Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Уфимский университет науки и технологий»

Кафедра Высокопроизводительных вычислительных технологий и систем

	1	2	3	4	5	6	7	8	9	10
100										
90										
80										
70										
60										
50										
40										
30										
20										
10										
0										

«ДВУХСЛОЙНАЯ АКУСТИЧЕСКАЯ СХЕМА ДЛЯ ЗАДАЧИ РАСПРОСТРАНЕНИЯ ВОЛН»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

«Теория разностных схем»

3952.336211.000 ПЗ

Группа МКН-316	Фамилия И.О.	Подпись	Дата	Оценка
Студент	Яковлев О.В.			
Консультант	Ямилева А.М.			
Принял	Лукащук В.О.			

Уфа 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Уфимский университет науки и технологий»

Кафедра Высокопроизводительных вычислительных технологий и систем

ЗАДАНИЕ

на курсовую работу по дисциплине

«Теория разностных схем»

Студент: Яковлев Олег Витальевич Группа: МКН-316

Консультант: Ямилева Альфия Маратовна

1. Тема курсовой работы

Двухслойная акустическая схема для задачи распространения волн.

2. Основное содержание

2.1. Изучить литературу по теме "Двухслойная акустическая схема для задачи распространения волн"
2.2. Разработать программу для численного решения волнового уравнения в неоднородной среде.
2.3. Проанализировать поведение волны на границе сред с разной акустической плотностью.
2.4. Оформить пояснительную записку к курсовой работе.

3. Требования к оформлению материалов работы

3.1. Требования к оформлению пояснительной записки

Пояснительная записка к курсовой работе должна быть оформлена в соответствии с требованиями ГОСТ и содержать

- титульный лист,
- задание на курсовую работу,
- содержание,

- введение,
- заключение,
- список литературы,
- приложение, содержащее листинг разработанной программы, если таковая имеется.

Дата выдачи задания

"__" _____ 202_ г.

Дата окончания работы

"__" _____ 202_ г.

Консультант _____ Ямилева А.М.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Существует целый класс задач (колебания струны, движение сжимаемого газа, распространение возмущения электромагнитных полей и многие другие), которые описываются гиперболическими дифференциальными уравнениями или системами уравнений. В случае нелинейности коэффициентов уравнения, сложных граничных и/или начальных условий решение таких уравнений в аналитическом виде затруднительно, поэтому их зачатую решают численно. В частности, подобные задачи возникают при моделировании распространения акустических волн при геолого- и сейсморазведке, где необходимо учесть и сложные граничные условия, и неоднородность среды с учетом эффектов отражения, преломления и поглощения волн.

Для задач акустики нередко оказывается более удобно для численного решения перейти к двухслойной акустической схеме, чем решать исходное волновое уравнение на классическом конечно-разностном шаблоне. Как и схема для волнового уравнения, она позволяет построить неравномерную сетку по времени и учесть неоднородность среды. Но в отличие от классической схемы, двухслойная акустическая схема упрощает реализацию неотражающих граничных условий (PML-слои) для моделирования неограниченного пространства. Для задач геолого- и сейсморазведки это является значимым фактором.

Целью данной курсовой работы является моделирование распространения акустических волн в неоднородной среде с использованием двухслойной акустической схемы. Для достижения данной цели поставлены следующие задачи.

1. Изучить метод построения акустической двухслойной разностной схемы для задачи распространения волн.

2. Выполнить численную реализацию схемы для волнового уравнения с неоднородными граничными условиями и параметрами среды.

3. Исследовать характер распространения волн для случая слоисто-неоднородной среды.

1. ДВУХСЛОЙНАЯ АКУСТИЧЕСКАЯ СХЕМА

1.1. Одномерная акустическая схема для плоских акустических волн

Плоские акустические волны описывает уравнение малых колебаний струны при наличии внешнего силового поля f.

$$\begin{cases} \frac{\partial^2 P}{\partial t^2} = c^2 \frac{\partial^2 P}{\partial x^2} + f(x,t) \\ P|_{t=0} = \mu_1(x) \\ \frac{\partial P}{\partial t}\big|_{t=0} = \mu_2(x) \\ P|_{x=0} = \mu_3(t) \\ P|_{x=a} = \mu_4(t) \end{cases} \tag{1}$$

Заменим уравнение 1 второго порядка системой уравнений первого порядка. Введем потенциалы скоростей и правой части.

$$V(x,t) = \int\limits_0^x \frac{\partial P(\xi,t)}{\partial t}\, d\xi$$

$$F(x,t) = \int\limits_0^x f(\xi,t)\, d\xi$$

Тогда функции $P(x,t)$,$V(x,t)$ удовлетворяют системе уравнений 2.

$$\begin{cases} \frac{\partial P}{\partial t} = \frac{\partial V}{\partial x} \\ \frac{\partial V}{\partial t} = c^2 \frac{\partial P}{\partial x} + F(x,t) \\ P|_{t=0} = \mu_1(x) \\ V|_{t=0} = \int\limits_0^x \mu_2(\xi)\, d\xi \\ P|_{x=0} = \mu_3(t) \\ P|_{x=a} = \mu_4(t) \end{cases} \tag{2}$$

Рассмотрим значения скоростей и давления в узлах равномерной про-

странственной и временной сеток сеточных функций

$$\omega_h = \{ih, i = \overline{0..N}\}$$
$$\omega_\tau = \{i\tau, i = \overline{0..M}, \tau = \frac{T}{M}\}$$
$$y_i = P(x_i, t), x_i \in \omega_h$$
$$\hat{y}_i = P(x_i, t+\tau), x_i \in \omega_h$$
$$z_i = V\left(x_i + \frac{h}{2}, t\right), i \leq N-1, x_i \in \omega_h$$
$$\hat{z}_i = V\left(x_i + \frac{h}{2}, t+\tau\right), i \leq N-1, x_i \in \omega_h$$

Составим явную акустическую схему, используя шаблон, изображенный на рисунке 1.

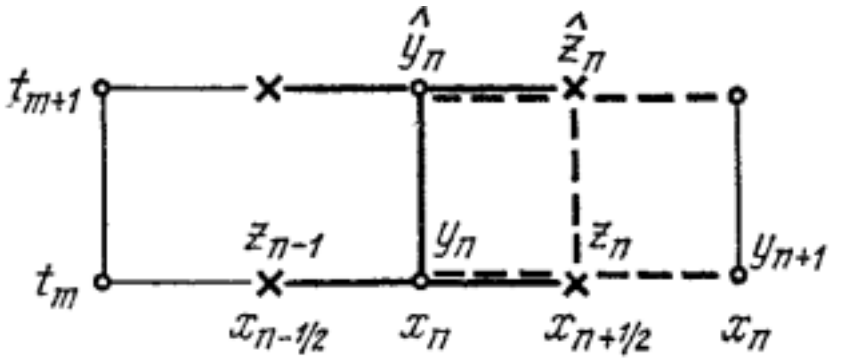


Рис. 1 – Шаблон одномерной акустической схемы

$$\begin{gathered} \frac{\hat{z}_i - z_i}{\tau} = \frac{c^2}{h}(y_{i+1} - y_i) + F_{i+1/2}^{m+1/2}, 0 \leq i \leq N-1 \\ \frac{\hat{y}_i - y_i}{\tau} = \frac{1}{h}(\hat{z}_i - \hat{z}_{i-1}), 1 \leq i \leq N-1 \end{gathered} \tag{3}$$

При выполнении условия Куранта $c\tau < h$ акустическая схема 3 сходится со скоростью $O(h^2 + \tau^2)$ [1]

1.2. Двухмерная акустическая схема. Метод FDTD

Аналогично одномерному случаю, сведем волновое уравнение второго порядка к системе дифференциальных уравнений первого порядка. Главные акустические дифференциальные уравнения для двух измерений

имеют вид

$$\begin{cases} \dfrac{\partial P}{\partial t} = -\,c^2 \left(\dfrac{\partial V_x}{\partial x} + \dfrac{\partial V_y}{\partial t} \right) \\ \dfrac{\partial V_x}{\partial t} = -\,\dfrac{\partial P}{\partial x} \\ \dfrac{\partial V_y}{\partial t} = -\,\dfrac{\partial P}{\partial y} \end{cases} \tag{4}$$

Для того, чтобы вычислить значения в следующий момент времени, сместим сетки компонент V_x и V_y векторов скоростей относительно сетки давлений P на полшага вдоль соответствующей оси по пространству и на полшага по времени[2]. В результате узлы сетки будут расположены так, как изображено на рисунке 2.

