ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ ИМЕНИ ПАТРИСА ЛУМУМБЫ»

На правах рукописи

Штепа Кристина Александровна

Моделирование трансформирующих сред средствами лучевой оптики

Специальность 1.2.2. Математическое моделирование, численные методы и комплексы программ

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель
д.ф.-м.н., профессор
Д. С. Кулябов

Москва — 2025

Оглавление

Введение **3**

1. Трансформирующие среды **9**
1.1. Методы моделирования оптических трансформирующих сред . . 9
1.2. Линзы Люнеберга, Максвелла и Итона 17
1.3. Численные методы на основе методов нейронных сетей 22

2. Моделирование оптических трансформирующих сред на основе лучевой оптики **43**
2.1. Формализм оптических лучей и уравнения эйконала 43
2.2. Символьные исследования уравнений Максвелла в формализме пространственно-временной алгебры 62
2.3. Численный метод FSM (Fast Sweeping Method) 72
2.4. Нейронные сети, основанные на физике 76

3. Моделирование трансформирующих оптических сред **91**
3.1. Реализация метода эйконала на основе численного метода FSM . 91
3.2. Реализация метода эйконала на основе численного метода PINN 97
3.3. Уравнение эйконала 97
3.4. Сравнение методик FSM и PINN 100

Заключение **101**

Список литературы **103**

A. Аннотированный код реализации численного метода FSM **110**
A.1. Реализация метода FSM 110
A.2. Решение уравнения эйконала с помощью модуля Eikonal 114

B. Аннотированный код реализации численного метода PINN **119**

C. Сведения из векторного анализа **121**
C.1. Основные определения 121
C.2. Векторно-дифференциальные выражения второго порядка . . . 123

Список иллюстраций **124**

Введение

Актуальность темы исследования

Актуальность исследования заключается в отсутствии единого универсального подхода к расчету оптических систем, таких как линзы и дифракционные решетки. В современных условиях каждая специфичная задача требует индивидуального подхода, что значительно усложняет моделирование и оптимизацию оптических устройств. Это особенно важно в контексте развития технологий, где необходимы высокоточные расчеты для создания новых оптических элементов и улучшения их характеристик.

Дифракционные решетки – ключевые компоненты спектральных приборов, лазерных систем и сенсоров, однако для их проектирования до сих пор не существует универсального подхода, объединяющего точность электродинамических моделей и инженерную простоту лучевой оптики.

Одновременно быстрый прогресс в области машинного обучения, особенно Physics-Informed Neural Networks (PINN), открывает возможность прямого включения физических законов в процесс оптимизации.

В сочетании с классическими методами геометрической (лучевой) оптики это обещает создать гибкий и вычислительно легкий инструментарий для моделирования трансформирующих сред нового поколения.

Текущие решения зачастую ограничены по области применения или требуют значительных вычислительных ресурсов. Исследование новых подходов, таких как нейросетевые методы, в сочетании с традиционными численными методами, может дать возможность преодолеть эти ограничения и предложить более универсальные и гибкие инструменты для решения задач оптики и фотоники.

Таким образом, работа направлена на решение актуальной научной проблемы – поиск эффективных методов расчета и моделирования дифракционных систем, что имеет значительное значение для фундаментальных и прикладных исследований в области оптики.

Работа посвящена разработке и комплексному анализу моделей распространения лучей с целью оптимизации расчета характеристик трансформирующих сред на основе методов лучевой оптики и нейронных сетей.

Степень разработанности темы исследования

В настоящее время тема моделирования трансформирующих сред на основе лучевой оптики находится на стадии активного развития. Традиционные аналитические методы детально описаны в работах Борна, Тихонравова и др., но их применение для решеток с большими периодами или в трансформирующих средах остается громоздким. Существующие исследования предлагают разнообразные подходы, включая численные методы и классические решения дифференциальных уравнений, однако универсального подхода, учитывающего все особенности, пока не сформировано. Недавние исследования активно внедряют нейронные сети, такие как нейронные сети, основанные на физике (Physics-informed neural networks, PINN), что открывает новые горизонты в точности и скорости моделирования. Первые попытки интеграции нейронных сетей в задачу расчета трансформирующих сред появились лишь в 2019–2024г. Однако, несмотря на значительный прогресс, остаются нерешенными вопросы комплексного учета трансформирующих сред и оптимизации геометрии решеток.

Таким образом, степень разработанности темы можно охарактеризовать как промежуточную: уже есть серьезные наработки, но остается широкое поле для дальнейших исследований и внедрения новых подходов.

Цели и задачи

Целью является разработка гибридного мультимодельного вычислительного подхода к моделированию трансформирующих сред, объединяющего лучевую оптику и методы машинного обучения, и экспериментальное подтверждение его эффективности.

Для достижения этой цели в работе решаются следующие задачи:

— Систематизация существующих численных и аналитических методов расчета трансформирующих сред, выделение ограничения лучевого приближения;

— Моделирование дифракции на основе численных методов решения дифференциальных уравнений FSM (Fast sweeping method);

— Разработка и применение нейронной сети, основанной на физике (Physics-informed neural networks, PINN), с использованием библиотеки NeuralPDE.jl для решения задачи моделирования дифракции;

— Сравнение точности, скорости и сложности реализации обоих подходов;

— Проведение серии численных экспериментов с различными параметрами трансформирующих сред.

Научная новизна

Научная новизна состоит в разработке и реализации мультимодельного подхода к моделированию дифракционных систем в трансформирующих средах на основе уравнения эйконала, сочетающего численный метод быстрого «подметания» (Fast Sweeping Method, FSM) и физически информированные нейронные сети (PINN) в инфраструктуре SciML/NeuralPDE.jl. Получены следующие результаты:

1. Разработаны и программно реализованы две взаимодополняющие вычислительные схемы решения уравнения эйконала для трансформирующих сред (FSM и PINN) с единым контуром визуализации фронтов и лучей на тестовых профилях (линзы Люнеберга, Максвелла).

2. Выполнена адаптация постановки PINN к задачам геометрической оптики с неоднородным показателем преломления, включая практические решения ограничений Symbolics/NeuralPDE для задания кусочно-непрерывных профилей.

3. Проведён сопоставительный анализ свойств FSM и PINN применительно к задачам моделирования трансформирующих сред (требования к зависимостям и инфраструктуре, устойчивость постановки, удобство задания граничных условий и профилей среды, визуализация), выявлены области предпочтимости каждого подхода.

4. Показана воспроизводимость расчётов и визуализаций в открытой среде Julia/SciML с возможностью переноса на типовые задачи оптического проектирования.

Теоретическая и практическая значимость работы

Полученные результаты могут быть полезными для производителей оптических компонентов и систем, в том числе для разработчиков радиолокационных систем, так как метод ускоряет проектирование и оптимизацию новых типов линз и трансформирующих сред, существенно сокращая цикл разработки и вывода продукции на рынок.

Кроме того, методика представляет интерес для исследовательских лабораторий и научных коллективов, так как мультимодельный подход позволяет усовершенствовать процессы проектирования оптических элементов за счет сокращения времени расчета, упрощения методов оптимизации и возможности выполнения расчетов на персональных компьютерах, что критично для научных лабораторий без доступа к ресурсам высокопроизводительных вычислительных кластеров.

Методология и методы исследования

В научно-квалификационной работе применен мультимодельный подход, объединяющий численные алгоритмы (Fast Sweeping Method, FSM) и обученные на физических уравнениях нейросетевые модели (Physics-Informed Neural Networks на базе NeuralPDE.jl).

Положения, выносимые на защиту

1. Разработан мультимодельный подход к моделированию дифракционных систем в трансформирующих средах, основанный на сочетании численного метода быстрого распространения фронта (Fast Sweeping Method, FSM) и физически информированных нейронных сетей (Physics-Informed Neural Networks, PINN) в инфраструктуре библиотеки NeuralPDE.jl.

2. Сформулирована математическая постановка задачи моделирования распространения лучей в трансформирующих средах на основе уравнения эйконала, адаптированная для использования в среде PINN с учётом физических ограничений и профиля показателя преломления.

3. Выполнена программная реализация и методика визуализации результатов моделирования в виде полей распространения и волновых фронтов для типовых оптических систем (линзы Люнеберга, Максвелла, Итона).

4. Выполнен сравнительный анализ классических численных и нейросетевых подходов (FSM и PINN), демонстрирующие их применимость к различным классам задач моделирования дифракционных систем, включая оценку вычислительных затрат, устойчивости и визуальной интерпретируемости решений.

5. Выработаны практические рекомендации по выбору и комбинированию численных и нейросетевых методов для решения задач лучевой оптики в трансформирующих средах, основанные на обобщении проведённых экспериментов.

Степень достоверности и апробация результатов

Достоверность полученных результатов обеспечивается правильностью выбранных методов и их перекрестной верификацией, а также численными экспериментами с применением численного анализа.

Основные результаты работы представлены на всероссийских и международных конференциях:

— международная научная конференция «The XXVII Saratov fall meeting 2023 (SFM’23) XXVII International School for Junior Scientists and Students on Optics, Laser Physics & Biophotonics» (г. Саратов, Саратовский государственный университет, 2023 г.);

— всероссийская конференция с международным участием «Информационно-телекоммуникационные технологии и математическое моделирование высокотехнологичных систем» (г. Москва, РУДН, 2023–2024 г.).

Основные результаты опубликованы в ведущих научных журналах: Programming and Computer Software, Discrete and Continuous Models and Applied Computational Science, Программирование, а также в трудах всероссийской конференции с международным участием.

Также основные результаты докладывались на научном семинаре «Математическое моделирование» кафедры прикладной информатики и теории вероятностей РУДН.

Основные результаты изложены в 3 работах, в том числе в изданиях, входящих в базу данных Scopus, Web of Science, список ВАК категорий К-1, К-2 и в 3 свидетельствах о государственной регистрации программ для ЭВМ.

Обозначения и соглашения

Основным математическим аппаратом, используемым в работе, является аппарат векторного анализа и тензорного анализа. Краткие сведения из векторного анализа представлены в приложении С).

Греческие индексы (α, β) будут относиться к четырёхмерному пространству и в компонентном виде будут иметь следующие значения: $\alpha = \overline{0,3}$. Латинские индексы из середины алфавита (i, j, k) будут относиться к трёхмерному пространству и в компонентном виде будут иметь следующие значения: $i = \overline{1,3}$.

Для записи уравнений электродинамики в работе преимущественно используется система СГС симметричная [74].

Все векторные величины выделены полужирным шрифтом, например, радиус вектор точки X обозначается как $\mathbf{x} = (x^1, x^2, x^3)^T = (x, y, z)^T$. Все векторы считаются столбцами и их компоненты номеруются верхними индексами.

Функция эйконала обозначена как $u(\mathbf{x})$, коэффициент преломления среды — $n(\mathbf{x})$.

Глава 1. Трансформирующие среды

1.1. Методы моделирования оптических трансформирующих сред

Оптические трансформирующие среды — это среды, изменяющие пространственное распределение направления и плотности потока света. К ним относятся призмы, зеркальные и градиентные структуры, волноводные переходы, а также композиционные материалы с пространственно-неоднородным показателем преломления. Такие среды применяются для управления формой волнового фронта, концентрации энергии и перенаправления лучей в заданные области пространства.

В основе их теоретического описания лежит геометрическая оптика — приближение волновой теории света, применимое в случае, когда характерные размеры неоднородностей и элементов системы значительно превышают длину волны излучения.

Моделирование оптических трансформирующих сред в приближении геометрической оптики основано на представлении света как совокупности лучей, которые распространяются в пространстве по законам преломления и отражения, подчиняясь принципу Ферма. Этот подход особенно эффективен для систем, размеры которых значительно превышают длину волны излучения, а пространственные изменения показателя преломления происходят плавно.

Геометрическая оптика рассматривает свет как поток энергии, движущийся вдоль направлений, перпендикулярных волновым фронтам — поверхностям, все точки которых колеблются в одинаковой фазе. В этом приближении игнорируется волновая природа света, а параметры электромагнитного поля заменяются геометрическими характеристиками — направлением и положением лучей, плотностью потока, а также оптической длиной пути. Таким образом, геометрическая оптика обеспечивает простую и интуитивную модель распространения света, позволяющую описывать фокусировку, коллимацию, расхождение и преобразование пучков без учёта интерференционных и дифракционных эффектов.

Основу геометрической оптики составляет принцип Ферма (1662) — постулат, согласно которому свет выбирает путь, для прохождения которого

требуется минимальное время. Величина времени распространения света между двумя точками определяется интегралом оптической длины пути:

$$T = \frac{1}{c} \int n(\mathbf{r})\, ds,$$

где $n(\mathbf{r})$ — показатель преломления среды, ds — элемент длины пути, а c — скорость света в вакууме. Минимизация этого интеграла по всем возможным траекториям приводит к принципу наименьшего действия для лучей света. На основе этого принципа Гамильтон (William Rowan Hamilton, 1831) вывел уравнение, описывающее эволюцию фазы волны — уравнение Эйконала, ставшее фундаментом математической оптики.

Термин *eikonal* (от греческого «образ») закрепился в XX веке благодаря работам в области волновой теории и акустики.

В современном виде уравнение Эйконала записывается как:

$$|\nabla S(\mathbf{r})|^2 = n^2(\mathbf{r}),$$

где $S(\mathbf{r})$ — функция фазы или оптический путь, представляющая собой эквивалент расстояния, пройденного волной в среде с неоднородным показателем преломления. Поверхности $S = \text{const}$ соответствуют волновым фронтам, а линии, перпендикулярные этим поверхностям, — траекториям лучей.

Это уравнение является асимптотическим приближением уравнения Гельмгольца при коротких длинах волн ($\lambda \to 0$) и позволяет перейти от волнового описания к лучевому. Таким образом, уравнение Эйконала связывает пространственное распределение показателя преломления с геометрией распространения света и позволяет вычислять пути лучей в неоднородных средах. При постоянном $n(\mathbf{r}) = n_0$ уравнение имеет простое решение — прямолинейные лучи. Если же показатель преломления изменяется в пространстве, траектории лучей искривляются, отражая градиент $n(\mathbf{r})$.

Уравнения лучей и численные методы их решения

Из уравнения Эйконала можно вывести систему уравнений лучей, описывающую динамику направления распространения света в неоднородной среде:

$$\frac{d\mathbf{r}}{ds} = \mathbf{t}, \qquad \frac{d(n\mathbf{t})}{ds} = \nabla n,$$

где $\mathbf{r}(s)$ — радиус-вектор точки на траектории луча, $\mathbf{t}$ — единичный вектор касательной, s — длина дуги, а $n(\mathbf{r})$ — показатель преломления среды.

Эта система выражает баланс между направлением луча и градиентом показателя преломления: при увеличении $n(\mathbf{r})$ луч изгибается в сторону области с более высоким показателем.

Для численного решения этой системы уравнений применяются различные методы интегрирования обыкновенных дифференциальных уравнений. Метод Эйлера представляет собой простейшую линейную аппроксимацию, пригодную для случаев, когда изменение показателя преломления $n(\mathbf{r})$ относительно мало.

Метод Рунге–Кутты четвёртого порядка обеспечивает более высокую точность и устойчивость при умеренных градиентах показателя преломления и является стандартным инструментом для интегрирования траекторий лучей в оптических расчётах. Наконец, симплектические схемы интегрирования применяются в задачах, где важно сохранять физические инварианты системы — такие как энергия или угловой момент, — что особенно актуально при моделировании длинных оптических трактов и волноводных структур.

Результатом интегрирования является семейство траекторий лучей, формирующее пучок света, для которого можно оценить расходимость, фокусировку и распределение плотности энергии в различных областях пространства.

Применение геометрической оптики к моделированию трансформирующих сред

Оптические трансформирующие среды проектируются таким образом, чтобы управлять направлением распространения света. Математически задача моделирования сводится к нахождению такого распределения показателя преломления $n(\mathbf{r})$, при котором траектории лучей принимают требуемую форму.

Это так называемая *обратная задача геометрической оптики* — задача восстановления пространственного профиля показателя преломления по заданной форме волнового фронта или требуемому распределению интенсивности светового поля.

Примеры таких применений включают различные классы оптических устройств и сред. К ним относятся оптические концентраторы — структуры,

предназначенные для направленного сбора и концентрации световых потоков в ограниченной области, что обеспечивает повышение энергоэффективности систем и уменьшение потерь излучения. К числу трансформирующих сред также относятся градиентные (GRIN) структуры, в которых показатель преломления изменяется плавно, создавая непрерывное искривление траекторий лучей без резких границ раздела сред. Широкое применение находят волноводные переходы и трансформаторы пучков, обеспечивающие равномерное перераспределение энергии между модами и согласование полей на стыках оптических элементов. Особый интерес представляют метаматериалы, в которых пространственная модуляция эффективного показателя преломления реализует заданные функции управления светом — например, направленный изгиб без отражения, фокусировку вне оси, создание оптических плащей невидимости и других эффектов пространственного перенаправления излучения.

В таких задачах геометрическая оптика используется как первый этап моделирования: она позволяет оценить направленное распределение энергии и определить начальные условия для дальнейшего волнового анализа.

Достоинства и ограничения подхода

Методы геометрической оптики обладают рядом существенных преимуществ, определяющих их широкое применение в теоретических и прикладных исследованиях. Прежде всего, данный подход отличается высокой вычислительной эффективностью, что позволяет проводить моделирование сложных трёхмерных оптических структур без необходимости прямого решения уравнений Максвелла, требующих значительно больших вычислительных ресурсов. Кроме того, геометрическая оптика обеспечивает наглядную и интуитивно понятную интерпретацию процессов распространения света: визуализация траекторий лучей даёт возможность ясно представить функционирование оптической системы, оценить её фокусирующие и преобразующие свойства, а также качественно проанализировать влияние формы и расположения элементов на результирующее поле. Ещё одним важным достоинством является масштабируемость метода — он может с одинаковой успешностью применяться как к макроскопическим системам, таким как линзы, зеркала и призмы, так и к микроструктурам, при условии, что размеры элементов

исследуемой среды значительно превышают длину волны излучения. Благодаря этим особенностям геометрическая оптика служит эффективным инструментом предварительного анализа и оптимизации трансформирующих оптических сред, обеспечивая разумный баланс между физической достоверностью и вычислительной простотой.

Однако приближение геометрической оптики имеет ограничения. Оно не учитывает дифракцию, интерференцию и поляризацию, а также теряет точность при размерах элементов, сопоставимых с длиной волны. В этих случаях необходим переход к волновым методам (уравнение Гельмгольца, метод Фурье-оптики, BPM или FDTD).

Тем не менее, геометрическая оптика остаётся фундаментальной теоретической основой для понимания процессов распространения света и служит отправной точкой при построении численных моделей трансформирующих сред различной сложности.

Численное моделирование лучей

В простейшем случае линзу можно рассматривать как тонкий преломляющий элемент, изменяющий направление распространения лучей в соответствии с законом Снеллиуса:

$$n_1 \sin\theta_1 = n_2 \sin\theta_2,$$

где n_1 и n_2 — показатели преломления двух сред, а θ_1 и θ_2 — углы падения и преломления соответственно. Этот закон определяет фундаментальное соотношение между углами и показателями преломления и служит основой для описания поведения лучей при переходе через границу раздела сред.

В компьютерных моделях линза представляется как совокупность оптических поверхностей, каждая из которых задаётся аналитическим выражением $z = z(x, y)$ либо параметрическим уравнением. На основе таких описаний производится вычисление точек пересечения лучей с поверхностями и последующее определение новых направлений распространения после преломления. Подобные модели позволяют анализировать траектории лучей, распределение интенсивности в фокальной области, а также оценивать влияние геометрии поверхности и показателя преломления на фокусирующие свойства системы.

Процесс моделирования хода лучей в трансформирующих оптических средах представляет собой последовательность этапов, направленных на воспроизведение и анализ процессов распространения света в заданной геометрии.

1. Определение параметров линзы. На первом этапе осуществляется определение параметров оптической системы, включающее выбор радиусов кривизны поверхностей, толщины элементов, показателя преломления материала и взаимного расположения компонентов. Эти характеристики задают геометрию среды, в которой будет происходить распространение излучения, и определяют её фокусирующие и трансформирующие свойства.

2. Задание исходного пучка лучей. Следующий этап заключается в задании исходных условий для моделирования — формировании пучка лучей, который будет использоваться в качестве входного сигнала. Определяются пространственные координаты источника, диапазон направлений, апертура пучка и его пространственно-угловое распределение. От корректности задания этих параметров зависит достоверность результатов, поскольку они определяют область и характер взаимодействия света с элементами системы.

3. Применение закона преломления на каждой границе. После этого выполняется вычисление взаимодействия лучей с поверхностями среды. На каждой границе раздела сред применяется закон Снеллиуса, описывающий изменение направления распространения света при переходе между областями с различными показателями преломления. На основе этого закона для каждого луча вычисляется новое направление распространения, что позволяет пошагово проследить его путь через всю систему.

4. Построение траекторий лучей после прохождения линзы. Далее осуществляется построение траекторий лучей и их визуализация в пространстве модели. Эти данные дают возможность определить, каким образом пучок света изменяет своё направление, форму и распределение интенсивности под воздействием исследуемой оптической структуры. На данном этапе выявляются фокусирующие свойства системы, оцениваются положения фокальных областей, а также анализируются возможные искажения. Для выполнения вычислений траекторий применяется метод трассировки лучей (*ray tracing*), представляющий собой алгоритм, последовательно вычисляющий путь каждого луча через систему поверхностей. В рамках этого метода луч рассматривается как дискретная частица, последовательно пересекающая

элементы оптической системы, отражающаяся и преломляющаяся в соответствии с физическими законами. В простейших случаях трассировка может выполняться аналитически, если уравнения поверхностей имеют известный функциональный вид. В более сложных конфигурациях, особенно при наличии неоднородных или градиентных сред, используется численное интегрирование траекторий, позволяющее учитывать непрерывное изменение показателя преломления вдоль пути распространения света.

5. Анализ фокусировки, аберраций и распределения интенсивности в фокальной плоскости. Завершающим этапом моделирования является анализ результатов расчёта. Проводится оценка фокусировки и расходимости пучка, исследуется распределение интенсивности в фокальной плоскости, выявляются области с максимальной концентрацией энергии и участки с потерями из-за отражений или аберраций. Полученные данные позволяют количественно оценить эффективность оптической системы, определить степень соответствия её параметров проектным требованиям и при необходимости скорректировать геометрию или материал среды.

Таким образом, численное моделирование хода лучей представляет собой комплексный процесс, сочетающий физические принципы распространения света с вычислительными методами анализа. Его результаты обеспечивают глубокое понимание работы оптических систем и служат основой для дальнейшей оптимизации их структуры, направленной на достижение требуемых характеристик фокусировки, пропускания и преобразования световых потоков.

Программные реализации

Современные вычислительные методы моделирования оптических систем реализуются в виде специализированных программных комплексов, предназначенных для решения задач трассировки лучей, анализа оптических характеристик и оптимизации параметров элементов. Наиболее широко применяются профессиональные пакеты *Zemax OpticStudio*, *Code V* и *TracePro*, представляющие собой полнофункциональные среды для проектирования и анализа оптических систем различной степени сложности. Эти программы обеспечивают возможность моделирования распространения света в трёхмерных структурах, расчёта фокусных характеристик, аберраций

и освещённости, а также позволяют выполнять автоматизированную оптимизацию конфигурации элементов в соответствии с заданными критериями качества изображения или эффективности передачи излучения. Благодаря развитым графическим интерфейсам и высокой точности вычислений, данные комплексы стали промышленным стандартом при проектировании объективов, осветительных приборов и других оптических устройств.

Особое место занимает программная среда *COMSOL Multiphysics* с модулем *Ray Optics Module*, которая объединяет возможности геометрической и волновой оптики в рамках единой многофизической платформы. Такой подход позволяет проводить комплексные исследования, в которых учитываются не только геометрические аспекты распространения света, но и электромагнитные, тепловые или механические эффекты, возникающие при взаимодействии излучения с материалами. Использование данной среды особенно актуально при моделировании гибридных оптических систем, где важно согласовать результаты, полученные в различных физических приближениях.

Для научных и образовательных целей всё более активно применяются открытые и гибкие инструменты, реализованные на языках программирования общего назначения. Среди них можно выделить библиотеки на языке Python, такие как *Raysect*, *PyOpTools* и *POV-Ray*. Эти программные средства позволяют строить настраиваемые модели оптических систем, выполнять численные расчёты траекторий лучей и визуализировать результаты моделирования. Их использование обеспечивает исследователям возможность адаптации алгоритмов под конкретные задачи, интеграции с другими вычислительными библиотеками (например, для анализа данных или оптимизации), а также воспроизводимости вычислительных экспериментов в рамках открытой научной среды.

Независимо от конкретного программного инструмента, принцип, лежащий в основе всех реализаций, остаётся единым и базируется на решении уравнений геометрической оптики, описывающих распространение света в пространстве с заданным распределением показателя преломления. Во всех случаях целью моделирования является получение достоверных данных о траекториях лучей, фокусирующих свойствах системы и распределении световой энергии, что позволяет как анализировать существующие оптические конструкции, так и разрабатывать новые, обладающие улучшенными характеристиками. Таким образом, программные средства моделирования выступают

неотъемлемым компонентом современного оптического проектирования, объединяя физическую строгость теоретических моделей с удобством вычислительных и визуальных инструментов.

1.2. Линзы Люнеберга, Максвелла и Итона

1.2.1. Общее описание линз

Рассмотрим линзу, представляющую собой сферу с центром в точке X_0 с радиус вектором $\mathbf{X}_0$. Источник электромагнитных волн разместим в точке с радиус вектором $\mathbf{x}_0$.

Следует сразу же подчеркнуть, что используемый метод численного решения уравнения эйконала влечет разное математическое описание электромагнитной волны.

— При использование метода характеристик естественной является интерпретация излучения в виде лучей. Каждый луч в этом случае является решением системы ОДУ для заданных начальных значений обобщенных координат $\mathbf{x}$ и импульсов $\mathbf{p}$. Начальные значения координат $\mathbf{x}_0$ задают положение источника, то есть начало луча, а начальные значения импульсов $\mathbf{p}_0$ — направление луча.

— При использовании метода FSM используется уже волновая интерпретация оптики и считается, что функция эйконала $u(\mathbf{x})$ изначально задана в каждой точки пространства, а точнее в каждой точке сетки (см. раздел 2.3). Местоположение источника задается граничным условием $u(\mathbf{x}_0) = 0$ системы (**??**), где $\mathbf{x}_0$ — радиус вектор точек, принадлежащих источнику.

На рисунках 1.1 и 1.2 изображены линзы, на которые падает электромагнитное излучение из точечного источника (рис. 1.1) и с плоского протяженного источника (рис. 1.2). Излучение на рисунках представлено в виде лучей, однако в методе FSM нет никакой возможности явно задавать направление и источник лучей, так как считается, что излучение уже присутствует в каждой точке рассматриваемой области. Это вызывает определенные сложности при необходимости визуализировать именно лучевую оптическую картину.

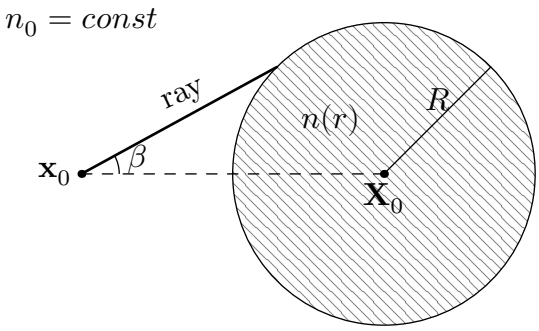


Рис. 1.1. Линза с точечным источником

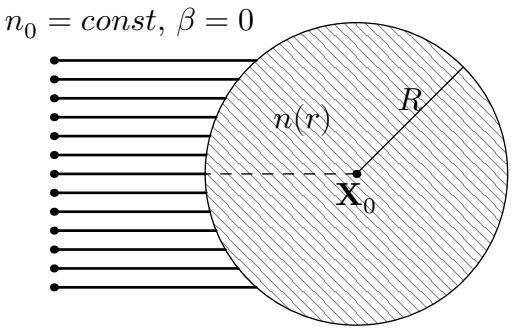


Рис. 1.2. Линза с плоским источником

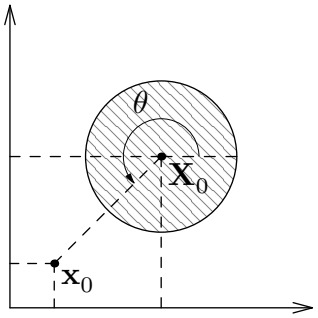


Рис. 1.3. Точечный источник относительно линзы

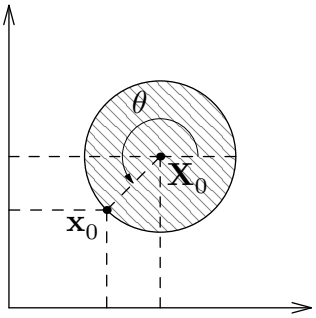


Рис. 1.4. Точечный источник на поверхности линзы

Показатель преломления вне линзы постоянен и равен n_0, а внутри линзы является функцией расстояния от центра линза до текущей точки $n(r)$, где $r = \|\mathbf{x} - \mathbf{X}_0\|$. Для упрощения вычислений следует разместить центр линзы в начале координат. Это особенно упрощает решение задачи в цилиндрических и сферических координатах.

Местоположение точечного источника может быть удобно задавать относительно линзы. Тогда его координаты определяются радиусом линзы R, расстоянием от линзы до источника d и у углом θ, который откладывается против часовой стрелки в правой системе координат, как показано на рисунке 1.3. Тогда радиус вектор источника задается с помощью параметрического уравнения окружности с радиусом $R + d$

$$\mathbf{x}_0 = \mathbf{X}_0 + (d + R)(\cos\theta, \sin\theta)^T.$$

В частности, если источник лежит на линзе так, как это показано на рисунке 1.4, то его координаты задаются радиус вектором

$$\mathbf{x}_0 = \mathbf{X}_0 + R(\cos\theta, \sin\theta)^T.$$

Для работы численной схемы метода FSM не требуется знать производные от функции показателя преломления $n(\mathbf{x})$, что можно рассматривать как преимущество данного метода по сравнению с методом характеристик.

1.2.2. Линза Люнеберга

Линза Люнеберга [19; 68] представляет собой сферическую линзу радиуса R с центром в точке (X_0, Y_0, Z_0) с коэффициентом преломления следующего вида

$$n(r) = \begin{cases} n_0\sqrt{2 - \left(\frac{r}{R}\right)^2}, & r \leqslant R, \\ n_0, & r > R, \end{cases}$$

где в декартовых координатах $r(x, y, z) = \sqrt{(x - X_0)^2 + (y - Y_0)^2 + (z - Z_0)^2}$ — расстояние от центра линзы до произвольной точки (x, y, z). Из формулы следует, что коэффициент n непрерывно меняется от $n_0\sqrt{2}$ до n_0 начиная от центра линзы и заканчивая ее границей. Для вычислений удобнее переписать выражение для r в индексном виде:

$$r(x^1, x^2, x^3) = \sqrt{(x^1 - X_0^1)^2 + (x^2 - X_0^2)^2 + (x^3 - X_0^3)^2} = \sqrt{\sum_{i=1}^{3}(x^i - X_0^i)^2}.$$

1.2.3. Линза Максвелла

Линза Максвелла [3; 68] также представляет собой сферическую линзу радиуса R с центром в точке $\mathbf{X}_0$ с коэффициентом преломления следующего вида:

$$n(r) = \begin{cases} \dfrac{n_0}{1 + \left(\frac{r}{R}\right)^2}, & r \leqslant R, \\ n_0, & r > R. \end{cases}$$

На рисунке 1.5 представлены графики изменения коэффициента преломления для линз Максвелла и Люнеберга в зависимости от радиус вектора точки.

