МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра дифференциальных уравнений и системного анализа

МОДЕЛИРОВАНИЕ БИОЛОГИЧЕСКИХ СИСТЕМ С ПОМОЩЬЮ КЛЕТОЧНЫХ АВТОМАТОВ

Курсовая работа

Ключник Карины
Владимировны

студентки 2 курса,
специальность 1-31 03 09
Компьютерная математика
и системный анализ

Научный руководитель:
кандидат физ.-мат. наук,
доцент О. А. Лаврова

Минск, 2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Клеточные автоматы имеют многочисленные приложения. Однако, большинство моделей, которые описывают клеточные автоматы не носят какого-то практического применения. Например, игра «Жизнь». Безусловно, там строятся интересные структуры, но она не может решить проблемы, касающиеся прикладных задач.

Данная курсовая работа покажет, как можно использовать клеточные автоматы для моделирования тех процессов, которые происходят в природе. В работе будет взята биологическая система на примере популяции. Таким образом, это будет значит то, что клеточные автоматы найдут еще одно применение, а именно, использование клеточных автоматов в прогнозировании развития популяций.

Курсовая работа носит исследовательский характер. Целью работы является изучение применения математических инструментов в биологии, а также изучение различных способов моделирования.

Основными задачами курсовой работы являются изучение и реализация с помощью разных подходов биологической системы, показывающей отношение между организмами, а также между организмами и средой (питательная среда – одноклеточные; питательная среда – одноклеточные – многоклеточные), а также сравнение построенной модели с математической моделью «хищник-жертва».

ГЛАВА 1
ОБЩИЕ ПОЛОЖЕНИЯ

1.1 Клеточные автоматы

Клеточные автоматы – это дискретные динамические системы, каждая из ячеек которых находится в одном из конечного множества возможных состояний (например, если возможны только два состояния, то ячейка находится либо в состоянии 1, либо в состоянии 0). Пространство клеточного автомата представляет собой сетку, каждая ячейка которой несет в себе конкретную информацию.

Клеточный автомат представляет собой мир со следующими особенностями:

- изменяется дискретными шагами.
- его поведение задается некоторыми правилами, по которым для каждой клетки одновременно вычисляется значение на следующем шаге (т.е. по предыдущему значению клетки можно определить следующее значение).
- правила перехода к новому состоянию ячейки зависят только от локальных значений из некоторой окрестности.

Так законы, по которым изменяется клеточный автомат, являются повсюду одинаковыми и, вообще говоря, локальными (т.е. по состоянию ближайших ячеек и самой клетки можно определить то, что произойдет с ней на следующем шаге).[3]

Решетка клеточного автомата может быть любой размерности, эта решетка является совокупность клеток и называется клеточным пространством, в котором функционирует клеточный автомат. Также для клетки автомата определяется множество ячеек, которое является окрестностью данной клетки (так называемое соседство). Например, окрестностью могут являться все ячейки, расположенные выше, ниже, слева, справа и по диагоналям от самой клетки. Для каждой конкретной задачи определяется свой тип окрестностей. На рисунке 1.1 представлены основные из них.

Чтобы исключить особый вид окрестностей для граничных клеток, обычно правый край совмещают с левым, а верхний с нижним, получая тем самым тор. Такой прием в математическом моделировании называется заданием условий циклического типа. Это продемонстрировано на рисунке 1.2.

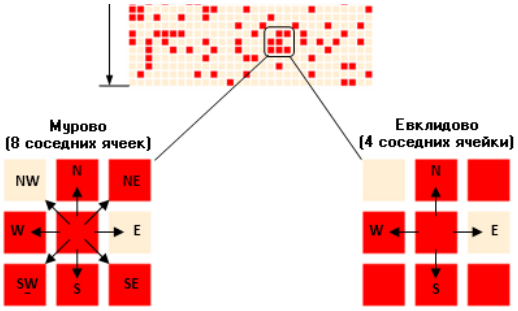

Рисунок 1.1 Типы соседства в дискретном пространстве. [9]

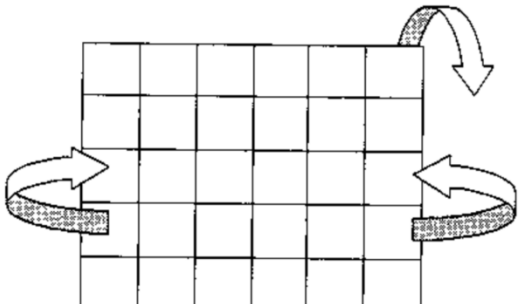

Рисунок 1.2 «Сворачивание» области. [1]

Наиболее развитое направление применения клеточных автоматов – это математическое моделирование динамических процессов. С помощью клеточных автоматов можно получать полезные модели для исследований в естественных и вычислительных науках. В настоящее время клеточные автоматы используются как вычислительный инструмент для широкого спектра различных задач. Они могут упрощать задачи там, где традиционные подходы приводят к сложным, требующим больших усилий вычислениям.

Также клеточные автоматы нашли широкое применение в следующих областях:

- криптография;
- моделирование физических процессов;
- и д.р.

1.2 Понятие биологической системы

Биологическая система – это объекты различный сложности, представляющие собой совокупность взаимодействующих и взаимосвязанных элементов. К биологическим системам относятся сложные системы разного уровня организации, начиная от микро- заканчивая макросистемами. Например, клетки, ткани, органы, организмы, популяции, виды, экосистемы, биосфера.

Биологические системы обладают совокупностью признаков, которые отличают их от неживой системы, перечислим те, которые актуальны для нашей задачи:

- обмен веществ и превращение энергии (т.е. живые существа питаются (поглощают питательные компоненты из окружающей среды), а также

имеют способность превращать компоненты поглощенной пищи в энергию, которую в последствии могут использовать на свои нужды);

- движение – возможность активного взаимодействия со средой, в том числе перемещение с места на место;
- рост и развитие;
- размножение – способность организмов воспроизводить себе подобных);
- эволюция – процесс формирования адаптаций, с помощью которых возможно гармоничное развитие организма/ов.

В данной курсовой будет рассмотрена биологическая система на примере популяции.

ГЛАВА 2
МОДЕЛЬ БИОЛОГИЧЕСКОЙ СИСТЕМЫ

2.1 Описание задачи

Как описано в первой главе, клеточные автоматы нашли широкое применение в математическом моделировании динамических процессов. Покажем, что при использовании клеточного автомата можно получить модель, соответствующую процессу, который может происходить в настоящей живой среде.

Задача взята из книги Трусова П.В. «Введение в математическое моделирование».

Итак, опишем задачу. Пусть есть раствор, который обладает питательностью. В этом растворе живут одноклеточные организмы (например, амебы, инфузории и т.п.), которые сами по себе обладают запасом энергии. Одноклеточные добывают энергию на свое существование из питательного раствора. Свою собственную энергию они тратят на свою жизнь, а также на размножение. Одноклеточные могут передвигаться, используя простейшие приспособления (например, реснички). Питательная среда, которую используют одноклеточные для своих нужд, имеет возможность восстанавливаться. Итак, мы хотим узнать, каким образом будет эволюционировать система «организмы-питательная среда» с течением времени.

Построим клеточный автомат, моделирующий описанную выше систему, который будет принимать во внимание следующие правила:

1. Размер пространства клеточного автомата 256x256 клеток.
2. Соседями организма являются 8 клеток вокруг него (такое соседство называется муровым и является альтернативой евклидова соседства).
3. Каждая клетки имеет значение p, которое соответствует степени питательности раствора, оно изменяется от 0 до $p_{max.}$
4. За каждый такт питательность раствора увеличивается на Δp следующим образом:

$$\Delta p = \begin{cases} 0 & p = p_{max} \\ r_p & p < p_{max} \end{cases}$$

5. Сумма питательности всех клеток P_{max} это общий запас энергии, который не может быть больше

$$P_{max} = 256^2 p_{max}$$

6. Общий запас энергии вычисляется по формуле

$$P = \sum_{x=1}^{256} \sum_{y=1}^{256} \frac{p_{xy}}{P_{max}}$$

7. Клетка либо содержит не более одного организма, либо является пустой.

