МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования  
**«Национальный исследовательский нижегородский  
государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра Высокопроизводительных вычислений и системного программирования**

Направление подготовки: «Прикладная математика и информатика»

Профиль подготовки: «Общий профиль»

**ОТЧЕТ**

по учебной технологической (проектно-технологической) практике

на тему:

**«Изучение и реализация метода конечных разностей во временной области»**

**Выполнил:** студент группы 3821Б1ПМоп3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Борисов С.А.

Подпись

**Научный руководитель:**

к.т.н., доцент каф. ВВСП ИИТММ  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мееров И.Б.

Подпись

Нижний Новгород  
2023

Оглавление

[Введение 3](#_Toc3149)

[1 Постановка задачи 4](#_Toc22262)

[2 Алгоритм 5](#_Toc13700)

[3 Программная реализация 7](#_Toc19631)

[4 Эксперименты 9](#_Toc25436)

[4.1 Сравнение численного и истинного решений 9](#_Toc27829)

[4.2 Исследование сходимости метода 10](#_Toc697)

[5 Заключение 12](#_Toc7156)

[6 Список литературы 13](#_Toc10781)

[Приложение 1](#_Toc28420)4

Введение

В рамках данной практики представлена реализация метода конечных разностей во временной области (FDTD) – численного метода, основанного на разделении пространства и времени на дискретную сетку, позволяющую аппроксимировать уравнения Максвелла. Метод FDTD является мощным инструментом для исследования электромагнитных полей. Он обеспечивает основу для моделирования волновых процессов в различных средах и структурах. Область применения этого метода охватывает фотонику, электронику и изучение распространения радиоволн. В данной работе представлены основные этапы реализации метода FDTD, включая математические основы и алгоритмы вычислений.

# Постановка задачи

Необходимо изучить метод конечных разностей во временной области (FDTD), реализовать его с помощью языка программирования C++, проверить корректность программной реализации на тестовой задаче и исследовать сходимость метода.

Рассматривается двумерная сетка, в узлах которой определены сеточные значения

электрического поля и магнитного поля. Временная и пространственная области дискретизируются на равномерную сетку, содержащей узлов. Численное моделирование происходит в дискретные моменты времени с заданным постоянным шагом и завершается при превышении задаваемого .

При использовании явной разностной схемы сходимость метода – условная. В случае метода FDTD для шагов по времени и пространству должно соблюдаться условие Куранта ( – скорость света в вакууме):

(1)

Реализация метода производится для двух версий:

1. **Без сдвигов:**

В данной реализации дискретизация уравнений Максвелла осуществляется на одной временной и пространственной сетке без введения дополнительных смещений между компонентами полей и .

1. **Со сдвигами:**

В данной реализации поля хранятся со смещениями относительно друг друга на полшага по пространству и времени. Таким образом при смещении на полшага по времени вычисляется в точках и т.д., а в точках и т.д.. При смещении на полшага по пространству вычисляется в точках и т.д., а в точках и т.д.

# Алгоритм

Метод FDTD представляет из себя набор формул для вычисления компонент электрического и магнитного полей в момент времени , используя значения, полученные на предыдущем временном шаге.

Алгоритм задачи вычисления компонент электрического и магнитного полей с использованием метода FDTD включает в себя следующие шаги:

1. **Инициализация**

Задание начальных условий для электрического и магнитного полей и установка параметров моделирования, таких как размеры сетки, максимальное время и другие.

1. **Расчёт и обновление компонент электрического и магнитного полей**

С помощью дискретизированных уравнений Максвелла вычисляются значения компонент полей в каждом узле сетки на основе значений, полученных на предыдущем временном шаге. Таким образом происходит последовательное обновление полей, продолжающееся в течение заданного времени с использованием формул метода FDTD:

***Для реализации без сдвигов:***

(2)

(3)

(4)

(5)

(6)

(7)

***Для реализации со сдвигами:***

(8)  
 (9)  
 (10)

(11)  
 (12)  
 (13)

1. **Задание граничных условий**

Для корректного вычисления значений полей в узлах на границе расчетной области используются периодические граничные условия, в которых обращение по индексу эквивалентно обращению по индексу , а обращение по индексу эквивалентно обращению по индексу .

