МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра дифференциальных уравнений и системного анализа

СОРОКО

Анна Дмитриевна

ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ЭНЕРГЕТИЧЕСКИХ СОСТОЯНИЙ КВАНТОВОЙ ТОЧКИ

Дипломная работа

Научный руководитель:
кандидат физ.-мат. наук,
доцент О. А. Лаврова

Допущена к защите

«___»_____ 2018 г.

Зав. кафедрой дифференциальных уравнений и системного анализа

Доктор физ.-мат. наук, профессор В. И. Громак

Минск 2018

ОГЛАВЛЕНИЕ

РЕФЕРАТ

В дипломной работе 52 страницы, 20 рисунков, 10 источников, 3 приложения.

КВАНТОВАЯ ТОЧКА, УРАВНЕНИЕ ШРЕДИНГЕРА, ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ, МЕТОД КОНЕЧНЫХ ЭЛЕМЕНТОВ.

Целью дипломной работы является численное моделирование энергетических состояний цилиндрической квантовой точки с применением конечно-элементного подхода к решению задач на собственные значения. Задача на собственные значения на основе уравнение Шредингера формулируется внутри квантовой точки и в окружающей полупроводниковой среде. Нелинейность математической модели обусловлена зависимостью коэффициентов уравнения от энергии, которая представляет собой неизвестное собственное значение уравнения Шредингера.

В дипломной работе получены следующие результаты:

- Разработаны и протестированы конечно-элементные алгоритмы для моделирования энергетических состояний квантовой точки в осесимметричном и трехмерном случаях средствами Matlab и пакета FEniCS (Python).
- Сравнительный анализ полученных численных результатов показал, что осесимметричная модель эффективно реализуется средствами Matlab и Python. Показано, что точность трехмерных расчетов зависит от учета геометрии поверхности раздела двух сред при построении сетки.

Дипломная работа носит научно-практический характер. Разработанные алгоритмы расчета энергетических состояний цилиндрической квантовой точки в осесимметричном и трехмерном случаях можно использовать как основу для построения алгоритмов при решении более сложных задач, связанных с квантовыми точками любой формы либо взаимодействием квантовых точек.

Дипломная работа выполнена автором самостоятельно.

РЭФЕРАТ

У дыпломнай працы 52 стороны, 20 малюнкаў, 10 крыніц, 3 прыкладання.

КВАНТАВАЯ КРОПКА, РАЎНАННЕ ШРЕДІНГЕРА, КОЛЬКАСНАЕ МАДЭЛЯВАННЕ, МЕТАД КАНЧАТКОВЫХ ЭЛЕМЕНТАЎ.

Мэтай дыпломнай працы з’яўляецца колькаснае мадэляванне энергетычных станаў цыліндрычная квантавай кропкі з выкарыстаннем вядома-элементнага падыходу да вырашення задач на ўласныя значэння. Задача на ўласныя значэння на аснове раўнання Шрэдінгера фармулюецца ўнутры квантавай кропкі і ў навакольным паўправадняковым асяродзі. Нелінейнасць матэматычнай мадэлі абумоўлена залежнасцю каэфіцыентаў раўняння ад энергіі, якая ўяўляе сабой невядомае ўласнае значэнне раўнання Шрэдінгера.

У дыпломнай працы атрыманы наступныя вынікі:

- Распрацараваны і пратэставаны вядома-элементныя алгарытмы для мадэлявання энергетычных станаў квантавай кропкі ў восевасіметрычным і трохмерным выпадках сродкамі Matlab і пакета FEniCS (Python).
- Параўнальны аналіз атрыманых лікавых вынікаў паказаў, што восесіметрычная мадэль эфектыўна рэалізуецца сродкамі Matlab і Python. Паказана, што дакладнасць трохмерных разлікаў залежыць ад уліку геаметрыі паверхні падзелу двух асяроддзяў пры пабудове сеткі.

Дыпломная праца носіць навукова – практычны характар. Распрацаваныя алгарытмы разліку энергетычных станаў цыліндрычнай квантавай кропкі ў восесіметрычным і трохмерным выпадках можна выкарыстоўваць як аснову для пабудовы алгарытмаў пры рашэнні больш складаных задач, звязанных з квантавымі кропкамі альбо узаемадзеяннем квантавых кропак.

Дыпломная праца выканана аўтарам самастойна.

ABSTARCT

Graduation project consist of 52 pages, 20 illustrations, 10 sources, 3appendices.

QUANTUM DOT, SCHRÖDINGER EQUATION, NUMERICAL SIMULATION, FINITE ELEMENT METHOD.

The aim of graduation project is the numerical simulation of the energy states of a cylindrical quantum dot using a finite element approach for solving eigenvalues problems. The eigenvalues problem based on the Schrödinger equation is formulated inside the quantum dot and in the surrounding semiconductor medium. The nonlinearity of the mathematical model is due to the dependence of the coefficients of the equation on energy, which is an unknown eigenvalue of the Schrödinger equation.

In the graduation project the following results were obtained:

- Developed and tested finite element algorithms for modeling the energy states of a quantum dot in axisymmetric and three-dimensional cases were developed and testing using Matlab and FEniCS package (Python) software tools.
- Comparative analysis of the numerical results showed that the axisymmetric model is effectively implemented by Matlab and Python software tools. It is shown that the accuracy of three-dimensional calculations depends on the accounting of the interface between two media in domain triangulation.

The graduation project has a scientific and practical nature. The developed algorithms for calculating the energy states of a cylindrical quantum dot in axisymmetric and three-dimensional cases can be used as a basis for constructing algorithms to solve more complex problems associated with quantum dots of any geometry or the interaction of quantum dots.

The graduation project was made by the author himself.

ВВЕДЕНИЕ

Моделирование играет важную роль в современном мире, поскольку множество различных процессов и явлений изучаются на основе математических моделей. Моделирование представляет собой исследование какого-либо объекта или системы объектов путем построения и изучения их моделей, которые используются для анализа результатов, оптимизации процессов, поиска более качественного решения тех или иных проблем.

Любое научное исследования основывается на идее построения математической модели – мысленного аналога, описания, схемы, чертежа, процесса или явления, которая используется в качестве заменителя или представителя. Сам объект, процесс или явление называется оригиналом модели.

При изучении сложных систем, состоящих из взаимодействия разных объектов, реальные процессы заменяются некоторой упрощенной копией, довольно часто такие модели служат для изучения определенных качеств, которые позволяют разобраться в структуре системы, явления, в последовательности ее работы. Основная мысль моделирования – дать возможность предсказать изменение системы.

Иногда получается так, что исходный объект системы оказывается недоступным, в связи с чем эксперименты с ним стоят дорого или могут привести к серьезным последствиям, поэтому наиболее рациональным является использование модели, соответствующей исходному объекту системе.

Особую роль в науке играет математическое моделирование, на основе которого проектируются, конструируются многие системы. Модели применимы абсолютно во всех сферах жизни. С течением времени они развиваются и применяются для изучении все более сложных механизмов.

Математическая модель является базисом численного моделирования, которое применяется при больших объемах вычислений, необходимых для исследования. Данный метод имеет большое значение, поскольку модели для практически применимых задач являются нелинейными и получить аналитическое решение не представляется возможным. Поэтому важно уметь производить численный анализ моделируемых систем. Численное моделирование подразумевает создание математической модели изучаемой системы и дальнейшее ее исследование с использованием численных методов, в частности, с помощью методов конечных элементов, которые реализуются с использованием специальных программ и пакетов.

Методы конечных элементов на сегодняшний день являются одним из наиболее распространенных способов решения задач математической физики. Большую популярность метод приобрел за счет универсальности, сочетая в себе лучшие качества вариационных и разностных методов. Достоинство метода заключается в том, что можно использовать разные способы построения сетки, уточняя при этом параметры, которые влияют на точность решения. Востребованность метода подтверждается тем фактом, что в поисковой системе Google количество обращений по запросу «finite element method» составляет около 89.5 млн страниц.

Целью данной работы является численное моделировании энергетических состояний квантовой точки, представляющие собой полупроводниковую наноразмерную систему («искусственный атом»), электронными состояниями которой можно управлять с помощью внешних полей. В рамках данной работы будет рассмотрена квантовая точка цилиндрической формы, энергетическими состояниями которой являются решения задачи на собственные значения, сформулированной на основе уравнения Шредингера.

Численное моделирование энергетических состояний квантовой точки будет производиться с помощью конечно-элементного подхода к решению задач на

собственные значения. Также будет проведен сравнительный анализ для подтверждения правильности численного решения рассматриваемой модели на основе результатов статьи [8].

Для решения поставленной задачи будут использоваться программные системы Matlab и Python. Выбор указанных систем обусловлен возможностями построения с их помощью геометрических объектов со сложной внутренней и внешней структурой, использования триангуляции построенной геометрической модели с выбором различных параметров, применения методов конечных элементов с заданием различных пространств тестовых функций, маркирования областей геометрических фигур, определения граничных условий к поставленной задаче.

Актуальность данной работы для научной сферы заключается в развитии умения довести математическую модель до числового результата, который при этом можно интерпретировать как количественное описание реальной ситуации. Использование конечно – элементного подхода к решению задач на собственные значения открывает возможности для моделирования более сложных систем.

ГЛАВА 1
ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

1.1Теория квантовой точки

Квантовая точка представляет собой полупроводниковую наноразмерную систему («искусственный атом»), электронными состояниями которой можно управлять с помощью внешних полей.

Квантовые точки по своим размерам больше традиционных для химии молекулярных кластеров (~ 1 нм при содержании не больше 100 атомов). Квантовые точки объединяют физические и химические свойства молекул с оптоэлектронными свойствами полупроводников. [6]

Материал, который имеет электропроводность между проводником и диэлектриком называется полупроводником. Кремний, германий и графит некоторые примеры полупроводников. Полупроводниковые приборы являются основой современной электроники, включая транзисторы, светоизлучающие диоды, солнечные батареи и т. д.

В полупроводниках запрещенной зоной называют область энергии, отделяющую полностью заполненную электронами валентную зону от незаполненной зоны проводимости В этом случае шириной запрещенной зоны называется разность энергий между дном (нижним уровнем) зоны проводимости и потолком (верхним уровнем) валентной зоны.

Квантово-размерные эффекты играют важную роль в оптоэлектронных свойствах квантовых точек [3]. Энергетический спектр квантовой точки отличается от объемного полупроводника. Электрон в нанокристалле ведет себя как в трехмерной потенциальной «яме» (ограниченная область пространства с

пониженной потенциальной энергией частицы). Существует несколько стационарных уровней энергии для электрона с характерным расстоянием между ними $\frac{\hbar^2}{2md^2}$, где d – размер нано кристалла (квантовой точки). Таким образом, энергетический спектр квантовой точки зависит от ее размера. При переходе носителей заряда между энергетическими уровнями в квантовой точке может излучаться или поглощаться фотон. Частотами переходов, т.е. длиной волны поглощения или люминесценции, легко управлять, меняя размеры квантовой точки. Поэтому квантовые точки иногда называют «искусственными атомами». В терминах полупроводниковых материалов это можно назвать возможностью контроля эффективной ширины запрещенной зоны. Ширины запрещенной зоны в полупроводниках называют область энергии, отделяющую полностью заполненную электронами валентную зону от незаполненной зоны проводимости. Валентная зона – энергетическая область разрешенных электронных состояний в твердом теле, заполненная валентными электронами. Зона проводимости – первая из незаполненных электронами зон – диапазонов энергии, где могут находиться электроны, в полупроводниках и диэлектриках.[2]

Электроны из валентной зоны, преодолев запрещенную зону, при ненулевой температуре попадают в зону проводимости и начинают участвовать в проводимости, т.е. перемещаться под действием электрического поля. Ширина запрещенной зоны – разность энергий между дном зоны проводимости и потолком валентной зоны. Другими словами, ширина запрещенной зоны – это минимальная энергия, необходимая для перехода электрона из валентной зоны в зону проводимости (Рисунок 1.1).[2]

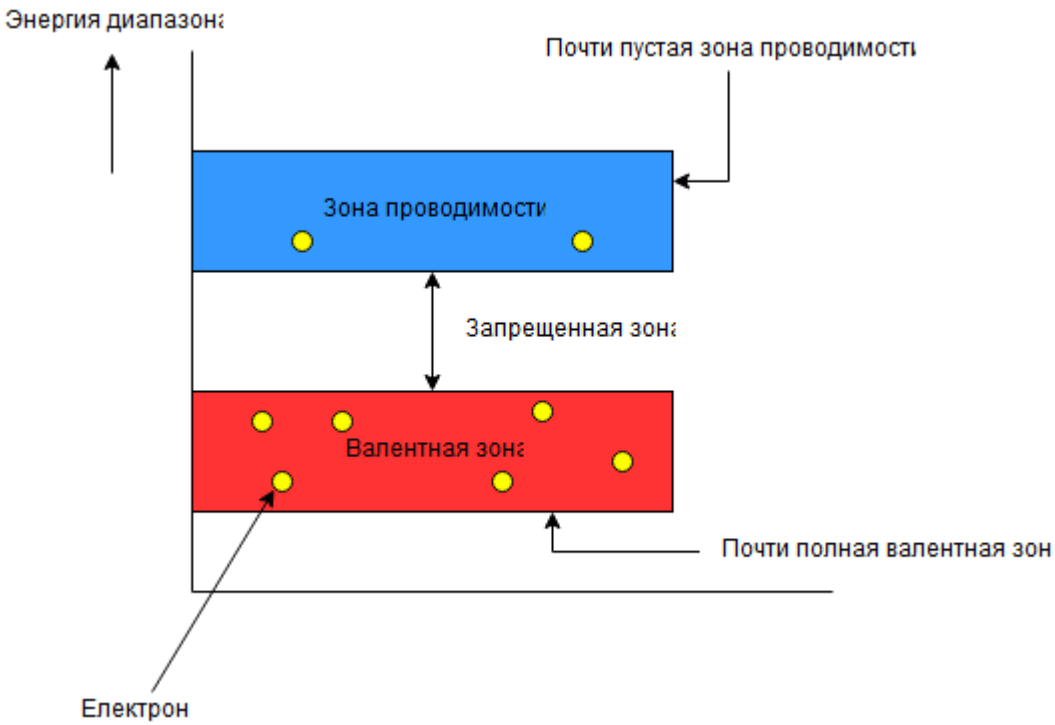

Рисунок 1.1 Полупроводник

Зависимость энергетического спектра от размера дает огромный потенциал для практического применения квантовых точек. Как уже отмечалось, квантовые точки могут найти применение в оптоэлектрических системах, таких как светоизлучающие диоды и плоские светоизлучающие панели, лазеры, ячейки солнечных батарей и фотоэлектрических преобразователей, как биологические маркеры, т.е. везде, где требуются варьируемые, перестраиваемые по длине волны оптические свойства. [6]

Существует определенная классификация квантовых точек:

1. Коллоидные квантовые точки.
2. Сферические квантовые точки (непосредственно quantum dots) – большая часть квантовых точек. В настоящий день имеют наибольшее практическое значение, поскольку наиболее просты в изготовлении.
3. Эллипсоидные квантовые точки (nanorods) – нанокристаллы, вытянутые вдоль одного направления. Указанные границы условны. С точки зрения

практического применения данный класс квантовых точек применяется как источник поляризованного излучения. [6]

4. Цилиндрические квантовые точки

В данной работе рассматриваются цилиндрические квантовые точки.

1.2Уравнение Шредингера в квантовой механике.