8. За каждый такт особь берет от питательного раствора Δp_N, т.е. снижает питательность раствора на Δp_N и повышает собственный запас энергии на Δp_N.

9. Особь может накапливать до p_N единиц энергии.

10. За каждый такт особь тратит на свои нужды Δe_N единиц энергии.

11. На каждом такте особь перемещается на любую свободную соседнюю клетку (на начальном этапе не зависит от количества энергии в клетке, на которую перемещается особь).

12. Особь имеет продолжительность жизни L.

13. Если запас энергии особи равен 0 или же особь живет больше чем L, то она умирает.

14. Начиная с определенного возраста, равного l_3 особь считается достаточно взрослой и может производить себе подобных, при этом тратит на размножение Δr_N единиц энергии. Новорожденная особь занимает любую свободную соседнюю клетку, а особь родитель остаётся на той клетке, на которой была. Если свободных соседних клеток нет, то размножение не происходит.

15. Начальное число особей $N_o\%$ от максимально возможного. Распределение особей подчиняется равномерному закону распределения. Общее число особей находится по формуле

$$N = \frac{n}{N_{max}}$$

где $n -$ количество особей в момент времени, а $N_{max} = 256^2 = 65536$.

16. Мир считается торическим.

Сложность построения клеточного автомата, соответствующего данной системе, состоит в выборе коэффициентов, т.к. трудно подобрать такие значения, что наш клеточный автомат будет точно соответствовать тому, что происходит в реальной природе.

2.2 Математическая модель системы

Ситуация, которую мы хотим смоделировать, описывается с помощью системы дифференциальных уравнений. Они известны как уравнения Лотки-Вольтера, это так называемая система «хищник-жертва».

Опишем, что из себя представляет эта система. Система «хищник - жертва» - это сложная экосистема, для которой реализованы долгосрочные отношения между видами хищника и жертвы, это вид так называемой коэволюции (т.е. совместной эволюции, при которой изменения признаков одного вида особей приводят к изменениям признаков у другой особи). В данной задаче в качестве хищника выступают одноклеточные, а питательность раствора является жертвой.

В математической форме модель Лотки-Вольтера имеет следующий вид:

$$\frac{dP}{dt} = \left(r_p - c_1 N - bP\right)P,$$

$$\frac{dN}{dt} = (-r_n + c_2 P)N,$$

где r_p – скорость восстановления питательности раствора; r_n – скорость гибели одноклеточных; b – степень замедления скорости восстановления питательности раствора; c_1 и c_2 – коэффициенты взаимного влияния питательности раствора и численности одноклеточных.

Графики для модели «хищник-жертва», полученные из решения уравнений Лотки-Вольтера имеют вид затухающих колебаний, причем колебания количества «хищника» и «жертвы» должны изменяться в противофазе, т.е. когда становится много «хищников», число «жертв» снижается, и наоборот. Поэтому и для нашей модели, построенной с помощью клеточного автомата, мы ожидаем увидеть колебания (система должна найти для себя оптимальные значения параметров, при котором колебания минимальные, а потом продолжить эти колебания с одинаковой амплитудой и периодом). А график зависимости P_t от N_t имеет вид спирали, закручивающейся к точке положения равновесия (производная по обоим компонентам равна 0), координаты которой можно найти по следующим формулам:

$$P_p = \frac{r_n}{c_2}, \qquad N_p = \frac{r_p c_2 - b r_n}{c_1 c_2}$$

В самой упрощенной модели «хищник-жертва» не учитывается b, а b в нашей задаче – это степень замедления питательности раствора (глобально в таких системах b – это внутривидовая конкуренция). [4] Без учета b на графиках будут получаться не затухающие колебания, а с учетом b модель хоть и будет сложнее, но и будет более правдоподобно отражать процессы, происходящие в реальных биологических системах. [2]

Это сравнимо с процессами, происходящими в природе. Логично, что существа в живой природе постоянно изменяются под действием внешних факторов. Приспособления, вырабатываемые жертвами для противодействия хищникам, способствуют выработке у хищников механизмов преодоления этих приспособлений. Важно отметить, что нарушение полученного равновесия в системе приводит в серьезным экологическим последствиям в реальной ситуации.

Таким образом с помощью построенной модели мы сможем проследить изменения количества особей в популяции, что покажет нам аналогичный процесс в живой природе. Мы сможем спрогнозировать динамику популяций «жертв» и «хищников».

2.3 Реализация модели в MATLAB

С помощью высокоуровневого языка MATLAB реализуем описанную выше задачу. MATLAB – это сокращение от двух слов: Matrix Laboratory, что значит, что данный язык представляет собой мощный инструмент в работе с матрицами.

Реализуем описанную задачу матричным способом.

Мир организмов – это матрица 256x256 (1 правило моделируемой системы). С помощью тензора можем хранить всю информацию, которой обладает каждая клетка, а отображать будем только слой, который указывает на то, есть ли организм на данной клетке.

Имеем тензор 256x256x5.

1 слой тензора: энергия раствора.

2 слой: есть организм или нет (1 – есть, 0 – нет).

3 слой: сколько лет организму.

4 слой: запас энергии в организме.

5 слой: вспомогательный слой, показывающий размножался ли организм на этом шаге.

Итак, нам даны начальные значения, от которых отталкивается система:

Питательный раствор: $p_{max} = 10$

$r_p = 1$

Одноклеточный организм: $N_0 = 30\%$ $\Delta p_N = 5$ $\Delta r_N = 3$

$L = 15$ $p_N = 35$

$L_c = 3$ $\Delta e_N = 2$

Соответственно этим значениям напишем программу, которая будет работать по приведенным выше правилам.

Продемонстрируем работу полученной модели.

Начальное заселение показано на рисунке 2.1. Промежуточное состояние системы продемонстрировано на изображении 2.2. Стабильное состояние, к которому пришла биологическая система отражено на рисунке 2.3.

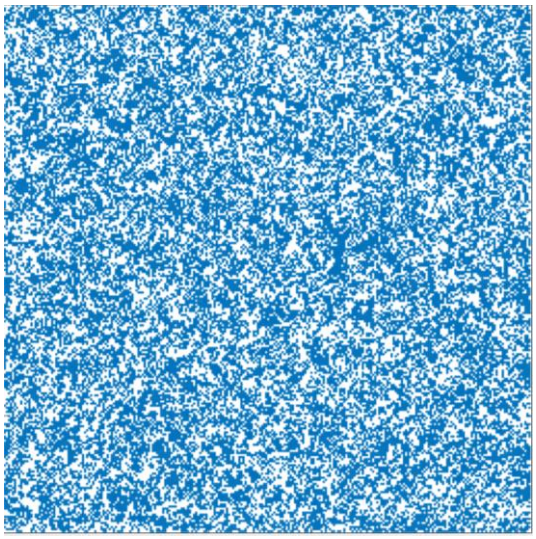

Рисунок 2.1 Начальное заселение.

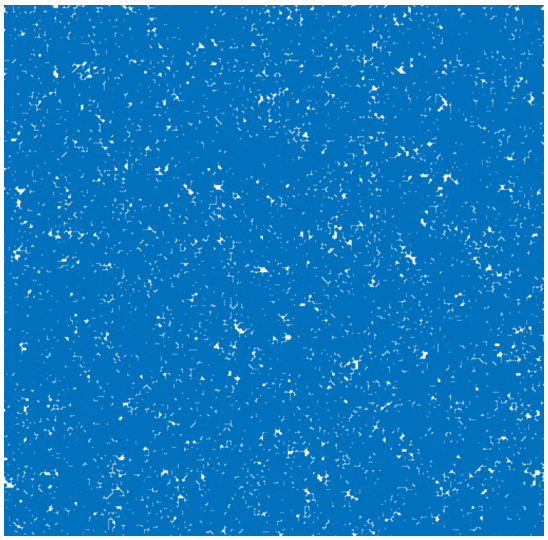

Рисунок 2.2 Промежуточное состояние системы.

