

**РЕФЕРАТ**

Выпускная квалификационная работа 66 с., 9 рис., 20 источн, 7 прил., 1 табл.

МЕТАМАТЕРИАЛЫ, МОДЕЛИРОВАНИЕ, ЧИСЛЕННЫЕ АЛГОРИТМЫ, МЕТОД КОНЕЧНЫХ РАЗНОСТЕЙ, PYTHON, РАЗРАБОТКА ПРИЛОЖЕНИЯ

Объектом исследования является создание приложения для моделирования метаматериалов методом конечных разностей.

Цель работы – разработка приложения для моделирования метаматериалов методом конечных разностей. Основная идея метода – замена производных в исходном дифференциальном уравнении конечно-разностными выражениями и решение полученной системы линейных алгебраических уравнений. Также будет рассмотрен способ решения данной задачи для множества ячеек.

Результатом исследования станет приложение, способное моделировать метаматериал, являющийся важными для многих областей науки и имеет широкое практическое применение.

|  |  |  |
| --- | --- | --- |
| Введение........................................................................................................... | | 4 |
| 1 | Метод конечных разностей...................................................................... | 6 |
|  | 1.1 Моделирование метаматериалов....................................................... | 6 |
|  | 1.2 Постановка задачи............................................................................... | 8 |
|  | 1.3 Метод конечных разностей численного решения............................ | 10 |
|  | 1.4 Достоинства и недостатки метода....................................................... | 13 |
| 2 | Программная реализация.......................................................................... | 16 |
|  | 2.1 Выбор языка и алгоритм программы................................................. | 16 |
|  | 2.2 Описание программы.......................................................................... | 17 |
|  | 2.3 Результат работы программы............................................................. | 20 |
| 3 | Решение задачи для периодического акустического метаматериала... | 23 |
|  | 3.1 Теорема Флоке-Блоха…….................................................................. | 23 |
|  | 3.2 Сведение к задаче на собственные значения..................................... | 24 |
|  | 3.3 Программная реализация..................................................................... | 29 |
| 4 | Разработка приложения............................................................................. | 31 |
|  | 4.1 Функциональные требования.............................................................. | 31 |
|  | 4.2 Нефункциональные требования.......................................................... | 33 |
|  | 4.3 Основные компоненты......................................................................... | 34 |
| Заключение....................................................................................................... | | 36 |
| Список использованных источников............................................................. | | 37 |
| Приложение А.................................................................................................. | | 40 |
| Приложение Б................................................................................................... | | 44 |
| Приложение В................................................................................................... | | 46 |
| Приложение Г.................................................................................................. | | 50 |
| Приложение Д................................................................................................... | | 52 |
| Приложение Е................................................................................................... | | 59 |
| Приложение Ж.................................................................................................. | | 60 |

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ**

Фононные кристаллы и акустические метаматериалы вызывают растущий научный интерес к самым разнообразным технологическим приложениям – от снижения шума до ультразвуковой визуализации, телекоммуникаций, терморегулирования и термоэлектричества. Они представляют собой искусственно структурированные композитные материалы, которые позволяют манипулировать дисперсионными свойствами вибрационных волн.

Данные материалы обладают уникальными свойствами. Например, при определенных условиях могут образовываться абсолютные акустические запрещенные зоны. Это спектральные зоны, в которых распространение волн запрещено независимо от направления распространения. Данная особенность может иметь широкое практическое применение, например, фильтрация частот, направление волн, мультиплексирование и демультиплексирование. Свойства волнового вектора возникают из проходящих зон с уникальными преломляющими характеристиками. Одной из таких характеристик является отрицательное преломление, его можно использовать для достижение фокусировки волн с помощью плоских линз. При определенных условиях, связанных с усилением затухающих волн, также можно получить визуализацию со сверхвысоким разрешением, то есть сформировать изображения, которые превосходят предел разрешения Релея.

Данные свойства и возможные их применения лишь одни из многих. Для их изучения можно применять численные алгоритмы моделирования. Они помогают предсказывать и анализировать поведение волн при взаимодействии с различными материалами и структурами.

Целью данной выпускной квалификационной работы является изучение метода конечных разностей для моделирования поведения звуковой волны в одной ячейке метаматериала. Будет разобрана теория метода, основные достоинства и недостатки, алгоритм программы и его реализация. Также в работе планируется рассмотреть аналогичное решение задачи моделирования метаматериала, состоящего из множества ячеек.

С помощью полученных знаний и результатов исследования можно не только углубить понимание моделирования акустических метаматериалов, но и применить полученные навыки и методы в решении конкретных задач и проектов.

Первая глава выпускной квалификационной работы содержит разбор понятия метаматериала и разбор метода конечных разностей, а также постановку задачи, которая будет решаться представленным методом.

Во второй главе разобрана программная реализация метода конечных разностей.

В третьей главе разбирается метод решения для метаматериала с периодически расположенными ячейками и его программная реализация.

В четвертой главе разбирается создание программного средства для моделирования ячейки метаматериала.

Перечисленные выше главы совпадают также с задачами, которые решаются в данной работе.

**1 Метод конечных разностей**

**1.1 Метаматериалы**

Метаматериалами называют композитные материалы, свойства которых обусловлены не столько индивидуальными физическими свойствами их компонентов, сколько микроструктурой. Метаматериалы можно рассматривать как однородные среды, обладающие электромагнитными или акустическими свойствами, сложно достижимыми технологически, либо не встречающимися в природе. Приставка «мета-» переводится с греческого как «многочисленный», так как эти материалы состоят из очень большого числа одинаковых или изменяющихся определённым, заданным заранее образом структурных единиц – "метаатомов" для получения желаемых свойств [1]. Их параметры зависят как от свойств отдельных метаатомов, так и от их распределения в пространстве.

Широкое распространение получили метаматериалы с электромагнитными и оптическими свойствами. Они по праву занимают ведущее положение в многочисленных исследованиях. Уникальным свойствам данных материалов нашли приложение в микроэлектронике и радиоэлектронике. Так, в журнале Advanced Science была показана возможность применения метаматериалов в магнонике – спиновой электронике [2]. Метаматериалы также могут обладать отрицательными коэффициентом преломления магнитных волн. В таком случае рассматривается применение метаматериалов при производстве миниатюрных антенн, увеличивающих полосу пропускания и силу излучения излучателя [3]. Для лучшего понимания примерной структуры на рисунке 1 изображен метаматериал, который из-за вышеописанного свойства обеспечивает отрицательную магнитную проницаемость. В оптике данный коэффициент преломления может быть полезен, например, для изготовления линз, способных проводить поляризацию света. Учитывая вышесказанное, сложно переоценить потенциал применения метаматериала в данных областях.

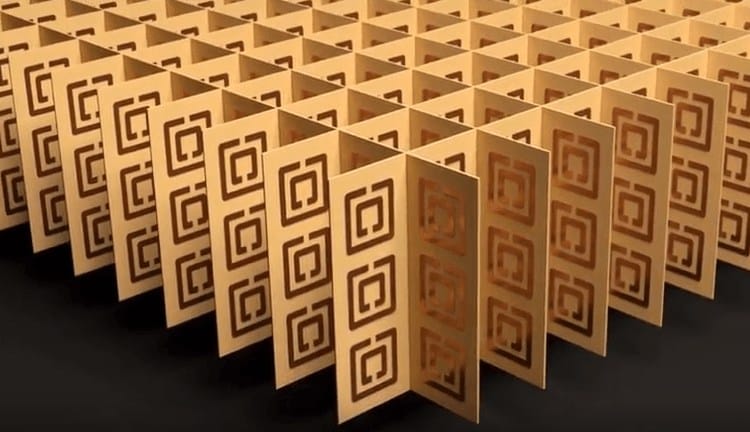


Рисунок 1 – Схема решетки метаматериала, используемого для демонстрации отрицательного показателя преломления, который придает материалу отрицательную магнитную проницаемость

Фононные кристаллы являются своего рода разновидностью метаматериалов. Фононные кристаллы обладают различными частотными характеристиками, например, такими, как запрещённые частотные зоны. В запрещённой зоне, распространение волн фактически невозможно. Это явление может быть использованным в широком спектре технологий и в различных масштабах. Приложения включают в себя упругую или акустическую фокусировку, минимизацию вибрации, звуковую коллимацию, акустическую маскировку, оптомеханические волновые преобразования в фотонных устройствах, снижение теплопроводности в полупроводниках и др. [4].

Помимо очевидного разделения по количеству измерений, в которых свойства меняются периодически, фононные кристаллы условно могут быть разделены на три категории: акустические фононные кристаллы с матрицей из жидкости, упругие фононные кристаллы с матрицей из сплошного упругого вещества и пьезоэлектрические/магнитные фононные кристаллы. Акустические метаматериалы обладают дополнительным свойством локального резонанса и, хотя их часто проектируют как периодические структуры, их свойства не зависят от периодичности [5]. Существование запрещённых зон у фононных кристаллов делает возможным многочисленные потенциальные применения. Например, фононные кристаллы могут быть использованы как эффективные звуковые изоляторы, полезные для акустической изоляции вибрирующих структур, гироскопы или механические резонаторы, устройств жёсткого сцепления.

В последние годы возрос интерес использования метаматериалов в таких областях, как материаловедение, физику, машиностроение и гражданское строительство, и т. д. [6]. В настоящее время активно изучаются способы использования уникальных свойств для звукопоглощения и снижения шума, а также улучшения и изменения акустических характеристик [7]. Например, исследователи из Бостонского университета разработали метаматериал способный блокировать до 94% шума. В России также активно исследуются акустические свойства различных материалов и их применения.

Поскольку метаматериалы являются достаточно востребованными для многих технологий, то моделирование их структуры является достаточно важной задачей. Существует множество способов моделирования. В дальнейшем будет рассмотрен один из таких способов, метод конечных разностей.

**1. 2 Постановка задачи**

Пусть имеются материалы A и B, них состоит ячейка метаматериала, как на рисунке 2. Эти материалы описываются плотностью и модулем сдвига. Обозначим µA, ρA – соответственно модуль сдвига материала A, плотность материала A, µB, ρB – соответственно модуль сдвига материала B, плотность материала B.

Когда на кристалл воздействуют звуковые волны, происходят колебания A и B. Они задаются уравнением Гельмгольца:

(1)

где

*µ* ∈ {*µA, µB*}, *ρ* ∈ {*ρA, ρB*};

ω – круговая частота колебаний;

*u (x, y)* – функция смещения, а :

. (2)

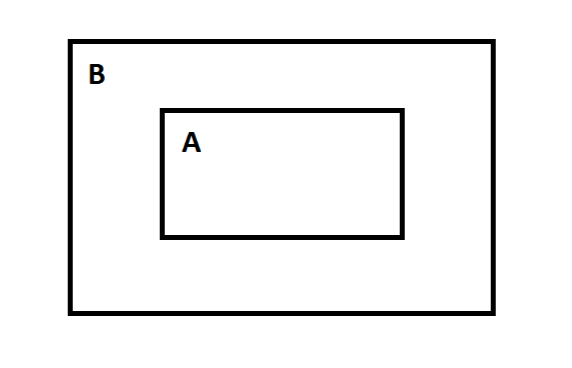


Рисунок 2 – Ячейка метаматериала, состоящая из материалов A и B

Граничные условия материала A на границе ∂A (∀ (x, y) ∈ ∂A) задаются как:

, (3)

где

– вектор нормали к границе в текущей точке, причем, как известно || = 1.

