МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

Кафедра математического моделирования

**КУРСОВАЯ РАБОТА**

**КЛЕТОЧНО-АВТОМАТНОЕ МОДЕЛИРОВАНИЕ ДИФФУЗИИ**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Кулькова

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность (профиль) Математическое моделирование и вычислительная математика: Математическое моделирование

Научный руководитель

канд. физ.-мат. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.Е. Рубцов

Нормоконтролер

канд. физ.-мат. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.Е. Рубцов

Краснодар

2018

РЕФЕРАТ

Курсовая работа 25 с., 8 рис., 4 источника, 1 прил.

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ, КЛЕТОЧНЫЕ АВТОМАТЫ, ОКРЕСТНОСТЬ МАРГОЛУСА, ДВУМЕРНАЯ ДИФФУЗИЯ

Объектом работы является клеточно-автоматное моделирование диффузии.

Цель работы – исследование процесса диффузии с помощью клеточных автоматов.

В результате написана программа на языке С++, моделирующая диффузионный процесс; средствами консольной графики этот процесс был визуализирован.

СОДЕРЖАНИЕ

Введение4

1 Клеточный автомат5

1.1 Общая информация5

1.2 Примеры клеточных автоматов6

1.3 Окрестность Марголуса7

2 КА-модели двумерной диффузии9

2.1 Общая информация9

2.2 КА-диффузия с окрестностью Марголуса9

3 Программная реализация13

3.1 Постановка задачи13

3.2 Инструментарий13

3.3 Структура программы и описание алгоритма14

3.4 Результаты работы15

Заключение18

Список использованных источников19

Приложение20

# ВВЕДЕНИЕ

Идею клеточного автомата в середине ХХ века предложил математик Джон фон Нейман. Всякая модель клеточного автомата состоит из множества клеток, которым ставится в соответствие конечный автомат. В конкретный момент времени каждая клетка может принимать одно из возможных состояний. Алгоритм вычисления следующего состояния вычисляется по определенным правилам. Так, меняя правила, возможно моделировать самые разные процессы. Клеточные автоматы широко используются в математике, теоретической биологии, физике, а также в экономике, информатике, социологии и многих других областях. Область их применения практически безгранична. Чаще всего клеточные автоматы используются при моделировании физических процессов.

В данной курсовой работе будет рассмотрена программная реализация клеточного автомата, моделирующего двумерную диффузию с помощью окрестности Марголуса. Такая модель является довольно популярной по причине своей «элегантности» – сочетания в себе простоты и эффективности.

Целью данной работы является исследование процесса распространения загрязнения с течением времени.

Итог работы – визуализированный процесс диффузии.

1.1 Клеточный автомат

1.1 Общие сведения

Как уже упоминалось во введении, каждая клетка клеточного автомата (далее – КА) является конечным автоматом, который в конкретный момент времени может находиться в одном из состояний, зависящем от состояния ее соседей.

КА можно рассматривать как некоторый стилизованный мир, законы которого выражаются единственным набором правил, по которому всякая клетка на каждом шаге вычисляет свое новое состояние по состоянию соседних клеток. Таким образом, чтобы узнать, что произойдет (или может произойти) с конкретной клеткой в следующее мгновение, достаточно посмотреть на состояние ее окружения, называемого соседством. Законы на всем поле действуют одни и те же, и клетки совершают переход в новое состояние синхронно и параллельно. При этом меняется глобальная картина распределения состояний по пространству клеточного автомата. Такая картина называется конфигурацией КА.

Из выше сказанного можно выделить несколько основных свойств клеточного автомата, описанных в [1].

* Дискретность пространства, времени и состояний.
* Однородность.
* Синхронный режим изменения состояний.
* Пространственная и временная локальности.

Итеративная смена конфигурации при переходах всех состояний в новые называется эволюцией.

1. 2 Примеры клеточных автоматов

Кратко рассмотрим несколько примеров клеточных автоматов, представленных в [3].

Самым известным примером клеточного автомата является игра «Жизнь», созданная в 1970 году математиком Джоном Хортоном Конуэем. Она является ярким примером того, как имея очень маленькое количество правил или законов, с помощью КА можно моделировать огромное количество процессов, происходящих в живой природе. Принцип состоит в следующем.

Рассматривается бесконечная плоская решетка квадратных ячеек-клеток. Клетка может находиться в одном из двух состояний: жива или мертва. В системе выполняются все основные свойства клеточного автомата. Мир игры жизнь управляется всего тремя основными правилами.