# Программная реализация

Программная реализация состоит из нескольких файлов, которые включают в себя исходные коды на языке C++ с реализацией самого метода FDTD и его тестирования, а также исходные коды на языке Python, в которых реализована визуализация полученных данных. Структура папок и файлов представлена на рисунке 1:

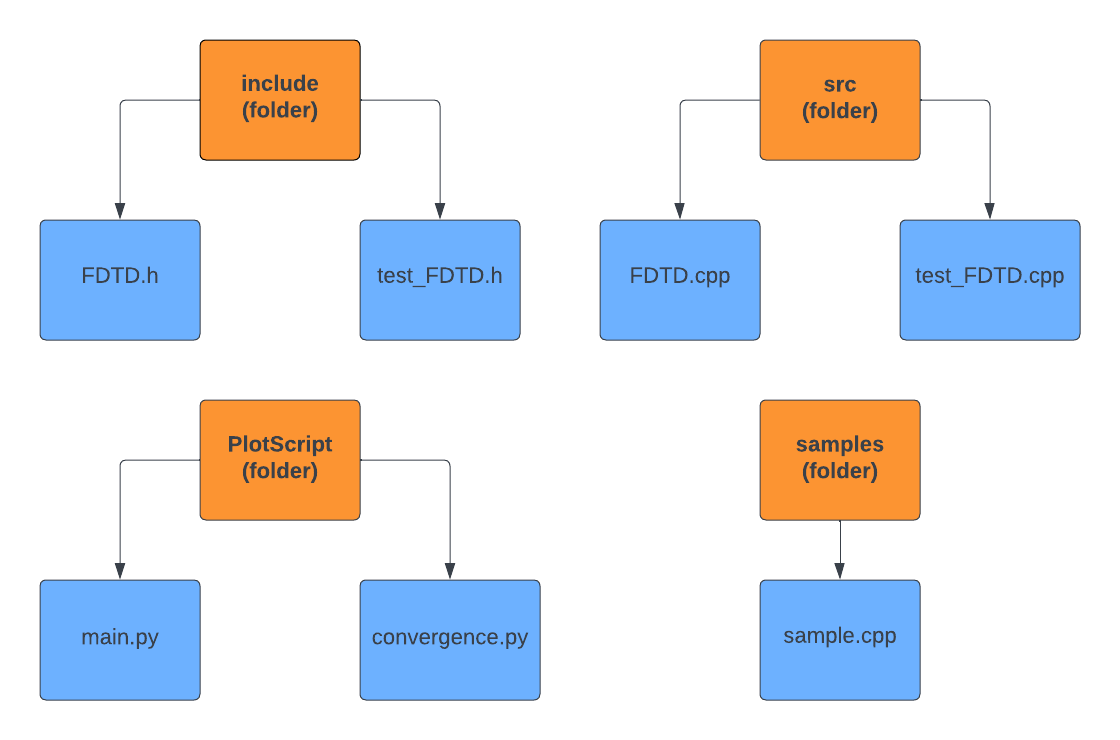


Рисунок 1. Структура проекта

* **FDTD.h** – заголовочный файл, содержащий класс Field, представляющий из себя контейнер и функтор для хранения значений компоненты поля или в текущий момент времени в узлах сетки и получения значений по индексам, а также класс FDTD, который содержит непосредственно реализацию самого метода конечных разностей во временной области с поддержкой версий со сдвигами и без сдвигов.
* **FDTD.cpp** – файл, содержащий реализацию методов классов Field и FDTD.
* **test\_FDTD.h** – заголовочный файл, содержащий класс Test\_FDTD, в котором реализован интерфейс тестирования метода FDTD.
* **test\_FDTD.cpp** – файл, содержащий реализацию методов класса Test\_FDTD.
* **sample.cpp** – вспомогательный файл для тестирования метода с помощью класса Test\_FDTD, записи полученных данных в файл и вывода погрешности метода.
* **main.py** – файл, предназначенный для запуска пользователем, в котором реализован ввод параметров моделирования и вывод графиков, визуализирующих распределение электрических и магнитных полей.
* **convergence.py** – файл, предназначенный для запуска пользователем, в котором реализован ввод параметров моделирования с последовательным изменением размеров сетки и количества разбиений по времени для исследования сходимости метода.

# Эксперименты

В ходе экспериментов на тестовой задаче проводилось сравнение численного решения с истинным (аналитическим) и исследование сходимости реализации метода без сдвигов и со сдвигами.

