |
| ПК-8 Способность приобретать и использовать организационно-управленческие навыки в профессиональной и социальной деятельности | В ходе работы были проанализированы условия различных мнений при принятии организационно-управленческих решений и своей ответственности. |
| ПК-9 Способность составлять и контролировать план выполняемой работы, планировать необходимые для выполнения работы ресурсы, оценивать результаты собственной работы | В процессе выполнения работы был сформулирован план решения поставленных задач и выполнены отдельные трудовые функции, технологические операции и отдельные приемы технологических операций; организовано выполнение работ и контроль их качества в соответствии с требованиями технической документации и нормами времени на выполнение соответствующих работ. |
| ПК-10 Способность реализации решений, направленных на поддержку социально-значимых проектов на повышение информационной грамотности, обеспечения общедоступности информационных услуг | Была определена тематика работы, обладающая большой актуальностью и значимостью для общественности. Выявлено практическое и теоретическое применения работы. |
| ПК-11 Способность к планированию и осуществлению педагогической деятельности в конкретной предметной области (математика и информатика) | Были приобретены навыки планирования занятий, разработки и обновления планов обучения; разработки и обновления основных программ профессионального обучения и рабочих программ учебных предметов, курсов, модулей основных программ профессионального обучения. |
| ПК-12 Способность к организации педагогической деятельности с учетом специфики предметной области в образовательных организациях | Выполнено решение образовательных задач и отдельных трудовых функций, технологических операций. Поддерживалась связь с руководителем. Были изучены нормативные документы для оформления ВКР. |
| ПК-13 Способность применять существующие и разрабатывать новые методы и средства обучения | Получен практический опыт создания новых методов и средств обучения с помощью применения математического аппарата и информационных технологий. |

# Литература

1. [Ю. Ю. Тарасевич «Перколяция: теория, приложения, алгоритмы», Учебное пособие. М: Едиториал УРСС, 2002 -112 с.](file:///C:\Users\User\Desktop\Дипломная\be2e8f3.pdf)
2. [А. Л. Эфрос “Физика и геометрия беспорядка” (Библиотечка “Квант”, выпуск 19), М., Изд. “Наука”, Гл. редакция физ.-мат. Литературы, 1982г. – 270с.](file:///C:\Users\User\Desktop\Дипломная\EFROS%20(2).pdf)
3. Курсовая работа «Применение методов теории перколяции для исследования процессов распространения эпидемиологических заболеваний. Определение эпидемиологического порога посредством плоских регулярных решеток степени 24» Анцырев А. С.
4. Тоффоли Т., Марголус Н. Т63 «Машины клеточных автоматов»: Пер. с англ. - М.: Мир, 1991. - 280 с, ил.
5. Гулд Х., Тобочник Я. «Компьютерное моделирование в физике»: Пер. с англ. Т.1,2, 1990. 752 с. ISBN 5-03-001593-0
6. Игра жизнь [Электронный ресурс] — URL: https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%B0\_%C2%AB%D0%96%D0%B8%D0%B7%D0%BD%D1%8C%C2%BB
7. Python documentation [Электронный ресурс] — URL: <https://www.python.org/doc/>
8. Pygame documentation [Электронный ресурс] —URL: https://www.pygame.org/docs/
9. Numpy documentation [Электронный ресурс] —URL: https://numpy.org/doc/
10. Numba documentation [Электронный ресурс] —URL: https://numba.pydata.org/numba-doc/latest/index.html
11. Краткое введение в numba [Электронный ресурс] —URL: https://pythobyte.com/numba-9f4c959a/

# Приложения

%%

import pygame

import random

import numpy as np

import pandas as pd

from copy import deepcopy

from numba import jit

#%%

epoch = 365 # T

limit\_time\_illness = 21 # T1

limit\_time\_vaccine = 90

limit\_time\_immun\_con = 90 # T2

probability\_immun\_abs = 0 # p1

probability\_infect = 0.01 # p2

probability\_lethal = 0.01 # p3

probability\_virus\_carrier = 0.3 # p4

time\_found\_illness = 7

#%%

matrix\_offsets = np.array(

[[1,1,1,1,1],

[1,1,1,1,1],

[1,1,1,1,1],

[1,1,1,1,1],

[1,1,1,1,1]])

dy = matrix\_offsets.shape[0] // 2

dx = matrix\_offsets.shape[1] // 2

N = matrix\_offsets.shape[0]

dict\_of\_colours = {0: "black",

1: "forestgreen",

2: "red",

3: "blue",

4: "yellow"}

#%%

dx, dy

#%%

data = {"вероятность заражения при контакте" : [],

"вероятность смерти во время болезни": [],

"процент людей с иммунитетои" : [],

"число нелокальных контактов": [],

"население": [] ,"число живых" : [],

"процент мёртвых": []}

df = pd.DataFrame(data=data)

df

#%%

info = pd.DataFrame(data = data)

info

#%%

def gameOver(surface):

surface.fill(pygame.Color('black'))

font = pygame.font.SysFont("comicsansms",32)

red = (255, 0, 0)

green = (0, 255, 0)

words = font.render("Вирус захватил мир!", 1, red, green)