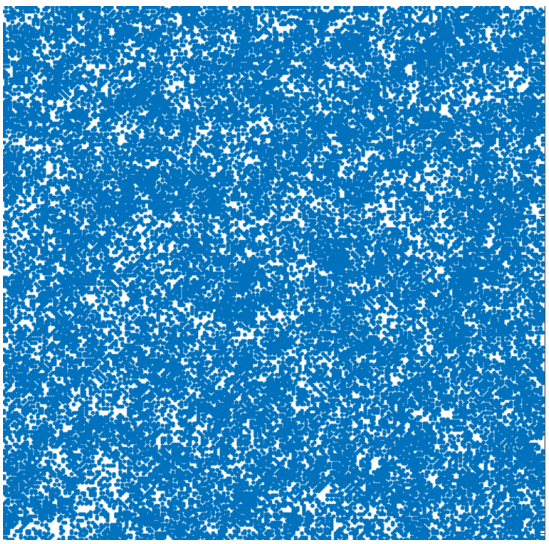

Рисунок 2.3 Стабильное состояние, к которому пришла система.

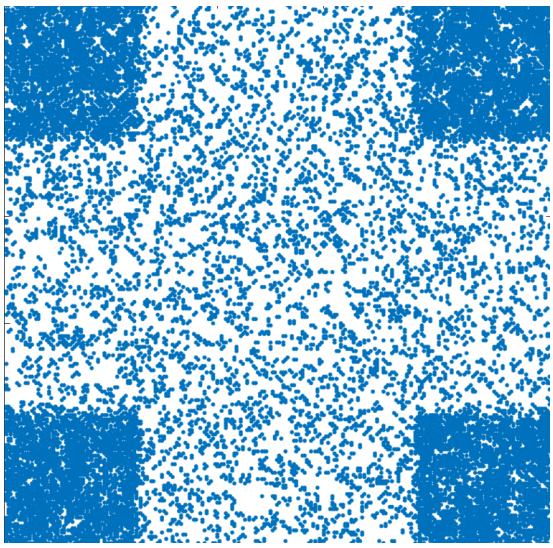

Рисунок 2.4 Локальное размещение энергии.

Если разместить на начальном этапе энергию локально, то на рисунке 2.4 можно увидеть, что в тех зонах, где нет энергии, наблюдается быстрая смерть живых организмов, а их местом обитания является пространство, где есть энергия.

Также можно посмотреть на поведение энергии, видно, что когда пространство густонаселено, количество энергии быстро сокращается. Состояние энергии при большом количестве одноклеточных отображено на

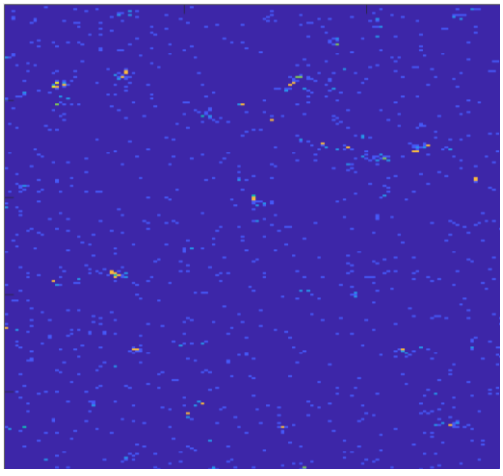

Рисунок 2.5. Энергия при большом числе одноклеточных.

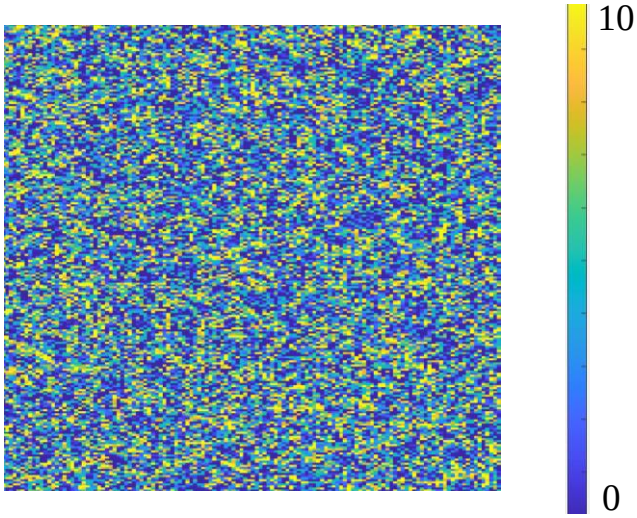

Рисунок 2.6. Энергия при уменьшении числа одноклеточных.

рисунке 2.5.

Также видно, что при небольшом числе организмов количество энергии восстанавливается. Состояние энергии при спаде числа одноклеточных показано на рисунке 2.6.

По истечению какого-то количества времени система приходит к стабильному состоянию, при котором количество энергии значительно не изменяется.

В программном коде зададим переменные, которые будут сохранять количество энергии (*SkolkoEnergii*) и число организмов (*SkolkoVsego*) на каждом шаге моделируемой системы. Теперь у нас есть возможность построить графики изменения количества организмов и количества энергии со временем, а также относительно друг друга. Для этого воспользуемся функцией plot в матлабе.

Для начала отобразим изменение количества энергии (*SkolkoEnergii*) и организмов (*SkolkoVsego*) во времени (*t*). Для этого введем следующую команду:

```
>> plot(SkolkoEnergii, t, SkolkoVsego, t)
```

Получим рисунок 2.7.

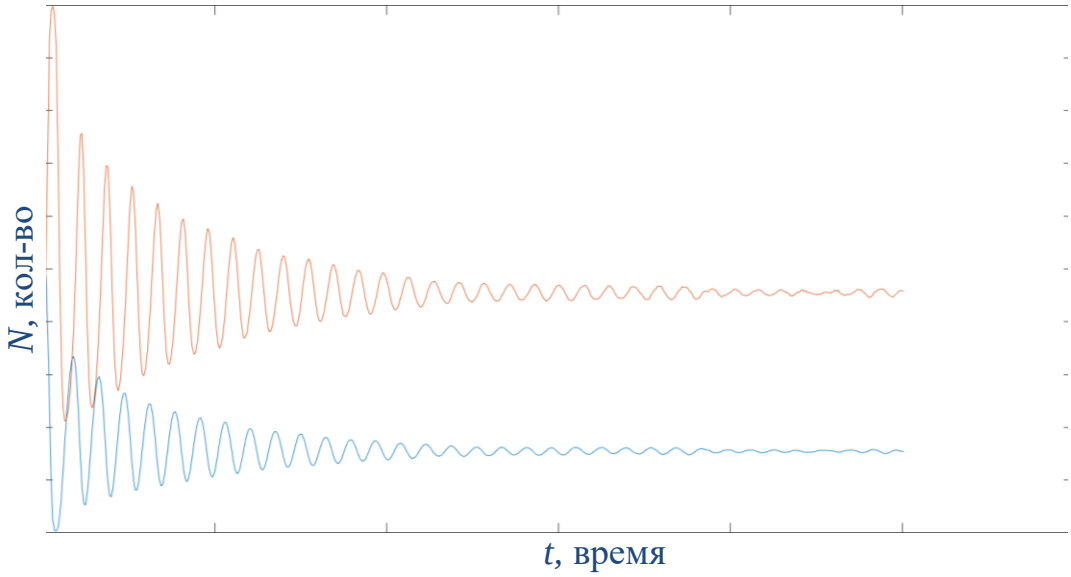

Рисунок 2.7. Изменение одноклеточных (оранжевый цвет) и численности энергии (синий цвет) во времени.

Для того, чтобы получить зависимость P_t от N_t вызовем следующую команду:

```
>> plot(10*SkolkoEnergii/65536, 100*SkolkoVsego/65536)
```

Полученное на рисунке 2.8.