Рождение. Новая клетка рождается на месте какой-нибудь мертвой клетки, если в окружении этой пустой клетки есть не менее трех живых клеток.

Выживание. Живая клетка продолжает жить, если имеет в своем окружении две или три клетки.

Смерть. Клетка погибает, если в ее окружении меньше двух или больше трех клеток.

Имея такой, казалось бы, скудный набор правил, игра «Жизнь» способна моделировать такие процессы, как морфогенез или логические элементы ЭВМ. В связи с этим, игра «Жизнь» нашла применение в различных науках. Более подробно работа этого клеточного автомата рассмотрена в [3].

Еще одним примером клеточного автомата, нашедшего свое применение в биологии, является игра «Аква-Тор». Игра моделирует поведение двух популяций, условно называемых «Хищники» и «Травоядные». «Хищники» едят «Травоядных», а пищи для «Травоядных» бесконечно много. Полем для игры является тор. Эта игра описывается следующими правилами.

Особь может переместиться на любую клетку, имеющую с данной общую сторону. Направление выбирается случайно из возможных свободных.

Особь может оставить потомство в той клетке, из которой она переместилась. Потомство появляется периодически.

Если особь является «хищной», то она может поглотить свою жертву, при этом перемещаясь на место жертвы.

Особь живет ограниченное количество времени, называемое временем жизни особи.

Если «хищная» особь не находит себе пищи в течение определенного времени (называемого временем голодной смерти), то она погибает.

КА описывает несколько важных явлений реальной жизни, например, процесс развития популяции на закрытом ареале. Более подробно рассмотреть действие «Аква-Тора» можно в [3].

1.3 Окрестность Марголуса

Рассмотрим еще один набор правил для клеточного автомата, представляющий собой так называемую окрестность Марголуса – полезную в физическом моделировании схему взаимосвязи клеток.

Имеется решетка – массив – клеток, разбитая на множество конечных однородных частей – блоков. В системе определяется правило не для одной клетки, а для блока в целом. Одно и то же правило применяется ко всем блокам в системе. Важно, блоки не перекрываются и не обмениваются информацией друг с другом, однако от шага к шагу разбиение меняется таким образом, чтобы происходило некоторое перекрытие блоков, используемых на предыдущих шагах.

Используя описанную схему, определим окрестность Марголуса. Решетка разбивается на блоки размера 2х2; шаги, в которых блоки соответствуют четному блоку, чередуются с шагами, использующими нечетные блоки, как показано на рисунке 1.

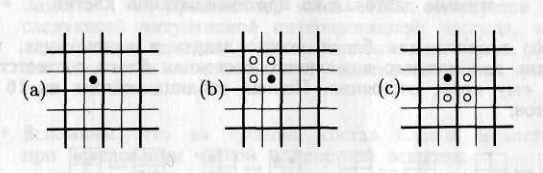


Рисунок 1 – Блоки 2х2 окрестности Марголуса

На рисунке 1 видно, что в зависимости от шага, клетка, помеченная в (а), будет иметь окрестностью либо четный блок (b), либо нечетный блок (c). Также нетрудно заметить, что блоки одной четности не пересекаются, и напротив, блоки разных четностей – перекрывают друг друга.

Заметим, что окрестность Марголуса использует только два различных разбиения. В общем случае их может быть больше, но их количество должно быть строго ограничено, и они должны циклически повторяться, чтобы не нарушалось условие однородности пространства и времени.

Пример использования окрестности Марголуса будет рассмотрен позднее.

2 КА-модели двумерной диффузии

2.1 Общая информация

Модели двумерной диффузии являются наиболее простыми и хорошо изученными клеточно-автоматными моделями. Клетки могут находиться в одном из двух возможных состояний: 0 или 1. Состояния определяют наличие или отсутствие единицы массы, наделённой скоростью. Сама диффузия представляет собой процесс беспорядочного блуждания частиц, который приводит к выравниванию концентрации вещества в пространстве.

Классические КА-модели диффузии имеют булев алфавит и эволюцию в виде последовательности булевых массивов. Все КА-модели диффузии являются стохастическими или случайными.

2.2 КА-диффузия с окрестностью Марголуса

Окрестность Марголуса уже была описана в пункте 1.3. Рассмотрим подробно её применение для моделирования процесса диффузии [1].

Дан клеточный массив размера NxN, где – область начального загрязнения. Этот массив имеет базовую часть с булевым алфавитом состояний AB и множеством имен , а также контекстную часть тоже с булевым алфавитом и множеством имен . В выделяются два подмножества имён и .