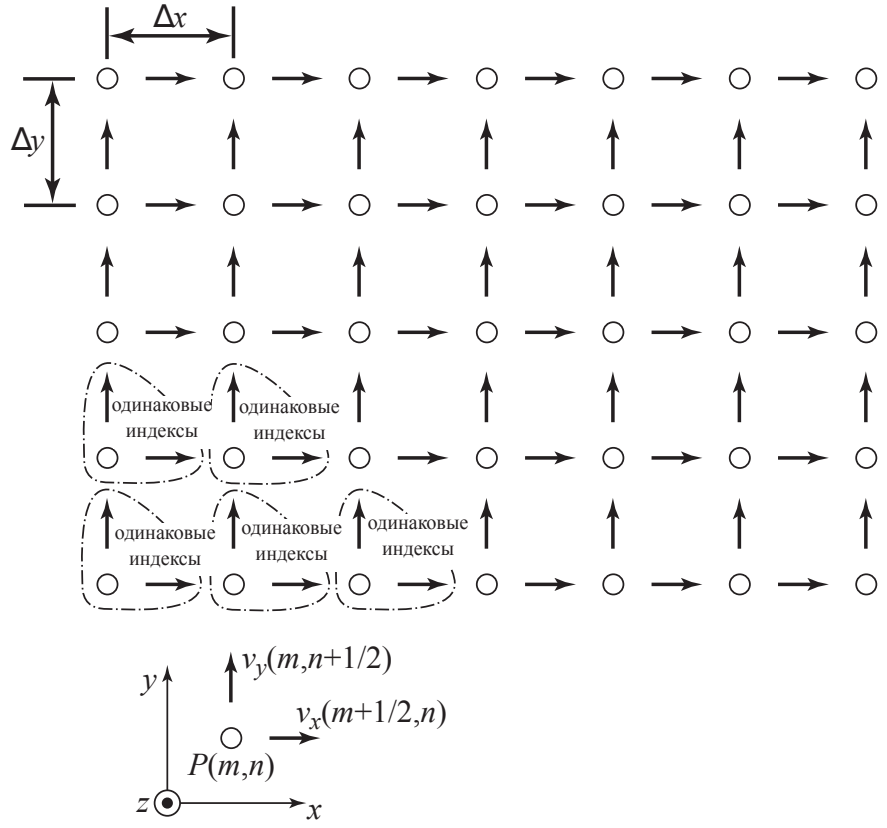


Рис. 2 – Расположение узлов в двухмерной сетке

Введем следующие обозначения:

$$\begin{aligned} P(x,y,t) &= P(m\Delta_t a_x, n\Delta_t a_y, q\Delta_t a_t) = P^q[m,n], \\ V_x(x,y,t) &= V_x\left([m+1/2]\Delta_x, n\Delta_y, [q+1/2]\Delta_t\right) = V_x^{q+1/2}[m,n], \\ V_y(x,y,t) &= V_y\left(m\Delta_x, [n+1/2]\Delta_y, [q+1/2]\Delta_t\right) = V_y^{q+1/2}[m,n] \end{aligned}$$

Заменим в системе 4 производные на разностные аналоги. Тогда давление

в следующий момент времени вычисляется по формуле

$$P^q[m,n] = P^{q-1}[m,n] - \rho c_a^2 \frac{\Delta_t}{\delta} \Big(V_x^{q-1/2}[m,n] - V_x^{q-1/2}[m-1,n] + V_y^{q-1/2}[m,n] - V_y^{q-1/2}[m,n-1] \Big) \tag{5}$$

Скорость на следующем временном шаге можно вычислить по формулам

$$V_x^{q+1/2}[m,n] = V_x^{q-1/2}[m,n] - \frac{1}{\rho}\frac{\Delta_t}{\delta} \left(P^q[m+1,n] - P^q[m,n] \right) \tag{6}$$

$$V_y^{q+1/2}[m,n] = V_y^{q-1/2}[m,n] - \frac{1}{\rho}\frac{\Delta_t}{\delta} \left(P^q[m,n+1] - P^q[m,n] \right) \tag{7}$$

Уравнения 5, 6, 7 составляют итоговую двухслойную акустическую схему для двухмерной прямоугольной области.

2. НЕОТРАЖАЮЩИЕ ГРАНИЧНЫЕ УСЛОВИЯ. PML

В задачах акустики часто необходимо моделировать процессы, происходящие на неограниченной области. Однако, многие виды граничных условий порождают колебания, отраженные от границы. Эти отраженные колебания накладываются на решение волнового уравнения и могут его исказить. Поэтому возникает необходимость моделирование неотражающих граничных условий на границе вычислительной области. Одним из таких методов является Perfectly Matched Layer(PML)

2.1. Аналитическое продолжение решения

Решением волнового уравнения является осциллирующая аналитическая функция. Следовательно, решение можно аналитически продолжить для всей комплексной плоскости (Рисунок 3). Однако, если функцию вычислять не вдоль действительной оси, а добавить линейно растущую мнимую часть на некоторой области, то решение будет экспоненциально затухающее в этой области. То есть, в области растущей мнимой части функция ведет себя как колебания в поглощающей среде. [3]