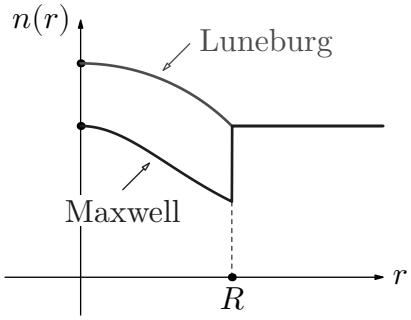


Рис. 1.5. Коэффициент преломления для линз Максвелла и Люнеберга

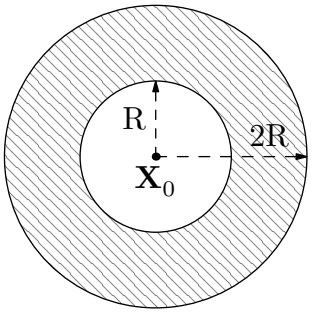


Рис. 1.6. Схема линзы Итона

1.2.4. Линза Итона

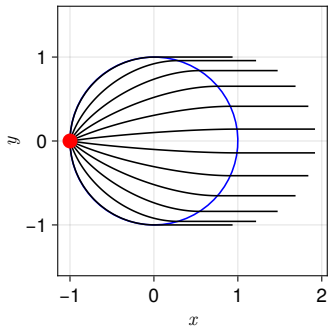
В плоском случае линза Итона [66] представляет собой диск образованный двумя кругами с радиусами R и $2R$, что изображено на схеме линзы на рисунке 1.6.

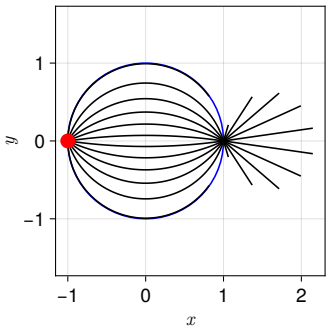
$$n(r) = \begin{cases} n_0\sqrt{\dfrac{2R}{r} - 1}, & r \in [R, 2R], \\ n_0 & r \notin [R, 2R]. \end{cases}$$

1.2.5. Размещение источников

Оптические свойства линз проявляются наглядно, если разместить источник излучения в определенной точке.

— Для линзы Люнеберга точечный источник обычно размещается на поверхности линзы, тогда исходящие лучи пройдя сквозь линзу располагаются параллельно друг-другу, как показано на рисунке 1.7.





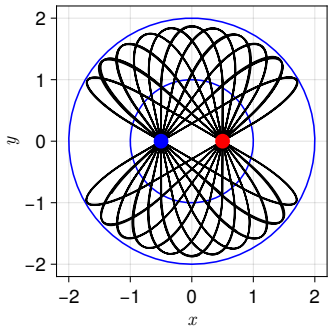
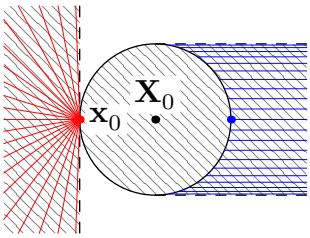


Рис. 1.7. Линза Люнеберга **Рис. 1.8. Линза Максвелла** **Рис. 1.9. Линза Итона**



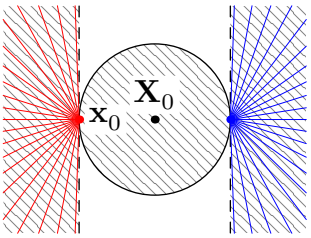


Рис. 1.10. Линза Люнеберга **Рис. 1.11. Линза Максвелла**

— Для линз Максвелла точечный источник также размещают на поверхности линзы. В этом случае исходящие лучи фокусируются и сходятся в диаметральной от источника точки, как это показано на рисунке 1.8.

— Для линзы Итона источник размещается внутри сферы (окружности) малого диаметра R. В этом случае все излучение не выходит за пределы линзы и лучи фокусируются в точке, симметричной относительно центра линзы, как это показано на рисунке 1.9.

Заметим, что ход лучей на рисунках 1.7, 1.8 и 1.9 вычислен с помощью метода характеристик [43].

Если рассматривать вышеописанную конфигурацию источников, то некоторые области пространства будут полностью свободны от лучей. Так, для линзы Итона все лучи будут заключены внутри большого круга линзы. Для линз Люнеберга и Максвелла области, где могут проходить лучи, показаны штриховкой на изображениях 1.10 и 1.11. В области белого цвета ни один луч проникнуть не может.

Отметим, что рисунки 1.10 и 1.11 являются лишь схемами, условно показывающими области присутствия и отсутствия лучей. Внутри линзы, однако траектории лучей не изображены.

1.3. Численные методы на основе методов нейронных сетей

В последние годы наблюдается устойчивый рост интереса к использованию методов машинного обучения, в частности нейронных сетей, в области численного моделирования физических процессов. Это связано как с теоретическими достижениями в области глубинного обучения, так и с практическими вызовами, с которыми сталкиваются традиционные численные подходы. Несмотря на высокую точность и развитость классических методов решения дифференциальных уравнений — таких как конечно-разностные, конечно-объёмные и конечно-элементные схемы — они имеют ряд ограничений, особенно при решении задач высокой размерности, в условиях сложной геометрии или при наличии лишь частичных данных.

Нейросетевые методы позволяют по-новому взглянуть на моделирование физических процессов [1; 16; 51]. Вместо явной дискретизации пространства и времени, характерной для большинства традиционных подходов, нейросети аппроксимируют решение уравнения в виде непрерывной функции, что особенно удобно для задач, где важны производные или гладкость решения. Кроме того, современные архитектуры нейросетей способны обучаться на основе ограниченного набора данных, комбинируя информацию, полученную из наблюдений, с априорными физическими знаниями. В частности, подход PINN (Physics-Informed Neural Networks) [51] предлагает включение дифференциального уравнения и граничных условий непосредственно в функцию потерь, тем самым обеспечивая согласование с законами физики и устойчивость к шумам и неполноте данных.

Особенно привлекательным применение нейросетей становится в задачах инверсного моделирования — когда необходимо восстановить параметры среды, источники или граничные условия по результатам наблюдений [8; 16]. Эти задачи, как правило, неустойчивы и плохо решаются классическими методами без регуляризации. Нейросети в этом контексте выступают не просто как аппроксиматоры, а как адаптивные модели, способные уловить сложные зависимости между параметрами задачи и её решением.

Ещё одно существенное преимущество нейросетевых методов — возможность многократного переиспользования обученной модели. Например, при обучении универсального оператора (в архитектурах типа DeepONet или

Fourier Neural Operator [20; 32]) можно один раз обучить нейросеть на множестве решений задачи с различными начальными и граничными условиями, после чего мгновенно получать предсказания для новых конфигураций без повторного численного решения уравнения. Это открывает широкие перспективы в задачах инжиниринга, оптимизации и инверсного дизайна, где требуется проводить тысячи или миллионы моделирований при разных параметрах.

Применение нейронных сетей в численном моделировании позволяет сочетать преимущества физических моделей с гибкостью и обучающей способностью современных нейросетевых архитектур. Эти методы особенно актуальны в тех случаях, когда традиционные численные подходы становятся избыточно ресурсоёмкими, требуют чрезмерной детализации геометрии или не справляются с неполными или шумными данными. Несмотря на сохраняющиеся вызовы — такие как высокая чувствительность к выбору архитектуры и сложности обучения — нейросетевые численные методы формируют новое направление в вычислительной физике и инженерии, обеспечивая качественно новый уровень адаптивности, обобщающей способности и интеграции данных с физикой.

Нейросетевые подходы выигрывают у традиционных численных методов прежде всего в тех задачах, где численные схемы сталкиваются с фундаментальными ограничениями — как вычислительными, так и методологическими [38; 59]. Одним из ключевых примеров являются задачи с высокой размерностью, где стандартные методы страдают от так называемого «проклятия размерности» [13; 32]. При увеличении числа переменных, особенно в многопараметрических дифференциальных уравнениях в частных производных или в системах с пространственно-временной динамикой, объём вычислений при использовании сеточных методов растёт экспоненциально. Нейросети, напротив, сохраняют устойчивую аппроксимирующую способность и позволяют эффективно работать в таких пространствах, обучаясь на ограниченном количестве примеров и не требуя явного построения сетки.

Другой важный класс задач — это те, где доступна только частичная или разреженная информация. Например, в практических задачах оптики, акустики, биофизики или геофизики мы зачастую имеем дело не с полным распределением поля, а с наблюдениями на границах или в ограниченном числе точек.

Для численных методов такие условия создают необходимость искусственного дополнения данных и могут приводить к нестабильным или нерелевантным результатам. Нейросети, и особенно PINN, встраивая в процесс обучения как дифференциальное уравнение, так и имеющиеся наблюдения, могут реконструировать глобальное поведение решения из локальных данных, при этом соблюдая физические законы [51].

Отдельно стоит отметить обратные задачи, в которых по известным выходным характеристикам системы необходимо восстановить её внутренние параметры или начальные условия. Эти задачи в силу своей некорректной постановки (неединственности, нестабильности) требуют регуляризации и тонкой настройки. Нейросетевые методы встраивают априорную информацию в архитектуру модели или функцию потерь, благодаря чему становятся более устойчивыми к шуму, лучше справляются с недоопределённостью и в ряде случаев дают решения, сопоставимые или превосходящие по точности классические регуляризационные методы.

Существенное преимущество нейросетей проявляется и в ситуациях, когда необходимо решать одну и ту же задачу многократно при разных входных параметрах. Это особенно актуально в задачах инжиниринга и оптимизации, где приходится многократно пересчитывать модели при варьировании параметров конструкции. Обученная нейросеть может выступать в роли универсального решателя, мгновенно выдавая предсказания без необходимости повторного численного решения. Такое поведение делает нейросетевые модели чрезвычайно ценными в реальном времени или в задачах, требующих массового параллельного моделирования.

Есть и другие аспекты, в которых нейросетевые подходы демонстрируют преимущество. Например, в задачах со сложной или плохо формализуемой геометрией: при наличии острых углов, разрывов, интерфейсов. Классические сеточные методы требуют специальной адаптации сетки, повышения плотности узлов, ручной обработки краевых условий. Нейросети в подобных условиях могут обучаться напрямую на исходной области, без необходимости в явном построении сетки, и при этом сохранять высокую точность за счёт гладкости аппроксимации.

Наконец, нейросетевые подходы находят применение в задачах, где важна интерполяция или экстраполяция за пределами обучающей выборки. Там, где

численные методы дают результат только на заранее заданной сетке и параметрах, нейросети могут предсказывать поведение системы при новых, ранее не встречавшихся конфигурациях, благодаря своей обобщающей способности. Это особенно ценно, если модель используется как часть более сложной системы — например, в робототехнике, цифровом двойнике, или при разработке систем управления в реальном времени.

Нейросетевые подходы оказываются наиболее выгодными в условиях ограниченности данных, высокой размерности, необходимости инверсного анализа или множественного моделирования. Их сила заключается в способности учиться на малых выборках, учитывать физику в структуре модели, работать без сетки и обеспечивать быструю генерализацию. Конечно, они не отменяют численные методы, а скорее дополняют их, расширяя границы применимости вычислительных моделей и открывая новые возможности для анализа сложных физических процессов.

Необходимость обхода ограничений традиционных численных методов возникает тогда, когда сами принципы этих методов — основанные на дискретизации, сеточных представлениях и строгой пошаговой аппроксимации — становятся узким горлышком в решении задачи. Это бывает не всегда, но в определённых условиях такие ограничения становятся критичными, и тогда применение альтернативных подходов, включая нейросетевые, становится не просто предпочтительным, а порой единственно возможным.

Во-первых, это задачи с высокой размерностью. Если задача описывается не просто двумя-тремя пространственными переменными, а десятками параметров или временных моментов, то классические методы страдают от экспоненциального роста числа узлов сетки. Например, при моделировании многомасштабных процессов или при параметризации среды с переменными физическими свойствами в пространстве и времени. Для трёхмерной задачи, к примеру, с сеткой 100^3 и мелким шагом по времени требуется хранить и обрабатывать миллионы, если не миллиарды точек — это приводит к чрезмерному потреблению памяти и времени, особенно если задача решается многократно. Нейросетевые подходы, такие как PINN или операторные модели, могут эффективно представлять такие системы в компактной форме, используя небольшое число параметров.

Во-вторых, сложности возникают в задачах с нерегулярной или динамически изменяющейся геометрией. Построение адекватной сетки для подобных

областей требует значительных усилий, адаптивных алгоритмов перестроения и ручной настройки. При этом возможны ошибки на границах, особенно при наличии острых краёв, разрывов или подвижных интерфейсов. В нейросетевых подходах, особенно тех, которые используют координаты как вход, проблема сетки исчезает как класс — аппроксимация строится глобально, и геометрия может быть задана, например, в виде маски или параметрической формы.

Третья категория — это обратные и плохо определённые задачи, где количество наблюдений намного меньше количества неизвестных. Это распространённая ситуация в задачах диагностики, томографии, параметрической идентификации, когда мы имеем только выход системы, но не знаем, какие входы или параметры его вызвали. Традиционные методы требуют регуляризации, априорной информации, часто страдают от неустойчивости и множественности решений. Нейросетевые методы, обучающиеся на множестве возможных сценариев или использующие физику в качестве ”направляющей” в обучении, дают более устойчивые и воспроизводимые результаты, особенно в условиях шума и неполных данных [51].

Четвёртая группа ограничений — это требования к скорости расчёта при многократных вызовах модели. Например, в задачах оптимизации формы дифракционной решётки, где на каждом шаге требуется пересчитать волновое поле, или при обучении цифрового двойника, который должен реагировать на изменения параметров в реальном времени. Классические методы не успевают пересчитывать задачу с нуля за миллисекунды или даже секунды. Нейросеть, один раз обученная на множестве входных конфигураций, способна выдавать решение мгновенно — без повторного решения системы уравнений.

Также ограничения проявляются при неполных, разреженных или шумных данных, особенно когда решение требуется продолжить за пределами измеренной области. В классических численных схемах такая задача может быть нерешаема или крайне нестабильна. Нейросети же хорошо приспособлены к работе с такими типами данных, так как могут обучаться не только на полном решении, но и на локальных наблюдениях, восполняя недостающую информацию за счёт встроенного априорного знания (например, физического уравнения).

Наконец, обход ограничений требуется, когда необходимо не просто численное решение, а обобщающая модель, способная переносить знания между различными задачами. Например, обучив нейросеть на дифракции в одном классе структур, можно использовать её для предсказаний в других, смежных классах, без повторного численного моделирования. Это особенно полезно в автоматизированном проектировании (inverse design), где требуется быстрый отклик модели на вариации входных параметров.

Обход ограничений численных методов необходим не из-за их недостатков как таковых, а потому что в реальных задачах условия выходят за рамки идеализированных, стационарных, хорошо определённых систем, под которые изначально и проектировались традиционные схемы. Нейросетевые подходы предлагают альтернативный путь — не замену, а расширение возможностей численного анализа в условиях, где прямое применение классики становится неэффективным, затратным или невозможным.

Численные методы на основе нейронных сетей формируют новую парадигму вычислений, в которой традиционные подходы к решению дифференциальных уравнений уступают место или дополняются обучаемыми моделями, способными учитывать физику, данные и априорные знания одновременно. Рассмотрим некоторые из них.

1.3.1. Методы, основанные на вариационных принципах. Deep Ritz Method (DRM)

Методы, основанные на вариационных принципах, применяемые в контексте нейронных сетей, представляют собой естественное развитие классических вариационных методов решения дифференциальных уравнений, которые являются ключевым элементом современной прикладной математики, механики и физики. Эти методы предполагают, что решение уравнения можно получить как экстремум некоторого функционала, обычно выражаемого в виде интеграла [13; 15]. Они находят широкое применение при решении обыкновенных и дифференциальных уравнений в частных производных, особенно в контексте физических и инженерных задач.

Текущие подходы берут за основу идею о том, что решение определённого класса уравнений может быть представлено как минимум или экстремум некоторого функционала. В наиболее частом случае — это интеграл, зависящий от

искомой функции и её производных, и задача сводится к нахождению такой функции, которая минимизирует этот интеграл.

В традиционной постановке, например, при решении уравнения Пуассона, мы ищем функцию $u(x)$, которая минимизирует функционал вида

$$J(u) = \int_\Omega \left(\frac{1}{2} |\nabla u(x)|^2 - f(x)u(x) \right) dx$$

где Ω — область, а $f(x)$ — заданная функция. Этот функционал возникает из вариационной формулировки соответствующего краевого уравнения и имеет физический смысл энергии системы. Классический подход к решению таких задач восходит к методу Ритца, в котором решение аппроксимируется линейной комбинацией базисных функций, а задача сводится к минимизации функционала по коэффициентам этой комбинации. Метод Ритца был подробно описан в работах Вальтера Ритца в начале 20 века и впоследствии обоснован в рамках вариационного исчисления.

Нейросеть в этом случае выступает как универсальный аппроксиматор искомой функции $u(x)$, минимизирующий интеграл от функционала, что аналогично методам Ритца или Галёркина, но без явного задания базисных функций. Такие методы особенно полезны в задачах с чётко выраженной вариационной природой, например, при решении уравнения Пуассона [13; 15], и позволяют легко переносить идею на произвольные геометрии или нелинейные задачи. Вместо разложения по жёстко заданному базису, нейросеть принимает на вход координаты xxx и возвращает значение $u(x)$ а минимизация функционала $J(u)$ осуществляется путём настройки весов сети с помощью градиентных методов. Это ключевое отличие от классического Ритца: здесь не нужно явно выбирать базисные функции, и нейросеть может сама “научиться” наиболее подходящей структуре решения. Такой подход получил название Deep Ritz Method, и впервые был подробно описан в работе Weinan E и Bing Yu в 2018 году [13].

Одним из преимуществ вариационного подхода с использованием нейросетей является его естественная способность адаптироваться к произвольной геометрии области и различным типам граничных условий. Поскольку интеграл в функционале берётся по всей области определения, можно использовать набор точек, равномерно или случайным образом распределённых

внутри этой области. При необходимости граничные условия могут быть учтены с помощью добавления специальных штрафных членов в функционал или за счёт выбора архитектуры сети, автоматически удовлетворяющей этим условиям. Такой подход особенно удобен при решении задач в геометрически сложных или трудноформализуемых областях, где построение классической расчётной сетки либо сопряжено со значительными трудностями, либо вовсе невозможно.

Дополнительным достоинством является то, что обучение нейросети в данном случае осуществляется не на заранее известных значениях функции в отдельных точках, а по глобальному критерию, связанному со значением всего функционала. Это придаёт методу устойчивость к локальным ошибкам и шуму, а также способствует получению сглаженных, физически осмысленных решений. В тех случаях, когда задача сводится к поиску собственных значений, как, например, в задачах квантовой механики или волновых процессов, вариационная постановка позволяет напрямую минимизировать энергетический функционал и находить приближённые собственные значения и соответствующие собственные функции.

Следует отметить, что у методов, основанных на вариационных принципах и реализованных с использованием нейросетей, есть и определённые особенности. В частности, вычисление функционала и его производных требует численного интегрирования по области, что может быть особенно ресурсоёмким в задачах с высокой размерностью. Кроме того, обучение таких моделей чувствительно к масштабу входных данных и параметров функционала: при наличии резко различающихся по величине вкладов, например, от локальных источников высокой интенсивности, может потребоваться дополнительная нормализация или балансировка весов в функции потерь. Несмотря на эти сложности, вариационные методы на базе нейросетей остаются мощным инструментом численного моделирования, объединяющим строгую физическую интерпретацию с гибкостью и аппроксимирующими возможностями современных алгоритмов машинного обучения.

1.3.2. Нейросетевые операторы. Neural Operators (DeepONet, FNO)

Развивается также класс методов, называемых нейросетевыми операторами. Они представляют собой новое поколение численных алгоритмов, в которых основное внимание уделяется не построению конкретного решения уравнения при заданных начальных или граничных условиях, а аппроксимации самого оператора — отображения, сопоставляющего исходным данным решение задачи. В классических численных методах решение уравнения с частными производными предполагает фиксированную дискретизацию области и решение соответствующей разностной или слабой формы задачи. При этом каждый новый набор входных данных требует повторного решения уравнения, зачастую с нуля. В противоположность этому, нейросетевые операторы стремятся обучить обобщённую модель, способную предсказывать решение для любого допустимого входа без повторного пересчёта.

Одной из первых реализованных архитектур нейросетевых операторов стала DeepONet, предложенная Л. Лу и Дж. Карниадакисом [32]. В её основе лежит теорема об универсальной аппроксимации операторов, согласно которой можно приближать отображения между функциями с помощью нейросетей. Архитектура DeepONet построена таким образом, чтобы обрабатывать информацию о входной функции и координате точки, в которой необходимо получить значение решения, через два параллельных потока обработки данных. Первый поток, называемый ветвью, принимает на вход значения входной функции в ряде опорных точек, формируя параметрическое описание задачи. Второй поток, или ствол, получает координату интересующей точки. Объединяя выходы этих двух компонентов, модель способна аппроксимировать значение решения в любой точке области для заданной функции на входе.

Другим важным направлением в развитии нейросетевых операторов стала архитектура Fourier Neural Operator (FNO) [20], основанная на спектральном подходе к обработке данных. В отличие от методов, которые работают напрямую с функциями в обычном пространстве, FNO использует преобразование Фурье для перехода к частотному представлению, где выполнение операций становится более эффективным. Это позволяет модели улавливать глобальные зависимости в решении, в том числе дальние связи между различными

участками области. После обработки в спектральной области результат преобразуется обратно в исходное пространство. Такая стратегия особенно хорошо проявила себя в задачах с пространственно сложной динамикой, например, при моделировании турбулентных потоков, атмосферных явлений или биологических процессов. Благодаря своей универсальности и высокой вычислительной эффективности FNO может применяться к задачам различной размерности и параметрическим уравнениям с высокой степенью обобщения.

Одним из ключевых достоинств нейросетевых операторов является их универсальность: после обучения на наборе задач с различными входными функциями они могут быть применены к новым входам без переобучения, тем самым обеспечивая мгновенное получение решения. Это делает такие методы особенно актуальными в сценариях, где требуется многократный пересчёт модели при изменяющихся условиях, а также в задачах с высокой размерностью входных параметров. Ещё одним преимуществом является их устойчивость к шуму и способность восстанавливать гладкие и физически обоснованные решения даже при наличии ошибок или неполных данных во входной информации.

Однако, несмотря на высокую эффективность, нейросетевые операторы остаются областью активных исследований. Важными остаются вопросы о свойствах аппроксимации, стабильности, интерпретируемости и строгости оценки ошибки. Тем не менее, уже сейчас можно утверждать, что методы типа DeepONet и FNO открывают принципиально новый путь в численном решении дифференциальных уравнений, обеспечивая сочетание вычислительной эффективности, адаптивности и способности к обобщению, которые недостижимы в рамках традиционных подходов.

1.3.3. Гибридные подходы. Data-Driven Hybrid Models

Существуют и гибридные подходы, в которых нейросеть не полностью заменяет численную схему, а встраивается внутрь неё.

В подобных подходах нейросеть не заменяет численное решение целиком, а встраивается в него как дополнительный элемент: уточняющий, дополняющий или аппроксимирующий. Это позволяет сохранять физическую интерпретируемость вычислений и использовать при этом преимущества

нейросетей — способность обучаться на данных, выявлять скрытые зависимости и компенсировать недостатки моделей [39; 55].

Наиболее естественное применение такие модели находят в задачах, где физическая структура процесса хорошо известна, но некоторые параметры — например, свойства среды, граничные условия или распределение источников — заданы неточно или вовсе неизвестны. В таких случаях нейросеть может быть обучена на экспериментальных или численно полученных данных и использоваться для аппроксимации этих недостающих элементов, в то время как основное уравнение решается традиционным методом, например методом конечных элементов или конечных разностей. Подобная комбинация обеспечивает более точное и устойчивое решение задачи, чем любой из подходов по отдельности.

Другой важный пример использования гибридных методов — ускорение вычислений в задачах, требующих многократного моделирования. Так, в оптимизационных задачах или при построении цифровых двойников нейросеть может быть обучена предсказывать поведение системы по ограниченному числу параметров, существенно сокращая объём расчётов. При этом она опирается не только на данные, но и на структуру уравнения, что позволяет сохранять физическую обоснованность предсказаний.

Гибридные методы также находят применение в инверсных задачах, где требуется восстановить скрытые характеристики объекта или среды по доступным наблюдениям. Нейросеть здесь играет роль аппроксиматора обратной зависимости, а численный решатель проверяет корректность полученного результата. Такая связка особенно полезна, например, в медицинской диагностике, геофизике, анализе конструкций и дефектоскопии, где прямая модель известна, но параметры измеряются косвенно или с шумом.

Основное достоинство гибридных подходов заключается в их балансе между надёжностью и гибкостью: с одной стороны, они сохраняют строгую физическую основу, с другой — позволяют учесть реальные данные и эмпирические особенности, которые трудно формализовать аналитически. Это делает их особенно востребованными в инженерных и прикладных задачах, где важно одновременно учитывать как теоретическую модель, так и поведение реальной системы.

1.3.4. Моделирование динамики в латентном пространстве с использованием автоэнкодеров. Latent Space Evolution via Autoencoders

Одним из наиболее оригинальных и активно развивающихся в применении нейронных сетей к численному моделированию является моделирование динамики в скрытом (латентном) пространстве, получаемом с помощью автоэнкодеров. Основная идея заключается в том, чтобы не решать уравнение в исходном пространстве напрямую, где оно может быть сложным, высокоразмерным или жёстким, а сначала преобразовать данные в более компактное представление, отражающее основные характеристики системы. В дальнейшем моделирование выполняется именно в этом сокращённом пространстве, где динамика становится проще и вычислительно менее затратной [33; 64].

Автоэнкодер состоит из двух основных компонентов: энкодера, преобразующего входные данные в сжатую форму — так называемый латентный вектор, — и декодера, позволяющего восстановить исходную информацию по этому вектору. Такое представление может эффективно устранять избыточность, шум и второстепенные детали, сохраняя при этом структуру, необходимую для точного описания поведения системы. В дальнейшем можно обучить дополнительную модель, которая будет моделировать, как именно эта скрытая переменная изменяется во времени или в зависимости от других параметров задачи.

Подобный подход особенно актуален в задачах, где численное моделирование оказывается затруднено из-за высокой размерности, жёсткости уравнений или необходимости многократного пересчёта решений. Вместо того чтобы каждый раз выполнять полное моделирование в физическом пространстве, можно предсказать, как будет изменяться компактное латентное представление, и затем восстановить искомое состояние системы при помощи декодера. Это позволяет значительно снизить вычислительные затраты без существенной потери точности.

Методы, основанные на автоэнкодерах, находят применение в самых разных областях: от моделирования турбулентных течений и химических реакций до задач биомеханики и материаловедения. Особенно они эффективны там, где наблюдаются устойчивые повторяющиеся структуры, высокая степень корреляции между параметрами, а также наличие неполных или зашумлённых данных

Главное преимущество такого подхода заключается в значительном снижении размерности моделируемой задачи при сохранении физически осмысленного поведения системы. Это делает его особенно ценным инструментом в тех случаях, когда классические численные методы либо слишком затратны, либо вовсе неприменимы. Вместе с тем, такие модели требуют тщательной настройки, а результаты — аккуратной валидации, особенно при прогнозировании за пределами обучающего диапазона.

1.3.5. Графовые нейронные сети. Graph Neural Networks for PDEs

Графовые нейронные сети (Graph Neural Networks, GNN) представляют собой перспективный класс моделей [14; 25; 36], способных обрабатывать данные, заданные не на регулярных сетках, а на графах — структурах, отражающих топологические и геометрические связи между объектами. В задачах численного моделирования это особенно актуально, поскольку многие реальные физические процессы происходят в средах со сложной, нерегулярной геометрией, где построение традиционной координатной сетки либо затруднено, либо приводит к значительным вычислительным затратам.

Суть подхода заключается в том, что рассматриваемая область моделирования представляется в виде графа, где узлы соответствуют отдельным точкам (например, элементам сетки или наблюдаемым точкам), а рёбра отражают физическую или геометрическую связь между ними. На узлы и рёбра могут быть назначены значения интересующих величин — например, температуры, давления или плотности. Графовая нейросеть обучается так, чтобы передавать информацию между связанными узлами, формируя аппроксимацию решения уравнения за счёт многократного обновления представлений узлов с учётом их окружения.

Подход с использованием GNN особенно эффективен при работе с неструктурированными или адаптивными сетками, где применение классических численных методов требует трудоёмкой подготовки сетки и её постоянной адаптации. В отличие от свёрточных нейросетей, которые хорошо работают только на регулярных решётках, графовые сети адаптируются к произвольной топологии, сохраняя при этом возможность учитывать локальные и глобальные зависимости в данных. Это делает их полезными в самых разных задачах:

от моделирования процессов в пористых материалах до задач биомеханики, гидродинамики и геофизики.

Дополнительным преимуществом графовых нейросетей является их способность к переносу обученной модели между задачами и геометриями. Благодаря универсальному представлению входных данных в виде графов, одну и ту же модель можно применять на различных объектах при сохранении общей топологической структуры, что открывает путь к созданию универсальных моделей-приближений, не требующих повторного обучения.

На практике применяются различные архитектуры графовых сетей: от базовых графовых свёрточных сетей до моделей с механизмами внимания и специализированных физических архитектур, в которых учитываются особенности конкретных уравнений. В последних случаях физические законы могут прямо включаться в функции агрегации или обновления состояний узлов, что позволяет сохранять интерпретируемость модели и устойчивость решений.

Однако, несмотря на явные преимущества, использование GNN в численном моделировании требует внимательного подхода. Необходимо обеспечить согласование структуры графа с физикой задачи, а также учитывать вычислительные сложности, возникающие при переходе к трёхмерным областям или при моделировании многомасштабных процессов. Тем не менее, графовые нейросети уже показали высокую эффективность в ряде прикладных задач и заслуженно рассматриваются как одно из наиболее перспективных направлений в области современных численных методов.

1.3.6. Физически информированные нейронные сети. Physics-Informed Neural Networks, PINN

Но самым распространенным методом являются физически информированные нейронные сети — Physics-Informed Neural Networks, PINN, где сама структура задачи интегрируется в процесс обучения. . В отличие от классических нейросетей, которые обучаются исключительно на данных, PINN позволяет учитывать уравнения, связывающие эти данные между собой. Это делает возможным решение не только прямых, но и обратных задач даже при ограниченной или неполной информации.

В основе метода лежит идея аппроксимации решения дифференциального уравнения с помощью нейросети, которая принимает на вход координаты и время, а на выходе возвращает значение физической величины — например, температуры, давления, напряжённости поля. Обучение осуществляется таким образом, чтобы сеть не только приближала известные значения в отдельных точках, но и минимизировала невязку уравнения, которое должно выполняться во всей области. Для этого используется функция потерь, включающая в себя как стандартную ошибку между предсказаниями и наблюдениями, так и специальные члены, отражающие соблюдение физического закона — обычно в форме дифференциального уравнения. Производные, необходимые для расчёта этих членов, вычисляются автоматически, что позволяет не использовать явную сеточную дискретизацию [1; 45; 51].