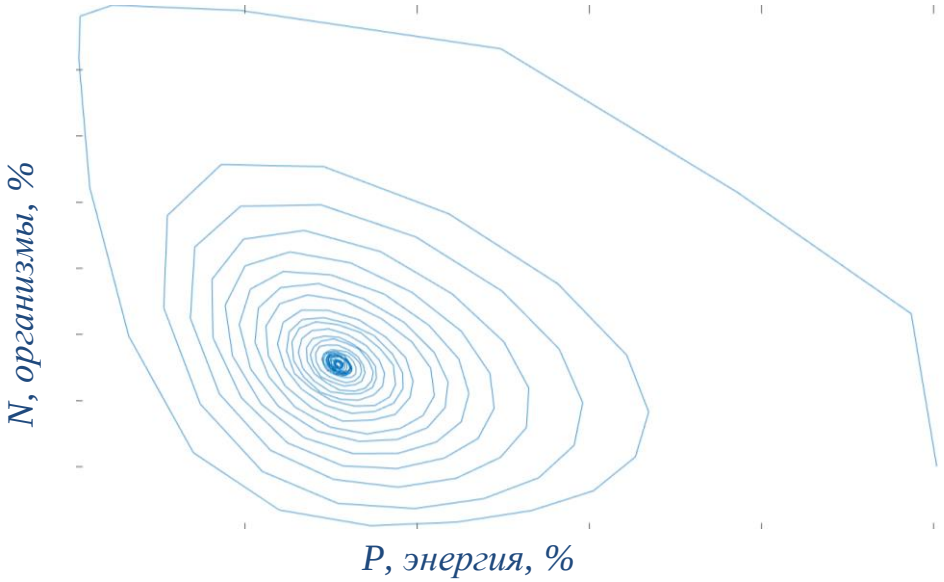

Рисунок 2.8. Зависимость количества энергии от количества одноклеточных.

2.4 Сравнение с математической моделью

Из модели, полученной при работе клеточного автомата, получим точку положения равновесия из графика зависимостей P_t от N_t. Из построенной модели получили, что точка равновесия приблизительно принимает следующие значения: P_p=15.53, N_p=45.58 (это приведено процентное соотношение).

Пока полученная модель соответствует тому, что ожидали получить от данной системы. Покажем, что наша система в действительности строит решение приведенной выше системы дифференциальных уравнений, что и показывает соответствие построенной с помощью клеточных автоматов модели процессам, происходящим в природе.

Для этого воспользуемся Wolfram Mathematica. У нас есть система дифференциальных уравнений, которую мы можем решить с помощью функции NDSolve. Однако для этого мы должны получить необходимые коэффициенты для уравнений. Известно, что r_n – это удельная скорость смертности хищника, тогда запустим нашу модель без жертвы и найдем скорость гибели хищников. Получим, что r_n=6.104. Т.к. r_p – это скорость восстановления питательного раствора, а нам известно, что за каждый шаг модели питательность раствора

увеличивается на 1, из этого следует, что коэффициент r_p=10 (все в процентном соотношении на клетку). Коэффициент b возмем равным 0.02 (b=0.02).

Теперь зная формулы, с помощью которых определяется точка равновесия, можем найти коэффициенты c_1 и c_2. Выразим c_1 и c_2 из выше приведенных формул, получим:

$$c_1 = \frac{r_p c_2 - b r_n}{N_p c_2}$$

$$c_2 = \frac{r_n}{P_p}$$

Теперь можем вызвать функцию, которая решает данную систему дифференциальных уравнений:

NDSolve[{P`[t]==($r_p$-$c_1$*Ne[t]-b*P[t])*P[t], Ne`[t]==(-r_n+c_2*P[t])*Ne[t], P[0]==28.28, Ne[0]==57.5}, {P, Ne}, {t, 40}]}]

N обозначается как Ne, т.к. в Wolfram у символа N есть определенное назначение. Начальные данные выбраны по построению клеточного автомата.

Таким образом, построив график (рисунок 2.9) решенного дифференциального уравнения увидим, что он подобен фазовому портрету, построенному в MATLAB, график. Это является качественной иллюстрацией того, что систему, построенную с помощью клеточного автомата, можно

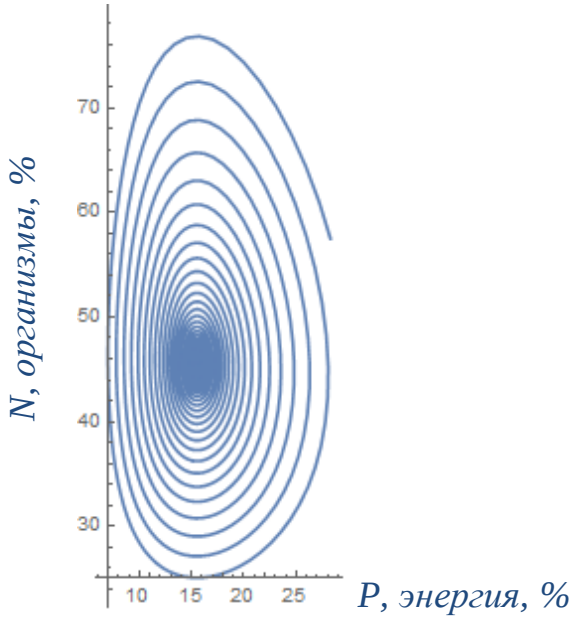

Рисунок 2.9. График зависимости энергии от числа одноклеточных, полученный при решении дифференциальной системы.

формализовать в виде математической модели «хищник-жертва», где хищником являются организмы, а жертвой – энергия.

Таким образом, полученный клеточный автомат по описанным законам строит такую же модель, как и та, что будет получена при решении дифференциальных уравнений, только в этом случае умение решать дифференциальные системы не понадобилось, достаточно было применить определенные правила к дискретному пространству, которое представляет из себя клеточный автомат.

2.5 Модификация правил функционирования системы

Теперь модифицируем 11 правило, по которому функционирует наша система. Пусть теперь каждая особь переходит на ту соседнюю клетку, энергетический запас которой самый большой. Также при размножении особь стремится поместить своего «ребенка» на соседнюю клетку, с максимальным запасом энергии. Таким образом мы получим следующую ситуацию при достижении равновесия, что можно увидеть на рисунке 2.10.

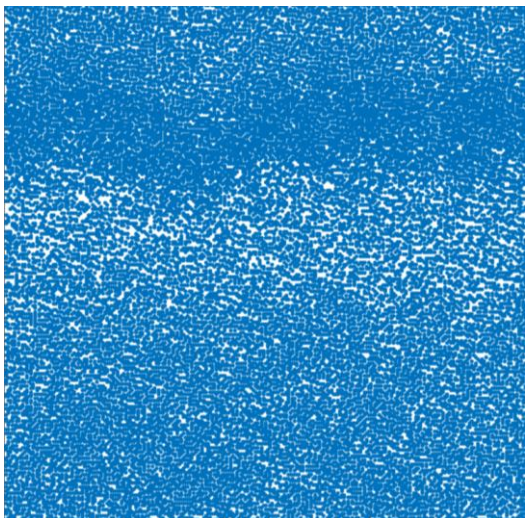

Рисунок 2.10. Стабильное состояние для «умного поведения» одноклеточных.

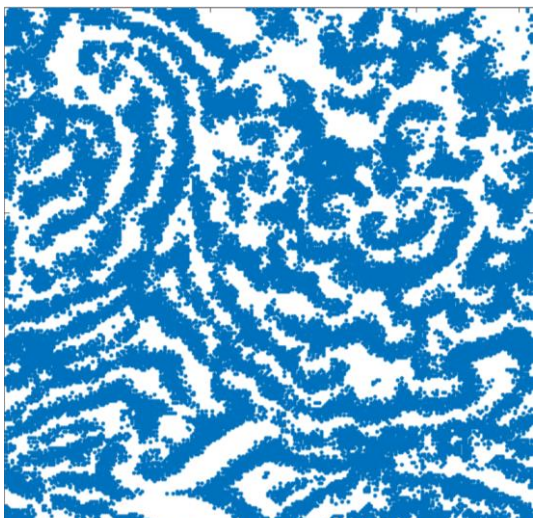

Рисунок 2.11. Расположение особей при некоторых допущениях.

Также при таком «умном» поведении особей точка равновесия, в окрестности которой в последствии совершаются небольшие колебания, достигается намного быстрее.

Если разрешить особям двигаться только если их запас энергии больше 3 и возраст больше либо равен 3, то получится очень интересное изображение (рисунок 2.11) их расположения, но тот же фазовый портрет все еще будет отображать затухающие колебания.

Добавим в нашу систему многоклеточный организм, который будет питаться одноклеточными, будем называть его хищником.

