# Аннотация

Выпускная квалификационная работа изложена на 77 страницах, состоит из введения, трех глав, заключения, включает 21 рисунок, 11 таблиц, библиографический список из \_\_ источников литературы, 5 приложений на 52 страницах.

Объектом выпускной квалификационной работы является разработка программной системы определения типа поверхности удаленных объектов поляриметрическими методами.

Предметом – математические модели и алгоритмы для программных комплексов вычисления оптических параметров покрытий космических объектов.

Цель выпускной квалификационной работы – разработка новой версии программной системы определения типа поверхности удаленных объектов поляриметрическими методами с использованием современных средств проектирования, которое позволит использовать разработанную систему в исследовательских и учебных целях.

При разработке системы использовались следующие средства разработки:

* *python* *3*;
* формат базы данных: *sqlite3*;
* *Visual Studio Code*;
* *git*.

Выпускная квалификационная работа и ее результаты могут быть использованы в исследовательских и учебных целях.

# Задание на дипломное проектирование

# Содержание

|  |  |  |
| --- | --- | --- |
|  | **Введение** | 5 |
| **1** | **Математические модели и алгоритмы прямой и обратной задачи фотометрии** | 9 |
| 1.1 | Постановка задачи | 9 |
| 1.2 | Модель рассеяния света от поверхности космического объекта и регистрации светового пучка | 10 |
| 1.3 | Применение вектор-функции Стокса | 18 |
| 1.4 | Расчет вектора Стокса падающего излучения | 25 |
| **2** | **Современные методы и алгоритмы определения типа покрытия космических объектов** | 29 |
| 2.1 | Постановка задачи | 29 |
| 2.2 | Определение типа покрытия космических объектов | 29 |
| 2.3 | Классификация типа покрытия | 35 |
| 2.4 | Алгоритм вычисления показателя преломления однородной диэлектрической поверхностью | 35 |
| 2.5 | Алгоритм вычисления показателя преломления металлических покрытий | 39 |
| **3** | **Программная система вычисления показателя преломления поверхностей космических объектов и определения материала покрытий STOKES** | 43 |
| 3.1 | Обоснование выбора средств разработки программной системы | 43 |
| 3.2 | Архитектура программной системы STOKES | 47 |
| 3.3 | Информационная модель программной системы STOKES | 50 |
| 3.4 | Руководство программиста программной системы STOKES | 66 |
| 3.5 | Тестирование программной системы STOKES | 70 |
|  | **Заключение** | 76 |
|  | **Библиографический список** | 77 |
|  | **Приложение А.** Листинг основных программных модулей | 78 |
|  | **Приложение Б**. Протокол расчета рассеянного изучения | 119 |
|  | **Приложение В**. Протокол расчета естественного изучения | 121 |
|  | **Приложение Г**. Протокол расчета поляризации | 123 |
|  | **Приложение Д**. Протокол расчета материала покрытия | 125 |
|  |  |  |

# Введение

Тема выпускной квалификационной работы – исследование алгоритма определения типа поверхности удаленных объектов поляриметрическими методами.

В связи с развитием космических технологий и постоянным пребыванием человека в околоземном космическом пространстве, проблема загрязнения окружающей среды становится насущной в отношении не только экосферы Земли, но и околоземного космического пространства. Данная работа посвящена разработке одного из методов решения данной проблемы – идентификации космических объектов с помощью методов фотометрии, поэтому задача разработки и исследование алгоритмов идентификации космических объектов является актуальной.

Впервые процессы рассеяния и переноса поляризованного света смоделировал математик и физик из Ирландии Джордж Габриель Стокс. В 1852 году он предложил вектор из четырёх компонент, названный вектором Стокса.

С применением интегрального уравнения Фредгольма Р. Азам, О. Френель, Э. Малюс, Д. Брюстер, Ф. Басс, Н. Башара и И. Фукс описали процессы отражения и переноса поляризованного пучка света.

Существенный вклад в реализацию разработки алгоритмов фотометрических задач был внесён отечественными учёными, такими как Т. А. Сушкевич, В. В. Коротаев, Г. Л. Башнина, Г. В Розенберг, С. А. Ухинов.

Учёные РГУ (Рязанский государственный университет) также внесли немалый вклад в разработку методов мониторинга околоземного космического пространства оптическими средствами. А. К Муртазов, А. В. Белошенков, В. И. Курышев и В. В. Куприянов впервые применили алгоритмы решения задач с параметрами рассеяния на космических околоземных объектах с использованием параметров Стокса.

Главным и основным материалом для данной выпускной квалификационной работы послужила диссертация Бодровой И.В. на тему "Математические модели и алгоритмы для программных комплексов вычисления оптических параметров покрытий космических объектов", которая посвящена изучению рассеяния светового электромагнитного излучения от поверхностей космических объектов, а также разработке алгоритмов решения обратных задач фотометрии с использованием поляризационных свойств отраженной световой волны. В работе Бодровой И.В. были реализованы алгоритмы и численный методы в виде программного комплекса для решения обратных фотометрических задач при обработке данных наблюдений за космическими объектами наземными техническими средствами.

1. Для развития темы диссертационной работы было решено исследовать алгоритмы, разработанные Бодровой И.В. и реализовать их в виде программной системы с исопльзованием свободно-распространяемого ПО, что позволит исопльзовать его в научных целях и учебном процессе.

Таким образом, **цель выпускной квалификационной работы** – разработка новой версии программной системы определения типа поверхности удаленных объектов поляриметрическими методами с использованием современных средств проектирования, которое позволит использовать разработанную систему в исследовательских и учебных целях.

Данная система позволит сократить объем «бумажной» работы, увеличить наглядность представления данных, систематизировать проведение исследований и расчетных задач.

Для достижения поставленной в работе цели необходимо решить следующие задачи:

* изучить математические модели и алгоритмы прямой и обратной задачи фотометрии;
* исследовать современные методы и алгоритмы определения типа покрытия космических объектов;
* на основе сделанного анализа спроектировать базу данных;
* разработать удобный и функциональный интерфейс для пользователя;
* реализовать заданные функции программно;
* выполнить экспериментальную проверку разработанной программной системы.

Пояснительная записка к дипломному проекту состоит из введения, трех разделов, заключения и приложений.

В первом разделе описаны математические модели и алгоритмы прямой и обратной задачи фотометрии, а также описаны модель рассеяния света от поверхности космического объекта и регистрации светового пучка.

Второй раздел содержит описание современных методов и алгоритмов определения типа покрытия космических объектов, включая алгоритмы вычисления показателя преломления однородной диэлектрической поверхностью и показателя преломления металлических покрытий.

Третий раздел содержит описание программной системы вычисления показателя преломления поверхностей космических объектов и определения материала покрытий STOKES. В разделе дается краткое описание архитектуры программной системы, ее информационной модели, руководство программиста и описание процесса тестирование программной системы STOKES.

В заключении описана успешность и полнота решения поставленных задач выпускной квалификационной работы, дается оценка перспектив развития вопросов по теме работы.

# 1 Математические модели и алгоритмы прямой и обратной задачи фотометрии

## 1.1 Постановка задачи фотометрии

Задачи фотометрии применяются в решении многих вопросов: гидродинамика, актинометрия, астрофотометрия. Среди прочих, одной из задач является мониторинга околоземного космического пространства – распознавание формы и типа поверхности и измерение геометрических параметров космических тел. Задача такого рода подразделяется на два вида.

1. Прямая задача фотометрии. Представляет собой измерение оптических характеристик изучаемого объекта.
2. Обратная задача фотометрии. По результатам измерения оптических параметров определяются геометрические параметры тела.

С использованием исходных данных изучаемого тела и средствами математического моделирования для измерения доступны следующие параметры:

1. параметры положения в пространстве (ось вращения, ориентация),
2. геометрические параметры (габаритные размеры, форма),
3. параметры структуры (компоненты тела и их взаимное расположение),
4. физические параметры поверхности (оптические характеристики, состояние поверхности).

Методы исследования с помощью фотометрии доказывают свою эффективность при обследовании поверхностей космических объектов и аппаратов. Физические свойства объекта, геометрия и габариты, химическое строение, структура и оптические параметры поверхности исследуются данным методом.

## 1.2 Модель рассеяния света от поверхности космического объекта и регистрации светового пучка

### 1.2.1. Математическая модель светового потока

На рисунке 1.1 представлена система из излучателя, объекта и приёмника в сферической системе координат.

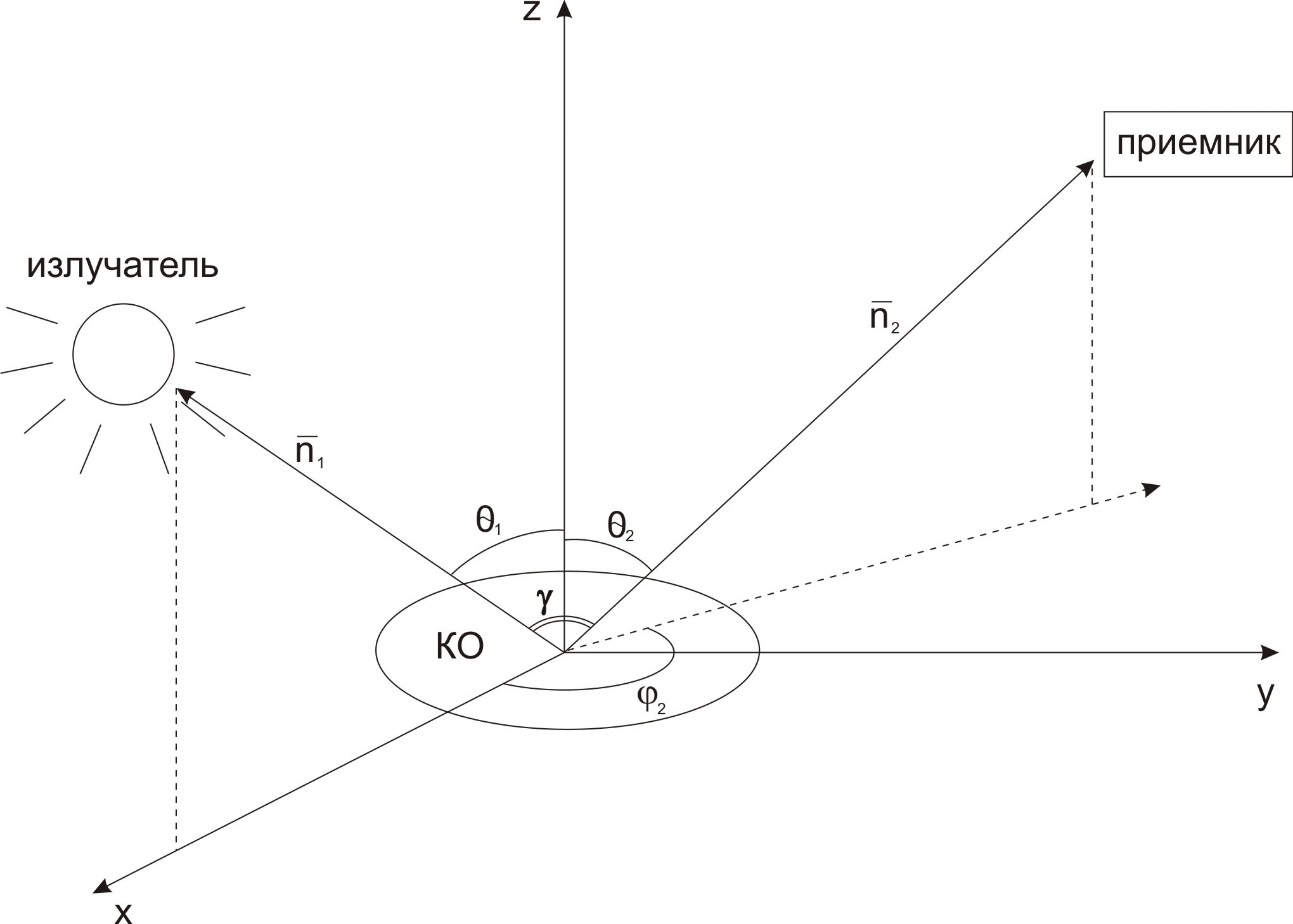


Рисунок 1.1 – Система из излучателя и приёмника в системе координат

Начало координат соответствует поверхности космического объекта, расположенного в плоскости XOY. Ось OZ соответствует нормали к плоскости XOY, то есть к поверхности объекта. Угол  соответствует углу между вектором «объект-излучатель»  и проекцией этого вектора на плоскость XOY (коллинеарно направлению оси OX). Угол  соответствует углу между проекцией вектора «объект-приёмник»  на плоскость XOY и осью OX. Углы  и  соответствуют углам между нормалью OZ и векторами  и  соответственно.  – угол между векторами  и .

Модель светового пучка в некоторой среде можно представить как электромагнитное поле с параметрами напряжённости , диэлектрического смещения , магнитной составляющей электромагнитного поля  и вектором магнитной индукции . Уравнения Максвелла позволяют связать эти параметры:



где переменными  обозначена плотность тока,  – плотность заряда по объёму.

Характер колебания вектора напряжённости определяет поляризацию светового поля. Распространение световой волны перпендикулярно направлению колебаний данного вектора, при этом вектор  колеблется гармонически.

Вектор  можно представить через проекции  и  на орты  и . Тогда направление распространения световой волны  будет перпендикулярно данным проекциям. Теперь можно рассмотреть следующие плоскости в системе координат из векторов , , :

1.  – плоскость поляризации, проходит через вектор и направление колебания вектора ;
2.  – плоскость референции, проходит через вектор  и орту .

Теперь вектор напряжённости электромагнитного поля светового потока  можно описать следующим образом:

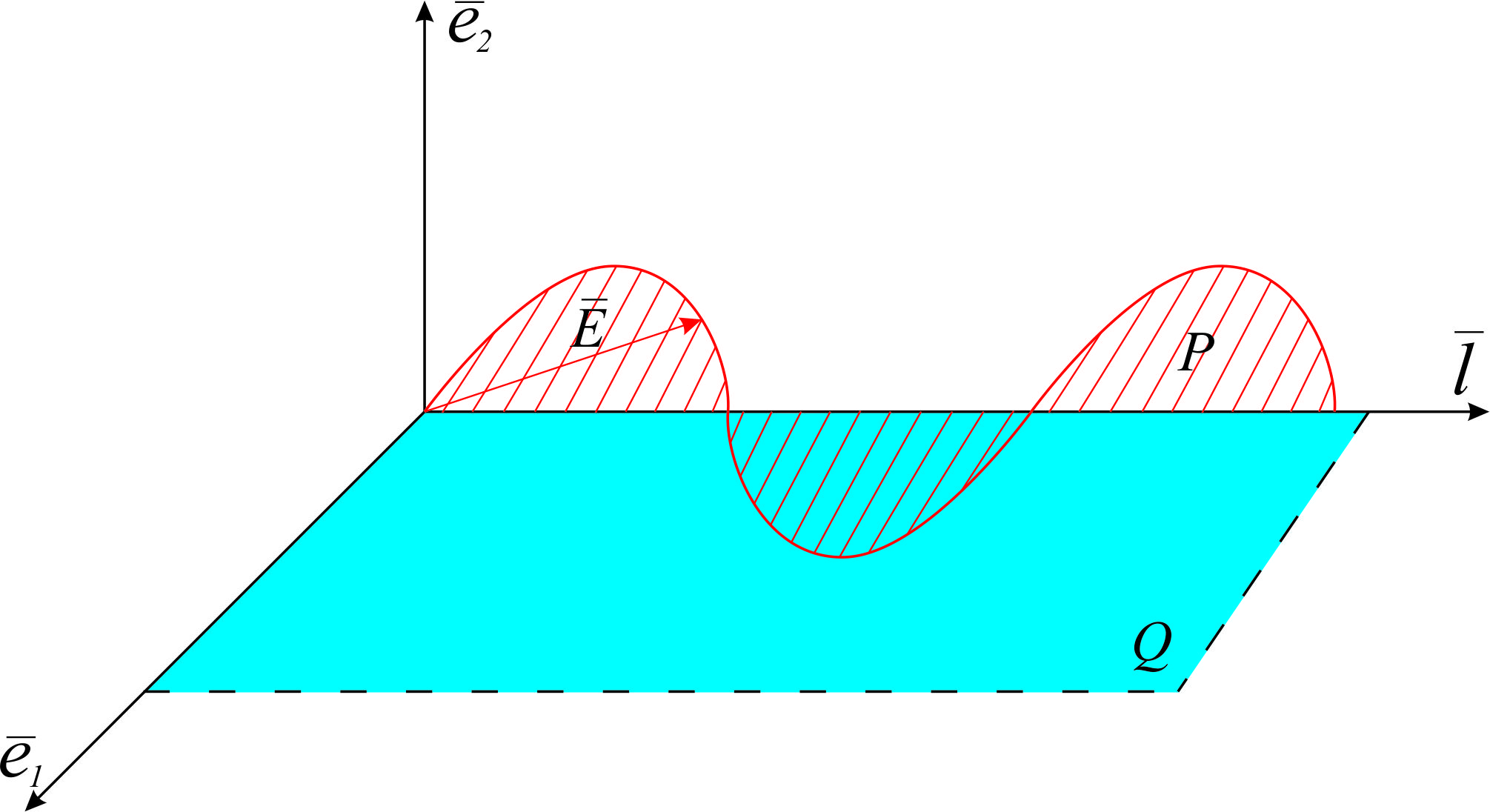


Рисунок 1.2 – Плоскости P, Q и вектор  в системе координат , , 

|  |  |
| --- | --- |
| , |  |

здесь  – амплитуды ,  – начальные фазы компонент ,  – фазовая скорость светового потока.

В зависимости от траектории вектора напряжённости светового потока вдоль направления распространения световой волны различают линейную, циркулярную или эллиптическую поляризации. Наибольший интерес представляет эллиптическая поляризация, так как остальные виды являются её частными случаями.

На рисунке 1.3 представлены параметры, характеризующие эллипс поляризации в плоскости XOY:

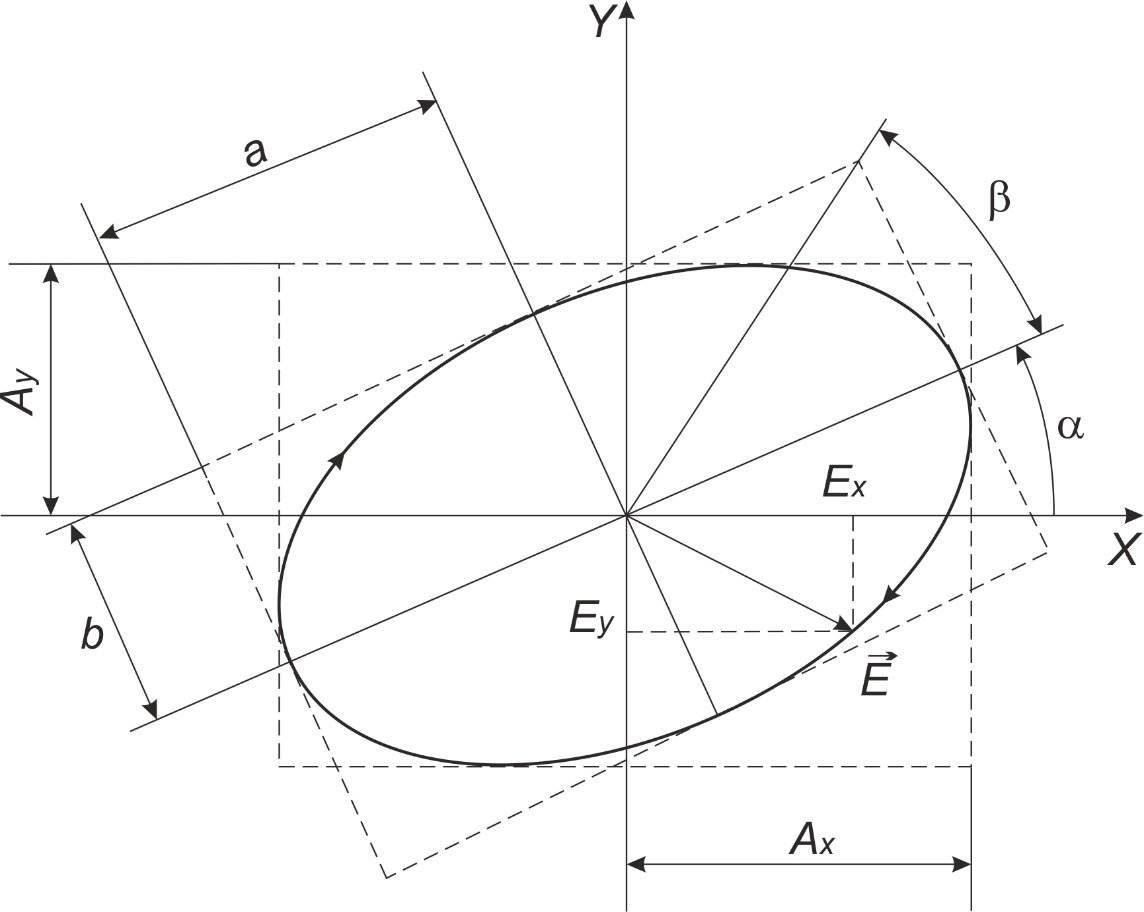


Рисунок 1.3 – Эллиптическая поляризация и её параметры

1. Эллиптичность. Выражается коэффициентом эллиптичности :. Здесь  – полуоси эллипса.  – угол эллиптичности. Параметры  и определяют знак эллиптичности, то есть направление вращения вектора напряжённости по отношению к вектору  направления светового потока.
2. Азимутальный угол , обозначает угол между большой полуосью поляризационного эллипса и направлением оси координат OX.
3. Амплитуда колебания эллипса . Переменная  обозначает общую интенсивность светового поля.
4. Параметры амплитуд  составляющих , вектора , равные описанным вокруг эллипса сторонам прямоугольника. Кроме того, имеется угол между диагональю данного прямоугольника и осью OX, выражаемый как .
5. Разность фаз составляющих  в точке отсчёта: .