Одним из главных достоинств PINN является возможность применять его в условиях, когда наблюдаемые данные скудны, разрежены или зашумлены. Даже в отсутствие полной информации о начальных или граничных условиях, модель, обученная с учётом физики, может выдавать осмысленные, физически обоснованные предсказания [49]. Это особенно важно в тех случаях, когда традиционные численные методы становятся неустойчивыми, или когда построение сетки требует чрезмерных вычислительных ресурсов [63].

PINN находят применение во многих научных и инженерных областях, особенно там, где необходимо решать дифференциальные уравнения при ограниченной информации или высокой сложности систем. Особую ценность этот подход представляет в обратных задачах — например, когда необходимо по экспериментальным данным восстановить параметры уравнения, граничные условия или внутреннюю структуру объекта. PINN позволяет объединить решение таких задач в рамках единой модели, без этапов регуляризации и отдельного численного решателя.Недавние обзоры отмечают эффективность PINN в задачах механики и теплопереноса [1; 16].

Вот где PINN особенно полезны:

1. Электромагнетизм. Используются для решения уравнений Максвелла, моделирования распространения электромагнитных волн, взаимодействия света с наноструктурами, плазмы и дифракции, включая задачи с труднодоступными или частично заданными граничными условиями.

2. Гидродинамика и аэродинамика. Используются для моделирования несжимаемых и сжимаемых течений, в том числе при больших числах Рейнольдса. Примеры включают задачи обтекания, турбулентности, кавитации, а также потоков с переменными границами (fluid-structure interaction).

3. Тепло- и массообмен. PINN применяются для задач теплопереноса, диффузии, реактивных потоков, включая каталитические процессы, горение, фазовые переходы и конвекцию.

4. Механика сплошных сред и материаловедение. PINN применяются для моделирования напряжений, деформаций и повреждений в твёрдых телах, включая нелинейные и анизотропные материалы. Это актуально при анализе композитов, микромеханики и сложных граничных условий, особенно когда экспериментальные данные ограничены.

5. Биомедицинская инженерия. PINN применяются для моделирования кровотока, теплопереноса в тканях, распространения лекарств, электрической активности в сердце и головном мозге. Особенность — возможность встраивания анатомических данных, полученных из МРТ или КТ, прямо в физическую модель.

6. Геофизика и климатология. Используются для решения обратных задач при сейсмической томографии, гидрогеологических моделях, моделировании атмосферы и океана. Особенно ценны в задачах с неполными измерениями и большими пространственными масштабами.

7. Финансовая математика. Моделирование процессов на основе стохастических дифференциальных уравнений, например, в задачах ценообразования опционов, оценки рисков, хеджирования, особенно в условиях высокой волатильности и нехватки исторических данных.

8. Квантовая физика. PINN применяются для приближённого решения уравнения Шрёдингера, расчёта собственных состояний, моделирования туннелирования, а также в задачах квантовой химии (например, электронная плотность в молекулах).

9. Химическая кинетика и системная биология

Используются для восстановления параметров реакций, предсказания поведения сложных сетей с обратными связями и моделирования внутриклеточных процессов.

Космические и астрофизические расчёты. Моделирование гравитационных волн, динамики аккреционных дисков, распространения излучения в звёздных атмосферах, задач небесной механики и общего моделирования эволюции астрофизических систем.

Physics-Informed Neural Networks (PINNs) находят широкое применение в электродинамике, особенно при решении уравнений Максвелла в сложных условиях.

1.3.7. Решение нестационарных уравнений Максвелла

Физически информированные нейронные сети активно применяются для моделирования временной эволюции электромагнитных полей, описываемой уравнениями Максвелл [42], а. Такой подход особенно полезен в задачах, где требуется проследить поведение волн во времени, например, при их распространении в неоднородной среде или отражении от границ. В отличие от традиционных методов, таких как метод конечных разностей во временной области (FDTD), PINN не требует явной сетки по пространству и времени. Это позволяет решать задачи в непрерывной постановке, получая устойчивые и физически обоснованные результаты, особенно там, где начальные или граничные условия заданы неполно или приблизительно.

1.3.8. Моделирование электромагнитных процессов в неоднородных средах

PINN показывает высокую эффективность при моделировании взаимодействия электромагнитных волн с неоднородными средами, в которых физические параметры — например, диэлектрическая или магнитная проницаемость — резко меняются в пространстве [35]. В традиционных численных схемах такие резкие переходы требуют очень мелкой сетки и специальной обработки граничных условий. PINN же может «мягко» аппроксимировать поведение на границах различных материалов, обеспечивая согласованность решения на всём интервале. Это особенно актуально при работе с метаматериалами, наноструктурами и волноводами сложной формы [46; 47].

1.3.9. Решение обратных задач и определение параметров среды

Одно из ключевых преимуществ PINN — возможность решения обратных задач, то есть восстановление неизвестных свойств среды по данным измерений. Например, по известному распределению электромагнитного поля можно восстановить диэлектрическую проницаемость материала или определить форму внутренних неоднородностей. Такие задачи традиционно считаются плохо обусловленными и требуют сложной регуляризации. Использование PINN позволяет встроить физику напрямую в процесс обучения, что существенно упрощает решение и делает его более устойчивым к шуму и недостатку данных [2].

1.3.10. Моделирование магнитных полей и микромагнитных структур

В задачах магнитостатики PINN применяется для вычисления распределения магнитного поля и потенциала, в том числе в магнитно активных материалах со сложной геометрией. Особенно интересны приложения в микромагнетизме, где требуется высокая точность при моделировании доменных структур и переходов между состояниями на микроуровне. Метод позволяет получить непрерывное представление магнитного поля в пространстве без необходимости вручную задавать сеточную аппроксимацию или использовать специальные преобразования. Это облегчает расчёты в задачах проектирования магнитных сенсоров, систем хранения информации и других устройств.

1.3.11. Электростатические задачи и расчёт потенциалов

PINN также широко применяются для решения задач электростатики — например, для расчёта потенциала вблизи заряженных тел, в присутствии диэлектрических или проводящих границ. Такие задачи характерны для электрохимии, микроэлектроники, биофизики и энергоёмких устройств. Преимущество метода — возможность моделировать сложные граничные условия и геометрии, сохраняя при этом физическую корректность распределения потенциала. PINN способен учитывать как точечные источники, так и пространственно распределённые заряды, а также особенности материала, не прибегая к ручному построению сетки.

Тем не менее, как и любой метод, PINN имеет свои ограничения [8; 48; 65]. Прежде всего, обучение может быть длительным и требовать значительных вычислительных ресурсов, особенно в задачах с большой размерностью или сложной геометрией. Кроме того, важную роль играет балансировка компонентов функции потерь: если веса выбраны неудачно, сеть может сосредоточиться либо на данных, игнорируя уравнение, либо наоборот. Также возможно возникновение трудностей при обучении в жёстких системах или при сильной нелинейности.

Также стоит отметить, что PINN не гарантирует сходимости к корректному решению даже при формальном выполнении уравнений и граничных условий. Итог может сильно зависеть от начальной инициализации сети, её архитектуры и выбора параметров оптимизации. Это означает, что две модели, обученные на одинаковых данных, могут дать разные и не всегда физически обоснованные результаты [8].

Сложность также представляет собой выбор архитектуры нейросети. В отличие от традиционных численных методов, для которых существует устоявшаяся теория построения разностных или конечных элементов, в случае PINN нет общепринятой методики определения глубины, ширины и других характеристик сети. Выбор приходится осуществлять эмпирически, что повышает риск переобучения или, наоборот, недостаточной выразительности модели.

PINN чувствительны к масштабам переменных. Если входные и выходные величины имеют различный порядок, это может привести к числовой нестабильности и нарушению баланса при обучении. Хотя существует практика нормализации и введения адаптивных весов в функцию потерь, эти решения не всегда универсальны и требуют ручной настройки.

Серьёзное ограничение связано с тем, что PINN плохо обобщают поведение системы за пределами области, в которой производилось обучение. Если, например, геометрия задачи изменилась или параметры уравнения вышли за пределы обучающего диапазона, модель может перестать работать корректно. В таких случаях требуется полное переобучение, что делает метод менее универсальным по сравнению с классическими схемами.

Особенность PINN в том, что они не обладают локальностью. В численных методах результат в каждой точке зависит только от локального окружения, и это позволяет контролировать точность решения по сетке. В PINN же влияние

каждого обучающего узла распространяется на всё решение, что затрудняет локальный контроль ошибки и делает диагностику проблем в конкретных областях более сложной.

Дополнительная трудность возникает при попытке аппроксимировать решения с высокочастотными компонентами, резонансами или осцилляциями. Стандартные архитектуры PINN, основанные на классических функциях активации, с такими задачами справляются плохо. Для их успешного решения требуется использование специальных архитектур, например, с представлением входных данных в спектральной области [7].

Если задача содержит разрывы, градиентные скачки или сингулярности, PINN также показывает нестабильное поведение. Нейросеть, обученная на гладких функциях, склонна сглаживать такие особенности, что искажает физическую картину процесса. Кроме того, как и любые нейросетевые модели, PINN остаются трудно интерпретируемыми — сложно объяснить, почему модель приняла то или иное решение, и это снижает её доверие со стороны исследователей и инженеров.

Эти ограничения не делают PINN непригодными, но требуют осознанного и осторожного применения, особенно в задачах, где важна высокая точность, надёжность и воспроизводимость.

Несмотря на это, физически информированные нейросети сегодня считаются одним из наиболее универсальных и перспективных инструментов интеграции машинного обучения и классического математического моделирования. PINN не только расширяет границы применимости численных методов, но и даёт новые возможности для анализа систем, о которых имеется неполная информация, или которые сложно описать исключительно аналитически или численно.

Метод	Суть метода	Тип задачи	Особенности
PINN (Physics-Informed Neural Networks)	Минимизация невязки PDE и граничных условий с помощью нейросети	Прямые и обратные задачи ОДУ/ЧДУ	Автоматическое дифференцирование, сложность обучения
Deep Ritz Method (DRM)	Решение вариационной задачи через минимизацию функционала с помощью нейросети	Задачи с вариационной формой (например, Пуассон)	Подходит для задач с энергетической формулировкой
Neural Operators (DeepONet, FNO)	Аппроксимация отображений между функциями (например, начальные условия → решение)	Параметризованные PDE, многоразовое применение	Высокая скорость после обучения, независимость от сетки
Data-Driven Hybrid Models	Объединение классических PDE моделей с ML для аппроксимации коэффициентов, условий и т.п.	Обратные задачи, суррогатные модели	Гибкость, но возможна потеря интерпретируемости
Graph Neural Networks for PDEs	Решение дискретных PDE через распространение информации по графу	Сеточные задачи (необязательно регулярные)	Естественная работа с нерегулярными сетками
Latent Space Evolution via Autoencoders	Сжатие поля в латентное пространство и моделирование эволюции в нём	Сложные динамические системы	Компрессия данных, ускорение моделирования

Глава 2. Моделирование оптических трансформирующих сред на основе лучевой оптики

2.1. Формализм оптических лучей и уравнения эйконала

2.1.1. Методический вывод уравнения эйконала

Обычно при работе с уравнением эйконала ссылаются на его вывод в монографии Борна и Вольфа. Вывод этого уравнения выполнен достаточно небрежно. Для того, чтобы разобраться в этом выводе требуется определённое число имплицитных предположений. Для лучшего понимания приближения эйконала и для методических целей авторы решили повторить вывод уравнения эйконала, эксплицировав все возможные допущения. Методически предлагается следующий алгоритм вывода уравнения эйконала. Из уравнения Максвелла выводится волновое уравнение. При этом явно вводятся все условия, при которых это возможно сделать. Далее от волнового уравнения осуществляется переход к уравнению Гельмгольца. От уравнения Гельмгольца, при приложении определённых допущений, производится переход к уравнению эйконала. После разбора всех допущений и шагов, реализуется собственно переход от уравнений Максвелла к уравнению эйконала. При выводе уравнения эйконала используется несколько формализмов. В качестве первого формализма используется стандартный формализм векторного анализа. Уравнения Максвелла и уравнение эйконала записывается в виде трёхмерных векторов. После этого и для уравнений Максвелла, и для уравнения эйконала используется ковариантный 4-мерный формализм.

Одной из основ применяемой программы моделирования оптических явлений является модель эйконала [6; 28]. Данная модель достаточно известна, однако процесс вывода её несколько запутан [73; 75]. А в известной монографии Борна и Вольфа [68] вывод вообще выглядит как некий вариант физической магии (вот у нас уравнения Максвелла, немного волшебства, и у нас уравнение эйконала).

Мы использовали аналитические методы для вывода уравнения эйконала из уравнений Максвелла в среде без токов и зарядов. Процесс включает анализ дифференциальных уравнений и применение методов математического анализа. Краткий план вывода приведён на схеме 2.1.

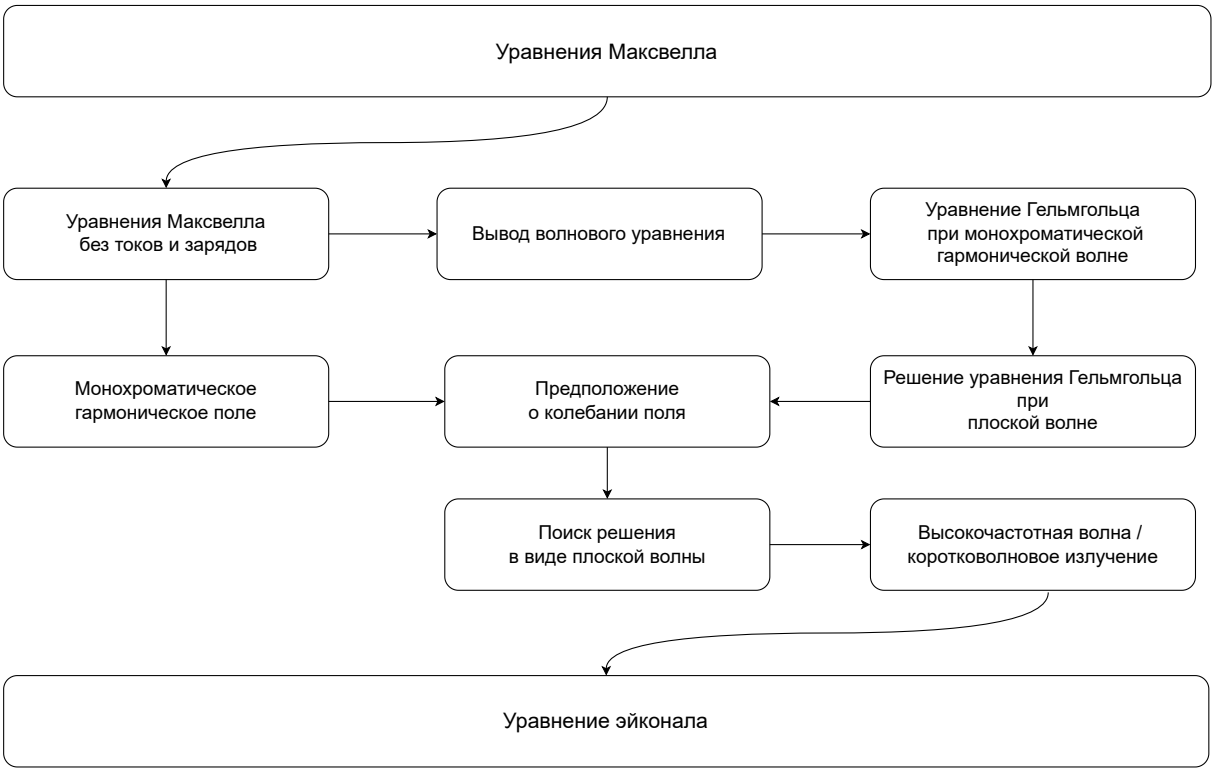


Рис. 2.1. План вывода уравнения эйконала

2.1.2. Уравнения Максвелла

Рассмотрим уравнения Максвелла в векторно-дифференциальной форме:

$$\nabla \times \mathbf{H} - \frac{1}{c}\frac{\partial \mathbf{D}}{\partial t} = \frac{4\pi}{c}\mathbf{j}, \tag{2.1}$$

$$\nabla \times \mathbf{E} + \frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} = \mathbf{0}, \tag{2.2}$$

$$\nabla \cdot \mathbf{D} = 4\pi\rho, \tag{2.3}$$

$$\nabla \cdot \mathbf{B} = 0. \tag{2.4}$$

— $\mathbf{E}(\mathbf{r}, t) = \mathbf{E}(x, y, z, t)$ — вектор напряженности электрического поля;

— $\mathbf{H}(\mathbf{r}, t) = \mathbf{H}(x, y, z, t)$ — вектор напряженности магнитного поля;

— $\mathbf{D}(\mathbf{r}, t) = \mathbf{D}(x, y, z, t)$ — вектор индукции электрического поля;

— $\mathbf{B}(\mathbf{r}, t) = \mathbf{B}(x, y, z, t)$ — вектор индукции магнитного поля;

— $\mathbf{j}(\mathbf{r}) = \mathbf{j}(x, y, z)$ — плотность внешнего электрического тока (сила тока через единицу площади);

— $\rho(\mathbf{r}) = \rho(x, y, z)$ — плотность электрического заряда;

— c — скорость света в вакууме;

— $\mathbf{r} = (x, y, z)^T$ — радиус-вектор точки, записанный в декартовых координатах.

Кратко опишем физический смысл каждого из уравнений Максвелла.

— Уравнение (2.1) означает, что электрический ток и изменение электрической индукции порождают соленоидальное магнитное поле — то есть такое поле, силовые линии которого закручиваются в вихрь вдоль вектора, указывающего направление тока.

— Уравнение (2.2) означает, что изменение во времени магнитного поля порождает электрическое поле.

— Уравнение (2.3) означает, что электрический заряд является источником электрической индукции.

— Уравнение (2.4) означает, что не существует свободных магнитных полюсов (опытным путем обнаружены только магнитные диполи, магнитные монополи науке не известны).

Справедливы также следующие соотношения, называемые *материальными уравнениями*:

$$\mathbf{j} = \sigma\mathbf{E}, \quad \mathbf{D} = \varepsilon\mathbf{E}, \quad \mathbf{B} = \mu\mathbf{H},$$

где $\sigma(\mathbf{r})$ — удельная проводимость, $\varepsilon(\mathbf{r})$ — диэлектрическая проницаемость и $\mu(\mathbf{r})$ — магнитная проницаемость. В изотропной среде ε и μ — скалярные величины, однако в общем случае являются тензорными.

Среда называется *изотропной* если ее физические свойства не зависят от направления. Термин происходит от греческих слов «изос» (ισος) — равный, одинаковый, подобный и «тропос» (τροπος) — направление, характер. В электродинамике изотропия среды связанна с одинаковостью величин $\varepsilon(\mathbf{r})$ и $\mu(\mathbf{r})$ во всех направлениях.

Магнитная проницаемость характеризует магнитные свойства среды (вещества). Если $\mu \neq 1$, то вещество называется *магнетиком*, если $\mu > 1$ — *парамагнетик*, если $\mu < 1$ — *диамагнетик*.

Далее будем рассматривать среду, не проводящую электричество, то есть $\sigma = 0$, а также свободную от токов, то есть $\mathbf{j} = \mathbf{0}$ и $\rho = 0$, то уравнения Максвелла несколько упрощаются:

$$\nabla \times \mathbf{H} - \frac{1}{c}\frac{\partial \mathbf{D}}{\partial t} = \mathbf{0}, \tag{2.5}$$

$$\nabla \times \mathbf{E} + \frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} = \mathbf{0}, \tag{2.6}$$

$$\nabla \cdot \mathbf{D} = 0, \tag{2.7}$$

$$\nabla \cdot \mathbf{B} = 0. \tag{2.8}$$

2.1.3. Волновое уравнение

2.1.4. Вывод волнового уравнения из уравнений Максвелла

Будем предполагать, что $\rho = 0$ и $\mathbf{j} = \mathbf{0}$ и рассмотрим уравнения (2.5) и (2.6):

$$\nabla \times \mathbf{H} - \frac{1}{c}\frac{\partial \mathbf{D}}{\partial t} = \mathbf{0},$$
$$\nabla \times \mathbf{E} + \frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} = \mathbf{0}.$$

Используем материальные уравнения $\mathbf{D} = \varepsilon\mathbf{E}$ и $\mathbf{B} = \mu\mathbf{H}$ и учтем зависимость диэлектрической и магнитной проницаемостей от координат: $\varepsilon = \varepsilon(x, y, z)$ и $\mu = \mu(x, y, z)$.

$$\nabla \times \mathbf{E} + \frac{1}{c}\frac{\partial}{\partial t}(\mu\mathbf{H}) = \mathbf{0} \Rightarrow \nabla \times \mathbf{E} + \frac{\mu}{c}\frac{\partial \mathbf{H}}{\partial t} = \mathbf{0} \Rightarrow \frac{1}{\mu}\nabla \times \mathbf{E} + \frac{1}{c}\frac{\partial \mathbf{H}}{\partial t} = \mathbf{0}.$$

Применим оператор ротора $\nabla\times$ к получившемуся уравнению:

$$\underbrace{\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{E}\right)}_{\text{(I)}} + \underbrace{\frac{1}{c}\nabla \times \frac{\partial \mathbf{H}}{\partial t}}_{\text{(II)}} = \mathbf{0}.$$

Рассмотрим вначале слагаемое (II) данного уравнения. Производную по времени можно вынести из под знака оператора ротора:

$$\nabla \times \frac{\partial \mathbf{H}}{\partial t} = \frac{\partial}{\partial t}(\nabla \times \mathbf{H}).$$

В силу (2.5) получим:

$$\frac{\partial}{\partial t}(\nabla \times \mathbf{H}) = \frac{\partial}{\partial t}\frac{1}{c}\frac{\partial \mathbf{D}}{\partial t} = \frac{1}{c}\frac{\partial^2 \mathbf{D}}{\partial t^2}.$$

Используем материальное уравнение $\mathbf{D} = \varepsilon\mathbf{E}$, запишем:

$$\frac{\partial^2 \mathbf{D}}{\partial t^2} = \frac{\partial^2(\varepsilon(\mathbf{r})\mathbf{E})}{\partial t^2} = \varepsilon\frac{\partial^2 \mathbf{E}}{\partial t^2} \Rightarrow \frac{1}{c}\nabla \times \frac{\partial \mathbf{H}}{\partial t} = \frac{\varepsilon}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2}.$$

Для того, чтобы упростить слагаемое (I) используем соотношение $\nabla \times f\mathbf{v} = f\nabla \times \mathbf{v} + \nabla f \times \mathbf{v}$, где $f(x, y, z)$ — скалярная функция, а $\mathbf{v}(x, y, z)$ — векторное

поле. Слагаемое (I) с помощью этого соотношения разлагается следующим образом:

$$\underbrace{\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{E}\right)}_{\text{(I)}} = \underbrace{\frac{1}{\mu}\nabla \times (\nabla \times \mathbf{E})}_{\text{(I.a)}} + \underbrace{\left(\nabla \frac{1}{\mu}, \nabla \times \mathbf{E}\right)}_{\text{(I.b)}}.$$

В свою очередь, для упрощения слагаемого (I.a) используем тождество $\nabla \times \nabla \times \mathbf{v} = \nabla(\nabla \cdot \mathbf{v}) - \nabla^2 \mathbf{v}$, где ∇^2 — оператор Лапласа.

$$\frac{1}{\mu}\nabla \times (\nabla \times \mathbf{E}) = \frac{1}{\mu}\nabla(\nabla \cdot \mathbf{E}) - \frac{1}{\mu}\nabla^2 \mathbf{E}.$$

Для упрощения выражения $\nabla(\nabla \cdot \mathbf{E})$ применим тождество $\nabla \cdot (f\,\mathbf{v}) = f\nabla \cdot \mathbf{v} + (\nabla f, \mathbf{v})$ к уравнению Максвелла (2.7), заменив индукцию $\mathbf{D}$ на напряженность с помощью материального уравнения $\mathbf{D} = \varepsilon \mathbf{E}$:

$$\nabla \cdot \mathbf{D} = \nabla \cdot (\varepsilon \mathbf{E}) = \varepsilon \nabla \cdot \mathbf{E} + (\mathbf{E}, \nabla \varepsilon) = 0 \Rightarrow \nabla \cdot \mathbf{E} = -\frac{1}{\varepsilon}(\nabla \varepsilon, \mathbf{E}) \Rightarrow \frac{1}{\mu}\nabla(\nabla \cdot \mathbf{E}) = -\frac{1}{\mu}\nabla\left(\frac{1}{\varepsilon}(\nabla \varepsilon, \mathbf{E})\right).$$

В результате слагаемое (I.a) преобразовалось к следующему виду:

$$\frac{1}{\mu}\nabla \times (\nabla \times \mathbf{E}) = -\frac{1}{\mu}\nabla\left(\frac{1}{\varepsilon}(\nabla \varepsilon, \mathbf{E})\right) - \frac{1}{\mu}\nabla^2 \mathbf{E}.$$

Комбинируя (I) и (II) запишем:

$$\underbrace{-\frac{1}{\mu}\nabla^2 \mathbf{E} - \frac{1}{\mu}\nabla\left(\frac{1}{\varepsilon}(\nabla \varepsilon, \mathbf{E})\right)}_{\text{(I.a)}} + \underbrace{\left(\nabla \frac{1}{\mu}, \nabla \times \mathbf{E}\right)}_{\text{(I.b)}} + \underbrace{\frac{\varepsilon}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2}}_{\text{(II)}} = 0,$$

$$-\nabla^2 \mathbf{E} - \nabla\left(\frac{1}{\varepsilon}(\nabla \varepsilon, \mathbf{E})\right) + \mu\left(\nabla \frac{1}{\mu}, \nabla \times \mathbf{E}\right) + \frac{\mu\varepsilon}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2} = 0,$$

$$\nabla^2 \mathbf{E} - \frac{\mu\varepsilon}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2} + \left[\nabla\left(\frac{1}{\varepsilon}(\nabla \varepsilon, \mathbf{E})\right) - \mu\left(\nabla \frac{1}{\mu}, \nabla \times \mathbf{E}\right)\right] = 0,$$

Полностью аналогичное уравнение можно получить и для вектора напряженности магнитного поля $\mathbf{H}$.

В случае изотропной среды, то есть $\varepsilon = \mu = \text{const}$ дополнительное слагаемое, взятое в квадратные скобки, обращается в ноль и мы получаем волновое уравнение:

$$\nabla^2 \mathbf{E} - \frac{\varepsilon\mu}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2} = \mathbf{0},$$
$$\nabla^2 \mathbf{H} - \frac{\varepsilon\mu}{c^2}\frac{\partial^2 \mathbf{H}}{\partial t^2} = \mathbf{0}.$$

Можно ввести величину $v = c/\sqrt{\varepsilon\mu}$ — скорость электромагнитной волны в среде.

2.1.5. Случай плоской волны

Рассмотрим волновое уравнение:

$$\nabla^2 \mathbf{U} - \frac{1}{v^2}\frac{\partial^2 \mathbf{U}}{\partial t^2} = 0.$$

Рассмотрим электромагнитную волну, которая распространяется в направлении $\mathbf{s}$, где $\mathbf{s} = (s_x, s_y, s_z)$ — некоторый единичный вектор ($\|\mathbf{s}\| = 1$) фиксированного направления. Любое решение данного уравнения, имеющее вид $\mathbf{U} = \mathbf{U}((\mathbf{r}, \mathbf{s}), t)$ представляет собой *плоскую* волну, так как в каждый момент времени вектор $\mathbf{U}$ постоянен в плоскости $(\mathbf{r}, \mathbf{s}) = -d$, где $|d|$ — расстояние от плоскости до начала координат. Выражение $(\mathbf{r}, \mathbf{s}) = -d$ фактически является нормальным уравнением плоскости, где вектор $\mathbf{s}$ выступает в роли единичного вектора нормали. Запишем в декартовых координатах:

$$s_x x + s_y y + s_z z + d = 0.$$

Можно упростить волновое уравнение путем введения новой системы координат. Так как вектор напряженности плоской волны целиком зависит только от расстояния d, то можно выбрать новую систему координат с осями $O\xi$, $O\eta$, $O\zeta$ так, что ось $O\zeta$ направлена вдоль вектора $\mathbf{s}$, а начало координат совпадает с прежней декартовой системой $Oxyz$. Тогда, координата по оси $O\zeta$ будет зависеть от прежних координат по формуле $\zeta(x, y, z) = (\mathbf{r}, \mathbf{s}) = s_x x + s_y y + s_z z$, в то время как ξ и η от прежних координат не зависят и могут быть выбраны произвольно, например так, чтобы координатная система $O\xi\eta\zeta$ была правой.