Из свойств нормали известно:

(4)

где

**–** градиент функции u.

Далее для тестов была рассмотрена задача со следующими граничными условиями на границе ячейки:

. (5)

**1.3 Метод конечных разностей численного решения**

Метод конечных разностей состоит из трех основных этапов: построение сетки узлов, построение конечно-разностных уравнений и решение системы [8].

Рассмотрим декартовую систему координат. Область непрерывного изменения аргументов заменяется дискретным множеством точек (узлов), которое называется сеткой или решеткой [9]. Выбирается шаг *hx*и *hy*. Соответственно строится однородная сетка, n на m, как на рисунке 3, в данном случае *hx*= *hy*.

Соответственно можно определить *xi* и *yi*, а также шаг *hx*и *hy*:

, (6)

, (7)

, (8)

, (9)

где

i = 1,…,n для ;

i = 1,…, m для .

Конечно-разностные уравнения в МКР получают путем замены производных в исходном дифференциальном уравнении соответствующими конечно-разностными выражениями. Конечно-разностные выражения для какой-либо частной производной можно получить из разложения функции в ряд Тейлора по соответствующей переменной. Существует несколько типов конечных разностей, в зависимости от того, какие узловые точки взяты для ее вычисления.

Центральные можно записать так:

, (10)

(11)

(12)

(13)

Левые выражения запишутся как:

, (14)

(15)

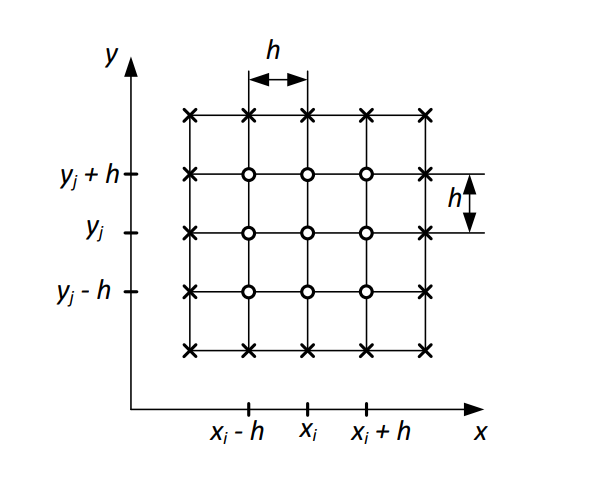


Рисунок 3 – Однородная сетка с шагом h

(16)

(17)

Соответственно, правые:

, (18)

(19)

(20)

(21)

Получается система линейных алгебраических уравнений вида:

(22)

где

*A* – матрица коэффициентов неизвестных при подстановке конечно-разностных уравнений в уравнение (1) и известных граничных условий;

*u* – вектор искомых значений системы, решение задачи;

*b* – вектор известных значений на границе.

Система решается любым подходящим методом для СЛАУ, но лучше в дальнейшем подобрать метод под вид матрицы *A*.

Метод конечных разностей является достаточно универсальным численным методом, ориентированным на решение задач с граничными условиями как в одномерных, так и многомерных системах. Общей идеей МКР является сведение исходной задачи с граничными условиями (краевой задачи) к более простой задаче решения системы линейных или нелинейных алгебраических уравнений. Вид получаемой системы алгебраических уравнений зависит от вида исходного дифференциального уравнения.

**1.4 Достоинства и недостатки метода**

Метод конечных разностей имеет свои достоинства и недостатки.

К достоинствам можно отнести следующие 6 пунктов:

1) Метод конечных разностей легко программируется, особенно для задач с регулярными областями (прямоугольные, кубические сетки). Основные формулы конечно-разностных интуитивно понятны. Применение этого метода нередко характеризуется относительной простотой построения решающего алгоритма и его программной реализации [10].

2) Данный метод подходит для широкого класса дифференциальных уравнений.

3) При реализации данного метода не требуется сложного преобразования исходного уравнения, достаточно заменить производные их разностными аналогами.

4) При уменьшении шага сетки погрешность аппроксимации снижается и результат вычислений приемлем при умеренных вычислительных затратах.

5) Данный метод может комбинироваться с методами решения СЛАУ.

6) Для простых задач построение разностной схемы выполняется быстрее.

К недостаткам отнесем следующие 6 пунктов:

1) Данный метод плохо работает с криволинейными границами, а неоднородные сетки усложняют реализацию и снижают точность. Пример наиболее часто используемых сеток можно увидеть на рисунке 4.

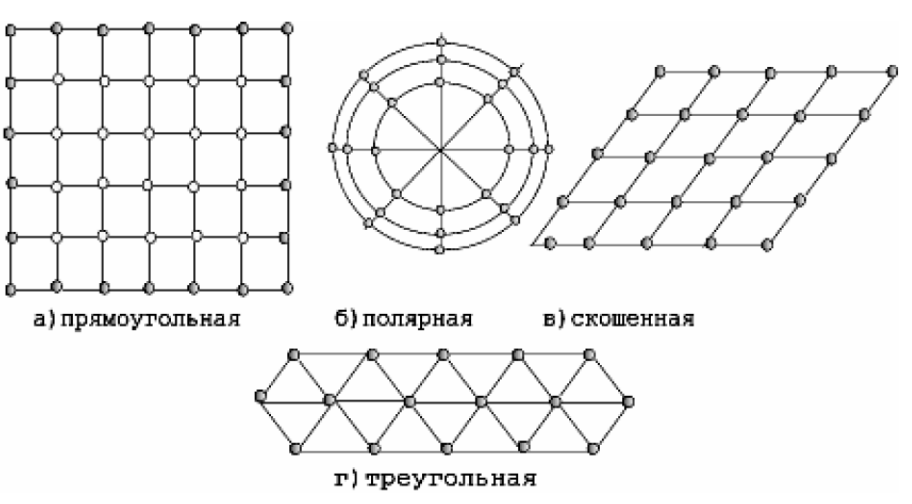


Рисунок 4 – Наиболее часто используемые виды сеток

2) При увеличении узлов сетки быстро растет вычислительная сложность, что приводит к огромным матрицам, то есть к высокому порядку систем алгебраических уравнений. Следовательно, решение СЛАУ будет тратить значительное количество вычислительных ресурсов [11].

3) Сложность аналитического исследования свойств разностной схемы.

Несмотря на вышеперечисленные недостатки, метод конечных разностей подходит для поставленной задачи моделирования акустического материала.

**2 Программная реализация**

**2.1 Выбор языка и алгоритм программы**

Метод конечных разностей может быть реализован на языках программирования высокого уровня без каких-либо трудностей. Для данной задачи выберем Python – простой, логичный язык с понятным синтаксисом. Блоки кода отделяются отступами, за счёт меньшего объёма код также воспринимается проще [12]. Конечно, этот язык не очень высокопроизводителен, но стандартные библиотеки и возможные дополнительные модули помогут при решении многих подзадач.

Перед написанием кода лучше составить алгоритм, что облегчит дальнейшую работу.

Алгоритм МКР состоит из 5 пунктов:

1) Задаем размер области материала B, границы материала A, модули сдвига, плотности, круговую частоту колебаний.

2) Выбираем шаг hx для построения сетки. От этих значений будет зависеть точность полученного ответа.

3) Составляем систему СЛАУ. Стоит отметить, что именно этот этап в методе конечных разностей самый сложный, он зависит от исходного уравнения, количества параметров в искомой функции. Зачастую в этом методе для многих задач матрица принимает трехдиагональный вид, но не в этом случае, хотя немного и будет ее напоминать.

4) Решаем СЛАУ. Систему можно решать любым подходящим под вид матрицы методом. Поскольку важнейший этап данной задачи – составление системы линейных алгебраических уравнений, то позволим себе не выбирать какой-то метод и программировать его, а просто возьмем функцию из встроенных библиотек.

5) Визуализируем результаты. Отметим, что анализ и какая-либо работа с результатами – это уже другая задача. Тогда в этом случае просто будем выводить цветовую карту решения.

Таким образом, есть готовый алгоритм, по которому можно ориентироваться при написании кода.

**2.2 Описание программы**

В первых строчках кода, обычно импортируют нужные библиотеки, если это необходимо. Для данной программы понадобится NumPy. Это открытая бесплатная Python-библиотека для работы с многомерными массивами, ее чаще всего используют в анализе данных и обучении нейронных сетей – в каждой из этих областей нужно проводить много вычислений с такими матрицами. Библиотека написана частично на Python, а частично на C и C++ – в тех местах, которые требуют скорости. Кроме того, код NumPy оптимизирован под большинство современных процессоров. Как и у Matlab, для NumPy существуют пакеты, расширяющие её функциональность, — например, библиотека SciPy или Matplotlib. Последняя понадобится для визуализации результатов, ее также импортируем. Matplotlib — это библиотека с открытым исходным кодом для визуализации данных в Python. С помощью неё можно создавать точечные и круговые диаграммы, линейные графики, гистограммы, диаграммы ошибок, 3D-графики. Также понадобится импортировать Seaborn — библиотеку для создания статистических графиков на Python. Она построена на основе Matplotlib и тесно интегрируется со структурами данных pandas. Seaborn помогает изучить и понять данные. Его функции построения графиков работают с датасетами и выполняют все необходимы преобразования для создания информативных графиков [13].

Далее определяем функцию, которая будет определять находится ли узел сетки в зоне материала A или нет. Соответственно у нее будет 6 входных параметров: координата по оси X, координата по оси Y и четыре точки, определяющие границу материала A. Функция будет возвращать True, если узел находится в зоне материала A, иначе False. Также сразу определим функцию, которая будет выводить полученный результат в виде цветовой карты матрицы значений решения системы.

Основная функция этой программы будет решать поставленную задачу по построению и решению СЛАУ. Зададим основные параметры такие, как границы материала A, размер ячейки материала B, шаг для сетки, модуль сдвига, плотности материалов, круговую частоту колебаний. Для данного метода существует много видов сеток, однако для нашей задачи выберем самую обычную квадратную сетку.

Далее создаем матрицы A и b с необходимой размерностью. Пока что все их значения – нули. Причем A – двумерный массив (квадратная матрица), а b – одномерный массив (вектор). Чтобы было легче вычислять конечные разности, сразу введем две переменные:

, (23)

где

– выбранный шаг.

Следующий этап самый сложный, это составление системы линейных алгебраических уравнений. Для удобства вычислений введем переменную count. Она будет отвечать за номер узла, относительно которого будет составляться конечно-разностное уравнение или изменение значений матриц A и b. В двойном цикле while это удобнее всего реализовать. В каждой итерации сначала проверяем принадлежность точки к материалу A и, исходя из этой информации, задаем значение плотности и модуля сдвига соответствующего материала. Далее проверяем более точное месторасположение точки и в зависимости от случая, возможные случаи на рисунке 5, меняем значения матриц системы. Если это случай 1, то есть узел на границе – соответствующие значение A[count][count] и B[count] принимаем за единицу, то есть соответствующее значение искомой функции однозначно определено. Так действуем исходя из условий задачи – это места, где “входит” звуковая волна. Если это случай 2, то составляется конечно-разностное уравнение:

, (24)

где

*µ* ∈ {*µA, µB*}, *ρ* ∈ {*ρA, ρB*} в зависимости от принадлежности узла к одному из материалов A и B.