Напряжённость поля монохроматического светового потока можно выразить так:



Вектор Джонса представляет собой выражение световой волны в начальном положении  и в начальный момент времени :

|  |  |
| --- | --- |
| . |  |

Данный вектор является комплексным. Компоненты пар вектора Джонса описывают поляризационное состояние волны. При этом ,  комплексно сопряжены компонентам , .

Удобнее всего состояние поляризации представить с помощью **параметров Стокса** – четырёхкомпонентного вектора, компоненты которого представляют собой линейные комбинации координат вектора Джонса (угловыми скобками обозначено усреднённое во времени значение):

|  |  |
| --- | --- |
|  |  |

Понятие вектора Стокса  в четырёхмерном пространстве было введено Дж. Г. Стоксом в 1852 году.

Смысл компонент Стокса с точки зрения физики:

* – интенсивность светового потока;
* – степень поляризации;
* – представление плоскости поляризации;
* – эллиптичность светового потока.

Параметр интенсивности  не может быть меньшим или равным нулю.

Степень поляризации  выражается через разность интенсивностей компонент  и . При этом  достигает максимума при линейной поляризации с  и минимума при .

Параметр , выражающий плоскость поляризации, представляет собой соотношение линейно-поляризованных компонент с азимутальными углами  ( > 0) или  ( < 0).

Эллиптичность светового потока  показывает соотношение левой и правой циркуляции в световой волне. Направление циркуляции определяет знак данного параметра.

Формула 1.4 описывает основные свойства составляющих вектора Стокса:

|  |  |
| --- | --- |
|  |  |

Численно степень поляризация  описывается через связь остальных компонент (формула 1.5):

|  |  |
| --- | --- |
|  |  |

Из формулы следует, что нормированный вектор Стокса выражается как , так как для неполяризованной волны .

При степени поляризации равной единице  световая волна полностью поляризована.

Связь вектора Стокса и Джонса можно выразить по формуле 1.6:

|  |  |
| --- | --- |
|  |  |

Наряду с определением направления распространения волны, параметры Стокса определяют её состояние, что делает данную вектор-функцию базовой для определения свойств поляризованного света.

Для решения фотометрических задач применяется интегральное уравнение, характеризующее рассеяние световых волн исследуемыми поверхностями:



где  – вектор Стокса падающей волны,  – вектор Стокса рассеянной волны,  – матричная яркость компоненты покрытия с индексом  (описывается с помощью матрицы Мюллера),  – единичные орты, направление которых соответствует направлениям на источник и приёмник излучения, описывает нормаль к компоненте поверхности с индексом .

### 1.2.2. Матрицы рассеяния и приборные матрицы

Для описания связи поляризованного излучения, описываемого вектором Стокса, и рассеивающей поверхности, была введена матрица Мюллера (так же именуется матрицей рассеяния, фазовой или поляризационной матрицей). Она служит функциональным оператором в теории рассеяния света. Матрица четвёртого порядка преобразует вектор Стокса падающей волны в параметр Стокса рассеянного излучения.

В результате столкновения с поверхностью вектор напряжённости  в точке с направлением  преобразуется в новый вектор , для которого задаются новые направление  и координаты . Данное преобразование можно описать следующим образом:

.

Таким образом, с помощью матрицы  можно модельно описать воздействие объекта на излучение. При этом матрица состоит из комплексных элементов. В случае многократного взаимодействия излучения с различными объектами конечный вектор будет рассчитываться как результат произведения промежуточных матриц преобразований. В случае единовременных преобразований результирующий вектор рассчитывается как сумма матриц единичных преобразований.

В случае взаимодействия излучения с веществом линейное однородное преобразование функции координат и направления при преобразовании вектора Стокса так же имеет место быть:

.

Здесь  – квадратная матрица преобразования четвёртого порядка:

,

## 1.3. Применение вектор-функции Стокса

Параметры Стокса можно выразить через функции от интенсивности *I*, угла  и эллиптичности :

|  |  |
| --- | --- |
|  | (1.12) |

Параметры ветора Стокса используются в следующих задачах.

1. Расчёт компонент вектор-функции Стокса.

Расчёт проводится с применением характеризующих поляризационный эллипс параметров (азимутальный угол, угол эллиптичности, разность фаз составляющих ), а также с использованием метода аналитического моделирования с применением выражений Джонса. С помощью новых параметров вектор-признак Стокса расширяется, что даёт возможность разработки более точных методов определения параметров покрытий.

1. Расчёт матриц оператора рассеяния.

Расчёт проводится для двух видов матриц: рассеяния от металлической и диэлектрической поверхностей. В качестве инструмента расчёта выступают выражения Френеля и Стокса.

### 1.3.1. Матрицы линейного оператора рассеяния металлическими и диэлектрическими поверхностями

Вектор Стокса может быть представлен в виде нормального восходящего телесного конуса [Бодрова]. Параметры Стокса могут быть представлены как функции азимутального угла, угла эллиптичности и разности фаз.

Для полностью поляризованного излучения:

|  |  |
| --- | --- |
|  | (1.13) |

Для излучения с неполной поляризацией:

|  |  |
| --- | --- |
|  | (1.14) |

При этом если разность фаз  вектора напряжённости , то:



Рассеяние светового потока некоторой поверхностью можно описать линейным уравнением  В диссертационной работе был проведён расчёт матрицы *M* линейного оператора рассеяния металлической поверхностью, в ходе которого для линейного преобразования была получена матрица рассеяния Мюллера:

|  |  |
| --- | --- |
| . | (1.15) |

Здесь ,  – амплитудные коэффициенты Френеля, равные отношению амплитуд падающей и отражённой световых волн (перпендикулярно и параллельно соответственно).

Собственная матрица линейного оператора рассеяния [Бодрова]:

,

и в базисе этой матрицы был представлен вектор Стокса:

.

Матрица рассеяния от диэлектрической поверхности может быть рассчитана следующим образом:

|  |  |
| --- | --- |
| , | (1.16) |

был получен собственный базис:



и через него выражен вектор Стокса:

.

### 1.3.2 Моделирование отраженного излучения и вычисление параметров Стокса отражённого излучения

Для моделирование отраженного излучения [Бодрова] используется установка, которая представляет собой реализацию схемы (рисунок 1.3): модель, излучатель и приёмник. Модель может симулировать вращение объекта вокруг двух осей с различной скоростью. В качестве излучателя выступает монохроматор УМ-2 с призмой постоянного отклонения Аббе. Объектив – ахроматический астрономического рефрактора с относительным отверстием 1:10. Источник света – стабилизированная кинолампа. В качестве приёмного устройства выступает фотоэлектрический фотометр с сопротивлением делителя напряжения 3Мом. Установка также включает люксометр для контроля постоянства излучения и лампу накаливания малой мощности в качестве контрольного источника света.

Характеристика величины светового потока и её изменение осуществляется на выходе приёмного устройства путём измерения напряжений:

,

 – напряжение, вызванное световым потоком от модельного объекта,  – фоновое напряжение,  – темновое напряжение ФЭУ-УПТ.

Относительный световой поток выражается:

,

 – напряжение от контрольного источника света.

На рисунке 1.4 представлена функциональная схема, адаптированная под наблюдения за космическим мусором околоземного космического пространства, при этом поляризационные элементы находятся между рассеивающим телом и фотоприёмником. Исходя из этого в оптической установке приборы поляризации располагались между приёмным устройством и моделью.

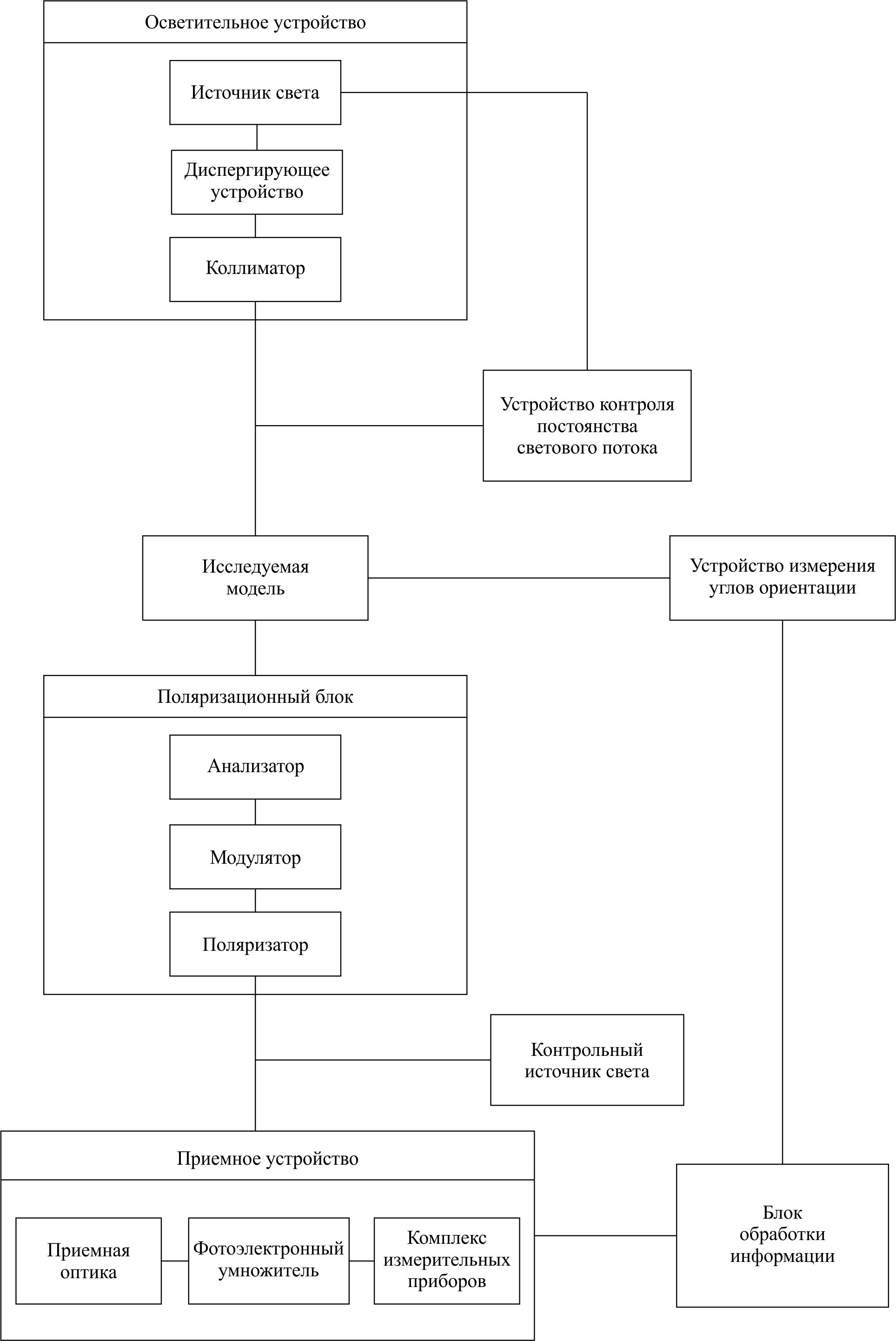


Рисунок 1.4 – Функциональная блок-схема модифицированной оптической установки

Данная установка послужила основой для разработки схемы вычисления параметров поляризации отражённого излучения.

В качестве параметров модуляции рассматриваются следующие:  – угол поворота поляризационной плоскости,  – сдвиг по фазе составляющих электрического вектора.

Путём четырёх измерений интенсивности с помощью изменения угла сдвига фаз  между гармониками  и  и поляризации на каждом измерении был рассчитан вектор Стокса.

Результат решения системы линейных уравнений – компоненты вектора Стокса отражённого потока:

,





Рисунок 1.5 – Алгоритм расчета компонент вектора Стокса

### 1.3.3. Вычисление параметров Стокса падающего излучения

Поскольку естественное солнечное излучение имеет нулевую поляризацию, вектор Стокса для такого излучения имеет параметры . Однако в связи с прохождением излучения через верхние слои атмосферы, световой поток поляризуется. Если известны характеристики космического объекта (в том числе и фотометрические), то можно охарактеризовать данный объект с помощью матрицы рассеяния. Математически рассеяние моделируется с помощью операторного уравнения вида 

В случае наличия данных о показателях преломления поверхности, угле падения луча и составе материала, можно найти вектор Стокса и поляризацию падающего луча.

В соответствии с операторным выражением координаты вектора Стокса выражаются уравнениями:

,

,

,

.

Рисунок 2.2 иллюстрирует поле рассеяния параметра Q отражённого излучения.

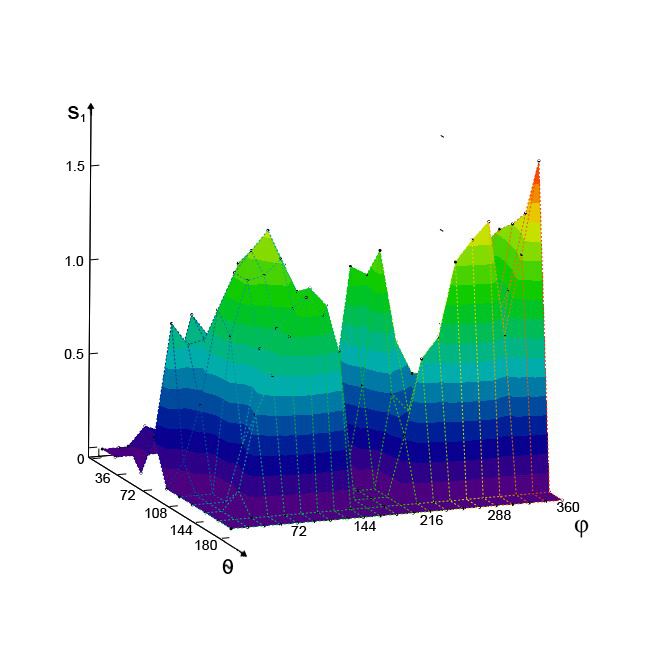


Рисунок 1.6 – Поле рассеяния параметра Q отражённого света.



Рисунок 1.7– Алгоритм расчета компонент вектора Стокса   
естественного излучения

## 1.4. Алгоритмы нахождения дополнительных параметров поляризации рассеянного светового потока

Рассмотрим алгоритм нахождения дополнительных параметров поляризации рассеянного светового потока:

1. при модуляции  (разность начальных фаз компонент () используются значения фазового сдвига , близкие к  и находящиеся в одной четверти, амплитуда модуляции при этом не более ;
2. находятся значения  и путём решения системы уравнений:

|  |  |
| --- | --- |
|  | (1.17) |

1. уточняются знаки значений  и  в зависимости от знаков координат Стокса *Q*, *U*, *V и значений фазового сдвига;*
2. в случае модуляции фазового сдвига значения  в интервале  имеет место погрешность определения угла .

Ищутся особые состояния световой волны, при которых азимутальный угол равен нулю. Под особыми состояниями также понимаются особые точки поляризационного эллипса:

* особая точка:  = 0 при *U=0* ,
* особая точка:  = 0 при *U=0* ,
* особая точка: .



Рисунок 1.8 – Алгоритм расчета дополнительных параметров   
численными методами

# 2 СОВРЕМЕННЫЕ МЕТОДЫ И АЛГОРИТМЫ ОПРЕДЕЛЕНИЯ ТИПА ПОКРЫТИЯ КОСМИЧЕСКИХ ОБЪЕКТОВ

## 2.1 Постановка задачи

Рассморим алгоритмы определения параметров поверхностей космических тел, в том числе и объекты техногенного происхождения. Под параметрами поверхностей понимаются показатели преломления, состояние материала, тип покрытия.

Для вычисления параметров отражающая поверхность классифицируется как принадлежащая к одному из типов: металлическому или диэлектрическому.

Процесс решения задачи идентификации типа поверхности состоит из следующих этапов:

* генерация признаков для характеристики исследуемого объекта;
* отбор признаков, способных охарактеризовать тип покрытия;
* отбор классификатора;
* оценка классификатора;
* определение типа исследуемой поверхности.

## 2.2 Определение типа покрытия космических объектов

*Первым этапом* определения типа покрытия является генерация и селекция характеризующих признаков.

При *N* числе измерений фотометрических параметров определяются параметры поляризации:

,

где  – координаты вектора Стокса,

 – коэффициент поляризации,

 – азимутальный угол эллипса поляризации,

 – угол эллиптичности эллипса поляризации,

 – комплексный параметр поляризации.

Признаки для классификации диэлектрического покрытия могут быть вычислены следующим образом:

1. при постоянном полярном угле  –

|  |  |
| --- | --- |
| . | (2.1) |

1. при изменяющемся угле  и при  или  –

|  |  |
| --- | --- |
| . | (2.2) |

1. при изменяющемся угле  и при  или  –

|  |  |
| --- | --- |
|  | (2.3) |

*Вторым этапом* определения типа поверхности является отбор классификатора. Определено 2 класса: диэлектрический – , и металлический – . Принадлежность признаков классу определяется через нормальное распределение с выборочными средними значениями , , , и выборочными дисперсиями . Гипотеза о нормальном распределении признаков сводится к критерию Пирсона .

Отыскивается доверительный интервал для генеральной дисперсии:

: ,

где  вычисляются как:

, 

при уровне значимости .

Таким образом, обозначено 2 гипотезы:

1.  (нулевая гипотеза) – признаки нормально распределённые, разница несущественна, признаки относится к классу диэлектрических поверхностей;
2.  (альтернативная гипотеза) – распределение признаков не относится к нормальному или относится с дисперсией выше допустимых значений, признак относится к классу металлических поверхностей.

Расчёт максимально допустимой дисперсии для каждого признака, результатом которого является условие классификации объекта классом :

|  |  |
| --- | --- |
| . | (2.4) |

*Третьим этапом* определения типа поверхности является оценка ошибки классификатора:





Здесь  – известная дисперсия признака *x*,  – предполагаемая дисперсия признака *x*,  – известное математическое ожидание.

*Четвёртым этапом* является построение разделяющей поверхности решения и непосредственно определение типа поверхности.

Координаты Стокса *J*, *Q*, *U*, *V* найдены путём четырёх измерений интенсивности излучения с изменением угла наклона , сдвигом фаз между компонентами и  и линейной поляризации на каждом измерении.

Исходя из полученных параметров Стокса осуществляется азимутальный угол  и угол эллиптичности  отражённого излучения в соответствии с алгоритмом расчёта дополнительных параметров поляризации.

С помощью полученных значений вычислены значения признаков , , . Результаты были представлены в виде таблиц для каждого из признаков.

1. 2.1. Предобработка признака №1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| , град. | Параметры выборки | | | |
|  |  |  |  |
|  | 1.85 | -0.07 | 0.0049 | -1.892 |
|  | 1.89 | -0.03 | 0.009 | -0.967 |
|  | 1.95 | 0.03 | 0.009 | 0.967 |
|  | 1.92 | 0 | 0 | 0 |
|  | 1.77 | -0.15 | 0.0225 | -4.05 |
|  | 1.96 | 0.04 | 0.0016 | 1.081 |
|  | 1.99 | 0.07 | 0.0049 | 1.89 |
|  | 5.5 | - | - | - |
|  | 1.85 | -0.07 | 0.0049 | -1.89 |
|  | 1.92 | 0 | 0 | 0 |
|  | 2.12 | 0.20 | 0.04 | 5.4 |
|  | 1.88 | -0.04 | 0.0016 | -1.081 |
|  | -0.2 | - | - | - |
|  | 1.69 | -0.23 | 0.0529 | -6.22 |
|  | 2.15 | 0.23 | 0.0529 | 6.22 |
|  | 1.65 | -0.27 | 0.0729 | -7.2 |
|  | 2.25 | 0.33 | 0.1089 | 8.92 |
|  | 6.5 | - | - | - |

Отброшены значения 5.5; 6.5; -0.2. Из остальных .