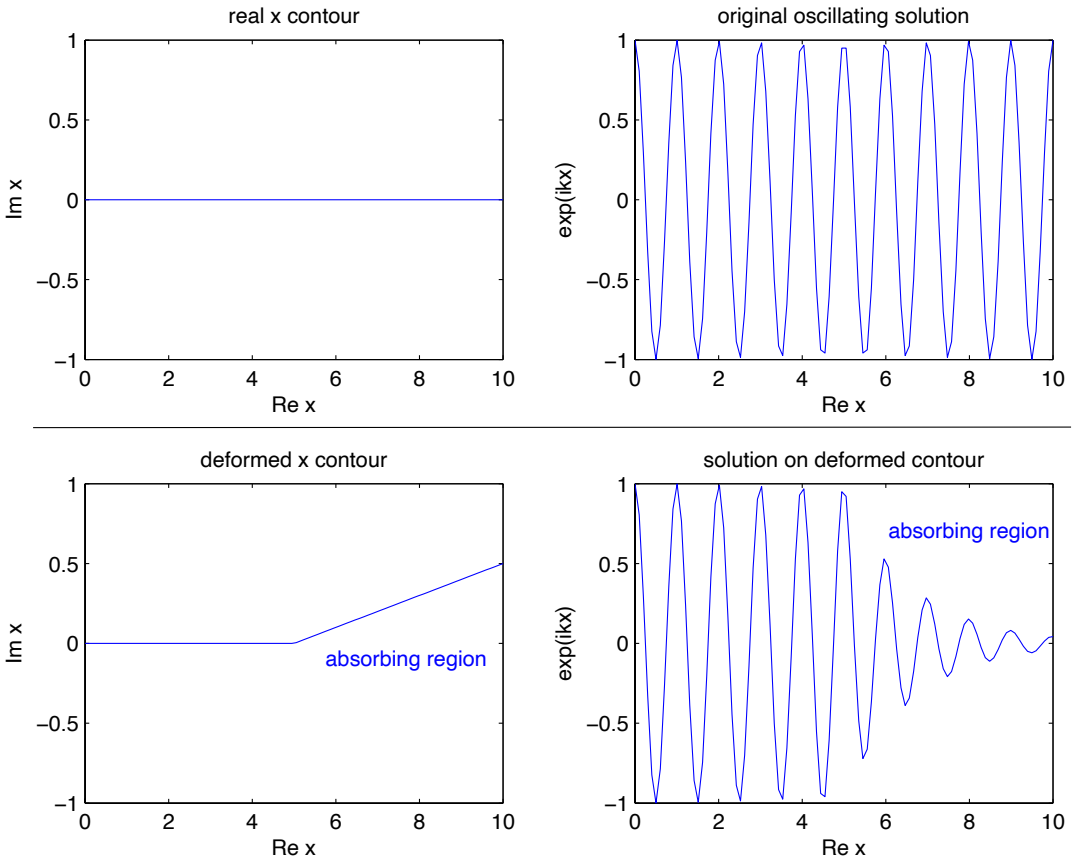


Рис. 3 – *Сверху*: действительная часть осциллирующего решения (справа), значения x вдоль действительной оси комплексной плоскости x (слева). *Снизу*: искривленный контур (слева), экспоненциально затухающее решение (справа).

При этом решение в области, где мнимая часть равна 0, не изменилось, то есть слой PML ведет себя как неотражающая поглощающая среда.

2.2. PML для двухмерных скалярных волн

Пусть решение $P(x, y, t)$ зависит от времени как $e^{-i\omega t}$. Подставляя в систему 4, получим систему 8.

$$\begin{cases} -i\omega P = -c^2 \left(\dfrac{\partial V_x}{\partial x} + \dfrac{\partial V_y}{\partial y} \right) \\ -i\omega V_x = -\dfrac{\partial P}{\partial x} \\ -i\omega V_y = -\dfrac{\partial P}{\partial y} \end{cases} \tag{8}$$

Применяя PML-преобразование $\frac{\partial}{\partial x} \mapsto \frac{1}{1+i\frac{\sigma_x}{\omega}} \cdot \frac{\partial}{\partial x}$, получим

$$\begin{cases} -i\omega P + \sigma_x P = -c^2 \left(\dfrac{\partial V_x}{\partial x} + \dfrac{\partial V_y}{\partial y} \left(1 + i\dfrac{\sigma_x}{\omega}\right)\right) \\ -i\omega V_x + \sigma_x V_x = -\dfrac{\partial P}{\partial x} \\ -i\omega V_y = -\dfrac{\partial P}{\partial y} \end{cases} \tag{9}$$

При обратном преобразовании Фурье, выражение $\frac{i}{\omega}\sigma_x \frac{\partial V_y}{\partial y}$ перейдет в интеграл другой величины. Обозначим его за Ψ. Тогда к системе добавляется еще одно уравнение

$$\frac{\partial \Psi}{\partial t} = -c^2 \sigma_x \frac{\partial V_y}{\partial y} \tag{10}$$

Итоговая система имеет вид

$$\begin{cases} \dfrac{\partial P}{\partial t} = -c^2 \left(\dfrac{\partial V_x}{\partial x} + \dfrac{\partial V_y}{\partial y} \right) - \sigma_x P + \Psi \\ \dfrac{\partial V_x}{\partial t} = -\dfrac{\partial P}{\partial x} - \sigma_x V_x \\ \dfrac{\partial V_y}{\partial t} = -\dfrac{\partial P}{\partial y} \\ \dfrac{\partial \Psi}{\partial t} = -c^2 \sigma_x \dfrac{\partial V_y}{\partial y} \end{cases} \tag{11}$$

3. РАСЧЕТ И АНАЛИЗ РЕЗУЛЬТАТОВ

3.1. Сходимость численного решения

Для проверки правильности составленной разностной схемы была выбрана задача 12 колебания прямоугольной мембраны с нулевыми граничными условиями.

$$\begin{cases} \dfrac{\partial^2 P}{\partial t^2} = \dfrac{\partial^2 P}{\partial x^2} + \dfrac{\partial^2 P}{\partial y^2} \\ P|_{x=0} = 0 \\ P|_{x=1} = 0 \\ P|_{y=0} = 0 \\ P|_{y=1} = 0 \\ P|_{t=0} = \sin \pi x \sin \pi y \\ \left.\dfrac{\partial P}{\partial t}\right|_{t=0} = 0 \end{cases} \tag{12}$$

Решением данной задачи является функция

$$P(x, y, t) = \cos \sqrt{2}\pi t \sin \pi x \sin \pi y$$

Вычислим норму ошибки

$$||E|| = \max_{x_i, y_j \in \omega_h, t_k \in \omega_\tau} |P^k_{i,j} - P(x_i, y_j, t_k)|$$

для численного решения. Норма ошибки в зависимости от количества узлов пространственной сетки представлен на рисунке 4.