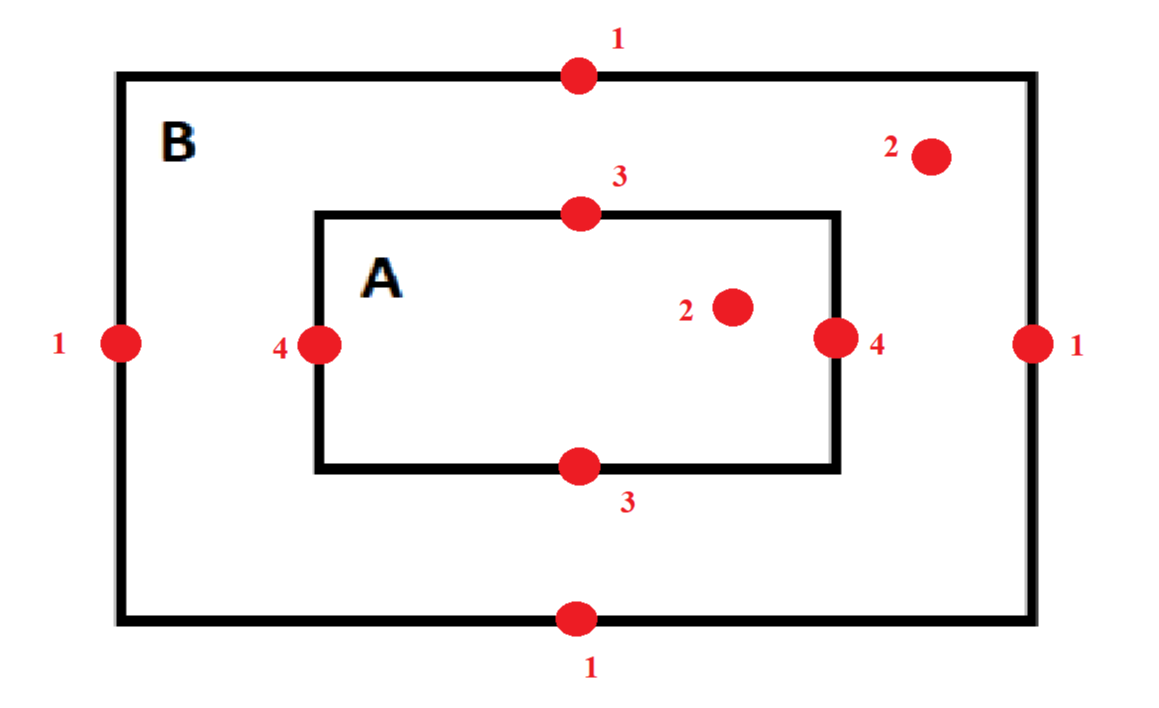


Рисунок 5 – Варианты расположения узлов

Раскрываем скобки и для каждого u в этом выражении коэффициент записывается в соответствующую ячейку матрицы A, а соответствующее данному u элемент вектора-матрицы b остается нулевым. Если же узел попадает в случай 3 или 4, то тогда следует учесть граничные условия ∂A (3). Они дописываются в уравнение (24) через конечные разности (10) и (11) с противоположными знаками, не меняя равенство нулю:

. (25)

Распишем 2 последних слагаемых (24):

(26)

где

,

Таким образом, получили систему линейных алгебраических уравнений для данной задачи. Вид матрицы *A* для этой задачи приведен в приложении Б. Для решения системы будем использовать модуль numpy.linalg, позволяющий делать многие операции из линейной алгебры. Функция numpy.linalg.solve позволит найти искомый вектор u. Остается только вывести результат. Пример такого вывода можно увидеть на рисунке 6.

Текст написанной программы приведен в приложении А.

**2.3 Результат работы программы**

Чтобы проверить результат работы программы необходимо удостовериться, что при уменьшении шага результат работы функции решение сходится к определенному значению. Возьмем определенные точки ячейки метаматериала. Будем постепенно уменьшать шаг и смотреть, как

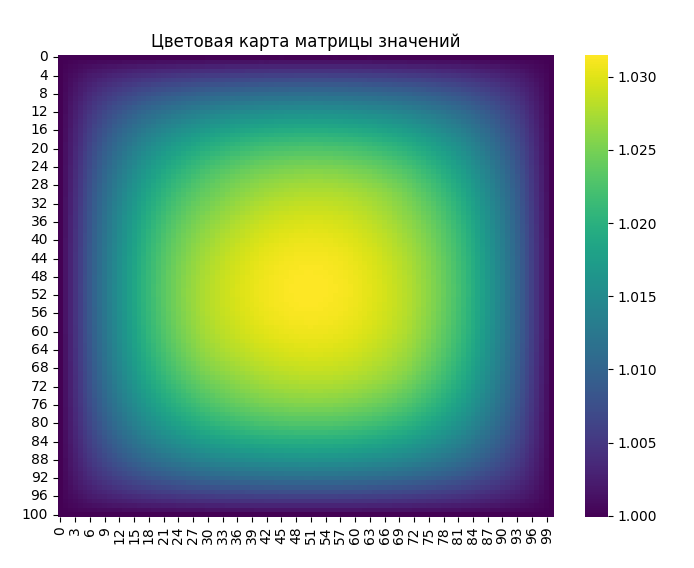


Рисунок 6 – Пример вывода программы

изменяются значения в этих точках. Полученные результаты представлены в таблице 1.

Таблица 1 – Результаты работы программы для различных точек с разным шагом выполнения метода

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Шаг | Точка №1 | Точка №2 | Точка №3 | Точка №4 | Точка №5 |
| h | 1,01095804 | 1,01926460 | 1,024181814 | 1,022613915 | 1,012085074 |
| h/2 | 1,01235824 | 1,02396529 | 1,028418771 | 1,025818534 | 1,013045907 |
| h/5 | 1,01326211 | 1,02702964 | 1,030890809 | 1,027695162 | 1,013533937 |
| h/10 | 1,01350138 | 1,02781192 | 1,031468435 | 1,028119090 | 1,013628578 |

Можно заметить, что во всех точках результат сходится к определенному значению. Следовательно, решение, получаемое программой, сходится. Это также можно визуально оценить, в приложении В приведен пример цветовых карт для разной точности шага метода.

Аналитическое решение данной задачи получить невозможно, так что сравним результаты работы полученной программы с результатами из COMSOL Multiphysics – комплексной интегрированной средой для моделирования физических явлений и разработки приложений, в которой есть все необходимое для создания удобного пользовательского интерфейса [14]. Получим решение этой же задачи в данной среде. Полученную ячейку можно увидеть на рисунке 7. Отметим, что полученные результаты выглядят похоже, но необходимо сравнить значения в конкретных точках. Для этого построим графики по точкам некоторых срезов из полученных решений. Некоторые из полученных графиков приведены в приложении Г. В приложении график точек из результатов решения COMSOL Multiphysics подписан как МКЭ, так как именно этим методом моделируются задачи в данной программе. Можно заметить, что они почти совпадают, хотя имеется погрешность. Это позволяет говорить, что написанная программа, моделирующая ячейку методом конечных разностей, дает достоверные результаты.

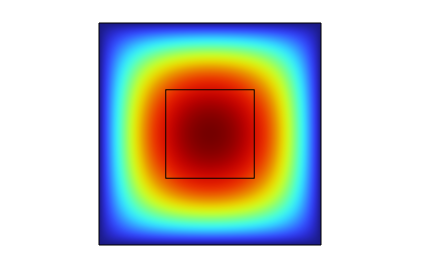


Рисунок 7 – Ячейка, полученная в COMSOL Multiphysics

**3 Решение задачи для периодического акустического метаматериала**

**3.1 Теорема Флоке-Блоха**

Чтобы найти способ решения для метаматериала, составленного из множества периодических ячеек, как на рисунке 8, воспользуемся теоремой Блоха (для одномерного случая теорему называют теоремой Флоке [15]). Задача имеет свойство периодичности, которое можно записать как:

(27)

где

(28)

где

– период;

– константа.

Тогда решение задачи колебаний в периодической среде имеет вид:

*,* (29)

где

– решение в одной ячейке;

– волновой вектор, = {*k1, k2*};

*i* – мнимая единица.

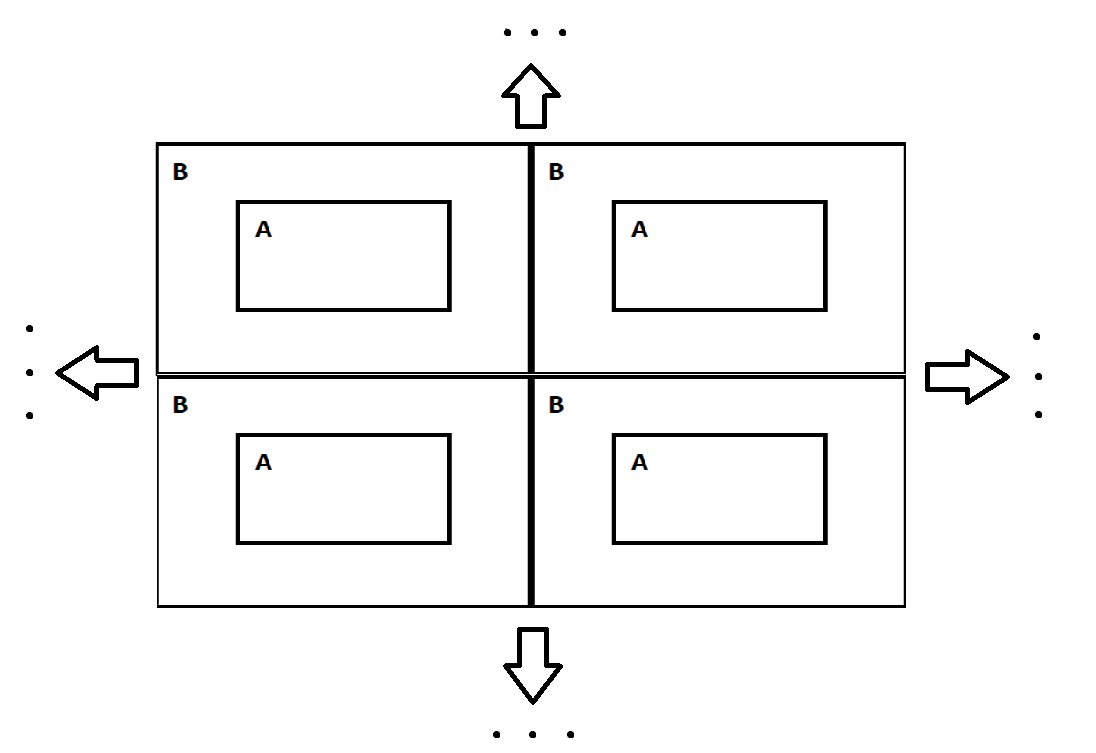


Рисунок 8 – Множество ячеек метаматериала

**3.2 Сведение к задаче на собственные значения**

В программе для решения задачи в одной ячейке граничные условия принимали значение 1, теперь вместо этого, будут учитываться равенства:

, (30)

, (31)

, (32)

(33)

где

– значения в узлах сетки,

i = 1,…, N для ;

j = 1,…, M для ;

размер одной ячейки по X;

размер одной ячейки по Y;

Вектор показывает направление звуковой волны, задается параметром α, как на рисунке 9, причем:

, (34)

где

α – угол, показывающий направление вектора.