Среднее выборочное , выборочная дисперсия , выборочное СКО  Выборка соответствует нормальному распределению по критерию Пирсона:  при .

1. 2.2. Предобработка признака №2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| , град. | Параметры выборки | | | |
|  |  |  |  |
|  | 2.52 | -1.24 | 1.53 | 1.19 |
|  | 2.80 | -0.96 | 0.92 | 0.72 |
|  | 3.25 | -0.51 | 0.26 | 0.20 |
|  | 3.75 | -0.01 | 0.001 | 0.0008 |
|  | 4.5 | 0.74 | 0.548 | 0.43 |
|  | 3.35 | -0.41 | 0.168 | 0.13 |
|  | 3.86 | 0.1 | 0.001 | 0.0008 |
|  | 4.54 | 1.78 | 3.17 | 2.48 |
|  | 5.23 | 1.47 | 2.16 | 2.47 |
|  | 2.66 | -1.1 | 1.21 | 0.95 |
|  | 4.5 | 0.74 | 0.548 | 0.43 |
|  | 5.28 | 1.52 | 2.31 | 1.8 |
|  | 2.53 | -1.23 | 1.51 | 1.18 |
|  | 12.5 | - | - | - |
|  | 2.55 | -1.21 | 1.46 | 1.16 |
|  | 5.21 | 1.45 | 2.1 | 1.64 |
|  | 11.5 | - | - | - |
|  | 20.5 | - | - | - |
|  | 10.6 | - | - | - |
|  | - | - | - | - |

Отброшены значения 12.5; 11.5; 20.5; 10.6. Из остальных . Среднее выборочное , выборочная дисперсия, выборочное СКО  Выборка соответствует нормальному распределению по критерию Пирсона:  при .

1. 2.3. Предобработка признака №3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| , град. | Параметры выборки | | | |
|  |  |  |  |
|  | 0.56 | -1.24 | 1.53 | -3.44 |
|  | 1.43 | -0.37 | 0.14 | -1.02 |
|  | 1.55 | -0.25 | 0.06 | -0.69 |
|  | 1.92 | 0.12 | 0.014 | 0.33 |
|  | 2.41 | 0.61 | 0.37 | 1.69 |
|  | 2.87 | 1.07 | 1.14 | 2.97 |
|  | 0.78 | -1.02 | 1.04 | -2.83 |
|  | 1.45 | -0.35 | 0.12 | -0.97 |
|  | 1.56 | -0.24 | 0.06 | -0.66 |
|  | 1.78 | -0.02 | 0.004 | -0.055 |
|  | 2.45 | 0.65 | 0.42 | 1.80 |
|  | 2.34 | 0.54 | 0.29 | 1.5 |
|  | 1.56 | -0.24 | 0.58 | -0.66 |
|  | 1.87 | 0.07 | 0.0049 | 0.19 |
|  | 1.86 | 0.06 | 0.0036 | 0.17 |
|  | 1.89 | 0.08 | 0.0064 | 0.22 |
|  | 1.67 | -0.13 | 0.017 | -0.36 |
|  | 2.5 | 0.7 | 0.49 | 1.94 |
|  | 10.6 | - | - | - |
|  | - | - | - | - |

Отброшены значения 10.6; 0.56. Из остальных 

Среднее выборочное , выборочная дисперсия, выборочное СКО  Выборка соответствует нормальному распределению по критерию Пирсона:  при .

## 2.3. Классификация типа покрытия

Классификация производится в соответствии с алгоритмом, приведённом в предыдущем параграфе. Чтобы классифицировать поверхность классом диэлектрических покрытий , необходимо выполнить следующие условия.

1. Для каждого из признаков выборка должна обладать нормальным распределением с выборочной дисперсией не больше единицы.
2. Найденные путём оценки максимально допустимой дисперсии вероятности принадлежности поверхности к классу  должны быть больше 0.8.
3. Определение вероятности классификации изучаемой поверхности соответствует формуле (2.4).

## 2.4. Алгоритм вычисления показателя преломления однородной диэлектрической поверхностью

В [Бодрова] описано алгоритм для расчёта параметра преломления для металлических и диэлектрических поверхностей. Вид материала покрытия определяется исходя из полученного параметра преломления.

Для разных типов поверхностей параметр преломления рассчитывается следующим образом:

*  – показатель преломления металлических покрытий, является комплексным,  – показатель преломления,  – показатель поглощения;
*  – показатель преломления диэлектрических покрытий, является вещественным.

Модельный параметр  представляет собой функцию зависимости от показателя преломления материала  и угла падения светового пучка :

|  |  |
| --- | --- |
|  | (2.6) |

Показатель преломления рассчитывается исходя из знака подмодульного выражения (2.6).

Если подмодульное выражение отрицательно, то:

|  |  |
| --- | --- |
| . | (2.7) |

Если же положительно, то:

|  |  |
| --- | --- |
| . | (2.8) |

Параметр *λ* выражается через коэффициент поляризации:

|  |  |
| --- | --- |
| . | (2.9) |

Представленный алгоритм сводится к следующим этапам:

1. расчёт значения параметра  во всех точках измерений интенсивности отражённого излучения в соответствии с выражением (2.9);
2. расчёт значений показателя преломления  во всех точках измерений в соответствии с выражениями (2.7) и (2.8);
3. поиск математического ожидания ;
4. в соответствии с таблицей 2.4 определить материал поверхности;
5. вычислить абсолютную и относительную погрешности расчёта показателя преломления.



Рисунок 2.1– Алгоритм определения коэффициента преломления   
неметаллической поверхности

## 2.5. Алгоритм вычисления показателя преломления металлических покрытий

Показатель преломления металлических поверхностей представляет собой комплексное число вида , модельный параметр *λ* выражается как:

|  |  |
| --- | --- |
| . | (2.10) |

Рассчитывается параметр *a*:



и параметр *b*:



Для отрицательного подмодульного значения выражения (2.11) параметр *a* может быть выражен по формуле:

|  |  |
| --- | --- |
| . | (2.12) |

Для положительного же соответствует выражение:

|  |  |
| --- | --- |
| . | (2.13) |

Для нахождения показателя преломления для металлических покрытий:

|  |  |
| --- | --- |
| . | (2.15) |

Алгоритм вычисления показателя преломления металлических покрытий сводится к следующим этапам:

1. расчёт значений параметра  во всех *k* точках измерений интенсивности отражённого излучения в соответствии с выражением (2.9);
2. для всех точек измерений  находится параметр  в соответствии с выражениями (2.12) и (2.13) в зависимости от знака подмодульного выражения в уравнении (2.11);
3. вычисление *k / 2* пар значений *u* и *v путём решения* систем линейных алгебраических уравнений в соответствии с выражением (2.14);
4. вычисление математического ожидания *k / 2* значений параметров *u* и *v*;
5. расчёт показателя преломления для металлических покрытий в соответствии с выражением (2.15);
6. согласно таблице 2.4 отыскивается материал, соответствующий показателю преломления исследуемой поверхности;
7. расчёт абсолютной погрешности измерений.



Рисунок 2.2 – Алгоритм определения коэффициента преломления металлических покрытий (часть 1)



Рисунок 2.2 – Алгоритм определения коэффициента преломления металлических покрытий (часть 2)

# 3 Программная система вычисления показателя преломления поверхностей космических объектов и определения материала покрытий STOKES

## 3.1 Обоснование выбора средств разработки программной системы

### 3.1.1 Обзор Python

Python (далее «Пайтон») представляет собой объектно-ориентированный интерпретируемый язык программирования. Будучи языком общего назначения, Пайтон включает пакет библиотек для различных целей: выполнение математических операций (*math, numpy*), построение графиков (*matplotlib*), работа с базами данных (*sqlite3*), работа с языком HTML (*html*), работа с сетевыми протоколами (*socket*) и др. Существует и множество сторонних библиотек, расширяющих функционал языка: создание GUI (*wx*), создание анимации и игр (*pygame*), разработка веб-приложений (*webapp2*) и др.

Основными чертами языка Пайтон являются следующие.

1. Динамическая типизация.

При такой типизации переменная привязывается к типу данных не во время её объявления (что в принципе отсутствует в языке как таковое), а во время связывания переменной с некоторым значением. Данная особенность, во-первых, избавляет от необходимости объявлять переменную и тип присваиваемых данных перед включением в код, а во-вторых позволяет одну и ту же переменную в рамках одной программы связывать со значениями различных типов.

2. Автоматическое управление памятью (сборщик мусора).

В отличие от других языков программирования, таких как *C/C++* и *Pascal*, в Пайтоне реализован так называемый «сборщик мусора». Переменные автоматически удаляются из памяти, как только они перестают участвовать в выполнении программы.

3. Интроспекция.

Возможность получения типа и структуры данных объекта в процессе выполнения программы. Под объектом подразумеваются в том числе и любые переменные, привязанные к значению конкретного типа, поскольку типы данных на языке Пайтон также представляют собой классы объектов, а данные – представители этих классов. Например, целое число *5* является объектом класса *int* с соответствующей структурой (методами и атрибутами). Один из способов реализации интроспекции – функции *type()* и *dir()*.

4. Реализация обработки исключений.

Данная особенность позволяет предписывать поведение программы в случае возникновения ошибок (исключений). Осуществляется с помощью операторов *try – except – else – finally*.

5. Поддержка многопоточных вычислений.

Позволяет при необходимости разбивать ход выполнения программы на несколько потоков, обрабатываемых одновременно или в заранее описанном порядке. Инструменты реализации предоставляются, например, библиотекой *threading*.

6. Структуры данных высокого уровня.

Будучи высокоуровневым языком программирования, Пайтон реализует высокоуровневые структуры данных – абстракции, описание которых на низкоуровневых языках или машинном коде требует больших усилий. В Пайтоне примерами таких структур являются *списки* и *словари*.

7. Перегрузка операторов.

В отличие от таких популярных языков программирования, как *Java* и *JavaScript*, Пайтон поддерживает перегрузку операторов. Любой встроенный функционал (операторы арифметических операций, возвращаемые встроенными функциями значения и т.п.) может быть описан заново в соответствии с задачей. Например, при описании некоторого класса можно задать метод *\_\_str\_\_*, отвечающий за значение, возвращаемое функцией *print()* в стандартный поток вывода при взятии объекта данного класса в качестве аргумента.

8. Синтаксические особенности.

Отличительной чертой Пайтона, заметной с первого знакомства с языком, являются синтаксические особенности. Отсутствуют, например, операторные и фигурные скобки, определяющие блоки кода; вместо них используются отступы.

Как и многие другие языки программирования, Пайтон поддерживает инкапсуляцию, полиморфизм, методы управления жизненным циклом объекта, метапрограммирование и многое другое.

Как объектно-ориентированный язык программирования, Пайтон поддерживает механизм одиночного и множественного наследования. При описании классов можно задавать так называемые родительские классы. В таком случае объект «дочернего» класса кроме собственных атрибутов и методов также «наследует» атрибуты и методы «родительских» классов.

Одним из главных недостатков языка Пайтон, как и других интерпретируемых языков, является низкое в сравнении с компилируемыми языками быстродействие. Компилятор обрабатывает весь код сразу и переводит его в машинный код, интерпретатор же делает это «на ходу», что, с одной стороны, даёт такие преимущества, как динамическая типизация, а с другой, – значительно (до 100 – 200 порядков по сравнению с компилируемыми языками) уменьшает скорость выполнения программы.

Другим недостатком, в отличие от таких языков, как *Ruby*, является отсутствие возможности модификации или изменения встроенных классов: *object*, *int*, *list*. Но и у такого подхода есть свои положительные стороны – уменьшение ресурсоёмкости программ.

### 3.1.2 Среда разработки Visual Studio Code

Visual Studio Code — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации.

Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом.

### 3.1.3 База данных sqlite

SQLite – это встроенная библиотека, которая реализует автономный, безсерверный, нулевой конфигурации, транзакционный механизм СУБД SQL. Это база данных, которая настроена на нуль, что означает, как и другие базы данных, которые вам не нужно настраивать в вашей системе.

Особенности данной базы данных:

* SQLite не требует отдельного процесса сервера или системы для работы (без сервера).
* SQLite поставляется с нулевой конфигурацией, что означает отсутствие необходимости в настройке или администрировании.
* Полная база данных SQLite хранится в одном кросс-платформенном диске.
* SQLite очень маленький и легкий, менее 400KiB полностью сконфигурированный или менее 250KiB с дополнительными функциями, опущенными.
* SQLite является автономным, что означает отсутствие внешних зависимостей.
* SQLite-транзакции полностью совместимы с ACID, обеспечивая безопасный доступ к нескольким процессам или потокам.
* SQLite поддерживает большинство функций языка запросов, найденных в стандарте SQL92 (SQL2).
* SQLite написан на ANSI-C и предоставляет простой в использовании API.
* SQLite доступен в UNIX (Linux, Mac OS-X, Android, iOS) и Windows (Win32, WinCE, WinRT).

Python получил модуль sqlite3 в версии 2.5, что значит, что можно создавать базу данных SQLite в любой версии Python, без необходимости скачивания дополнительных инструментов.

*3.1.4 Система контроля версий git*

*Git* – децентрализованная система управления версиями, то есть система, сохраняющая изменения в одном или нескольких файлах так, чтобы потом можно было восстановить определённые старые версии.

Задачи, решаемые *git*:

* вернуть файлы к прежнему виду;
* вернуть к прежнему состоянию весь проект;
* сравнить изменения с какого-то времени;
* увидеть, кто последним изменял модуль, который дал сбой, кто создал проблему;
* восстановить проект в любом состоянии.

Все файлы проекта хранились с использованием git на репозитории в gtihub.com.

## 3.2 Архитектура программной системы STOKES

Программа для расчета показателя преломления, а также определения типа материала покрытия.



1. 3.1 — Блок-схема комплекса программ расчета поляризационных параметров и оптических характеристик покрытий

Программная система состоит из следующих модулей.

Программа расчета параметров Стокса и коэффициента поляризации отраженного светового потока.

Программа для вычисления азимутального угла, угла эллиптичности и комплексного показателя поляризации светового потока.

Программа для расчета показателя преломления и определения материала покрытия.

Управляющая программа.

Программы функционирует в любой операционной системе, удовлетворяющей следующим требованиям:

При написании программ использовались следующие средства разработки:

* среда: *Visual Studio Code*;
* язык программирования: *python 3*;
* формат базы данных: *sqlite3*;
* построение графиков: *matplotlib*.

Разработанная программная система реализован в виде набора следующих программных модулей:

* модуль расчета параметров вектора Стокса;
* модуль работы с базой данных комплекса;
* модуль реализации алгоритма градиентного спуска;
* модуль вычисления нормированной функции Лапласа;
* модуль отображения графиков и диаграмм;
* модуль импорта данных;
* модуль расчета естественного излучения;
* модуль расчета рассеянного излучения;
* модуль расчета поляризации и определения типа покрытия;
* главный управляющий модуль.

Структура основных действий, выполняемых программой (структура программы), представлена на рисунке 3.2



1. 3.2 — Связь программных модулей программной системы для вычисления поляризационных характеристик светового потока и расчета оптических параметров покрытий

## 3.3 Разработка схемы данных программной системы STOKES

### 3.3.1 Спецификация сущностей

На основе проведённого анализа предметной области выделим её основные объекты и сформируем соответствующие им сущности.

С целью наиболее наглядного и правильного обозначение наименований сущностей и соответствующих им атрибутов необходимо выполнение следующих правил:

* Идентификаторы сущностей, атрибутов и ключей должны отражать их семантику (проблемный смысл).
* Следует помнить, что эти идентификаторы выражаются комбинациями существительных, иногда в сочетании с прилагательными. Глаголы в этих именах не используются.
* Идентификатор может представлять собой слитную последовательность слов или их сокращений. Каждое слово (сокращение) в этой последовательности должно начинаться с большой буквы.
* Описания сущностей должны быть представлены таблицами и схемами.

Спецификация полученных сущностей и атрибутов представлена в таблице 3.1.

Таблица 3.1 — Описание сущностей

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Название сущности** | **Определение сущности** |
| 1 | Параметры расчета | Глобальные параметры, задаваемые при расчетах |
| 2 | Интервалы | Интервалы параметров расчета для контроля того, что значение параметра не выходит за границы интервала |
| 3 | Коэффициенты материалов | Показатели отражения для металлических и диэлектрических поверхностей |
| 4 | Результаты расчета задачи 1, 2 | Хранение результатов расчета задачи 1 и 2 |
| 5 | Результаты расчета задачи 3 | Хранение результатов расчета задачи 3 |
| 6 | Данные признака C1 | Входные данные для задачи 3 по признаку 1 |
| 7 | Данные признака C2 | Входные данные для задачи 3 по признаку 2 |
| 8 | Данные признака C3 | Входные данные для задачи 3 по признаку 3 |

Рассмотрим полученные сущности более подробно (таблица 3.2.).

Сущность «Параметры расчета» имеет атрибуты:

* код параметра;
* наименование параметра;
* значение параметра.

Сущность «Интервалы» имеет атрибуты:

* имя параметра;
* минимальное значение;
* максимальное значение.

Сущность «Коэффициенты материалов» имеет атрибуты:

* наименование материала;
* минимальное и максимальное значение действительной части показателя;
* минимальное и максимальное значение мнимой части показателя.

Сущность «Результаты расчета задачи 1, 2» имеет атрибуты:

* углы α и β;
* значения интенсивностей τ, φ, I для четырех измерений;
* значения параметров вектора Стокса рассеянного излучения;
* значения параметров вектора Стокса естественного излучения;
* значение параметров угла α1 и β1;
* значение параметра χ.

Сущность «Результаты расчета задачи 3» имеет атрибуты:

* номер признака;
* порядковые номер параметра;
* обозначение и код параметра;
* значение параметра.

Сущность «Данные признака C1» имеет атрибуты:

* значение угла θ;
* значения параметров Q, U, V;
* вычисляемые атрибуты.

Сущность «Данные признака C2» имеет атрибуты:

* значение угла γ;
* значения параметров Q, U, V;
* вычисляемые атрибуты.

Сущность «Данные признака C3» имеет атрибуты:

* значение параметров угла α1 и β1;
* значения параметра С3;
* вычисляемые атрибуты.

Таблица 3.2 — Спецификация сущностей. Ключи и атрибуты

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Название и обозначение сущности | Ключ сущности  и его обозначение | Атрибуты сущности  и их обозначение |
| 1 | Параметры расчета (Пар) | Код параметра (Код) | Наименование параметра (НаимПар)  Значение параметра (ЗначПар) |
| 2 | Интервалы (Инт) | Имя параметра (ИмяПар) | Минимальное значение (Мин);  Максимальное значение (Макс) |
| 3 | Коэффициенты материалов (Коэф) | Наименование материала (НаимМат) | Минимальное и максимальное значение действительной части показателя  (ДействМин, ДействМакс)  Минимальное и максимальное значение мнимой части показателя (МнимМин, МнимМакс) |

Продолжение таблицы 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Название и обозначение сущности | Ключ сущности  и его обозначение | Атрибуты сущности  и их обозначение |
| 4 | Результаты расчета задачи 1, 2 (Расч12) | Угол α (Alfa)  Угол β (Betta) | значения интенсивностей τ, φ, I для четырех измерений (Tau1..Tau4, Phi1...Phi4, I1...I4);  значения параметров вектора Стокса рассеянного излучения (J, Q, U, V, P);  значения параметров вектора Стокса естественного излучения (J0, Q0, U0, V0, P0);  значение параметров угла α1 и β1 (Alfa1, Beta1);  значение параметра χ (Hi) |
| 5 | Результаты расчета задачи 3 (Расч3) | номер признака (НомПр);  код параметра (КодПар) | порядковый номер параметра  (НомПарам);  обозначение параметра (НаимПарам)  значение параметра  (Знач Парам). |
| 6 | Данные признака C1 (ПрС1) | значение угла θ (Tetta) | значения параметра Q  значения параметра U  значения параметра V; |

Окончание таблицы 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Название и обозначение сущности | Ключ сущности  и его обозначение | Атрибуты сущности  и их обозначение |
| 7 | Данные признака C2  (ПрС2) | значение угла γ (Gamma); | значения параметра Q  значения параметра U  значения параметра V; |
| 8 | Данные признака C3  (ПрС3) | значение параметра угла α1 (Alfa1)  значение параметра и β1 (Beta1) | значения параметра С3 |

Схемы сущностей представляются в традиционной нотации, ключи в схеме выделяются подчеркиванием (таблица 3.3.).