Тогда правила системы дополнятся следующими пунктами:

17. «Хищник» может нападать на свою «жертву», если она занимает соседние клетки. При нападении «хищника» жертва погибает с вероятностью P_N, а «хищник» занимает ее клетку. При этом запас энергии «хищника» увеличивается на энергетическую ценность «жертвы». Вероятность гибели «жертвы» не равна нулю, т.к. предполагается, что «жертва» имеет некоторые средства защиты, чтобы противостать хищнику. Если «хищнику» не удается одолеть «жертву», он погибает.

18. Если по соседству с «хищником» нет одноклеточных, то он перемещается на любую свободную соседнюю клетку.

19. Характеристики «хищника»:

- максимально возможное количество запасаемой энергии p_x;
- энергия, которую «хищник» тратит на себя, равная Δe_x;
- продолжительность жизни «хищника» - L_x;
- возраст, начиная с которого «хищник» может размножаться – L_{cx};
- энергия, которая необходима организму при размножении – Δr_x;
- вероятность остаться в живых при нападении на одноклеточного – P_N;

20. Если энергия хищника снизилась до нуля, л ибо количество прожитых лет превысило максимально возможное, «хищник» умирает.

21. Начальное число «хищников» $X_o\%$ от максимально возможного. Распределение особей подчиняется равномерному закону распределения. Общее число особей находится по формуле

$$X = \frac{x}{N_{max}}$$

где $x -$ количество "хищников" в момент времени, а $N_{max} = 256^2 = 65536$.

Возьмем следующие исходные данные:

Питательный раствор: $p_{max} = 10$

$r_p = 1$

Одноклеточный организм: $N_0 = 30\%$ $\Delta p_N = 5$ $\Delta r_N = 3$
$L = 15$ $p_N = 35$
$L_c = 3$ $\Delta e_N = 2$

«Хищник»: $X_0 = 5\%$ $p_x = 20$ $P_N = 93\%$
$L_x = 10$ $\Delta e_x = 4$
$L_{cx} = 5$ $\Delta r_x = 5$

Будем строить систему также при помощи MATLAB с помощью тензора.

Теперь каждый слой тензора примет следующие значения:

1 слой: энергия раствора.

2 слой: есть организм или нет (1 – есть, 0 – нет).

3 слой: сколько лет организму.

4 слой: запас энергии в организме.

5 слой: вспомогательный слой, показывающий размножался ли организм на этом шаге.

6 слой: есть «хищник» или нет (1 – есть, 0 – нет).

7 слой: сколько лет «хищнику».

8 слой: запас энергии у «хищника».

9 слой: показывает размножался ли «хищник» на данном шаге.

В моей реализации предположим, что «хищник» нападает на ту жертву из соседних, у которой энергетический запас больше всех, чтобы в случае победы над жертвой завладеть большим количеством энергии. Также одноклеточные организмы действуют по модифицированному правилу 11, т.е. переходят на ту клетку, где энергии больше всего.

Первое, что бросается в глаза у полученной модели, это отсутствие равномерного распределения одноклеточных, полученное по истечению времени, как в предыдущих моделях.

Также появляются области четырех типов:

- свободные от одноклеточных и хищников;
- практически полностью занятые одноклеточными, похожие на полк, который движется в сторону свободной области;
- область с малым (разреженным) количеством одноклеточных, их расположение напоминает равновесное состояние предыдущих систем;
- область, занятая хищниками.

Полки одноклеточных постоянно перемещаются, как и скопления хищников. Интересной особенностью построенной системы является то, что хищники как бы догоняют полки одноклеточных. Они движутся за группами одноклеточных вместо того, чтобы нападать спереди. Отсутствие хищников перед фронтом объясняется тем, что у одноклеточных есть средства защиты. Высокая плотность одноклеточных внутри полка, следовательно, частые встречи хищников с множеством одноклеточных почти не оставляют хищнику шанс на выживание. Поэтому хищник нападает на области с меньшей плотностью многоклеточных.

Также интересно наблюдать, когда случаются такие моменты, где хищники окружают целый полк одноклеточных и от них не остается и следа.

Изображение полученной системы приведено на рисунке 2.12.

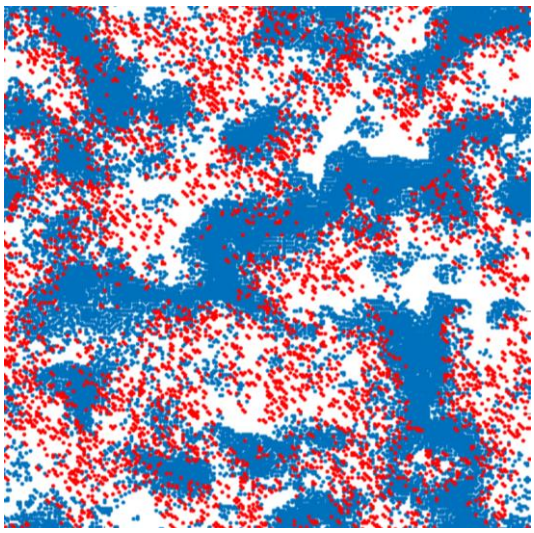

Рисунок 2.12. Система многоклеточные-
одноклеточные-питательный раствор.

Для подсчета количества хищников на каждом шаге введем переменную – *SkolkoKotikov*.

Построим графики (рисунок 2.13) зависимостей количества одних организмов от других, а также зависимости организмов от раствора.

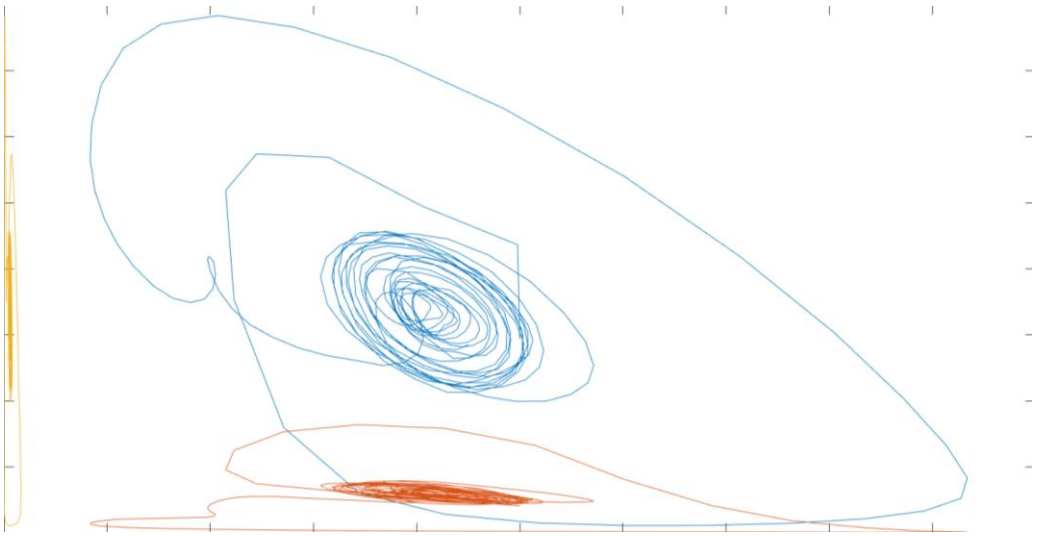

Рисунок 2.13. Зависимость энергии раствора от одноклеточных (синий); зависимость энергии от многоклеточных (красный); зависимость одноклеточных от многоклеточных (оранжевый).