В общем, решения зависящего от времени уравнения Шредингера будут описывать динамическое поведение частицы, в некотором смысле аналогично тому, как уравнение Ньютона $F = ma$ описывает динамику частицы в классической физике. Однако есть важное отличие. Решая уравнение Ньютона мы можем определить положение частицы как функцию времени, то, решая уравнение Шредингера, мы получаем волновую функцию $\Psi(\mathrm{x}, T)$, которая говорит нам, как вероятность найти частицу в некоторой области пространства изменяется как функция времени.

Уравнение Шредингера имеет две "формы", одна из которых явно зависит от времени, и поэтому описывает, как волновая функция частицы будет развиваться во времени. В целом, волновая функция ведет себя как волна, и поэтому уравнение часто называют волновым уравнением Шредингера, зависящим от времени. Другая форма – это уравнение, в котором временная зависимость "удалена" и, следовательно, такое уравнение известно, как независимое от времени уравнение Шредингера и, как оказалось, описывает, среди прочего, допустимые энергии частицы. Это не два отдельных, независимых уравнения – независимое от времени уравнение может быть легко выведено из уравнения, зависящего от времени (за исключением того, если потенциал зависит от времени).

Рассмотрим уравнение Шредингера, зависящее от времени.

Вид волнового уравнения физической системы определяется ее гамильтонианом, приобретающим в силу этого фундаментальное значение во всем математическом аппарате квантовой механики.

Вид Гамильтониана свободной частицы устанавливается уже общими требованиями, связанными с однородностью и изотропией пространства и принципом относительности Галилея. В классической механике эти требования приводят к квадратичной зависимости энергии частицы от ее импульса: $E = \frac{p^2}{2m}$, где постоянная m называется массой частицы. В квантовой механике те же требования приводят к такому же соотношению для собственных значений энергии и импульса – одновременно измеримых сохраняющихся (для свободной частицы) величин.

Таким образом, гамильтониан свободно движущейся частицы выглядит следующим образом:

$$\widehat{H} = \frac{\hbar^2}{2m}\Delta ,$$

где $\Delta = \frac{\partial^2}{\partial^2 x} + \frac{\partial^2}{\partial^2 y} + \frac{\partial^2}{\partial^2 z}$ – оператор Лапласа.

При рассмотрении частицы в бесконечной потенциальной яме было отмечено, что волновая функция частицы с фиксированной энергией E вполне естественно может быть записана как линейная комбинация волновых функций вида

$$\Psi(x,t) = Ae^{i(kx-\omega t)}, \qquad (1.1)$$

которая представляет волну, движущуюся в положительном направлении x, и соответствующую волну, движущуюся в противоположном направлении, что приводит к возникновению стационарной волны, необходимую для удовлетворения граничных условий. Описанная формулировка волновой функции соответствует классическому понятию частицы, отскакивающей назад и вперед между стенками потенциальной ямы, поэтому мы принимаем волновую функцию, описанную выше,

как соответствующую волновую функцию свободной частицы, импульс которой равен $p = k^2$, а энергия $E = \hbar\omega$. Имея ввиду определенные величины, можно записать, что

$$\frac{\partial^2\Psi}{\partial x^2} = -k^2\Psi, \tag{1.2}$$

используя то, что $E = \frac{p^2}{2m} = \frac{\hbar^2 k^2}{2m}$, можно переписать следующим образом:

$$-\frac{\hbar^2}{2m}\frac{\partial^2\Psi}{\partial x^2} = \frac{p^2}{2m}\Psi, \tag{1.3}$$

что, то же самое, что и

$$\frac{\partial\Psi}{\partial t} = -i\omega\Psi, \tag{1.4}$$

используя $E = \hbar\omega$, получим

$$i\hbar\frac{\partial\Psi}{\partial t} = \hbar\omega\psi = E\Psi. \tag{1.5}$$

Перепишем уравнение с учетом кинетической и потенциальной энергий $E = \frac{p^2}{2m} + V(x)$ следующим образом

$$E = \frac{p^2}{2m}\Psi + V(x)\Psi, \tag{1.6}$$

где Ψ теперь – волновая функция частицы, движущейся под воздействием потенциала $V(x)$. Но, если предположить, что результаты (1.3) и (1.5) применимы для данного случая, тогда

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} + V(x) = i\hbar\frac{\partial\Psi}{\partial t}. \tag{1.7}$$

Выражение, представленное выше, является знаменитым волновым уравнением Шредингера, зависящим от времени. Создание и решение этого уравнения, анализ его решений составляют основу квантовой механики, известной как волновая механика.

Несмотря на то, что уравнение не похоже на стандартное волновое уравнение, которое описывает волны на растянутой струне, тем не менее, оно называется волновым уравнением, поскольку может иметь решения, которые представляют волны, распространяющиеся в пространстве. Рассмотрим пример: гармоническая волновая функция для свободной частицы с энергией E и моментом p

$$\Psi(x,t) = Ae^{i(px-Et)/\hbar}\,, \tag{1.8}$$

является решением уравнения, соответствующее свободной частице, $V(x) = 0$. Но это уравнение может иметь явно неволновые решения, форма которых зависит от природы потенциала $V(x)$.[10]

Решением уравнения Шредингера является нахождение энергетических состояний рассматриваемой частицы и волновой функции.

Волновая функция – это комплекснозначная функция f, определенная в R^1 (если электрон рассматривается на линии), в R^2 (если электрон рассматривается в 2D области) или в R^3 (если электрон рассматривается в пространстве) и удовлетворяет следующему условию:

$$\int |f|^2 = 1,$$

где интеграл определен в области, в которой рассматривается электрон (линия, плоскость, пространство)

Каждый электрон имеет связанную с ним волновую функцию.

Физический смысл волновой функции заключается в том, что если A – любая рассматриваемая область, и если решается задача, заключающаяся в том, что находится положение электрона в области A, тогда вероятность того, что электрон находится в рассматриваемой области[10]:

$$\int_A |f|^2.$$

Т.е. волновая функция описывает положение и состояние электрона, а ее квадрат дает «плотность вероятности» (плотность распределения) электронов в рассматриваемой области.

1.3Метод конечных элементов

Метод конечных элементов (МКЭ) является одним из самых эффективных и наиболее часто используемых численных методов решения научных и прикладных инженерных задач, математические модели которых описываются с помощью уравнений в частных производных. В частности, МКЭ применяется для решения задач математической физики (механика твердых тел, теплопроводность, электромагнетизм, аэро- и гидродинамика).

Основная идея МКЭ состоит в том, что:

1. Любую непрерывную величину (температуру, давление, перемещение) можно аппроксимировать дискретной моделью, которая строится на множестве кусочно-непрерывных функций, определенных на конечном числе подобластей (элементов);
2. Кусочно-непрерывные функции определяются с помощью значений непрерывной величины в конечном числе точек рассматриваемой области. [7]

Основная концепция МКЭ состоит в построении дискретной модели непрерывной величины:

1. В рассматриваемой области фиксируется конечное число точек. Эти точки называются узловыми (или просто узлами).
2. Значение непрерывной величины в каждой узловой точке считается переменной, которая должна быть определена. [7]

3. Область определения непрерывной величины разбивается на конечное число подобластей, называемых элементами (или конечными элементами). Эти элементы имеют общие узловые точки и в совокупности аппроксимируют форму области.
4. Непрерывная величина аппроксимируется на каждом элементе полиномом (или какой-либо другой функцией), который определяется с помощью узловых значений этой величины. [7]

Для каждой элемента определяется свой полином, но полиномы подбираются так, чтобы сохранилась непрерывность величины вдоль границ элемента.

Для того чтобы наглядно показать, как работает МКЭ рассмотрим построение дискретной модели на примере одномерной задачи о распределении температуры в стержне: необходимо выполнить разбиение области в виде отрезка фиксированной длины для получения распределения температуры в нем $T(x)$:

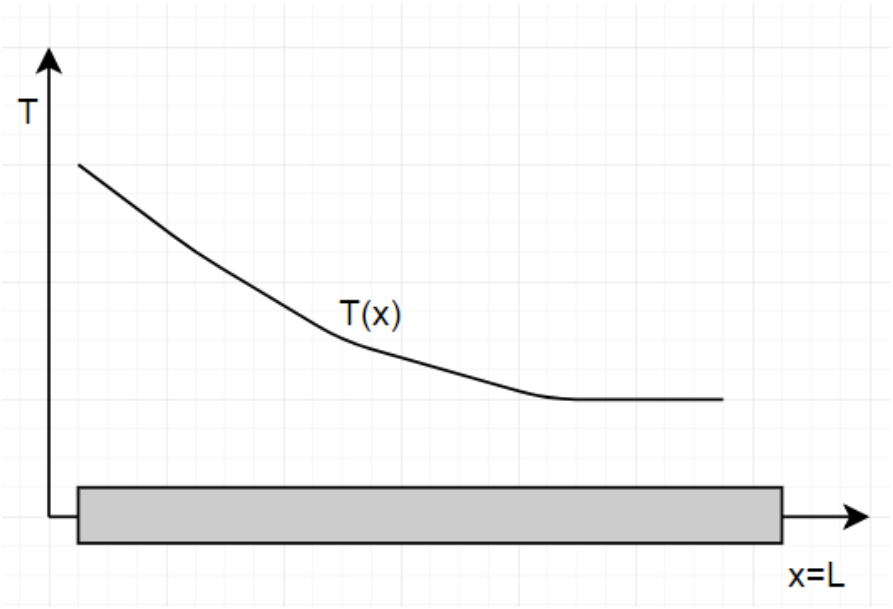

Рисунок 1.2 График исходной функции

На оси Ox фиксируются и нумеруются точки – узловые точки. Расстояние между точками не играет важной роли (Рисунок 1.3)

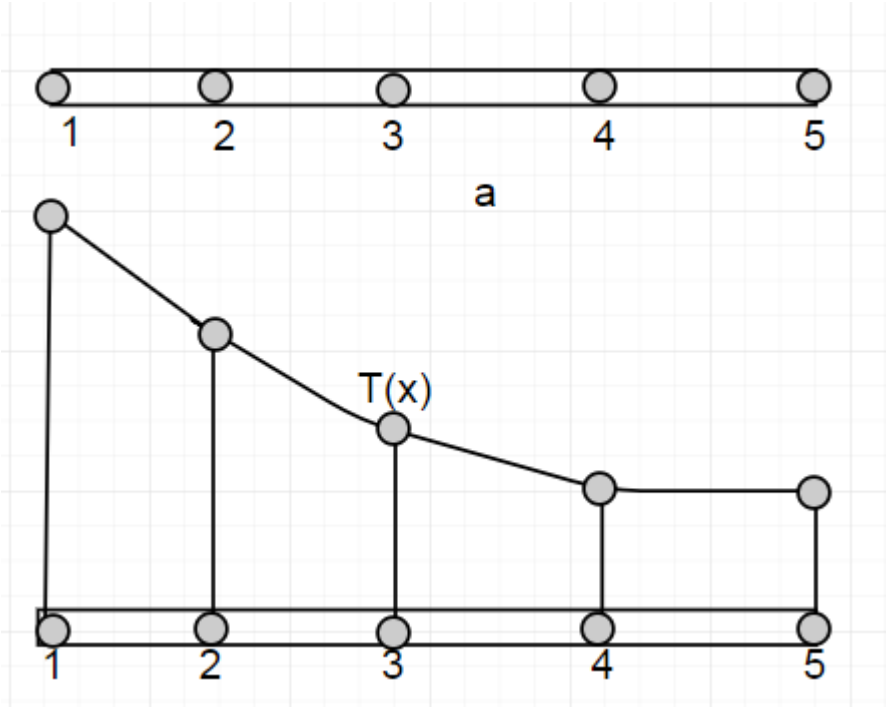

Рисунок 1.3 Выбор узловых точек

Разбиение области на элементы можно совершать двумя способами:

1. Каждый элемент ограничивать двумя соседними узлами (Рисунок 1.6).
2. В данном случае можно разбить область на два элементам, каждый из которых будет содержать по три узла (Рисунок 1.5).

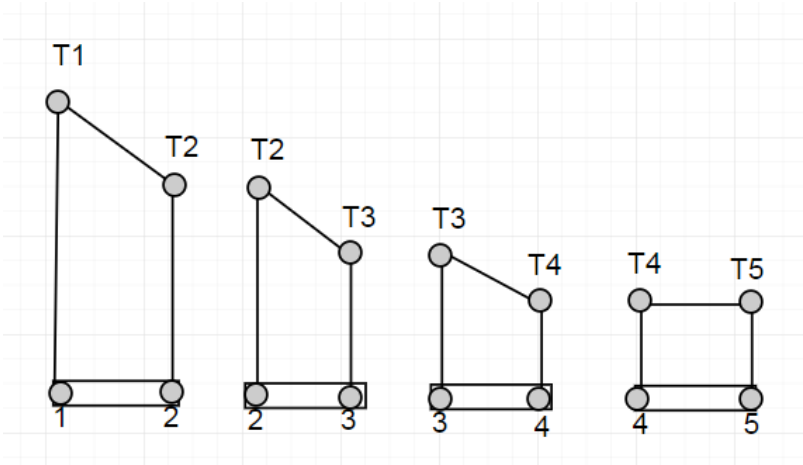

Рисунок 1.4 Разделение области на элементы а

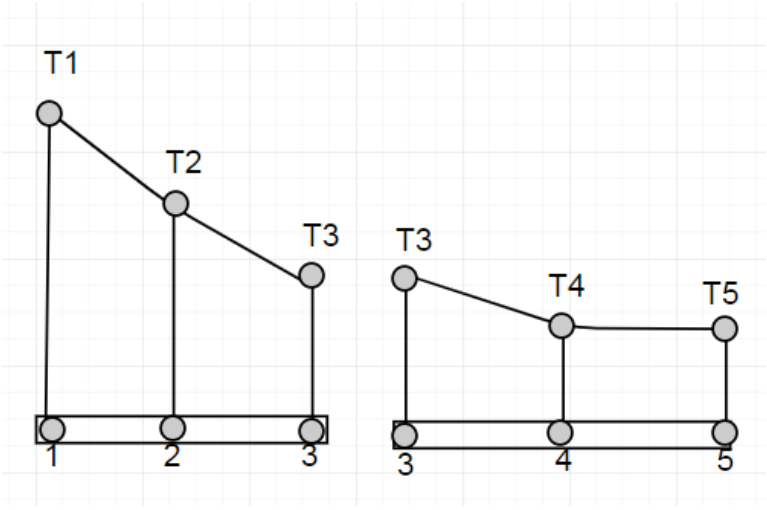

Рисунок 1.5 Разбиение области на элементы б

Соответствующая аппроксимирующая функция определяется значениями $T(x)$ в узловых точках.

Если рассматриваем разбиение области на четыре элемента, на каждый элемент приходится по два узла, т.е. функция будет линейна по x. В данном случает окончательная аппроксимация $T(x)$ будет состоять из четырех кусочно-линейных функций, которые определены каждая на своем элементе(Рисунок 1.6).