Таблица 3.3. Схемы сущностей

|  |  |  |
| --- | --- | --- |
| № п/п | Название и обозначение сущности | Схема сущности |
| 1 | Параметры расчета (Пар) | **Пар** (Код, НаимПар, ЗначПар) |
| 2 | Интервалы (Инт) | **Инт** (ИмяПар, Мин, Макс) |
| 3 | Коэффициенты материалов (Коэф) | **Коэф** (НаимМат, МинДейств, МаксДейств, МинМним, МаксМним) |
| 4 | Результаты расчета задачи 1, 2 (Расч12) | **Расч12** (Alfa, Betta, Tau1..Tau4, Phi1...Phi4, I1...I4, J, Q, U, V, P, J0, Q0, U0, V0, P0, Alfa1, Beta1, HiДейств, HiМним) |
| 5 | Результаты расчета задачи 3 (Расч3) | **Расч3** (НомПр, КодПар, НомПарам, НаимПарам, ЗначПарам) |

Окончание таблицы 3.3

|  |  |  |
| --- | --- | --- |
| № п/п | Название и обозначение сущности | Схема сущности |
| 6 | Данные признака C1 (ПрС1) | **ПрС1** (Tetta, Q, U, V) |
| 7 | Данные признака C2  (ПрС2) | **ПрС2** (Gamma, Q, U, V) |
| 8 | Данные признака C3  (ПрС3) | **ПрС3** (Alfa1, Beta1, C3) |

### 3.3.3 Спецификация связей

Спецификация связей представляет собой два перечня – список связей и список их идентификаторов. В разработанной программной системе связи между сущностями отсутствуют.

### 3.3.4 Формализация зависимостей

Отношение состоит из двух частей — заголовка отношения и тела отношения. Заголовок отношения — это аналог заголовка таблицы. Заголовок отношения состоит из атрибутов. Количество атрибутов называется степенью отношения. Тело отношения — это аналог тела таблицы. Тело отношения состоит из кортежей. Кортеж отношения является аналогом строки таблицы. Количество кортежей отношения называется мощностью отношения.

Отношения обладают следующими свойствами.

* В отношении нет одинаковых кортежей.
* Кортежи не упорядочены (сверху вниз).
* Атрибуты не упорядочены (слева направо).
* Все значения атрибутов атомарные.

При преобразовании ER-модели в реляционную модель данных использовались следующие правила:

* степень связи 1:1, класс принадлежности обязат:обязат => количество таблиц 1, первичный ключ (Л или П);
* степень связи 1:1, класс принадлежности необязат:обязат => количество таблиц 2, первичный ключ (Л, Л или П);
* степень связи 1:1, класс принадлежности необязат:необязат => количество таблиц 3, первичный ключ (Л, П, Л или П);
* степень связи 1:М, класс принадлежности обязат:обязат => количество таблиц 2, первичный ключ (Л или П);
* степень связи 1:М, класс принадлежности обязат(необязат):обязат => количество таблиц 3, первичный ключ (Л, П, П);
* степень связи М:М, класс принадлежности обязат(необязат): обязат(необязат) => количество таблиц 2, первичный ключ (Л, Л, ЛП);
* м-связей, класс принадлежности обязат(необязат): обязат(необязат) => количество таблиц М+1, первичный ключ (К1, К2, …, КМ, К) [6].

На основании анализа сущностей составим таблицу с предварительных описанием отношений по вышеприведённым правилам (таблицы 3.5).

Таблица 3.5 — Описание предварительных отношений на основе сущностей

|  |  |  |
| --- | --- | --- |
| № п/п | Название и обозначение сущности | Отношение |
| 1 | Параметры расчета (Пар) | **Пар** (Код, НаимПар, ЗначПар) |
| 2 | Интервалы (Инт) | **Инт** (ИмяПар, Мин, Макс) |
| 3 | Коэффициенты материалов (Коэф) | **Коэф** (НаимМат, МинДейств, МаксДейств, МинМним, МаксМним) |
| 4 | Результаты расчета задачи 1, 2 (Расч12) | **Расч12** (Alfa, Betta, Tau1..Tau4, Phi1...Phi4, I1...I4, J, Q, U, V, P, J0, Q0, U0, V0, P0, Alfa1, Beta1, HiДейств, HiМним) |
| 5 | Результаты расчета задачи 3 (Расч3) | **Расч3** (НомПр, КодПар, НомПарам, НаимПарам, ЗначПарам) |
| 6 | Данные признака C1 (ПрС1) | **ПрС1** (Tetta, Q, U, V) |
|  | Данные признака C2  (ПрС2) | **ПрС2** (Gamma, Q, U, V) |
|  | Данные признака C3  (ПрС3) | **ПрС3** (Alfa1, Beta1, C3) |

На основе описания отношений (таблица 3.5, 3.6) определим первичные, внешние и потенциальные ключи отношений. Полученные результаты представим в виде таблицы (таблица 3.7.).

Таблица 3.7. Ключи отношений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Отношение | Первичный ключ | Внешний ключ | Потенциаль-ный ключ |
| 1 | **Пар** (Код, НаимПар, ЗначПар) | Код | - | НаимПар |
| 2 | **Инт** (ИмяПар, Мин, Макс) | ИмяПар | - | - |
| 3 | **Коэф** (НаимМат, МинДейств, МаксДейств, МинМним, МаксМним) | НаимМат | - | - |
| 4 | **Расч12** (Alfa, Betta, Tau1..Tau4, Phi1...Phi4, I1...I4, J, Q, U, V, P, J0, Q0, U0, V0, P0, Alfa1, Beta1, HiДейств, HiМним) | Alfa, Bett | - | - |
| 5 | **Расч3** (НомПр, КодПар, НомПарам, НаимПарам, ЗначПарам) | НомПр, КодПар | - | - |
| 6 | **ПрС1** (Tetta, Q, U, V) | Tetta | - | - |
| 7 | **ПрС2** (Gamma, Q, U, V) | Gamma | - | - |
| 8 | **ПрС3** (Alfa1, Beta1, C3) | Alfa1, Beta1 | - | - |

Все получившиеся отношения не являются подмножественными, не имеют дублирования, следовательно, отношения составлены правильно и являются итоговыми.

### 3.3.5 Определение таблиц базы данных

Далее каждое отношение трансформируется в таблицу (таблица 3.8.). Имена отношений становятся именами таблиц, а имена атрибутов – именами колонок.

Таблица 3.8. Отношения и таблицы

|  |  |  |
| --- | --- | --- |
| № п/п | Отношение | Таблица |
| 1 | **Пар** (Код, НаимПар, ЗначПар) | Constant |
| 2 | **Инт** (ИмяПар, Мин, Макс) | Intervals |
| 3 | **Коэф** (НаимМат, МинДейств, МаксДейств, МинМним, МаксМним) | MaterialRefraction |
| 4 | **Расч12** (Alfa, Betta, Tau1..Tau4, Phi1...Phi4, I1...I4, J, Q, U, V, P, J0, Q0, U0, V0, P0, Alfa1, Beta1, HiДейств, HiМним) | Calculation |
| 5 | **Расч3** (НомПр, КодПар, НомПарам, НаимПарам, ЗначПарам) | Calculation\_Task3 |
| 6 | **ПрС1** (Tetta, Q, U, V) | C1 |
| 7 | **ПрС2** (Gamma, Q, U, V) | C2 |
| 8 | **ПрС3** (Alfa1, Beta1, C3) | C3 |

Более подробная информация о свойствах полученных в таблице 3.8. таблиц представлена в таблице 3.9.

Таблица 3.9. Описание свойств и полей таблиц

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя таблицы** | **Имя**  **атрибута** | **Тип** | **Ключевое поле** | **Обязатель- ное поле** |
| Constant | Code | text | да | да |
| Name | text | нет | да |
| Value | real | нет | да |
| Intervals | FieldName | text | да | да |
| FieldMin | real | нет | да |
| FieldMax | real | нет | да |
| MaterialRefraction | MaterialName | text | да | да |
| ReValueMin | real | нет | да |
| ReValueMax | real | нет | да |
| ImValueMin | real | нет | да |
| ImValueMax | real | нет | да |
| Calculation | Alfa | real | да | да |
| Beta | real | да | да |
| Tau1 | real | нет | да |
| Phi1 | real | нет | да |
| I1 | real | нет | да |
| Tau2 | real | нет | да |
| Phi2 | real | нет | да |
| I2 | real | нет | да |
| Tau3 | real | нет | да |
| Phi3 | real | нет | да |
| I3 | real | нет | да |
| Tau4 | real | нет | да |
| Phi4 | real | нет | да |
| I4 | real | нет | да |
| J | real | нет | да |
| Q | real | нет | да |
| U | real | нет | да |
| V | real | нет | да |
| P | real | нет | да |
| J0 | real | нет | да |
| Q0 | real | нет | да |
| U0 | real | нет | да |
| V0 | real | нет | да |

Окончание таблицы 3.9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя таблицы** | **Имя**  **атрибута** | **Тип** | **Ключевое поле** | **Обязатель- ное поле** |
|  | P0 | real | нет | да |
| Alfa1 | real | нет | да |
| Beta1 | real | нет | да |
| Hi | real | нет | да |
| Calculation\_Task3 | Sign | real | да | да |
| Number | real | нет | да |
| ParameterName | real | нет | да |
| ParameterValue | real | нет | да |
| C1 | Tetta | real | да | да |
| Q | real | нет | да |
| U | real | нет | да |
| V | real | нет | да |
| C2 | Gamma | real | да | да |
| Q | real | нет | да |
| U | real | нет | да |
| V | real | нет | да |
| C3 | Gamma | real | да | да |
| Alfa | real | нет | да |
| Beta | real | нет | да |

### 3.3.6 Схема баз данных

*Схемой реляционной базы* данных называется набор заголовков отношений, входящих в базу данных [11].

Схема призвана визуализировать (наглядно представить) состав отношений (таблиц) и взаимосвязей между ними. Каждое отношение изображается в виде прямоугольника, поделенного на части. Каждая часть символизирует атрибут и содержит его наименование. Стрелки показывают ссылки дочерних отношений на родительские [6].

В схеме данных устанавливаются параметры обеспечения целостности связей в базе данных [6]. Схема базы данных приведена на рисунке 3.3.

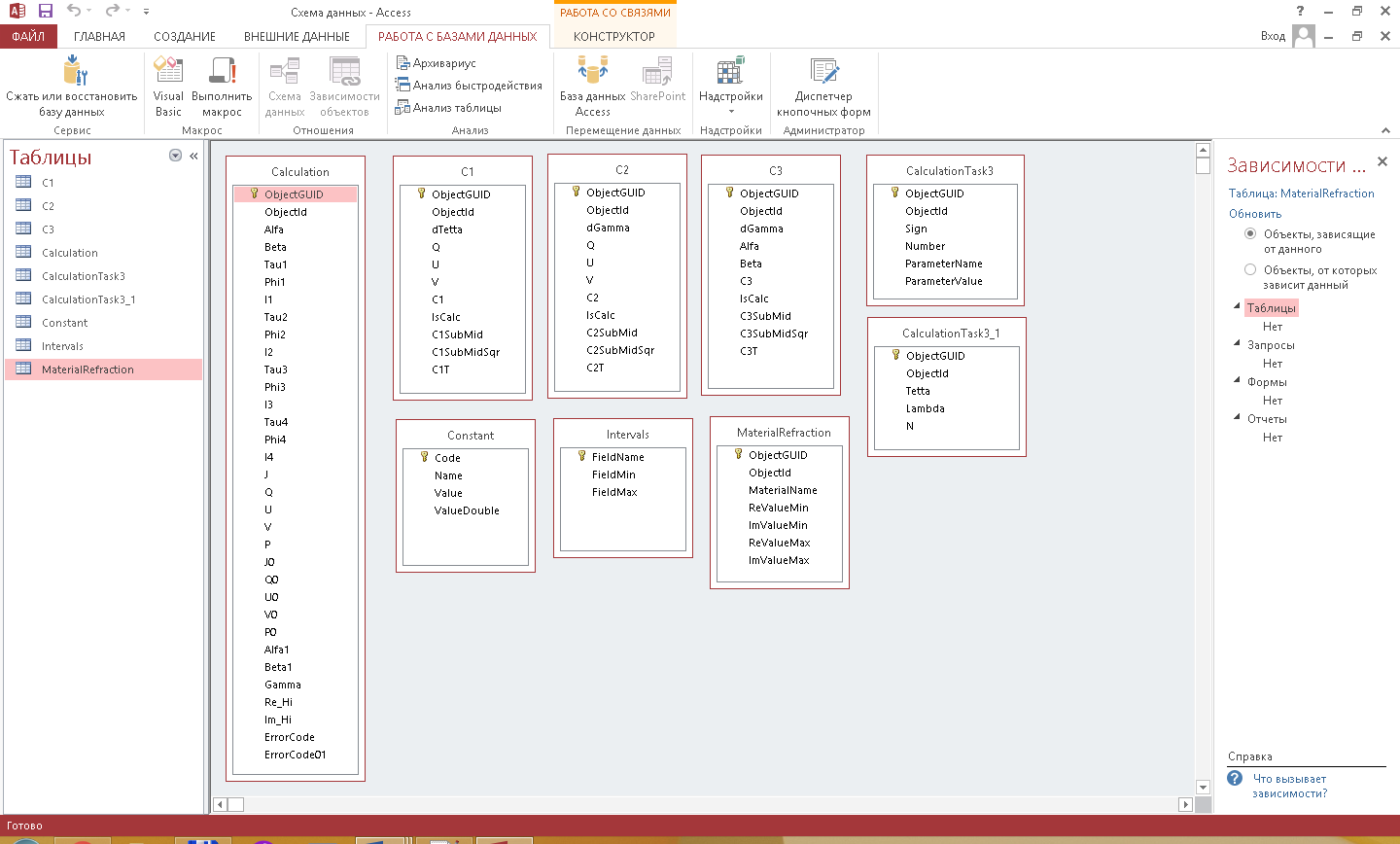


Рисунок 3.3 — Схема базы данных в Microsoft Access

### 3.3.7 Нормализация баз данных

С целью построения реляционной базы без дублирования данных, с обеспечением возможности поддержания целостности при внесении изменений докажем нормальность полученных отношений и в случае необходимости приведём их к четвёртой нормальной форме с помощью алгоритма нормализации.

*Нормализация* — разбиение одной таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных [P].

*Алгоритм нормализации состоит из следующих этапов*

1. Привести отношение в ***1НФ*** (первая нормальная форма), нужно исключить повторяющиеся группы и многозначные поля (декомпозиция или выравнивание).
2. Чтобы привести отношение в ***2НФ*** (вторая нормальная форма), нужно разбить отношения на проекции для исключения приводимых функциональных зависимостей (от части ключа). Отметим, что поле *В* таблицы функционально зависит от поля *А* той же таблицы в том и только в том случае, когда в любой заданный момент времени для каждого из различных значений поля *А* обязательно существует только одно из различных значений поля *В*. Отметим, что здесь допускается, что поля *А* и *В* могут быть составными.
3. Чтобы привести отношение в ***3НФ***, нужно получившееся отношение разбить на проекции для исключения транзитивных функциональных зависимостей. Функциональная зависимость *R.X  R.Y* называется транзитивной, если существует такой атрибут *Z*, что имеются функциональные зависимости *R.X  R.Z* и *R.Z  R.Y* и отсутствует - *R.Z → R.X*.
4. Чтобы привести отношение в ***НФБК*** (нормальная форма Бойса-Кодда), нужно разбить на проекции для исключения любых функциональных зависимостей, в которых детерминант не является ключом. Детерминант – любой атрибут, от которого полностью функционально зависит некоторый другой атрибут [6].
5. Чтобы привести отношение в ***4НФ*** (четвёртая нормальная форма), нужно разбить его на проекции для исключения многозначных зависимостей. При этом атрибуты (множества атрибутов) *Y* и *Z* многозначно зависятот *Х* (*Х→→Y|Z*), тогда и только тогда, когда из того, что в отношении *R* содержатся кортежи *r1 = (x,y,z1)* и *r2 = (x,y1,z)* следует, что в отношении *R* содержится также и кортеж к *r3 = (x,y,z)*
6. Чтобы привести отношение в ***5НФ*** (пятая нормальная форма), нужно разбить на проекции для исключения любых зависимостей соединения, которые не подразумеваются потенциальными ключами. Имеют место зависимости специального вида, когда отношение не может быть подвергнуто декомпозиции без потерь на две проекции, но может быть декомпозировано на большее число проекций. Такие зависимости называются зависимостями соединения и являются обобщением понятия многозначной зависимости. На практике приводить отношения к 5НФ необязательно.

*Проверка полученных итоговых отношений (таблица 3.6) на принадлежность к 4НФ с помощью алгоритма нормализации показала следующие результаты:*

1. Все приведённые отношения находятся в 1НФ, так как не содержат повторяющиеся группы и многозначные поля, то есть удовлетворяют следующим свойствам:

* в отношении нет одинаковых картежей;
* картежи не упорядочены;
* атрибуты не упорядочены и различаются по наименованию;
* все значения атрибутов атомарные.

2. Все приведённые отношения находятся в 2НФ, так как находятся в 1НФ, и не содержит неключевых атрибутов, зависящих от части сложного ключа. Отметим, что если первичный ключ отношения является простым, то отношение автоматически находится в 2НФ.

3. Все приведённые отношения находятся в 3НФ, так как находятся в 2НФ и все их неключевые атрибуты взаимонезависимы. Атрибуты называются взаимонезависимыми, если не один из них не является функционально зависимым от другого.

4. Все приведённые отношения находятся в НФБК, так как детерминанты всех функциональных зависимостей являются потенциальными ключами. Если отношение находится в НФБК, то оно автоматически находится в 3НФ. Если отношение имеет два не только первичный, но и потенциальный ключ, и других атрибутов в нём нет, то для анализа отношения можно ограничиться анализом 3НФ и не прибегать к НФБК.

5. Все приведённые отношения находятся в 4НФ, так как находятся в НФБК и не содержит нетривиальных многозначных зависимостей.

Таким образом, все итоговые отношения, полученные в результате тщательно проведённого анализа, находятся в 4НФ и обеспечивают адекватность предметной области, целостность информации, высокую скорость выполнения процедуры обновления данных, а следовательно и гибкую структуру хранимых данных.

## 3.4 Руководство программиста программной системы STOKES

### 3.4.1 Назначение и условия применения программной системы

#### 3.4.1.1 Назначение программной системы

Программная система предназначена для решения следующих задач:

* расчет параметров Стокса и коэффициента поляризации отраженного светового потока.
* вычисление азимутального угла, угла эллиптичности и комплексного показателя поляризации светового потока;
* расчет показателя преломления и определения материала покрытия.

В состав системы входят:

* модуль расчета параметров вектора Стокса;
* модуль работы с базой данных комплекса;
* модуль реализации алгоритма градиентного спуска;
* модуль вычисления нормированной функции Лапласа;
* модуль отображения графиков и диаграмм;
* модуль импорта данных;
* модуль расчета естественного излучения;
* модуль расчета рассеянного излучения;
* модуль расчета поляризации и определения типа покрытия;
* главный управляющий модуль..

#### 3.4.1.2 Условия применения комплекса

Для функционирования комплекса необходима ПЭВМ со следующими характеристиками:

* процессор с тактовой частотой не менее 2 ГГц;
* оперативная память не менее 1 Гбайт;
* дисковая подсистема: суммарный объем свободного пространства не менее 1 Гб;
* монитор не менее 15".

На ПЭВМ должно быть установлено следующее общее программное обеспечение:

* операционная система – Windows или на основе ядра GNU/Linux;
* python 3;

### 3.4.2 Характеристики программной системы

Программная система обеспечивает выполнение задачи определение материала покрытия. Программная система реализована в виде отдельного приложения Python с интерфейсом пользователя.

Программная система работает в автомаическом режиме выполнения оператором необходимых действий посредством работы с использованием командной строки.

Программная система содержит все необходимые функции по обработке информации, необходимой для решения фотометрических задач, ее обработки, создания и экспорта отчетов.