Попробуем привести задачу к виду, похожей на задачу на собственные значения.

Вектор b в данной задачи будет нулевым, так за граничные условия принимаем формулы (30) и (31). Тогда система будет иметь вид:

, (35)

где

– матрица системы;

– вектор решений.

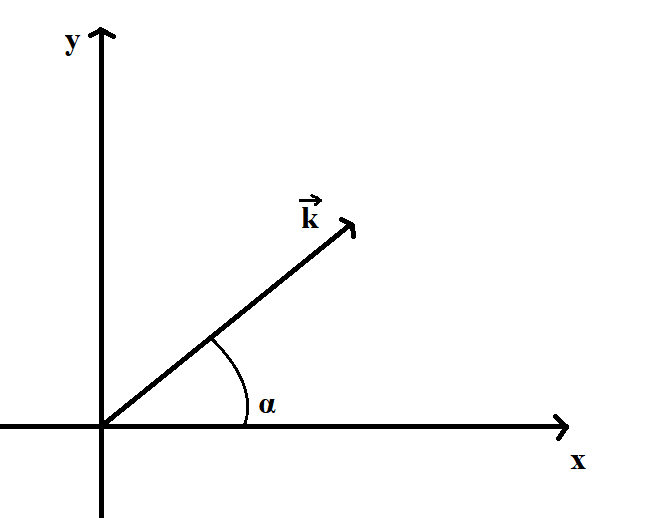


Рисунок 9 – Вектор

Если принять значение α за ноль и , то в таком случае будет верно:

, (36)

. (37)

Разобьем матрицу системы на сумму нескольких матриц, одна из которых будет учитывать новые граничные условия (30), а другая – внутренние точки и часть граничных точек:

(38)

где

матрица с коэффициентами конечно-разностных уравнений, как в задаче для одной ячейки, причем в ней также учитываются уравнения для граничных точек и ;

– матрица, в которой учитываются коэффициенты уравнений (29) и (31).

Возьмем пока *m=n*. Отметим, что в данных двух матрицах будут нулевые строки, то есть они будут принимать вид:

, (39)

, (40)

где

aml – некие константы конечно-разностных уравнений, m = 1,…, , l = 1,…, n2.

Матрицу (40) также можно представить в виде суммы двух матриц:

, (41)

где

(42)

(43)

где 1 стоят на том же самом месте, что и в матрице (40) соответственно.

Учитывая (33) и то, что мы ищем ненулевое решение:

то есть определитель матрицы равен 0. Тогда умножим сумму первых двух матриц на ее обратную и эту же обратную матрицу на в (44):

где

единичная матрица.

Тогда, если разделить слагаемые в определителе на число , справедливо:

.

Выражение (20) похоже на задачу о собственных значениях:

где

собственное значение.

Тогда решение системы переходит к задаче на собственные значения и собственные вектора.

Таким образом, сформулирован метод решения задачи для периодической структуры при нормальном угле падения волны.

**3.3 Программная реализация**

Для поставленной задачи моделирования множества периодических ячеек нужно реализовать решающую ее программу. Для этого также будем использовать язык программирования Python, так как в нем есть возможность решать задачи на собственные значения. Будем четко разделять процесс на функции.

Поскольку есть программная реализация задачи для одной ячейки метаматериала с определенными граничными условиями, то некоторые функции в данной программе будут идентичны уже имеющимся, а какие-то получат изменения.

Остается прежней функция, проверяющая принадлежность точки к зоне материала A. Функция построения матрицы системы имеет некоторые изменения. Она будет возвращать матрицу – основная матрица с коэффициентами конечно-разностных уравнений для внутренних узлов, она также учитывает граничные условия на левой и правой границе ячейки. Для остальных точек там нулевые значения. Для данной функции также написана вспомогательная для определения узла по координате. Добавятся 2 новые функции для вычисления и , учитывающие некоторые граничные условия. будет учитывает условия (30) на верхней границе ячейки, а – на нижней.

Каждая нужная матрица для (47) будет отдельно вычисляться при помощи вышеописанных функций. Для суммы матриц и необходимо получить обратную матрицу, затем умножить на нее. После этого собственные значения можно вычислить встроенной функцией. После этого можно отбросить нулевые решения и вычислить волновое число k.

Текст программы приведен в приложении Д. Примеры решения представлены в приложении Е.

**4 Разработка приложения**

**4.1 Варианты использования и функциональные требования**

Разработаем специальный программный продукт, автоматизирующий процесс моделирования ячейки метаматериала с граничными условиями (5) из 1 главы. Проектирование программного обеспечения представляет собой процесс создания спецификаций ПО на основе исходных требований к нему [16]. Сначала обозначим 4 специальные задачи, которое разрабатываемое программное средство будет решать:

1) Моделирование ячейки метаматериала с разными входными параметрами. Назначение задачи – обеспечить корректную работу программы с разными входными параметрами, такими как шаг метода моделирования, плотность материала и модуль сдвига, размер ячейки.

2) Получение результатов. Назначение задачи – организация вывода результатов работы программы в виде файла.

3) Визуализация результатов. Назначение задачи – построение по результатам работы программы визуальной модели распространения звука в ячейки.

4) Сравнение двух матриц. Назначение задачи – сделать удобный функционал сравнения двух матриц.

Эти задачи также отражают варианты использования будущего приложения. Вариант использования описывает поведение системы при ее ответах на запрос одного из участников, называемого основным действующим лицом, в различных условиях [17]. К вариантам использования можно привязать все функциональные требования и многие нефункциональные требования [18]. Разработка требований к программному обеспечению — это ключевой этап в процессе создания любого программного продукта. Требования определяют, что система должна делать и как она должна это делать. Функциональные требования представляют собой детальное описание поведения и сервисов программного продукта, его функционала [19]. Они определяют конкретные функции, которые система должна выполнять для удовлетворения потребностей пользователей. Эти требования включают в себя описание процессов, данных и взаимодействий между различными компонентами системы.

Обозначим тогда следующие 4 функциональных требования:

1) Для ввода должны быть доступны все необходимые для моделирования параметры. Программа должна проверять корректность введенной информации, в том числе на значение ограничений некоторых параметров. Для удобства использования можно добавить также случайную генерацию параметров.

2) Согласно полученным входными данным программа должна произвести вычисления численным методом конечных разностей. Получаемую матрицу значений можно записать в файл. Каждая строка файла – это одна строка матрицы, в строках значения разделены пробелом. Название файла пользователь может выбрать сам. Также должна быть возможность сохранить решения для периодической системы ячеек.

3) Результаты, представленные в двумерном массиве необходимо визуализировать в виде цветовой сетки. Она должна представлять собой поле, заполненное цветом, который определяется заданной цветовой картой и численными значениями элементов массива. Цвета, выбранные для данной задачи, должны наглядно отражать результаты, то есть быть контрастными, например, красный-синий, желтый-фиолетовый и т. п. У пользователя должна быть возможность самому определять палитру. Сетка также должна иметь возможность выведения в отдельном окне выделенного участка.

4) В отдельном окне должна быть возможность отдельного построения двух матриц по указанным параметрам. Каждую из матриц можно сохранить в удобном формате и настроить их цветовую карту.

**4.2 Нефункциональные требования**

Нефункциональные требования — это ограничения или требования, предъявляемые к системе. Они включают в себя характеристики, которые влияют на производительность, надежность, практичность и возможность обслуживания [20]. Эти требования часто называют качественными атрибутами системы. Нефункциональные требования помогают определить, насколько хорошо система будет выполнять свои функции и удовлетворять потребности пользователей.

Укажем 5 нефункциональных требования к разрабатываемой программе.

1) Наглядность интерфейса: интерфейс должен быть удобным, интуитивно понятным и продуманным с точки зрения инженерной психологии, эргономики и методов. Интерфейс не должен вызывать раздражение при длительном использовании приложения, удобно предоставлять данные пользователю.

2) Сопровождаемость: возможность проведения конкретных изменений системы. ПС должно быть спроектировано таким образом, чтобы были возможны модификации или изменения системы.

3) Безотказность: система должна сохранять работоспособность в течение требуемого интервала времени непрерывно без вынужденных перерывов. ПС должно функционировать длительное время без отказов.

4) Работоспособность: система должна выполнять свои функции с эксплуатационными показателями не ниже заданных. ПС должно учитывать все возможные значения входных параметров с указанными ограничениями и обеспечить корректность выходных данных.

5) Целостность: противостояние системы изменениям, искажениям или порче при возникновении сбоев или ошибок. ПС должно учитывать возможные ошибки и сбои, сохраняя при этом работоспособность и данные.

**4.3 Основные компоненты**

Для соблюдения некоторых нефункциональных требований код приложения имеет смысл разделить на несколько файлов. Это в свою очередь будет иметь такие преимущества, как улучшенная читаемость и упрощение разработки и поддержки.

В отдельный файл вынесем генерацию матрицы, разделим GUI (основное окно и окно сравнения), отделим такие функции, как сохранения.

Опишем ключевые компоненты приложения, каждый из которых отвечает за определенную функциональность.

Функция generate\_matrix() составляет матрицу решений системы линейных уравнений, моделирующей распределение звуковой волны в ячейке математериала. Параметрами данной функции являются все записанные в поля ввода пользователем данные. Алгоритм у данной функции состоит из следующих 3 пунктов:

1) Проверка входных параметров.

2) Построение СЛАУ методом конечных разностей.

3) Решение системы с помощью функций специальной библиотеки.

Также есть функция generate\_solve\_periodic() для решения задачи для периодического метаматериала, которая имеет аналогичный алгоритм.

Класс MatrixApp реализует графический интерфейс для управления расчетами и визуализацией. Компонентами являются 11 полей ввода параметров, инструменты (кнопки: сгенерировать, случайные значения, сохранить матрицу), визуализация (холст для отображения матрицы и интерактивное выделение областей). К основным методам можно отнести следующие 3:

1) random\_fill\_parameters() – заполнение полей случайными значениями.

2) draw\_heatmap() – создание цветовой карты матрицы.

3) save\_matrix() – сохранение матриц в файл формата txt или xlsx.

4) save\_solve\_periodic() – сохранение решения для периодической системы в файл формата txt или xlsx.

Класс CompareMatricesWindow реализует графический интерфейс для управления расчетами и визуализацией сравнения двух матриц с разными параметрами. В соответствующем окне осуществляется параллельный ввод параметров двух матриц, два окна вывода цветовых карт и отдельное сохранение каждой матрицы. Этот класс наследует слить и логику из MatrixApp и использует тот же метод расчета матрицы решений.

Данные компоненты соответствуют заданным функциональным и нефункциональным требованиям. Основное окно программного средства можно увидеть в приложении Ж. 1, построенную матрицу в основном окне в приложении Ж. 2. Выбор фрагмента и его увеличенную версия показаны на скриншотах из приложения Ж. 3 и Ж. 4. Изменённая палитра для цветовой карты продемонстрирована приложении Ж. 5. Окно сравнения матриц в приложении Ж. 6 и Ж. 7. Разработанное программное средство будет работать на подавляющем числе популярных операционных систем, имеет минимальные требования для полноценной работы, удобно и просто в использовании.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения работы была достигнута поставленная цель – изучение и применение метода конечных разностей для моделирования периодических акустических метаматериалов. Работа позволила углубить понимание численных методов и их практического применения в области моделирования.