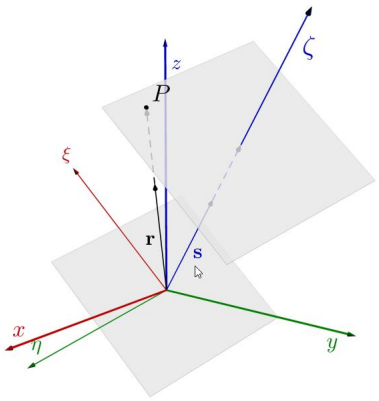


Рис. 2.2. Плоскость $(\mathbf{r}, \mathbf{s}) = \text{const}$. Новые оси координат выбираются так, чтобы вектор $\mathbf{s}$ был ортом оси $O\zeta$. Две другие оси $O\xi$ и $O\eta$ выбираются произвольно и образуют правую систему координат $O\xi\eta\zeta$

Замена дифференциальных операторов осуществляется с помощью матрицы Якоби следующим образом:

$$\begin{bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \xi}{\partial x} & \dfrac{\partial \eta}{\partial x} & \dfrac{\partial \zeta}{\partial x} \\ \dfrac{\partial \xi}{\partial y} & \dfrac{\partial \eta}{\partial y} & \dfrac{\partial \zeta}{\partial y} \\ \dfrac{\partial \xi}{\partial z} & \dfrac{\partial \eta}{\partial z} & \dfrac{\partial \zeta}{\partial z} \end{bmatrix} \begin{bmatrix} \dfrac{\partial}{\partial \xi} \\ \dfrac{\partial}{\partial \eta} \\ \dfrac{\partial}{\partial \zeta} \end{bmatrix} \quad \left(\frac{\partial(\xi, \eta, \zeta)}{\partial(x, y, z)} \right)^T = \begin{bmatrix} \dfrac{\partial \xi}{\partial x} & \dfrac{\partial \eta}{\partial x} & \dfrac{\partial \zeta}{\partial x} \\ \dfrac{\partial \xi}{\partial y} & \dfrac{\partial \eta}{\partial y} & \dfrac{\partial \zeta}{\partial y} \\ \dfrac{\partial \xi}{\partial z} & \dfrac{\partial \eta}{\partial z} & \dfrac{\partial \zeta}{\partial z} \end{bmatrix}$$

Так как $\xi = \text{const}$ и $\eta = \text{const}$, а $\zeta = (\mathbf{r}, \mathbf{s})$, то:

$$\begin{bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & s_x \\ 0 & 0 & s_y \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} \dfrac{\partial}{\partial \xi} \\ \dfrac{\partial}{\partial \eta} \\ \dfrac{\partial}{\partial \zeta} \end{bmatrix} = \begin{bmatrix} s_x \dfrac{\partial}{\partial \zeta} \\ s_y \dfrac{\partial}{\partial \zeta} \\ s_z \dfrac{\partial}{\partial \zeta} \end{bmatrix} \Longrightarrow \begin{aligned} \frac{\partial}{\partial x} &= s_x \frac{\partial}{\partial \zeta}, \\ \frac{\partial}{\partial y} &= s_y \frac{\partial}{\partial \zeta}, \\ \frac{\partial}{\partial z} &= s_z \frac{\partial}{\partial \zeta}. \end{aligned}$$

Оператор Лапласа после замены координат преобразуется к следующему виду:

$$\nabla^2 \mathbf{U} = s_x^2 \frac{\partial^2 \mathbf{U}}{\partial \zeta^2} + s_y^2 \frac{\partial^2 \mathbf{U}}{\partial \zeta^2} + s_z^2 \frac{\partial^2 \mathbf{U}}{\partial \zeta^2} = (s_x^2 + s_y^2 + s_z^2) \frac{\partial^2 \mathbf{U}}{\partial \zeta^2} = \frac{\partial^2 \mathbf{U}}{\partial \zeta^2}$$

Волновое уравнение упрощается:

$$\frac{\partial^2 \mathbf{U}}{\partial \zeta^2} - \frac{1}{v^2}\frac{\partial^2 \mathbf{U}}{\partial t^2} = 0.$$

Проведём еще одну замену $p = \zeta - vt$ и $q = \zeta + vt$, что приведет к следующему преобразованию дифференциальных операторов:

$$\begin{bmatrix} \frac{\partial}{\partial \zeta} \\ \frac{\partial}{\partial t} \end{bmatrix} = \left(\frac{\partial(\zeta, t)}{\partial(p, q)}\right)^T \begin{bmatrix} \frac{\partial}{\partial p} \\ \frac{\partial}{\partial q} \end{bmatrix} = \begin{bmatrix} \frac{\partial p}{\partial \zeta} & \frac{\partial q}{\partial \zeta} \\ \frac{\partial p}{\partial t} & \frac{\partial q}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial p} \\ \frac{\partial}{\partial q} \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 \\ -v & v \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial p} \\ \frac{\partial}{\partial q} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial p} + \frac{\partial}{\partial q} \\ -v\frac{\partial}{\partial p} + v\frac{\partial}{\partial q} \end{bmatrix} \Rightarrow$$

$$\Rightarrow \begin{array}{l} \frac{\partial}{\partial \zeta} = \frac{\partial}{\partial p} + \frac{\partial}{\partial q} \\ \frac{\partial}{\partial t} = -v\frac{\partial}{\partial p} + v\frac{\partial}{\partial q}. \end{array}$$

Вторые производные выражаются через новые переменные следующим образом:

$$\frac{\partial^2}{\partial \zeta^2} = \frac{\partial^2}{\partial p^2} + 2\frac{\partial}{\partial p}\frac{\partial}{\partial q} + \frac{\partial^2}{\partial q^2},$$
$$\frac{\partial^2}{\partial t^2} = v^2\left(\frac{\partial^2}{\partial p^2} - 2\frac{\partial}{\partial p}\frac{\partial}{\partial q} + \frac{\partial^2}{\partial q^2}\right).$$

При подстановке операторов в волновое равнение, оно упрощается следующим образом:

$$\frac{\partial^2 \mathbf{U}}{\partial \zeta^2} - \frac{1}{v^2}\frac{\partial^2 \mathbf{U}}{\partial t^2} = \frac{\partial^2 \mathbf{U}}{\partial p^2} + 2\frac{\partial^2 \mathbf{U}}{\partial p \partial q} + \frac{\partial^2 \mathbf{U}}{\partial q^2} - \frac{1}{v^2}v^2\left(\frac{\partial^2 \mathbf{U}}{\partial p^2} - 2\frac{\partial^2 \mathbf{U}}{\partial p \partial q} + \frac{\partial^2 \mathbf{U}}{\partial q^2}\right) =$$

$$= 4\frac{\partial^2 \mathbf{U}}{\partial p \partial q} = 0 \Rightarrow \boxed{\frac{\partial^2 \mathbf{U}}{\partial p \partial q} = 0}$$

Общим решением преобразованного волнового уравнения служит функция

$$\mathbf{U} = \mathbf{U}_1(p) + \mathbf{U}_2(q) = \mathbf{U}_1\big((\mathbf{r}, \mathbf{s}) - vt\big) + \mathbf{U}_2\big((\mathbf{r}, \mathbf{s}) + vt\big).$$

Другой подход к решению использует разделение переменных. Будем искать решение в комплексном виде

$$\mathbf{U}(\mathbf{r}, t) = \mathbf{U}_0(\mathbf{r})e^{-i\omega t}.$$

При подстановке в волновое уравнение получим:

$$\frac{\partial^2 \mathbf{U}}{\partial t^2} = -\omega^2 e^{-i\omega t}\mathbf{U}_0(\mathbf{r}), \quad \nabla^2 \mathbf{U} = e^{-i\omega t}\nabla^2 \mathbf{U}_0(\mathbf{r}).$$

$$\nabla^2 \mathbf{U} - \frac{1}{v^2}\frac{\partial^2 \mathbf{U}}{\partial t^2} = 0 \Longrightarrow \nabla^2 \mathbf{U}_0 + \frac{\omega^2}{v^2}\mathbf{U}_0 = 0.$$

Введем некоторые скалярные величины: *волновое число* $k = \omega/v$, $k_0 = \omega/c$, *волновой вектор* $\mathbf{k} = k\mathbf{s}$. Напомним, что c — скорость света в вакууме, v — скорость электромагнитной волны в среде, $n = \sqrt{\varepsilon\mu}$ — коэффициент преломления среды, $\mathbf{s}$ — направление распространения волны. Скорости v и c связаны соотношениями $v = c/\sqrt{\varepsilon\mu} = c/n$, поэтому волновое число можно также записать в виде $k = \omega/v = \omega\sqrt{\varepsilon\mu}/c = k_0 n$. Теперь уравнение для $\mathbf{U}_0$ можно переписать в виде:

$$(\nabla^2 + k^2)\mathbf{U}_0 = 0.$$

Данное уравнение носит название *уравнения Гельмгольца* (однородное уравнение Гельмгольца). В общем случае его решение можно выразить в специальных функциях, но в случае плоской волны общее решение можно записать в следующем виде:

$$\mathbf{U}_0(\mathbf{r}) = \mathbf{u}_0(\mathbf{r})e^{ik(\mathbf{s},\mathbf{r})} = \mathbf{u}_0(\mathbf{r})e^{ik_0 n(\mathbf{s},\mathbf{r})}.$$

2.1.6. Вывод уравнения эйконала

Теорема 1 *Будем рассматривать монохроматическую гармоническую волну, векторы напряженности которой выражаются в следующем виде:*

$$\mathbf{E}(\mathbf{r}, t) = \mathbf{E}_0(\mathbf{r})e^{-i\omega t},$$
$$\mathbf{H}(\mathbf{r}, t) = \mathbf{H}_0(\mathbf{r})e^{-i\omega t},$$

где $\mathbf{r} = (x, y, z)^T$ *— радиус вектор точки пространства в декартовой системе координат,* ω *— циклическая частота,* $k_0 = \omega/c = 2\pi/\lambda_0$*, где* λ_0 *— длина волны в вакууме, а также предположим, что волна высокочастотная, то есть циклическая частота* ω *велика и, следовательно, велика и величина* k_0*. Тогда, уравнение Гельмгольца сводится к уравнению* эйконала*, которое имеет следующий вид:*

$$\|\nabla u\|^2 = n^2(\mathbf{r}),$$

Подставим выражения для монохроматической волны в уравнения Максвелла. Последовательно вычислим все дифференциальные операторы:

$$\nabla \times \mathbf{H} = \nabla \times \left(\mathbf{H}_0 e^{-i\omega t}\right) = e^{-i\omega t}\nabla \times \mathbf{H}_0,$$
$$\nabla \times \mathbf{E} = \nabla \times \left(\mathbf{E}_0 e^{-i\omega t}\right) = e^{-i\omega t}\nabla \times \mathbf{E}_0.$$

Используя материальные уравнения $\mathbf{D} = \varepsilon\mathbf{E}$ и $\mathbf{B} = \mu\mathbf{H}$ заменим везде $\mathbf{D}$ и $\mathbf{B}$ через $\mathbf{E}$ и $\mathbf{H}$, учитывая, что $\varepsilon(\mathbf{r}) = \varepsilon(x, y, z)$ и $\mu(\mathbf{r}) = \mu(x, y, z)$.

$$\nabla \cdot \mathbf{D} = \nabla \cdot (\varepsilon(x, y, z)\mathbf{E}) = e^{-i\omega t}\nabla \cdot (\varepsilon\mathbf{E}_0),$$
$$\nabla \cdot \mathbf{B} = \nabla \cdot (\mu(x, y, z)\mathbf{H}) = e^{-i\omega t}\nabla \cdot (\mu\mathbf{H}_0).$$

Заменим $\mathbf{D}$ и $\mathbf{B}$ также в выражениях для производных, учитывая, что ε и μ не зависят от времени, также как и $\mathbf{E}_0$ с $\mathbf{H}_0$ из формул $\mathbf{E}(x, y, z, t) = \mathbf{E}_0(x, y, z)e^{-i\omega t}$, $\mathbf{H}(x, y, z, t) = \mathbf{H}_0(x, y, z)e^{-i\omega t}$.

$$\frac{\partial \mathbf{D}}{\partial t} = \frac{\partial}{\partial t}\left(\varepsilon\mathbf{E}_0 e^{-i\omega t}\right) = \varepsilon(x, y, z)\mathbf{E}_0(x, y, z)\frac{\partial e^{-i\omega t}}{\partial t} = -i\varepsilon\omega\mathbf{E}_0 e^{-i\omega t},$$
$$\frac{\partial \mathbf{B}}{\partial t} = \frac{\partial}{\partial t}\left(\mu\mathbf{H}_0 e^{-i\omega t}\right) = \mu(x, y, z)\mathbf{H}_0(x, y, z)\frac{\partial e^{-i\omega t}}{\partial t} = -i\mu\omega\mathbf{H}_0 e^{-i\omega t}.$$

Подставим полученные выражения в уравнение (2.5):

$$\nabla \times \mathbf{H} - \frac{1}{c}\frac{\partial \mathbf{D}}{\partial t} = 0 \Rightarrow \cancel{e^{-i\omega t}}\nabla \times \mathbf{H}_0 + i\varepsilon\frac{\omega}{c}\mathbf{E}_0\cancel{e^{-i\omega t}} = 0 \Rightarrow \boxed{\nabla \times \mathbf{H}_0 + i\varepsilon k_0 \mathbf{E}_0 = 0},$$

затем в уравнение (2.6):

$$\nabla \times \mathbf{E} + \frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} = 0 \Rightarrow \cancel{e^{-i\omega t}}\nabla \times \mathbf{E}_0 - i\varepsilon\frac{\omega}{c}\mathbf{H}_0\cancel{e^{-i\omega t}} = 0 \Rightarrow \boxed{\nabla \times \mathbf{E}_0 - i\varepsilon k_0 \mathbf{H}_0 = 0},$$

в уравнение (2.7):

$$\nabla \cdot \mathbf{D} = 0 \Rightarrow e^{-i\omega t}\nabla \cdot (\varepsilon \mathbf{E}_0) = 0 \Rightarrow \boxed{\nabla \cdot (\varepsilon \mathbf{E}_0) = 0},$$

и, наконец, в уравнение (2.8):

$$\nabla \cdot \mathbf{B} = 0 \Rightarrow e^{-i\omega t}\nabla \cdot (\mu \mathbf{H}_0) = 0 \Rightarrow \boxed{\nabla \cdot (\mu \mathbf{H}_0) = 0}$$

В результате система уравнений (2.5)–(2.8) принимает следующий упрощенный вид:

$$\begin{cases} \nabla \times \mathbf{H}_0 + i\varepsilon k_0 \mathbf{E}_0 &= 0, \\ \nabla \times \mathbf{E}_0 - i\mu k_0 \mathbf{H}_0 &= 0, \\ \nabla \cdot (\varepsilon \mathbf{E}_0) &= 0, \\ \nabla \cdot (\mu \mathbf{H}_0) &= 0. \end{cases} \tag{2.9}$$

Сделаем очередное упрощение, предположив, что

$$\mathbf{E}_0(x, y, z) = \mathbf{e}(x, y, z)\exp(ik_0 u(x, y, z)) = \mathbf{e}(\mathbf{r})\exp(ik_0 u(\mathbf{r})),$$
$$\mathbf{H}_0(x, y, z) = \mathbf{h}(x, y, z)\exp(ik_0 u(x, y, z)) = \mathbf{h}(\mathbf{r})\exp(ik_0 u(\mathbf{r})).$$

где $u(x, y, z) = u(\mathbf{r})$ — скалярная вещественная функция, называемая *оптическим путем*, а $\mathbf{e}$ и $\mathbf{h}$ — векторные функции положения. Вновь вычислим дифференциальные операторы, на этот раз от $\mathbf{E}_0$ и $\mathbf{H}_0$, используя формулы (С.2):

$$\nabla \times \mathbf{H}_0 = \nabla \times \left(e^{ik_0 u(\mathbf{r})}\mathbf{h}(\mathbf{r})\right) = e^{ik_0 u(\mathbf{r})}\nabla \times \mathbf{h} + \nabla\left(e^{ik_0 u(\mathbf{r})}\right) \times \mathbf{h}.$$

Градиент от функции $e^{ik_0 u(\mathbf{r})}$ вычисляется следующим образом:

$$\nabla\left(e^{ik_0u(\mathbf{r})}\right) = \left(\frac{\partial e^{ik_0u(\mathbf{r})}}{\partial x}, \frac{\partial e^{ik_0u(\mathbf{r})}}{\partial y}, \frac{\partial e^{ik_0u(\mathbf{r})}}{\partial z}\right) =$$
$$= ik_0e^{ik_0u(\mathbf{r})}\left(\frac{\partial u(x,y,z)}{\partial x}, \frac{\partial u(x,y,z)}{\partial y}, \frac{\partial u(x,y,z)}{\partial z}\right) = ik_0e^{ik_0u(\mathbf{r})}\nabla u(x,y,z).$$

В результате слагаемое $\nabla \times \mathbf{H}_0$ первого уравнения системы (2.9) принимает вид:

$$\nabla \times \mathbf{H}_0 = (\nabla \times \mathbf{h} + ik_0\nabla u \times \mathbf{h})e^{ik_0u(\mathbf{r})}. \tag{2.10}$$

Полностью аналогично получаем выражение для $\nabla\times\mathbf{E}_0$ во втором уравнении системы (2.9):

$$\nabla \times \mathbf{E}_0 = (\nabla \times \mathbf{e} + ik_0\nabla u \times \mathbf{e})e^{ik_0u(\mathbf{r})}. \tag{2.11}$$

Вычисление дивергенции несколько сложнее, потому что формулу (С.2) придется применять дважды. Первый раз используем ее чтобы расписать выражение $\nabla \cdot \varepsilon\mathbf{E}_0$:

$$\nabla \cdot \varepsilon\mathbf{E}_0 = \varepsilon(\mathbf{r})\nabla \cdot \mathbf{E}_0 + (\nabla\varepsilon, \mathbf{E}_0).$$

Далее используем ее же для вычисления $\nabla \cdot \mathbf{E}_0$, где вместо $\mathbf{E}_0$ подставим выражение $\mathbf{E}_0 = \mathbf{e}(\mathbf{r})\exp(ik_0u(\mathbf{r}))$:

$$\nabla \cdot \mathbf{E}_0 = \nabla \cdot \left[\mathbf{e}(\mathbf{r})e^{ik_0u(\mathbf{r})}\right] = e^{ik_0u(\mathbf{r})}\nabla \cdot \mathbf{e} + \left(\nabla\left(e^{ik_0u(\mathbf{r})}\right), \mathbf{e}\right) =$$
$$= e^{ik_0u(\mathbf{r})}\nabla \cdot \mathbf{e} + ik_0e^{ik_0u(\mathbf{r})}(\nabla u, \mathbf{e}) = (\nabla \cdot \mathbf{e} + ik_0(\nabla u, \mathbf{e}))e^{ik_0u(\mathbf{r})},$$

$$(\nabla\varepsilon, \mathbf{E}_0) = (\nabla\varepsilon, \mathbf{e})e^{ik_0u(\mathbf{r})}.$$

В итоге третье уравнение системы (2.9) принимает вид:

$$\nabla \cdot (\varepsilon(\mathbf{r})\mathbf{E}_0(\mathbf{r})) = \left[\varepsilon(\mathbf{r})\nabla \cdot \mathbf{e}(\mathbf{r}) + ik_0\varepsilon(\mathbf{r})(\nabla u(\mathbf{r}), \mathbf{e}(\mathbf{r})) + (\nabla\varepsilon(\mathbf{r}), \mathbf{e}(\mathbf{r}))\right]e^{ik_0u(\mathbf{r})}.$$

Полностью аналогично получаем выражение для напряженности магнитного поля то есть четвертого уравнения системы (2.9):

$$\nabla \cdot (\mu(\mathbf{r})\mathbf{H}_0(\mathbf{r})) = \left[\mu(\mathbf{r})\nabla \cdot \mathbf{h} + (\nabla\mu(\mathbf{r}), \mathbf{h}) + ik_0\mu(\mathbf{r})(\nabla u(\mathbf{r}), \mathbf{h})\right]e^{ik_0u(\mathbf{r})}.$$

После подстановки в уравнения Максвелла, получим:

$$\nabla\times\mathbf{H}_0+i\varepsilon k_0\mathbf{E}_0=\mathbf{0}\Rightarrow\underbrace{\nabla\times\mathbf{h}+ik_0\nabla u\times\mathbf{h}+i\varepsilon k_0\mathbf{e}=\mathbf{0}}_{(2.10)}\Rightarrow\nabla u\times\mathbf{h}+\varepsilon\mathbf{e}=-\frac{1}{ik_0}\nabla\times\mathbf{h}.$$

$$\nabla\times\mathbf{E}_0-i\mu k_0\mathbf{H}_0=\mathbf{0}\Rightarrow\underbrace{\nabla\times\mathbf{e}+ik_0\nabla u\times\mathbf{e}-i\mu k_0\mathbf{h}=\mathbf{0}}_{(2.11)}\Rightarrow\nabla u\times\mathbf{e}-\mu\mathbf{h}=-\frac{1}{ik_0}\nabla\times\mathbf{e}.$$

$$\nabla\cdot(\varepsilon\mathbf{E}_0)=0\Rightarrow\varepsilon\nabla\cdot\mathbf{e}+ik_0\varepsilon(\nabla u,\mathbf{e})+(\nabla\varepsilon,\mathbf{e})=0$$

$$ik_0\varepsilon(\nabla u,\mathbf{e})=-(\nabla\varepsilon,\mathbf{e})-\varepsilon\nabla\cdot\mathbf{e}=0\Rightarrow(\nabla u,\mathbf{e})=-\frac{1}{ik_0}\left[\left(\frac{1}{\varepsilon}\nabla\varepsilon,\mathbf{e}\right)+\nabla\cdot\mathbf{e}\right]$$

Так как

$$\nabla(\ln\varepsilon)=\left(\frac{\partial\ln\varepsilon}{\partial x},\frac{\partial\ln\varepsilon}{\partial y},\frac{\partial\ln\varepsilon}{\partial z}\right)=\frac{1}{\varepsilon}\left(\frac{\partial\varepsilon}{\partial x},\frac{\partial\varepsilon}{\partial y},\frac{\partial\varepsilon}{\partial z}\right)=\frac{1}{\varepsilon}\nabla\varepsilon,$$

$$(\nabla u,\mathbf{e})=-\frac{1}{ik_0}((\nabla(\ln\varepsilon),\mathbf{e})+\nabla\cdot\mathbf{e}).$$

Полностью аналогично проводятся вычисления для магнитного поля, в результате получаем четвертое уравнение:

$$(\nabla u,\mathbf{h})=-\frac{1}{ik_0}((\nabla(\ln\mu),\mathbf{h})+\nabla\cdot\mathbf{h}).$$

$$\begin{cases}\nabla u\times\mathbf{h}+\varepsilon\mathbf{e} &= -\dfrac{1}{ik_0}\nabla\times\mathbf{h},\\ \nabla u\times\mathbf{e}-\mu\mathbf{h} &= -\dfrac{1}{ik_0}\nabla\times\mathbf{e},\\ (\nabla u,\mathbf{e}) &= -\dfrac{1}{ik_0}\Big((\nabla(\ln\varepsilon),\mathbf{e})+\nabla\cdot\mathbf{e}\Big),\\ (\nabla u,\mathbf{h}) &= -\dfrac{1}{ik_0}\Big((\nabla(\ln\mu),\mathbf{h})+\nabla\cdot\mathbf{h}\Big).\end{cases}\tag{2.12}$$

Третье и четвертое уравнения из данной системы следуют из первых двух. Это можно доказать умножив скалярно первые два уравнения на ∇u и использовать тот факт, что результат векторного произведения ортогонален обоим его сомножителям:

$$(\nabla u, \nabla u \times \mathbf{h}) + \varepsilon(\nabla u, \mathbf{e}) = 0 \Rightarrow (\nabla u, \mathbf{e}) = 0.$$
$$\underbrace{\qquad\qquad}_{=0}$$

Будем рассматривать только первые два уравнения. Выразим $\mathbf{h}$ из второго уравнения через u и $\mathbf{e}$ и подставим в первое:

$$\mathbf{h} = \frac{1}{\mu}\nabla u \times \mathbf{e} \Rightarrow \nabla u \times \left(\frac{1}{\mu}\nabla u \times \mathbf{e}\right) + \varepsilon\mathbf{e} = \mathbf{0} \Rightarrow \nabla u \times \nabla u \times \mathbf{e} + \varepsilon\mu\mathbf{e} = 0.$$

Для векторного произведения справедливо тождество $\mathbf{a} \times \mathbf{b} \times \mathbf{c} = \mathbf{b}(\mathbf{a}, \mathbf{c}) - \mathbf{c}(\mathbf{a}, \mathbf{b})$ из которого следует

$$\nabla u \times \nabla u \times \mathbf{e} = \nabla u(\nabla u, \mathbf{e}) - \mathbf{e}(\nabla u, \nabla u) = \nabla u(\nabla u, \mathbf{e}) - \mathbf{e}\|\nabla u\|^2$$

$$\nabla u(\nabla u, \mathbf{e}) - \mathbf{e}\|\nabla u\|^2 + \varepsilon\mu\mathbf{e} = 0.$$

Из третьего уравнения системы (2.12) следует, что $(\nabla u, \mathbf{e})$, поэтому

$$-\mathbf{e}\|\nabla u\|^2 + \varepsilon\mu\mathbf{e} = 0 \Rightarrow \mathbf{e}\|\nabla u\|^2 = \varepsilon\mu\mathbf{e},$$

Приравнивая коэффициенты перед вектором $\mathbf{e}$ и учитывая, что $n(\mathbf{r}) = \sqrt{\varepsilon(\mathbf{r})\mu(\mathbf{r})}$ запишем уравнение:

$$\|\nabla u\|^2 = n^2(\mathbf{r}), \tag{2.13}$$

которое и есть *уравнение эйконала*. ■

Функция $u(\mathbf{r}) = u(x, y, z)$ также называется *эйконалом*, а поверхности $u(x, y, z) = \text{const}$ — *геометрическими волновыми фронтами*. В компонентном виде в декартовых координатах уравнение (2.13) принимает вид:

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial z}\right)^2 = \varepsilon(x, y, z)\mu(x, y, z) = n^2(x, y, z),$$

$$\|\nabla u\|^2 = (\nabla u, \nabla u) = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial z}\right)^2.$$

2.1.7. Вывод эйконала в ковариантной форме

Продемонстрируем вывод уравнения эйконала в тензорном формализме.

2.1.8. Векторные операторы в ковариантном виде

Векторные операторы в ковариантном виде:

- $\nabla_{\vec{V}}$ — ковариантная производная по векторному полю $\vec{V}$;
- $\vec{e}_i = \frac{\partial}{\partial x^i}$ — координатный базис, $\nabla\, \vec{e}_i = \nabla_i$;
- $\varepsilon_{ijk} = \varepsilon^{ijk}$ — символы Леви-Чивиты;
- $e_{ijk} = \sqrt{|g|}\, \varepsilon_{ijk}, \quad e^{ijk} = \frac{1}{\sqrt{|g|}} \varepsilon^{ijk}$ — альтернирующие тензоры (тензоры Леви-Чивиты);
- $\boldsymbol{\nabla} f = \nabla_i f = \partial_i f, \quad f$ — скалярное поле;
- $\boldsymbol{\nabla} \cdot f = \nabla_i V^i = \frac{1}{\sqrt{g}} \partial_i (\sqrt{g} V^i)$;
- $\vec{x} = (x^1, x^2, x^3)^T$ — контравариантный вектор;
- $\boldsymbol{\nabla} \times \vec{V} = e^{ijk} \nabla_j V_k = e^{ijk} \partial_j V_k = \frac{1}{\sqrt{g}} \varepsilon^{ijk} \partial_j V_k$.

2.1.9. Уравнения Максвелла без токов и зарядов

Напряжённость электрического и магнитного полей в виде ковектора (обозначается над буквой, возможно изменение обозначений), а D и B — векторы:

$$\tilde{E} = (E_1, E_2, E_3), \quad \tilde{H} = (H_1, H_2, H_3), \quad \vec{D} = (D^1, D^2, D^3)^T, \quad \vec{B} = (B^1, B^2, B^3)^T.$$

Материальные уравнения:

$$B^i = \mu^{ij} H_j, \quad D^i = \varepsilon^{ij} E_j.$$

Векторные, ковекторные поля:

$$\tilde{E}(\vec{x}, t), \quad \tilde{H}(\vec{x}, t), \quad \vec{D}(\tilde{x}, t), \quad \vec{B}(\tilde{x}, t).$$

Тензорные поля:

$$\mu^{ij}(\vec{x}), \varepsilon^{ij}(\vec{x}).$$

2.1.10. Векторно-дифференциальная форма записи уравнений Максвелла

$$\begin{cases} \nabla \times \vec{H} - \dfrac{1}{c}\dfrac{\partial \vec{D}}{\partial t} = 0, \\ \nabla \times \vec{E} + \dfrac{1}{c}\dfrac{\partial \vec{B}}{\partial t} = 0, \\ \nabla \cdot \vec{D} = 0, \\ \nabla \cdot \vec{B} = 0. \end{cases}$$

$$\begin{cases} \dfrac{1}{\sqrt{g}}\varepsilon^{ijk}\partial_j E_k + \dfrac{1}{c}\dfrac{\mathrm{d}B^i}{\mathrm{d}t} = 0, \\ \dfrac{1}{\sqrt{g}}\varepsilon^{ijk}\mathrm{d}_j H_k - \dfrac{1}{c}\dfrac{\mathrm{d}D^i}{\mathrm{d}t} = 0, \\ \dfrac{1}{\sqrt{g}}\partial_i(\sqrt{g}D^i) = 0, \\ \dfrac{1}{\sqrt{g}}\partial_i(\sqrt{g}B^i) = 0. \end{cases}$$

2.1.11. Монохроматическая гармоническая волна

Сформулируем и докажем следующую теорему о ковариантном виде уравнения эйконала.

Теорема 2 *Для монохроматической волны уравнение эйконала в ковариантной форме принимает следующий вид:*

$$g^{ij}\partial_i u \partial_j u = \varepsilon^{ij}\mu_{ij},$$

где ε^{ij} — тензор диэлектрической проницаемости среды, μ_{ij} — тензор магнитной проницаемости среды, g^{ij} — компоненты метрического тензора евклидова пространства с поднятыми верхними индексами.

Будем рассматривать монохроматическую гармоническую волну.

$$E_k = E_{0k}e^{-i\omega t}, H_k = H_{0k}e^{-i\omega t}, D^k = \varepsilon^{kl}E_l = \varepsilon^{kl}E_{0l}e^{-i\omega t},$$
$$B^k = \mu^{kl}H_l = \mu^{kl}H_{0k}e^{-i\omega t},$$
$$\frac{\mathrm{d}D^i}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\varepsilon^{ij}E_{0j}e^{-i\omega t}\right) = -i\omega\varepsilon^{ij}E_{0j},$$
$$\frac{\mathrm{d}B^i}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\mu^{ij}H_{0j}e^{-i\omega t}\right) = -i\omega\mu^{ij}H_{0j},$$
$$\partial_i(\sqrt{g}D^i) = \partial_i(\sqrt{g}\varepsilon^{ij}E_{0j}e^{-i\omega t}) = e^{-i\omega t}\partial_i(\sqrt{g}\varepsilon^{ij}E_{0j}),$$
$$\partial_i(\sqrt{g}B^i) = \partial_i(\sqrt{g}\mu^{ij}H_{0j}e^{-i\omega t}) = e^{-i\omega t}\partial_i(\sqrt{g}\mu^{ij}H_{0j}).$$

Формулы:

$$\frac{1}{\sqrt{g}}\varepsilon^{ijk}\partial_j E_{0k} - ik_0\mu^{ij}H_{0j} = 0, \tag{2.14}$$
$$\frac{1}{\sqrt{g}}\varepsilon^{ijk}\partial_j H_{0k} + ik_0\varepsilon^{ij}E_{0j} = 0, \tag{2.15}$$
$$\partial_i(\sqrt{g}\varepsilon^{ij}E_{0j}) = 0, \tag{2.16}$$
$$\partial_i(\sqrt{g}\mu^{ij}H_{0j}) = 0. \tag{2.17}$$

Предположение №2: $E_{0k} = e_k e^{ik_0 u(\vec{x})}$, $H_{0k} = h_k e^{ik_0 u(\vec{x})}$, где $u(\vec{x})$ — эйконал.

$$\partial_j E_{0k} = (\partial_j e_k)e^{ik_0 u} + e_k e^{ik_0 u} ik_0 \partial_j u = (\partial_j e_k + ik_0 e_k \partial_j u)e^{ik_0 u},$$
$$\partial_j H_{0k} = (\partial_j h_k)e^{ik_0 u} + h_k e^{ik_0 u} ik_0 \partial_j u = (\partial_j h_k + ik_0 h_k \partial_j u)e^{ik_0 u}.$$

Из уравнения (2.16):

$$\partial_i(\sqrt{g}\varepsilon^{ij}e_j e^{ik_0 u}) = \frac{\partial\sqrt{g}}{\partial x^i}\varepsilon^{ij}e_j e^{ik_0 u} + \sqrt{g}\frac{\partial\varepsilon^{ij}}{\partial x^i}e_j e^{ik_0 u} + \sqrt{g}\varepsilon^{ij}\frac{\partial e_j}{\partial x^i}e^{ik_0 u} + \sqrt{g}\varepsilon^{ij}e_j ik_0 e^{ik_0 u}\frac{\partial u}{\partial x^i} =$$
$$= (\partial_i\sqrt{g}\varepsilon^{ij}e_j + \sqrt{g}\partial_i\varepsilon^{ij}e_j + \sqrt{g}\varepsilon^{ij}\partial_i e_j + ik_0\sqrt{g}\varepsilon^{ij}e_j\partial_i u)e^{ik_0 u} = 0.$$

$$\partial_i\sqrt{g}\varepsilon^{ij}e_j + \sqrt{g}\partial_i\varepsilon^{ij}e_j + \sqrt{g}\varepsilon^{ij}\partial_i e_j + ik_0\sqrt{g}\varepsilon^{ij}e_j\partial_i u = 0$$
$$\sqrt{g}\varepsilon^{ij}e_j\partial_i u = \frac{-1}{ik_0}(\partial_i\sqrt{g}\varepsilon^{ij}e_j + \sqrt{g}\partial_i\varepsilon^{ij}e_j + \sqrt{g}\varepsilon^{ij}\partial_i e_j).$$

Аналогично из уравнения (2.17):

$$\sqrt{g}\mu^{ij}h_j\partial_i u = -\frac{1}{ik_0}(\partial_i\sqrt{g}\mu^{ij}h_j + \sqrt{g}\partial_i\mu^{ij} + \sqrt{g}\mu^{ij}\partial_i h_j)$$

Из условия того, что величина λ мала и ω — велика, следует k_0 — велико, а $\frac{1}{k_0}$ — мало. Как следствие:

$$\begin{cases}\sqrt{g}\varepsilon^{ij}e_j\partial_i u = 0,\\ \sqrt{g}\mu^{ij}h_j\partial_i u = 0,\end{cases}$$

$$\begin{cases}\varepsilon^{ij}e_j\partial_i u = 0,\\ \mu^{ij}h_j\partial_i u = 0.\end{cases}$$

Первые два уравнения (2.14) и (2.15) преобразуется проще:

$$\frac{1}{\sqrt{g}}\varepsilon^{ijk}(\partial_j e_k + ik_0 e_k\partial_j u) - ik_0\mu^{ij}h_j = 0$$
$$\Rightarrow$$
$$-\frac{1}{\sqrt{g}}\varepsilon^{ijk}e_k\partial_j u + \mu^{ij}h_0 = \frac{1}{ik_0}\frac{1}{\sqrt{g}}\varepsilon^{ijk}\partial_j e_k$$

$$\frac{1}{\sqrt{g}}\varepsilon^{ijk}\partial_j h_k + \frac{1}{\sqrt{g}}\varepsilon^{ijk}ik_0 h_k\partial_j u + ik_0\varepsilon^{ij}e_j = 0$$
$$\frac{1}{\sqrt{g}}\varepsilon^{ijk}h_k\partial_j u + \varepsilon^{ij}e_j = -\frac{1}{ik_0}\frac{1}{\sqrt{g}}\varepsilon^{ijk}\partial_j h_k.$$

Уравнения Максвелла приводятся к следующему виду:

$$\begin{cases}\varepsilon^{ijk}e_k\partial_j u - \sqrt{g}\mu^{ij}h_j = 0,\\ \varepsilon^{ijk}h_k\partial_j u + \sqrt{g}\varepsilon^{ij}e_j = 0,\\ \varepsilon^{ij}e_j\partial_i u = 0,\\ \mu^{ij}h_j\partial_i u = 0.\end{cases},$$

где ε^{ijk} — символ Леви-Чивиты, ε^{ij} — диэлектрическая проницаемость, при условии $k_0 \to \infty$.