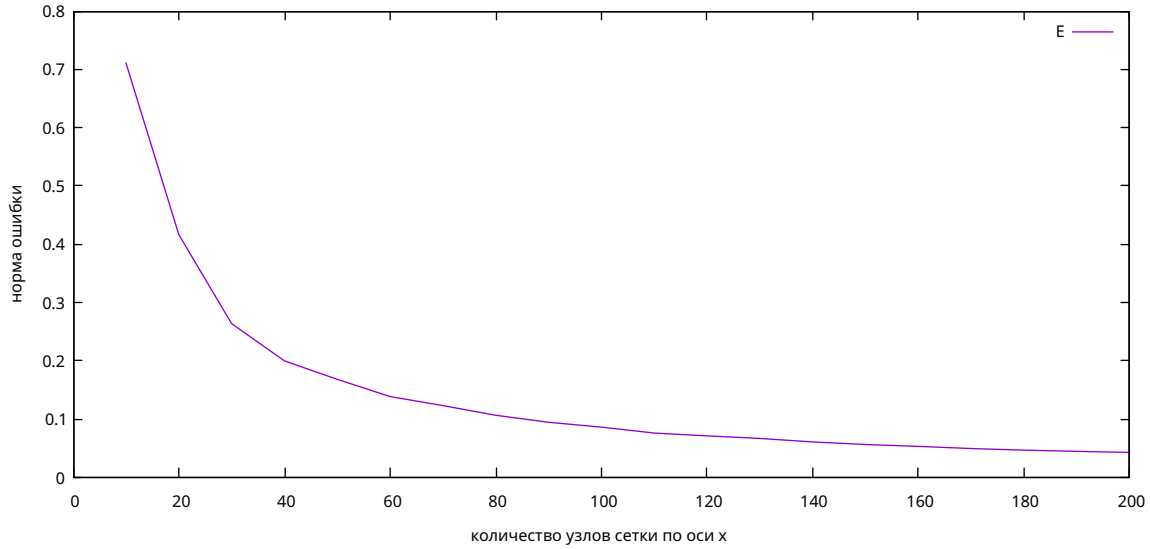


Рис. 4 – Зависимость нормы ошибки от количества узлов сетки по оси x

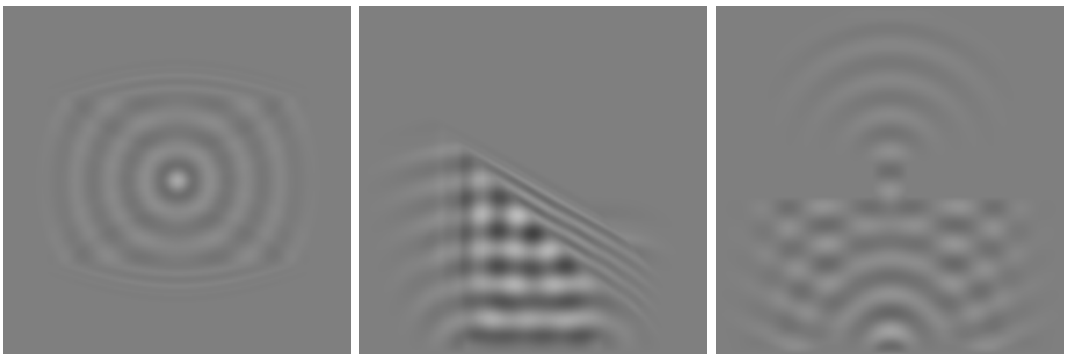
Схема имеет линейную скорость сходимости. Выберем сетку с количеством узлов $N = 300$ для дальнейших расчетов.

3.2. Поведение волн на границе сред с разной скоростью звука

Пусть задан точечный источник колебаний в центре вычислительной области. На верхней и нижних границах расположена акустически плотная среда. При переходе акустической волны в более акустически плотную среду волна отражается от границы раздела сред. На рисунке 5a показано отражение волн, распространяющихся в среде с со скоростью звука $C_1 = 1$, от границы со средой с меньшей скоростью звука $C_2 = 0.38$.

Пусть задан источник колебаний в виде прямоугольника $|y| < \frac{1}{20}, |x| < \frac{1}{5}$ у нижней границы вычислительной области. Неоднородность имеет форму $|x - y| < 0.1$. При попадании волны в более акустически плотную среду под углом к границе раздела сред происходит преломление, изображенное на рисунке 5b.

Пусть задан точечный источник колебаний в середине нижней границы вычислительной области. Зададим неоднородность в виде акустически



(a) Отражение (b) Преломление (c) Дифракция

Рис. 5

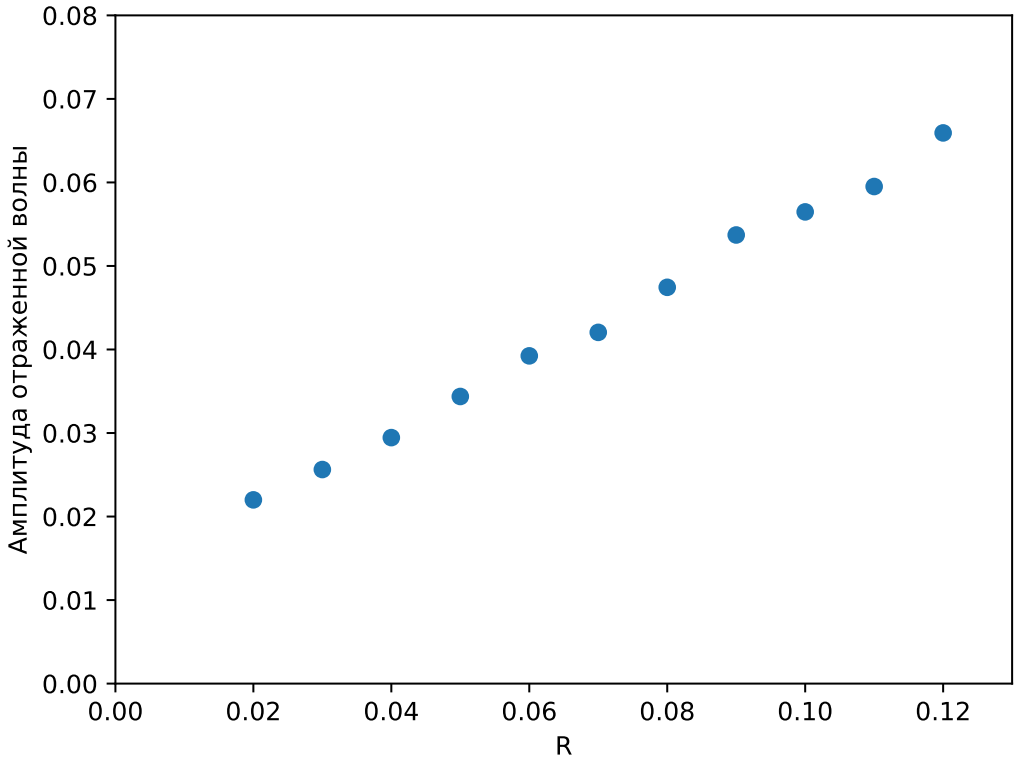
непроницаемой среды в области $|y| < 0.05, |x| > 0.05$. При прохождении волны между границами абсолютно акустически непроницаемых сред, возникает дифракция волн (рисунок 5c).

На левой и правой границах вычислительных областей, изображенных на рисунке 5, расположены неотражающие слои PML с $\sigma_x = e^{8|x-0.5|}$ при $|x - 0.5| > 0.3$. Большая часть амплитуды волны при попадании в область PML поглощается. Так после отражения от левой границы волны с начальной амплитудой $A_P = 0.1$ амплитуда отраженных колебаний уменьшается до $A_P = 0.004$.

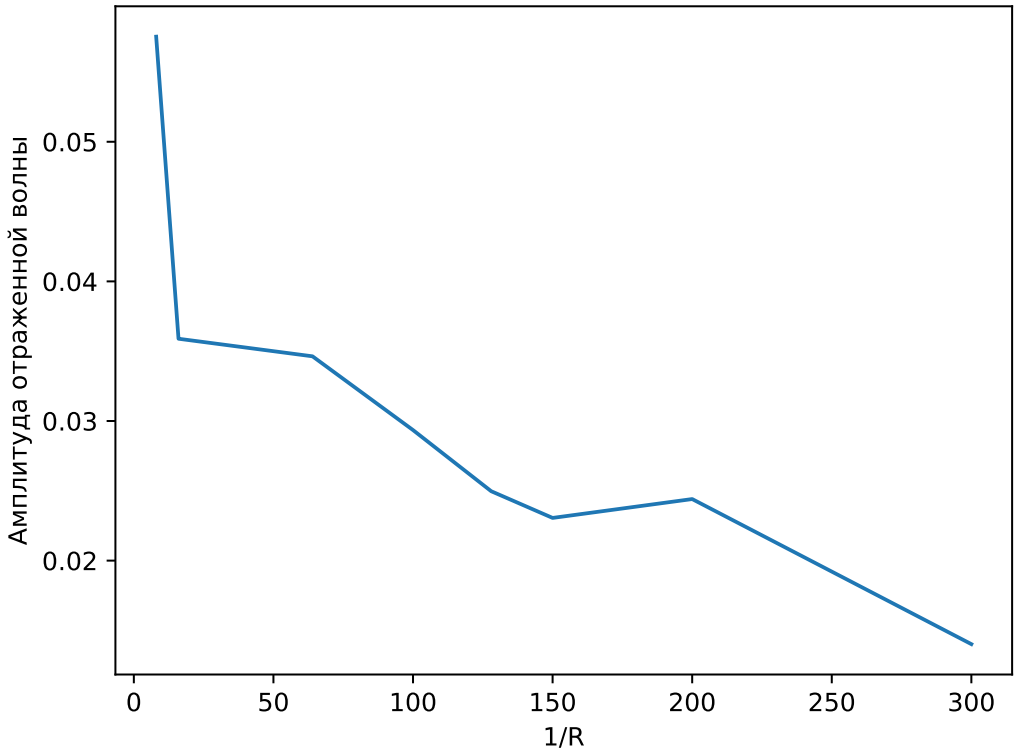
3.3. Задача обнаружения объекта внутри вычислительной области