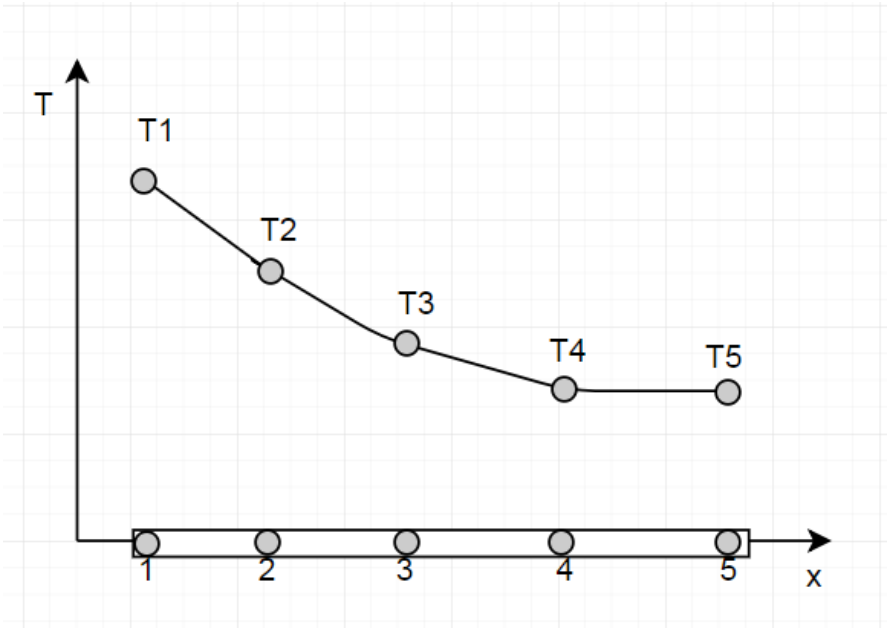

Рисунок 1.6 Аппроксимация исходной функции по разбиению а

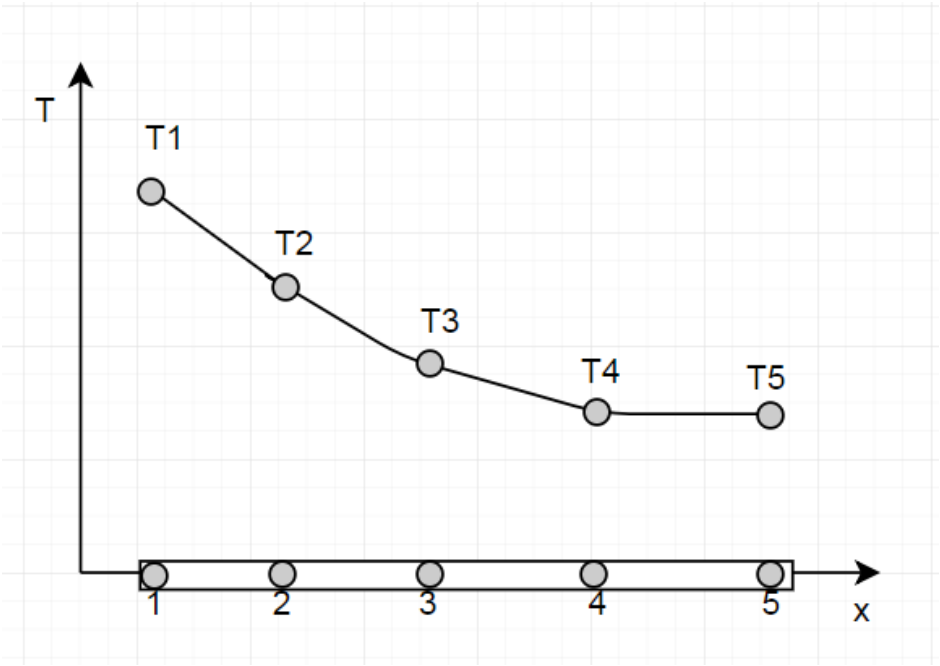

Рисунок 1.7 Аппроксимация исходной функции по разбиению б

В случае разбиения области на два элемента с тремя узловыми точками, аппроксимирующая функция будет представлять полином второй степени (Рисунок 1.7) и окончательной аппроксимацией $T(x)$ будет совокупность двух кусочно-непрерывных квадратичных функций.[7]

Если рассматриваем построение дискретной модели непрерывной величины, которая определена в двух- или трехмерной области, основная концепция МКЭ будет такой же. В двумерном случае элементы описываются функциями от x, y, а элементы рассматриваются чаще всего в виде треугольника или четырехугольника. Функции элементов будут представлять собой плоские (Рисунок 1.8) или криволинейные поверхности (Рисунок 1.9).

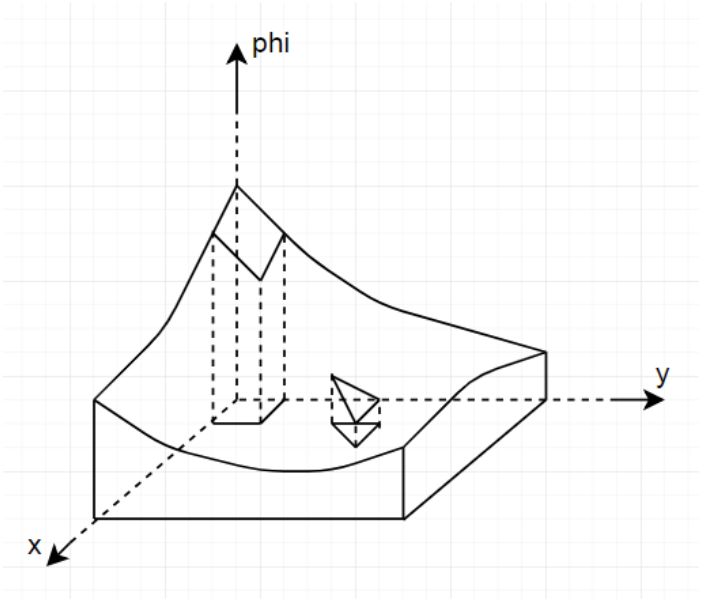

Рисунок 1.8 Плоская поверхность функции элементов

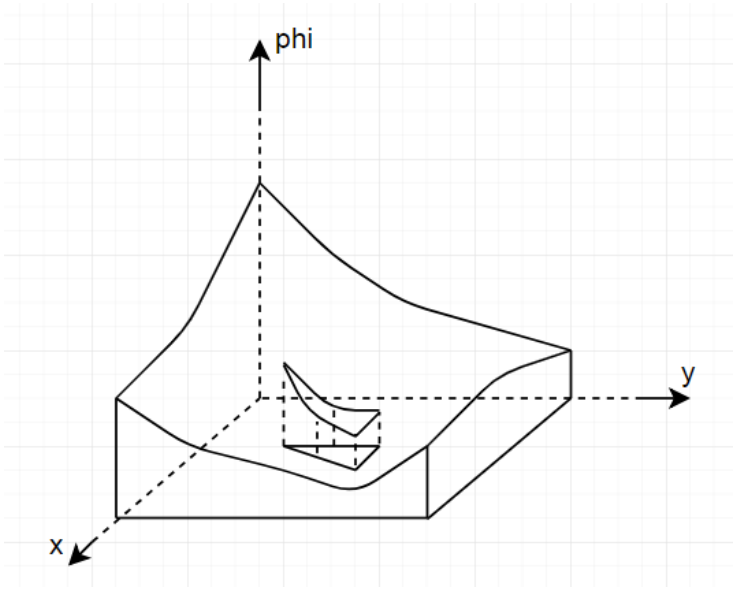

Рисунок 1.9 Криволинейная поверхность функции элементов

В случае плоской поверхности минимальное количество узловых точек, которое необходимо взять для функция элемента, равно трем, для треугольного элемента и четырем для четырехугольного.

Если число узлов, которое используется для построения функции элемента, больше минимального, тогда функции элемента соответствует криволинейная поверхность,

а если брать избыточное число узлов, того можно рассматривать элементы с криволинейными границами.

Таким образом, окончательная аппроксимация двумерной величины $T(x, y)$ будет служить совокупность кусочно-непрерывных поверхностей, каждая из которых определяется на отдельном элементе с помощью значений $T(x, y)$ в соответствующих узлах.[7]

Основная задача МКЭ – определить искомую функцию (величину) в узлах сетки. Для решения данной задачи используются принципы вариационного исчисления: минимизация специально построенного функционала, поэтому МКЭ относится к вариационным методам.

Для того чтобы искомые узловые значения имели «наилучшее» приближение к истинному значению температуры, необходимо отрегулировать их с помощью минимизации некоторой величины, которая связана с физической сущностью задачи.

В данном примере минимизирующий функционал связан с уравнением теплопроводности.

В конечном итоге процесс минимизации сводится к решению СЛАУ относительно узлов T(x).

Основные преимущества МКЭ:

1. Свойства материалов смежных элементов не должны быть обязательно одинаковыми, поэтому можно применять данный метод для тел, состоящих из нескольких материалов
2. Криволинейная область может быть аппроксимирована как с помощью прямолинейных, так и криволинейных элементов. Таким образом метод можно использовать для областей с любой формой.

3. Есть возможность укрупнять и умельчать элементы, т.е. делать сетку более крупную или более мелкую.
4. Данный метод позволяет рассматривать задачи с разрывными и смешанными граничными условиями.

Но также у метода есть явные недостатки:

1. Необходимость дискретизация всей расчетной области связано со значительными вычислительными усилиями при переходе от двумерных к трехмерным задачам;
2. Задачи, сформулированные в неограниченной области, сводятся к ограниченной области, что является дополнительным источником ошибки решения.

При использовании метода конечных элементов необходимо формулировать исходную задачу в вариационной формулировке, для этого используется формула интегрирования по частям.

Теорема 1 (интегрирование по частям). Пусть область $\Omega \in \mathbb{R}^n$ имеет C^1 - границу $\partial\Omega$ и внешнюю нормаль n: $\partial\Omega \to \mathbb{R}^n$. Тогда для функции $u, v \in C^1(\overline{\Omega}, \mathbb{R})$ выполняется

$$\int_\Omega \nabla uv = -\int_\Omega u\nabla v + -\int_{\partial\Omega} uvn. \tag{1.9}$$

Применим формулу интегрирования по частям для -ой компоненты векторного поля w, заданного на области Ω, в i -ом направлении и просуммируем по $i = 1, \dots, n$

$$\int_\Omega \nabla wv = -\int_\Omega w\nabla v \pm \int_{\partial\Omega} wnv. \tag{1.10}$$

Частный случай соотношения (1.9) для тестовой функции $v = 1$

$$\int_\Omega \nabla w = -\int_{\partial\Omega} wn. \tag{1.11}$$

Называется теоремой о дивергенции или формулой Остроградского(Гаусса-Остроградского).

Полагая $w = \nabla u$ в (1.9), получаем формулу Грина

$$\int_{\Omega} \nabla uv = -\int_{\Omega} \nabla u \nabla v + -\int_{\partial\Omega} \nabla unv. \tag{1.12}$$

ГЛАВА 2

ФОРМУЛИРОВКА МАТЕМАТИЧЕСКОЙ МОДЕЛИ

2.1 Основные понятия и определения

Модель – это система (объект), исследование которой служит средством для получения информации о другой системе. [4]

Моделирование – процесс отражения свойств одного объекта (оригинала) в другом объекте (модели). Это могут быть объекты «как есть» в целом и (или) их отдельные сущности – процессы и явления. Явления – например, поведение животного, состояния погоды – рассматриваются как сложные процессы. [1]

Гамильтониан – в квантовой теории это оператор полной энергии системы (функция Гамильтона).

Гетероструктура – термин в физике полупроводников, обозначающий выращенную на подложке слоистую структуру из различных полупроводников, в общем случае отличающихся шириной запрещённой зоны.

Запрещенная зона – термин из физики твердого тела — зона — область значений энергии, которыми не может обладать электрон в идеальном (бездефектном) кристалле. Этот диапазон называют шириной запрещённой зоны и обычно численно выражают в электрон-вольтах. [4]

2.2Математическая модель

Рассматривается трехмерная структура квантовой точки, где эффективный Гамильтониан представлен следующим образом:

$$\widehat{H} = -\frac{\hbar^2}{2}\nabla_r\left(\frac{1}{m(E,r)}\right)\nabla_r + V(r), \tag{2.1}$$

где ∇_r обозначает пространственный градиент, $m(E,r)$ – это эффективная масса электрона, которая зависит не только от энергии но и от позиции электрона. Выражение для эффективной массы электрона рассматривается как:

$$\frac{1}{m(E,r)} = -\frac{P^2}{\hbar^2}\left[\frac{2}{E+E_g(r)-V(r)} + \frac{1}{E+E_g(r)-V(r)+\Delta(r)}\right], \tag{2.2}$$

где $V(r)$ – это задерживающий потенциал, $E_g(r)$ – позиция, зависящая от зазора и $\Delta(r)$ – это спин-орбитальное взаимодействие в валентной зоне, P – это элемент матрицы моментов. Такая аппроксимация обычно используется для расчета энергий квантовой точки и хорошо описывает электрические свойства в 2D гетероструктруах. Это тот случай, когда должен учитываться непараболический эффект. В этом случае применяется такая аппроксимация для вычисления энергетических состояний трехмерной квантовой точки в узкой щели полупроводника.

Для систем с резким разрывом зоны проводимости на границе раздела квантовой точки (Ω_1) и кристаллической матрицы (Ω_2), задерживающий потенциал границы можно представить, как:

$$V(r) = \begin{cases} 0, & r \in \Omega_1 = \{(x,y,z), |x| < R_0, |y| < R_0, |z| < \frac{z_0}{2}\} \\ V_0 & r \in \Omega_2 = \{(x,y,z), |x| \geq R_0, |y| \geq R_0, |z| \geq \frac{z_0}{2}\} \end{cases}, \tag{2.3}$$

Интегрируя уравнение Шредингера с гамильтонианом (1) вдоль направления, перпендикулярного интерфейсу, получаем граничное условие Ben-Daniel Duke для волновой функции $\psi(r)$

$$\psi_{\Omega_1}(r_s) = \psi_{\Omega_2}(r_s), \tag{2.4}$$

$$\frac{\hbar^2}{2m(E,r_s)}\nabla_n\psi(r_s) = const, \tag{2.5}$$

где r_s представляет позицию в области. Такое граничное условие зависит от энергии электрона и возникает из-за разницы параметров в материалах(областях). Рассматривается квантовая точка, имеющая форму диска с радиусом R_0 и толщиной z_0 в цилиндрических координатах (R, ϕ, z). Начало координат системы находится в центре диска и ось z выбирается вдоль оси вращения. Пока система цилиндрически симметрична, волновая функция может быть представлена в следующем виде:

$$\psi(r) = \Phi(R,z)exp(il\phi), \tag{2.6}$$

где $l = 0, \pm1, \pm2, \ldots$ электронное орбитальное квантовое число и в (R, z) координатах задача остается двумерной:

$$\begin{gathered} -\frac{\hbar^2}{2m_1(E)}\left(\frac{\partial}{\partial R^2}+\frac{\partial}{R\partial R}+\frac{\partial^2}{\partial z^2}-\frac{l^2}{R^2}\right)\Phi_1(R,z) = E\Phi_1(R,z),\ R \le R_0, |z| \le \frac{z_0}{2}, \\ -\frac{\hbar^2}{2m_2(E)}\left(\frac{\partial}{\partial R^2}+\frac{\partial}{R\partial R}+\frac{\partial^2}{\partial z^2}-\frac{l^2}{R^2}\right)\Phi_2(R,z) + V_0\Phi_2(R,z) = E\Phi_2(R,z), \\ R > R_0, |z| > \frac{z_0}{2}, \end{gathered} \tag{2.7}$$

граничные условия (2.4), будут выглядеть следующим образом:

$$\begin{gathered} \Phi_1(R_0,z) = \Phi_2(R_0,z),\ |z| \le \frac{z_0}{2}, \\ \Phi_1\left(R, \pm\frac{z_0}{2}\right) = \Phi_2\left(R, \pm\frac{z_0}{2}\right),\ R \le R_0, \\ \frac{1}{m_1(E)}\left(\frac{\partial\Phi_1(R,z)}{\partial R}\right)_{R_0} = \frac{1}{m_2(E)}\left(\frac{\partial\Phi_2(R,z)}{\partial R}\right)_{R_0}, \quad |z| \le \frac{z_0}{2}, \\ \frac{1}{m_1(E)}\left(\frac{\partial\Phi_1(R,z)}{\partial z}\right)_{\pm\frac{z_0}{2}} = \frac{1}{m_2(E)}\left(\frac{\partial\Phi_2(R,z)}{\partial z}\right)_{\pm\frac{z_0}{2}}, \quad R \le R_0. \end{gathered} \tag{2.8}$$

Энергетические состояния представляются как сложная функция от параметров квантовой точки и момент импульса электрона. Решение получается путем

численного решения уравнения Шредингера (2.7), совместно с граничными условиями (2.8).