Из первого уравнения выразим h_j и подставим во второе:

$$\mu_{li}^{-1}\varepsilon^{ijk}e_k\partial_j u - \sqrt{g}\mu_{li}^{-1}\mu^{ij}h_j = 0.$$

Выполним замену: $\mu_{li}^{-1}\mu^{ij} = g_l^j$:

$$\mu_{li}^{-1}\varepsilon^{ijk}e_k\partial_j u - \sqrt{g}g_l^j h_j = 0 \Rightarrow \sqrt{g}h_l = \mu_{li}^{-1}\varepsilon^{ijk}e_k\partial_j u \Rightarrow h_l = \frac{1}{\sqrt{g}}\mu_{li}^{-1}\varepsilon^{ijk}e_k\partial_j u.$$

Преобразуем индексы, чтобы подставить во второе уравнение:

$$\begin{aligned}
&h_k = \frac{1}{\sqrt{g}}\mu_{kl}^{-1}\varepsilon^{lmn}e_n\partial_m u,\\
&\varepsilon^{ijk}\frac{1}{\sqrt{g}}\mu_{kl}^{-1}\varepsilon^{lmn}e_n\partial_m u\partial_j u + \sqrt{g}\varepsilon^{ij}e_j = 0,\\
&\varepsilon^{ijk}\mu_{kl}^{-1}\varepsilon^{lmn}e_n\partial_m u\partial_j u + g\varepsilon^{ij}e_j = 0,\\
&\varepsilon^{ijk}\mu_{kl}^{-1}\varepsilon^{lmn}\partial_m u\partial_j u e_n + g\varepsilon^{in}e_n = 0.
\end{aligned}$$

Уравнение эйконала (2.13) принимает вид:

$$g^{ij}\partial_i u\partial_j u = \varepsilon^{ij}\mu_{ij}.$$

■

2.2. Символьные исследования уравнений Максвелла в формализме пространственно-временной алгебры

2.2.1. Геометрическая алгебра пространства-времени

Основные понятия геометрической алгебры

Геометрическая алгебра является реализацией абстрактной алгебры Клиффорда [10], где элементами являются *мультивекторы*, а операцией «умножения» — операция *геометрического умножения*.

Мультивектор является градуированным объектом, представляющим собой линейную комбинацию разложимых p-векторов (кососимметричных ковариантных тензоров). Сами p-векторы совместно с операцией внешнего произведения $\wedge$ представляют собой реализацию абстрактной алгебры Грассмана [22]. Основным отличительным свойством внешнего произведения векторов является свойство *кососимметричности*: $\mathbf{v} \wedge \mathbf{v} = 0$. Более подробно

о внешней алгебре можно прочитать в фундаментальных работах [9; 11; 12; 30; 53; 54; 56].

Рассмотрим четырехмерное пространство Минковского (пространство-время) $E^4_{1,3}$ с базисом $\langle \mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \rangle$, метрическим тензором $g = \mathrm{diag}(1, -1, -1, -1)$ (иначе говоря с сигнатурой $(+, -, -, -)$):

$$(\mathbf{e}_0, \mathbf{e}_0) = 1, \quad (\mathbf{e}_0, \mathbf{e}_i) = 0, \quad (\mathbf{e}_i, \mathbf{e}_j) = -\delta_{ij}.$$

Операцию *геометрического умножения* для двух векторов $\mathbf{u}, \mathbf{v} \in E^4_{1,3}$ можно определить конструктивно с помощью формулы:

$$\mathbf{uv} = (\mathbf{u}, \mathbf{v}) + \mathbf{u} \wedge \mathbf{v},$$

где сама операция не обозначается никаким знаком. Результатом действия геометрического произведения является элемент градуированной алгебры Клиффорда, называемой *геометрической алгеброй*.

Работа с геометрической алгеброй упрощается, если введен ортогональный базис. В случае пространства Минковского из определения геометрического умножения выводятся два ключевых свойства для базисных векторов:

$$\mathbf{e}_\alpha \mathbf{e}_\beta = -\mathbf{e}_\beta \mathbf{e}_\alpha, \ \alpha \neq \beta \quad \text{и} \quad \mathbf{e}_i \mathbf{e}_i = -1, \ \mathbf{e}_0 \mathbf{e}_0 = +1.$$

Также $\mathbf{e}_\alpha \wedge \mathbf{e}_\beta = \mathbf{e}_\alpha \mathbf{e}_\beta$, $\mathbf{e}_\alpha \wedge \mathbf{e}_\beta \wedge \mathbf{e}_\gamma = \mathbf{e}_\alpha \mathbf{e}_\beta \mathbf{e}_\gamma$ и т.д. Часто используют обозначение $\mathbf{e}_\alpha \mathbf{e}_\beta = \mathbf{e}_{\alpha\beta}$ и т.д.

В пространстве-времени операция внешнего произведения порождает четыре внешние алгебры:

— алгебру 1-векторов с базисом $\langle \mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \rangle$;

— алгебру 2-векторов (бивекторов) с базисом $\langle \mathbf{e}_0\mathbf{e}_1, \mathbf{e}_0\mathbf{e}_2, \mathbf{e}_0\mathbf{e}_3, \mathbf{e}_1\mathbf{e}_2, \mathbf{e}_1\mathbf{e}_3, \mathbf{e}_2\mathbf{e}_3 \rangle$;

— алгебру 3-векторов с базисом $\langle \mathbf{e}_{012}, \mathbf{e}_{013}, \mathbf{e}_{023}, \mathbf{e}_{123} \rangle$;

— алгебру 4-векторов (квадривекторов) с одним базисным 4-вектором $\mathbf{e}_{0123}$, который обычно обозначают как $\mathbf{I}$.

Шесть бивекторных базисов распадаются на два класса — пространственно-временной базис $\{\mathbf{e}_0\mathbf{e}_1, \mathbf{e}_0\mathbf{e}_2, \mathbf{e}_0\mathbf{e}_3\}$ и чисто пространственный базис

$\mathbf{e}_1\mathbf{e}_2, \mathbf{e}_1\mathbf{e}_3, \mathbf{e}_2\mathbf{e}_3$. Пространственно-временные базисы обладают свойством гиперболической мнимой единицы, т.е. $\mathbf{e}_{0i}\mathbf{e}_{0i} = 1$, а чисто пространственные — свойством эллиптической мнимой единицы, т.е. $\mathbf{e}_{ij}\mathbf{e}_{ij} = -1$.

Можно составить следующие таблицы геометрического умножения:

	$\mathbf{e}_{01}$	$\mathbf{e}_{02}$	$\mathbf{e}_{03}$
$\mathbf{e}_{01}$	1	$-\mathbf{I}\mathbf{e}_{03}$	$\mathbf{I}\mathbf{e}_{02}$
$\mathbf{e}_{02}$	$\mathbf{I}\mathbf{e}_{03}$	1	$-\mathbf{I}\mathbf{e}_{01}$
$\mathbf{e}_{03}$	$-\mathbf{I}\mathbf{e}_{02}$	$\mathbf{I}\mathbf{e}_{01}$	1

	$\mathbf{e}_{12}$	$\mathbf{e}_{13}$	$\mathbf{e}_{23}$
$\mathbf{e}_{12}$	-1	$\mathbf{e}_{23}$	$-\mathbf{e}_{13}$
$\mathbf{e}_{13}$	$-\mathbf{e}_{23}$	-1	$\mathbf{e}_{12}$
$\mathbf{e}_{23}$	$\mathbf{e}_{13}$	$-\mathbf{e}_{12}$	-1

Из первой таблицы видно, что пространственно-временные базисы изоморфны матрицам Паули (с точности до знаков), поэтому в [11, стр. 135] для обозначения $\mathbf{e}_{0i}$ используется буква сигма σ_i.

Дифференциальные операторы

Введем оператор *векторной производной* (vector derivative) [11, стр. 168], определяемый следующей формулой:

$$\nabla = \mathbf{e}^i \frac{\partial}{\partial x^i} = \mathbf{e}^1 \frac{\partial}{\partial x^1} + \ldots + \mathbf{e}^n \frac{\partial}{\partial x^n},$$

где $\{\mathbf{e}^i\}$ — *взаимный базис* (reciprocal frame), векторы которого определяются следующим образом:

$$(\mathbf{e}^i, \mathbf{e}_j) = \delta^i_j.$$

Для произвольного вектора $\mathbf{x}$ выполняется равенство

$$(\mathbf{e}^i, \mathbf{x}) = (\mathbf{e}^i, x^j\mathbf{e}_j) = x^j(\mathbf{e}^i, \mathbf{e}_j) = \\ = x^j\delta^i_j = x^i \Rightarrow x^i = (\mathbf{e}^i, \mathbf{x}).$$

В случае ортонормированного евклидова пространства $\mathbf{e}^i = \mathbf{e}_i$. В случае трехмерного или двумерного ортонормированного евклидова пространства оператор ∇ совпадает с классическим дифференциальным оператором $\boldsymbol{\nabla}$. С помощью ∇ и геометрического умножения можно записать операции градиента, дивергенции и ротора классического векторного анализа, а также обобщить их на большие размерности и метрики, отличные от ортонормированной.

Умножив геометрически ∇ на скалярную функцию $f(\mathbf{x})$, где $\mathbf{x} = x^i \mathbf{e}_i = x^1 \mathbf{e}_1 + \ldots + x^n \mathbf{e}_n$, получим формулу для градиента:

$$\nabla f(\mathbf{x}) = \mathbf{e}^i \frac{\partial f}{\partial x^i} = \left(\frac{\partial f}{\partial x^1}, \ldots, \frac{\partial f}{\partial x^n} \right),$$

который представляет собой вектор, записанный в взаимном базисе $\langle \mathbf{e}^1, \ldots, \mathbf{e}^n \rangle$.

Умножим ∇ на векторное поле $\mathbf{u}(\mathbf{x})$ и получим сумму скалярной и бивекторной частей:

$$\nabla \mathbf{u} = (\nabla, \mathbf{u}) + \nabla \wedge \mathbf{u}.$$

Скалярная часть является обобщением оператора дивергенции. Используем равенство $x^i = (\mathbf{e}^i, \mathbf{x})$ и запишем:

$$(\nabla, \mathbf{u}(\mathbf{x})) = \left(\frac{\partial}{\partial x^i} \mathbf{e}^i, \mathbf{u} \right) = \\ = \frac{\partial}{\partial x^i} \left(\mathbf{e}^i, \mathbf{u} \right) = \frac{\partial u^i}{\partial x^i} = \frac{\partial u^1}{\partial x^1} + \ldots + \frac{\partial u^n}{\partial x^n}.$$

Бивекторная часть обобщает операцию ротора:

$$\nabla \wedge \mathbf{u} = \mathbf{e}^i \wedge \frac{\partial \mathbf{u}}{\partial x^i} = \frac{\partial u^j}{\partial x^i} \mathbf{e}^i \wedge \mathbf{e}_i.$$

2.2.2. Мультивекторная запись уравнений Максвелла

Получим мультивекторную запись вакуумных уравнений Максвелла.

Уравнения Максвелла в дифференциальной форме в системе СИ имеют следующий вид:

$$\begin{cases} \nabla \times \mathbf{H} = \dfrac{\partial \mathbf{D}}{\partial t} + \mathbf{j}, \\ \nabla \times \mathbf{E} = -\dfrac{\partial \mathbf{B}}{\partial t}, \\ \nabla \cdot \mathbf{D} = \rho, \\ \nabla \cdot \mathbf{B} = 0. \end{cases} \tag{2.18}$$

Здесь $\mathbf{B}$ — вектор индукции магнитного поля, $\mathbf{E}$ — вектор напряженности электрического поля, $\mathbf{H}$ — напряжённость магнитного поля, $\mathbf{D}$ — электрическая индукция, $\mathbf{j}$ — вектор плотности внешнего электрического тока, ρ — плотность электрического заряда.

Уравнения Максвелла в дифференциальной форме для изотропной среды

Диэлектрическую и магнитную проницаемости положим равными единице, т.е. $\varepsilon = \mu = 1$ (вакуум). Кроме того, будем считать, что

$$\mathbf{H} = \mathbf{B},$$
$$\mathbf{D} = \mathbf{E}.$$

Тогда вакуумные уравнения Максвелла в дифференциальной форме (2.18) приобретают вид:

$$\begin{cases} \nabla \times \mathbf{B} = \dfrac{\partial \mathbf{E}}{\partial t} + \mathbf{j}, \\ \nabla \times \mathbf{E} = -\dfrac{\partial \mathbf{B}}{\partial t}, \\ \nabla \cdot \mathbf{E} = \rho, \\ \nabla \cdot \mathbf{B} = 0. \end{cases} \tag{2.19}$$

При этом для плотностей тока $\mathbf{j}$ и заряда ρ выполняется уравнение непрерывности:

$$\nabla \cdot \mathbf{j} + \frac{\partial \rho}{\partial t} = 0.$$

Символом ∇ в данной системе обозначен оператор набла, с помощью которого записаны операторы ротора $\nabla\times$ и дивергенции $\nabla\cdot$. Далее данный символ будет выступать только в роли оператора векторной производной.

Отметим, что все векторы в данной записи уравнений Максвелла являются векторами трехмерного Евклидова пространства (а точнее векторными полями, зависящими явно от трех координат и неявно от времени t). Их компоненты обозначаются с нижними индексами x, y, z, например E_x, E_y, E_z.

Мультивектор Фарадея

Рассмотрим теперь необходимые элементы, с помощью которых можно записать уравнения Максвелла в мультивекторном виде. Для этого введем пространство Минковского с сигнатурой $(+,-,-,-)$ и базисными векторами $\langle \mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \rangle$. Оператор векторной производной в этом базисе записывается следующим образом:

$$\nabla = \mathbf{e}_0 \frac{\partial}{\partial x^0} - \mathbf{e}_1 \frac{\partial}{\partial x^1} - \mathbf{e}_2 \frac{\partial}{\partial x^2} - \mathbf{e}_3 \frac{\partial}{\partial x^3},$$

так как взаимный базис $\{\mathbf{e}^\alpha\}$, $\alpha = 0, 1, 2, 3$ имеет вид $\langle \mathbf{e}_0, -\mathbf{e}_1, -\mathbf{e}_2, -\mathbf{e}_3 \rangle$, в чем можно убедится вычислив скалярные произведения:

$$\begin{aligned}
(\mathbf{e}_0, \mathbf{e}_0) &= 1, \\
(\mathbf{e}_1, -\mathbf{e}_1) &= 1, \\
(\mathbf{e}_2, -\mathbf{e}_2) &= 1, \\
(\mathbf{e}_3, -\mathbf{e}_3) &= 1.
\end{aligned}$$

Единичный элемент объема $\mathbf{e}_0\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$ обозначим как $\mathbf{I}$.

Напряженность электрического поля $\mathbf{E}$ представим в виде бивектора с тремя пространственно-временными компонентами:

$$\mathbf{E} = E^{10}\mathbf{e}_1\mathbf{e}_0 + E^{20}\mathbf{e}_2\mathbf{e}_0 + E^{30}\mathbf{e}_3\mathbf{e}_0.$$

Бивекторные компоненты E^{10}, E^{20}, E^{30} соответствуют координатам вектора E_x, E_y, E_z классической векторной записи. Хотя у бивектора в четырехмерном пространстве Минковского может быть 6 ненулевых компонент, бивектор $\mathbf{E}$ имеет только три указанные ненулевые компоненты.

Аналогично представим индукцию магнитного поля $\mathbf{B}$:

$$\mathbf{B} = B^{10}\mathbf{e}_1\mathbf{e}_0 + B^{20}\mathbf{e}_2\mathbf{e}_0 + B^{30}\mathbf{e}_3\mathbf{e}_0 =$$

$$= B_x \mathbf{e}_1 \mathbf{e}_0 + B_y \mathbf{e}_2 \mathbf{e}_0 + B_z \mathbf{e}_3 \mathbf{e}_0.$$

Объединим плотность электрического заряда ρ (скаляр) и плотность внешнего электрического тока $\mathbf{j}$ (вектор) в четырехмерный пространственно-временной вектор $\mathbf{J}$. Вектор $\mathbf{J}$ будем называть пространственно-временной плотностью электрического тока [11, стр. 230] и запишем его в следующем виде:

$$\mathbf{J} = \rho \mathbf{e}_0 + j^1 \mathbf{e}_1 + j^2 \mathbf{e}_2 + j^3 \mathbf{e}_3 = \rho \mathbf{e}_0 + j_x \mathbf{e}_1 + j_y \mathbf{e}_2 + j_z \mathbf{e}_3.$$

Если умножить $\mathbf{J}$ справа на $\mathbf{e}_0$, то получим мультивектор со скалярной и бивекторной частями:

$$\mathbf{J}\mathbf{e}_0 = \rho + j_x \mathbf{e}_1 \mathbf{e}_0 + j_y \mathbf{e}_2 \mathbf{e}_0 + j_z \mathbf{e}_3 \mathbf{e}_0.$$

Следовательно, плотность электрического тока $\mathbf{j}$ можно представить в виде бивектора $j^{10} \mathbf{e}_1 \mathbf{e}_0 + j^{20} \mathbf{e}_2 \mathbf{e}_0 + j^{30} \mathbf{e}_3 \mathbf{e}_0$ с пространственно-временными компонентами (по аналогии с представлением $\mathbf{E}$ и $\mathbf{B}$).

Наконец составим бивектор Фарадея $\mathbf{F}$ как сумму бивекторов $\mathbf{E}$ и $\mathbf{IB}$:

$$\mathbf{F} = \mathbf{E} + \mathbf{IB}.$$

При умножении на $\mathbf{I}$ пространственно-временные бивекторные базисы $\mathbf{e}_0 \mathbf{e}_1$, $\mathbf{e}_0 \mathbf{e}_2$, $\mathbf{e}_0 \mathbf{e}_3$ превращаются в чисто пространственные:

$$\mathbf{I}\mathbf{e}_1 \mathbf{e}_0 = -\mathbf{e}_2 \mathbf{e}_3, \ \mathbf{I}\mathbf{e}_2 \mathbf{e}_0 = +\mathbf{e}_1 \mathbf{e}_3, \ \mathbf{I}\mathbf{e}_3 \mathbf{e}_0 = -\mathbf{e}_1 \mathbf{e}_2,$$

из-за чего бивектор Фарадея имеет шесть компонент — максимально возможное количество компонент в четырехмерном пространстве Минковского:

$$\mathbf{F} = E^{10} \mathbf{e}_1 \mathbf{e}_0 + E^{20} \mathbf{e}_2 \mathbf{e}_0 + E^{30} \mathbf{e}_3 \mathbf{e}_0 - \\ - B^{10} \mathbf{e}_2 \mathbf{e}_3 + B^{20} \mathbf{e}_1 \mathbf{e}_3 - B^{30} \mathbf{e}_3 \mathbf{e}_2.$$

Теперь уравнения Максвелла можно записать в очень компактной форме:

$$\nabla \mathbf{F} = \mathbf{J}.$$

2.2.3. Вывод дифференциальной формы уравнений Максвелла из мультивекторной с помощью символьных вычислений

Используем библиотеку Galgebra для того, чтобы вычислить компоненты мультивектора $\nabla\mathbf{F} - \mathbf{J}$ в декартовых координатах.

Вначале импортируем все необходимые модули:

```
import sympy as sp
from galgebra.ga import Ga
# Функция для распечатки исходного кода LaTeX формул:
from galgebra.printer import latex
# Функции для отображения обработки и отображения LaTeX формул:
from IPython.display import Math, DisplayObject
```

Все вычисления мы будем проводить в интерактивной оболочке Jupyter Notebook [62]. Настроим отображение формул с помощью вызова следующей функции:

```
sp.init_printing(latex_printer=latex, use_latex='mathjax', use_unicode=True)
```

После всех настроек зададим пространство Минковского и структуру геометрической алгебры на нем:

```
m4d = Ga('e', g=[1,-1,-1,-1], coords=sp.symbols('t, x, y, z',real=True))
```

Здесь мы указываем символ `e`, который будет использоваться для обозначения базисного вектора, а также символы `t`, `x`, `y`, `z`, используемые для обозначения индексов. Также в параметре `g` указывается метрика, которая может быть только диагональной, но не обязательно нормированной.

Далее в отдельные переменные записываем базисные векторы, пространственно-временную часть бивекторного базиса и квадривекторный базисный элемент $\mathbf{I}$:

```
# Векторный базис:
e0, e1, e2, e3 = m4d.mv()
# Пространственно-временная часть бивекторного базиса:
σ1 = e10 = e1*e0
σ2 = e20 = e2*e0
σ3 = e30 = e3*e0

I = e0*e1*e2*e3
```

Греческие буквы можно вводить в блокноте Jupyter с помощью их обозначений в формате LaTeX. Для этого следует набрать, например, `\sigma` и нажать клавишу Tab.

Для использования оператора векторной производной следует вызвать метод `grads` объекта `m4d` класса `Ga`:

```
(grad, rgrad) = o3d.grads()
```

Этот метод возвращает правый и левый операторы векторной производной, так как геометрическое умножение не коммутативно и при умножении слева следует использовать `grad`, а справа `rgrad`. Их значения равны соответственно:

$$\mathbf{e}_t\frac{\partial}{\partial t} - \mathbf{e}_x\frac{\partial}{\partial x} - \mathbf{e}_y\frac{\partial}{\partial y} - \mathbf{e}_z\frac{\partial}{\partial z},$$
$$\frac{\partial}{\partial t}\mathbf{e}_t - \frac{\partial}{\partial x}\mathbf{e}_x - \frac{\partial}{\partial y}\mathbf{e}_y - \frac{\partial}{\partial z}\mathbf{e}_z.$$

Такое разграничение двух операторов имеет смысл только в рамках модуля `Galgebra` в силу ограничений синтаксиса языка Python.

Далее задаем компоненты электрического и магнитного полей, вектор тока и плотность заряда как функции от переменных t, x, y, z:

```
Ex = sp.Function('E_1')(t,x,y,z)
Ey = sp.Function('E_2')(t,x,y,z)
Ez = sp.Function('E_3')(t,x,y,z)

Bx = sp.Function('B_1')(t,x,y,z)
By = sp.Function('B_2')(t,x,y,z)
Bz = sp.Function('B_3')(t,x,y,z)

jx = sp.Function('j_1')(t,x,y,z)
jy = sp.Function('j_2')(t,x,y,z)
jz = sp.Function('j_3')(t,x,y,z)

ρ = sp.Function('ρ')(t,x,y,z)
```

После этого можно записать $\mathbf{E}$, $\mathbf{B}$, $\mathbf{j}$ и ρ:

```
# Напряженность электрического поля:
E = Ex*σ1 + Ey*σ2 + Ez*σ3
# Индукция магнитного поля:
B = Bx*σ1 + By*σ2 + Bz*σ3
# Плотность внешнего электрического тока:
```

```
j = jx*σ1 + jy*σ2 + jz*σ3
# Пространственно-временной вектор тока:
J = ρ*e0 + jx*e1 + jy*e2 + jz*e3
```

Теперь можно составить бивектор Фарадея:

```
F = E + I*B
```

В результате получим следующий бивектор с полным набором как пространственных, так и пространственно–временных компонент:

$$\mathbf{F} = -\mathbf{E}_x\mathbf{e}_{tx} - \mathbf{E}_y\mathbf{e}_{ty} - \mathbf{E}_z\mathbf{e}_{tz} - \mathbf{B}_z\mathbf{e}_{xy} + \mathbf{B}_y\mathbf{e}_{xz} - \mathbf{B}_x\mathbf{e}_{yz}.$$

Запишем теперь уравнения Максвелла в виде мультивектора:

```
Eq = grad*F - J
```

Данный мультивектор имеет ненулевую векторную и тривекторную части. Приравняв весь мультивектор к нулю, мы получим дифференциальную форму уравнений Максвелла, если распишем отдельные компоненты данного мультивектора.

Векторную часть можно вычленить, вызвав метод `grade`:

```
Eq.grade(1)
```

Эта часть обладает следующими компонентами:

$$\begin{aligned}
&-\rho + \frac{\partial E_x}{\partial x} + \frac{\partial E_y}{\partial y} + \frac{\partial E_z}{\partial z},\\
&-j_x - \frac{\partial B_y}{\partial z} + \frac{\partial B_z}{\partial y} - \frac{\partial E_x}{\partial t},\\
&-j_y + \frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x} - \frac{\partial E_y}{\partial t},\\
&-j_z - \frac{\partial B_x}{\partial y} + \frac{\partial B_y}{\partial x} - \frac{\partial E_z}{\partial t}.
\end{aligned}$$

Приравнивание полученных компонент векторной части к нулю даст нам первое и третье уравнения из системы (2.19). В свою очередь, тривекторная часть получается вызовом метода `Eq.grade(3)` и также имеет четыре компоненты:

$$-\frac{\partial B_z}{\partial t} + \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x},$$

$$\frac{\partial B_y}{\partial t} + \frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x},$$
$$-\frac{\partial B_x}{\partial t} + \frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y},$$
$$\frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y} + \frac{\partial B_z}{\partial z},$$

приравняв которые к нулю мы получим второе и четвертое уравнения из (2.19).

Взяв повторно векторную производную:

$$\nabla\nabla\mathbf{F} - \nabla\mathbf{J} = 0, \tag{2.20}$$

получим мультивектор со скалярной и полной бивекторной частями (с шестью компонентами).

Равенство нулю скалярной части выражения (2.20) даёт уравнение непрерывности. Равенство нулю пространственно–временных компонент в выражении (2.20) даёт волновое уравнение для электрического поля. Равенство нулю чисто пространственных компонент выражения (2.20) даёт волновое уравнение для магнитного поля.

2.3. Численный метод FSM (Fast Sweeping Method)

Можно выделить два подхода к численному решению уравнения эйконала.

— Преобразование к системе обыкновенных дифференциальных уравнений (системе уравнений Гамильтона) методом характеристик [71; 72], а затем применение одного из многочисленных методов численного решения таких уравнений.

— Подход к задаче как с стационарной краевой задаче. Разработка эффективного численного алгоритма решения системы нелинейных уравнений, получившихся при дискретизации. К этому типу методов относится, например, метод быстрой прогонки (fast marching method).

Название метода FSM — *Fast Sweeping Method* [23; 27; 29; 43; 67] можно перевести на русский как метод быстрого подметания. Автору не известен стандартный перевод названия данного метода, поэтому везде в тексте используется аббревиатура FSM.

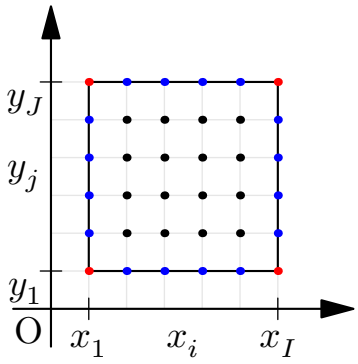


Рис. 2.3. Различные точки сетки разбиения области интегрирования

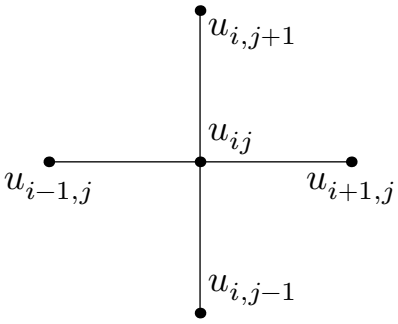


Рис. 2.4. Шаблон численной схемы

Метод был предложен в 2000 году [67]. Основная идея метода заключается в использовании противопотоковой разностной схемы Годунова и итерационной схемы Гаусса–Зейделя с переменным порядком прохода узлов сетки. Подробное описанием численной схемы приведено в разделе 2.3.

FSM прост в реализации и требует конечного числа итераций. Сложность алгоритма $O(N)$ для N точек сетки. Число итераций не зависит от числа узлов сетки (от размера сетки). FSM метод можно распространить на общий случай уравнения Гамильтона–Якоби.

2.3.1. FSM для двумерного уравнения эйконала

Перейдем к описанию численной схемы. Разобьем всю области интегрирования на дискретные узлы с помощью прямоугольной сетки, показанной на рисунке 2.3, где

— по оси Ox имеем I точек разбиения $x_1 < x_2 < x_3 < \ldots < x_{I-1} < x_I$,

— по оси Oy имеем J точек разбиения $y_1 < y_2 < y_3 < \ldots < y_{J-1} < y_J$.

Сетка будет состоять из $I \times J$ узлов с координатами (x_i, y_j), где $i = 1, \ldots, I$, а $j = 1, \ldots, J$.

Предположим, что разбиение выбрано так, что шаг сетки h одинаков для обеих осей. Сеточная функция u_{ij} аппроксимирует функцию $u(x, y)$ в узлах сетки, т.е. $u_{ij} \approx u(x_i, y_j)$ и только в точке (x_0, y_0) выполняется точное равенство.

Все точки сетки можно разделить на три группы, которые выделены на рисунке 2.3 разными цветами:

1. Внутренние точки сетки с индексами $i = 2, \dots, I-1$ и $j = 2, \dots, J-1$ показаны на схеме 2.3 черным цветом.