В задачах сейсморазведки зачастую необходимо обнаружить область со скоростью звука, отличающейся от скорости звука в среде. Пусть в центре вычислительной области находится неоднородность в форме круга радиуса R с со скоростью звука $C_2 = \frac{1}{4}$ в среде со скоростью звука $C_1 = \frac{1}{2}$. Сенсоры расположены вокруг объекта. Источник колебаний расположен в середине нижней границы вычислительной области. Измерим значения давления на датчике, расположенным в точке $\left(\frac{1}{2}; \frac{3}{4}\right)$ (отмечен красной точкой на рисунке 6).

Рис. 6 – Отражение от неоднородности в форме круга радиусом $R = \frac{1}{300}$

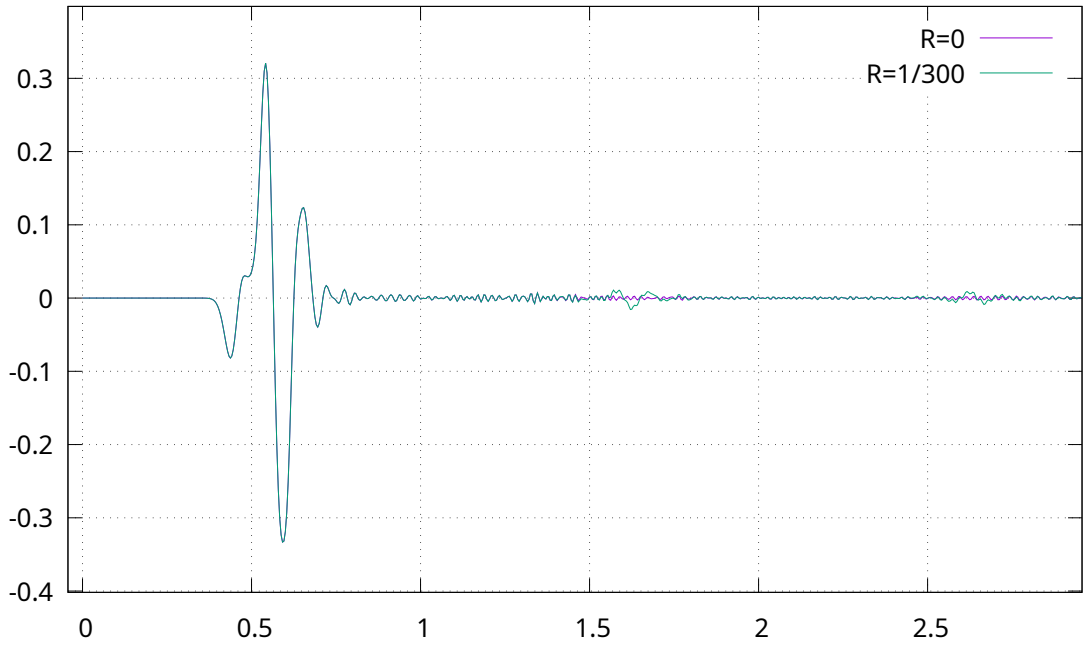


(a) Зависимость амплитуды отраженной волны от радиуса неоднородности в форме круга

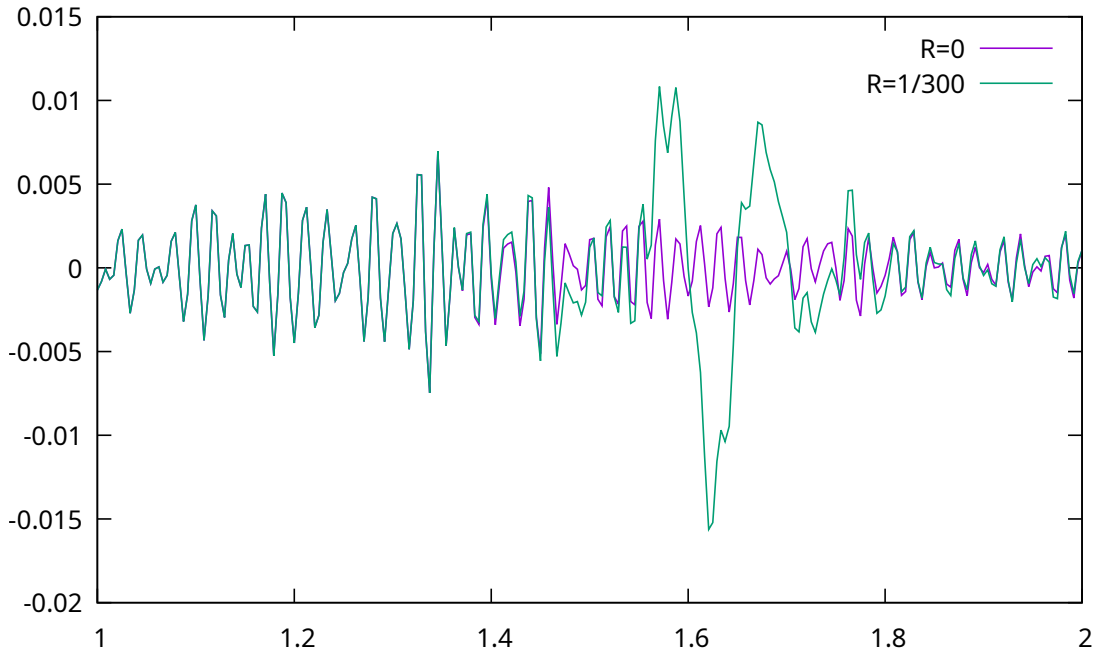


(b) Зависимость амплитуды от величины, обратной радиусу неоднородности

Рис. 7



(a) Показания датчика давления, расположенного в точке $\left(\frac{1}{2}; \frac{3}{4}\right)$, для круга $R = \frac{1}{300}$ и без круга $R = 0$



(b) График 8a для $x \in [1; 2]$

Рис. 8

С уменьшением радиуса амплитуда отраженной волны убывает (рисунок 7b). Так как волна отражается от границ вычислительной области, что вызывает интерференцию с отраженной волной, то существует минимальный радиус неоднородности, который может обнаружить датчик. Его можно определить при помощи вычислительных экспериментов. Наименьший радиус неоднородности в форме круга приблизительно равен $R_{\text{min}} \approx \frac{1}{300}$ при шаге сетки по пространству $h_x = h_y = \frac{1}{400}$. Зависимость значения датчика давления от времени представлена на рисунке 8a.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была изучена литература о двухслойных акустических схемах для задачи распространения волн. Рассмотрены одномерные и двумерные уравнения колебаний и переход от дифференциального уравнения второго порядка к акустической схеме. Была выведена двухслойная акустическая разностная схема с слоем PML, моделирующим неотражающие граничные условия на границе вычислительной области.