Для каждого имени определена локальная конфигурация

,

представляющая из себя блок из четырёх клеток.

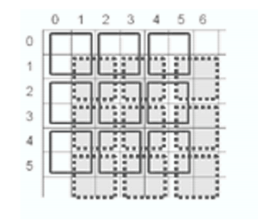


Рисунок 2 – КА диффузия с окрестностью Марголуса

На рисунке 2 изображены покрытия блоками: белым показаны четные блоки, серым – нечетные.

Таким образом для любой пары блоков одной четности справедливо утверждение о том, что они не пересекаются:

Множество локальных конфигураций представляет собой четное разбиение клеточного множества , а – его нечетное разбиение, т. е.

Функционирование КА происходит в двухтактном синхронном

режиме. Каждая итерация делится на два такта. На четных тактах базовая

подстановка применяется к четным блокам, на нечетных - к нечетным.

Подстановка выполняет сдвиг состояний в клетках блока с вероятностью

p по часовой стрелке и с вероятностью (1 - p) – против часовой стрелки. Графическое представление базовой подстановки можно увидеть на рисунке ниже.

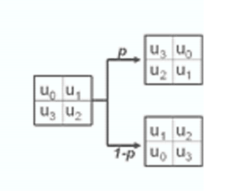


Рисунок 3 – Базовая подстановка

Для вычисления направления поворота блока состояний в каждую

подстановку вводится контекстная клетка , а для управления чередованием четного и нечетного тактов – контекстная клетка .

Полная система подстановок имеет следующий вид:

,

,

,

,

где

*rand –* случайно сгенерированное число.

Изменяя значение *p*, можно существенно влиять на скорость диффузии.

3 Программная реализация

3.1 Постановка задачи

Клеточно-автоматное моделирование двумерной диффузии с окрестностью Марголуса имеет свои достоинства. Основное из них – простота программной реализации.

Диффузия – это процесс беспорядочного блуждания частиц, который со временем приводит к выравниванию концентрации вещества в пространстве. В данной работе рассматривается открытая область с некоторым начальным загрязнением. Выйдя за границы области, частицы не способны вернуться обратно (вещество улетучивается). Поэтому одной из задач является необходимость пронаблюдать зависимость количества частиц в области от времени протекания процесса диффузии.

Диффузия, в том числе и смоделированная с помощью КА, является стохастическим процессом. Проанализировать зависимость скорости распространения вещества в зависимости от вероятности частиц переместиться в ту или иную область пространства также является важной задачей.

3.2 Инструментарий

Программа написана как консольное приложение Windows в среде разработки Code::Blocks. Поскольку сама программа довольно проста в исполнении и не требует сложного графического представления, консольное приложение является хорошим решением, единственным недостатком которого в данном случае будет отсутствие красивого интерфейса.

В качестве языка программирования выбран C++. Однако выбор другого языка программирования не сыграл бы существенной роли, поскольку никаких особенностей, присущих именно C++, в этой программе не используется.

Для визуализации работы используются графические возможности консоли.

3.3 Структура программы и описание алгоритма

Структура программы выглядит следующим образом [4].

В блоке «Функции и процедуры» содержатся процедуры для работы с вычислениями в решётке клеточного автомата и для работы с графикой. Такие процедуры как GetDesktopResolution, gotoxy и ShowPixels позволяют получить разрешение экрана, переместить курсор на выбранную позицию и визуализировать массив клеток в конкретный момент времени. Процедуры RightTurn и LeftTurn осуществляют поворот блоков окрестности Марголуса вправо и влево соответственно. Также имеется три процедуры с различными формами начальных загрязнений: CentrePollution – загрязнение небольшого участка в центре области, CornerPollution – загрязнение двух участков разных размеров по углам области и T\_Polution – загрязнение в форме буквы «Т».

В основной программе происходит реализация окрестности Марголуса по схеме, описанной в пункте 2.2.

Работа программы представляет следующее. Пользователю предлагается ввести время Т – продолжительность диффузии и вероятность P – вероятность смещения частиц (визуально – скорость их распространения).

После получения этих данных программа запускает цикл по Т, внутри которого происходит смена конфигурации клеточного автомата в зависимости от четности шага и значения высчитываемого случайного числа, которое сравнивается с P: происходит поворот четного или нечетного блоков в левую или правую стороны.

После завершения цикла получена итоговая конфигурация клеточного автомата.