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

surface.blit(words, (800,500))

pygame.display.update()

#%%

def generate\_left\_border(states):

rand\_vector = np.random.sample(states.shape[2])

states[1][0] = ((states[1][0] + rand\_vector) <= probability\_virus\_carrier) + 0

#%%

def generate\_imun(states, add\_abs\_imun = 0, percent\_vaccinated\_people = 0):

rand\_matrix = np.random.sample((states.shape[1]-1,states.shape[2]))

states[2][1:states.shape[1]] = ((states[2][1:states.shape[1]] + rand\_matrix) <=

probability\_immun\_abs + add\_abs\_imun ) + 0

rand\_matrix = np.random.sample((states.shape[1]-1,states.shape[2]))

states[3][1:states.shape[1]] = ((states[3][1:states.shape[1]] + rand\_matrix) <=

percent\_vaccinated\_people) + 0

#%%

# TODO:

def create\_info(number, states, add\_abs\_imun, nonlocal\_contact):

new\_row = [probability\_infect, probability\_lethal,

add\_abs\_imun , nonlocal\_contact, states.shape[2]\*states.shape[1],

(states.shape[2]\*states.shape[1] - np.sum(states[0]) ) /states.shape[2]/states.shape[1]\* 100 ,

np.sum(states[0])/states.shape[2]/states.shape[1] \*100]

df.loc[len(df)] = new\_row

return True

#%%

def visualisation(states, surface, WIDTH, HEIGHT, TILE):

surface.fill(pygame.Color('black'))

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

[pygame.draw.line(surface, pygame.Color('darkslategray'), (x, 0), (x, HEIGHT))

for x in range(0, WIDTH, TILE)]

[pygame.draw.line(surface, pygame.Color('darkslategray'), (0, y), (WIDTH, y))

for y in range(0, HEIGHT, TILE)]

for x in range(0, states.shape[2]):

for y in range(0, states.shape[1]):

#жив - зелёный

colour = dict\_of\_colours[1]

# мёртв - чёрный

if states[0][y][x] == 1:

colour = dict\_of\_colours[0]

# болен - красный

elif states[1][y][x] == 1:

colour = dict\_of\_colours[2]

# приобретённый имун - жёлтый

elif states[4][y][x] or states[3][y][x]:

colour = dict\_of\_colours[4]

# абс имун - синий

elif states[2][y][x]:

colour = dict\_of\_colours[3]

pygame.draw.rect(surface, pygame.Color(colour),

(x \* TILE + 2, y \* TILE + 2, TILE - 2, TILE - 2))

#%%

@jit(nopython=True)

def update\_cell(states, times, nonlocal\_contact = 1):

# если мёртв и абс имун - скип

# если болен

#return 0

H = states.shape[1]

W = states.shape[2]

times[0] = times[0] + (times[0] != 0)

# если умер

rand\_matrix = np.random.sample((H,W))

died\_people = (times[0] > 0)\*(times[0] <= limit\_time\_illness) \* (rand\_matrix <= probability\_lethal) \* states[1]

states[0][1:H] += died\_people[1:H]

# если переболел

recovered\_people = (times[0] > limit\_time\_illness) + 0

states[4] += recovered\_people

times[2] += recovered\_people

times[0] = times[0] \* (died\_people == 0) \* (recovered\_people == 0)

states[1] = states[1] \* (died\_people == 0) \* (recovered\_people == 0)

# если переболел

times[2] = times[2] + (times[2] != 0)

with\_imun = (times[2] <= limit\_time\_immun\_con) \* (times[2] > 0 )

states[4] = states[4] \* with\_imun

times[2] = times[2] \* with\_imun

# если вакцинирован

times[1] = times[1] + (times[1] != 0)

with\_vaccine = (times[1] <= limit\_time\_vaccine)

states[3] = states[3] \* with\_vaccine

times[1] = times[1] \* with\_vaccine

# если здоров

infected\_people = np.zeros((5,H+2\*dy,W+2\*dx))

infected\_people[:,dy:H+dy,dx:W+dx] = states

width\_time\_0 = np.zeros((H+2\*dy,W+2\*dx))

width\_time\_0[dy:H+dy,dx:W+dx] = times[0]