### 3.4.3 Настройка программной системы

Для настройки программной системы необходимо выполнить следующие действия:

* открыть файл basic.ini или создать его копию с произвольным именем и расширением ini;
* в секции [*Common*] ввести значения следующих параметров для настройки работы алгоритмов:
  + Fi – угол падения;
  + Nju\_Re – действительная часть показателя преломления;
  + Nju\_Im – комплексная часть показателя преломления;
  + Gamma – угол.
* в секции [*Intervals*] ввести значения следующих параметров для настройки работы алгоритмов:
  + U – допустимый интервал значений параметра U;
  + V – допустимый интервал значений параметра V;
  + U0 – допустимый интервал значений параметра U0;
  + V0 – допустимый интервал значений параметра V0;
  + J – допустимый интервал значений параметра J;
  + P – допустимый интервал значений параметра P;
  + Q – допустимый интервал значений параметра Q;
  + J0 – допустимый интервал значений параметра J0;
  + P0 – допустимый интервал значений параметра P0;
  + Q0 – допустимый интервал значений параметра Q0.

### 3.4.4 Обращение к программной системе

Обращение к программной комплексу «ПОВК» происходит через интерфейс командной строки (терминал). Следует перейти в папку /stokes запуск осуществляется командой “python main.py [опции] ”.

Программа запускакетя в окне командного процессора, при этом интерфейс пользователя не отображается: все параметры.

Программе могут быть переданы следующие параметры:

* '-c', '--config'
  + назначение: конфигурационный файл;
  + значение по-умолчанию 'config/basic.ini',
* '-t', '--task'
  + назначение: номер задачи (1, 2 или 3)
  + значение по-умолчанию '1',
* '-f', '--file'
  + назначение: файл для загрузки данных;
  + значение по-умолчанию '/data/csv/calculation.csv',
* '-m', '--material'
  + назначение:файл с показателями преломления матералов;
  + значение по-умолчанию '/data/csv/MaterialRefraction.csv',

### 3.4.5 Входные и выходные данные

При выполнении функций входными данными являются:

* параметры настройки;
* допустимые интервалы значений параметров;
* входные данные для расчета вектора Стокса;
* входные данные для расчета признаков C1-C3;
* показатели преломления материалов.

При выполнении функций выходными данными являются:

* .результаты расчета векторов Стокса естественного излучения;
* результаты расчета векторов Стокса рассеянного излучения;
* результаты расчета параметров поляризации;
* результаты расчета критериев χ2;
* расчет вероятности попадания объекта в класс;
* результат расчета материала покрытия.

Результаты расчета сохраняются в виде текстовых протоколов расчета и в виде значений в соответствующие таблицы базы данных.

## 3.5 Тестирование программной системы STOKES

Для исследования алгоритмов определения типа покрытия был разработан следующий план тестирования программной системы Stokes:

* тестирование корректности импорта данных;
* тестирование расчета параметров вектора Стокса;
* тестирование расчета параметров рассеянного излучения;
* тестирование расчета параметров поляризации;
* тестирование расчета критериев;
* тестирование расчета определение материалов покрытия
* тестирование модуля вывода графической информации.

Для тестирования корректности импорта данных были протестированы модули импорта конфигурации и модули загрузки данных в базу данных (рисунк 3.4).

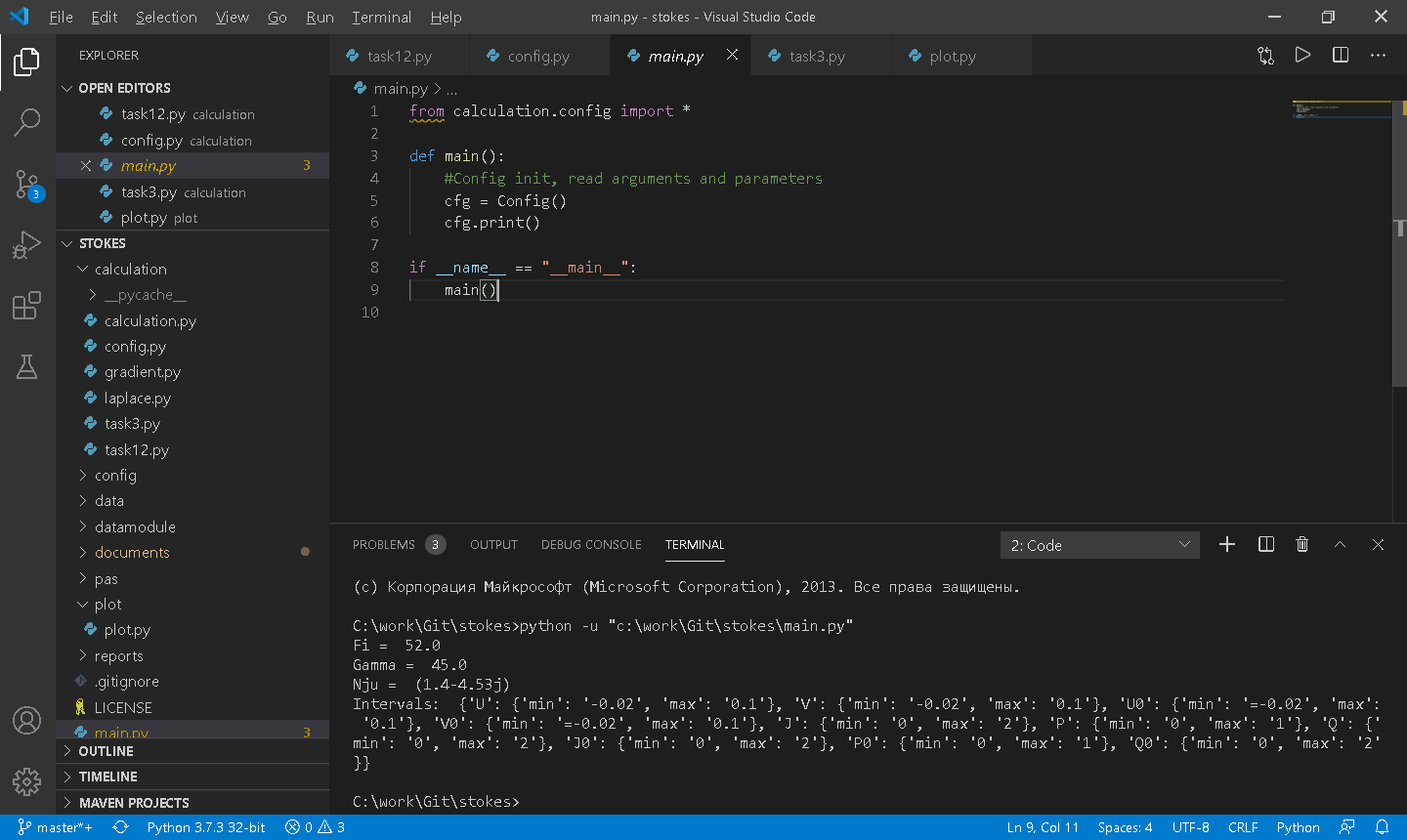


Рисунок 3.4 – Тестирование импорта параметров конфигурации

В качестве исходных данных для тестирования расчета параметров вектора Стокса и расчета параметров рассеянного излучения были использованы данные из [Бодрова]. В [Бодрова] был проведён эксперимент с целью апробации алгоритмов и расчёта преломления для моделей с диэлектрическими поверхностями и заранее известным показателем преломления. Модель – объект с диэлектрическим покрытием, сферической формой и  Инструментом эксперимента выступала оптическая установка, описанная в 1 главе. Осветительный прибор в ходе эксперимента менял угол падения светового пучка в диапазоне от 0° до 90°. Фотоприёмник совместно с поляризационным блоком регистрировал показатели интенсивности излучения. Эти данные в формате csv являлись входными данными для тестовых расчетов.

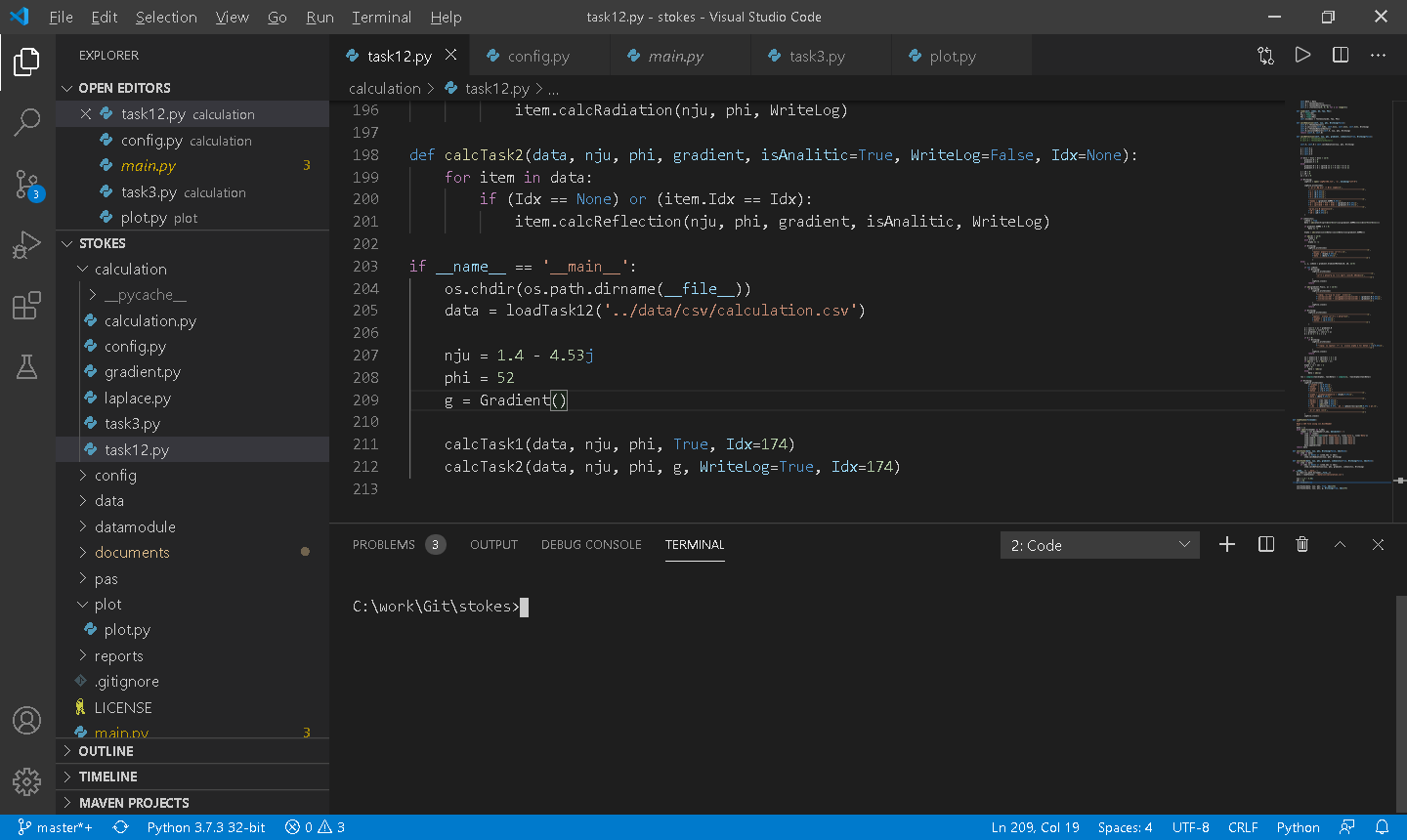


Рисунок 3.5 – Тестирование расчета (задачи 1 и 2)

В резульатате запуска данных задач были получене протоколы расчета (рисунок 3.6, рисунок 3.7, Приложение).

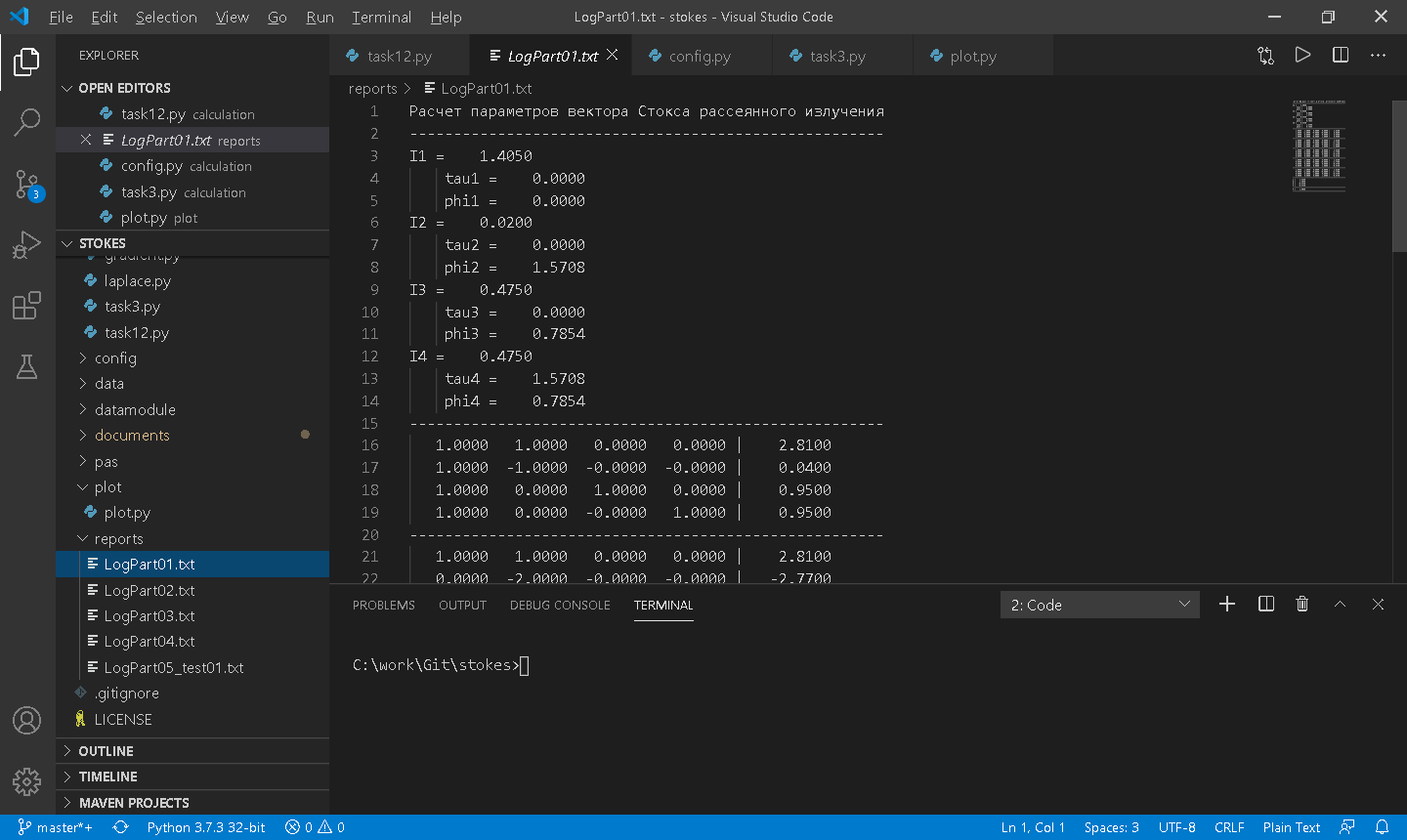


Рисунок 3.6 – Протокол расчета (задача 1)

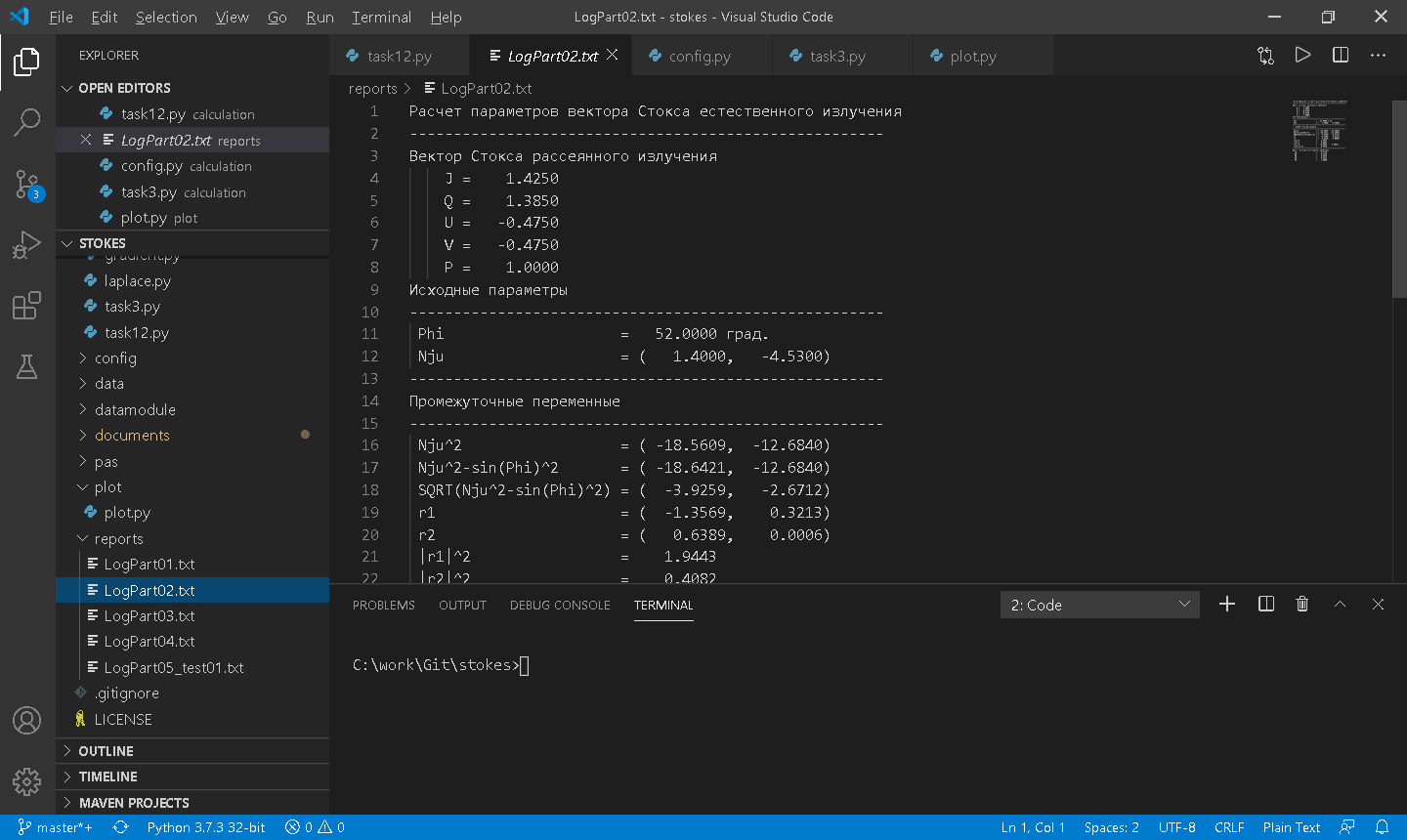


Рисунок 3.7 – Протокол расчета (задача 2)

В резульатате запуска задачи 3 были получене протоколы расчета (рисунок 3.8, рисунок 3.9, Приложение).