Далее была разработана и протестирована программа для численного решения волнового уравнения в неоднородной среде. Показано, что при увеличении количества узлов сетки норма ошибки уменьшается.

Было проанализировано поведение волны на границе сред с неоднородной акустической плотностью. Продемонстрированы моделирования физических явлений отражения волн от границы сред с разной скоростью звука, преломление волн при попадании в среду с меньшей скоростью звука и дифракция волн между границами абсолютно акустически плотных сред.

Было изучено, как отображается наличие неоднородности в области на показаниях удаленного датчика (на примере неоднородности круглой формы), в том числе зависимость показаний от размера зоны неоднородности. В результате вычислительных экспериментов получена амплитуда $A_P = 0.004$ волны с начальной амплитудой $A_{P0} = 0.1$,отраженной от поглощающего слоя PML. Оценен минимальный радиус $R_{\text{min}} \approx \frac{1}{300}$ объекта, который можно обнаружить при помощи датчиков отраженных волн для сетки с шагом $h_x = h_y = \frac{1}{400}$.

СПИСОК ЛИТЕРАТУРЫ

1. *Калиткин Н. Н.* Численные методы. — М. : Издательство «Наука», 1978.
2. *Schneider J. B.* Understanding the Finite-Difference Time-Domain Method. — 2010. — URL: www.eecs.wsu.edu/~schneidj/ufdtd.
3. *Johnson S. G.* Notes on Perfectly Matched Layers (PMLs) // CoRR. — 2021. — T. abs/2108.05348. — arXiv: 2108.05348. — URL: https://arxiv.org/abs/2108.05348.

ПРИЛОЖЕНИЕ А

Листинг программы

problem.hpp

```
#pragma once
#include <vector>
typedef std::vector<std::vector<double>> grid;
double Pt0(double x,double y);
double Vxt0(double x,double y);
double Vyt0(double x,double y);
double F(double x, double y, double t);
double Csq(double x,double y);
double Sigma_x(double x);
void init();
void output_layer(int m,double t, const grid &P1, const grid &Vx,const grid
    &Vy);
void finished();
extern int M;
extern int N;
extern double Time;
```

waves.cpp

```
#include <cstdlib>
#include <iostream>
#include <vector>
#include <cmath>
#include <sys/stat.h>
#include <sys/types.h>
#include <fstream>
#include "problem.hpp"
using namespace std;
/* u_t = v_x
 * v_t = c^2*u_x + F(x,t)
 * u(x,0) = m1
 * v(x,0) = \int m2(\xi) d \xi
 * u(0,t) = m3(t)
 * u(a,t) = m4(t)
 */
int main(int argc,char *argv[]){
    if(argc>1){
        M = atol(argv[1]);
        N = sqrt(M);
    }
    if(argc>2){
        N = atol(argv[2]);
    }
    grid P_a(N,std::vector<double>(N,0));
    grid P_b(N,std::vector<double>(N,0));
    grid Vx_a(N,std::vector<double>(N,0));
    grid Vx_b(N,std::vector<double>(N,0));
    grid Vy_a(N,std::vector<double>(N,0));
    grid Vy_b(N,std::vector<double>(N,0));
    grid Psi_a(N,std::vector<double>(N,0));
    grid Psi_b(N,std::vector<double>(N,0));


    grid &P0 = P_a;
    grid &P1 = P_b;
    grid &Vx0 = Vx_a;
    grid &Vx1 = Vx_b;
```

```
    grid &Vy0 = Vy_a;
    grid &Vy1 = Vy_b;
    grid &Psi0 = Psi_a;
    grid &Psi1 = Psi_b;

    double dt = Time/M;
    double dx = 1.0/N;
    double dy = dx;
    init();
    for(int j=0;j<N;j++){
        double y = j*dy;
        for(int i = 0; i < N; i++){
            double x = i*dx;
            P0[i][j] = Pt0(x,y);
            Vx0[i][j] = Vxt0(x,y);
            Vy0[i][j] = Vyt0(x,y);
        }
    }
    for(int m=0;m<M;m++){
        double t = m * dt;
        #pragma omp parallel for
        for(int j=0;j<N;j++){
            double y = j*dy;
            #pragma omp parallel for
            for(int i = 0; i < N; i++){
                double x = i*dx;
                if(i>=1&&j>=1&&i<N-1&&j<N-1){
                    P1[i][j] = P0[i][j] - Csq(x,y) * dt*(
                            (Vx0[i][j]-Vx0[i-1][j])/dx
                            + (Vy0[i][j] - Vy0[i][j-1])/dy
                            ) + dt*F(x,y,t) - dt * Sigma_x(x)*P0[i][j] +
                                dt*Psi0[i][j];
                }
                if(i<N-1&&j<N-1){
                    Vx1[i][j] = Vx0[i][j] - dt/dx *( P1[i+1][j] - P1[i][j])
                        - dt*Sigma_x(x)*Vx0[i][j];
                    Vy1[i][j] = Vy0[i][j] - dt/dy *( P1[i][j+1] - P1[i][j]);
                }
                if(i>=1&&j>=1&&i<N-1&&j<N-1){
                    Psi1[i][j] = Psi0[i][j] - dt/dy * (Vy0[i][j] -
                        Vy0[i][j-1])*Csq(x,y)*Sigma_x(x);
                }

            }

        }
        std::swap(P0,P1);
        std::swap(Vx0,Vx1);
        std::swap(Vy0,Vy1);
        std::swap(Psi0,Psi1);
        output_layer(m, t, P1,Vx1,Vy1);
    }
    finished();

    return 0;

}
```

problem.cpp

```
#include "problem.hpp"
#include <cmath>
#include <vector>
#include <fstream>
#include <iostream>
```

```
#include <sys/stat.h>
#include <sys/types.h>
using namespace std;
double Pt0(double x,double y){
    x-=0.5;
    y-=0.5;

    if(x*x + y*y < 1.0/100)
        return 0;
    return 0;
}
double Vxt0(double x,double y){
    return 0;
}
double Vyt0(double x,double y){
    return 0;
}
double F(double x, double y, double t){
        x-=0.5;
        y-=0.5;
    double R = 1.0/20;
        if(x*x + y*y<R*R)
                return cos(M_PI*2*10*t+sqrt(x*x+y*y)/R*M_PI)*10;
        return 0;
}
double Csq(double x,double y){
        x-=0.5;
        y-=0.5;
        if(fabs(y)<0.25)
                return 1;
//      if(fabs(y)>0.25)
//              return 2;
    return 0.15;
}
double Sigma_x(double x){
    return (x<0.2||x>0.8)*exp(8*fabs(x-0.5));
}
void init(){
    int dir_status = mkdir("data", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
    if(dir_status != 0 && errno != EEXIST){
        cerr << "Failed to create data directory" <<endl;
        exit(1);
    }
}
typedef std::vector<std::vector<double>> grid;
#define NN 300
#define VLEN 30
#define VRATE 24
double Time = 1.0;
int M = NN*NN*Time; // Time
int N = NN; // X Y

void output_layer(int m,double t, const grid &P1,const grid &Vx,const grid
    &Vy){
    int T=M/VLEN/VRATE;
    int print_T = 5*T;
    if(m%T)
        return;
    char fname[128];
    snprintf(fname,128, "data/%04d.pgm",m/T);
    if(m%print_T==0)
        fprintf(stderr,"[%d/%d] Computing image %s\033[0K\r",m/T,M/T,fname);
    std::ofstream data_stream(fname);
    if(!data_stream.is_open()){
        cerr <<"Can't open data file" <<endl;
```

```
        exit(1);
    }
    data_stream<<"P5"<<endl;
    data_stream<<N<<' '<<N<<endl;
    data_stream<<255<<endl;
    for(int j=0;j<N;j++){
        for(int i=0;i<N;i++){
            char byte = fmax(fmin(((P1[i][j]+1)/2)*255,255),0);
            data_stream<< byte;
        }
    }
    //std::cout <<std::endl<<std::endl;
    data_stream.close();

}
void finished(){

}
```