2. Точки четырех границ сетки со следующими индексами
   - левая граница: $i = 1, \quad j = 2, \dots, J-1$,
   - правая граница: $i = I, \quad j = 2, \dots, J-1$,
   - нижняя граница $i = 2, \dots, I-1, \quad j = 1$,
   - верхняя граница $i = 2, \dots, I-1, \quad j = J$,

   показаны на схеме 2.3 синим цветом.

3. Угловые точки со следующими фиксированными индексами:
   - левая нижняя угловая точка $i = 1, \quad j = 1$,
   - левая верхняя угловая точка $i = 1, \quad j = J$,
   - правая нижняя угловая точка $i = I, \quad j = 1$,
   - правая верхняя угловая точка $i = I, \quad j = J$,

   обозначены на схеме 2.3 красным цветом.

Перейдем к изложению алгоритма. В качестве схемы дискретизации используется схема Годунова (противоточная разностная схема) для внутренних точек области. Введем следующие обозначения:

$$u_{xmin} = \min(u_{i-1,j}, u_{i+1,j}), \; u_{ymin} = \min(u_{i,j-1}, u_{i,j+1}), \; n_{ij} = n(x_i, y_j),$$

а также индикаторную функцию:

$$(x)^+ = \begin{cases} x, & x > 0, \\ 0, & x \leqslant 0. \end{cases}$$

Для инициализации вычислений в первую очередь следует задать значения сеточной функции $u_{ij} = 0$ на границе Г. Эти значения в последующих вычислениях останутся неизменными. Для всех остальных точек сеточной функции u_{ij} следует присвоить достаточно большие положительные значения, которых она заведомо достичь не может. В ходе работы численной схемы эти значения будут пересчитываться.

Вычислительный процесс состоит из четырех заметаний всей прямоугольной области. Каждое такое заметание представляет собой два вложенных цикла, где индексы пробегают в следующем порядке:

- $i = 1, \dots, I$ и $j = 1, \dots, J$ — прямой порядок,
- $i = I, \dots, 1$ и $j = 1, \dots, J$ — смешанный порядок,
- $i = I, \dots, 1$ и $j = J, \dots, 1$ — обратный порядок,
- $i = 1, \dots, I$ и $j = J, \dots, 1$ — смешанный порядок.

На каждом шаге следует решить нелинейное уравнение, коэффициенты которого для каждой группы точек будут немного изменяться.

Группа I Точки $i = 2, \dots, I-1$ и $j = 2, \dots, J-1$

$$[(u_{ij} - u_{xmin})^+]^2 + [(u_{ij} - u_{ymin})^+]^2 = n_{ij}^2 h^2$$

Группа II К этой группе относятся следующие точки:

- левая граница: $i = 1, \quad j = 2, \dots, J-1$:

$$[(u_{1j} - u_{2j})^+]^2 + [(u_{1j} - u_{ymin})^+]^2 = n_{1j}^2 h^2,$$

- правая граница: $i = I, \quad j = 2, \dots, J-1$:

$$[(u_{Ij} - u_{I-1,j})^+]^2 + [(u_{Ij} - u_{ymin})^+]^2 = n_{Ij}^2 h^2,$$

- нижняя граница $i = 2, \dots, I-1, \quad j = 1$:

$$[(u_{i1} - u_{xmin})^+]^2 + [(u_{i1} - u_{i2})^+]^2 = n_{i1}^2 h^2,$$

- верхняя граница $i = 2, \dots, I-1, \quad j = J$:

$$[(u_{iJ} - u_{xmin})^+]^2 + [(u_{iJ} - u_{i,J-1})^+]^2 = n_{iJ}^2 h^2.$$

Группа III К этой группе относятся следующие точки:

— левая нижняя угловая точка $i = 1, \quad j = 1$:

$$[(u_{11} - u_{21})^+]^2 + [(u_{11} - u_{12})^+]^2 = n_{11}^2 h^2,$$

— левая верхняя угловая точка $i = 1, \quad j = J$:

$$[(u_{1J} - u_{2J})^+]^2 + [(u_{1J} - u_{1,J-1})^+]^2 = n_{1J}^2 h^2,$$

— правая нижняя угловая точка $i = I, \quad j = 1$:

$$[(u_{I1} - u_{I-1,1})^+]^2 + [(u_{I1} - u_{I2})^+]^2 = n_{I1}^2 h^2,$$

— правая верхняя угловая точка $i = I, \quad j = J$:

$$[(u_{IJ} - u_{I-1,J})^+]^2 + [(u_{IJ} - u_{I,J-1})^+]^2 = n_{IJ}^2 h^2.$$

Каждое из этих уравнений различается лишь числовыми коэффициентами и имеет следующий вид:

$$\left[(x-a)^+\right]^2 + \left[(x-b)^+\right]^2 = n_{ij}^2 h^2.$$

Его можно свести к квадратному уравнения и записать решение в виде

$$x = \begin{cases} \min(a,b) + n_{ij}h, & |a-b| \geqslant n_{ij}h, \\ \dfrac{a+b+\sqrt{2n_{ij}^2h^2 - (a-b)^2}}{2}, & |a-b| < n_{ij}h. \end{cases}$$

Структура вычислительного процесса метода быстрого заметания приведена на рисунке 2.5.

2.4. Нейронные сети, основанные на физике

Нейронные сети, основанные на физике (Physics-informed neural networks, PINN), становятся все более эффективным способом решения дифференциальных уравнений и создания нейронных аналогов физических моделей. Классические нейронные сети выводят решения исключительно на основе

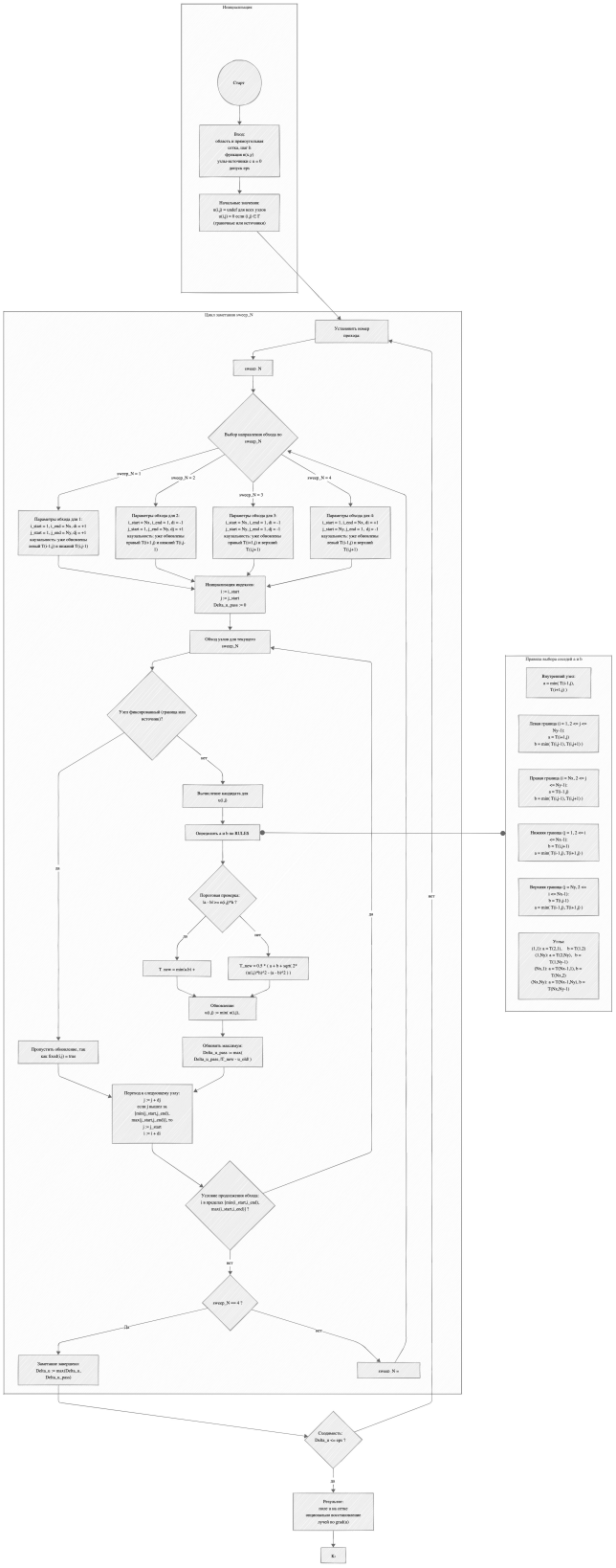


Рис. 2.5. Блок-схема алгоритма быстрого заметания FSM (Fast Sweeping Method)

данных, представленных набором пар «состояние-значение». Особенностью PINN является то, что они изначально учитывают физику проблемы, лежащую в основе уравнений в частных или обыкновенных производных. То есть функция потерь включает в себя ODE/PDE и начальные/граничные условия. Термин PINN был введён в [52], где PINN описан как новый класс универсальных аппроксиматоров функций, которые способны кодировать любые основные физические законы и которые могут быть описаны уравнениями в частных производных.

2.4.1. Общее описание

Рассмотрим дифференциальное уравнение в следующей форме:

$$F(u(x); \lambda) = 0,$$

где F - дифференциальный оператор, u - решение ДУ, λ — параметры уравнения, а $x = x_1, \ldots, x_n \in \Omega$ - n-мерный вектор координат, принадлежащий области Ω.

Пусть B - граничный оператор, а функция u удовлетворяет граничным условиям:

$$B(u(x); \lambda) = 0$$

И так же пусть I - оператор начальных условий, а функция u удовлетворяет начальным условиям:

$$I(u(x); \lambda) = 0.$$

Нейронные сети, основанные на физике (PINN), решают УЧП, используя теорему универсального приближения(Universal Approximation Theorem) [26], которая утверждает, что для любой измеримой u существует достаточно большая нейронная сеть N с весами w, такими, что $||N(x, w) - u(x) < \varepsilon||$ для всех $x \in \Omega$. Это предполагает, что произвольное ДУ можно решить, заменив неизвестное решение $u(x)$ нейронной сетью $N(x; w)$ и найдя веса w такие, что $F(N; \lambda) \approx 0$ по всем $x \in \Omega$. Формально это условие можно записать в одном уравнении, суммируя разность в каждой точке x,

$$L(w) = \int_{\Omega} ||F(N(x; w); \lambda)||dx, \tag{2.21}$$

где мы хотим найти веса нейронной сети w, которые минимизируют $L(w)$. В этом отличие от точного решения, где если $L(w) = 0$, то по определению нейронная сеть является решением дифференциального уравнения.

Поскольку граничные условия должны удовлетворяться только на некотором подмножестве $\partial\Omega$, полезно отделить граничные и начальные условия в свое собственное уравнение. Таким образом, получим:

$$L(w) = \int_{\Omega \setminus \partial\Omega} ||F(N(x, w); \lambda)||dx + \int_{\partial\Omega} ||B(N(x, w); \lambda)||dx + ||I(N(x, w); \lambda)||. \tag{2.22}$$

Уравнение (2.21) эквивалентно (2.22), но оно поясняет реализацию. Запишем его в следующих обозначениях:

$$L(w) = L_r + L_{ic} + L_{bc},$$

где L_r - невязка ДУ, L_{ic} - ошибка в начальных условиях, L_{bc} - ошибка в граничных условиях.

2.4.2. Общий алгоритм работы

Алгоритм работы PINN включает следующие основные шаги.

1. Определение задачи, основанной на физических законах, и формулировка управляющих уравнений, описывающих рассматриваемую систему.
2. Сбор данных, представляющих поведение системы, из экспериментов, симуляций или других источников.
3. Выбор архитектуры нейронной сети и инициализация её параметров.
4. Формулировка функции потерь, которая включает в себя как соответствие экспериментальным данным, так и удовлетворение физических уравнений.
5. Обучение нейронной сети путём минимизации функции потерь.

6. Проверка условий остановки обучения (например, достижение заданного количества эпох или минимизация потерь).
7. Анализ результатов и их интерпретация.

Или более развёрнуто.

1. Определение физики-ориентированной задачи:
   - формулировка управляющих уравнений, которые описывают поведение исследуемой системы (эти уравнения могут быть получены из фундаментальных принципов, таких как законы сохранения или конститутивные соотношения);
   - определение граничных и начальных условий для задачи.
2. Сбор данных:
   - получение данных, характеризующих поведение системы, — из экспериментов, симуляций или других источников;
   - подготовка данных для обучения: выбор точек (координат или временных значений), в которых будут оцениваться предсказания и потери.
3. Выбор и настройка архитектуры нейронной сети:
   - определение типа нейронной сети (например, полносвязная сеть);
   - выбор количества слоёв и нейронов в каждом слое;
   - подбор функций активации;
   - инициализация параметров сети (весов и смещений).
4. Формулировка функции потерь:
   - включение в функцию потерь двух основных компонентов:
     - data fidelity term — мера расхождения между предсказаниями сети и наблюдаемыми данными;
     - physics-informed term — обеспечение выполнения физических уравнений, выступающих в роли ограничений;

   - настройка весов компонентов функции потерь для баланса между соответствием данным и удовлетворением физических законов.

5. Обучение нейронной сети:
   - подача данных на вход сети и вычисление предсказаний;
   - расчёт потерь на основе сформулированной функции потерь;
   - обновление параметров сети с помощью оптимизатора (алгоритма оптимизации);
   - использование автоматического дифференцирования для вычисления производных, необходимых для обучения.
6. Проверка условий остановки обучения:
   - контроль достижения заданного количества эпох (итераций) обучения;
   - отслеживание минимизации функции потерь или других критериев остановки;
   - проверка стабильности и качества предсказаний.
7. Итеративное повторение этапов обучения и проверки до выполнения условий остановки. Этот процесс можно описать в виде блок-схемы (рис. 2.6)
8. Оценка результатов:
   - анализ предсказаний сети на соответствие как экспериментальным данным, так и физическим законам;
   - оценка точности и интерпретируемости полученного решения.
9. Интерпретация результатов и их применение:
   - использование обученной модели для предсказания поведения системы в новых условиях;
   - применение модели для решения прямых и обратных задач, ассимиляции данных и других целей.
10. Оптимизация и доработка модели при необходимости:

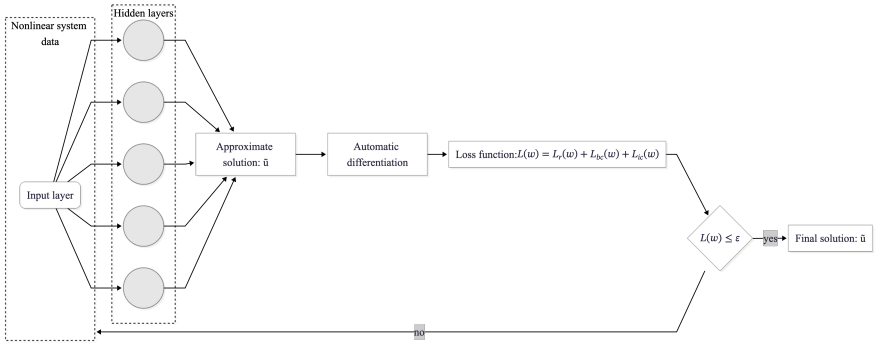


Рис. 2.6. Схема обучения PINN

— корректировка архитектуры сети, функции потерь или процесса обучения для улучшения результатов;

— повторение цикла обучения с изменёнными параметрами при неудовлетворительных результатах.

2.4.3. Общая характеристика пакета NeuralPDE

Пакет NeuralPDE [41] для языка Julia входит в коллекцию пакетов и утилит SciML [57]. В этой коллекции собраны пакеты, которые позволяют обсчитывать математические модели на основе дифференциальных уравнений разного вида, причем численные методы сочетаются с машинным обучением.

В частности NeuralPDE основывается на нейросетях, функция потерь которых строится с учетом дифференциальных уравнений математической модели, что позволяет обучать нейросеть с учетом решаемой физической задачи. Собственно это и есть Physics-Informed Neural Networks (PINN) [5].

NeuralPDE используются для решения трех объемных групп задач:

— аппроксимация решений систем обыкновенных дифференциальных уравнений (ОДУ);

— аппроксимация решений уравнений в частных производных (УрЧП);

— решение обратных задач, которые заключаются в определении коэффициентов ОДУ и УрЧП по известным решениям.

Пакет NeuralPDE разрабатывается с 2019 года [58]. Пакет можно установить стандартными средствами Julia, выполнив команду `add NeuralPDE` в режиме

управления пакетами. Стоит отметить, что при установке скачивается большое количество зависимостей — как дополнительных пакетов, так и *артефактов*. Артефактами в Julia называются сторонние бинарные файлы библиотек или дополнительных утилит, которые используются в том или ином устанавливаемом пакете.

По умолчанию NeuralPDE производит все расчеты силами центрального процессора. Для использования видеокарты необходимо дополнительно установить пакет Flux.jl [18] или Lux.jl [34].

Для задания ОДУ или УрЧП используется синтаксис пакета ModelingToolkit.jl [40], который также входит в коллекцию SciML. Согласно официальному описанию, ModelingToolkit.jl фреймворк для высокопроизводительных символьно-численных вычислений в области математического моделирования и наукоемкого машинного обучения. Он позволяет задать высокоуровневое описание задачи в символьном виде, для дальнейших расчетов и анализа. Символьное описание основывается ещё на одном пакете Julia — Symbolics.jl [24], который позиционируется как система компьютерной алгебры (CAS).

2.4.4. Решение ОДУ на Julia

Математически, `ODEProblem` определяет проблему:

$$u' = f(u, p, t)$$

для интервала $t \in (t_0, t_f)$ с начальным условием $u(t_0) = u_0$.

Решим простое ОДУ:

$$u' = cos(2\pi t)$$

для $t \in (0, 1)$ и $u_0 = 0$ с помощью `NNODE` и численным методом, затем сравним результат.

Чтобы решить эту задачу, мы определяем тип задачи, задавая ей уравнение, начальное условие и временной интервал, используя метод `ODEProblem`.

```
linear(u, p, t) = cos(t * 2 * pi)
tspan = (0.0, 1.0)
u0 = 0.0
prob = ODEProblem(linear, u0, tspan)
```

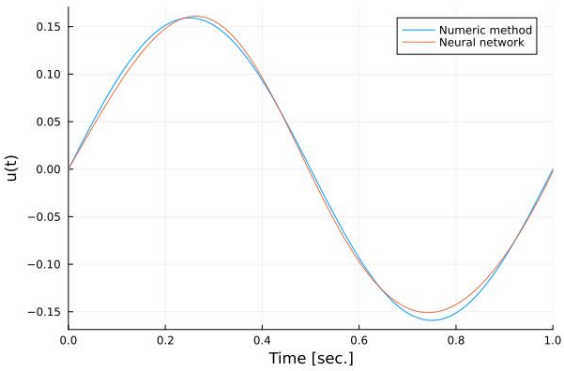


Рис. 2.7. График сравнения решений на интервале [0, 1]

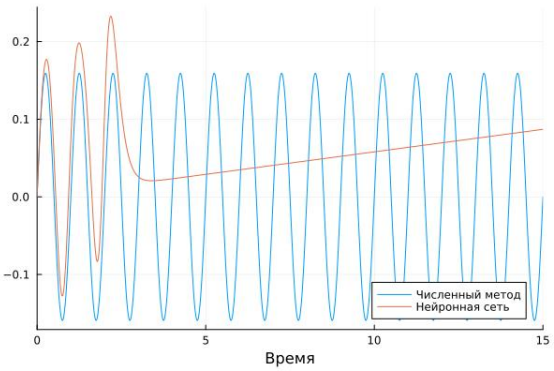


Рис. 2.8. График сравнения решений на интервале [0, 15]

Зададим `NeuralPDE.NNODE()` — алгоритм решения обыкновенных дифференциальных уравнений с помощью нейронной сети. Это специализация нейронной сети, основанной на физике, которая используется в качестве решателя стандартной задачи ОДУ. Также мы должны выбрать архитектуру нейронной сети. Для этого мы будем использовать Lux.jl, чтобы определить многослойный персептрон (MLP) с одним скрытым слоем из 5 узлов и сигмоидной функцией активации. Это выглядит как:

```
rng = Random.default_rng()
Random.seed!(rng, 0)
chain = Chain(Dense(1, 5, σ), Dense(5, 1))
ps, st = Lux.setup(rng, chain) |> Lux.f64
```

Вызываем метод `solve`, чтобы решить заданную проблему. В качестве метода решения задаем `Tsit5()` с шагом 0.01 (см. рис. 2.7):

```
sol_num = solve(prob, Tsit5(), saveat = 0.01)
```

Аналогичным образом построим график на временном интервале [0, 15] (рис. 2.8).

2.4.5. Модель Лотки–Вольтерры

Описание модели Лотки–Вольтерры

Рассмотрим математическую модель Лотки–Вольтерры (модель хищник–жертва), которая описывает взаимодействие двух видов животных. Причем один из них охотится за другими, которые обеспечены неисчерпаемыми пищевыми ресурсами [61; 70]. Модель описывается следующей системой уравнений:

$$\begin{cases} \dfrac{dx}{dt} = \alpha x(t) - \beta x(t) y(t), \\ \dfrac{dy}{dt} = -\gamma y(t) + \delta x(t) y(t). \end{cases}$$

В этой модели x — число жертв, y - число хищников. Коэффициент a описывает скорость естественного прироста числа жертв в отсутствие хищников, c - естественное вымирание хищников, лишенных пищи в виде жертв. Вероятность взаимодействия жертвы и хищника считается пропорциональной как количеству жертв, так и числу самих хищников. Каждый акт взаимодействия уменьшает популяцию жертв, но способствует увеличению популяции хищников (члены $-bxy$ и dxy в правой части уравнения).

Первый интеграл системы имеет вид [69]:

$$\alpha \log y - \beta y + \gamma \log x - \delta x = C, \ C = \text{const}.$$

Для решения системы мы задали параметры: $\alpha = 1.5$, $\beta = 1.0$, $\gamma = 3.0$, $\delta = 1.0$. Будем решать задачу Коши с начальными условиями:

$$\begin{cases} x(0) = 1, \\ y(0) = 1. \end{cases}$$

2.4.6. Численное исследование

Решим систему численным методом `Tsit5()` с шагом 0.01, используя библиотеку `DifferentialEquations.jl` [50]. Рассмотрим временной интервал [0, 4]. При использовании численного метода получаем следующие графики (рис. 2.9, 2.10).

Из (рис. 2.9) видно, что траектория численного решения замкнута, следовательно объем фазового портрета сохраняется, как и при аналитическом решении.

Решение модели с использованием PINN

Теперь решим эту систему с помощью библиотеки NeuralPDE.jl. Для этого зададим архитектуру нейронной сети с помощью библиотеки Lux.jl. Используем трехслойную нейронную сеть, на входе расположим 1 нейрон, на выходе — 2,

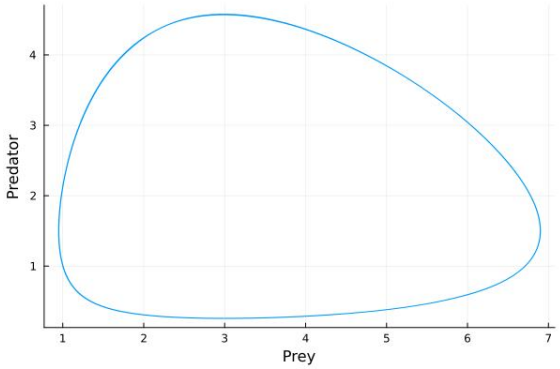


Рис. 2.9. Фазовый портрет модели Лотки-Вольтерры при решении численным методом

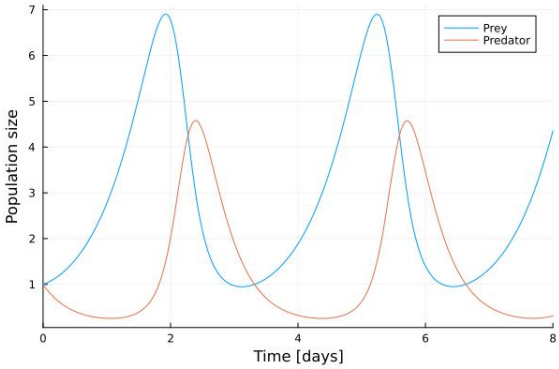


Рис. 2.10. График численного решения модели Лотки-Вольтерры

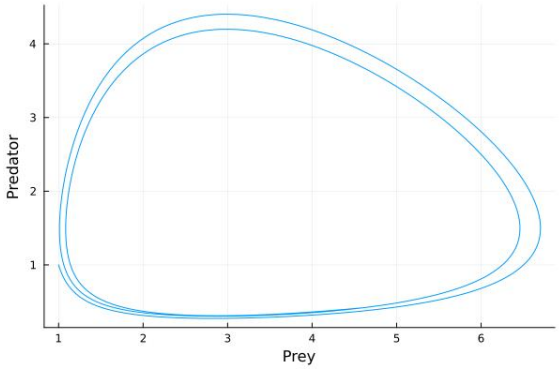


Рис. 2.11. Фазовый портрет модели Лотки-Вольтерры при решении с использованием PINN

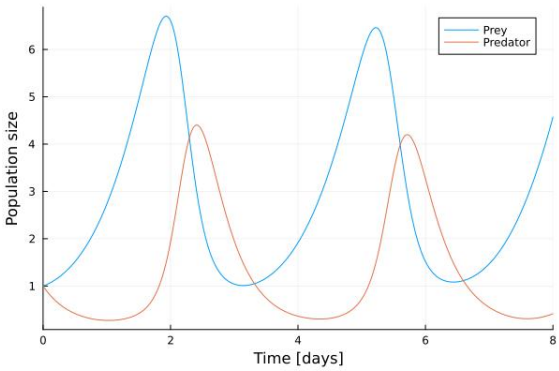


Рис. 2.12. График решения модели Лотки-Вольтерры с использованием PINN

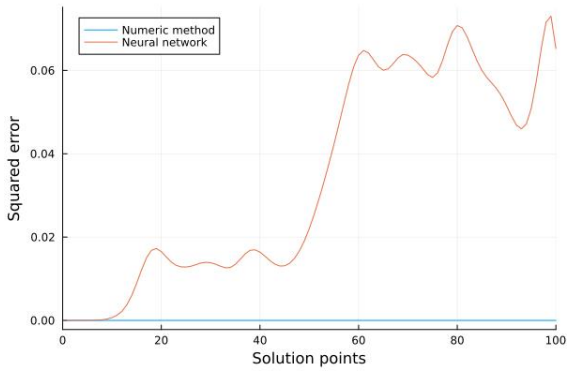


Рис. 2.13. Графики ошибок решений

на промежуточном слое – 16. Активационная функция, заданная на первых двух слоях, – гиперболический тангенс. Также используем алгоритм Бройдена — Флетчера — Гольдфарба — Шанноoптимизатор (BFGS) из библиотеки OptimizationOptimJL.jl. Зададим NeuralPDE.NNODE() – алгоритм решения обыкновенных дифференциальных уравнений с помощью нейронной сети. Это специализация нейронной сети, основанной на физике, которая используется в качестве решателя стандартной задачи ОДУ:

```
chain = Lux.Chain(Lux.Dense(1, 16, tanh),  Lux.Dense(16, 16,tanh),
 ↪  Lux.Dense(16, 2))
opt = OptimizationOptimJL.BFGS()
alg = NeuralPDE.NNODE(chain, opt)
```

Далее мы вызываем `solve` точно так же, как с любыми другими ODEProblem. Включим verbose, чтобы можно было видеть потери с течением времени в процессе обучения. Поставим максимальное количество эпох (итераций) равным 1000:

```
sol = solve(prob, alg, verbose = true, abstol=1e-8, maxiters = 1000)
```

В результате получаем следующие графики (рис. 2.11, 2.12).

Сравнение методов

Сравним фазовую траекторию, полученную с помощью численного решения и с помощью нейронной сети с первым интегралом системы. Для этого построим графики квадратов ошибок (рис. 2.13).

Из графика видно, что решение рассматриваемой системы, полученное численными методами, более точное по сравнению с решением с помощью PINN.

Также сравним производительности обоих методов. Для этого используем пакет BenchmarkTools.jl. Оценим производительность численного метода, посмотрев время и память, затраченные на вычисления:

```
233.154 μs (7052 allocations: 567.62 KiB)
```

Выполним аналогичную оценку для библиотеки NeuralPDE.jl:

```
2463.046 s (3569138613 allocations: 2682.28 GiB)
```

Нейронные сети требуют гораздо больше ресурсов и времени по сравнению с численными методами.

2.4.7. Модель SIR

Рассмотрим систему ДУ, решения которых являются апериодическими.

Компартментальные модели — это очень общий метод моделирования. Их часто применяют для математического моделирования инфекционных заболеваний. Популяция распределяется по отсекам с метками, например S, I или R (восприимчивый, инфекционный или выздоровевший). Люди могут перемещаться между отсеками.

Модель SIR — одна из самых простых компартментальных моделей [69], и многие модели являются производными от этой базовой формы. Модель состоит из трех отделений:

— S: Число восприимчивых людей. Когда восприимчивый и заразный человек вступают в «инфекционный контакт», восприимчивый человек заражается болезнью и переходит в инфекционный отсек.

— I: Число заразных. Это лица, которые были инфицированы и способны заразить восприимчивых лиц.

— R: Количество удаленных (и неуязвимых) или умерших особей. Это лица, которые были инфицированы и либо выздоровели от болезни и попали в удаленный отсек, либо умерли. Предполагается, что число смертей незначительно по отношению к общей численности населения. Этот отсек также можно назвать «восстановленным» или «устойчивым».

До того, как число заболевших не превышает критического значения I^*, считаем, что все больные изолированы и не заражают здоровых. Когда $I(t) > I^*$, тогда инфицирование способны заражать восприимчивых к болезни особей.

Система SIR без динамики жизнедеятельности (рождения и смерти, иногда называемой демографией) может быть выражена следующей системой обыкновенных дифференциальных уравнений:

$$\begin{cases} \frac{dS}{dt} = -\frac{\beta IS}{N}, \\ \frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I, \\ \frac{dR}{dt} = \gamma I, \end{cases}$$

где S – численность восприимчивой популяции, I – численность инфицированных, R – численность удаленной популяции (в результате смерти или выздоровления), и N — это сумма этих трёх, а β и γ - это коэффициенты заболеваемости и выздоровления соответственно.

Решим систему численным методом `Tsit5()` с шагом 0.1, используя библиотеку `DifferentialEquations.jl` [50]. Рассмотрим временной интервал $[0, 50]$ и начальные значения $S = 990.0$, $I = 10.0$, $R = 0.0$. В результате получаем следующий график решения (рис. 2.14.

Теперь решим эту систему с помощью библиотеки NeuralPDE.jl. Используем трехслойную нейронную сеть, на входе расположим 1 нейрон, на выходе – 3, на промежуточном слое – 32. Активационная функция, заданная на первых двух слоях, – сигмоида. Также используем алгоритм Бройдена — Флетчера — Гольдфарба — Шанноптимизатор (BFGS) из библиотеки OptimizationOptimJL.jl.

```
chain = Lux.Chain(Lux.Dense(1, 32, σ), Lux.Dense(32, 32, σ), Lux.Dense(32, 3))
opt = OptimizationOptimJL.BFGS()
alg = NeuralPDE.NNODE(chain, opt)
```

При установленном максимальном количестве эпох равном 1000 получим следующий график решения (рис. 2.15).

В случае с простой моделью эпидемии решение с помощью нейронной сети основанной на физике показало низкую точность, в связи с чем мы не можем рекомендовать этот инструмент для решения рассмотренной задачи.