Были рассмотрены основы метода конечных разностей, его достоинства и недостатки, а также особенности для применения к данной задаче. Была разработана программа на языке Python, позволяющая моделировать распространение звуковой волны в ячейке метаматериала. На основе теоремы Флоке-Блоха был предложен метод решения задачи для множества периодических ячеек. Было создано программное средство с графическим интерфейсом, автоматизирующее процесс решения задачи. Оно соответствует заданным функциональным и нефункциональным требованиям.

Выполненная работа демонстрирует эффективность численных методов для решения задач моделирования в различных областях. Разработанные алгоритмы и программное средство могут служить основой для дальнейших научных и прикладных исследований в этой области.

**СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ**

1. Фононные кристаллы: сайт / Институт математики, механики и информатики. – URL: http://immi.kubsu.ru/ru/research/phononic-crystals.php (дата обращения: 1.05.2025).

2. Мещеринов В.В. Метаматериал для альтернативной магнонной электроники / В.В. Мещеринов // Журнал «За науку». – 2019. – URL: mihttps://zanauku.mipt.ru/2019/08/13/metamaterial-dlya-alternativnoj-magnonnoj-elektroniki/pt.ru (дата обращения: 29.04.2025).

3. Слюсар В.И. Метаматериалы в конструкциях антенн / В.И. Слюсар // Электроника: наука, технология, бизнес. – 2010. – № 30-4. – С. 44-60.

4. Слюсар В.И. Метаматериалы в антенной технике: история и основные принципы / В.И. Слюсар // Электроника: наука, технология, бизнес. – 2009. – № 7. – С. 70–79.

5. Deymier, P. A. Acoustic Metamaterials and Phononic Crystals / P. A. Deymier – Dresden, Germany: Editor, 2013. – 378 p. – ISBN 978-3-642-31231-1.

6. Transmission and band gaps of elastic SH waves in functionally graded periodic laminates / M.V. Golub, S.I. Fomenko, T.Q. Bui [и др.]. – International Journal of Solids and Structures. – 2012. – Vol. 49, P. 344-354.

7. Ашкрофт Н., Физика твердого тела. Т.1. / Н. Ашкрофт, Н. Мермин. – Москва: Мир, 1979. – 399 с.

8. Самарский, А. А. Методы решения сеточных уравнений: учебное пособие / А. А. Самарский, Е. С. Николаев. – Москва: Наука, 1978. – 592 с.

9. Метод конечных разностей: лекция – URL: https://portal.tpu.ru/SHARED/a/ALEX1479/study/Matmod/Tab1/Tab/11%20%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%20%D0%9C%D0%B0%D1%82\_%D0%BC%D0%BE%D0%B4\_%D1%84%D0%B8%D0%B7\_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D0%B2.pdf (дата обращения: 1.05.2025).

10. Дегтярев, А. А. Метод конечных разностей: учебное пособие / А. А. Дегтярев. – Самара, 2011. – 83 с. – URL: http://repo.ssau.ru/bitstream/Uchebnye-posobiya/Metod-konechnyh-raznostei-Elektronnyi-resurs-elektron-ucheb-posobie-54144/1/%D0%94%D0%B5%D0%B3%D1%82%D1%8F%D1%80%D0%B5%D0%B2%20%D0%90.%D0%90.%20%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%20%D0%BA%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D1%8B%D1%85.pdf (дата обращения: 7.04.2025).

11. Метод конечных разностей: сайт – URL: https://stroitmeh.ru/lect83.htm (дата обращения: 2.05.2025).

12. Преимущества языка Python: сайт – URL: https://www.hocktraining.com/blog/preimuschestva-yazyka-python#steer9 (дата обращения: 2.05.2025).

13. Seaborn для визуализации данных в Python: сайт – URL: https://pythonru.com/biblioteki/seaborn-plot (дата обращения: 2.05.2025).

14. Введение в COMSOL Multiphysics: руководство – URL: <https://cdn.comsol.com/doc/5.4/IntroductionToCOMSOLMultiphysics.ru_RU.pdf> (дата обращения: 29.04.2025).

15. Теорема Блоха: сайт– URL: https://dic.academic.ru/dic.nsf/ruwiki/1149

946 (дата обращения: 17.04.2025).

16. Черушева Т.В. Проектирование программного обеспечения: учеб. пособие / Т. В. Черушева. – Пенза: Изд-во ПГУ, 2014. – 172 с. – ISBN 978-5-94170-859-8.

17. Коберн А. Современные методы описания функциональных требований к системам / А. Коберн – Москва: Лори, 2002. – 263 c. – ISBN 5-85582-152-8.

18. Якобсон А. Унифицированный процесс разработки программного обеспечения / А Якобсон, Г. Буч, Дж. Рамбо. – Санкт-Петербург: Питер Принт, 2002. – 495 с. ISBN 5-318-00358-3.

19. Полетайкин А.Н. Социальные и экономические информационные системы: законы функционирования и принципы построения: учебное пособие / А.Н. Полетайкин – Новосибирск: Редакционно-издательский отдел СибГУТИ, 2016. – 241 с.

20. Леффингуэлл Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход / Д. Леффингуэлл, Д. Уидриг – Москва: Вильямс: 2002, 448 c. – ISBN 5-8459-0275-4.

**ПРИЛОЖЕНИЕ А**

**Текст программы для моделирования одной ячейки метаматериала**

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

def dA(x, y, x1, x2, y1, y2): #Функия, определяющая, находится ли точка в зоне материала A

return (x1<=x<=x2 and y1<=y<=y2)

def plot\_matrix(matrix):

plt.figure(figsize=(8, 6))

sns.heatmap(matrix, annot=False, cmap='viridis')

plt.title("Цветовая карта матрицы значений")

plt.show()

def generate\_matrix(params):

x1 = params[0]

x2 = params[1]

y1 = params[2]

y2 = params[3]

xn = params[4]

hx = params[5]

ma = params[6]

mb = params[7]

po\_a = params[8]

po\_b = params[9]

w = params[10]

n = int(xn / hx)

n += 1

A = np.zeros((n \*\* 2, n \*\* 2))

B = np.zeros((n \*\* 2))

hx2 = 1 / hx \*\* 2 # Переменные, облегчающие вычисление конечных разностей

count = 0 # Счетчик, по которому заполняется матрица СЛАУ

i = 0

while i <= xn: # Составление СЛАУ

j = 0

while j <= xn:

if dA(i, j, x1, x2, y1, y2) == True: # Задаем плотность и модуль сдвига, которые будут использоваться в данной итерации, в зависимости от местонахождения узла

mu = ma

po = po\_a

else:

mu = mb

po = po\_b

if i == 0 or i == xn:

B[count] = 1

A[count][count] = 1

elif i == x1 or i == x2:

if j == 0 or j == xn:

B[count] = 1

A[count][count] = 1

elif j >= y1 and j <= y2:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2)

A[count][count - n] = mu \* (hx2)

A[count][count + 1] = mu \* (hx2) + ma / (2 \* hx) - mb / (2 \* hx)

A[count][count - 1] = mu \* (hx2) - ma / (2 \* hx) + mb / (2 \* hx)

else:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2)

A[count][count - n] = mu \* (hx2)

A[count][count + 1] = mu \* (hx2)

A[count][count - 1] = mu \* (hx2)

elif i > x1 and i < x2:

if j == 0 or j == xn:

B[count] = 1

A[count][count] = 1

elif j == y1 or j == y2:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2) + ma / (2 \* hx) - mb / (2 \* hx)

A[count][count - n] = mu \* (hx2) - ma / (2 \* hx) + mb / (2 \* hx)

A[count][count + 1] = mu \* (hx2)

A[count][count - 1] = mu \* (hx2)

else:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2)

A[count][count - n] = mu \* (hx2)

A[count][count + 1] = mu \* (hx2)

A[count][count - 1] = mu \* (hx2)

else:

if j == 0 or j == xn:

B[count] = 1

A[count][count] = 1

elif j != xn:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2)

A[count][count - n] = mu \* (hx2)

A[count][count + 1] = mu \* (hx2)

A[count][count - 1] = mu \* (hx2)

count += 1

j += hx

i += hx

U = np.linalg.solve(A, B)

U1 = np.zeros((n, n))

count = 0

for i in range (0, n):

for j in range(0, n):

U1[i][j] = U[count]

count+=1

return U1

matrix = generate\_matrix([3, 7, 3, 7, 10, 0.5, 5\*10\*\*9, 6\*10\*\*9, 2000, 4500, 10])

plot\_matrix(matrix)

**ПРИЛОЖЕНИЕ Б**

**Вид матрицы *A* из СЛАУ для заданных параметров**

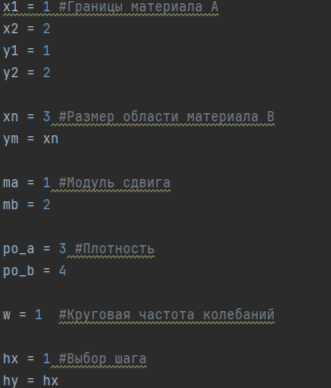


Рисунок Б. 1 – Пример входных параметров программы

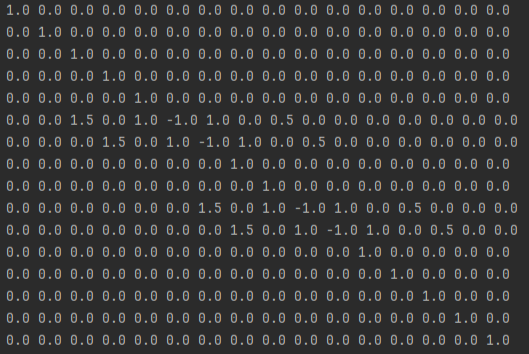


Рисунок Б. 2 – Вид матрицы *А* для входных параметров с рисунка Б.1

.