## Сравнение численного и истинного решений

Для сравнения численного решения с истинным выбрана тестовая задача, в которой начальные условия в момент времени заданы следующим образом:

(14)  
 (15)

При заданных начальных условиях истинное решение в момент времени следующее:

(16)  
 (17)

В качестве примера возьмём размеры сетки 32×32, наложенной на расчётную область , а также число разбиений по времени, равное 32 и максимальное время . Представлена визуализация численного решения на графике (рисунок 2). Максимальная глобальная погрешность для данного примера оказалась равной .

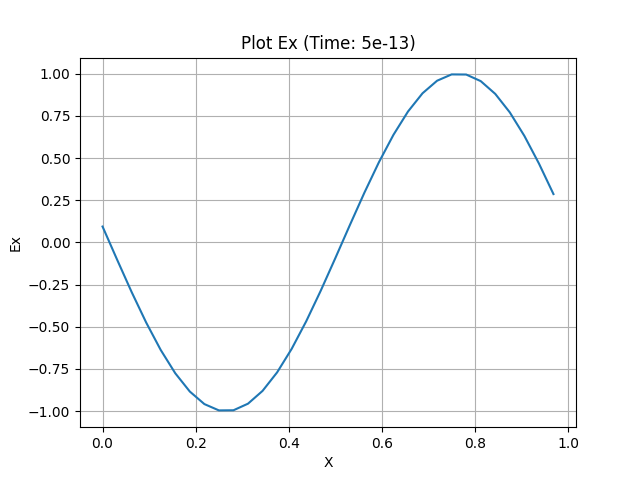


Рисунок 2. Пример визуализации распределения компоненты электрического поля

## Исследование сходимости метода

Порядок аппроксимации схемы без сдвигов − первый по пространству и второй по времени. То есть при увеличении размеров сетки в 2 раза и увеличении числа разбиений по времени в 4 раза погрешность уменьшится в 4 раза.

В схеме со сдвигами порядок аппроксимации второй как по времени, так и по пространству. То есть при увеличении размеров сетки в 2 раза и увеличении числа разбиений по времени во столько же раз погрешность уменьшится в 4 раза.

Для исследования сходимости метода возьмём начальные размеры сетки 16×16 и начальное число разбиений по времени, равное 16. Остальные параметры моделирования возьмём те же, что были использованы в сравнении численного решения с истинным. сетки и числа разбиений по времени.

Для схемы без сдвигов будем последовательно увеличивать размеры сетки в 2 раза, а число разбиений по времени в 4 раза. Для схемы со сдвигами же будем удваивать и размеры сетки, и число разбиений по времени.

Результаты исследований сходимости схемы без сдвигов приведены в таблице 1, со сдвигами − в таблице 2. Как можно видеть, в обоих таблицах погрешности метода в соседних строчках отличаются друг от друга примерно в 4 раза, что соответствует нашим ожиданиям.