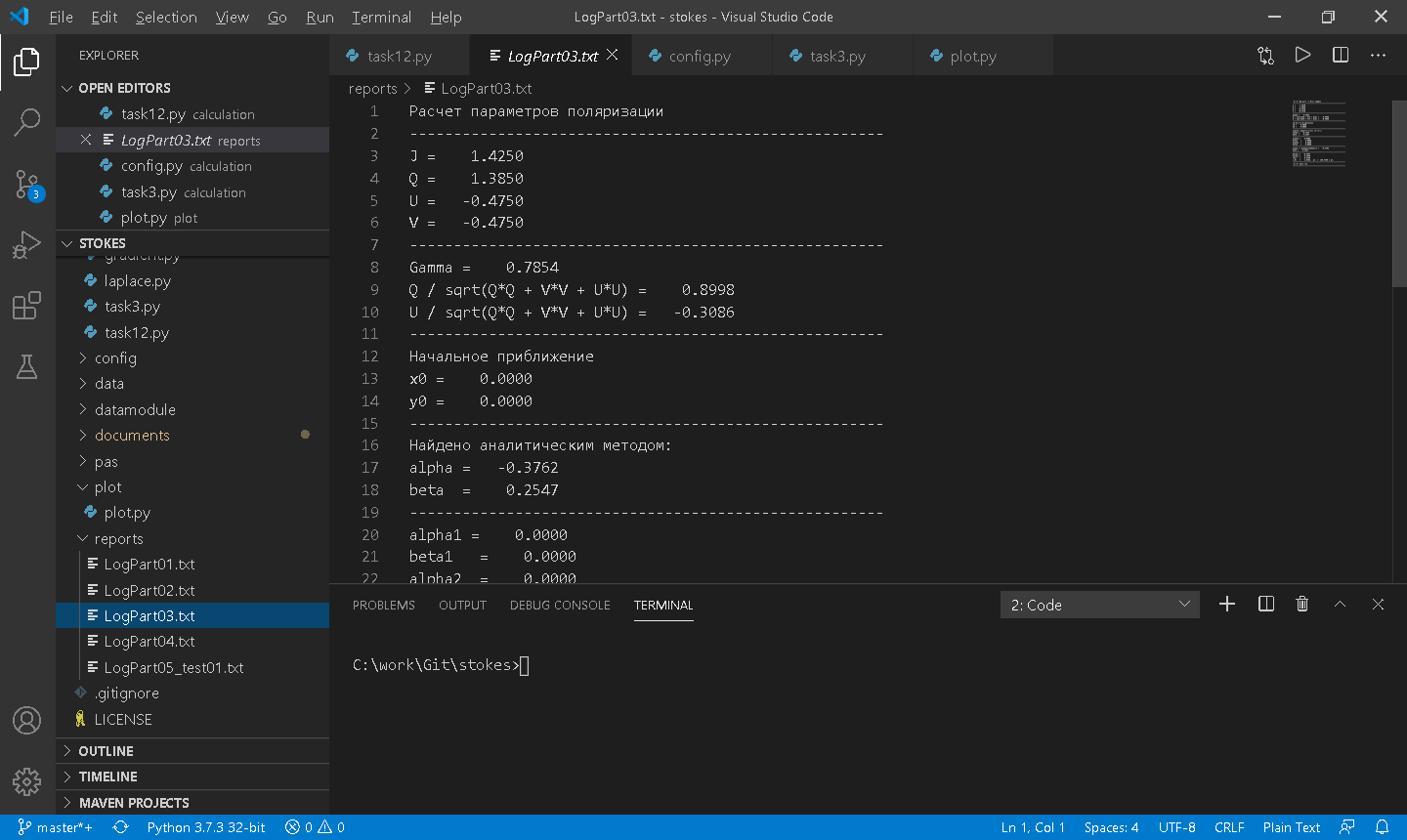


Рисунок 3.8 – Протокол расчета 1 (задача 3)

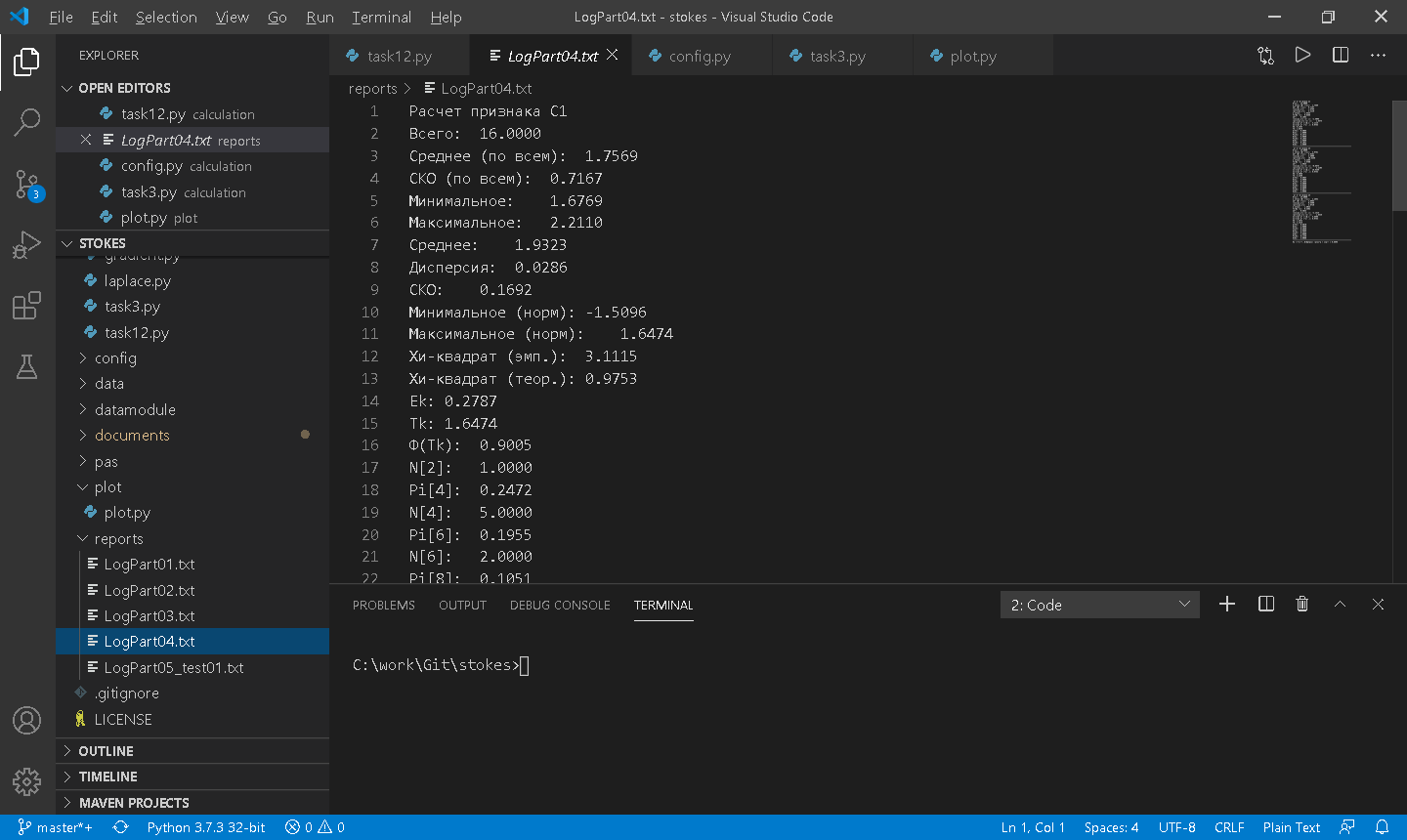


Рисунок 3.9 – Протокол расчета 2 (задача 3)

Для тестирования определения материала покрытия были использованы данны из таблицы 3.4 и получены протоколы расчета (рисунок 3.10).

1. 3.4. Показатели преломления основных материалов объектов космического мусора в видимой обласми (λ = 0.5 мкм*) при* t = 20°

|  |  |  |
| --- | --- | --- |
| Тип космического мусора | Материал покрытия | Показатель преломления |
| Корпус | Алюминий  Сплавы алюминия  Сплавы титана  Нержавеющая сталь  Стеклоткань | 0.5-*i*4.59  (0.5-0.8)-*i*(4.2-4.6)  (2.1-2.2)-*i*(2.8-2.9)  1.46-*i*3.17  10.89 |
| Двигатели | Нержавеющая сталь  Сплавы титана  Алюминий | 1.46-*i*3.17  (0.5-0.8)-*i*(4.2-4.6)  0.5-*i*4.59 |
| Солнечные батареи | Кремниевое стекло  Двуокись кремния  Кварц | 1.5-1.8  1.46  1.54-1.55 |
| Антенны | Молибденовая проволока с золотым покрытием  Алюминий  Сплав алюминия со сталью  Серебро  Вольфрам | 0.9-*i*2.5  0.5-*i*4.59  0.95-*i*3.8  0.11-*i*2.94  3.31-*i*0.89 |
| Телескопы | Сульфид свинца  Алюминий | 1.7-*i*3.30  0.5-*i*4.59 |
| Отражающие поверхности | Серебро  Никель  Золото  Ниобий  Молибден  Платина | 0.11-*i*2.94  1.54-*i*3.1  0.5-*i*2.04  2.13-*i*3.07  3.15-*i*3.72  1.76-*i*3.59 |

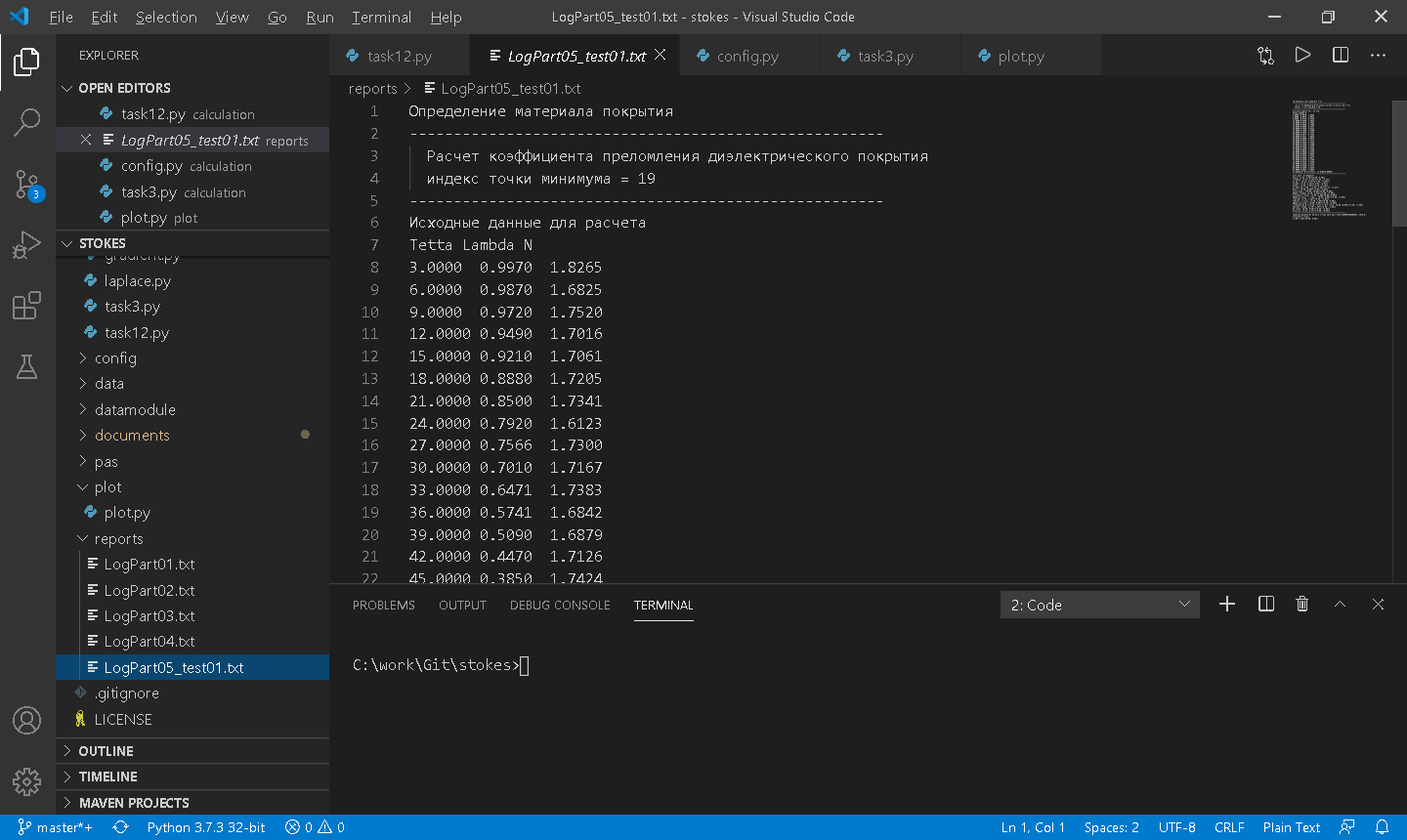


Рисунок 3.10 – Протокол расчета 3 (задача 3)

Тестирование программной системы показало правильность разработанных алгоритмов, так как результаты программных расчетов совпали с теоретическими расчетами.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы были поставлены и решены следующие задачи:

* ислледованы существующие модели и алгоритмы определения типа покрытий и расчета оптических параметров покрытий космических объектов;
* реализован и протестирован алгоритм для расчета поляризационных характеристик отраженной световой волны;
* реализован и протестирован алгоритм определения типа покрытий и расчета оптических параметров покрытий космических объектов на основе вероятностного метода распознавания образов, методов аналитического и статистического моделирования;
* разработана программная система Stokes на основе свободно-распространяемого ПО для решения поставленных задач.

Разработанная система программной системы тестирования позволила произвести полную проверку проекта с выявлением и исправлением всех скрытых дефектов, возникающих в процессе длительной эксплуатации.

Таким образом, следует считать, что задачи проекта полностью выполнены, цель достигнута.

Вместе с тем, можно указать направления дальнейшего продолжения работы и развития использованных в ней идей:

* разработка графического интерфеса пользователя;
* разработка модуля в виде jupiter notebook для интерактивного решения исследовательских задач;
* апробация разработанной программныой системы на новых наборах выходных данных;
* внедрение разработанной системы в учебный процесс РГРТУ.