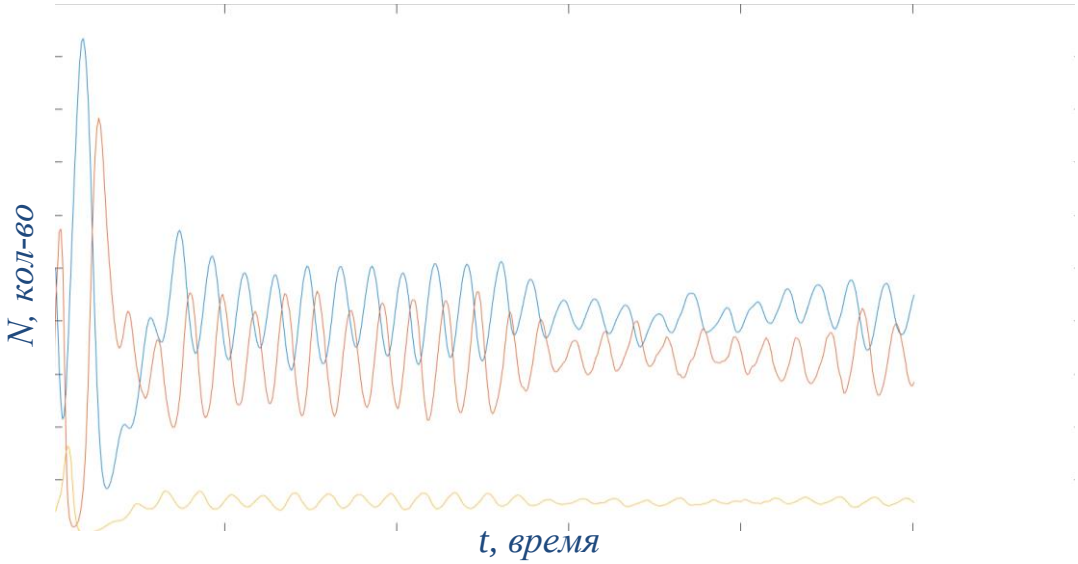

Рисунок 2.14. Зависимость энергии (синий) от времени; зависимость одноклеточных (красный) от времени; зависимость многоклеточных (оранжевый) от времени.

По полученным графикам видно, что организмы в данной системы подчиняются тем же дифференциальным уравнениям, только теперь у нас сразу функционирует несколько моделей. Например многоклеточные – это хищники, а одноклеточные – это жертвы. В то же самое время, одноклеточные – это хищники, а питательный раствор – жертва. Тогда по закону транзитивности многоклеточные

– это хищники, а раствор – жертва, что можно и наблюдать на получившихся графиках.

Таким образом, динамика эксперимента аналогична той, что описывается уравнениями модели Лотки-Вольтерры.

Если предположить, что наши организмы не простейшие, то, дополнив систему еще некоторыми правилами (например, добавить половое размножение и т.д.), получим модель, по которой сможем предсказать оптимальные условия для сосуществования организмов.[5]

ГЛАВА 3
АГЕНТНОЕ МОДЕЛИРОВАНИЕ

3.1 Понятие об агентном моделировании

Имитационное моделирование – это метод, который позволяет строить модели, описывающий процессы так, как они бы происходили в действительности.

Имитационное моделирование применяют для систем, которым свойственно нелинейное взаимодействие, «память», неочевидные зависимости между переменными, причинно-следственные связи, неопределенность и большое количество параметров. С задачами такого рода аналитическое (т.е. основанное на формулах) решение крайне сложно найти и тем более построить ментальную модель.[6]

Обозначим три вида имитационного моделирования:

- системная динамика (построение диаграмм причинно-следственных связей и влияний одних параметров на другие, на основе этих диаграмм модель имитируется на компьютере);
- дискретно-событийное моделирование (рассматриваются только основные события моделируемой системы, несущественные детали опускаются, т.е. происходит абстрагирование от непрерывной природы);
- агентное моделирование (исходя из предположений о поведении конкретного объекта моделируется поведение группы таких объектов);

Агентное моделирование – это один из видов имитационного моделирования. Основным его преимуществом является то, что можно не знать поведение системы в целом, необходимы лишь сведения о то, как ведут себя отдельные объекты. Часто нужно видеть, как агенты ведут себя при взаимодействии друг с другом, со средой, которая может иметь собственную динамику. Таким образом, глобальное поведение системы формируется из многих процессов, протекающих параллельно, т.е. индивидуальное поведение каждого агента формирует глобальное поведение системы.

Агентная модель состоит из агентов, отношений и среды. Не существует единого соглашения об определении агента в агентном моделировании. Одно из определений агента приведено далее.

Агент – это элемент модели, который моделирует некую сущность, и обладает определенными свойствами. Агент обладает свойством автономности. Агент взаимодействует (локально) с другими агентами в некоторой среде и/или со средой. Взаимодействие может определяться простыми правилами или абстрактными моделями (нейронные сети или генетические алгоритмы). Агент имеет состояния, которые определяют поведение агента. Агент имеет границу. Агент может обладать ресурсами и памятью. Агент может быть целенаправленным, адаптироваться и обучаться.

Необходимо отметить несколько фактов, касающихся агентов.

- агенты – это не обязательно люди; агентом может быть все, что угодно: человек, транспортное средство, земельные участки, проект, оборудование, организация или даже идея.
- для данной работы важно отметить, что агенты в своем глобальном понимании – это не клеточные автоматы и, вообще говоря, они не обязательно обитают в дискретном пространстве. Во многих агентных моделях пространства вообще нет, либо же оно непрерывное.
- агентом может быть даже тот объект, который кажется пассивным;
- агенты могут вообще не воздействовать друг с другом;

Итак, при разработке агентной модели создаются агенты, задается их поведение и далее агентов помещают в некую среду, устанавливают возможные связи между агентами, так модель готова к запуску.

Агентное моделирование нашло применение в различных областях знаний: от моделирования процессов фондовых бирж до предсказания распространения инфекционных заболеваний. [8]

Не существует какого-либо стандартного языка, на котором реализуются агентные модели.

3.2 Агентная реализация

На данный момент существует удобная среда моделирования AnyLogic. AnyLogic – это инструмент имитационного моделирования, который поддерживает все виды такого моделирования и позволяет их комбинировать. Отметим, что AnyLogic – это единственное ПО для агентного моделирования профессионального уровня [9]; также это современная среда разработки моделей на языке Java с

русскоязычным пользовательским интерфейсом и тщательно продуманной справочной системой. [7]

Воспользуемся возможностями AnyLogic для построения решения задачи, рассматриваемой в данной курсовой работе.

Для начала отметим, что создание стандартной агентной модели в AnyLogic заключается в задании агентов двух типов. В данной реализации это тип агента MyCell для описания высокоуровнего объекта, где содержатся все агенты, и тип агента Cell для описания агента. Каждый тип будет подтипом базового типа Agent. Число агентов будет включено в тип MyCell как дубликаты типа Cell.

Ранее упоминалось о том, что агенты, вообще говоря, это не клеточные автоматы, но задача поставлена так, что необходимо моделировать с помощью клеточных автоматов. Поэтому для данной работы будет использоваться дискретное пространство, так как клеточные автоматы представляют собой именно его.

Дискретное пространство представляет собой прямоугольный массив ячеек, который полностью или частично заполнен агентами. В одной ячейке может находится не более одного агента. Ранее упоминалось о типах соседства, в AnyLogic можно выбрать из двух соседств: Мурово (8 соседних ячеек) или Евклидово (4 соседних ячейки). Здесь это делается просто выбором типа соседства в настройках свойств, тогда как в MATLAB было необходимо самому продумывать логику, при которой соседство было бы какого-то определенного вида.

Создадим среду, в которой будут обитать одноклеточные. Модифицируем данную задачу, возьмем размерность 100x100 ячеек вместо 256x256.

Итак, есть тип агента, который называется MyCell, он будет содержать в себе массив агентов Cell. Тогда Cell – это агент, содержащий функции, параметры, описывающие поведение агента. В MyCell будут прописаны те функции, который показывают взаимодействие агентов Cell.

Опишем создание Cell и MyCell.

Создадим нового агента и назовем его Cell. В окно открывшегося графического редактора агента Cell поместим Прямоугольник в начало координат. Прямоугольник будет отображать состояние ячейки, т.е. показывать есть в данной клетке одноклеточный организм или нет. Далее агенту Cell задаются параметры:

1. energy – энергия клетки.
2. organism – параметр типа boolean, показывающий есть ли организм в данной клетке или нет.
3. org_energy – запас энергии в организме.
4. years – продолжительность жизни организма.

5. est_deti – вспомогательный параметр, необходимый для того, чтобы не перемещать организм, если он является новорожденным либо размножался на данном шаге.

Также агенту Cell задается ряд переменных, которые по сути являются входными данными модели:

1. len – максимально возможная продолжительность жизни.
2. lr – возраст, с которого возможно размножение.
3. org_est – столько организм берет от питательного раствора за один шаг.
4. org_max – максимально возможный запас энергии в организме.
5. en – столько организм тратит на себя за такт времени.
6. rn – организм тратит на размножение.
7. p_max – максимальная питательность раствора.
8. rp – прирост питательности за единицу времени.