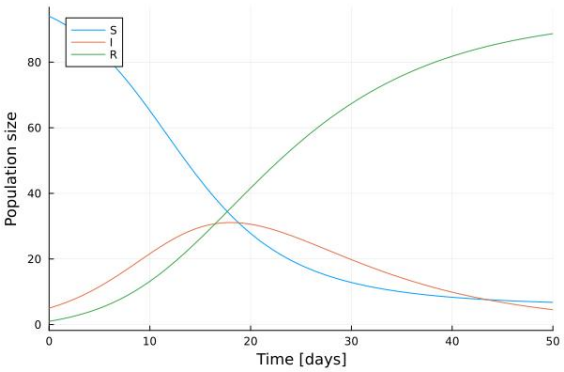


Рис. 2.14. График решения модели SIR при решении численным методом

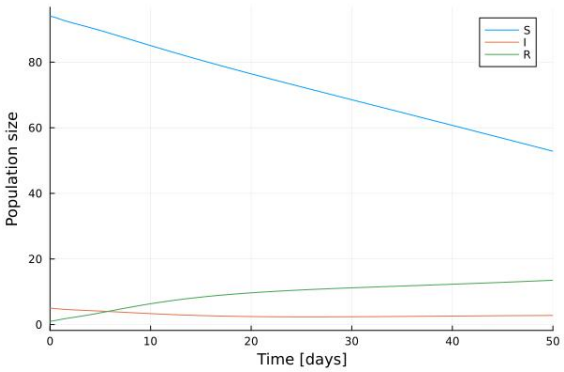


Рис. 2.15. График решения модели SIR с использованием PINN

Глава 3. Моделирование трансформирующих оптических сред

3.1. Реализация метода эйконала на основе численного метода FSM

3.1.1. Расчет методом FSM на языке Julia

Для численного моделирования линз методом FSM мы использовали пакет Eikonal [17] для языка программирования Julia [21; 31; 44]. Пакет Eikonal представляет собой небольшую библиотеку, в которой реализованы методы быстрого заметания (Fast Sweeping) и и быстрой прогонки (Fast Marching). Пакет зарегистрирован в официальном репозитории пакетов Julia и может быть установлен стандартными методами.

Оба метода реализованы для произвольной размерности и их исходный код умещается в одном исходном файле. Для такого небольшого пакета документации довольно подробная и позволяет сравнительно быстро разобраться в функционале пакета. Также в наличии набор тестов, которые также могут послужить иллюстративными примерами.

Рассмотрим использование библиотеки Eikonal для вычисления фронтов в случае плоских линз Максвелла и Люнеберга. Кроме данной библиотеки будем использовать наш модуль Lenses, описанный выше, а также пакет SVector.

Вначале импортируем все необходимые модули. Исходный код модуля Lenses считываем с помощью `include`, а затем добавляем в общую область имен с помощью `using`. Модули из официального репозитория импортируем сразу с помощью `using`.

```
include("../src/lense.jl")

# Используем метод FSM, реализованный в библиотеке Eikonal

using Eikonal
using StaticArrays: SVector
using .Lenses
```

Нам понадобится параметрическое уравнение окружности, которое возвращает координаты в виде целых чисел так как модуль Eikonal использует

целочисленную сетку. Зададим его в виде однострочной функции. Использование синтаксиса точки с оператором или функцией (например, `.+`), позволяет применить функцию и оператор сразу ко всем элементам массива или кортежа.

```
# Параметрическое уравнение окружности
circle_xy(center, R, φ) = round.(Int, center .+ (R*cos(φ), R*sin(φ)))
```

Зададим необходимые параметры в виде констант. Каждая константа снабжена строкой документации, поэтому дополнительные комментами и не требуются. Тип линзы выбирается с помощью функции `select_lense` из модуля Lenses, что позволяет выбирать для какой линзы производить вычисления непосредственно при запуске программы на выполнение.

```
# Центр линзы
const center = (500, 500)
# Радиус линзы
const R = 300
# Коэффициент преломления среды
const n₀ = 1.0
# Позиция источника относительно линзы
const source = circle_xy(center, R, pi)

const LENSE = select_lense(length(ARGS)>=1 ? ARGS[1] : "")(R, n₀,
 ↪  SVector(center..., 0.0))
const RAYS = false
```

Далее настраиваем размер сетки (I, J) для аппроксимирующей функции u_{ij}, инициализируем метод с помощью функции `FastSweeping` и инициализируем массив `v`, который в данной библиотеки обозначает значения коэффициента преломления n_{ij} в узлах сетки. Для вычисления значений n_{ij} используем функцию `n` из Lenses.

```
const tsize = (1000, 1000)

fsm = FastSweeping(Float64, tsize)

# Вычисление коэффициента преломления в точках сетки
for x=1:tsize[1], y=1:tsize[2]
    fsm.v[x, y] = Lenses.n(SVector(x, y, 0.0), LENSE)
end
```

На следующем шаге запускаем вычисления. Функция `init!` позволяет задать граничные значения, которые в нашем случае состоят из единственной точки — источника лучей. Результат вычислений запишется в атрибут `fsm.t`, в наших обозначениях это аппроксимирующая сеточная функция u_{ij}.

```
38 init!(fsm, source)

39 println("Подметание")
40 sweep!(fsm, verbose=false)
```

После получения результатов вычислений остается визуализировать фронты и лучи. Мы используем библиотеку Makie, в то время как в примерах к пакету Eikonal используется библиотека Plots. Создаем изображения, оси на нем, контур линзы в виде окружности и отображаем его на координатной плоскости. Источник изображаем в виде точки. Фронты визуализируются тривиально, с помощью функции `contour!`, которая отображает линии уровня для функции от двух переменных u_{ij}. Массивы координат x и y ей можно не передавать, так как они совпадают с индексами i и j поскольку мы задали целочисленную сетку.

```
44 using CairoMakie

45 fig01 = Figure(fontsize=18, pt_per_unit=1)
46 ax01 = Axis(fig01[1, 1], xlabel = L"$x_n$", ylabel = L"$y_n$")

47 ax01.aspect = DataAspect()
48 ax01.autolimitaspect = 1
49 # colsize!(fig01.layout, 1, Aspect(1, 1))
50 rowsize!(fig01.layout, 1, Aspect(1, 1))

51 # Контур линзы виде окружности
52 const lense_contour = Circle(Point2(center .▷ Float64), R)

53 # Рисуем контур линзы
54 lines!(ax01, lense_contour, color=:blue)
```

Визуализация лучей уже менее тривиальная задача. Автор библиотеки Eikonal предусмотрел функцию `ray`, которая вычисляет точки лучей с помощью метода наискорейшего градиентного спуска. Для ее использования следует задать конечную точку луча. За начало луча будет выбран его источник.

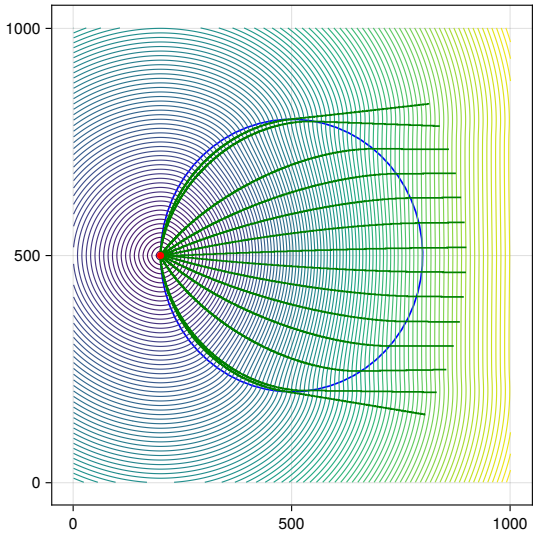


Рис. 3.1. Фронты для линзы Люнеберга

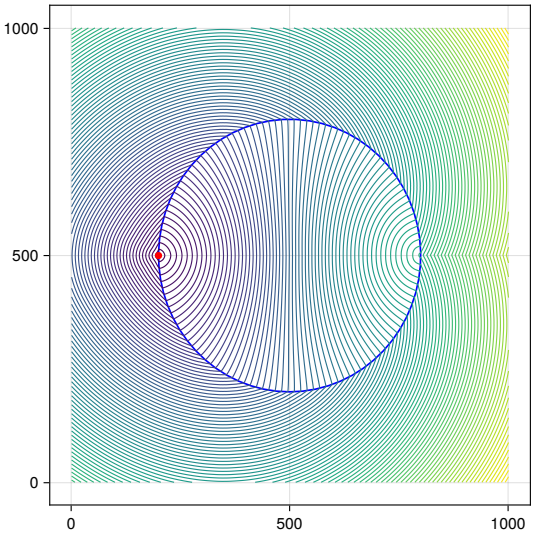


Рис. 3.2. Фронты для линзы Маквелла

```
61 contour!(ax01, fsm.t, levels=100, colormap=:grays)

62 # Источник в виде точки
63 scatter!(ax01, source..., color=:red)

64 # Вычисление лучей
65 if RAYS
```

Следует отметить, что функция `ray` работает не стабильно, так как зачастую ее выполнение сопровождается выходом за границы массива `fsm.t`, даже если конечная точка луча указывается внутри прямоугольной области.

Результат визуализации показан на рисунках 3.1 и 3.2. Здесь следует отметить, что в случае линзы Максвелла применение функции `ray` наталкивается на проблему, которая заключается в следующем.

Исходящие из источника на поверхности линзы лучи должны фокусироваться в точку (фокус), расположенную на диаметрально противоположной от источника стороне линзы и могут выйти из линзы только пройдя эту точку, как это показано на рисунке 3.3. Таким образом при моделировании электромагнитного излучения в виде лучей, вне линзы могут существовать только лучи, прошедшие линзу и вышедшие из точки фокуса. Следует оговорится, что мы рассматриваем пучок лучей, вышедших из точечного источника под углом из интервала $(-\pi/2, \pi/2)$.

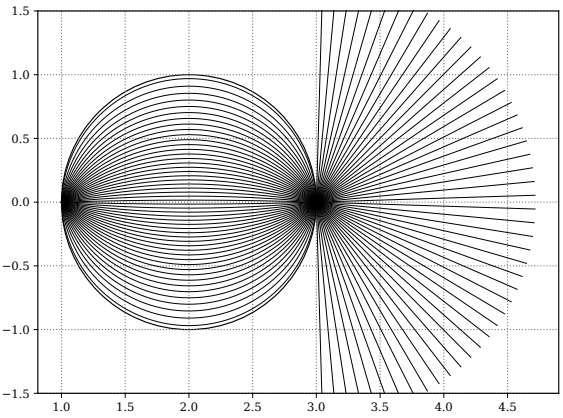


Рис. 3.3. Корректное изображение лучей линзы Максвелла

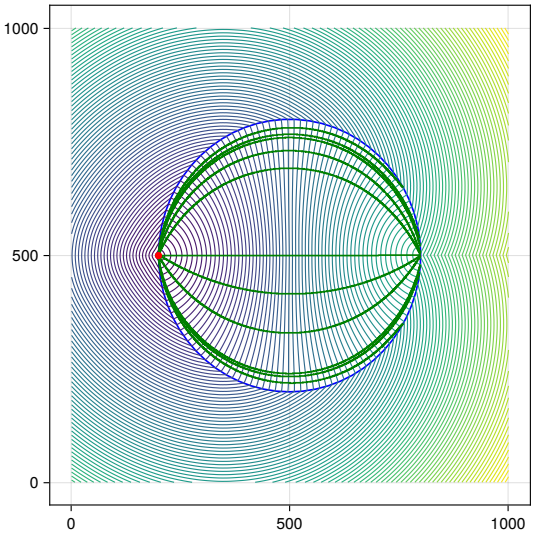


Рис. 3.4. Условно корректное изображение лучей линзы Максвелла

Так как метод FSM аппроксимирует функцию эйконала $u(\mathbf{x})$ в каждой точке сетки, то при таком моделировании электромагнитное излучение будет присутствовать повсюду вне линзы, что можно видеть по изображению фронтов. Поэтому, если задать при вызове функции `ray` параметр `pos` в виде точки за поверхности линзы, то лучи будут нарисованы некорректно, как это показано на рисунке 3.5.

Можно добиться условно корректного изображения, если задавать в виде конечной позиции только те точки сетки, которые лежат исключительно на поверхности линзы.

3.1.2. Обсуждение

При решении уравнения эйконала методом характеристик, оно сводится к системе ОДУ. Каждое решение системы дает траекторию одного луча. Для получения пучка лучей, следует многократно решить полученную систему ОДУ, изменяя каждый раз начальные значения. Можно сказать, что такой подход больше соответствует геометрической оптике, чем волновой. Вычислить точки волновых фронтов непосредственно методом характеристик не представляется возможны и требует дополнительной манипуляции с координатами точек траектории лучей.

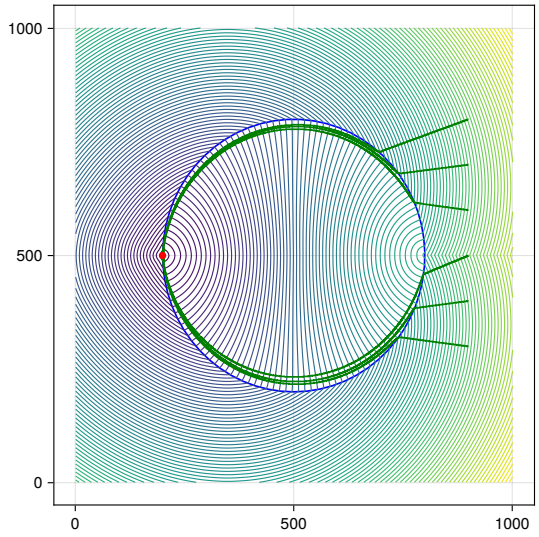


Рис. 3.5. Ошибочное изображение лучей линзы Максвелла, моделируемой методом FSM

Метод FSM, напротив, работает сразу с уравнением эйконала, не требует его преобразования в какую-либо другую форму. Также не требуются вычисления производных от функции коэффициента преломления $n(\mathbf{x})$. Результатом работы метода являются аппроксимированные значения функции $u(\mathbf{x})$ для всех точек сетки. Имея эти значения довольно просто визуализировать фронты путем изображения линий уровня, а вот с визуализацией лучей возникают проблемы, описанные нами выше.

Еще одной особенностью метода FSM является тот факт, что изначально предполагается присутствие электромагнитного поля в каждой точке моделируемой области. Как было показано на рисунках 1.10 и 1.11 при интерпретации электромагнитного излучения в виде лучей, существуют области, куда излучение не проникает.

3.2. Реализация метода эйконала на основе численного метода PINN

3.3. Уравнение эйконала

3.3.1. Уравнение Пуассона

В официальной документации пакета NeuralPDE есть пример кода для решения двумерного уравнения Пуассона

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = -\sin(\pi x)\sin(\pi y),$$

в прямоугольной области, задаваемой отрезками $x \in [0,1]$ и $y \in [0,1]$ и с граничными условиями следующего вида:

$$u(0,y) = 0, u(1,y) = 0,$$
$$u(x,0) = 0, u(x,1) = 0.$$

Вычисления продлились около 30–40 минут и в результате были получены графики, совпадающие с теми, что приведены в официальной документации (рис. 3.6).

3.3.2. Решение уравнение эйконала средствами NeuralPDE

Запишем уравнение эйконала [6; 28; 37; 60] в декартовых координатах на плоскости

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 = n^2(x,y).$$

Функция $n(x,y)$ является кусочно-непрерывной.

Для данного примера была взята функция для линзы Максвелла для двумерного случая [4]:

$$n(r) = \begin{cases} \dfrac{n_0}{1 + \left(\dfrac{r}{R}\right)^2}, & r \leqslant R, \\ n_0, & r > R, \end{cases}$$

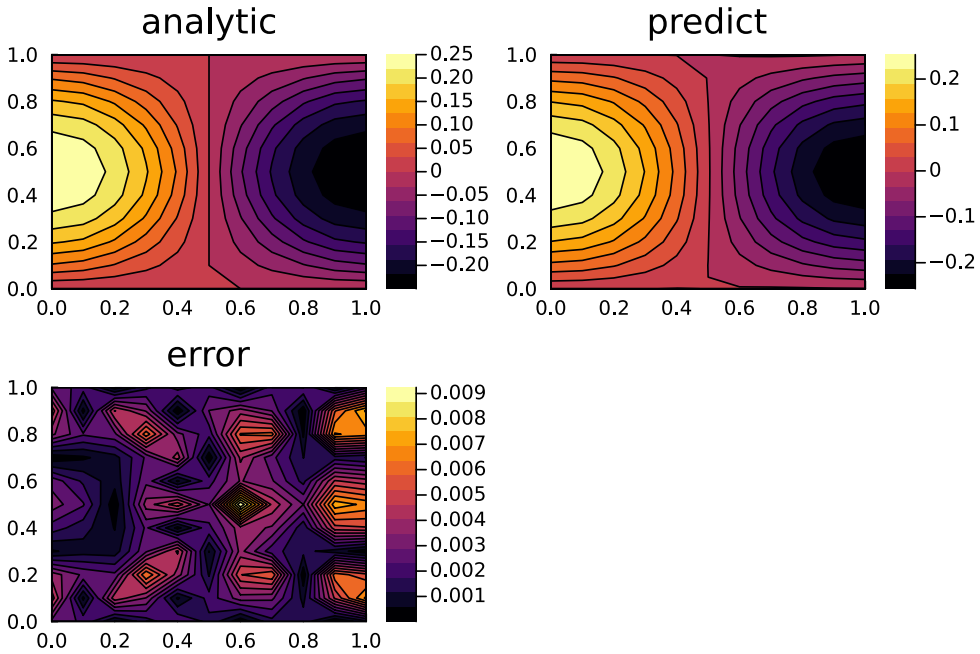


Рис. 3.6. Решение уравнения Пуассона с официального сайта документации NeuralPDE

где $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$ — расстояние от центра координат до точек линзы, которая имеет форму круга. Центр линзы находится в точке с координатами $(x_0, y_0) = (0, 0)$. Радиус линзы $R = 1$, коэффициент преломления среды $n_0 = 1$.

Пример из официальной документации был модифицирован путем замены уравнения Пуассона и его граничных условий на уравнение эйконала. Первая правка заключалась в замене вторых производные на квадраты первых производных:

```
Dx = Differential(x)
Dy = Differential(y)
```

Также для линзы Максвелла:

```
eq = Dx(u(x, y))*Dx(u(x, y)) + Dy(u(x, y))*Dy(u(x, y)) ~ n(x, y)
```

Символ `^` при использовании с функцией `Differential` имеет смысл порядка производной, а не степени, поэтому его пришлось заменить умножением производной самой на себя.

Также были изменены граничные условия и диапазоны значений x, y:

```
# Граничное условие (означает, что точечный источник находится в центре
↪ координат)
```

```
bcs = [
    u(-1, 0) ~ 0.0
]
# Область (x, y)
domains = [x in (-1.0, 2.0), y in (-1.0, 2.0)]
```

Функция `n(x, y)` — после ряда упрощений и удаления почти всех внутренних переменных — выглядела следующим образом:

```
function n(x, y)
  r = hypot(x, y)
  if r <= 1
    return 1 / (1 + r^2)
  else
    return 1
  end
end
```

Однако в таком виде программа не заработала. После инициализации нейронной сети (процесс занимал в среднем около 10 минут) программа падала с ошибкой о не булевых переменных в булевом контексте `ERROR: TypeError: non-boolean (Num) used in boolean context`.

В официальной документации данная ошибка описывается и удалось установить две причины ошибки:

— функции `hypot` в символьном виде не существует, то есть она не определена в пакете Symbolics.jl;

— инструкция `if-else-end` также для символьных значений не поддерживается и её следует заменить на функцию `Base.ifelse` из стандартной библиотеки.

После исправлений функция преломления следующий вид:

```
n(x, y) = ifelse(sqrt(x^2+y^2) <= 1, 1 / (1 + (x^2 + y^2))^2, 1)
```

После чего программа заработала.

Получившиеся результаты после некоторой обработки изображены на рисунке 3.7.

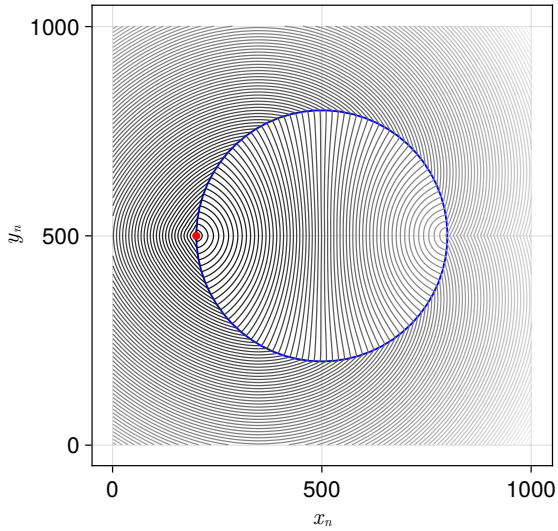


Рис. 3.7. Фронты для линзы Максвелла

3.4. Сравнение методик FSM и PINN

В ходе работы с NeuralPDE были обнаружены следующие недостатки.

— Пакет имеет внушительное количество зависимостей (более сотни). В зависимости входят как другие пакеты Julia, так и сторонние бинарные файлы (утилиты, библиотеки). Прямым следствием является повышенные требования к свободному дисковому пространству и время установки в систему. Однако основная проблема большого количества зависимостей заключается в понижении надежности работы пакета.

— Решаемое уравнение и граничные условия записываются в символьном виде, который крайне ограниченно поддерживает даже стандартные конструкции языка. Даже в случае простой функции $n(x, y)$ не заработала инструкция `if-else-end` и стандартная функция `hypot`. Особенно неочевидным выглядит необходимость замены `if-else-end` на `Base.ifelse`.

— На порядки большее время вычислений по сравнению с классическими методами. Время работы программы с NeuralPDE измеряется десятками минут, в то время как вычисления по классическим численным схемам занимают лишь десятки секунд.

Заключение

В настоящей научно-квалификационной работе была рассмотрена проблема моделирования трансформирующих оптических сред, в частности дифракционных решёток, с использованием численных методов. Работа включала аналитический обзор предметной области, разработку математического аппарата и реализацию двух подходов к численному моделированию — метода быстрого распространения (FSM) и методов, основанных на нейронных сетях (PINN).

Основные результаты работы:

1. Разработан мультимодельный подход к моделированию дифракционных систем в трансформирующих средах, основанный на сочетании численного метода быстрого распространения фронта (Fast Sweeping Method, FSM) и физически информированных нейронных сетей (Physics-Informed Neural Networks, PINN) в инфраструктуре библиотеки NeuralPDE.jl.

2. Сформулирована математическая постановка задачи моделирования распространения лучей в трансформирующих средах на основе уравнения эйконала, адаптированная для использования в среде PINN с учётом физических ограничений и профиля показателя преломления.

3. Выполнена программная реализация и методика визуализации результатов моделирования в виде полей распространения и волновых фронтов для типовых оптических систем (линзы Люнеберга, Максвелла, Итона).

4. Выполнен сравнительный анализ классических численных и нейросетевых подходов (FSM и PINN), демонстрирующие их применимость к различным классам задач моделирования дифракционных систем, включая оценку вычислительных затрат, устойчивости и визуальной интерпретируемости решений.

5. Выработаны практические рекомендации по выбору и комбинированию численных и нейросетевых методов для решения задач лучевой оптики в трансформирующих средах, основанные на обобщении проведённых экспериментов.

Практическая значимость работы заключается в разработке универсального вычислительного подхода, который может быть использован для моделирования оптических компонентов сложной структуры. Полученные результаты обладают как теоретической, так и практической значимостью и могут быть использованы в задачах проектирования оптических элементов, фотонных структур и компонентов радиолокационных систем. В дальнейшем планируется расширить предложенный метод на задачи обратного проектирования и векторной электродинамики, а также интегрировать разработанные алгоритмы в автоматизированные системы синтеза и оптимизации оптических устройств.

Направления дальнейших исследований включают:

1. Расширение моделей на более сложные многослойные и анизотропные оптические структуры.

2. Оптимизацию методов обучения PINN для повышения точности и снижения вычислительных затрат.

3. Интеграцию разработанных методов в комплексные системы проектирования оптических приборов.

Таким образом, выполненное исследование способствует развитию методов математического моделирования в области оптики и открывает перспективы для дальнейшего совершенствования вычислительных технологий в этой области.

Список литературы

1. A comprehensive review of advances in physics-informed neural networks and their applications in complex fluid dynamics / C. Zhao [et al.] // Physics of Fluids. — 2024. — Oct. — Vol. 36, no. 10. — P. 101301. — DOI: 10.1063/5.0226562.
2. A Domain-adaptive Physics-informed Neural Network for Inverse Problems of Maxwell's Equations in Heterogeneous Media / S. Piao [et al.]. — 08/2023. — DOI: 10.48550/arXiv.2308.06436. — arXiv:2308.06436 [cs].
3. *Abbasi M. A. B.*, *Fusco V. F.* Maxwell Fisheye Lens Based Retrodirective Array // Scientific Reports. — 2019. — Nov. — Vol. 9, no. 1. — DOI: 10.1038/s41598-019-52779-1.
4. Algorithm for Lens Calculations in the Geometrized Maxwell Theory / D. S. Kulyabov [et al.] // Saratov Fall Meeting 2017: Laser Physics and Photonics XVIII; and Computational Biophysics and Analysis of Biomedical Data IV. Vol. 10717 / ed. by V. L. Derbov, D. E. Postnov. — Saratov : SPIE, 04/2018. — 107170Y.1–6. — (Progress in Biomedical Optics and Imaging - Proceedings of SPIE). — DOI: 10.1117/12.2315066. — arXiv: 1806.01643.
5. *Bararnia H.*, *Esmaeilpour M.* On the application of physics informed neural networks (PINN) to solve boundary layer thermal-fluid problems // International Communications in Heat and Mass Transfer. — 2022. — Vol. 132. — P. 105890. — DOI: 10.1016/j.icheatmasstransfer.2022.105890.
6. *Bruns H.* Das Eikonal // Abhandlungen der Königlich-Sächsischen Gesellschaft der Wissenschaften. Bd. 21. — Leipzig : S. Hirzel, 1895.
7. Challenges in Training PINNs: A Loss Landscape Perspective / P. Rathore [et al.]. — 06/2024. — DOI: 10.48550/arXiv.2402.01868. — arXiv:2402.01868 [cs].
8. Characterizing possible failure modes in physics-informed neural networks / A. S. Krishnapriyan [et al.]. —.
9. *Chisolm E.* Geometric Algebra. — 2012. — arXiv: 1205.5935. — Pre-published.
10. *Clifford W. K.* Applications of Grassmann's Extensive Algebra // American Journal of Mathematics. — 1878. — Vol. 1, no. 4. — P. 350–358. — DOI: 10.2307/2369379.

11. *Doran C.*, *Lasenby A.* Geometric Algebra for Physicists. — Cambridge : Cambridge University Press, 05.2003. — 578 c. — DOI: 10.1017/cbo9780511807497.

12. *Dorst L.*, *Fontijne D.*, *Mann S.* Geometric algebra for computer science (with errata). — 1st ed. — Morgan Kaufmann, 2007. — (The Morgan Kaufmann Series in Computer Graphics).

13. *E W.*, *Yu B.* The Deep Ritz Method: A deep learning-based numerical algorithm for solving variational problems // Communications in Mathematics and Statistics. — 2018. — Т. 6, № 1. — С. 1—12. — DOI: 10.1007/s40304-018-0127-z.

14. *Eliasof M.*, *Haber E.*, *Treister E.* PDE. —.

15. *Evans L. C.* Partial Differential Equations. T. 19. — American Mathematical Society, 1998. — (Graduate Studies in Mathematics).

16. Evolutionary Optimization of Physics-Informed Neural Networks: Survey and Prospects / J. C. Wong [et al.]. — 03/2025. — DOI: 10.48550/arXiv.2501.06572. — arXiv:2501.06572 [cs].

17. *Févotte F.* Fast Sweeping and Fast Marching methods for the solution of eikonal equations. — Version 0.2.0. — 2024. — URL: https://github.com/triscale-innov/Eikonal.jl (visited on 09/01/2023).

18. Flux The Elegant Machine Learning Stack. — 2025. — URL: https://fluxml.ai/.

19. *Fonseca N. J. G.*, *Tyc T.*, *Quevedo–Teruel O.* A solution to the complement of the generalized Luneburg lens problem // Communications Physics. — 2021. — Vol. 4, no. 1. — DOI: 10.1038/s42005-021-00774-2.

20. Fourier Neural Operator for Parametric Partial Differential Equations / Z. Li [et al.]. — 05/2021. — DOI: 10.48550/arXiv.2010.08895. — arXiv:2010.08895 [cs].

21. *Gevorkyan M. N.*, *Kulyabov D. S.*, *Sevastyanov L. A.* Review of Julia programming language for scientific computing // The 6th International Conference "Distributed Computing and Grid-technologies in Science and Education". — 2014. — P. 27.

22. *Grassmann H. G.* Die Mechanik nach den Principien der Ausdehnungslehre // Mathematische Annalen. — 1877. — Juni. — Jg. 12, Nr. 2. — S. 222–240. — DOI: 10.1007/bf01442659.

23. *Gremaud P. A.*, *Kuster C. M.* Computational Study of Fast Methods for the Eikonal Equation // SIAM Journal on Scientific Computing. — 2006. — Jan. — Vol. 27, no. 6. — P. 1803–1816. — DOI: 10.1137/040605655.

24. High-Performance Symbolic-Numerics via Multiple Dispatch / S. Gowda [et al.] // ACM Commun. Comput. Algebra. — New York, NY, USA, 2022. — Jan. — Vol. 55, no. 3. — P. 92–96. — DOI: 10.1145/3511528.3511535.

25. *Horie M.*, *Mitsume N.* Physics-Embedded Neural Networks: Graph Neural PDE Solvers with Mixed Boundary Conditions. —.

26. *Hornik K.* Approximation capabilities of multilayer feedforward networks // Neural networks. — 1991. — Т. 4, № 2. — С. 251—257.

27. *Jeong W.*, *Whitaker R.* A fast eikonal equation solver for parallel systems // SIAM conference. — 2007. — Vol. 84112. — P. 1–4.

28. *Klein F. C.* Über das Brunssche Eikonal // Zeitscrift für Mathematik und Physik. — 1901. — Jg. 46. — S. 372–375.

29. *Kulyabov D. S.*, *Gevorkyan M. N.*, *Korolkova A. V.* Software Implementation of the Eikonal Equation // Proceedings of the Selected Papers of the 8th International Conference "Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems" (ITTMM-2018), Moscow, Russia, April 16, 2018. Vol. 2177 / ed. by D. S. Kulyabov, K. E. Samouylov, L. A. Sevastianov. — Moscow, 04/2018. — P. 25–32. — (CEUR Workshop Proceedings).

30. *Lasenby A.*, *Doran C.*, *Arcaute E.* Applications of Geometric Algebra in Electromagnetism, Quantum Theory and Gravity // Clifford Algebras. Vol. 34 / ed. by R. Abłamowicz. — Birkhäuser Boston, 2004. — Chap. 30. — (Progress in Mathematical Physics). — DOI: 10.1007/978-1-4612-2044-2_30.

31. *Lauwens B.*, *Downey A.* Think Julia : How to Think Like a Computer Scientist. — O'Reilly Media, Inc., 2019. — 229 p.

32. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators / L. Lu [et al.] // Nature Machine Intelligence. — 2021. — Mar. — Vol. 3, no. 3. — P. 218–229. — DOI: 10.1038/s42256-021-00302-5.