problem–detector.cpp

```
#include "problem.hpp"
#include <cmath>
#include <vector>
#include <fstream>
#include <iostream>
#include <sys/stat.h>
#include <sys/types.h>
using namespace std;
double Pt0(double x,double y){
    x-=0.5;
    y-=0.5;

    if(x*x + y*y < 1.0/100)
        return 0;
    return 0;
}
double Vxt0(double x,double y){
    return 0;
}
double Vyt0(double x,double y){
    return 0;
}
double F(double x, double y, double t){
        x-=0.5;
        y-=1.0;
    double R = 1.0/20;
        if(x*x + y*y<R*R && t < 0.1)
                return cos(M_PI*2*10*t+sqrt(x*x+y*y)/R*M_PI)*10*2;
        return 0;
}
#ifndef RR
#define RR 1.0/300
#endif
double Csq(double x,double y){
        x-=0.5;
        y-=0.5;
        double R = RR;//1.0/300;
        if(x*x+y*y<R*R)
                return 1.0/16;
//      if(fabs(y)>0.25)
//              return 2;
    return 0.25;
}
double Sigma_x(double x){
```

```
    return (x<1.0/8||x>7/8.0)*exp(8*fabs(x-0.5));
}
struct sensor{
        int i;
        int j;
};
std::vector<sensor> sensors;
void mksensor(int i, int j){
    char fname[128];
    snprintf(fname,128, "sensor/sensor_%dx%d.data",i,j);
        sensors.push_back({i,j});
        ofstream sensor_f(fname);
        sensor_f.close();

}
void init(){
    int dir_status = mkdir("data", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
    cerr << "R = " <<RR<<endl;
    if(dir_status != 0 && errno != EEXIST){
        cerr << "Failed to create data directory" <<endl;
        exit(1);
    }
    dir_status = mkdir("sensor", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
    if(dir_status != 0 && errno != EEXIST){
        cerr << "Failed to create sensor directory" <<endl;
        exit(1);
    }
        mksensor(N/2, N/4);
        mksensor(N/4, N/2);
        mksensor(N/8, N/2);
        mksensor(N/2,N*3/4);

}
typedef std::vector<std::vector<double>> grid;
#define NN 300
#define VLEN 30
#define VRATE 24
double Time = 3;
int M = 2*NN*NN*Time; // Time
int N = NN; // X Y

void output_layer(int m,double t, const grid &P1,const grid &Vx,const grid
    &Vy){
    int T=M/VLEN/VRATE;
    int print_T = 5*T;
    if(m%T)
        return;
    char fname[128];
    snprintf(fname,128, "data/%04d.pgm",m/T);
    if(m%print_T==0)
        fprintf(stderr,"[%d/%d] Computing image %s\033[0K\r",m/T,M/T,fname);
    std::ofstream data_stream(fname);
    if(!data_stream.is_open()){
        cerr <<"Can't open data file" <<endl;
        exit(1);
    }
    data_stream<<"P5"<<endl;
    data_stream<<N<<' '<<N<<endl;
    data_stream<<255<<endl;
    for(int j=0;j<N;j++){
        for(int i=0;i<N;i++){
            char byte = fmax(fmin(((P1[i][j]+1)/2)*255,255),0);
            data_stream<< byte;
        }
    }
```

```
    //std::cout <<std::endl<<std::endl;
    data_stream.close();
        for(sensor &s: sensors){
                snprintf(fname,128, "sensor/sensor_%dx%d.data",s.i,s.j);
                std::ofstream sensor(fname,ios_base::app);
                sensor << t<<' ' << P1[s.i][s.j]<<' ' <<Vx[s.i][s.j]<<'
                    '<<Vy[s.i][s.j]<<endl;
                sensor.close();
        }
}
void finished(){

}
```

problem–difraction.cpp

```
#include "problem.hpp"
#include <cmath>
#include <vector>
#include <fstream>
#include <iostream>
#include <sys/stat.h>
#include <sys/types.h>
using namespace std;
double Pt0(double x,double y){
    x-=0.5;
    y-=0.5;

    if(x*x + y*y < 1.0/100)
        return 0;
    return 0;
}
double Vxt0(double x,double y){
    return 0;
}
double Vyt0(double x,double y){
    return 0;
}
double F(double x, double y, double t){
        x-=0.5;
        y-=1.0;
    double R = 1.0/20;
        if(x*x + y*y<R*R)
                return cos(M_PI*2*10*t+sqrt(x*x+y*y)/R*M_PI)*10;
        return 0;
}
double Csq(double x,double y){
        x-=0.5;
        y-=0.5;
        if(fabs(y)<0.05 && fabs(x)>0.05)
                return 0.0;
//      if(fabs(y)>0.25)
//              return 2;
    return 1;
}
double Sigma_x(double x){
    return (x<0.2||x>0.8)*exp(8*fabs(x-0.5));
}
void init(){
    int dir_status = mkdir("data", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
    if(dir_status != 0 && errno != EEXIST){
        cerr << "Failed to create data directory" <<endl;
        exit(1);
    }
}
```

```
typedef std::vector<std::vector<double>> grid;
#define NN 300
#define VLEN 30
#define VRATE 24
double Time = 1.5;
int M = NN*NN*Time*Time; // Time
int N = NN; // X Y

void output_layer(int m,double t, const grid &P1,const grid &Vx,const grid
    &Vy){
    int T=M/VLEN/VRATE;
    int print_T = 5*T;
    if(m%T)
        return;
    char fname[128];
    snprintf(fname,128, "data/%04d.pgm",m/T);
    if(m%print_T==0)
        fprintf(stderr,"[%d/%d] Computing image %s\033[0K\r",m/T,M/T,fname);
    std::ofstream data_stream(fname);
    if(!data_stream.is_open()){
        cerr <<"Can't open data file" <<endl;
        exit(1);
    }
    data_stream<<"P5"<<endl;
    data_stream<<N<<' '<<N<<endl;
    data_stream<<255<<endl;
    for(int j=0;j<N;j++){
        for(int i=0;i<N;i++){
            char byte = fmax(fmin(((P1[i][j]+1)/2)*255,255),0);
            data_stream<< byte;
        }
    }
    //std::cout <<std::endl<<std::endl;
    data_stream.close();

}
void finished(){

}
```