Рисунок Б. 3 – Общий вид матрицы *А* для решения системы n на m

**ПРИЛОЖЕНИЕ В**

**Цветовые карты матрицы решения для заданных параметров при различных значениях h**

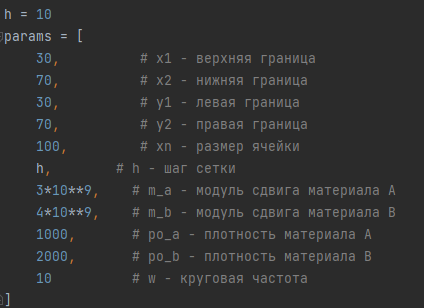
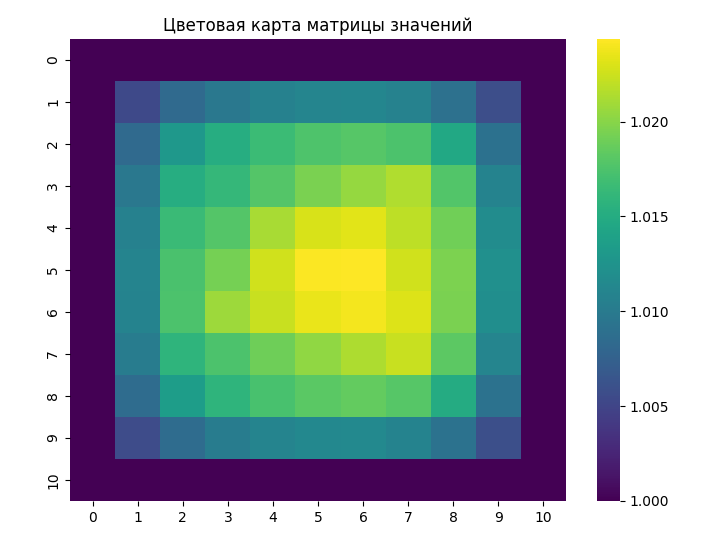
****

Рисунок В. 1 – Параметры моделирования

****Рисунок В. 2 – Значения *u* при шаге h = 10

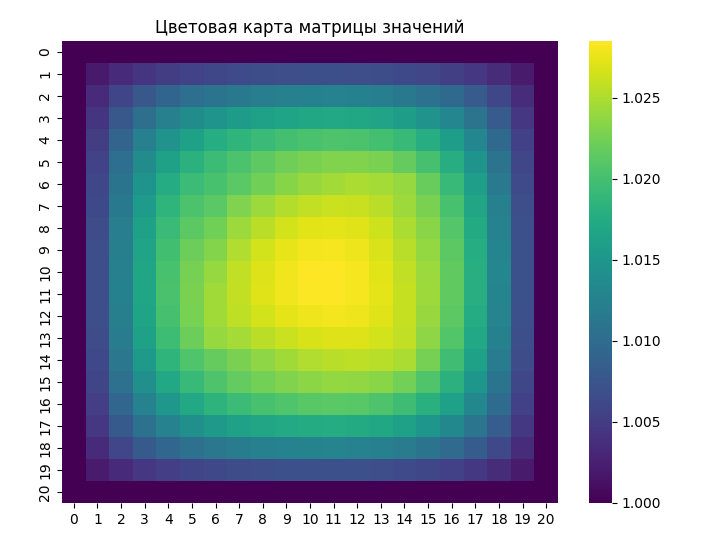


Рисунок В. 3 – Значения *u* при шаге h/2 = 5

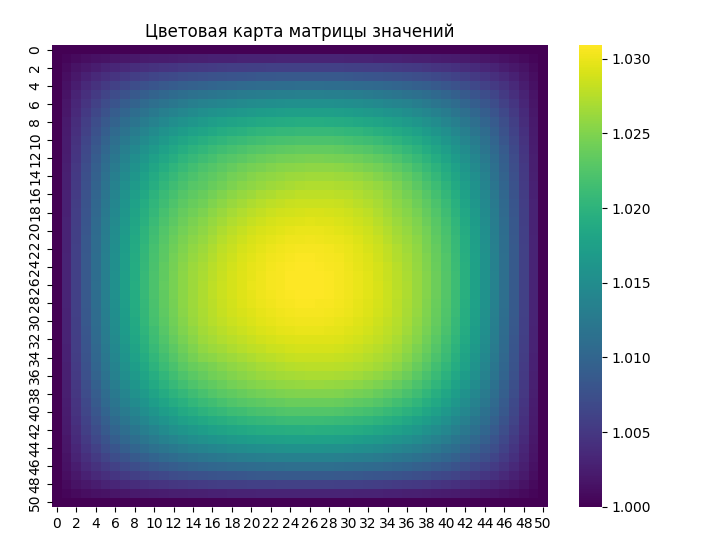


Рисунок В. 4 – Значения *u* при шаге h/5 = 2



Рисунок В. 5 – Значения *u* при шаге h/10 = 1

**ПРИЛОЖЕНИЕ Г**

**Сравнения решений МКЭ и МКР**

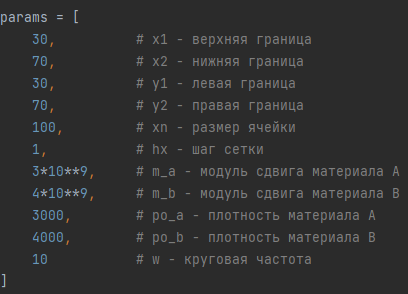
****

Рисунок Г. 1 – Параметры ячейки

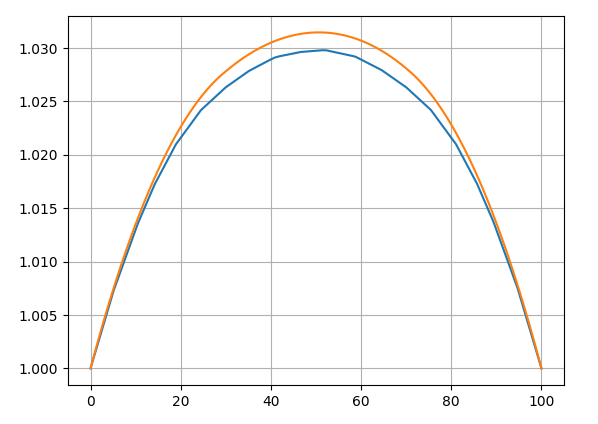
****

Рисунок Г. 2 – Сравнение результатов на срезе у = 5

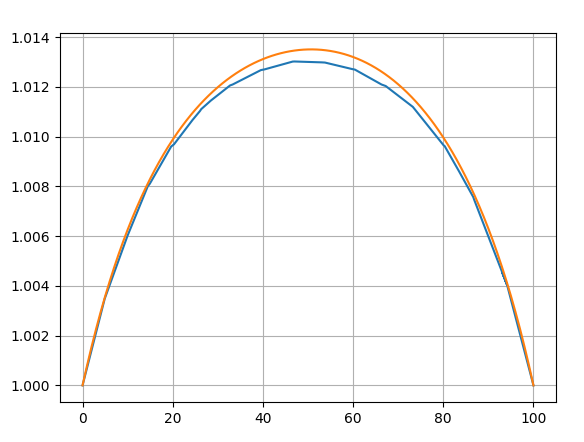
****

Рисунок Г. 2 – Сравнение результатов на срезе у = 3

**ПРИЛОЖЕНИЕ Д**

**Текст программы для моделирования системы ячеек метаматериала**

import numpy as np

def dA(x, y, x1, x2, y1, y2): #Функция, определяющая, находится ли точка в зоне материала A

return (x1<=x<=x2 and y1<=y<=y2)

def generate\_matrix\_A\_1(x, h):

xn = x

hx = h

n = int(xn / hx)

A = np.zeros(((n + 1) \*\* 2, (n + 1) \*\* 2))

n\_row = n \* (n + 1)

for j in range(n + 1):

A[n\_row + j][n\_row + j] = 1

return A

def generate\_matrix\_A\_e(x, h):

xn = x

hx = h

n = int(xn / hx)

A = np.zeros(((n + 1) \*\* 2, (n + 1) \*\* 2))

n\_row = n \* (n + 1)

for j in range(n + 1):

A[n\_row + j][j] = 1

return A

def get\_point\_index(h, coord):

if h > 1:

return int(coord / h - 1)

else:

return int(coord / h)

def generate\_matrix\_A\_inner(params):

x1 = params[0]

x2 = params[1]

y1 = params[2]

y2 = params[3]

xn = params[4]

hx = params[5]

ma = params[6]

mb = params[7]

po\_a = params[8]

po\_b = params[9]

w = params[10]

n = int(xn / hx)

n += 1

A = np.zeros((n \*\* 2, n \*\* 2))

hx2 = 1 / hx \*\* 2 # Переменные, облегчающие вычисление конечных разностей

count = 0 # Счетчик, по которому заполняется матрица СЛАУ

i = 0

while i <= xn: # Составление СЛАУ

j = 0

while j <= xn:

if dA(i, j, x1, x2, y1, y2) == True: # Задаем плотность и модуль сдвига, которые будут использоваться в данной итерации, в зависимости от местонахождения узла

mu = ma

po = po\_a

else:

mu = mb

po = po\_b

if i == 0:

A[count][count] = mu \* 2 \* hx2 + po \* w \*\* 2

A[count][count + n] = -2 \* mu \* (hx2)

A[count][count + 2\*n] = mu \* (hx2)

A[count][count + 1] = -2 \* mu \* (hx2)

A[count][count + 2] = mu \* (hx2)

elif i == x1 or i == x2:

if j == 0:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2)

A[count][count - n] = mu \* (hx2)

A[count][count + 1] = mu \* (hx2)

A[count][count - 1] = mu \* (hx2)

elif j >= y1 and j <= y2:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2)

A[count][count - n] = mu \* (hx2)

A[count][count + 1] = mu \* (hx2) + ma / (2 \* hx) - mb / (2 \* hx)

A[count][count - 1] = mu \* (hx2) - ma / (2 \* hx) + mb / (2 \* hx)

elif j != xn:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2)

A[count][count - n] = mu \* (hx2)

A[count][count + 1] = mu \* (hx2)

A[count][count - 1] = mu \* (hx2)

else:

A[count][count] = 1

A[count][get\_point\_index(hx, i) \* xn + 1] = -1

elif i > x1 and i < x2:

if j == 0:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2)

A[count][count - n] = mu \* (hx2)

A[count][count + 1] = mu \* (hx2)

A[count][count - 1] = mu \* (hx2)

elif j == y1 or j == y2:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2) + ma / (2 \* hx) - mb / (2 \* hx)

A[count][count - n] = mu \* (hx2) - ma / (2 \* hx) + mb / (2 \* hx)

A[count][count + 1] = mu \* (hx2)

A[count][count - 1] = mu \* (hx2)

elif j != xn:

A[count][count] = mu \* (-2 \* hx2 - 2 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2)

A[count][count - n] = mu \* (hx2)

A[count][count + 1] = mu \* (hx2)

A[count][count - 1] = mu \* (hx2)

else:

A[count][count] = 1

A[count][get\_point\_index(hx, i) \* xn + 1] = -1

elif i != xn:

if j == 0:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2)

A[count][count - n] = mu \* (hx2)

A[count][count + 1] = mu \* (hx2)

A[count][count - 1] = mu \* (hx2)

elif j != xn:

A[count][count] = mu \* (-4 \* hx2) + po \* w \*\* 2

A[count][count + n] = mu \* (hx2)

A[count][count - n] = mu \* (hx2)

A[count][count + 1] = mu \* (hx2)

A[count][count - 1] = mu \* (hx2)

else:

A[count][count] = 1

A[count][get\_point\_index(hx, i) \* xn + 1] = -1

else:

A[count][count] = 0

count += 1

j += hx

i += hx

return A

params = [

3, # x1 - верхняя граница

7, # x2 - нижняя граница

3, # y1 - левая граница

7, # y2 - правая граница

10, # xn - размер ячейки

1, # hx - шаг сетки

3\*10\*\*9, # m\_a - модуль сдвига материала A

4\*10\*\*9, # m\_b - модуль сдвига материала B

1000, # po\_a - плотность материала A

2000, # po\_b - плотность материала B

10 # w - круговая частота

]

A\_inn = generate\_matrix\_A\_inner(params)

A\_1 = generate\_matrix\_A\_1(params[4], params[5])

A\_e = generate\_matrix\_A\_e(params[4], params[5])

A\_inv = np.linalg.inv(A\_inn+A\_1)

A\_all = A\_e.dot(A\_inv)

eigenvalues = np.linalg.eigvals(A\_all)

non\_zero\_eigenvalues = eigenvalues[eigenvalues != 0]

print("Решение:")

for i in non\_zero\_eigenvalues:

print(np.log(i)\*complex(0, 1)/(params[4]))