Таблица 1: исследование сходимости схемы без сдвигов

|  |  |  |
| --- | --- | --- |
| Размеры сетки | Число разбиений по времени | Погрешность метода |
| 16×16 | 16 | 0.00332169 |
| 32×32 | 64 | 0.00085494 |
| 64×64 | 256 | 0.00021528 |
| 128×128 | 1024 | 5.39166e-05 |

Таблица 2: исследование сходимости схемы со сдвигами

|  |  |  |
| --- | --- | --- |
| Размеры сетки | Число разбиений по времени | Погрешность метода |
| 16×16 | 16 | 0.000601226 |
| 32×32 | 32 | 0.000150519 |
| 64×64 | 64 | 3.78107e-05 |
| 128×128 | 128 | 9.45353e-06 |
| 256×256 | 256 | 2.36344e-06 |

# Заключение

В ходе данной работы был изучен и реализован метод конечных разностей во временной области (FDTD), были проведены эксперименты, подтверждающие корректность программной реализации и исследующие сходимость используемой разностной схемы для версий без сдвигов и со сдвигами.

Были получены результаты, доказывающие порядок сходимости схемы без сдвигов − первый по пространству и второй по времени, а также, что более важно, второй порядок сходимости схемы со сдвигами (оригинальный метод FDTD) по пространству и по времени.

На этапе разработки программной реализации также проводились попытки распараллеливания метода с помощью OpenMP, однако эта тема требует большего изучения и в дальнейшем можно будет с помощью данной технологии ускорить вычисления.

# Список литературы

1. Yu W. et al. Parallel finite-difference time-domain method. – Norwood : Artech House, 2006. – С. 1-32.
2. Бэдсел Ч., Ленгдон А. Физика плазмы и численное моделирование. - 1989
3. Taflove A., Hagness S. C., Piket-May M. Computational electromagnetics: the finite-difference time-domain method //The Electrical Engineering Handbook. – 2005. – Т. 3. – №. 629-670. – С. 15

# Приложение

Файл FDTD.h

#include <vector>

#include <cmath>

#include <iostream>

#include <omp.h>

namespace FDTD\_Const

{

const double C = 3e10;

}

class Field

{

private:

int Ni;

int Nj;

std::vector<double> field;

public:

Field(const int, const int);

Field& operator= (const Field& other);

double& operator() (int \_i, int \_j);

int get\_Ni() { return Ni; }

int get\_Nj() { return Nj; }

};

enum class Component { EX, EY, EZ, BX, BY, BZ };

class FDTD

{

private:

Field Ex, Ey, Ez, Bx, By, Bz;

int Ni, Nj;

double ax, bx, ay, by, dx, dy, dt;

public:

FDTD(int size\_grid[2], double size\_x[2], double size\_y[2], double \_dt);

Field& get\_field(Component);

void update\_field(const int);

void shifted\_update\_field(const int);

int get\_Ni() { return Ni; }

int get\_Nj() { return Nj; }

};

Файл FDTD.cpp

#include "FDTD.h"

Field::Field(const int \_Ni = 1, const int \_Nj = 1) : Ni(\_Ni), Nj(\_Nj)

{

int size = Ni \* Nj;

field = std::vector<double>(size, 0.0);

}

Field& Field::operator= (const Field& other)

{

if (this != &other)

{

field = other.field;

Ni = other.Ni;

Nj = other.Nj;

}

return \*this;

}

double& Field::operator() (int i, int j)

{

int i\_isMinusOne = (i == -1);

int j\_isMinusOne = (j == -1);

int i\_isNi = (i == Ni);

int j\_isNj = (j == Nj);

int truly\_i = (Ni - 1) \* i\_isMinusOne + i \* !(i\_isMinusOne || i\_isNi);

int truly\_j = (Nj - 1) \* j\_isMinusOne + j \* !(j\_isMinusOne || j\_isNj);

int index = truly\_j + truly\_i \* Nj;

return field[index];

}

FDTD::FDTD(int size\_grid[2], double size\_x[2], double size\_y[2], double \_dt) : dt(\_dt)