problem–refraction.cpp

```
#include "problem.hpp"
#include <cmath>
#include <vector>
#include <fstream>
#include <iostream>
#include <sys/stat.h>
#include <sys/types.h>
using namespace std;
double Pt0(double x,double y){
    x-=0.5;
    y-=0.5;

    if(x*x + y*y < 1.0/100)
        return 0;
    return 0;
}
double Vxt0(double x,double y){
    return 0;
}
double Vyt0(double x,double y){
    return 0;
}
double F(double x, double y, double t){
```

```
        x-=0.5;
        y-=1.0;
    /*double R = 1.0/20;
        if(x*x + y*y<R*R)
                return cos(M_PI*2*10*t+sqrt(x*x+y*y)/R*M_PI)*10;*/
    if(abs(y)<1.0/20&&abs(x)<4.0/20)
        return 10*cos(M_PI*2*10*t+fabs(y)*20*M_PI);
        return 0;
}
double Csq(double x,double y){
        x-=0.5;
        y-=0.5;
        if(fabs(-y+x)<0.1)
                return 0.15;
//      if(fabs(y)>0.25)
//              return 2;
    return 1;
}
double Sigma_x(double x){
    return (x<0.2||x>0.8)*exp(8*fabs(x-0.5));
}
void init(){
    int dir_status = mkdir("data", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
    if(dir_status != 0 && errno != EEXIST){
        cerr << "Failed to create data directory" <<endl;
        exit(1);
    }
}
typedef std::vector<std::vector<double>> grid;
#define NN 300
#define VLEN 30
#define VRATE 24
double Time = 2.0;
int M = NN*NN*Time; // Time
int N = NN; // X Y

void output_layer(int m,double t, const grid &P1,const grid &Vx,const grid
    &Vy){
    int T=M/VLEN/VRATE;
    int print_T = 5*T;
    if(m%T)
        return;
    char fname[128];
    snprintf(fname,128, "data/%04d.pgm",m/T);
    if(m%print_T==0)
        fprintf(stderr,"[%d/%d] Computing image %s\033[0K\r",m/T,M/T,fname);
    std::ofstream data_stream(fname);
    if(!data_stream.is_open()){
        cerr <<"Can't open data file" <<endl;
        exit(1);
    }
    data_stream<<"P5"<<endl;
    data_stream<<N<<' '<<N<<endl;
    data_stream<<255<<endl;
    for(int j=0;j<N;j++){
        for(int i=0;i<N;i++){
            char byte = fmax(fmin(((P1[i][j]+1)/2)*255,255),0);
            data_stream<< byte;
        }
    }
    //std::cout <<std::endl<<std::endl;
    data_stream.close();

}
void finished(){
```

```
}
```

problem–simple.cpp

```
#include "problem.hpp"
#include <cmath>
#include <vector>
#include <fstream>
#include <iostream>
#include <sys/stat.h>
#include <sys/types.h>
#include "problem.hpp"
#include <cmath>
using namespace std;
double Pt0(double x,double y){
    return sin(M_PI*x)*sin(M_PI*y);
}
double Vxt0(double x,double y){
    return 0;
}
double Vyt0(double x,double y){
    return 0;
}
double F(double x, double y, double t){
        return 0;
}
double Csq(double x,double y){
    return 1;
}
double Sigma_x(double x){
    return 0;
}
double P(double x,double y,double t){
    return cos(M_PI*M_SQRT2*t)*sin(M_PI*x)*sin(M_PI*y);
}
void init(){
    cerr << "Test 1" <<endl;
}
typedef std::vector<std::vector<double>> grid;
double max_diff = 0;
int M = 10000; // Time
int N = 100; // X Y
double Time = 1.0;
void output_layer(int m,double t, const grid &P1,const grid &Vx,const grid
    &Vy){
    double dt = 1.0/M;
    double dx = 1.0/N;
    double dy = dx;
    for(int j=0;j<N;j++){
        for(int i=0;i<N;i++){
            double diff = fabs(P1[i][j] - P(i*dx,j*dy,t) );
            max_diff = fmax(max_diff,diff);
        }
    }
    //std::cout <<std::endl<<std::endl;
    //if(m%100==0)
    //    fprintf(stderr,"[%d/%d] max_diff = %lf\033[0K\r",m,M,max_diff);

}
void finished(){

        printf("max_diff = %lf\n",max_diff);
}
```

problem–analytic.cpp

```
#include "problem.hpp"
#include <cmath>
#include <vector>
#include <fstream>
#include <iostream>
#include <sys/stat.h>
#include <sys/types.h>
#include "problem.hpp"
#include <cmath>
using namespace std;
double Pt0(double x,double y){
    return sin(M_PI*x)*sin(M_PI*y);
}
double Vxt0(double x,double y){
    return 0;
}
double Vyt0(double x,double y){
    return 0;
}
double F(double x, double y, double t){
        return 0;
}
double Csq(double x,double y){
    return 1;
}
double Sigma_x(double x){
    return 0;
}
double P(double x,double y,double t){
    return cos(M_PI*M_SQRT2*t)*sin(M_PI*x)*sin(M_PI*y);
}
void init(){
    cerr << "Test 1" <<endl;
    int dir_status = mkdir("data", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
    if(dir_status != 0 && errno != EEXIST){
        cerr << "Failed to create data directory" <<endl;
        exit(1);
    }
}
typedef std::vector<std::vector<double>> grid;
double max_diff = 0;
int M = 10000; // Time
int N = 100; // X Y
double Time = 1.0;
void output_layer(int m,double t, const grid &P1,const grid &Vx,const grid
    &Vy){
    double dt = 1.0/M;
    double dx = 1.0/N;
    double dy = dx;
    for(int j=0;j<N;j++){
        for(int i=0;i<N;i++){
            double diff = fabs(P1[i][j] - P(i*dx,j*dy,t) );
            max_diff = fmax(max_diff,diff);
        }
    }
    //std::cout <<std::endl<<std::endl;
    int T=100;
    int print_T = 5*T;
    if(m%T)
        return;
    char fname[128];
    snprintf(fname,128, "data/%04d.pgm",m/T);
    if(m%print_T==0)
```

```
        fprintf(stderr,"[%d/%d] Computing image %s\033[0K\r",m/T,M/T,fname);
    std::ofstream data_stream(fname);
    if(!data_stream.is_open()){
        cerr <<"\nCan't open data file" <<endl;
        exit(1);
    }
    data_stream<<"P5"<<endl;
    data_stream<<N<<' '<<N<<endl;
    data_stream<<255<<endl;
    for(int j=0;j<N;j++){
        for(int i=0;i<N;i++){
            char byte = fmax(fmin(((P1[i][j]+1)/2)*255,255),0);
            data_stream<< byte;
        }
    }
    //std::cout <<std::endl<<std::endl;
    data_stream.close();
    //if(m%100==0)
    //    fprintf(stderr,"[%d/%d] max_diff = %lf\033[0K\r",m,M,max_diff);

}
void finished(){

        printf("max_diff = %lf\n",max_diff);
}
```

ПЛАН-ГРАФИК

выполнения курсовой работы

обучающегося Яковлева О. В.

Наименование этапа работ	Трудоемкость выполнения, час.	Процент к общей трудоемкости выполнения	Срок предъявления консультанту
Получение и согласование задания	0,3	0,8	28 неделя
Знакомство с литературой по теме курсовой работы	2,7	7,5	29 неделя
Вывод уравнений для неотражающего PML слоя	4,5	12,5	30 неделя
Разработка программы для численного решения волнового уравнения	9,5	26,39	32 неделя
Сранение результата работы программы с аналитическим решением	3,6	10,0	34 неделя
Проведение расчетов и анализ результатов вычислительных экспериментов	8,4	23.33	36 неделя
Составление и оформление пояснительной записки и подготовка к защите	2,7	7,5	37 неделя
Защита	0,3	0,8	38 неделя
Итого	36	100	-