Из-за зависимости энергетических состояний от эффективной массы электрона расчет должен состоять из нескольких итераций для достижения самосогласованного решения. Для достижения самосогласованного решения предлагается нелинейная итерационная схема с обратной связью: первым шагом определяем начальную энергию как нулевую, второй шаг – вычисляем эффективную массу m, третий шаг – решаем уравнение Шредингера для определенной энергии, четвертый шаг – обновляем значение энергии как значение решения, возвращаемся ко второму шагу. Итерационный процесс прекращается, кода выполняется специальный критерий для значения энергии. Для получения полного численного результата, на третьем шаге дискретизируется уравнение Шредингера (2.7) на определенной сетке. Такая дискретизация совместно с граничными условиями (2.8) приводит к проблеме собственных значений.

$$AX = \lambda X, \tag{2.9}$$

где A – это матрица, полученная после дискретизации уравнения Шредингера совместно с граничными условиями, λ и X – это собственные значения и собственные вектора (волновые функции) матрицы A, соответственно. В общем случае, матрица A несимметрическая и разреженная, соответственно собственные значения такой матрицы достаточно чувствительны при малых изменениях элементов матрицы. В данной работе для решения проблемы собственных значений, используется функциональность пакета FEniCS, которая состоит в использования специальной функции SLEPcEigenSolver, более подробно рассмотренной в пункте 3.3 настоящей работы.[8]

Задача, описанная выше, может быть представлена как математическая модель в следующем виде:

$$\begin{cases} H_1\Phi_1(R,z) = E\Phi_1(R,z), \ \ (R,z) \in \Omega_1 \\ H_2\Phi_2(R,z) = E\Phi_2(R,z), \ \ (R,z) \in \Omega_2 \\ \Phi_1(R_0,z) = \Phi_2(R_0,z), \ R \le R_0 \\ \Phi_1\left(R, \pm\frac{z_0}{2}\right) = \Phi_2\left(R, \pm\frac{z_0}{2}\right), \ |z| \le \frac{z_0}{2}, \\ \frac{1}{m_1(E)}\left(\frac{\partial\Phi_1(R,z)}{\partial R}\right)_{R_0} = \frac{1}{m_2(E)}\left(\frac{\partial\Phi_2(R,z)}{\partial R}\right)_{R_0}, \quad |z| \le \frac{z_0}{2}, \\ \frac{1}{m_1(E)}\left(\frac{\partial\Phi_1(R,z)}{\partial z}\right)_{\pm\frac{z_0}{2}} = \frac{1}{m_2(E)}\left(\frac{\partial\Phi_2(R,z)}{\partial z}\right)_{\pm\frac{z_0}{2}}, \quad R \le R_0 \end{cases}, \tag{2.10}$$

где

$$H_1 = -\frac{\hbar^2}{2m_1(E)}\left(\frac{\partial}{\partial R^2} + \frac{\partial}{R\partial R} + \frac{\partial^2}{\partial z^2} - \frac{l^2}{R^2}\right),$$
$$H_2 = -\frac{\hbar^2}{2m_2(E)}\left(\frac{\partial}{\partial R^2} + \frac{\partial}{R\partial R} + \frac{\partial^2}{\partial z^2} - \frac{l^2}{R^2}\right) + V_0 \ . \tag{2.11}$$

2.3 Вариационная формулировка

Для решения уравнения и представления в вариационной формулировке мы рассматриваем исходной уравнение в цилиндрической области как двумерное уравнение в прямоугольной области осесимметричного сечения в первой четверти с соответствующими граничными условиями для сохранения симметричности уравнения относительно координат. В данном случае происходит обезразмеривание переменных по длине(масштаб длины 1нм) и энергии(масштаб энергии 1eV).Уравнение в прямоугольной области осесимметричного сечения выглядит:

$$\begin{cases} \varepsilon_1(E)\nabla^2\Phi = E\Phi, \ \ (x,y) \in \Omega_1 = \{(x,y), |x| < R_0, |y| < \frac{z_0}{2}\} \\ \varepsilon_2(E)\nabla^2\Phi + V_0\Phi = E\Phi, \ \ (x,y) \in \Omega_2 = \{(x,y), |x| \geq R_0, |y| \geq \frac{z_0}{2}\} \\ [\Phi] = 0, \ \ (x,y) \in \partial\Omega = \{(x,y), x = z, y = R\} \\ \left[\varepsilon \frac{\partial\Phi}{\partial n}\right] = 0, \ \ (x,y) \in \Gamma^* = \{(x,y), x = [0,z], y = [0,R]\} \end{cases}, \quad (2.12)$$

где

$$\varepsilon_1(E) = \mathrm{c}\ \mathrm{c}_1\left(\frac{2}{E+E_{1g}} + \frac{1}{E+E_{1g}+\Delta_1}\right),$$

$$\varepsilon_2(E) = \mathrm{c}\ \mathrm{c}_2\left(\frac{2}{E+E_{2g}} + \frac{1}{E+E_{2g}+\Delta_2-V_0}\right),$$

$$\mathrm{c} = \frac{\hbar^2}{2m_0d^2} = 0.381,\ c_1 = \frac{22.2}{3},\ c_2 = \frac{24.2}{3},$$

$$E_{1g} = 0.42, E_{2g} = 1.52,$$

$\Delta_1 = 0.48,\ \Delta_2 = 0.34$ (заданные параметры).

Для решения уравнения (2.11) используется специальная утилита FEniCS, в которой реализован метод конечных элементов, позволяющий решать уравнения такого типа.

Утилита FeniCS требует специального синтаксиса для решения уравнения (2.11). Синтаксис предполагает представления уравнения (2.11) в вариационной (интегральной, слабой) формулировке исходной задачи. Переход от исходной задачи к вариационной формулировке осуществляется с применением формулы интегрирования по частям (пункт 1.3).

Для того чтобы построить вариационную формулировку исходной задачи необходимо совершить следующие действия:

1. Определить пространство тестовых функций V и умножить исходное дифференциальное уравнение на тестовую функцию $v \in V$.

2. Проинтегрировать полученное уравнение по области Ω и применить формулу интегрирования по частям (1.9).
3. Учитываем граничные условия

Для уравнения (2.12) с граничными условиями (2.8) или граничной задачи (2.9), вариационная формулировка, следующая:

$$\text{найти } u \in V,$$

$$V = \{u: u \in H^1(\Omega), u = 0 \text{ на } \partial\Omega_2\}$$

$$\varepsilon_1(E)\int_{\Omega_1} \nabla v \nabla u \, dxdy + \varepsilon_2(E)\int_{\Omega_2} (\nabla v \nabla u + + V_0 u \, v) dxdy = , \tag{2.13}$$

$$= \int_{\Omega} E \, v \, u \, dxdy,$$

где $\Omega = \Omega_1 \cup \Omega_2$. Условия Дирихле выполняются в определении тестовых функции ($u = 0$ на $\partial\Omega_2$), условие Неймана выполняется автоматически. При интегрировании по частям возникают интегралы по поверхности, так как направление нормали разные в смежных областях, то явно возникают в уравнениях условие скачка.

ГЛАВА 3
ИЗУЧЕНИЕ ВОЗМОЖНОСТЕЙ ПАКЕТА FENICS В PYTHON

3.1Решение задач с разрывными коэффициентами

Для решения задач с разрывными коэффициентами пакет FEniCS программы python располагает большим количеством возможностей. Для того, чтобы продемонстрировать возможности решения задач с разрывными коэффициентам, рассмотрим задачу распространения тепла в пластине, которая состоит из двух слоев равной толщины с коэффициентами теплопроводности k_1, k_2 соответственно. Процесс распространения тепла описывается следующим уравнением:

$$-\nabla(k\nabla T) = 0 \ \ (x, y) \in \Omega = (0,1) \times (-\delta, \delta), \tag{3.1}$$

$$k = \begin{cases} k_1, & x \in (0,0.5) \\ k_2, & x \in (0.5,1) \end{cases}. \tag{3.2}$$

На границе двух сред выполняется условие переноса

$$[T] = 0, \ \left[k\frac{\partial T}{\partial n}\right] = 0 \ \text{ при } x = 0.5, \tag{3.3}$$

где [.] обозначает скачок функции при переходе через линию раздела. Необходимо найти температуру раздела при заданной температуре на боковых стенках

$$T = 0 \ \text{ при } x = 0,$$

$$T = T_0 \ \text{ при } x = 1,$$

при отсутствии теплового потока на верхней и нижней стенках

$$\frac{\partial T}{\partial n} = 0 \text{ при } y = \pm\delta, \tag{3.4}$$

где $\delta = 1, k_1 = 1, k_2 = 10, T_0 = 10$.

Задачу решаем с помощью МКЭ. Первоначально нужно определить область решения задачи (Рисунок 3.1). С помощью специальных пакетов среды Python определяем прямоугольную область, а также подобласти с разными слоями.

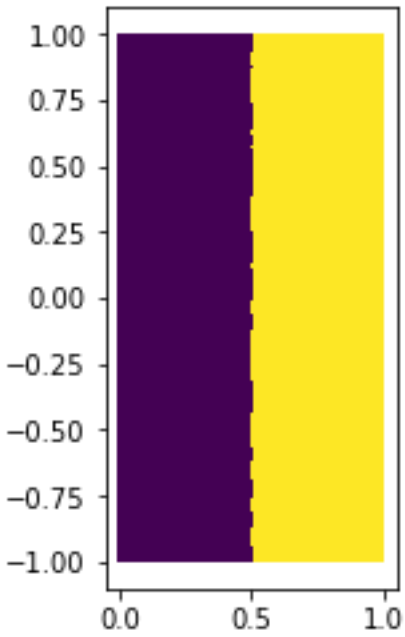

Рисунок 3.1 Моделируемая область

Далее строим сетку на этой области, или, другими словами, триангулируем область. Как можно заметить по рисунку (Рисунок 3.2) FEniCS строит сетку на подобластях, включенных в рассматриваемую область.[9]

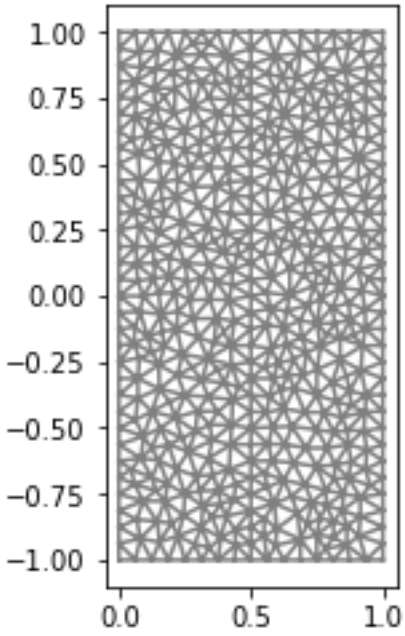

Рисунок 3.2 Триангуляция моделируемой области

Следующий шаг – это выбор пространства тестовых функций, которые будут использоваться для численного решения исходного уравнения в узлах сетки.

Для того, чтобы решать задачу с помощью МКЭ в FEniCS, необходимо придерживаться определенного синтаксиса при постановке задачи, а именно подавать задачу в вариационной формулировке.

Также, поскольку задача с разрывным коэффициентом k, для его задания в FEniCS используется следующий синтаксис:

$$k = Expression(\text{x[0]>0 \&\& x[0]<0.5+1E-14 ? k\_1 : k\_2}, degree = 0, \text{k\_1} = k1, \text{k\_2} = k2),$$

где $k1 = k_1, k2 = k_2$. Т.е. для задания разрывных коэффициентов мы используем строковое выражение с условием. Таким образом задаются любые уравнения и переменные, которые включают в себя разрывные коэффициенты.

$$a = k * dot\big(grad(u), grad(v)\big) * dx;$$

$$f = Constant(0);$$

$$L = f * v * dx.$$

Способом, описанным выше мы определили исходную задачу с разрывными коэффициентами в вариационной форме и следующим шагом является непосредственно решение поставленной задачи.

Получаем следующий график распространения тепла (Рисунок 3.3).

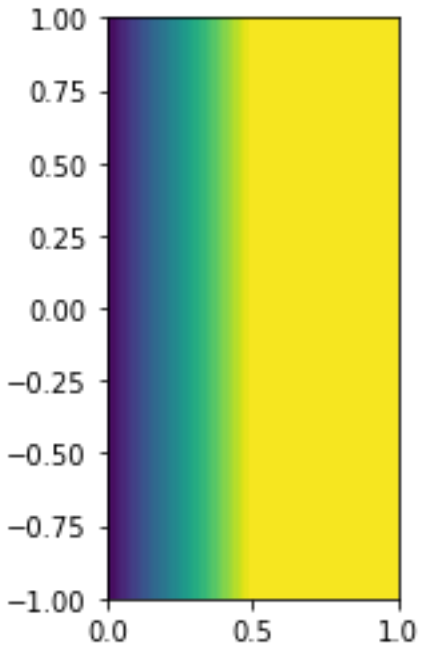

Рисунок 3.3 График распространения тепла

Также представим изолинии функции распространения тепла (Рисунок 3.4.).

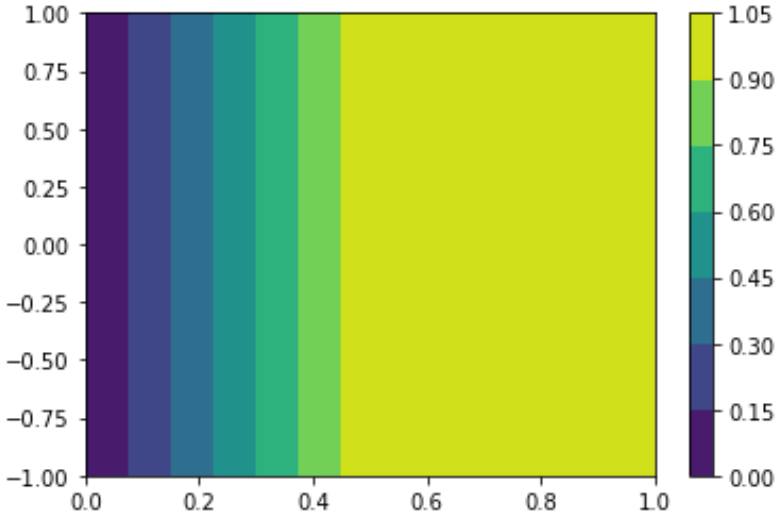

Рисунок 3.4 Изолинии функции распространения тепла

Для того чтобы понять правильность решения исходной задачи, необходимо вычислить ошибку, для этого вычислим решение исходной задачи путем решения системы уравнений и вычислим разницу решений. Для того, чтобы задать функцию с разрывными коэффициентами, пользуемся возможностями FEniCS и задаем функцию в следующем виде:

$$T(x) = \begin{cases} \frac{200000}{100001}x, \ 0 < x < 0.5, \\ \frac{2}{100001}x + \frac{99999}{100001}, \text{иначе} \end{cases},$$

где $a = \frac{200000}{100001}, b = 0, c = \frac{2}{100001}, d = \frac{99999}{100001}$.