# np.sum(states[1][1:H])

number\_infected\_people = np.sum(states[1]) / (W\*H - H) \* nonlocal\_contact\* probability\_infect

for x in range(dx,W+dx):

for y in range(dy+1,H+dy):

random\_matrix = (np.random.sample((N,N)) <= (probability\_infect + number\_infected\_people))\*matrix\_offsets

random\_matrix = random\_matrix\*(width\_time\_0[y-dy:y+dy+1,x-dx:x+dx+1] <= time\_found\_illness)

random\_matrix = random\_matrix\*np.min((states[1:5,y-dy,x-dx] == 0))

random\_matrix = random\_matrix\*(infected\_people[0][y-dy:y+dy+1,x-dx:x+dx+1] == 0)

random\_matrix = random\_matrix\*infected\_people[1][y-dy:y+dy+1,x-dx:x+dx+1]

random\_matrix = random\_matrix\*(infected\_people[3][y-dy:y+dy+1,x-dx:x+dx+1] == 0)

random\_matrix = random\_matrix\*(infected\_people[2][y-dy:y+dy+1,x-dx:x+dx+1] == 0)

random\_matrix = random\_matrix\*(infected\_people[4][y-dy:y+dy+1,x-dx:x+dx+1] == 0)

states[1][y-dy][x-dx] += np.max(random\_matrix)

times[0][y-dy][x-dx] += np.max(random\_matrix)

#%%

#there\_is\_leak(show = False)

#%%

def there\_is\_leak(add\_abs\_imun = 0.1, number = 0, nonlocal\_contact=1,

percent\_vaccinated\_people = 0.01,

WIDTH = 2000, HEIGHT = 2000, show = False, info = False):

TILE = 40

if show:

WIDTH = 1920

HEIGHT = 1020

TILE = 50

pygame.init()

RES = WIDTH, HEIGHT

surface = pygame.display.set\_mode(RES)

clock = pygame.time.Clock()

W, H = WIDTH // TILE, HEIGHT // TILE

FPS = 1

n = 5 # количество состояний

# 0 - жив/ мёртв

# 1 - болен

# 2 - абсолютный иммунитет

# 3 - вакцинирован

# 4 - переболел

states = np.zeros((n,H,W))

# 0 - время болезни

# 1 - время вакцины

# 2 - время после болезни

times = np.zeros((3,H,W))

generate\_imun(states,add\_abs\_imun , percent\_vaccinated\_people)

iter\_count = 1

while iter\_count <= epoch:

generate\_left\_border(states)

if show:

visualisation(states, surface, WIDTH, HEIGHT, TILE)

pygame.display.flip()

clock.tick(FPS)

update\_cell(states, times, nonlocal\_contact)

if np.max(states[1][H-1]) == 1:

if info:

create\_info(number, states, add\_abs\_imun, nonlocal\_contact)

return False

iter\_count +=1

if info:

create\_info(number, states, add\_abs\_imun, nonlocal\_contact)

return True

#%%

def calculate\_critical\_value(number = 0, nonlocal\_contact = 1, eps = 0.0001, info = True):

start, current, end = 0, 0.5, 1

i = 0

while abs(start - end) >= eps:

i += 1

if not(there\_is\_leak(number = number, nonlocal\_contact = nonlocal\_contact,

add\_abs\_imun = current, info = info)):

current, start = (current + end) / 2, current

else:

current, end = (start + current) / 2, current

return current

#%%

def calculate\_average\_critical\_value(nonlocal\_contact, repeats = 1, eps = 0.001):

critical\_value = sum([calculate\_critical\_value(number = i, nonlocal\_contact = nonlocal\_contact)

for i in range(repeats)]) / repeats

return critical\_value

#%%

def there\_is\_leak\_repeats(add\_abs\_imun = 0.05, nonlocal\_contact = 1, repeat = 10):

data = {"вероятность заражения при контакте" : [],

"вероятность смерти во время болезни": [],

"процент людей с иммунитетои" : [],

"число нелокальных контактов": [],

"население": [] ,"число живых" : [],

"процент мёртвых": []}

global df

df = pd.DataFrame(data=data)

[there\_is\_leak(add\_abs\_imun = add\_abs\_imun ,nonlocal\_contact = nonlocal\_contact,

info = True) for \_ in range (repeat)]

info.loc[len(info)] = (df.mean()).tolist()