3.4 Результат работы

После написания программной реализации необходимо протестировать программу. Как и в любом консольном приложении интерфейс у программы отсутствует, но есть небольшой диалог с программой, в котором пользователь может ввести данные, такие как время протекания диффузии и вероятность распространения частиц.

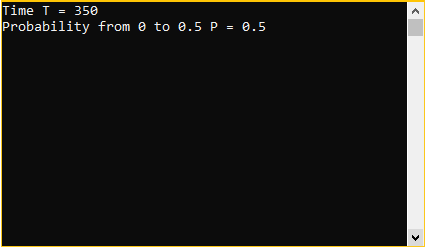


Рисунок 4 – Ввод данных пользователем

По окончании работы программы на экране будет показано распространение частиц в пространстве.

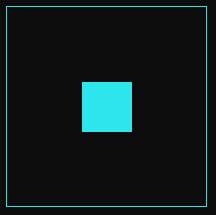


Рисунок 5 – Результат работы при Т = 0 и любом Р

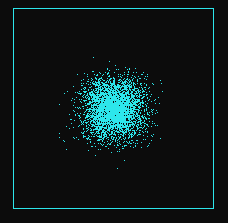


Рисунок 6 – Результат работы при Т = 50 и Р = 0,5

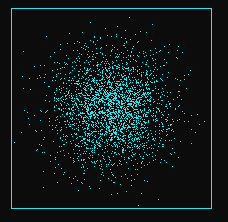


Рисунок 7 – Результат работы при Т = 350 и Р = 0,5

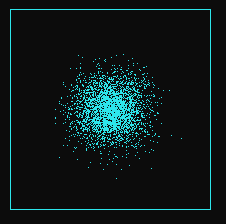


Рисунок 8 – Результат работы при Т = 350 и Р = 0,2

На основе полученных результатов можно судить о прямой зависимости концентрации вещества в пространстве от времени и вероятности перемещения частиц. Так на рисунках 5-7 видно, что при равном Р и увеличивающемся Т концентрация частиц в области снижается. А из рисунков 7-8 можно сделать вывод, что за одно и то же время при уменьшении Р скорость распространения частиц заметно снижается.

Конечно, данная модель описывает некую абстрактную, идеальную модель диффузии, на которую не действуют никакие внешние факторы и не учитываются условия среды, в которой она происходит. Эта модель является базой для дальнейшего исследования диффузии и более детального погружения в эту тему.

ЗАКЛЮЧЕНИЕ

В курсовой работе были рассмотрены основные теоретические основы клеточных автоматов и реализации с их помощью двумерной диффузии. В результате разработана программная реализация процесса диффузии, позволяющая пронаблюдать ее ход с течением времени и зависимость скорости ее распространения от вероятности частицы совершить перемещение. Тем самым достигнута поставленная цель курсовой работы.

В дальнейшем предполагается перейти к трехмерному пространству и рассмотреть влияние силы притяжения на протекание диффузии.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Бандман, О.Л. Клеточно-автоматные модели пространственной динамики / О.Л. Бандман // Системная информатика, 2006. С. 59 – 113.

2 Тоффоли, Т. Т63 Машины клеточных автоматов. / Т. Тоффоли, Н. Марголус пер. с англ. - М.: Мир, 1991. – 280 с.

3 Астафьев, Г.Б. Клеточные автоматы: Учебно-методическое пособие. / Г.Б. Астафьев, А.А. Короновский, А.Е. Храмов, Саратов: Изд–во ГосУНЦ «Колледж», 2003. С. 3 – 13.

4 Павловская, Т.А. С/С++. Программирование на языке высокого уровня / Т.А. Павловская, – СПб: Питер, 2003. – 461 с.

ПРИЛОЖЕНИЕ

Код программы

#define \_WIN32\_WINNT 0x0500

#include <iostream>

#include <windows.h>

#include <ctime>

#include <wtypes.h>

using namespace std;

//---------------ГЛОБАЛЬНЫЕ КОНСТАНТЫ И ПЕРЕМЕННЫЕ-------------------

const int n = 200;

const int N = n+2;

HWND hWnd = GetConsoleWindow();

HDC hDc = GetDC(hWnd);

//----------------------------ФУНКЦИИ И ПРОЦЕДУРЫ---------------------------------

void GetDesktopResolution(int& horizontal, int& vertical) //Разрешение экрана

{

RECT desktop;

const HWND hDesktop = GetDesktopWindow();