Таким образом, ошибка вычислений составляет $0.032 * 10^{-12}$, что является хорошим результатом, из чего делаем вывод, что задача решена достаточно верно.

Также представим график разности функций: найденной и исходной.

3.2 Решение задач с комплекснозначными функциями

Пакет FEniCS программы Python предоставляет возможность решать задачи с комплекснозначными функциями. Для этого используется специальный синтаксис при постановке задачи непосредственно в пространстве Python.

Для того, чтобы показать возможности работы с комплекснозначными функциями в FEniCS, рассмотрим соответствующее уравнение:

$$\begin{cases} -\Delta u - k^2 u = 0, \ \ (x,y) \in \Omega \\ ki(2+u) + \frac{\partial u}{\partial n} = 0, \ \ \text{на входе канала: } (x,y) \in \Gamma_{in} = \{(x,y) \in \Omega \mid x = 0\} \\ kiu + \frac{\partial u}{\partial n} = 0, \text{на выходе канала: } (x,y) \in \Gamma_{out} = \{(x,y) \in \Omega \mid y = 0.8\} \\ u = 0, \text{на стенках канала: } (x,y) \in \partial\Omega(\Gamma_{in} \cup \Gamma_{out}) \end{cases}, \tag{3.5}$$

где $k = \frac{2\pi f}{c} = \frac{2\pi}{\lambda}$ – волновое число, λ – длина волны.

Уравнение рассматриваем в следующей области (Рисунок 3.6).

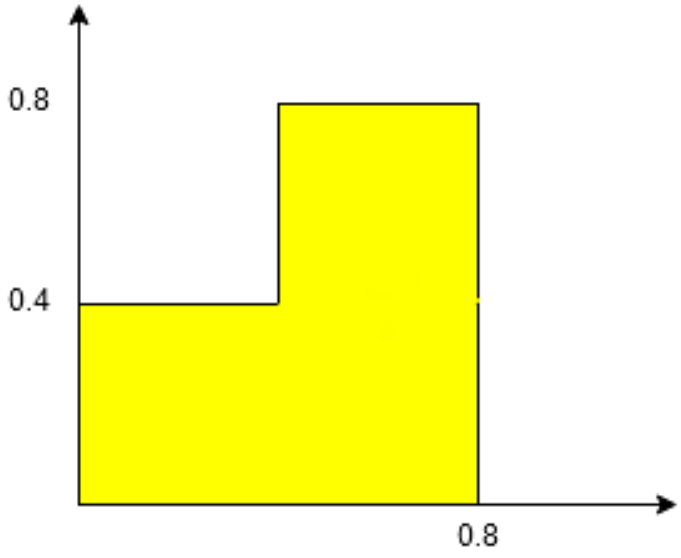

Рисунок 3.5 Исходная область

Граничное условие Неймана рассматриваются неявно. Поскольку FEniCS не поддерживает комплексные числа, то вышеописанная проблема может быть

разделена на мнимую и реальную части, которые в последствии можно соединить и решать уравнение с частными производными с помощью стандартных методов в среде python. Для того, чтобы решать такого рода уравнения, искомую функцию представляют в виде $u = u^R + iu^I$, где u^R, u^I это реальная и мнимая части функции, соответственно. Для решения вышеописанного уравнения мы воспользуемся МКЭ. МКЭ имеет свой алгоритм решения, изложенный в пункте 1.3.

Шаг первый. Строим прямоугольную область и определяем соответствующую сетку (триангуляция исходной области (Рисунок 3.7)).

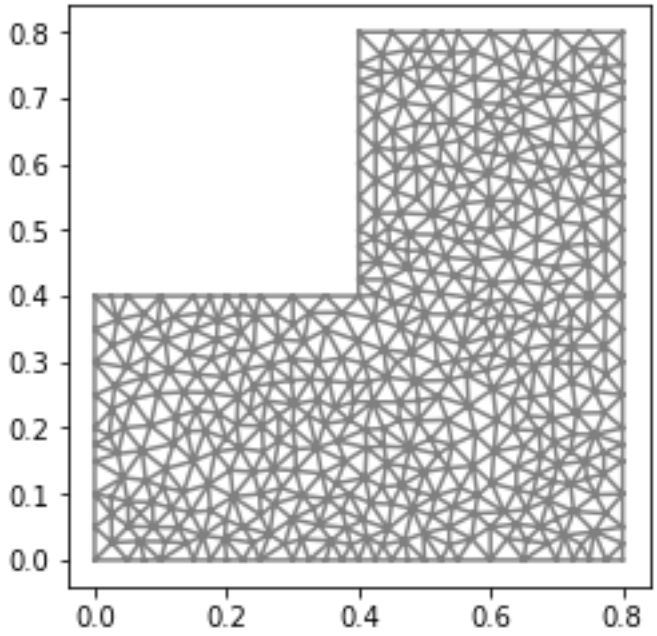

Рисунок 3.6 Триангуляция рассматриваемой области для комплекснозначной функции

Шаг второй. Определяем пространство тестовых функций. Определяем пространство коплекснозначных функций путем декартевого произведения заданного векторного пространства[9]:

$$V = VectorFunctionSpace(mesh,\text{"CG"},1);$$

$$Vcomplex = V * V.$$

Стандартным способом задаем граничные условия к уравнению.

Шаг третий. Определяем мнимую и реальную части искомой функции:

$$uR, uI = TrialFunctions(Vcomplex);$$

$$wR, wI = TestFunctions(Vcomplex).$$

Используя синтаксис FEniCS, определяем исходную задачу, используя вариационную формулировку, находим решение поставленной задачи

$$f\_r = Constant(0);$$

$$f\_i = Constant(0);$$

$$a_r = inner\big(grad(ur), grad(vr)\big) * dx - inner\big(grad(ui), grad(vi)\big) * dx -$$

$$-pow(k, 2) * ur * vr * dx + pow(k, 2) * ui * vi * dx + k * ur * vr * ds -$$

$$-k * ui * vi * ds;$$

$$a_i = inner\big(grad(ur), grad(vi)\big) * dx + inner\big(grad(ui), grad(vr)\big) * dx -$$

$$-pow(k, 2) * ur * vi * dx - pow(k, 2) * ui * vr * dx + k * ur * vi * ds +$$

$$+k * ui * vr * ds;$$

$$L_r = inner(f_r, vr) * dx - inner(f_i, vi) * dx - (2 * k * vr + 2 * k * vi) * ds(1);$$

$$L\_i = inner(f\_r, vi) * dx + inner(f\_i, vr) * dx - (2 * k * vi + 2 * k * vr) * ds(1);$$

$$a = a\_r + a\_i;$$

$$L = L\_r + L\_i;$$

$$A = assemble(a);$$

$$b = assemble(L).$$

Шаг четвертый. Искомую функцию определяем как функцию пространства комплекснозначных функций $Vcomplex$:

$$solvF = Function(Vcomplex).$$

Шаг пятый. Пусть решения реальной и мнимой частей искомой задачи – это $solvR, solvI$, тогда реальные и мнимые части функции будут выглядеть следующим образом:

$$solvR, solvI = solv.split().$$

После этого мы имеем решений исходной задачи в виде комплекснозначной функции (Рисунок 3.8).

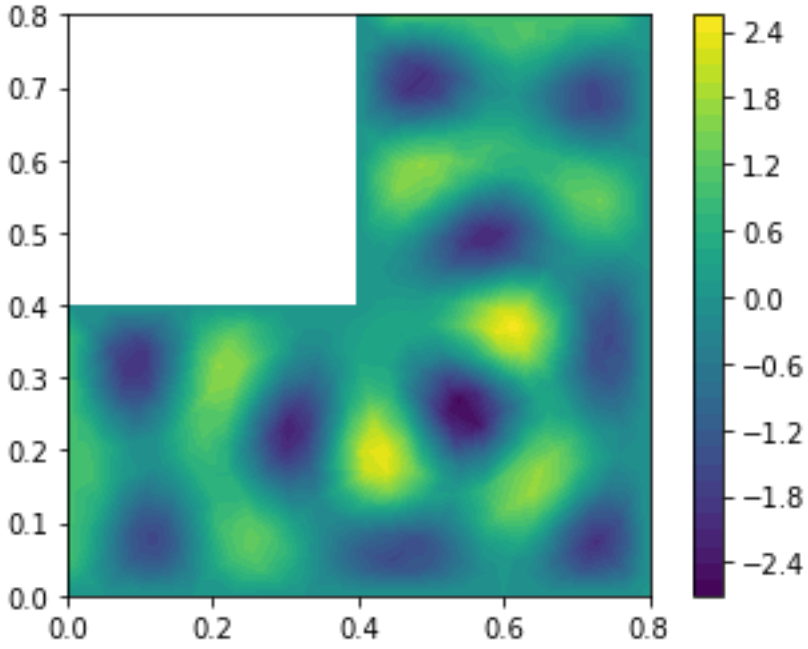

Рисунок 3.7 График решения комплекснозначной функции

По принципу, изложенному выше, решаются все задачи, которые включают в себя комплекснозначные функции, параметры и т.п., поскольку, как говорилось выше, FEniCS не имеет определения для комплексных значений.

3.3 Решение задач на собственные значения

Пакет FEniCS, среды Python также позволяет решать задачи на собственные значения, поскольку в нем определены специальные «решатели». Чтобы продемонстрировать некоторые возможности решения таких уравнений, рассмотрим стандартную проблему собственных значений:

$$Ax = \lambda x. \tag{3.6}$$

Матрица A может быть матрицей жесткости, соответствующей билинейной форме:

$$a(u, v) = \int_{\Omega} \nabla u \cdot \nabla v \, dx. \tag{3.7}$$

Задача рассматривается в области, представленной тетраэдром (Рисунок 3.9).

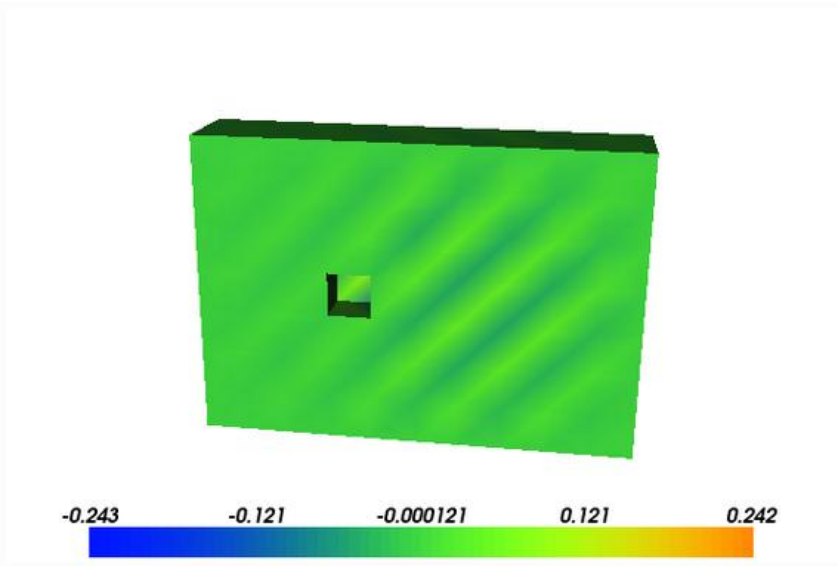

Рисунок 3.8 Область для решения проблемы собственных значений

Для того чтобы построить матрицу A, нужно: определить сетку и пространство тестовых функций, ассоциированного с сеткой; определить вариационную форму, определяющую матрицу и дискретизировать уравнение, собрав при этом в матрицу. Для того, чтобы собрать матрицу, необходимо задать матрицу A как матрицу класса

$PETScMatrix()$ и затем использовать функцию $assemble()$ для того, чтобы из билинейной формы собрать матрицу A.

После задания матрицы, необходимо определить «решатель», который позволяет решать проблем собственных значений в FEniCS. Такой «решатель» определяем с помощью класса $SLEPcEigenSolver()$, который на вход принимает параметры, соответствующие специальным матрицам, определенным с помощью класса $PETScMatrix()$. Решаем исходную задачу с помощью функции $solve()$, определенной в этом классе.

Решение может быть извлечено как реальная и мнимая части вектора собственного значения, с помощью специальной функции: $get\_eigenpair()$, которая принимает на входе параметр, соответствующий количеству собственных значений.

Также в параметрах «решателя» мы можем задавать метод решения проблемы собственных значений, параметр сортировки собственных значений по наибольшей и наименьшей величине.

Для того чтобы по имеющимся собственным значениям и векторам определить искомую функцию, необходимо воспользоваться стандартной функцией Fenics: $vector()$, которая определяет функцию через вектор, соответствующий собственному значению. В поставленной задаче искомая функций задается следующим образом:

$$u = Function(V);$$

$$u.vector() = rx,$$

где rx – это набор векторов, соответствующий нулевому собственному значению, нулевому, поскольку индексации в среде python начинается с числа нуль.

Способом, описанным выше, решается проблема собственных значений во всех задачах, которые связаны с данной проблемой, отличие может быть только в

добавлении граничных условий, либо в определении более сложной проблемы собственных значений.

ГЛАВА 4
ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ КВАНТОВОЙ ТОЧКИ

4.1 Моделирование цилиндрической квантовой точки с учетом осевой симметрии

4.1.1 Разработка алгоритма Matlab

Рассматривается моделируемая система с граничными условиями Дирихле и Неймана

$$\begin{cases} \varepsilon_1(E)\nabla^2\Phi = E\Phi, \ \ (x,y) \in \Omega_1 \\ \varepsilon_1(E)\nabla^2\Phi + V_0\Phi = E\Phi, \ \ (x,y) \in \Omega_2 \\ [\Phi] = 0, \ \ (x,y) \in \Gamma \\ \left[\varepsilon \frac{\partial \Phi}{\partial n}\right] = 0, \ \ (x,y) \in \Gamma \end{cases}, \tag{4.1}$$

где

$$\varepsilon_1(E) = \mathrm{c}\,\mathrm{c}_1\left(\frac{2}{E+E_{1g}} + \frac{1}{E+E_{1g}+\Delta_1}\right),$$

$$\varepsilon_2(E) = \mathrm{c}\,\mathrm{c}_2\left(\frac{2}{E+E_{2g}} + \frac{1}{E+E_{2g}+\Delta_2-V_0}\right),$$

$$\mathrm{c} = \frac{\hbar^2}{2m_0d^2} = 0.381,\ c_1 = \frac{22.2}{3},\ c_2 = \frac{24.2}{3},$$

$$E_{1g} = 0.42, E_{2g} = 1.52,$$

$\Delta_1 = 0.48,\ \Delta_2 = 0.34$ (заданные параметры).

Уравнение рассматривается в прямоугольной области (Рисунок 4.1).