33. Learning the Intrinsic Dynamics of Spatio-Temporal Processes Through Latent Dynamics Networks / F. Regazzoni [и др.] // Nature Computational Science. — 2024.

34. LuxDL DocsElegant and Performant Deep Learning in JuliaLang. — 2025. — URL: https://lux.csail.mit.edu/stable/.

35. Magnetostatics and micromagnetics with physics informed neural networks / A. Kovacs [et al.] // Journal of Magnetism and Magnetic Materials. — 2022. — Apr. — Vol. 548. — P. 168951. — DOI: 10.1016/j.jmmm.2021.168951.

36. Mesh-based GNN surrogates for time-independent PDEs / R. J. Gladstone [et al.] // Scientific Reports. — 2024. — Feb. — Vol. 14, no. 1. — P. 3394. — DOI: 10.1038/s41598-024-53185-y.

37. Methodological derivation of the eikonal equation / A. V. Fedorov [и др.] // Discrete and Continuous Models and Applied Computational Science. — 2023. — Дек. — Т. 31, № 4. — С. 399—418. — DOI: 10.22363/2658-4670-2023-31-4-399-418.

38. *Mishra S., Molinaro R.* Estimates on the Generalization Error of Physics-Informed Neural Networks for Approximating PDEs // IMA Journal of Numerical Analysis. — 2022. — Т. 42, № 2. — С. 981—1022. — DOI: 10.1093/imanum/drab069.

39. *Mitusch S. K., Funke S. W., Kuchta M.* Hybrid FEM-NN models: Combining artificial neural networks with the finite element method // Journal of Computational Physics. — 2021. — Dec. — Vol. 446. — P. 110651. — DOI: 10.1016/j.jcp.2021.110651.

40. ModelingToolkit: A Composable Graph Transformation System For Equation-Based Modeling / Y. Ma [et al.]. — 2021. — arXiv: 2103.05244 [cs.MS].

41. NeuralPDE: Automating Physics-Informed Neural Networks (PINNs) with Error Approximations / K. Zubov [et al.]. — 2021. — DOI: 10.48550/ARXIV.2107.09443.

42. *Nohra M., Dufour S.* Physics-Informed Neural Networks for the Numerical Modeling of Steady-State and Transient Electromagnetic Problems with Discontinuous Media. — 06/2024. — DOI: 10.48550/arXiv.2406.04380. — arXiv:2406.04380 [physics].

43. Numerical analysis of eikonal equation / D. S. Kulyabov [et al.] // Saratov Fall Meeting 2018: Laser Physics, Photonic Technologies, and Molecular Modeling. Vol. 11066 / ed. by V. L. Derbov. — Saratov : SPIE, 06/2019. — P. 56. — (Progress

in Biomedical Optics and Imaging - Proceedings of SPIE). — DOI: 10.1117/12.2525142. — arXiv: 1906.09467.

44. *Phillips L.* Practical Julia : A Hands-On Introduction for Scientific Minds. — No Starch Press, 10/31/2023. — 528 p.

45. Physics informed neural networks for fluid flow analysis with repetitive parameter initialization / J. Lee [et al.] // Scientific Reports. — 2025. — May. — Vol. 15, no. 1. — P. 16740. — DOI: 10.1038/s41598-025-99354-5.

46. Physics-informed learning in artificial electromagnetic materials / Y. Deng [et al.] // Applied Physics Reviews. — 2025. — Mar. — Vol. 12, no. 1. — P. 011331. — DOI: 10.1063/5.0232675.

47. Physics-informed neural network for inversely predicting effective electric permittivities of metamaterials / P. Pillai [et al.]. —.

48. Physics–Informed Neural Networks (PINNs) as intelligent computing technique for solving partial differential equations: Limitation and Future prospects / W. Zhang [et al.]. —.

49. PINNacle / Z. Hao [et al.]. — 10/2023. — DOI: 10.48550/arXiv.2306.08827. — arXiv:2306.08827 [cs].

50. *Rackauckas C., Nie Q.* DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia // Journal of Open Research Software. — 2017. — Т. 5, № 1. — С. 15—25. — DOI: 10.5334/jors.151.

51. *Raissi M., Perdikaris P., Karniadakis G.* Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations // Journal of Computational Physics. — 2019. — Feb. — Vol. 378. — P. 686–707. — DOI: 10.1016/j.jcp.2018.10.045.

52. *Raissi M., Perdikaris P., Karniadakis G.* Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics // J. Comput. Phys. — 2018. — DOI: 10.1016/j.jcp.2018.10.045.

53. *Rodrigues Jr W. A., Oliveira E. C. de*. The Many Faces of Maxwell, Dirac and Einstein Equations : A Clifford Bundle Approach. Vol. 922. — Springer International Publishing, 2016. — 587 p. — (Lecture Notes in Physics). — DOI: 10.1007/978-3-319-27637-3.

54. *Rosén A.* Geometric Multivector Analysis : From Grassmann to Dirac. — Springer International Publishing, 2019. — 465 p. — DOI: 10.1007/978-3-030-31411-8.

55. *Rudolph M., Kurz S., Rakitsch B.* Hybrid modeling design patterns // Journal of Mathematics in Industry. — 2024. — Mar. — Vol. 14, no. 1. — P. 3. — DOI: 10.1186/s13362-024-00141-0.

56. *Sabbata V. de, Datta B. K.* Geometric Algebra and Applications to Physics. — Taylor & Francis, 12/2006. — 184 p. — DOI: 10.1201/9781584887737.

57. SciML Open Source Scientific Machine Learning. — 2021. — URL: https://github.com/SciML.

58. SciML Open Source Scientific Machine Learning. — 2025. — URL: https://github.com/SciML/NeuralPDE.jl.

59. *Sirignano J., Spiliopoulos K.* DGM: A deep learning algorithm for solving partial differential equations // Journal of Computational Physics. — 2018. — T. 375. — C. 1339—1364. — DOI: 10.1016/j.jcp.2018.08.029.

60. Solving the eikonal equation by the FSM method in Julia language / C. A. Stepa [et al.] // Discrete and Continuous Models and Applied Computational Science. — 2024. — Vol. 32, no. 1. — P. 48–60. — DOI: 10.22363/2658-4670-2024-32-1-48-60.

61. Symbolic-numeric approach for the investigation of kinetic models / E. A. Demidova [et al.] // Discrete and Continuous Models and Applied Computational Science. — 2024. — Vol. 32, no. 3. — P. 306–318. — DOI: 10.22363/2658-4670-2024-32-3-306--318.

62. *Toomey D.* Learning Jupyter. — Packt Publishing Ltd., 2016. — 305 p.

63. *Torres E., Schiefer J., Niepert M.* Adaptive Physics-informed Neural Networks: A Survey. — 03/2025. — DOI: 10.48550/arXiv.2503.18181. — arXiv:2503.18181 [cs].

64. Understanding the Role of Autoencoders for Stiff Dynamical Systems Using Information Theory / V. V. [и др.]. — 2024. — arXiv:2401.12345. arXiv preprint.

65. *Wang S., Sankaran S., Perdikaris P.* Respecting causality is all you need for training physics-informed neural networks. — 03/2022. — DOI: 10.48550/arXiv.2203.07404. — arXiv:2203.07404 [cs].

66. *Zeng Y.*, *Werner D. H.* Two-dimensional inside-out Eaton Lens : Design technique and TM-polarized wave properties // Optical Express. — 2012. — Jan. — Vol. 20, no. 3. — P. 2335–2345. — DOI: 10.1364/OE.20.002335.

67. *Zhao H.* A fast sweeping method for Eikonal equations // Mathematics of Computation. — 2004. — May. — Vol. 74, no. 250. — P. 603–627. — DOI: 10.1090/s0025-5718-04-01678-3.

68. *Борн М.*, *Вольф Э.* Основы оптики : пер. с англ. — 2-е изд. — Москва : Наука, 1973. — 720 с. — Пер. по изд.: *Born M.*, *Wolf E.* Principles of Optics. — 7th. — Cambridge University Press, 1999. — 952 p.

69. *Братусь А. С.*, *Новожилов А. С.*, *Платонов А. П.* Динамические системы и модели биологии. — М. : Физматлит, 2010. — 400 с.

70. *Вольтерра В.* Математическая теория борьбы за существование : пер. с фр. — Москва : Наука, 1976. — 288 с. — Пер. по изд.: *Volterra V.* Leçons sur la Théorie mathématique de la lutte pour la vie. — Paris : Gauthiers-Villars, 1931.

71. *Иванов Д. И.*, *Иванов И. Э.*, *Крюков И. А.* Алгоритмы приближенного решения некоторых задач прикладной геометрии, основанные на уравнении типа Гамильтона-Якоби // Журнал вычислительной математики и математической физики. — 2005. — Т. 45, вып. 8. — С. 1345—1358.

72. *Кабанихин С. И.*, *Криворотько О. И.* Численное решение уравнения эйконала // Сибирские электронные математические известия. — 2013. — Т. 10. — С. 28—34.

73. *Ландау Л. Д.*, *Лифшиц Е. М.* Теоретическая физика : Теория поля : в 10 т. Т. 2. — 8-е. — Москва : Физматлит, 2012. — 536 с.

74. *Сивухин Д. В.* О Международной системе физических величин // Успехи физических наук. — 1979. — Т. 129, № 10. — С. 335—338. — DOI: 10.3367/UFNr.0129.197910h.0335.

75. *Стрэттон Д. А.* Теория электромагнетизма. — М.-Л. : ГИТТЛ, 1948.

Приложение А. Аннотированный код реализации численного метода FSM

А.1. Реализация метода FSM

```
module FSM


export fsm

using Base.Iterators: countfrom, takewhile

"""Решение уравнения на каждом шагу
n – значение коэффициента преломления,
h – шаг метода,
a, b –
"""
function gauss_seidel(n, h, a, b)
  if abs(a - b) >= n * h
    return min(a, b) + n * h
  else
    return 0.5(a + b) + 0.5sqrt(2(n*h)^2 - (a - b)^2)
  end
end


# Один проход метода, модифицирует массив u
function sweep!(u, n, G, i, j, Nx, Ny, h)
  if G[j, i]
    u[j, i] = u[j, i]
  # I группа
  elseif 2<=i<=(Nx-1) && 2<=j<=(Ny-1)
    u_xmin = min(u[j, i-1], u[j, i+1])
    u_ymin = min(u[j-1, i], u[j+1, i])
    u[j,i] = min(u[i,j], gauss_seidel(n[j, i], h, u_xmin, u_ymin))
  # II группа, левая граница
  elseif i==1 && 2<=j<=(Ny-1)
    u_ymin = min(u[j-1, i], u[j+1, i])
    u[j,i] = min(u[i,j], gauss_seidel(n[j, i], h, u[j, 2], u_ymin))
  # II группа, правая граница
  elseif i==Nx && 2<=j<=(Ny-1)
    u_ymin = min(u[j-1, i], u[j+1, i])
    u[j,i] = min(u[i,j], gauss_seidel(n[j, i], h, u[j, Nx-1], u_ymin))
```

```
  # II группа, нижняя граница
  elseif 2<=i<=(Nx-1) && j==1
    u_xmin = min(u[j, i-1], u[j, i+1])
    u[j,i] = min(u[i,j], gauss_seidel(n[j, i], h, u_xmin, u[2, i]))
  # II группа, верхняя граница
  elseif 2<=i<=(Nx-1) && j==Ny
    u_xmin = min(u[j, i-1], u[j, i+1])
    u[j,i] = min(u[i,j], gauss_seidel(n[j, i], h, u_xmin, u[Ny-1, i]))
  # III группа, левая нижняя угловая точка
  elseif i==1 && j==1
    u[j,i] = min(u[i,j], gauss_seidel(n[j, i], h, u[1, 2], u[2, 1]))
  # III группа, левая верхняя угловая точка
  elseif i==1 && j==Ny
    u[j,i] = min(u[i,j], gauss_seidel(n[j, i], h, u[Ny, 2], u[Ny-1, 1]))
  # III группа, правая нижняя угловая точка
  elseif i==Nx && j==1
    u[j,i] = min(u[i,j], gauss_seidel(n[j, i], h, u[1, Nx-1], u[2, Nx]))
  # III группа, правая верхняя угловая точка
  elseif i==Nx && j==Ny
    u[j,i] = min(u[i,j], gauss_seidel(n[j, i], h, u[Ny, Nx-1], u[Ny-1, Nx]))
  end
end

"""
Аргументы функции
nf – коэффициент преломления в уравнении эйконала в виде функции.
u – сеточная функция.
G – логический массив, задающий сетку. В граничных точках принимает
  значение 1, во всех остальных 0
Nx, Ny – число узлов сетки по осям x и y.
Gx, Gy – координаты границы сетки по осям Ox и Oy.
N – число итераций метода.
"""
function fsm(nf, G, h, NSweeps)

  println("Инициализация")
  # число столбцов – количество точек по x
  # число строк – количество точек по y
  Ny, Nx = size(G)

  n = Matrix{Float64}(undef, Ny, Nx)

  X = collect(takewhile(<=((Nx-1)*h), countfrom(0, h)))
  Y = collect(takewhile(<=((Ny-1)*h), countfrom(0, h)))
```

```
  u = Matrix{Float64}(undef, Ny, Nx)
  # Для граничных точек сетки «замораживаем» значение u[i, j]
  for i=1:Nx, j=1:Ny
    if G[j, i]
      u[j, i] = 0.0
    else
      # Остальным присваиваем некоторое, достаточно большое значение
      u[j, i] = 1000.0
    end
    n[j, i] = nf(X[i], Y[j])
  end

  println("Вычисления")

  for s = 1:NSweeps
    println("Проход №$s")

    for i=1:1:Nx, j=1:1:Ny
      sweep!(u, n, G, i, j, Nx, Ny, h)
    end

    for i=Nx:-1:1, j=1:1:Ny
      sweep!(u, n, G, i, j, Nx, Ny, h)
    end

    for i=Nx:-1:1, j=Ny:-1:1
      sweep!(u, n, G, i, j, Nx, Ny, h)
    end

    for i=1:1:Nx, j=Ny:-1:1
      sweep!(u, n, G, i, j, Nx, Ny, h)
    end
  end
  return X, Y, u
end

end #module fsm
```

```
include("../src/parameters.jl")
include("../src/lense.jl")
include("../fsm/fsm.jl")

using StaticArrays: SVector
```

```
using .FSM: fsm
using .Lenses

# Используем собственную реализацию FSM

const h = 0.01
const NSweeps = 10

const G = zeros(Bool, 1000, 1000)

# Центр линзы
const centre = (5.0, 5.0)
const R = 3.0
# Положение точечного источника
const source = (centre .- R / sqrt(2))
const LENSE = Maxwell(R, 1.0, SVector(centre..., 0.0))
# Точечный источник, на диагонали
# чтобы получить индексы мы делим на шаг h
G[round.(Int, (centre .- R / sqrt(2)) ./ h)...] = true

function nf(x, y)
  return Lenses.n(SVector(x, y, 0.0), LENSE)
end

X, Y, u = fsm(nf, G, h, NSweeps)


using CairoMakie

fig01 = Figure(size=(500, 500))
ax01 = Axis(fig01[1, 1])

# Рисуем контур линзы виде окружности
const lense_contour = Circle(Point2(centre...), R)
lines!(ax01, lense_contour, color=:blue)

contour!(ax01, X, Y, u, levels=100)
scatter!(ax01, source..., color=:red)

save("fsm.png", fig01)

#=
Получаемая картинка совпадает с библиотекой fsm
но происходит дублирование источников, с чем надо разобраться.
```

```
Также возникло сомнение в правильности рисования фронтов
при решении методом характеристик
=#
```

A.2. Решение уравнения эйконала с помощью модуля Eikonal

```
include("../src/lense.jl")

# Используем метод FSM, реализованный в библиотеке Eikonal

using Eikonal
using StaticArrays: SVector
using .Lenses

# Параметрическое уравнение окружности
circle_xy(center, R, φ) = round.(Int, center .+ (R*cos(φ), R*sin(φ)))

# Все координаты задаем в виде целых чисел, так как используется целочисленная
 ↪  сетка

# Центр линзы
const center = (500, 500)
# Радиус линзы
const R = 300
# Коэффициент преломления среды
const n₀ = 1.0
# Позиция источника относительно линзы
const source = circle_xy(center, R, pi)

const LENSE = select_lense(length(ARGS)>=1 ? ARGS[1] : "")(R, n₀,
 ↪  SVector(center..., 0.0))
const RAYS = false

println("Выбрана линза $(LENSE.name)")

# Размер сетки
const tsize = (1000, 1000)

fsm = FastSweeping(Float64, tsize)

# Вычисление коэффициента преломления в точках сетки
for x=1:tsize[1], y=1:tsize[2]
    fsm.v[x, y] = Lenses.n(SVector(x, y, 0.0), LENSE)
```

```
26 end

27 init!(fsm, source)

28 println("Подметание")
29 sweep!(fsm, verbose=false)

30 println("Рисование")
31 using CairoMakie

32 fig01 = Figure(fontsize=18, pt_per_unit=1)
33 ax01 = Axis(fig01[1, 1], xlabel = L"$x_n$", ylabel = L"$y_n$")

34 ax01.aspect = DataAspect()
35 ax01.autolimitaspect = 1
36 # colsize!(fig01.layout, 1, Aspect(1, 1))
37 rowsize!(fig01.layout, 1, Aspect(1, 1))

38 # Контур линзы виде окружности
39 const lense_contour = Circle(Point2(center .▷ Float64), R)

40 # Рисуем контур линзы
41 lines!(ax01, lense_contour, color=:blue)

42 # Фронты в виде контуров функции от двух переменных u(x, y)
43 contour!(ax01, fsm.t, levels=100, colormap=:grays)

44 # Источник в виде точки
45 scatter!(ax01, source..., color=:red)

46 # Вычисление лучей
47 if RAYS
48   #for θ = [-pi/6, -pi/24, -pi/48, -pi/960, 0, pi/960, pi/300, pi/200, pi/48,
     ↪  pi/24, pi/6]
49   # pos = circle_xy(center, R, θ)
50   for pos in ((900, 300), (900, 400), (900, 500),
51     (900, 600), (900, 700), (900, 800))
52     r1 = ray(fsm.t, pos)
53     lines!(ax01, r1, color=:green)
54   end
55 end


56 println("Сохранение")
```

```
if RAYS
  save("img/FSM/Линза_$(LENSE.name)_фронты_и_лучи.pdf", fig01)
else
  save("img/FSM/Линза_$(LENSE.name)_фронты.pdf", fig01)
end
```

```
include("../src/lense.jl")

# Используем метод FSM, реализованный в библиотеке Eikonal
# для расчета линзы Итона. Местоположение источников и сама
# конфигурация линзы для Итона настолько отличается, что
# имеет смысл написать отдельную программу

using Eikonal
using StaticArrays: SVector
using .Lenses

# Параметрическое уравнение окружности
circle_xy(center, R, φ) = round.(Int, center .+ (R*cos(φ), R*sin(φ)))

# Все координаты задаем в виде целых чисел, так как используется целочисленная
 ↪  сетка

# Размер сетки
const tsize = (1000, 1000)
# Центр линзы
const center = (500, 500)
# Радиус линзы
const R = 250
# Коэффициент преломления среды
const n₀ = 1.0
# Позиция источника относительно центра линзы
const source = (center[1]+125, center[2])
# Местонахождение фокуса (симметрично относительно центра линзы)
const focus =  (center[1]-125, center[2])

const LENSE = Eaton(R, n₀, SVector(center..., 0.0))
const RAYS = true

println("Выбрана линза $(LENSE.name)")


fsm = FastSweeping(Float64, tsize)

# Вычисление коэффициента преломления в точках сетки
```

```
for x=1:tsize[1], y=1:tsize[2]
    fsm.v[x, y] = Lenses.n(SVector(x, y, 0.0), LENSE)
end

init!(fsm, source)

println("Подметание")
sweep!(fsm, verbose=false)

println("Рисование")
using CairoMakie

fig01 = Figure(fontsize=18, pt_per_unit=1)
ax01 = Axis(fig01[1, 1], xlabel = L"$x_n$", ylabel = L"$y_n$")

ax01.aspect = DataAspect()
ax01.autolimitaspect = 1
# colsize!(fig01.layout, 1, Aspect(1, 1))
rowsize!(fig01.layout, 1, Aspect(1, 1))

# Контур линзы виде окружности
const lense_contour01 = Circle(Point2(center .▷ Float64), R)
const lense_contour02 = Circle(Point2(center .▷ Float64), 2R)

# Рисуем контур линзы
lines!(ax01, lense_contour01, color=:blue)
lines!(ax01, lense_contour02, color=:blue)

# Фронты в виде контуров функции от двух переменных u(x, y)
contour!(ax01, fsm.t, levels=100, colormap=:grays)

# Для линзы Итона адекватно восстановить лучи не получается
if RAYS
  #for θ = [-pi/6, -pi/24, -pi/48, -pi/960, 0, pi/960, pi/300, pi/200, pi/48,
   ↪  pi/24, pi/6]
  # pos = circle_xy(center, R, θ)
  for pos in ((400, 300), (400, 400), (400, 500),
    (400, 600), (400, 700), (400, 800))
    r1 = ray(fsm.t, pos)
    lines!(ax01, r1, color=:green)
  end
end

# Источники в виде точек
```

```
62 scatter!(ax01, source..., color=:red)
63 scatter!(ax01, focus..., color=:blue)

64 println("Сохранение")
65 if RAYS
66   save("img/FSM/Линза_$(LENSE.name)_фронты_и_лучи.pdf", fig01)
67 else
68   save("img/FSM/Линза_$(LENSE.name)_фронты.pdf", fig01)
69 end
```

Приложение В. Аннотированный код реализации численного метода PINN

```
using NeuralPDE, Lux, Optimization, OptimizationOptimJL, LineSearches, Plots
using ModelingToolkit: Interval

@parameters x y
@variables u(..)
Dx = Differential(x)
Dy = Differential(y)

#=
function n(x, y)
  r = hypot(x, y)
  if r <= 1
    return 1 / (1 + r^2)
  else
    return 1
  end
end
=#
# Работает данная версия
n(x, y) = ifelse(sqrt(x^2+y^2) <= 1, 1 / (1 + (x^2 + y^2))^2, 1)

# Эйконал с линзой Максвелла
eq = Dx(u(x, y))*Dx(u(x, y)) + Dy(u(x, y))*Dy(u(x, y)) ~ n(x, y)

bcs = [
    u(-1, 0) ~ 0.0
]
# Область (x, y)
domains = [x ∈ (-1.0, 2.0), y ∈ (-1.0, 2.0)]

# Создание нейронной сети
dim = 2 # размерность
chain = Chain(Dense(dim, 16, σ), Dense(16, 16, σ), Dense(16, 1))

# Дискретизация
discretization = PhysicsInformedNN(
    chain, QuadratureTraining(; batch = 200, abstol = 1e-6, reltol = 1e-6))

@named pde_system = PDESystem(eq, bcs, domains, [x, y], [u(x, y)])
prob = discretize(pde_system, discretization)
```

```
#Функция обратного вызова (Callback)
callback = function (p, l)
    println("Current loss is: $l")
    return false
end

# Оптимизатор
opt = LBFGS(linesearch = BackTracking())
res = solve(prob, opt, maxiters = 1000)
phi = discretization.phi

# Массив значений функции аналитического решения
dx = 0.05
xs, ys = [infimum(d.domain):(dx / 10):supremum(d.domain) for d in domains]


u_predict = reshape([first(phi([x, y], res.u)) for x in xs for y in ys],
    (length(xs), length(ys)))
```

Приложение C. Сведения из векторного анализа

C.1. Основные определения

Если в каждой точке P определённой пространственной области евклидова пространства $\mathbb{R}^n$ связанна некоторая скалярная или векторная величина, то говорят, что задано *поле* (*скалярное* или *векторное*).

— Примерами векторных полей могут служить поле скоростей $\mathbf{v}(x, y, z)$, поле сил $\mathbf{F}(x, y, z)$, поле электрической напряженности $\mathbf{E}(x, y, z)$.

— Примеры скалярных полей: поле температур $T(x, y, z)$, поле электрического потенциала $\varphi(x, y, z)$.

Везде далее рассматривается трёхмерное точечное евклидово пространство, на котором введена декартова система координат. Орты (базисные векторы) этой системы координат обозначим как $\langle \mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z \rangle$. Координаты точки задаются радиус-вектором $\mathbf{r} = (x, y, z)^T$, который откладывается от начала координат O. Наряду с обозначениями координат x, y, z иногда удобно пользоваться индексами: x^1, x^2, x^3, а также записывать радиус-вектор в виде $\mathbf{x} = (x^1, x^2, x^3)^T$. Индексные обозначения дают возможность кратко записывать формулы с помощью знака суммирования Σ, что особенно удобно, если используется неединичная метрика.

Скалярное поле в некоторой области пространства $\mathbb{R}^3$ представляет собой вещественнозначную функцию f:

$$f : \mathbb{R}^3 \to \mathbb{R}, \quad f(x, y, z) = f(\mathbf{r}) \in \mathbb{R}.$$

В свою очередь векторное поле в области пространства $\mathbb{R}^3$ — это векторозначная функция $\mathbf{V}$:

$$\mathbf{V} : \mathbb{R}^3 \to \mathbb{R}^3, \quad \mathbf{V}(x, y, z) = \mathbf{V}(\mathbf{r}) = V_x(\mathbf{r})\mathbf{e}_x + V_y(\mathbf{r})\mathbf{e}_y + V_z(\mathbf{r})\mathbf{e}_z \in \mathbb{R}^3.$$

Градиентом скалярного поля $f(\mathbf{r})$ называется вектор, вычисляемый в декартовых координатах следующим образом:

$$\nabla f(x, y, z) = \operatorname{grad} f(x, y, z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right), \quad \nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right).$$

С помощью знака набла ∇ обозначен векторный дифференциальный оператор Гамильтона. Для того, чтобы подчеркнуть его «векторность», символ $\boldsymbol{\nabla}$ записан полужирным шрифтом.

Для упрощения изложения нами допущены некоторые неточности в изложении, о которых следует сказать отдельно.

— Строго говоря, градиент является ковектором. В нашем определении это отражено тем, что компоненты вектора записаны в строку, а не в столбец.

— Определение градиента опирается на декартову систему координат. Более общее определение должно быть дано в безкомпонентном виде.

Скалярное поле $f(\mathbf{r})$ порождает векторное поле $\boldsymbol{\nabla} f$, которое характеризует направление наибольшего изменения скалярного поля $f(\mathbf{r})$.

Дивергенция векторного поля $\mathbf{V} = (V_x, V_y, V_z)^T$ есть скаляр, в декартовых координатах вычисляемый следующим образом:

$$\boldsymbol{\nabla} \cdot \mathbf{V} = \mathrm{div}\mathbf{V} = \frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z} = \sum_{i=1}^{3} \frac{\partial V^i}{\partial x^i}.$$

Здесь через «·» обозначена операция скалярного умножения $\boldsymbol{\nabla} \cdot \mathbf{V} = (\boldsymbol{\nabla}, \mathbf{V})$.

Ротором векторного поля $\mathbf{V}$ называют вектор, вычисляемый в декартовых координатах следующим образом:

$$\boldsymbol{\nabla} \times \mathbf{V} = \begin{vmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z \\ \partial/\partial x & \partial/\partial y & \partial/\partial z \\ V_x & V_y & V_z \end{vmatrix} = \left(\frac{\partial V_z}{\partial y} - \frac{\partial V_y}{\partial z}\right)\mathbf{e}_x + \left(\frac{\partial V_x}{\partial z} - \frac{\partial V_z}{\partial x}\right)\mathbf{e}_y + \left(\frac{\partial V_y}{\partial x} - \frac{\partial V_x}{\partial y}\right)\mathbf{e}_z.$$

Отметим также, ротор не является вектором в строгом смысле. В классическом векторном анализе его называют *псевдовектором*, но более глубокий геометрический смысл раскрывается только при привлечении тензорной алгебры, где ротор можно представить или в виде 2–формы или в виде бивектора.

Также, при записи волнового уравнения, будет использоваться оператор Лапласа, который записывается в следующем виде:

$$\nabla^2 = (\nabla, \nabla) = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}.$$

Также нам понадобятся следующие два соотношения [68]:

$$\begin{aligned} \nabla \times f\mathbf{V} &= f\nabla \times \mathbf{V} + \nabla f \times \mathbf{V}, \\ \nabla \cdot f\mathbf{V} &= f\nabla \cdot \mathbf{V} + (\nabla f, \mathbf{V}). \end{aligned} \tag{C.1}$$

$$\begin{aligned} \nabla \times f\mathbf{V} &= f\nabla \times \mathbf{V} + \nabla f \times \mathbf{V}, \\ \nabla \cdot f\mathbf{V} &= f\nabla \cdot \mathbf{V} + (\nabla f, \mathbf{V}). \end{aligned} \tag{C.2}$$

C.2. Векторно-дифференциальные выражения второго порядка

Векторное поле называется *потенциальным*, если существует скалярное поле $f(x, y, z)$ такое, что

$$\mathbf{V} = \nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right),$$
$$\mathrm{d}f = V_x\,\mathrm{d}x + V_y\,\mathrm{d}y + V_z\,\mathrm{d}z\,.$$

В свою очередь векторное поле называется *солиноидальным* (трубчатым), если существует векторное поле $\mathbf{U}$, такое что

$$\mathbf{V} = \nabla \times \mathbf{U}.$$

Список иллюстраций

1.1. Линза с точечным источником 18
1.2. Линза с плоским источником 18
1.3. Точечный источник относительно линзы 18
1.4. Точечный источник на поверхности линзы 18
1.5. Коэффициент преломления для линз Максвелла и Люнеберга . 20
1.6. Схема линзы Итона 20
1.7. Линза Люнеберга 21
1.8. Линза Максвелла 21
1.9. Линза Итона 21
1.10. Линза Люнеберга 21
1.11. Линза Максвелла 21

2.1. План вывода уравнения эйконала 44
2.2. Плоскость $(\mathbf{r}, \mathbf{s}) = \text{const}$. Новые оси координат выбираются так, чтобы вектор $\mathbf{s}$ был ортом оси $O\zeta$. Две другие оси $O\xi$ и $O\eta$ выбираются произвольно и образуют правую систему координат $O\xi\eta\zeta$. 50
2.3. Различные точки сетки разбиения области интегрирования . . 73
2.4. Шаблон численной схемы 73
2.5. Блок-схема алгоритма быстрого заметания FSM (Fast Sweeping Method) 77
2.6. Схема обучения PINN 82
2.7. График сравнения решений на интервале [0, 1] 84
2.8. График сравнения решений на интервале [0, 15] 84
2.9. Фазовый портрет модели Лотки-Вольтерры при решении численным методом 86
2.10. График численного решения модели Лотки-Вольтерры 86
2.11. Фазовый портрет модели Лотки-Вольтерры при решении с использованием PINN 86
2.12. График решения модели Лотки-Вольтерры с использованием PINN 86
2.13. Графики ошибок решений 87
2.14. График решения модели SIR при решении численным методом 90
2.15. График решения модели SIR с использованием PINN 90

3.1. Фронты для линзы Люнеберга 94
3.2. Фронты для линзы Маквелла 94
3.3. Корректное изображение лучей линзы Максвелла 95
3.4. Условно корректное изображение лучей линзы Максвелла . . . 95
3.5. Ошибочное изображение лучей линзы Максвелла, моделируемой методом FSM 96
3.6. Решение уравнения Пуассона с официального сайта документации NeuralPDE 98
3.7. Фронты для линзы Максвелла 100