Также агенту Cell задается ряд функций, которые изменяют параметры агента в соответствии правилам системы, также все они вызываются на каждом шаге моделирования:

1. vozrast() – функция, которая изменяет возраст одноклеточного.
2. prirost_energii() – увеличивает питательность раствора на rp за такт.
3. eda() – описывает поглощение организмом энергии из питательного раствора.
4. na_sebya() – уменьшает запас энергии организма на en.

Цвет Прямоугольника, который отображает есть ли на данном месте одноклеточный организм, будет красным, что означает, что организм есть, либо белым. Это задается в поле «Цвет заливки» выражением organizm ? red : white.

Так создана одна ячейка клеточного автомата. Теперь определим весь клеточный автомат, состоящий из 10000 таких ячеек.

Создадим нового агента MyCell. Из окна структуры перетащим элемент класса Cell и зададим количество клеток. В настройках свойства класса MyCell выберем тип пространства - дискретное, размерность 100x100 (100 столбцов и 100 строк) и соответственно Ширину и Высоту среды – 1000 и 1000, тип соседства – Мурово.

Для агента MyCell пропишем действия, которые будут выполняться после каждого шага (это действия, зависящие от положения одноклеточных друг от друга): движение и размножение. Для этого нам необходимы специальные встроенные функции:

int getR() – возвращает строку ячейки текущего расположения агента.

int getC() – возвращает столбец ячейки текущего расположения агента.

Agent getAgentAtCell(int r, int c) – возвращает агента, который расположен на строке r в столбце c.

С помощью данных функций найдем самую оптимальную (с наибольшим запасом энергии), незанятую соседнюю клетку и поместим в нее новорожденного либо, при нежелании размножаться сам организм будет перемещаться в найденную клетку.

Далее можно запускать модель. На рисунке 3.1 изображена модель, достигшая относительно стабильного состояния. Видно, что эффект получится аналогичный моделированию с помощью языка MATLAB.

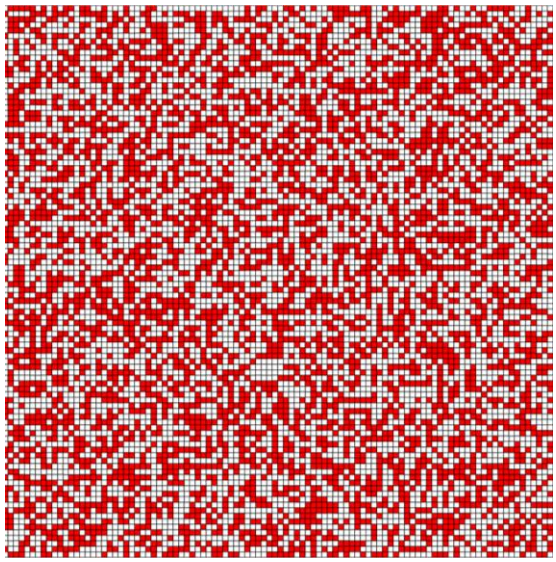

Рисунок 3.1. Стабильное состояние модели.

AnyLogic хорош тем, что можно быстро собирать необходимую информацию с модели и строить графики. Так, добавив элемент «График» из раздела «Статистика» можно отобразить зависимость между энергией питательного раствора и количеством одноклеточных в системе. Для этого добавим в MyCell переменные ener и num, которые будут хранить суммарное количество энергии раствора и количество одноклеточных соответственно. Для того, чтобы находить значения этих переменных пропишем команды в поле «Перед выполнением шага» (т.е. значения этих переменных будет пересчитываться на каждом шаге):

```
ener = 0;
num = 0;
for (Agent b: cell)
{
	if (((Cell)b).energy > 0){ener = ener+((Cell)b).energy;}
	if (((Cell)b).organizm){num++;}
}
```

Теперь можно видеть фазовый портрет (рисунок 3.2), который ведет себя таким образом, как при построении модели в MATLAB.

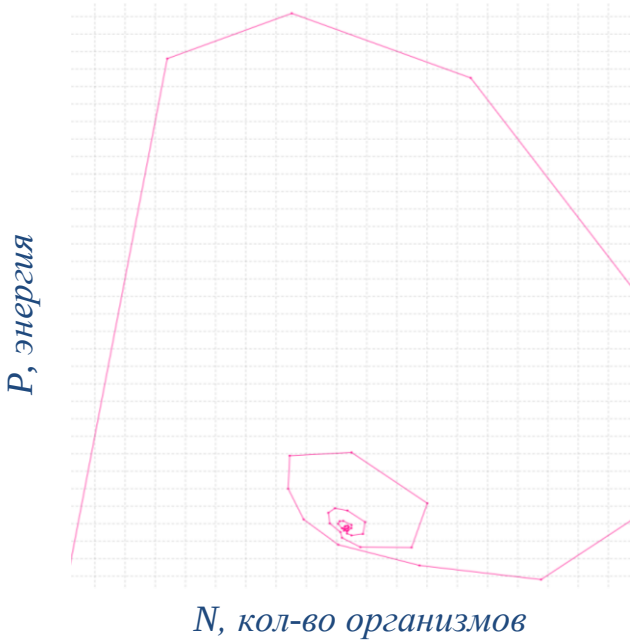

Рисунок 3.2. Зависимость количества энергии от количества одноклеточных.

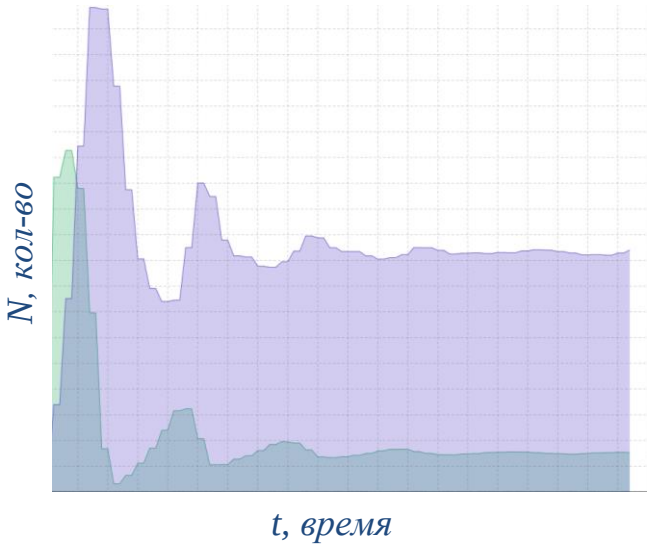

Рисунок 3.3. Зависимость энергии (зеленый) и одноклеточных (фиолетовый) от времени.

Также можно построить графики (рисунок 3.3) зависимости параметров ener и num от времени. Получим, что значения параметров изменяется в противофазе, что и характерно для подобных систем.

Таким образом, в помощью агентного подхода была построена модель, аналогичная построению с помощью пакета MATLAB, а также визуально соответствующую математическому описанию поведения таких систем.

ЗАКЛЮЧЕНИЕ

На основе проведенных теоретических и экспериментальных исследований в работе можно сделать вывод, что одним из приложений клеточных автоматов является построение биологических систем, а именно, построение подели развития популяций.

В данной курсовой работе была изучена, описана и реализована в среде MATLAB (матричная реализация) и в среде AnyLogic (агентная реализация) модель, которая представляла из себя клеточный автомат, отражающий поведение биологических систем. Также было показано соответствие построенной модели дифференциальным уравнениям, которые описывают взаимное влияние одних организмов на других. Из чего можно сделать вывод, что модель работает правильно, а также правильно подобраны коэффициенты модели.