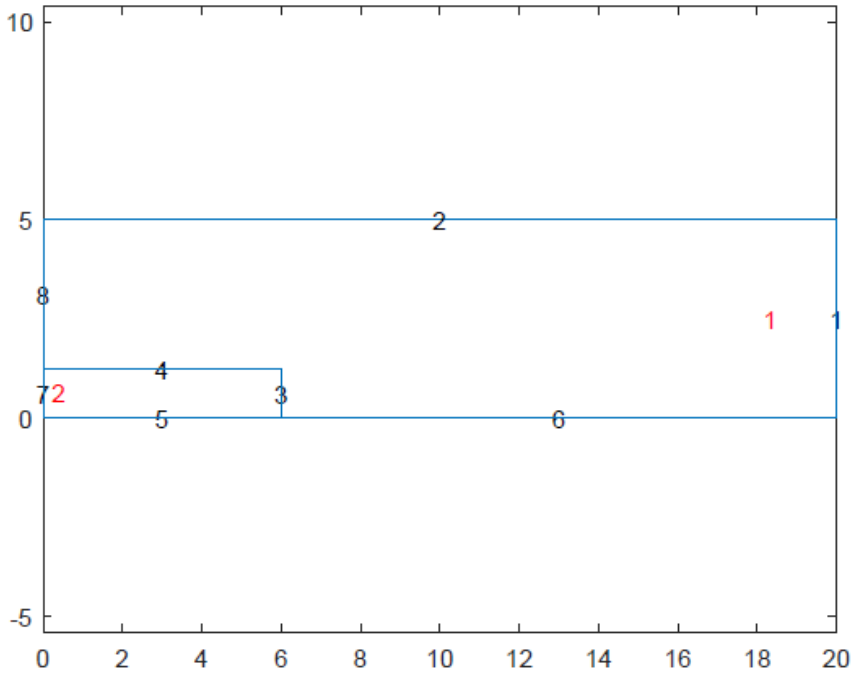

Рисунок 4.1 Моделируемая область

Для задания области, в которой моделируется система необходимо использовать класс $createpde$ программы Matlab, которая позволяет задавать любые двухмерные области:

$$modelQuantumDot = createpde.$$

Первый шаг. Задаем прямоугольную область, которая представляет внешний цилиндр:

$$gd = [2, 4, 0\ R\ R\ 0, 0\ 0\ z\ z]'.$$

Задаем прямоугольную область, представляющую саму квантовую точку

$$gd = [gd\ [2, 4, 0\ R0\ R0\ 0, 0\ 0\ z0/1\ z0/2]'],$$

где $z0$ – высота внутреннего цилиндра, $R0$ – радиус внутреннего цилиндра, z – высота внешнего цилиндра, R – радиус внешнего цилиндра, в данной и

последующих задачах внутренний цилиндр представляет квантовую точку, а внешний цилиндр – область в которой рассматривается квантовая точка.

Далее необходимо объединить области с помощью функциональности Matlab:

$$g = decsg(gd);$$

$$geometryFromEdges(modelQuantumDot, g).$$

Второй шаг. Конструируем сетку на основе построенной области с помощью соответствующей функции:

$$mesh =$$

$$= generateMesh(modelQuantumDot, 'Hmax', 0.1, 'GeometricOrder', 'linear').$$

Далее, по синтаксису программы Matlab, необходимо ассоциировать построенную сетку с областью:

$$modelQuantumDot.Mesh = mesh.$$

Для данной системы необходимо задавать все необходимые граничные условия: условия Неймана и Дирихле следующим образом:

$$applyBoundaryCondition(modelQuantumDot, 'Edge', [1,2], 'h', 1, 'r', 0);$$

$$applyBoundaryCondition(modelQuantumDot, 'Edge', [5,6,7,8], 'q', 0, 'g', 0).$$

Условие Дирихле определяет поведение функции на границе области: на границе рассматриваемой области функция должна обращаться в нуль. Условие Неймана необходимо для задания симметричности по осям области решения задачи.

Шаг третий. Определяем соответствующие уравнения для рассматриваемой системы и ассоциируем их с областями, в которых они должны быть решены.

Шаг четвертый. Строим итерационный алгоритм для достижения самосогласованного результата энергетических состояний. Для того, чтобы решить поставленную задачу, в программе Matlab есть специальный пакет функций для

решения проблемы собственных значений PDE, соответственно и специальные функции, такие как $pdeeig$. Подробный код решения можно посмотреть в Приложение А.

Изолинии получившейся волновой функции выглядят следующим образом:

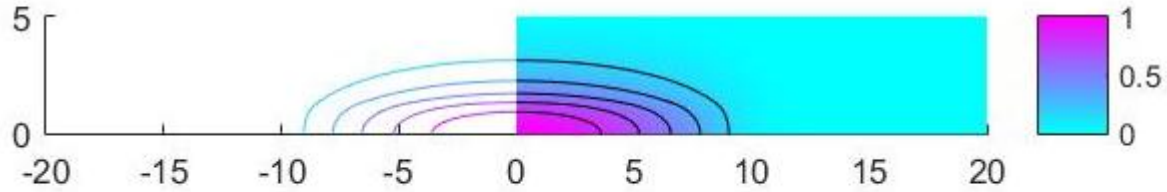

Рис. 4.1 Изолинии волновой функции

4.1.2 Разработка алгоритма средствами пакета FEniCS

Рассматриваем моделируемую систему (4.1) с соответствующими граничными условиями для того чтобы было соответствие с исходной цилиндрической областью в первом квадранте двумерной области.

Для решения системы (4.1) рассматриваем область следующего вида:

внешний прямоугольник представляет внешний цилиндр области $(z_0, R_{external})$, а внутренний прямоугольник представляет внутренний цилиндр $(\frac{z_0}{2}, z_{external})$, который представляет квантовую точку. Область представляет собой осевое сечение исходной задачи (2.7). В среде Python область задается следующим образом:

$$domain = Rectangle(Point(0,0), Point(R\_external, z\_external)).$$

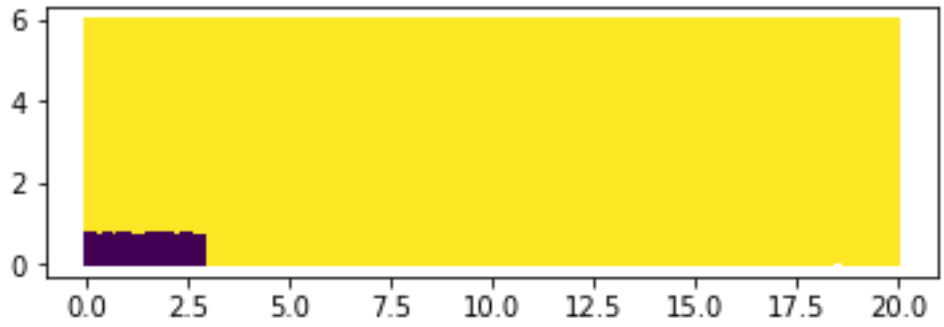

Рисунок 4.2 Область для решения осесимметричной задачи.

Таким образом мы определили прямоугольник, где $R\_external, z\_external$ – это радиус и высота внешнего цилиндра, соответственно.

Далее нужно определить соответствующие подобласти, представляющие цилиндрическую квантовую точку и область, в которой она рассматривается:

$$rec1 = Rectangle(Point(R0,0), Point(R\_external, z0/2));$$

$$rec2 = Rectangle(Point(0, z0/2), Point(R0, z\_external));$$

$$rec3 = Rectangle(Point(R0, z0/2), Point(20, z\_external));$$

$$omega1 = Rectangle\left(Point(0,0), Point\left(R0, \frac{z0}{2}\right)\right);$$

$$omega2 = rec1 + rec2 + rec3.$$

Последним шагом задания области для решения уравнения, является включение подобластей в область для дальнейшего построения сетки.

$$domain.set\_subdomain(1, omega1);$$

$$domain.set\_subdomain(2, omega2).$$

Таким образом, мы не только включили в прямоугольную область соответствующие подобласти, представляющие цилиндрическую квантовую точку

и область, в которой она рассматривается, но и промаркировали для дальнейшего решения.

После определения области и соответствующих подобластей строим сетку путем триангуляции исходной области с максимальной длиной стороны треугольника равной 0.1:

$$mesh = generate\_mesh(domain, 30).$$

Построенная сетка выглядит следующим образом:

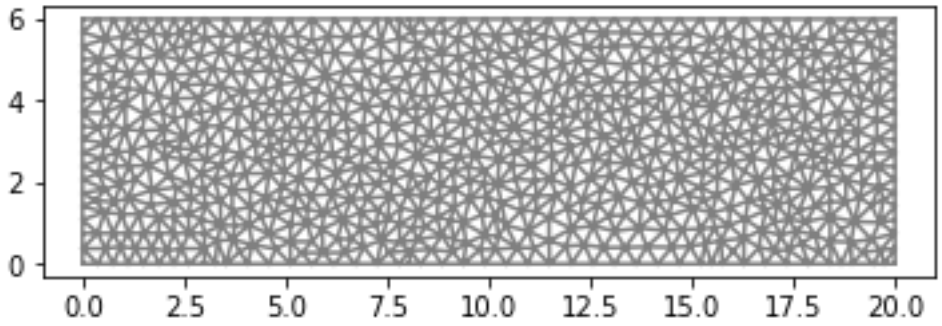

Рисунок 4.3 Триангуляция рассматриваемой области

Как можно заметить, FEniCS строит сетку отдельно для каждой из подобластей, что позволяет в дальнейшем практически исключить погрешность вычислений.

Далее задаем линейное пространство Лагранжа, ассоциированное с построенной сеткой, которое используем для реализации МКЭ:

$$V = FunctionSpace(mesh, 'CG', 1).$$

После этого задаем граничные условия, которые предполагают, что на границе области по осям x и y функция должна быть симметричной. Формулируем вариационную проблему, основываясь на вариационной форме исходной задачи (2.13) и задаем уравнение в соответствии с синтаксисом пакета FEniCS:

$$c1 = (k * (coeff * 22.2/\ 3.* (2./(E + E1g) + 1./(E + E1g + delta1))) *$$

$$* inner(grad(u), grad(v)) + k * 0 * inner(u, v)) * dx(1);$$

$$c2 = (k * (coeff * 24.2/3.* (2./(E + E2g - V0) + 1./(E + E2g + delta2 -$$

$$-V0))) * inner(grad(u), grad(v)) + k * V0 * inner(u, v)) * dx(2);$$

$$a = c1 + c2;$$

$$m = k * inner(u, v) * dx, \quad (16)$$

где

$$k = Expression(x[0], degree = 1),$$

$$coeff = \frac{\hbar^2}{2m_0 d^2} = 0.381,$$

$$delta1 = 0.48, delta2 = 0.34,$$

$$E1g = 0.42, E2g = 1.52,$$

$dx(1)$ – обозначение, что уравнение должно быть решено в подобласти, маркированной номером 1, $dx(2)$ – обозначение, что уравнение должно быть решено в подобласти, маркированной номером 2, $a -$ билинейная форма, $m -$ правая часть проблемы собственных значений.

Далее решаем проблему собственных значений и для этого нужно собирать матрицы правой и левой частей, соответствующие билинейной форме, a и m. Как говорилось выше, для того, чтобы собрать матрицы, необходимо воспользоваться классом $PETScMatrix$:

$$A = PETScMatrix();$$

$$B = PETScMatrix();$$

$$assemble(a, tensor = A);$$

$$assemble(m, tensor = B).$$

Следующий шаг предполагает решение проблемы собственных значений уравнения Шредингера, которое и задает моделируемую систему с помощью класса

$SLEPcEigenSolver$ и функции, позволяющей получить решение проблемы собственных значение $solve()$.

Для решения проблемы собственных значений в FEniCS реализован специальный класс $SLEPcEigenSolver$, который дает возможность получать необходимые решения. Первое, с помощью МКЭ дискретизируем уравнение Шредингера с граничными условиями и получаем матрицу A, также дискретизиурем правую часть уравнения, которая представляет соответствующие собственные векторы в стандартной формулировке проблемы собственных значений. С помощью класса $SLEPcEigenSolver$ создаем «solver» который в качестве входных параметров принимает матрицы A, B как левую и правую части проблемы собственных значений. Класс $SLEPcEigenSolver$ предоставляет возможность задать определенные параметры для «solver», в данной задаче нам понадобиться параметр, который из всех собственных значений выбирает их с наименьшей величиной. Далее с помощью стандартных функций решается соответствующее уравнения. В качестве искомого решения берем первое собственное значение и соответствующий ему вектор, которые представляют энергию и волновую функцию, соответственно, используя функцию $get\_eigenpair()$, где в качестве входного значения берется количество собственных значений, либо одно, если входной параметр равен нулю, поскольку нумерация в программе Python начинается с нуля.

Подробный код решения исходной задачи в осесимметричном случае представлен в Приложении В.

4.2 Моделирование трехмерной квантовой точки

Рассматриваем моделируемую систему (2.10) в 3D области. Для решения уравнение (2.10) используем первый квадрант в качестве используемой области, а для того, чтобы решение согласовывалось с областью, определенной для уравнения (2.10)

задаем граничные условия, которые показывают симметричность относительно координат x, y и z.

Для того, чтобы задать используемую область, способом, изложенным в пункте 3.1 воспользоваться нельзя, поскольку в FEniCS нет возможности явно определять подобласти в 3D пространстве, поэтому область, определяющую квантовую точку задаем с помощью класса – множества точек, заданных в области, представляющей первый квадрант цилиндра.

Шаг первый. Определяем область и задаем сетку, с помощью функции, с помощью которой мы задавали область для решения уравнения (2.12) в прямоугольной области:

$$box = Box(Point(R\_external, 0, 0), Point(0, R\_external, z\_external));$$

$$domain = box;$$

$$mesh = generate\_mesh(domain, 37).$$

Получаем сетку, соответствующую заданной области:

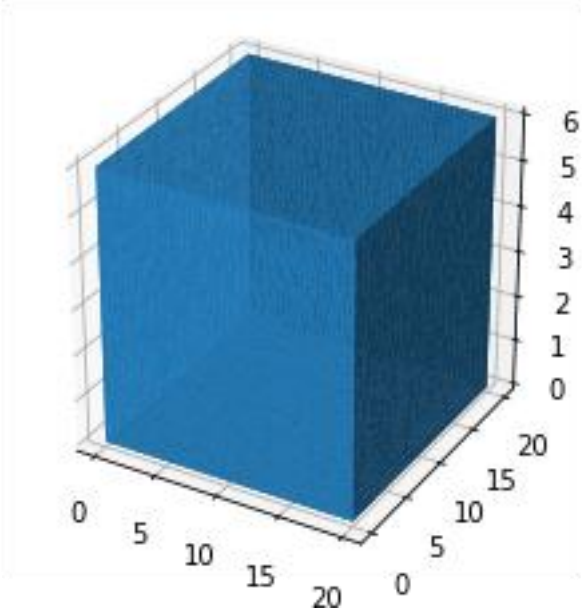

Рисунок 4.4 Триангуляция области в 3D

Шаг второй. Определяем цилиндрическую подобласть, которая представляет квантовую точку как класс – множество точек, подчиненных определенному правилу и включаем заданную подобласть в исходную область.

Заданная цилиндрическая подобласть в первом квадранте выглядит следующим образом:

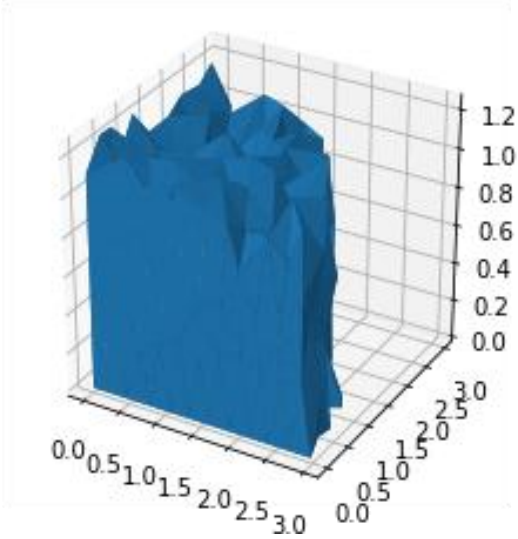

Рисунок 4.5 Подобласть, представляющая квантовую точку

Шаг третий. Определяем пространство тестовых функций, путем задания линейного пространства Лагранжа:

$$V = FunctionSpace(mesh, 'CG', 1).$$

Шаг третий. Определяем граничные условия Дирихле, которые показывают симметричность поставленной задачи относительно осей x, y, z.