**ПРИЛОЖЕНИЕ Е**

**Решение задачи для периодических ячеек**

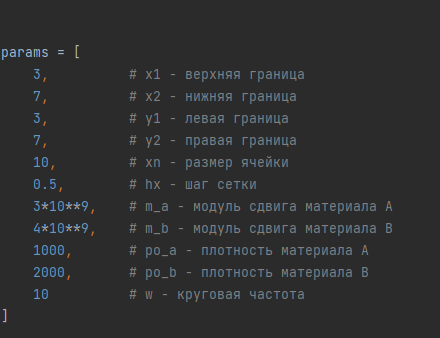


Рисунок Е. 1 – Пример входных параметров программы

Таблица Е – 1 Решение задачи для периодических ячеек для параметров с рисунка Е. 1

|  |  |
| --- | --- |
| Вещественная часть | Мнимая часть |
| 0 | 0.00034785688515283456 |
| -0.3141592653589793 | -0.5981927359062545 |
| -0.172246875044629 | -0.7699704608823967 |
| 0.172246875044629 | -0.7699704608823967 |
| -0.10502596384361598 | -1.2013314416271896 |
| 0.10502596384361598 | -1.2013314416271896 |
| -0.15928456726304496 | -1.4912338138937788 |
| 0.15928456726304496 | -1.4912338138937788 |
| -0.25687239062021544 | -1.6405755857891013 |
| 0.25687239062021544 | -1.6405755857891013 |