GetWindowRect(hDesktop, &desktop);

horizontal = desktop.right;

vertical = desktop.bottom;

}

void gotoxy(int x, int y) //Перемещение курсора

{

COORD pos = {x, y};

HANDLE output = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleCursorPosition(output, pos);

}

void RightTurn(bool A[N][N], int i, int j) //Поворот блока вправо

{

bool tea;

tea = A[i][j];

A[i][j] = A[i+1][j];

A[i+1][j] = A[i+1][j+1];

A[i+1][j+1] = A[i][j+1];

A[i][j+1] = tea;

}

void LeftTurn(bool A[N][N], int i, int j) //Поворот блока влево

{

bool tea;

tea = A[i][j];

A[i][j] = A[i][j+1];

A[i][j+1] = A[i+1][j+1];

A[i+1][j+1] = A[i+1][j];

A[i+1][j] = tea;

}

void ShowPixels(bool A[N][N], double scale, double shiftX, double shiftY, COLORREF COLOR) //Визуализация процесса

{

HPEN pen;

SelectObject(hDc, GetStockObject(NULL\_BRUSH));

pen = CreatePen(PS\_SOLID, 1, COLOR);

SelectObject(hDc, pen);

Rectangle(hDc, shiftX\*scale, scale\*(shiftY), scale\*(n+1+shiftX), scale\*(shiftY+n+1));

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++)

if (A[i][j]==1)

SetPixel(hDc, scale\*(j + shiftX), scale\*(i + shiftY), COLOR);

}

void CentrePollution (bool A[N][N]) //Загрязнение в центральном блоке

{

for (int i = N/2-N/8; i < N/2+N/8; i++)

for (int j = N/2-N/8; j < N/2+N/8; j++)

A[i][j] = 1;

}

void CornerPollution (bool A[N][N]) //Загрязнение по углам

{

for (int i = N/8; i < N/2-N/8; i++)

for(int j = N/8; j < N/2-N/8; j++)

A[i][j] = 1;

for (int i = N/2; i < N-N/8; i++)

for(int j = N/2; j < N-N/8; j++)

A[i][j] = 1;

}

void T\_Polution (bool A[N][N]) // Загрязнение буквой Т

{

for (int i = N/8; i < 2\*N/8; i++)

for (int j = N/8; j < N-N/8; j++ )

A[i][j] = 1;

for (int i = N/8; i < N-N/8; i++)

for (int j = N/2-N/16; j < N/2+N/16; j++ )

A[i][j] = 1;

}

//-----------------------------ОСНОВНАЯ-ФУНКЦИЯ-----------------------------------

int main()

{

srand(time(0));

//-----------------------------------ПЕРЕМЕННЫЕ------------------------------------------

COLORREF COLOR;

bool A[N][N];

double P = 0.5;

double Rand;

int N1, N2;

int T = 300;

int horizontal = 0, vertical = 0;

double scale, shiftX, shiftY;

//--------------------------------ТЕЛО-ПРОГРАММЫ -------------------------------------

COLOR = RGB(45,230,235);

GetDesktopResolution(horizontal, vertical);

scale = vertical/(3\*N);

shiftX = horizontal/(3\*scale) - N/scale;

shiftY = 10;

if (N%2 == 0)

{

N1 = N;

N2 = N-2;

}

else

{

N1 = N-2;

N2 = N;

}

cout << "Time T = "; cin >> T;

cout << "Probability from 0 to 0.5 P = "; cin >> P;

system ("cls");

//Заполнение поля начальным загрязнением

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++)

A[i][j] = 0;

CentrePollution(A);

for (int t = 0; t < T; t++)

{

for (int i = 0; i < N-1; i ++)

{

for (int k = 0; k < N; k++)

{

A[0][k] = 0;

A[k][0] = 0;

A[N-1][k] = 0;

A[k][N-1] = 0;

}

if (i%2 == 0)

{

for(int j = 0; j < N1; j+=2)

{

Rand = (rand()%100)\*0.01;

if (Rand < P)

RightTurn(A, i, j);

if (Rand > 1-P)

LeftTurn(A, i, j);

}

}

else

{

for(int j = 1; j < N2; j+=2)

{

Rand = (rand()%100)\*0.01;

if (Rand < P)

RightTurn(A, i, j);

if (Rand > 1-P)

LeftTurn(A, i, j);

}

}

}

system("cls");

}

ShowPixels(A, scale, shiftX, shiftY, COLOR);

ReleaseDC(hWnd, hDc);

gotoxy(0,40);

return 0;

}