Шаг четвертый. Определяем уравнение, используя вариационную формулировку (2.13) и задаем матрицы, соответствующие билинейной форме и правой части проблемы собственных значений, а они будут аналогичные тем, которые мы определяли для уравнения (2.12) в прямоугольной области:

$$\text{найти } u \in V$$

$$V = \{u: u \in H^1(\Omega), u = 0 \text{ на } \partial\Omega_2\}$$

$$\varepsilon_1(E)\int_{\Omega_1} \nabla v \nabla u \, dxdydz + \varepsilon_2(E)\int_{\Omega_2} (\nabla v \nabla u + \tag{4.2}$$

$$+V_0 u\, v)dxdydz = \int_{\Omega} E\, v\, u\, dxdydz.$$

Шаг пятый. Задаем проблему собственных значений решением которого является пара значений: собственное значение и вектор, соответствующий собственному значению, представляющие энергию и волновую функцию.

После нахождения энергетических состояний и волновых функций, строим изолинии для того чтобы показать вероятность нахождения электрона в области (Рисунок 4.6).[5]

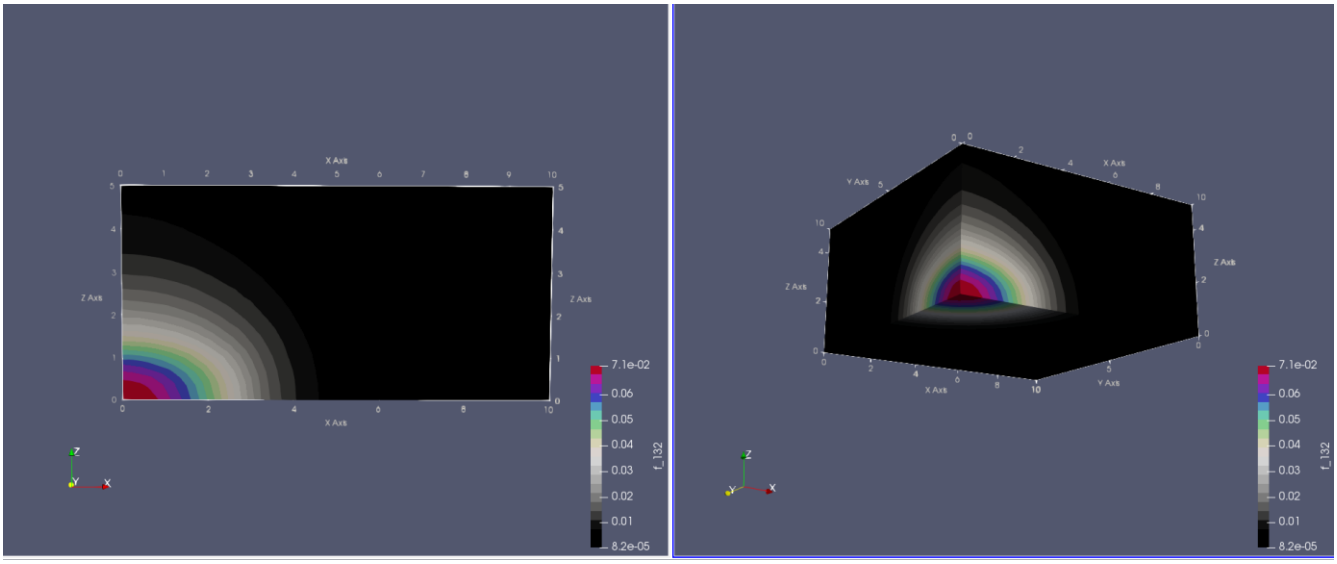

Рисунок 4.6 Изолинии найденной волновой функции

В данном случае по графику изолиний можно увидеть, что вероятность нахождения электрона в рассматриваемой области уменьшается в положительном направлении координатных осей.

4.3 Построение итерационного алгоритма для решения исходной задачи

Из-за зависимости энергии от эффективной массы электрона размеров квантовой точки, расчет должен состоять из нескольких итерационных циклов, для достижения результата для энергии с заданной точностью.

Итерационный алгоритм выглядит следующим образом:

1. Задаем начальное значение энергии равным нулю
2. Решаем уравнения Шредингера с заданными граничными условиями.
3. Обновляем значение энергии и используем его для расчета последующей энергии.
4. Если разность полученного значения и использованного меньше, чем 0,000005, тогда останавливаем алгоритм, иначе переходим к шагу 2.

По вышеописанному алгоритму нужно использовать цикл while в среде Python для достижения необходимого результата, поскольку в цикле while мы можем посчитать количество итераций, которые прошли до достижения необходимой точности в расчете энергии квантовой точки. Реализацию итерационного алгоритма можно посмотреть в Приложении А.

Итерационный алгоритм используется для расчета энергии как в осесимметричном случае, так и в цилиндрическом, изменяются только входные параметры.

Основываясь на результатах рассматриваемой статьи, необходимой точности в расчете энергии мы должны достигнуть за 8-9 итераций, что и подтверждает реализованный итерационный алгоритм.

Подробный код реализации вышеописанного алгоритма и решения задачи в 3D области представлен в Приложении С.

ГЛАВА 5
ТЕСТИРОВАНИЕ ВЫЧИСЛЕНИЙ

5.1 Тестирование вычислений для осесимметричной задачи

Данный раздел посвящен тестированию представленного выше решения уравнения (2.12). Для тестирования были взяты параметры и рассчитаны в двух системах Matlab и FEniCS и построен сравнительный график. Для среды Matlab невозможно решить поставленное уравнение с квадратичным пространством, что является преимуществом среды Python.

Для расчетов были использованы следующие параметры:

1. Линейное пространство:
    a. $z_0 = \{1.5, 2.5\}, R0 = [3, 15]$
2. Квадратичное пространство:
    a. $z_0 = \{1.5, 2.5\}, R0 = [3, 15]$.

Основываясь на вышеприведенном графике, построенном с помощью программы Matlab, графиком из рассматриваемой статьи [8] можно сделать следующие выводы:

Качественно численные результаты в Matlab и FEniCS совпадают и согласуются с результатами в статье (Рисунок 5.1)

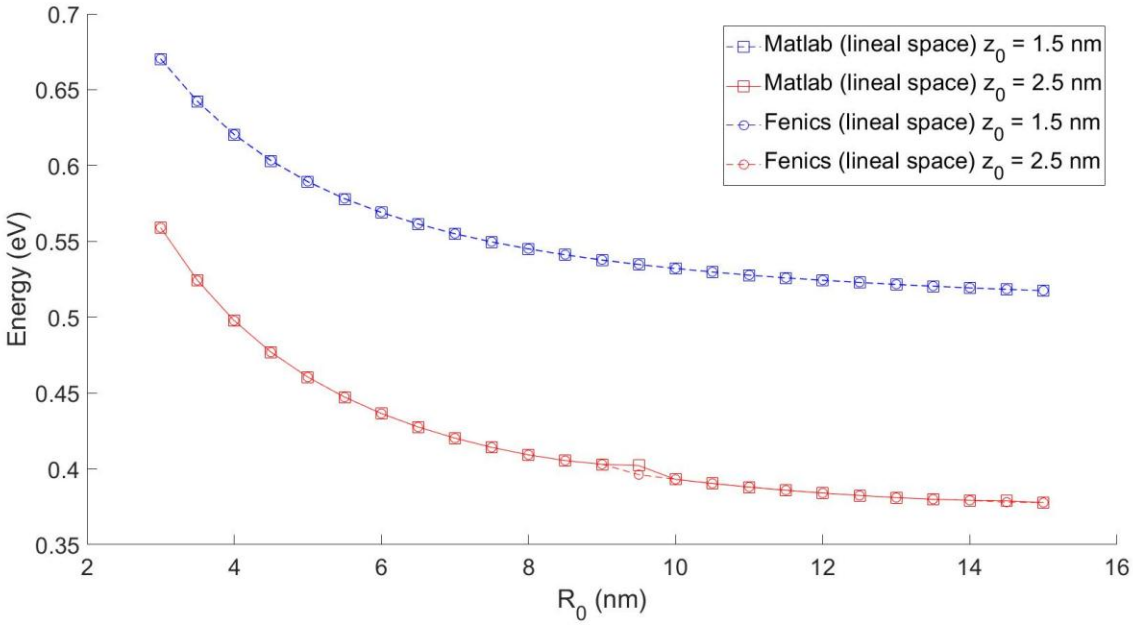

Рисунок 5.1 График зависимости энергии от радиуса квантовой точки, вычисленной в программных средах Matlab и FEniCS

При одинаковых геометрических параметрах (размер квантовой точки, размер внешней области, размер сетки) для линейных конечных элементов максимальная относительная погрешность для численных значений энергии, посчитанный средствами Matlab и FEniCS составляет 0.115626 %.

Вывод: количественное совпадение численных результатов в Matlab и FEniCS

Для расчетов в FEniCS максимальная относительная погрешность для численных значений энергии, посчитанных на линейных элементах (размер сетки равен 0.1) и квадратичных элементах (размер сетки 0.6), составляет 0.626535%.

График зависимости энергии от радиуса квантовой точки, вычисленной в линейных и квадратичных пространствах программной среды FEniCS (Рисунок 5.2).

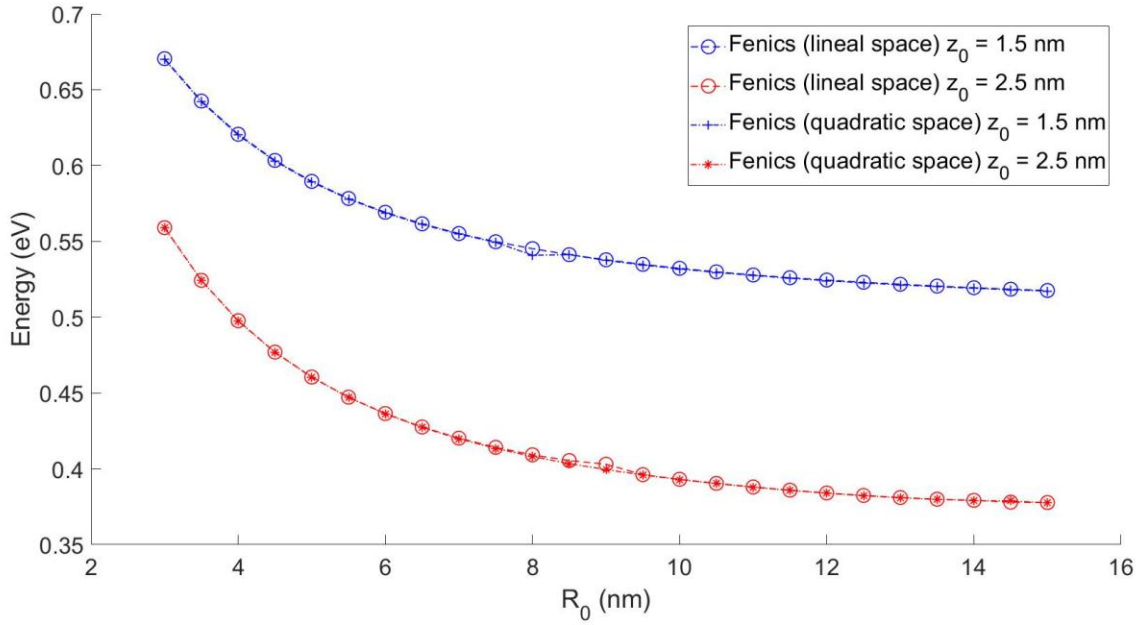

Рисунок 5.2 График зависимости энергии от радиуса квантовой точки, вычисленной в программных средах Matlab и FEniCS

Вывод: применение квадратичных элементов вместо линейных элементов позволяет осуществлять расчеты на более грубых сетках.

5.2 Тестирование вычислений для 3D области

Для тестирования результатов в 3D области был построен алгоритм в среде FEniCS и среде Matlab, из-за технической невозможности определения подобластей явным образом, как в среде Matlab, так и в среде FEniCS, для первой среды использовалось определения функции на различных положениях и определения класса элементов, в случае второй среды.

Для расчетов были использованы следующие параметры:

1. Линейное пространство:
    a. $z_0 = \{2.5\}, R0 = \{3, 6, 9\}$.

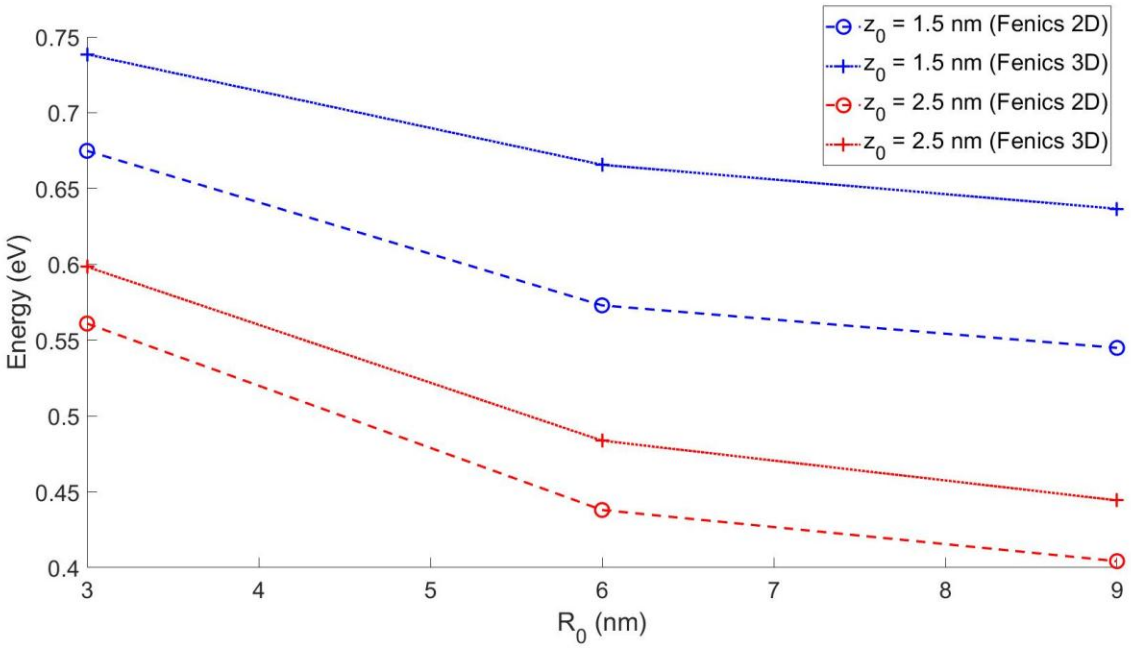

Рисунок 5.3 График зависимости энергии от радиуса квантовой точки, вычисленной в линейных и квадратичных пространствах программных средах Matlab и FEniCS

Тестовые вычисления в Python для трехмерной модели иллюстрируют потерю точности. Проблема связана с отсутствием функционала для построения сетки, который явно учитывает геометрию поверхности раздела двух сред.