{

Ni = size\_grid[0];

Nj = size\_grid[1];

Ex = Ey = Ez = Bx = By = Bz = Field(Ni, Nj);

ax = size\_x[0];

bx = size\_x[1];

ay = size\_y[0];

by = size\_y[1];

dx = (bx - ax) / static\_cast<double>(Ni);

dy = (by - ay) / static\_cast<double>(Nj);

}

Field& FDTD::get\_field(Component this\_field)

{

switch (this\_field)

{

case Component::EX: return Ex;

case Component::EY: return Ey;

case Component::EZ: return Ez;

case Component::BX: return Bx;

case Component::BY: return By;

case Component::BZ: return Bz;

}

}

void FDTD::update\_field(const int time)

{

for (double t = 0; t <= time; ++t)

{

#pragma omp parallel for collapse(2)

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

{

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

{

Ex(i, j) += FDTD\_Const::C \* dt \* (Bz(i, j + 1) - Bz(i, j - 1)) / (2.0 \* dy);

Ey(i, j) -= FDTD\_Const::C \* dt \* (Bz(i + 1, j) - Bz(i - 1, j)) / (2.0 \* dx);

Ez(i, j) += FDTD\_Const::C \* dt \* ((By(i + 1, j) - By(i - 1, j)) / (2.0 \* dx) - (Bx(i, j + 1) - Bx(i, j - 1)) / (2.0 \* dy));

}

}

#pragma omp parallel for collapse(2)

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

{

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

{

Bx(i, j) -= FDTD\_Const::C \* dt \* (Ez(i, j + 1) - Ez(i, j - 1)) / (2.0 \* dy);

By(i, j) += FDTD\_Const::C \* dt \* (Ez(i + 1, j) - Ez(i - 1, j)) / (2.0 \* dx);

Bz(i, j) -= FDTD\_Const::C \* dt \* ((Ey(i + 1, j) - Ey(i - 1, j)) / (2.0 \* dx) - (Ex(i, j + 1) - Ex(i, j - 1)) / (2.0 \* dy));

}

}

}

}

void FDTD::shifted\_update\_field(const int time)

{

for (double t = 0; t < time; t++)

{

#pragma omp parallel for collapse(2)

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

{

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

{

Bx(i, j) -= FDTD\_Const::C \* dt / 2.0 \* (Ez(i, j + 1) - Ez(i, j)) / dy;

By(i, j) += FDTD\_Const::C \* dt / 2.0 \* (Ez(i + 1, j) - Ez(i, j)) / dx;

Bz(i, j) -= FDTD\_Const::C \* dt / 2.0 \* ((Ey(i + 1, j) - Ey(i, j)) / dx - (Ex(i, j + 1) - Ex(i, j)) / dy);

}

}

#pragma omp parallel for collapse(2)

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

{

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

{

Ex(i, j) += FDTD\_Const::C \* dt \* (Bz(i, j) - Bz(i, j - 1)) / dy;

Ey(i, j) -= FDTD\_Const::C \* dt \* (Bz(i, j) - Bz(i - 1, j)) / dx;

Ez(i, j) += FDTD\_Const::C \* dt \* ((By(i, j) - By(i - 1, j)) / dx - (Bx(i, j) - Bx(i, j - 1)) / dy);

}

}

#pragma omp parallel for collapse(2)

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

{

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

{

Bx(i, j) -= FDTD\_Const::C \* dt / 2.0 \* (Ez(i, j + 1) - Ez(i, j)) / dy;

By(i, j) += FDTD\_Const::C \* dt / 2.0 \* (Ez(i + 1, j) - Ez(i, j)) / dx;

Bz(i, j) -= FDTD\_Const::C \* dt / 2.0 \* ((Ey(i + 1, j) - Ey(i, j)) / dx - (Ex(i, j + 1) - Ex(i, j)) / dy);

}

}

}

}