Таким образом, клеточные автоматы являются еще одним из способов построения моделей, встречающихся в реальной среде. Это наталкивает на мысль, что способов моделирования бесконечное множество, и с помощью различных инструментов можно получить аналогичный результат.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Введение в математическое моделирование, Трусов П.В., 2007.
2. Математическое моделирование /Под ред. Дж.Эндрюса и Р.Мак-Доуна – М.. Мир, 1979. 240 с.
3. Машины клеточных автоматов, Т.Тоффоли, Н.Марголус – М. Мир, 1991. 280 с.
4. Модели динамики популяций на основе дифференциальных уравнений: реализация в среде R: учебно-методическое пособие / Зарипов Ш.Х., Никоненкова Т.В., Толмачева С.А. – Казань: Изд-во Казанского федерального университета, 2017. – 30 с.
5. Мир математики: в 40 т. Т. 28: Рафаэль Лаос-Бертра. Математика жизни. Численные модели в биологии и экологии. / Пер. с исп. – М.: Де Агостини, 2014. – 160 с.
6. Григорьев И. AnyLogic за три дня. Практическое пособие по имитационному моделированию, Интернет издание, 2017.
7. Куприяшкин А. Г. Основы моделирования систем: учеб. пособие //Норильск: НИИ. – 2015.
8. Киселев И. Н. Агентное моделирование сердечно-сосудистых систем человека.
9. Справочная система Anylogic. [Электронный ресурс] – Режим доступа: https://www.anylogic.ru

ПРИЛОЖЕНИЕ А.

Код программы

Ниже приведен код реализации на MATLAB для самого простого устройства системы.

```
L = 15; %продолжительность жизни организма
Max_en = 35; % максимальная энергия, запасаемая в организме
P_max = 10; % максимальная энергия раствора
L_r = 3; % возраст, с которого возможно размножение
From_en = 5; % столько организм берет от питательного раствора
En = 2; % организм тратит на себя
Rn = 3; % организм тратит на размножение
R_p = 1; % прирост питательности раствора

T = zeros(256, 256, 5);
T(:, :, 2) = rand(256) < 0.3;

M = 256;
P = [2:M 1];
n = [M 1:M-1];

% инициализация
for i = 1:256
for j = 1:256
     if T(i, j, 2) == 1;
          T(i, j, 3) = randi([1 L]);
          T(i, j, 4) = randi([1 Max_en]);
          T(i, j, 5) = 0;
     else T(i, j, 3:5) = 0;
     end
    T(i, j, 1) = randi([0 P_max]);
end
end

SkolkoVsego = sum(T(:, :, 2), 'all');
SkolkoEnergii = sum(T(:, :, 1), 'all');

for k = 1:500
```

```
    % возраст
    for i = 1:256
    for j = 1:256
        T(i, j, 5) = 0;
        if T(i, j, 2) == 1
            T(i, j, 3) = T(i, j, 3)+1;
            if T(i, j, 3) > L
                T(i, j, 2:5) = 0;
            end
        end
    end
    end

    % энергия раствора
    for i = 1:256
    for j = 1:256
        if T(i, j, 1) < P_max
            T(i, j, 1) = T(i, j, 1) + 1;
        end
    end
    end

    % питание
    for i = 1:256
    for j = 1:256
        if T(i, j, 1) >= From_en & T(i, j, 2) == 1
            if T(i, j, 4) <= Max_en-From_en
               T(i, j, 4) = T(i, j, 4) + From_en;
               T(i, j, 1) = T(i, j, 1) - From_en;
            elseif T(i, j, 4) > Max_en - From_en & T(i, j, 4) <
Max_en
                T(i, j, 1) = T(i, j, 1)-(Max_en-T(i, j, 4));
                T(i, j, 4) = Max_en;
            end
        elseif T(i, j, 1) < From_en & T(i, j, 2) ==1
            if T(i, j, 4) <= Max_en - From_en
               T(i, j, 4) = T(i, j, 4) + T(i, j, 1);
               T(i, j, 1) = 0;
            elseif T(i, j, 4) > Max_en - From_en
               if Max_en - T(i, j, 4) >= T(i, j, 1)
                   T(i, j, 4) = T(i, j, 4) + T(i, j, 1);
                   T(i, j, 1) = 0;
               elseif Max_en-T(i, j, 4) < T(i, j, 1)
                   T(i, j, 1) = T(i, j, 1)-(Max_en-T(i, j, 4));
                   T(i, j, 4) = Max_en;
```

```
                end
            end
        end
    end
    end

     % тратит на свои нужды
    for i = 1:256
    for j = 1:256
        if T(i, j, 2) == 1
            T(i, j, 4) = T(i, j, 4) - En;
            if T(i, j, 4) <= 0
                T(i, j, 2:5) = 0;
            end
        end

    end
    end


     N = T(P, :, 2) + T(n, :, 2) + T(:, P, 2) + T(:, n, 2) +
     T(P, P, 2) + T(n, n, 2) + T(n, P, 2) + T(P, n, 2);

    % размножение
    for i = 1:256
    for j = 1:256
        N_j = j; N_i = i;
        k = 0;
        if T(i, j, 3) >= L_r & N(i, j)~=8 & T(i, j, 2) == 1 &
T(i, j, 4) > R_n
            while T(N_i, N_j, 2) == 1 & k < 100
                k = k + 1;
                N_j = randi([j-1 j+1]);
                if N_j == j
                    N_i = i;
                    while N_i == i
                        N_i = randi([i-1 i+1]);
                        if N_i == 0
                            N_i = size(T, 1);
                        elseif N_i == size(T, 1)+1
                            N_i = 1;
                        end
                    end
                elseif N_j == 0
                     N_j = size(T, 1);
                     N_i = randi([i-1 i+1]);
```

```
                    if N_i == 0
                        N_i = size(T, 1);
                    elseif N_i == size(T, 1)+1
                        N_i = 1;
                    end
                elseif N_j == size(T, 1)+1
                    N_j = 1;
                    N_i = randi([i-1 i+1]);
                    if N_i == 0
                        N_i = size(T, 1);
                    elseif N_i == size(T, 1)+1
                        N_i = 1;
                    end

                else
                    N_i = randi([i-1 i+1]);
                    if N_i == 0
                        N_i = size(T, 1);
                    elseif N_i == size(T, 1)+1
                        N_i = 1;
                    end
                end
            end
            if k < 100
                T(N_i, N_j, 2) = 1;
                T(N_i, N_j, 3) = 1;
                T(N_i, N_j, 4) = 2;
                T(N_i, N_j, 5) = 2;
                T(i, j, 4) = T(i, j, 4) - R_n;
                T(i, j, 5) = 1;
            end
        end
    end
end

N = T(P, :, 2) + T(n, :, 2) + T(:, P, 2) + T(:, n, 2) + T(P,
P, 2) + T(n, n, 2) + T(n, P, 2) + T(P, n, 2);

%  движение
for i = 1:256
for j = 1:256
     N_j = j; N_i = i;
     k = 0;

     % если нет места
```

```
if T(i, j, 2) == 1 & N(i, j) == 8
    i;
elseif T(i, j, 5) == 1 || T(i, j, 5) == 2
    i;
else
   while T(N_i, N_j, 2) == 1 & k < 100
       k = k+1;
       N_j = randi([j-1 j+1]);
       if N_j == j
           N_i = i;
           while N_i == i
               N_i = randi([i-1 i+1]);
               if N_i == 0
                   N_i = size(T, 1);
               elseif N_i == size(T, 1)+1
                   N_i = 1;
               end
           end
       elseif N_j == 0
            N_j = size(T, 1);
            N_i = randi([i-1 i+1]);
            if N_i == 0
               N_i = size(T, 1);
            elseif N_i == size(T, 1)+1
                N_i = 1;
            end
       elseif N_j == size(T, 1)+1
           N_j = 1;
           N_i = randi([i-1 i+1]);
           if N_i == 0
               N_i = size(T, 1);
            elseif N_i == size(T, 1)+1
                N_i = 1;
           end

       else
           N_i = randi([i-1 i+1]);
           if N_i == 0
               N_i = size(T, 1);
           elseif N_i == size(T, 1)+1
               N_i = 1;
           end
       end
   end
   if k < 100
       T(N_i, N_j, 2) = 1;
```

```
                T(N_i, N_j, 3) = T(i, j, 3);
                T(N_i, N_j, 4) = T(i, j, 4);
                T(N_i, N_j, 5) = 0;
                T(i, j, 2:5) = 0;
            end
        end
    end
    end
    spy(T(:,:,2)); drawnow;
    SkolkoVsego = [SkolkoVsego, sum(T(:, :, 2), 'all')];
    SkolkoEnergii = [SkolkoEnergii, sum(T(:, :, 1), 'all')];
end
```