ЗАКЛЮЧЕНИЕ

По результатам решения задачи на основе математической модели, взятой из вышеуказанной статьи [8], при помощи программных средств Matlab и Python и последующего сопоставления полученного графика зависимости энергии от размеров цилиндрической квантовой точки с аналогичным графиком, содержащимся в вышеприведённой статье установлено, что результаты имеют схожие значения с незначительной погрешностью, показывая одинаковую зависимость энергии от размеров цилиндрической квантовой точки.

Тестирование метода для программных средств Matlab и Python показало, что вышеописанный подход с использованием указанных вычислительных программ дает правильные решения с относительной ошибкой 0,1%, из чего можно сделать вывод, что метод применим к численному решению проблемы на собственные значения.

При построении конечно – элементного пространства для решения задачи на собственные значения с помощью метода конечных элементов использовались линейные и квадратичные элементы. Тестирование метода с использованием разных типов элементов в программе Python, показало, что, применение квадратичных элементов при определении пространства позволяет использовать более грубые сетки.

Также важно отметить, что трехмерное моделирование рассматриваемой системы является проблематичным средствами FEniCS на Python из-за нехватки функциональности рассматриваемых программных средств, поскольку, не представляется возможным построение сложной геометрии вышеописанной системы с разделением областей для дальнейшего построения сетки, используемой в МКЭ. Для построения сложной геометрии рассматривалось программное

обеспечение Freecad, которое позволяет конструировать точные геометрические объекты и на основе их строить более мелкие сетки, но, после использования данной геометрии в программе Python, оказалось, что функционал данной программы не может обработать более качественную сетку, из-за нехватки имеющихся вычислительных мощностей.

Поскольку Python, после применения конечно-элементного подхода к решению задач на собственные значения представляет определенную структуру решения, появилась проблема построения графика изолиний для представления конечного результат. Для того, чтобы показать результаты решения более качественно как для осесимметричного случая, так и для трехмерной модели, использовалась программное обеспечение Paraview, которое позволяет, используя структуру решения Python, строить разнообразные графики.

Проверка конечно-разностного подхода к решению задач на собственные значения является необходимым звеном в изучении энергетических состояний, поскольку многие вычислительные программы имеют возможности для решения задач с помощью метода конечных элементов.

В дальнейшем на основе результатов данной работы возможно рассмотрение сферических квантовых точек, изучений их свойств и поведения. Далее предполагается рассмотрение системы квантовых точек с целью изучения их поведения в случае их взаимодействия, анализ энергетических состояний и волновых функций, которые получаются как решение уравнения Шредингера.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Куприяшкин, А.Г. Основы моделирования систем: учеб. пособие / А.Г. Куприяшкин; Норильский индустр. ин-т. – Норильск: НИИ, 2015. – 135 с.
2. Тауц Я. Фото- и термоэлектрические явления в полупроводниках. М.: Издательство иностранной литературы, 1962. – 256 с.
3. V.Klimov . Semiconductor and metal nanocrystals. - New York, Marcel Dekker Inc. – 2004.
4. Образовательный ресурс. – Режим доступа: https://ru.wikipedia.org/wiki/Вики
5. Morten Willatzen, Lok C. Lew Yan Voon.Separable Boundary-Value Problems in Physics – 2011. – 399 p.
6. Васильев Р.Б., Дирин Д.Н. Квантовые точки: синтез, свойства, применение: учеб. пособие / Васильев Р.Б., Дирин Д.Н. Московский Ордена Лениниа, Ордена Октябрьской Революции и Ордена Трудового Красного Знамени, Государственный Университет им. М.В.Ломоносова, Фак. наук о материалах – Москва, 2007. – 34 стр.
7. Презентация: Методы конечных элементов. – Режим доступа: http://portal.tpu.ru:7777/SHARED/b/BGA/bio/bachelors/Tab/MKE.pdf
8. Li Y., Voskoboynikov O., Lee C.P., Sze S.M. // Solid State Communications. – 2001. – 5 p.
9. Hans Petter Langtangen, Anders Logg. Solving PDEs in Python – The FEniCS Tutorial Volume I. – 2017. – 153 p.
10. Schrodinger wave function. Режим доступа: – http://physics.mq.edu.au/~jcresser/Phys201/LectureNotes/SchrodingerEqn.pdf

ПРИЛОЖЕНИЕ А. Код программы на Matlab

Ниже приведен код алгоритма на Matlab.

Parameters

% space variables are dimensionless over nm

z0 = 1.5; % height of the inner cylinder

R0 = 6; % radius of the inner cylinder

z = 5; % height of the outer (artificial) cylinder

R = 20; % radius of the outer (artificial) cylinder

%No scale for energies. Energy is measured in eV

V0 = 0.77;

E1g = 0.42;

E2g = 1.52;

delta1 = 0.48;

delta2 = 0.34;

coeff = 0.0381; % coeff = h_Plank^2/2/mass_electron/d^2

% create PDEModel object

modelQuantumDot = createpde; % for scalar equation

gd = [2, 4, 0 R R 0, 0 0 z z]′; % a polygon solid for the whole domain [0, R] x [0, z]

$$gd = \left[gd \left[\frac{2, 4, 0\ R0\ R0\ 0, 0\ 0 \frac{z0}{1} z0}{2} \right]' \right];$$

% a polygon solid for the quantum dot [0, R0] x [0, z0/2]

g = decsg(gd); % corresponds to the union of all objects in gd

geometryFromEdges(modelQuantumDot, g);

% mesh construction

```
mesh
= generateMesh(modelQuantumDot,' Hmax', 0.1,' GeometricOrder',' linear');
%Hmax − maximum mesh edge length, GeometricOrder − element type
[p, e, t] = meshToPet(mesh);

modelQuantumDot. Mesh = mesh;

% boundary conditions
% Dirichlet conditions
applyBoundaryCondition(modelQuantumDot, 'Edge', [1,2], 'h', 1, 'r', 0);
%Neumann conditions
applyBoundaryCondition(modelQuantumDot, 'Edge', [5,6,7,8], 'q', 0, 'g', 0);
% Remark: we get from g
% x = R: 1 − − set Dirichlet u = 0
% y = z: 2 − − set Dirichlet u = 0
% y = 0: 5 и 6 − − set Neumann due to symmetry
% x = 0: 7 и 8 − − set Neumann due to axisymmetry

% describe and solve the problem
c1 = 'x ∗ coeff ∗ 22.2/3.∗ (2./(E + E1g) + 1./(E + E1g + delta1))';
c2 = 'x ∗ coeff ∗ 24.2/3.∗ (2./(E + E2g − V0) + 1./(E + E2g + delta2 − V0))';
a1 = '0';
a2 = 'x ∗ V0';
c = [strjoin({c2,' !', c1},'' )];
% geometry g denotes the quantum dot as the domain 2
a = [strjoin({a2,' !', a1},'' )];
% geometry g denotes the quantum dot as the domain 2
a = strrep(a, 'V0', num2str(V0));
c = strrep(c, 'V0', num2str(V0));
c = strrep(c, 'coeff', num2str(coeff));
```

```
c = strrep(c,'E1g',num2str(E1g));
c = strrep(c,'E2g',num2str(E2g));
c = strrep(c,'delta1',num2str(delta1));
c = strrep(c,'delta2',num2str(delta2));
c_tmp = c

Energy = 0; % initial value
Energy_old = 1;
k = 0;
while abs((Energy - Energy_old)/Energy_old) > 10^(-5)
  k = k + 1;
  c = strrep(c_tmp,'E',num2str(Energy));
  [v,E] = pdeeig(modelQuantumDot,c,a,'x',[0 2]); % pdeeig for 2 regions
  Energy_old = Energy;
  Energy = E(1) % the first eigenvalue
  pause(0.2)
end
k
```

ПРИЛОЖЕНИЕ В. Код программы в Python для осесимметричного случая

Ниже приведен код алгоритма на Python, пакет FEniCS для осесимметричного случая.

```
from dolfin import *
from fenics import *
from mshr import *
%matplotlib inline
from matplotlib import pyplot
from numpy import *
import matplotlib.pyplot as plt
import matplotlib.tri as tri
z0 = 2.5 #thiknes of quantum dot
R0 = 3   #Radius of quantum dot
R_external = 10 # radius of the external cylinder
z_external = 5 # height of the external cylinder

#Define the external domain
domain = Rectangle(Point(0,0), Point(R_external, z_external))
rec1 = Rectangle(Point(R0,0), Point(R_external, z0/2));
rec2 = Rectangle(Point(0, z0/2), Point(R0, z_external));
rec3 = Rectangle(Point(R0, z0/2), Point(R_external, z_external));
omega2 = rec1 + rec2 + rec3;

#Define the internal domain
omega1 = Rectangle(Point(0,0), Point(R0, z0/2));
domain.set_subdomain(1, omega1)
```

```
domain. set_subdomain(2, omega2)
mesh = generate_mesh(domain, 20)

#define subdomains
sub_domains = MeshFunction('size_t', mesh, 2, mesh. domains())
dx = Measure('dx', domain = mesh, subdomain_data = sub_domains)

#define mesh and function space
V = FunctionSpace(mesh, 'CG', 1)
plot(mesh)
plot(sub_domains)
#build essential boundary conditions
tol = 1E − 14
t = Constant(0.0)
def boundary(x, on_boundary):
  tol = 1E − 14
   return on_boundary and (near(x[1], z_external, tol) or near(x[0], R_external, tol))

#Initial values
V0 = 0.77 # the confinement potential outside the quantum dot
E1g = 0.42 # the energy gap for InAs
E2g = 1.52 # the energy gap for GaAs
delta1 = 0.48 # the spin − orbit splitting for InAs
delta2 = 0.34 # the spin − orbit splitting for GaAs
E = 0 # energy of the ground state
coeff = 0.0381; #coeff = h_Plank^2/2/mass_electron/d^2
k = Expression("x[0]", degree = 1)

#Solve eigenvalues problem
u = TrialFunction(V)
```

```
v = TestFunction(V)
bc = DirichletBC(V, t, boundary)
E = 0; #initial value
Energy_old = 1;
k1 = 0;
#Create loop for 'self − consistent'
while (abs((E − Energy_old)) > 0.0000005):
  c1 = (k ∗ (coeff ∗ 22.2/3.∗ (2./(E + E1g) + 1./(E + E1g + delta1)))
        ∗ inner(grad(u), grad(v)) + k ∗ 0 ∗ inner(u, v)) ∗ dx(1)
  c2 = (k ∗ (coeff ∗ 24.2/3.∗ (2./(E + E2g − V0) + 1./(E + E2g + delta2 − V0)))
        ∗ inner(grad(u), grad(v)) + k ∗ V0 ∗ inner(u, v)) ∗ dx(2)
  a = c1 + c2
  m = k ∗ inner(u, v) ∗ dx
  A = PETScMatrix()
  assemble(a, tensor = A)
  B = PETScMatrix()
  assemble(m, tensor = B)
  bc.apply(A)        # apply the boundary conditions
  bc.apply(B)
  eigensolver = SLEPcEigenSolver(A, B)
  eigensolver.parameters['spectrum'] = 'smallest magnitude'
  eigensolver.solve(0)
  r, c, rx, cx = eigensolver.get_eigenpair(0)
  print("Energy: ", r)
  Energy_old = E
  E = r #first eigenvalue
  k1 = k1 + 1;
end

#Plot the function
```

```
# Initialize function and assign eigenvector
u = Function(V)
u.vector()[:] = rx

# Plot eigenfunction
p = plot(u, mesh = mesh)
#pyplot.colorbar(p);

#Plot isolines of function
triang = tri.Triangulation(* mesh.coordinates().reshape((-1, 2)).T,
                           triangles = mesh.cells())
Z = u.compute_vertex_values(mesh)
plt.figure()
plt.tricontourf(triang, Z)
plt.colorbar()
plt.show()
```

ПРИЛОЖЕНИЕ С. Код программы в Python для трехмерной области

Ниже приведен код алгоритма на Python, пакет FEniCS для трехмерной области.

```
from dolfin import *
from fenics import *
from mshr import *
%matplotlib inline
from matplotlib import pyplot

#Initial values
z0 = 2.5 #thiknes of quantum dot
R0 = 3   #Radius of quantum dot
R_external = 10 # radius of the external cylinder
z_external = 5 # height of the external cylinder

V0 = 0.77 # the confinement potential outside the quantum dot
E1g = 0.42 # the energy gap for InAs
E2g = 1.52 # the energy gap for GaAs
delta1 = 0.48 # the spin − orbit splitting for InAs
delta2 = 0.34 # the spin − orbit splitting for GaAs
E = 0 # energy of the ground state
coeff = 0.0381; #coeff = h_Plank^2/2/mass_electron/d^2

#Define the external domain
box = Box(Point(R_external, 0, 0), Point(0, R_external, z_external))
```

```
#Define the mesh
mesh = generate_mesh(box, 50)
plot(mesh)

#define subdomains
#Define cylinder subdomain
tol = 1E − 14
class subdomain_cylinder(dolfin. SubDomain):
   def inside(self, x, on_boundary):
      return (x[2] <= z0/2 + tol) and (x[0] ∗ x[0] + x[1] ∗ x[1] <= R0 ∗ R0 + tol)

#Mark subdomain
Subdomain = subdomain_cylinder()
domains = CellFunction("size_t", mesh)
domains. set_all(0)
Subdomain. mark(domains, 1)
submesh = SubMesh(mesh, domains, 1)
dx = Measure('dx', domain = mesh, subdomain_data = domains)
plot(submesh)

#Define function cpace
element = FiniteElement("Lagrange", mesh. ufl_cell(), 1)
V = FunctionSpace(mesh, element)

#build essential boundary conditions
tol = 1E − 14
t = Constant(0.0)
def boundary(x, on_boundary):
   tol = 1E − 14
   return on_boundary and (near(x[2], z_external, tol) or
```

```
        near(x[0], R_external, tol) or near(x[1], R_external, tol))
bc = DirichletBC(V, t, boundary)
#Solve eigenvalues problem
u = TrialFunction(V)
v = TestFunction(V)
bc = DirichletBC(V, t, boundary)
E = 0; #initial value
Energy_old = 1;
k1 = 0;
#Create loop for 'self − consistent'
while (abs((E − Energy_old)) > 0.0000005):
  c1 = (coeff * 22.2/3.* (2./(E + E1g) + 1./(E + E1g + delta1))
        * dot(grad(u), grad(v)) + 0 * dot(u, v)) * dx(1)
  c2 = (coeff * 24.2/3.* (2./(E + E2g − V0) + 1./(E + E2g + delta2 − V0))
        * dot(grad(u), grad(v)) + V0 * dot(u, v)) * dx(0)
  a = c1 + c2
  m = k * inner(u, v) * dx
  A = PETScMatrix()
  assemble(a, tensor = A)
  B = PETScMatrix()
  assemble(m, tensor = B)
  bc.apply(A)        # apply the boundary conditions
  bc.apply(B)
  eigensolver = SLEPcEigenSolver(A, B)
  eigensolver.parameters['spectrum'] = 'smallest magnitude'
  eigensolver.solve(0)
  r, c, rx, cx = eigensolver.get_eigenpair(0)
  print("Energy: ", r)
  Energy_old = E
  E = r #first eigenvalue
```

```
  k1 = k1 + 1;
end

#Plot the function
# Initialize function and assign eigenvector
u = Function(V)
u.vector()[:] = rx

# Plot eigenfunction
p = plot(u, mesh = mesh)
#pyplot.colorbar(p);
```