# Библиографический список

|  |
| --- |
|  |
| Приложение А. Листинг основных программных модулейА.1 Модуль работы с конфигурацией config.py import configparser  import argparse  import textwrap  class Config:  def \_\_init\_\_(self):  parser = argparse.ArgumentParser(description='Stokes parameters calculation',  formatter\_class=argparse.RawDescriptionHelpFormatter,  epilog=textwrap.dedent('''\  Keys used:  q or ESC = Exit  '''))  parser.add\_argument('-c', '--config',  dest='cfg\_file',  default='config/basic.ini',  help='configuration file')  parser.add\_argument('-t', '--task',  dest='task',  default='1',  help='task to start (1, 2 or 3)')  parser.add\_argument('-f', '--file',  dest='file',  default='/data/csv/calculation.csv',  help='file to load')  parser.add\_argument('-m', '--material',  dest='material',  default='/data/csv/MaterialRefraction.csv',  help='material file to load')  args = parser.parse\_args()  self.config = configparser.ConfigParser()  self.config.read(args.cfg\_file)  sys = self.config['Common']  self.fi = float(sys.get("Fi", 52)) # Угол падения, град  re = float(sys.get("Nju\_Re", 1.4)) # Действительная часть показателя преломления  im = float(sys.get("Nju\_Im", -4.53)) #Мнимая часть показателя преломления  self.nju = complex(re, im) #показатель преломления  self.gamma = float(sys.get("Gamma", 45)) # Угол, град  ivl = self.config['Intervals']  # Интервалы для контроля параметров  self.intervals = {}  self.intervals["U"] = self.parse\_interval(ivl.get("U"))  self.intervals["V"] = self.parse\_interval(ivl.get("V"))  self.intervals["U0"] = self.parse\_interval(ivl.get("U0"))  self.intervals["V0"] = self.parse\_interval(ivl.get("V0"))  self.intervals["J"] = self.parse\_interval(ivl.get("J"))  self.intervals["P"] = self.parse\_interval(ivl.get("P"))  self.intervals["Q"] = self.parse\_interval(ivl.get("Q"))  self.intervals["J0"] = self.parse\_interval(ivl.get("J0"))  self.intervals["P0"] = self.parse\_interval(ivl.get("P0"))  self.intervals["Q0"] = self.parse\_interval(ivl.get("Q0"))  def parse\_interval(self, str\_value):  values = str\_value.split(',')  return {"min":values[0], "max":values[1]}  def print(self):  print("Fi = ", self.fi)  print("Gamma = ", self.gamma)  print("Nju = ", self.nju)  print("Intervals: ", self.intervals) А.2 Модуль градиентного спуска gradient.py from math import sin, cos, pi, sqrt  class Gradient:  def \_\_init\_\_(self, GAMMA=pi/4, W=0.77, V=-0.3):  self.GAMMA = GAMMA  self.W = W  self.V = V  def Fxy(self, x, y):  return (cos(2 \* x) \* cos(2 \* y) - cos(self.GAMMA) \* sin(2 \* x) \* sin(2 \* y) - self.W) \*\* 2 \  + (sin(2 \* x) \* cos(2 \* y) + cos(self.GAMMA) \* cos(2 \* x) \* sin(2 \* y) - self.V) \*\* 2  def dFdx(self, x, y):  return 4 \* cos(2 \* x) \* cos(self.GAMMA) \* sin(2 \* y) \* self.W + 4 \* cos(self.GAMMA) \  \* sin(2 \* x) \* sin(2 \* y) \* self.V \  - 4 \* cos(2 \* x) \* cos(2 \* y) \* self.V + 4 \* cos(2 \* y) \* sin(2 \* x) \* self.W  def dFdy(self, x, y):  return - 4 \* cos(2 \* x) \* cos(2 \* y) \* cos(self.GAMMA) \* self.V \  + 4 \* cos(2 \* y) \* cos(self.GAMMA) \* cos(self.GAMMA) \* sin(2 \* y) \  + 4 \* cos(2 \* y) \* cos(self.GAMMA) \* sin(2 \* x) \* self.W \  + 4 \* cos(2 \* x) \* sin(2 \* y) \* self.W \  + 4 \* sin(2 \* x) \* sin(2 \* y) \* self.V - 4 \* cos(2 \* y) \* sin(2 \* y)  def GradientMethod(self, x0, y0, eps):  t = pi / 8  x = x0; y = y0  i = 0; n = 1000  while True:  if i > n:  x = x0  y = y0  return (x, y, False)  xt = x - t \* self.dFdx(x, y)  yt = y - t \* self.dFdy(x, y)  if abs(sqrt((x - xt) \* (x - xt)) + sqrt((y - yt) \* (y - yt))) < eps:  x = xt  y = yt  return (x, y, True)  Fx = self.Fxy(x, y)  Fxt = self.Fxy(xt, yt)  if Fx < Fxt:  t /= 2  else:  x = xt  y = yt  i += 1  if \_\_name\_\_ == '\_\_main\_\_':  g = Gradient()  a = 2  b = 3  e = 4  print(f'F({a}, {b}) = {g.Fxy(a, b)}')  print(f'dFdx({a}, {b}) = {g.dFdx(2, 3)}')  print(f'dFdy({a}, {b}) = {g.dFdy(2, 3)}')  print(f'GradientMethod({a}, {b}, {e}) = {g.GradientMethod(2, 3, 4)}') А.3 Модуль вычисления функции Лапласа Laplace.py from math import exp, sqrt, pi  def Simpson(f, a, b, eps):  n = 1; s = 0; s0 = eps + 1  while abs(s - s0) > eps:  n \*= 2; h = (b - a) / n; s0 = s  s = f(a) + f(b)  for i in range(1, n):  s += 2 \* (1 + i % 2) \* Fun(a + i \* h)  s \*= (b - a) / (3 \* n)  return s  def Fun(t):  return exp(-t \* t / 2)  def Laplace(x):  return Simpson(Fun, 0, x, 1E-12) / sqrt(2 \* pi)  if \_\_name\_\_ == '\_\_main\_\_':  for i in range(-5, 5):  print(f'FLaplace({i}) = {Laplace(i)}') А.4 Модуль расчета компонент ветора Стокса from math import sqrt, sin, cos, pi, atan  import os  class TIntensity:  def \_\_init\_\_(self, i, tau, phi):  self.\_i = i  self.\_tau = tau  self.\_phi = phi    @property  def i(self):  return self.\_i  @property  def tau(self):  return self.\_tau  @property  def phi(self):  return self.\_phi    @i.setter  def set\_i(self, value):  self.\_i = value    @tau.setter  def set\_tau(self, value):  self.\_tau = value  @phi.setter  def set\_phi(self, value):  self.\_phi = value  def \_\_repr\_\_(self):  return f'{self.i}, {self.tau}, {self.phi}'  class TStokesVector:  def \_\_init\_\_(self, j=0, q=0, u=0, v=0):  self.J = j  self.Q = q  self.U = u  self.V = v  self.error\_code = -1  @property  def P(self):  if abs(self.J) < 1e-8:  return 0.0  else:  result = sqrt(self.Q\*\*2 + self.U\*\*2 + self.V\*\*2) / self.J  if result > 1:  result = 1  return result    def calculate(self, I1, I2, I3, I4, WriteLog):  A = [([0] \* 5) for \_ in range(4)]  X = [0] \* 4  def Swap(k, n):  z = A[k][k]  i = k  for j in range(k + 1, n):  if abs(A[j][k]) > z:  z = A[j][k]  i = j  if i > k:  for j in range(k, n):  A[i][j], A[k][j] = A[k][j], A[i][j]    def LogMatrix():  if not WriteLog:  return  LogFile.write('------------------------------------------------------\n')  for i in range(4):  for j in range(4):  LogFile.write(f'{A[i][j]:9.4f}')  LogFile.write(f' | {A[i][4]:9.4f}\n')    self.ErrorCode = -1  self.J = 0  self.Q = 0  self.U = 0  self.V = 0  if WriteLog:  LogFile = open('LogPart01.txt', 'w', encoding="utf-8")  LogFile.writelines(  ('Расчет параметров вектора Стокса рассеянного излучения\n',  '------------------------------------------------------\n',  f'I1 = {I1.i:9.4f}\n',  f' tau1 = {I1.tau:9.4f}\n',  f' phi1 = {I1.phi:9.4f}\n',  f'I2 = {I2.i:9.4f}\n',  f' tau2 = {I2.tau:9.4f}\n',  f' phi2 = {I2.phi:9.4f}\n',  f'I3 = {I3.i:9.4f}\n',  f' tau3 = {I3.tau:9.4f}\n',  f' phi3 = {I3.phi:9.4f}\n',  f'I4 = {I4.i:9.4f}\n',  f' tau4 = {I4.tau:9.4f}\n',  f' phi4 = {I4.phi:9.4f}\n')  )  A[0][0] = 1  A[1][0] = 1  A[2][0] = 1  A[3][0] = 1  A[0][1] = cos(2 \* I1.phi)  A[1][1] = cos(2 \* I2.phi)  A[2][1] = cos(2 \* I3.phi)  A[3][1] = cos(2 \* I4.phi)  A[0][2] = sin(2 \* I1.phi) \* cos(I1.tau)  A[1][2] = sin(2 \* I2.phi) \* cos(I2.tau)  A[2][2] = sin(2 \* I3.phi) \* cos(I3.tau)  A[3][2] = sin(2 \* I4.phi) \* cos(I4.tau)  A[0][3] = sin(2 \* I1.phi) \* sin(I1.tau)  A[1][3] = sin(2 \* I2.phi) \* sin(I2.tau)  A[2][3] = sin(2 \* I3.phi) \* sin(I3.tau)  A[3][3] = sin(2 \* I4.phi) \* sin(I4.tau)  A[0][4] = 2 \* I1.i  A[1][4] = 2 \* I2.i  A[2][4] = 2 \* I3.i  A[3][4] = 2 \* I4.i  n = 4  LogMatrix()  for k in range(n):  if abs(A[k][k]) < 1e-10:  List = [abs(value) for value in A[k]]  idx = List.index(max(List[k:]))  del List  if idx != k:  Swap(k, idx)  else:  if abs(A[k][n]) > 1e-8:  self.ErrorCode = 2  else:  self.ErrorCode = 1  return I1, I2, I3, I4  for j in range(k + 1, n):  r = A[j][k] / A[k][k]  for i in range(n + 1):  A[j][i] -= r \* A[k][i]  # B[j] -= r \* B[k]  LogMatrix()    for k in reversed(range(n)):  r = 0  for j in range(k + 1, n):  g = A[k][j] \* X[j]  r += g  if A[k][k] == 0:  self.error\_code = -1  LogFile.close()  return  X[k] = (A[k][4] - r) / A[k][k]    self.J = X[0]  self.Q = X[1]  self.U = X[2]  self.V = X[3]  if WriteLog:  LogFile.writelines(  ('------------------------------------------------------\n',  f'J = {X[0]:9.4f}\n',  f'Q = {X[1]:9.4f}\n',  f'U = {X[2]:9.4f}\n',  f'V = {X[3]:9.4f}\n',  '------------------------------------------------------\n',  f'P = {self.P:9.4f}\n',  '------------------------------------------------------\n')  )  LogFile.close()    self.error\_code = 0  def SquareRoot(v, index):  a = sqrt(sqrt(v.real\*\*2 + v.imag\*\*2))  phi = atan(v.imag/v.real)  if v.real < 0:  phi += pi  if (index < 0) or (index > 1):  index = 0  if index == 1:  phi += pi  return complex(a\*cos(phi), a\*sin(phi))  class TStokesNaturalVector(TStokesVector):  def \_\_init\_\_(self, J=0, Q=0, U=0, V=0):  super().\_\_init\_\_(J, Q, U, V)  def calculateNatural(self, vector, nju, phi, WriteLog):  if WriteLog:  LogFile = open('LogPart02.txt', 'w', encoding="utf-8")  LogFile.writelines(  ('Расчет параметров вектора Стокса естественного излучения\n',  '------------------------------------------------------\n',  'Вектор Стокса рассеянного излучения\n',  f' J = {vector.J:9.4f}\n',  f' Q = {vector.Q:9.4f}\n',  f' U = {vector.U:9.4f}\n',  f' V = {vector.V:9.4f}\n',  f' P = {vector.P:9.4f}\n')  )    nju2 = nju\*\*2    cphi = cos(phi/180.0)  sphi = sin(phi/180.0)  cosphi = complex(cphi, 0)  sinphi = complex(sphi, 0)  sinphi2 = sinphi \* sinphi  sn = nju2 - sinphi2  sq = SquareRoot(sn, 0)  if sq.imag > 0:  sq = SquareRoot(sn, 1)  ch1 = cosphi - sq  zn1 = cosphi + sq  nju2cosphi = nju2 \* cosphi  ch2 = nju2cosphi - sq  zn2 = nju2cosphi + sq  r1 = ch1 / zn1  r2 = ch2 / zn2  r1\_2 = abs(r1) \*\* 2  r2\_2 = abs(r2) \*\* 2  r1r2\_ = r1 \* r2.conjugate()  r1r2\_2 = abs(r1r2\_) \*\* 2  self.J = (vector.J \* (r1\_2 + r2\_2) - vector.Q \* (r1\_2 - r2\_2)) / (2 \* r1\_2 \* r2\_2)  self.Q = (vector.Q \* (r1\_2 + r2\_2) - vector.J \* (r1\_2 - r2\_2)) / (2 \* r1\_2 \* r2\_2)  self.U = (vector.U \* r1r2\_.real + vector.V \* r1r2\_.imag) / r1r2\_2  self.V = (vector.V \* r1r2\_.real - vector.U \* r1r2\_.imag) / r1r2\_2    if WriteLog:  LogFile.writelines(  ('Исходные параметры\n',  '------------------------------------------------------\n',  f' Phi = {phi:9.4f} град.\n',  f' Nju = ({nju.real:9.4f}, {nju.imag:9.4f})\n',  '------------------------------------------------------\n',  'Промежуточные переменные\n',  '------------------------------------------------------\n',  f' Nju^2 = ({nju2.real:9.4f}, {nju2.imag:9.4f})\n',  f' Nju^2-sin(Phi)^2 = ({sn.real:9.4f}, {sn.imag:9.4f})\n',  f' SQRT(Nju^2-sin(Phi)^2) = ({sq.real:9.4f}, {sq.imag:9.4f})\n',  f' r1 = ({r1.real:9.4f}, {r1.imag:9.4f})\n',  f' r2 = ({r2.real:9.4f}, {r2.imag:9.4f})\n',  f' |r1|^2 = {r1\_2:9.4f}\n',  f' |r2|^2 = {r2\_2:9.4f}\n',  f' r1\*\_r2 = {r1r2\_.real:9.4f}, {r1r2\_.imag:9.4f})\n',  f' |r1\*\_r2|^2 = {r1r2\_2:9.4f}\n',  '------------------------------------------------------\n',  'Вектор Стокса естественного излучения\n',  f' J0 = {self.J:9.4f}\n',  f' Q0 = {self.Q:9.4f}\n',  f' U0 = {self.U:9.4f}\n',  f' V0 = {self.V:9.4f}\n',  f' P0 = {self.P:9.4f}\n')  )  LogFile.close()  if \_\_name\_\_ == '\_\_main\_\_':  pass А.5 Модуль расчета задач 1 и 2 import os  import csv  from math import sqrt, pi, sin, cos, tan, atan  from cmath import phase  from calculation import TIntensity, TStokesVector, TStokesNaturalVector  from gradient import Gradient  class TTask12:  def \_\_init\_\_(self, Idx, Alfa, Beta):  self.Idx = Idx  self.Alfa = Alfa  self.Beta = Beta  self.V = TStokesVector()  self.N = TStokesNaturalVector()  self.I = [TIntensity(0, 0, 0) for i in range(4)]  def load(self, index, dI, Tau, Phi):  dI = float(dI)  Tau = float(Tau)  Phi = float(Phi)  self.I[index] = TIntensity(dI, Tau, Phi)  def calcRadiation(self, nju, phi, WriteLog=False):  self.V = TStokesVector()  self.V.calculate(self.I[0], self.I[1], self.I[2], self.I[3], WriteLog)  self.N = TStokesNaturalVector()  self.N.calculateNatural(self.V, nju, phi, WriteLog)  return (self.V, self.N)  def calcReflection(self, nju, phi, gradient, isAnalitic=True, WriteLog=False):  # self.V = TStokesVector()  # self.N = TStokesNaturalVector()  self.V, self.N = self.calcRadiation(nju, phi, WriteLog)  J = self.V.J  Q = self.V.Q  U = self.V.U  V = self.V.V  if Q\*\*2 + V\*\*2 + U\*\*2 < 1e-5:  gradient.W = 0  gradient.V = 0  else:  gradient.W = Q / sqrt(Q \*\* 2 + V \*\*2 + U \*\* 2)  gradient.V = U / sqrt(Q \*\* 2 + V \*\*2 + U \*\* 2)    x = x0 = 0  y = y0 = 0  x1 = x2 = 0    if WriteLog:  LogFile = open('LogPart03.txt', 'w', encoding="utf-8")  LogFile.writelines(  ('Расчет параметров поляризации\n',  '------------------------------------------------------\n',  f'J = {J:9.4f}\n',  f'Q = {Q:9.4f}\n',  f'U = {U:9.4f}\n',  f'V = {V:9.4f}\n',  '------------------------------------------------------\n',  f'Gamma = {gradient.GAMMA:9.4f}\n',  f'Q / sqrt(Q\*Q + V\*V + U\*U) = {gradient.W:9.4f}\n',  f'U / sqrt(Q\*Q + V\*V + U\*U) = {gradient.V:9.4f}\n',  '------------------------------------------------------\n',  'Начальное приближение\n',  f'x0 = {x0:9.4f}\n',  f'y0 = {y0:9.4f}\n')  )    if isAnalitic:  isDone = True  Beta = abs(atan(V/sqrt(Q\*\*2+U\*\*2+(cos(gradient.GAMMA)\*\*2)\*(Q\*\*2+V\*\*2+U\*\*2))))  if gradient.GAMMA \* V > 0:  Beta \*= -1    Alpha = abs(atan(sin(2\*Beta)/cos(2\*Beta)\*cos(gradient.GAMMA)))  if abs(U) < 1e-5:  Alpha = 0  elif U < 0:  Alpha \*= -1  if WriteLog:  LogFile.writelines(  ('------------------------------------------------------\n',  'Найдено аналитическим методом:\n',  f'alpha = {Alpha:9.4f}\n',  f'beta = {Beta:9.4f}\n',  '------------------------------------------------------\n')  )  else:  x, y, isDone = gradient.GradientMethod(x0, y0, 1e-5)  if not isDone:  if WriteLog:  LogFile.writelines(  ('------------------------------------------------------\n',  'Метод градиента не смог найти минимум функции\n',  '------------------------------------------------------\n')  )  LogFile.close()  return    if abs(gradient.Fxy(x, y) > 1e-5):  if WriteLog:  LogFile.writelines(  ('------------------------------------------------------\n',  'Ошибка: система не имеет решения\n',  f'cos(2a)cos(2b) - cos(gamma)sin(2a)sin(2b) = {gradient.W:9.4f}\n',  f'sin(2a)cos(2b) + cos(gamma)sin(2a)sin(2b) = {gradient.V:9.4f}\n',  '------------------------------------------------------\n')  )  LogFile.close()  return    if WriteLog:  LogFile.writelines(  ('------------------------------------------------------\n',  'Найдено решение методом градиента\n',  f'alpha1 = {x:9.4f}\n',  f'beta1 = {y:9.4f}\n',  '------------------------------------------------------\n')  )    a = cos(-2 \* y) + gradient.W  b = sqrt(2) \* sin(-2 \* y)  c = gradient.w - cos(-2 \* y)  d = b \*\* 2 - 4 \* c \* a  if d < 0:  if WriteLog:  LogFile.writelines(  ('------------------------------------------------------\n',  f'Ошибка: не найдено второе решение alpha 2 для beta2 = {-y:9.4f}\n',  '------------------------------------------------------\n')  )  LogFile.close()  return    x1 = atan((-b + sqrt(d)) / 2 / a)  x2 = atan((-b - sqrt(d)) / 2 / a)  if abs(x1 - x) > abs(x2 - x):  x2 = x1  Alpha = (x + x2) / 2  if V > 0:  Beta = -abs(y)  else:  Beta = abs(y)    res = complex(tan(Alpha), tan(Beta)) / complex(1, -tan(Alpha)\*tan(Beta))    if WriteLog:  LogFile.writelines(  (f'alpha1 = {x:9.4f}\n',  f'beta1 = {y:9.4f}\n',  f'alpha2 = {x2:9.4f}\n',  f'beta2 = {-y:9.4f}\n',  '------------------------------------------------------\n',  f'alpha = (alpha1+alpha2)/2 = {Alpha:9.4f}\n',  f'beta = {Beta:9.4f}\n',  '------------------------------------------------------\n',  f'Re(Hi) = {res.real:9.4f}\n',  f'Im(Hi) = {res.imag:9.4f}\n',  f'|Hi| = {abs(res):9.4f}\n',  f'/\_Hi = {phase(res):9.4f}, рад. = {phase(res)/pi\*180:9.4f} град.\n',  '------------------------------------------------------\n',  'Расчет завершен\n',  '------------------------------------------------------\n')  )  LogFile.close()  def loadTask12(filename):  """  Read a CSV file using csv.DictReader  """  data = []  with open(filename) as f\_obj:  reader = csv.DictReader(f\_obj, delimiter=';')  for line in reader:  item = TTask12(int(line['ObjectId']), line['Alfa'], line['Beta'])  item.load(0, line['I1'], line['Tau1'], line['Phi1'])  item.load(1, line['I2'], line['Tau2'], line['Phi2'])  item.load(2, line['I3'], line['Tau3'], line['Phi3'])  item.load(3, line['I4'], line['Tau4'], line['Phi4'])  data.append(item)  return data  def calcTask1(data, nju, phi, WriteLog=False, Idx=None):  for item in data:  if (Idx == None) or (item.Idx == Idx):  item.calcRadiation(nju, phi, WriteLog)  def calcTask2(data, nju, phi, gradient, isAnalitic=True, WriteLog=False, Idx=None):  for item in data:  if (Idx == None) or (item.Idx == Idx):  item.calcReflection(nju, phi, gradient, isAnalitic, WriteLog)  if \_\_name\_\_ == '\_\_main\_\_':  os.chdir(os.path.dirname(\_\_file\_\_))  data = loadTask12('../data/csv/calculation.csv')  nju = 1.4 - 4.53j  phi = 52  g = Gradient()  calcTask1(data, nju, phi, True, Idx=174)  calcTask2(data, nju, phi, g, WriteLog=True, Idx=174) А.6 Модуль расчета задачи 3 import os  import csv  from math import pi, sin, cos, tan, exp, sqrt  from statistics import mean, stdev  from laplace import Laplace  def det2(a11, a21, a12, a22):  return a11\*a22 - a21\*a12  def len2rect(re, im, re1, re2, im1, im2):  x1 = min(re1, re2)  x2 = max(re1, re2)  y1 = min(im1, im2)  y2 = max(im1, im2)  """  1 | 2 | 3  4 | - | 5  6 | 7 | 8  """  if (re >= x1) and (re <= x2) and (im >= y1) and (im <= y2):  return 0  elif re < x1:  if im > y2:  return sqrt((re - x1)\*\*2 + (im - y2)\*\*2)  elif im < y1:  return sqrt((re - x1)\*\*2 + (im - y1)\*\*2)  else:  return abs(re - x1)  elif re > x2:  if im > y2:  return sqrt((re - x2)\*\*2 + (im - y2)\*\*2)  elif im < y1:  return sqrt((re - x2)\*\*2 + (im - y1)\*\*2)  else:  return abs(re - x2)  else:  return min(abs(im - y1), abs(im - y2))  def chi2P(chi, df):  """Return prob(chisq >= chi, with df degrees of freedom).  df must be even.  """  assert df & 1 == 0  # If chi is very large, exp(-m) will underflow to 0.  m = chi / 2.0  sum = term = exp(-m)  for i in range(1, df//2):  term \*= m / i  sum += term  # With small chi and large df, accumulated  # roundoff error, plus error in  # the platform exp(), can cause this to spill  # a few ULP above 1.0. For  # example, chi2P(100, 300) on my box  # has sum == 1.0 + 2.0\*\*-52 at this  # point. Returning a value even a teensy  # bit over 1.0 is no good.  return min(sum, 1.0)  class itemC:  def \_\_init\_\_(self):  self.is\_calc = False  self.value = 0  self.value\_norm = 0  def calc(self):  pass  class itemC1:  def \_\_init\_\_(self, dTetta, Q, U, V):  super().\_\_init\_\_()  self.dTetta = dTetta  self.Q = Q  self.U = U  self.V = V  def calc(self):  if self.V != 0:  self.value = (self.Q\*sin(2\*self.dTetta\*pi/180) \  + self.U\*cos(2\*self.dTetta\*pi/180))/self.V  self.is\_calc = True  else:  self.value = None  self.is\_calc = False  return self.value  class itemC2:  def \_\_init\_\_(self, dGamma, Q, U, V):  super().\_\_init\_\_()  self.dGamma = dGamma  self.Q = Q  self.U = U  self.V = V  def calc(self):  if self.V != 0:  self.value = self.U/self.V  self.is\_calc = True  else:  self.value = None  self.is\_calc = False  class itemC3:  def \_\_init\_\_(self, dGamma, Alfa, Beta):  super().\_\_init\_\_()  self.dGamma = dGamma  self.Alfa = Alfa  self.Beta = Beta  self.is\_calc = False    def calc(self):  self.value = self.Alfa\*self.Beta  self.is\_calc = True  class TInterval:  INTERVAL\_COUNT = 5  def \_\_init\_\_(self):  self.Number = None  self.Low = None  self.High = None  self.ValueCount = None  self.LaplaceValueLow = None  self.LaplaceValueHigh = None  self.Pi = None  self.NPi = None  self.Ni\_NPi = None  self.Ni\_NPi\_Norm = None  def getIntervals(icount, imax, imin):  length = (imax - imin) / icount  intervals = [] #[TInterval() for \_ in range(icount)]  for i in range(icount):  item = TInterval()  item.Number = i  item.ValueCount = 0  item.Low = imin + i \* length  item.High = imin + (i + 1) \* length  item.LaplaceValueLow = Laplace(item.Low)  item.LaplaceValueHigh = Laplace(item.High)  item.Pi = item.LaplaceValueHigh - item.LaplaceValueLow  intervals.append(item)  return intervals  def calcStat(data):  stat = {}  countP = sum([1 for item in data if item.is\_calc])  midP = mean([item.value for item in data if item.is\_calc])  sdP = stdev([item.value for item in data if item.is\_calc])  stat[0] = ('Всего', countP)  stat[1] = ('Среднее (по всем)', midP)  stat[2] = ('СКО (по всем)', sdP)  data\_n = []  for item in data:  item.is\_calc = item.is\_calc and abs(item.value-midP)<sdP  if item.is\_calc:  data\_n.append(item)  # count = sum([1 for item in data\_n])  mid = mean([item.value for item in data\_n])  sd = stdev([item.value for item in data\_n])  min\_v = min([item.value for item in data\_n])  max\_v = max([item.value for item in data\_n])  for item in data:  item.value\_norm = (item.value - mid)/sd  stat[3] = ('Минимальное', min\_v)  stat[4] = ('Максимальное', max\_v)  stat[5] = ('Среднее', mid)  stat[6] = ('Дисперсия', sd\*sd)  stat[7] = ('СКО', sd)  count\_n = sum([1 for item in data\_n])  min\_n = min([item.value\_norm for item in data\_n])  max\_n = max([item.value\_norm for item in data\_n])  stat[8] = ('Минимальное (норм)', min\_n)  stat[9] = ('Максимальное (норм)', max\_n)  intervals = getIntervals(TInterval.INTERVAL\_COUNT, max\_n, min\_n)  # for item in intervals:  # print(item.Number, item.Low, item.High)  SumProb = sum([item.Pi for item in intervals])    for item in data\_n:  value = round(item.value\_norm, 2)  index\_f = -1  for j in range(len(intervals)):  if abs(value - intervals[j].High) < 1e-3:  if j < TInterval.INTERVAL\_COUNT // 2:  index\_f = j + 1  else:  index\_f = j  break  if abs(value - intervals[j].Low) < 1e-3:  if j < TInterval.INTERVAL\_COUNT // 2:  index\_f = j  else:  index\_f = j - 1  break  if (value > intervals[j].Low) and (value < intervals[j].High + 1e-3):  index\_f = j  break  if index\_f == -1:  if value <= min\_n + 1e-2:  index\_f = 0  else:  index\_f = TInterval.INTERVAL\_COUNT - 1  index = int(index\_f)  if index > TInterval.INTERVAL\_COUNT - 1:  index = TInterval.INTERVAL\_COUNT - 1  elif index < 0:  index = 0  intervals[index].ValueCount += 1    i = 0  for item in intervals:  item.NPi = count\_n\*item.Pi/SumProb  item.Ni\_NPi = (item.ValueCount - item.NPi) \*\* 2  item.Ni\_NPi\_Norm = item.Ni\_NPi / item.NPi  stat[10+2\*i] = (f'Pi[{i}]', item.Pi)  stat[10+2\*i+1] = (f'N[{i}]', item.ValueCount)  i += 2    SumHi = sum([item.Ni\_NPi\_Norm for item in intervals])  # InvChiSquareDistribution(IntervalCount - 3, 0.05);  SumHiTeor = chi2P(0.05, TInterval.INTERVAL\_COUNT-3)  stat[i] = (f'Хи-квадрат (эмп.)', SumHi)  stat[i+1] = (f'Хи-квадрат (теор.)', SumHiTeor)  i += 2  midE = mean([item.value for item in data if item.is\_calc])  sdE = stdev([item.value for item in data if item.is\_calc])  Ek = max([abs(item.value - midE) for item in data if item.is\_calc])  Tk = Ek / sdE  FTk = 2\*Laplace(Tk)  stat[i] = ('Ek', Ek)  stat[i+1] = ('Tk', Tk)  stat[i+2] = ('Ф(Tk)', FTk)  # i += 3    return FTk, stat  IS\_DIELECTRIC = 0  IS\_METALLIC = 1  class TSample:  def \_\_init\_\_(self, Tetta, Lambda):  self.Tetta = Tetta  self.Lambda = Lambda  self.N = None  class TParam:  def \_\_init\_\_(self, Tetta1, Tetta2, A1, A2, V1, V2, U1, U2):  self.Tetta1 = Tetta1  self.A1 = A1  self.V1 = V1  self.U1 = U1  self.Tetta2 = Tetta2  self.A2 = A2  self.V2 = V2  self.U2 = U2  class TMaterial:  def \_\_init\_\_(self, name, label, re\_min, re\_max, im\_min, im\_max):  self.name = name  self.label = label  self.re\_min = re\_min  self.re\_max = re\_max  self.im\_min = im\_min  self.im\_max = im\_max  self.length = 0  def calcPlaneMaterial(data, materials, planeType, writeLog=True, logFileName='LogPart05.txt'):  if writeLog:  LogFile = open(logFileName, 'w', encoding="utf-8")  LogFile.writelines(  ('Определение материала покрытия\n',  '------------------------------------------------------\n')  )  if planeType == IS\_DIELECTRIC:  LogFile.writelines(' Расчет коэффициента преломления диэлектрического покрытия\n')  elif planeType == IS\_METALLIC:  LogFile.writelines(' Расчет коэффициента преломления металлического покрытия\n')  else:  pass  i = 0  idx = 0  val = data[0].Lambda  for item in data:  if item.Lambda < val:  idx = i  val = item.Lambda  i += 1  if writeLog:  LogFile.writelines(  (f' индекс точки минимума = {idx}\n',  '------------------------------------------------------\n'  'Исходные данные для расчета\n')  )  if writeLog:  if planeType == IS\_DIELECTRIC:  LogFile.writelines('Tetta Lambda N\n')  elif planeType == IS\_METALLIC:  LogFile.writelines('Tetta Lambda A\n')  else:  pass    # вычисляем значение n  i = 0  for item in data:  Tetta = pi\*item.Tetta/180.0  Lambda = item.Lambda  if planeType == IS\_DIELECTRIC:  if i <= idx:  item.N = sin(Tetta)/cos(Tetta)\  \*sqrt(1+Lambda\*\*2-2\*Lambda\*cos(2\*Tetta))\  /(1-Lambda)  else:  item.N = sin(Tetta)/cos(Tetta)\  \*sqrt(1+Lambda\*\*2+2\*Lambda\*cos(2\*Tetta))\  /(1+Lambda)  elif planeType == IS\_METALLIC:  if i <= idx:  item.N = sin(Tetta)\*\*2/cos(Tetta)\*(1+Lambda)/(1-Lambda)  else:  item.N = sin(Tetta)\*\*2/cos(Tetta)\*(1-Lambda)/(1+Lambda)  else:  pass  i += 1  nmid = sum([item.N for item in data])/(len(data)-1)  if writeLog:  for item in data:  LogFile.writelines(f'{item.Tetta:0.4f}\t{item.Lambda:0.4f}\t{item.N:0.4f}\n')  if planeType == IS\_DIELECTRIC:  coef = complex(nmid, 0)  elif planeType == IS\_METALLIC:  count = len(data)-1  if count // 2 == 1:  ParamCount = count // 2 + 1  i2 = count-1  else:  ParamCount = count // 2  i2 = round(count/2)  if writeLog:  LogFile.writelines(  ('------------------------------------------------------\n',  'начальный индекс i1 = 0\n',  f'начальный индекс i2 = {i2}\n',  f'количество уравнений = {ParamCount}\n')  )  params = []  for \_ in range(ParamCount):  params.append(TParam(0, 0, 0, 0, 0, 0, 0, 0))    for i in range(len(data)):  if i < ParamCount:  params[i].Tetta1 = data.Tetta  params[i].A1 = data.N  if i >= i2:  params[i].Tetta2 = data.Tetta  params[i].A2 = data.N  if writeLog:  LogFile.writelines(  ('------------------------------------------------------\n',  'Tetta1, A1, Tetta2, A2, V, U\n',  '------------------------------------------------------\n')  )    u\_mid = 0  v\_mid = 0  c\_mid = 0  for item in params:  if writeLog:  LogFile.writelines(  f'{item.Tetta1}, {item.A1}, {item.Tetta2}, {item.A2}\n'  )  d1 = det2(  item.A1\*\*4 + item.A1\*sin(item.Tetta1\*pi/180)\*\*2,  item.A2\*\*4 + item.A2\*sin(item.Tetta2\*pi/180)\*\*2,  item.A1\*\*2,  item.A2\*\*2  )  d2 = det2(1, 1, item.A1\*\*2, item.A2\*\*2)  if d1/d2 <0:  if writeLog:  LogFile.writelines(  f' невозможно вычислить V, так как V\*V = {d1/d2}\n'  )  else:  v = sqrt(d1/d2)  d3 = det2(  1, 1,  item.A1\*\*4 + item.A1\*sin(item.Tetta1\*pi/180)\*\*2,  item.A2\*\*4 + item.A2\*sin(item.Tetta2\*pi/180)\*\*2  )  d4 = det2(1, 1, item.A1\*\*2, item.A2\*\*2)  u = d3/d4  if abs(d2)>1e-3 and abs(d4)>1e-3:  v\_mid += v  u\_mid += u  c\_mid += 1  if writeLog:  LogFile.writelines(f'V={v}, U={u}\n')  else:  if writeLog:  LogFile.writelines(f'система уравнений имеет неустойчивое решение!\n')  v\_mid /= c\_mid  u\_mid /=c\_mid  re = sqrt((u\_mid + sqrt(u\_mid\*\*2 + 4\*v\_mid\*\*2))/2)  coef = complex(re, -v\_mid/re)  if writeLog:  LogFile.writelines(  ('------------------------------------------------------\n',  ' Vmid = {Vmid}\n',  ' Umid = {Umid}\n',  '------------------------------------------------------\n',  'Коэффициент преломления:\n',  'Re={coef.real}:\n',  'Im={coef.imag}:\n',  '------------------------------------------------------\n')  )  if writeLog:  LogFile.writelines(f'Коэффициент преломления: ({coef.real:0.4f}+{coef.imag:0.4f}j)\n')  if writeLog:  LogFile.writelines(  ('------------------------------------------------------\n',  'Материалы, Коэффициент\n')  )  i = 0  idx = 0  length = 1e+38  for item in materials:  if writeLog:  LogFile.writelines(f'{item.name}, {item.label}\n')  item.length = len2rect(  coef.real, coef.imag,  item.re\_min, item.re\_max,  item.im\_min, item.im\_max  )  if item.length < length:  idx, length = i, item.length  i += 1  if writeLog:  LogFile.writelines(  ('------------------------------------------------------\n',  f'Наиболее близкий по характеристике материал к Re={coef.real}, Im={coef.imag}:\n',  f'{materials[idx].name}\n',  f'{materials[idx].label}\n')  )  LogFile.close()  return idx, materials[idx]  def loadC1(filename):  """  Read a C1.CSV file using csv.DictReader  """  data = []  with open(filename) as f\_obj:  reader = csv.DictReader(f\_obj, delimiter=';')  for line in reader:  dTetta = float(line['dTetta'])  Q = float(line['Q'])  U = float(line['U'])  V = float(line['V'])  item = itemC1(dTetta, Q, U, V)  item.calc()  data.append(item)  return data  def loadC2(filename):  """  Read a C2.CSV file using csv.DictReader  """  data = []  with open(filename) as f\_obj:  reader = csv.DictReader(f\_obj, delimiter=';')  for line in reader:  # dGamma, Q, U, V  dGamma = float(line['dGamma'])  Q = float(line['Q'])  U = float(line['U'])  V = float(line['V'])  item = itemC2(dGamma, Q, U, V)  item.calc()  data.append(item)  return data  def loadC3(filename):  """  Read a C3.CSV file using csv.DictReader  """  data = []  with open(filename) as f\_obj:  reader = csv.DictReader(f\_obj, delimiter=';')  for line in reader:  # dGamma, Alfa, Beta  dGamma = float(line['dGamma'])  Alfa = float(line['Alfa'])  Beta = float(line['Beta'])  item = itemC3(dGamma, Alfa, Beta)  item.calc()  data.append(item)  return data  def loadMaterial(filename):  """  Read a MaterialRefraction.CSV file using csv.DictReader  """  data = []  with open(filename, encoding="utf-8") as f\_obj:  reader = csv.DictReader(f\_obj, delimiter=';')  for line in reader:  name = line['MaterialName']  re\_min = float(line['ReValueMin'])  re\_max = float(line['ReValueMax'])  im\_min = float(line['ImValueMin'])  im\_max = float(line['ImValueMax'])  label = f'({re\_min:0.2f}..{re\_max:0.2f})+({im\_min:0.2f}..{im\_max:0.2f})j'  item = TMaterial(name, label, re\_min, re\_max, im\_min, im\_max)  data.append(item)  return data  def loadSamples(filename):  """  Read a samples file for plane type calculation  """  data = []  with open(filename, encoding="utf-8") as f\_obj:  reader = csv.DictReader(f\_obj, delimiter=';')  for line in reader:  Tetta = float(line['dTetta'])  Lambda = float(line['Lambda'])  item = TSample(Tetta, Lambda)  data.append(item)  return data  def printStat(stat, filename, mode='w', title=None):  LogFile = open(filename, mode, encoding="utf-8")  if title != None:  LogFile.writelines(f'{title}\n')  for key in sorted(stat.keys()):  LogFile.writelines(f'{stat[key][0]}:\t{stat[key][1]:0.4f}\n')  LogFile.writelines('-'\*60+'\n')  LogFile.close()  def calcTask3(data, nju, phi, WriteLog=False, Idx=None):  for item in data:  if (Idx == None) or (item.Idx == Idx):  pass  # item.calcRadiation(nju, phi, WriteLog)  def test\_Task03():  os.chdir(os.path.dirname(\_\_file\_\_))  c1 = loadC1('../data/csv/C1.csv')  c2 = loadC2('../data/csv/C2.csv')  c3 = loadC3('../data/csv/C3.csv')  p1, s1 = calcStat([item for item in c1 if item.value != None])  p2, s2 = calcStat([item for item in c2 if item.value != None])  p3, s3 = calcStat([item for item in c3 if item.value != None])  filename = 'LogPart04.txt'    printStat(s1, filename, 'w', 'Расчет признака С1')  printStat(s2, filename, 'a', 'Расчет признака С2')  printStat(s3, filename, 'a', 'Расчет признака С3')  LogFile = open(filename, 'a', encoding="utf-8")  LogFile.writelines(f'Вероятность попадания объекта в класс = {p1\*p2\*p3:0.4f}')  LogFile.close()  def test\_Task04():  os.chdir(os.path.dirname(\_\_file\_\_))    materials = loadMaterial('../data/csv/MaterialRefraction.csv')    data = loadSamples('../data/csv/task03\_test01\_diel.csv')  planeType = IS\_DIELECTRIC  \_, \_ = calcPlaneMaterial(data, materials, planeType, \  writeLog=True, logFileName='LogPart05\_test01.txt')  # print(idx, material)  if \_\_name\_\_ == "\_\_main\_\_":  test\_Task03()  test\_Task04() |
| Приложение БПротокол расчета рассеянного изучения Расчет параметров вектора Стокса рассеянного излучения  ------------------------------------------------------  I1 = 1.4050  tau1 = 0.0000  phi1 = 0.0000  I2 = 0.0200  tau2 = 0.0000  phi2 = 1.5708  I3 = 0.4750  tau3 = 0.0000  phi3 = 0.7854  I4 = 0.4750  tau4 = 1.5708  phi4 = 0.7854  ------------------------------------------------------  1.0000 1.0000 0.0000 0.0000 | 2.8100  1.0000 -1.0000 -0.0000 -0.0000 | 0.0400  1.0000 0.0000 1.0000 0.0000 | 0.9500  1.0000 0.0000 -0.0000 1.0000 | 0.9500  ------------------------------------------------------  1.0000 1.0000 0.0000 0.0000 | 2.8100  0.0000 -2.0000 -0.0000 -0.0000 | -2.7700  0.0000 -1.0000 1.0000 0.0000 | -1.8600  0.0000 -1.0000 -0.0000 1.0000 | -1.8600  ------------------------------------------------------  1.0000 1.0000 0.0000 0.0000 | 2.8100  0.0000 -2.0000 -0.0000 -0.0000 | -2.7700  0.0000 0.0000 1.0000 0.0000 | -0.4750  0.0000 0.0000 -0.0000 1.0000 | -0.4750  ------------------------------------------------------  1.0000 1.0000 0.0000 0.0000 | 2.8100  0.0000 -2.0000 -0.0000 -0.0000 | -2.7700  0.0000 0.0000 1.0000 0.0000 | -0.4750  0.0000 0.0000 0.0000 1.0000 | -0.4750  ------------------------------------------------------  1.0000 1.0000 0.0000 0.0000 | 2.8100  0.0000 -2.0000 -0.0000 -0.0000 | -2.7700  0.0000 0.0000 1.0000 0.0000 | -0.4750  0.0000 0.0000 0.0000 1.0000 | -0.4750  ------------------------------------------------------  J = 1.4250  Q = 1.3850  U = -0.4750  V = -0.4750  ------------------------------------------------------  P = 1.0000  ------------------------------------------------------ |
| Приложение ВПротокол расчета естественного изучения Расчет параметров вектора Стокса естественного излучения  ------------------------------------------------------  Вектор Стокса рассеянного излучения  J = 1.4250  Q = 1.3850  U = -0.4750  V = -0.4750  P = 1.0000  Исходные параметры  ------------------------------------------------------  Phi = 52.0000 град.  Nju = ( 1.4000, -4.5300)  ------------------------------------------------------  Промежуточные переменные  ------------------------------------------------------  Nju^2 = ( -18.5609, -12.6840)  Nju^2-sin(Phi)^2 = ( -18.6421, -12.6840)  SQRT(Nju^2-sin(Phi)^2) = ( -3.9259, -2.6712)  r1 = ( -1.3569, 0.3213)  r2 = ( 0.6389, 0.0006)  |r1|^2 = 1.9443  |r2|^2 = 0.4082  r1\*\_r2 = -0.8667, 0.2061)  |r1\*\_r2|^2 = 0.7936  ------------------------------------------------------  Вектор Стокса естественного излучения  J0 = 0.7716  Q0 = 0.6736  U0 = 0.3954  V0 = 0.6421  P0 = 1.0000 |
| Приложение ГПротокол расчета поляризации Расчет параметров поляризации  ------------------------------------------------------  J = 1.4250  Q = 1.3850  U = -0.4750  V = -0.4750  ------------------------------------------------------  Gamma = 0.7854  Q / sqrt(Q\*Q + V\*V + U\*U) = 0.8998  U / sqrt(Q\*Q + V\*V + U\*U) = -0.3086  ------------------------------------------------------  Начальное приближение  x0 = 0.0000  y0 = 0.0000  ------------------------------------------------------  Найдено аналитическим методом:  alpha = -0.3762  beta = 0.2547  ------------------------------------------------------  alpha1 = 0.0000  beta1 = 0.0000  alpha2 = 0.0000  beta2 = 0.0000  ------------------------------------------------------  alpha = (alpha1+alpha2)/2 = -0.3762  beta = 0.2547  ------------------------------------------------------  Re(Hi) = -0.3643  Im(Hi) = 0.2978  |Hi| = 0.4706  /\_Hi = 2.4563, рад. = 140.7369 град.  ------------------------------------------------------  Расчет завершен  ------------------------------------------------------ |
| Приложение ДПротокол расчета материала покрытия |