**ПРИЛОЖЕНИЕ Ж  
Приложение для моделирования метаматериала**

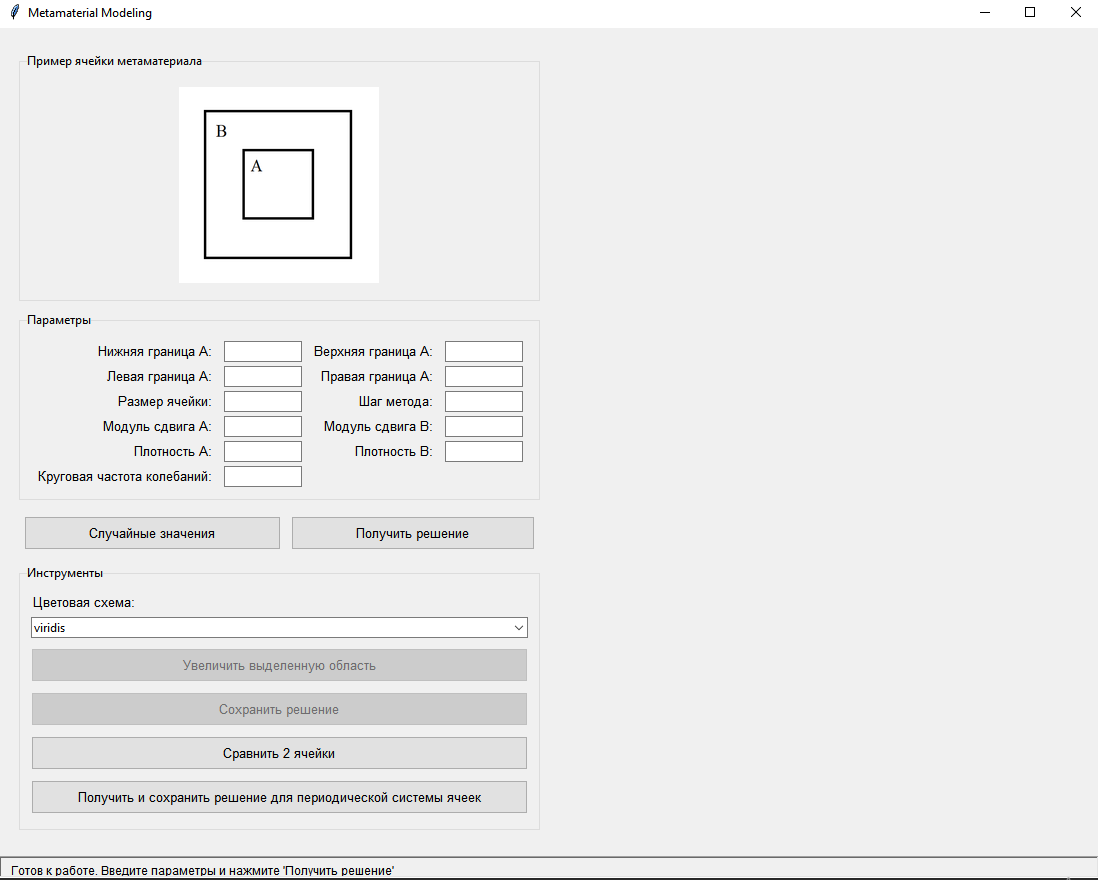


Рисунок Ж. 1 – Основное окно программного средства

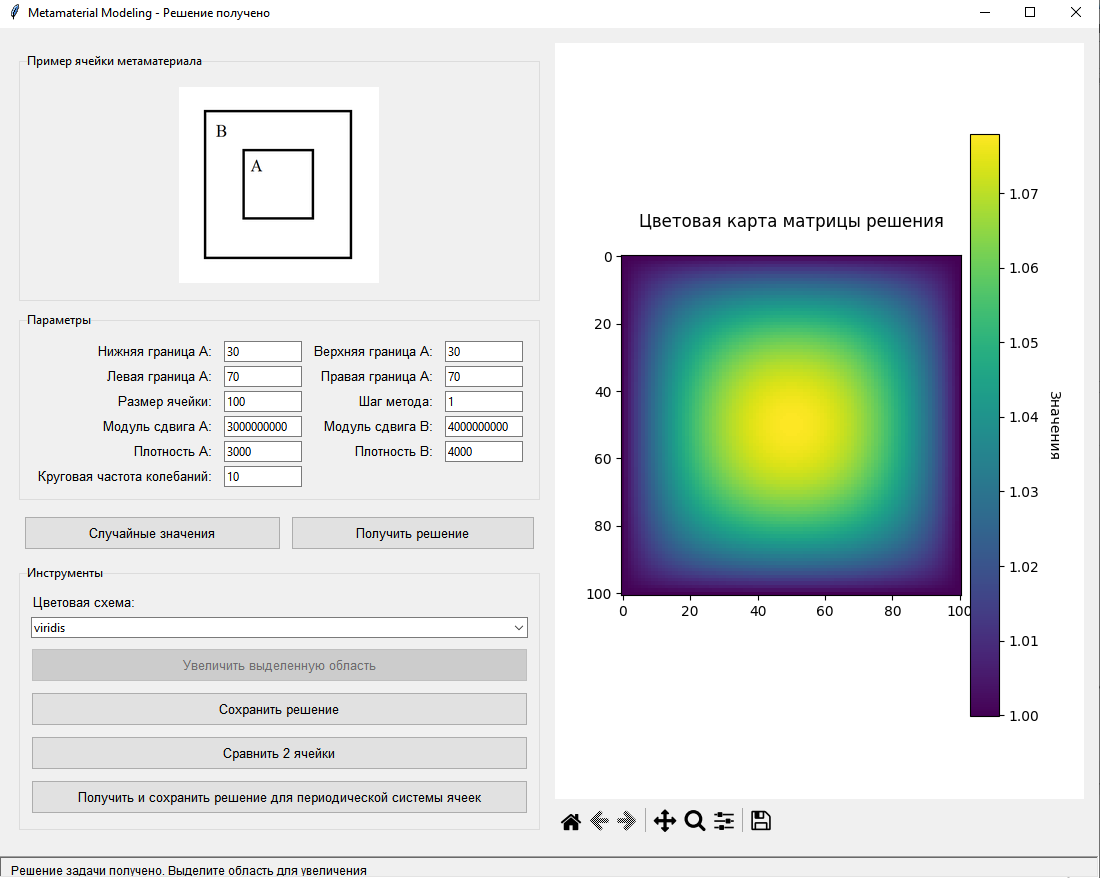


Рисунок Ж. 2 – Построенная цветовая карта

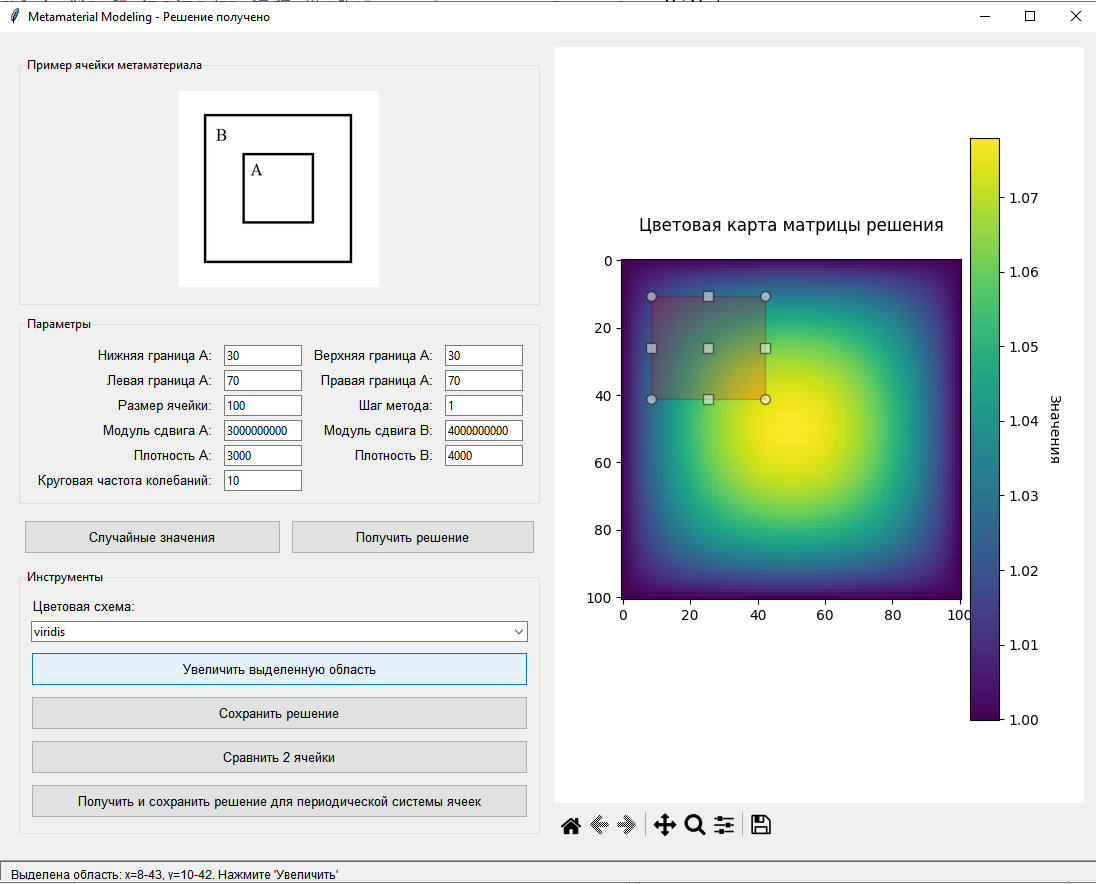


Рисунок Ж. 3 – Выделение области цветовой карты

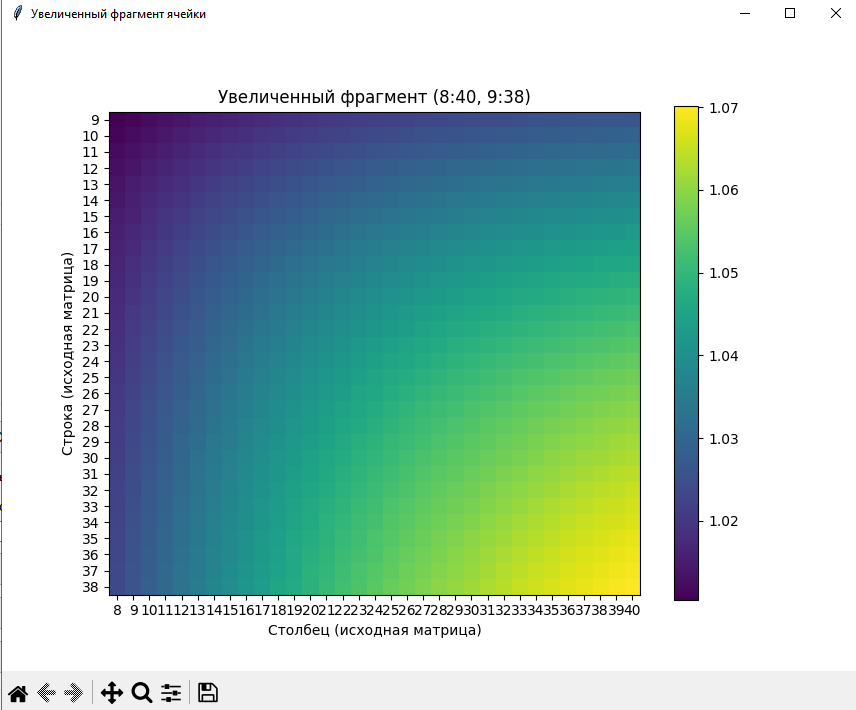


Рисунок Ж. 4 – Увеличенный фрагмент цветовой карты

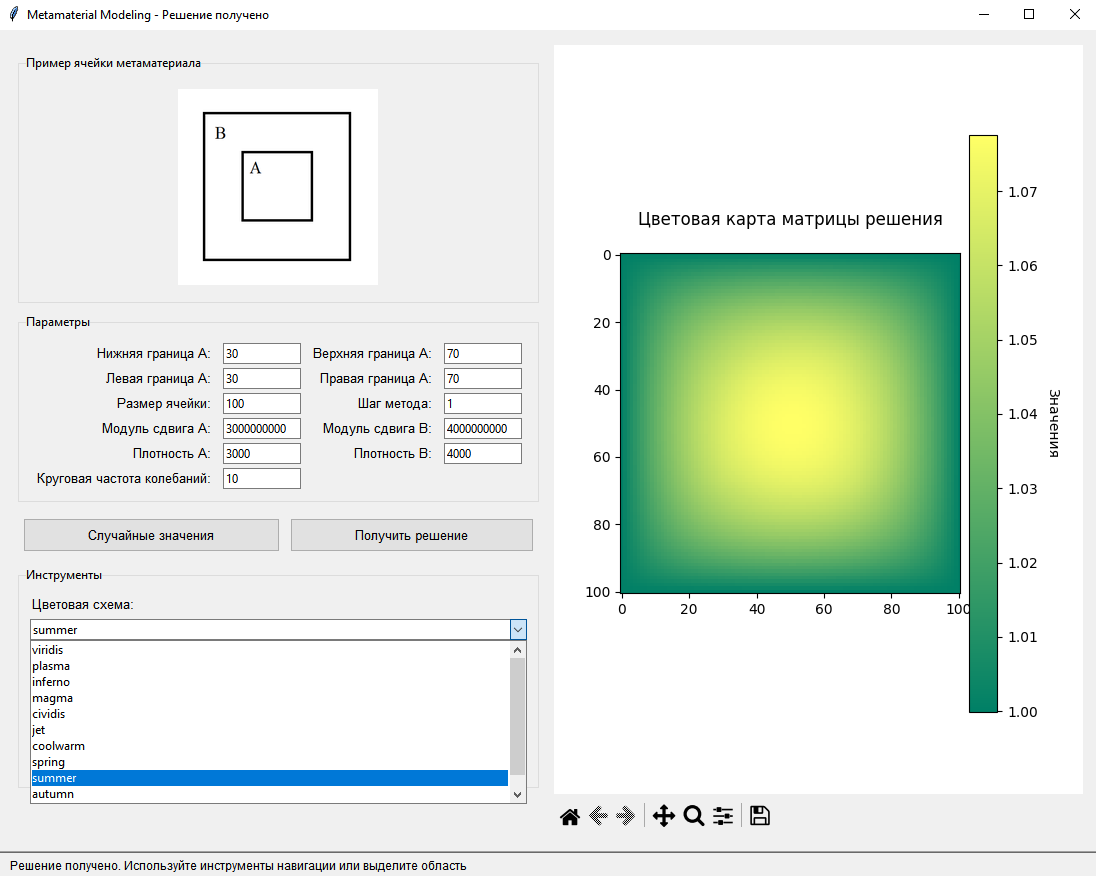


Рисунок Ж. 5 – Изменение палитры цветовой карты

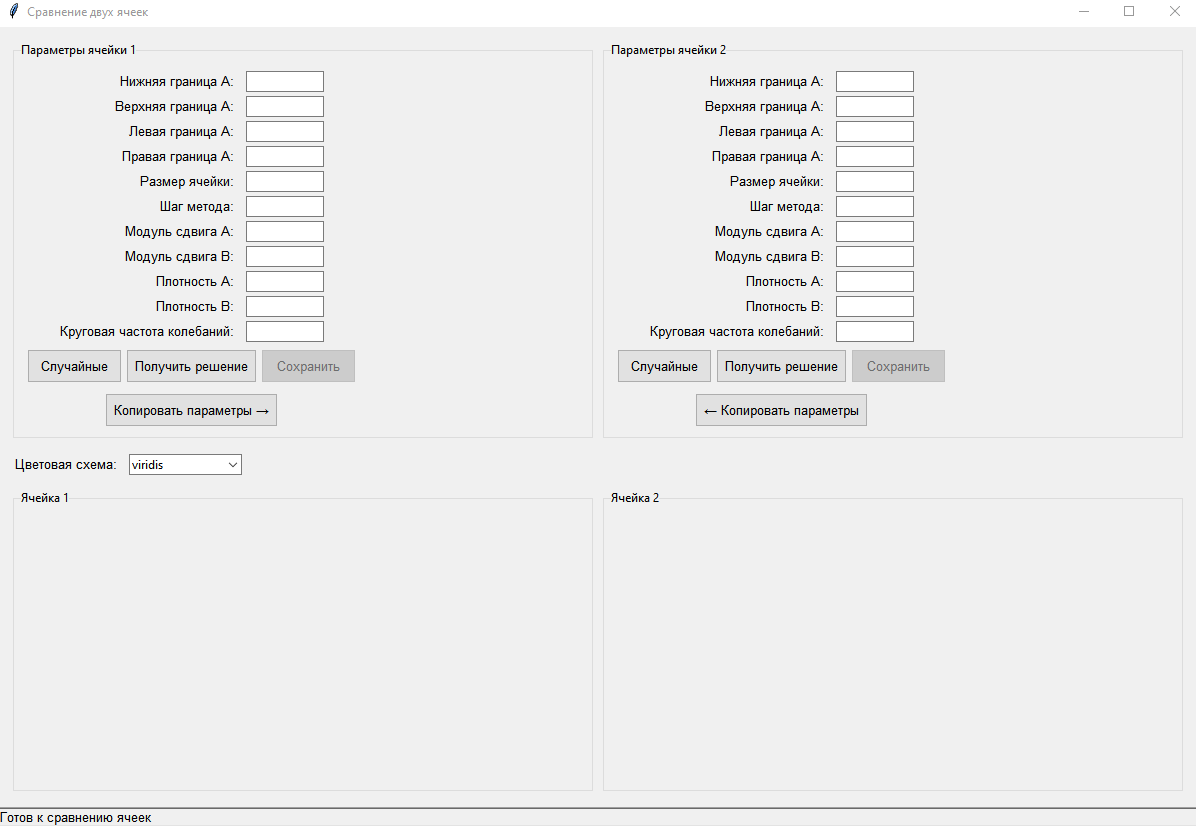


Рисунок Ж. 6 – Окно сравнения матриц

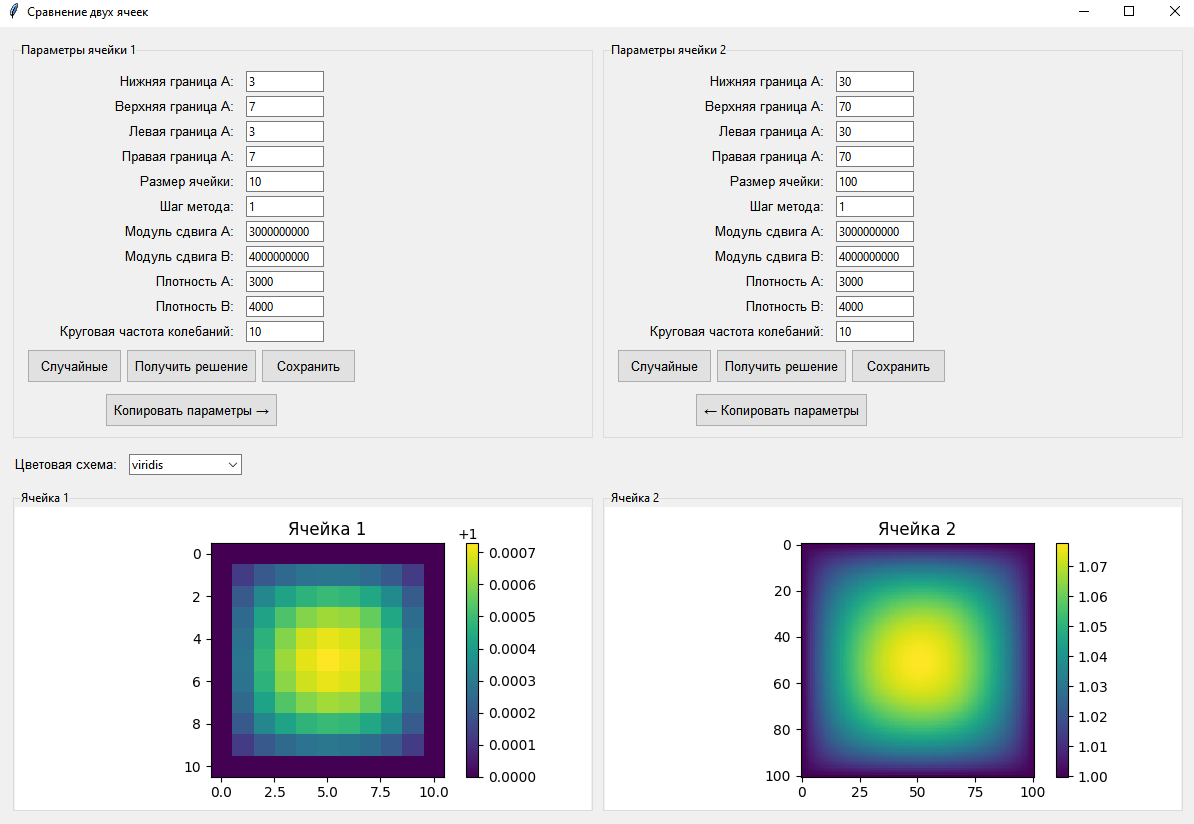


Рисунок Ж. 7 – Построение двух матриц