Расчет признака С1

Всего: 16.0000

Среднее (по всем): 1.7569

СКО (по всем): 0.7167

Минимальное: 1.6769

Максимальное: 2.2110

Среднее: 1.9323

Дисперсия: 0.0286

СКО: 0.1692

Минимальное (норм): -1.5096

Максимальное (норм): 1.6474

Хи-квадрат (эмп.): 3.1115

Хи-квадрат (теор.): 0.9753

Ek: 0.2787

Tk: 1.6474

Ф(Tk): 0.9005

N[2]: 1.0000

Pi[4]: 0.2472

N[4]: 5.0000

Pi[6]: 0.1955

N[6]: 2.0000

Pi[8]: 0.1051

N[8]: 2.0000

-------------------------------------------------------

Расчет признака С2

Всего: 19.0000

Среднее (по всем): 5.9761

СКО (по всем): 4.9261

Минимальное: 2.5000

Максимальное: 10.6250

Среднее: 4.2284

Дисперсия: 3.9700

СКО: 1.9925

Минимальное (норм): -0.8674

Максимальное (норм): 3.2104

Хи-квадрат (эмп.): 10.9259

Хи-квадрат (теор.): 0.9753

Ek: 6.3966

Tk: 3.2104

Ф(Tk): 0.9987

N[2]: 6.0000

Pi[4]: 0.1654

N[4]: 0.0000

Pi[6]: 0.0488

N[6]: 0.0000

Pi[8]: 0.0077

N[8]: 1.0000

-------------------------------------------------------

Расчет признака С3

Всего: 19.0000

Среднее (по всем): 2.2652

СКО (по всем): 2.0963

Минимальное: 0.5590

Максимальное: 2.8710

Среднее: 1.8022

Дисперсия: 0.3402

СКО: 0.5832

Минимальное (норм): -2.1315

Максимальное (норм): 1.8326

Хи-квадрат (эмп.): 4.8596

Хи-квадрат (теор.): 0.9753

Ek: 1.2432

Tk: 2.1315

Ф(Tk): 0.9670

N[2]: 2.0000

Pi[4]: 0.3050

N[4]: 9.0000

Pi[6]: 0.2532

N[6]: 2.0000

Pi[8]: 0.1158

N[8]: 3.0000

-------------------------------------------------------

Вероятность попадания объекта в класс = 0.8696

Определение материала покрытия

------------------------------------------------------

Расчет коэффициента преломления диэлектрического покрытия

индекс точки минимума = 19

------------------------------------------------------

Исходные данные для расчета

Tetta Lambda N

3.0000 0.9970 1.8265

6.0000 0.9870 1.6825

9.0000 0.9720 1.7520

12.0000 0.9490 1.7016

15.0000 0.9210 1.7061

18.0000 0.8880 1.7205

21.0000 0.8500 1.7341

24.0000 0.7920 1.6123

27.0000 0.7566 1.7300

30.0000 0.7010 1.7167

33.0000 0.6471 1.7383

36.0000 0.5741 1.6842

39.0000 0.5090 1.6879

42.0000 0.4470 1.7126

45.0000 0.3850 1.7424

48.0000 0.3150 1.7500

51.0000 0.2350 1.7333

54.0000 0.1590 1.7348

57.0000 0.0850 1.7460

60.0000 0.0080 1.7530

63.0000 0.0096 1.9330

66.0000 0.1670 1.7262

69.0000 0.2660 1.6911

72.0000 0.3430 1.7190

75.0000 0.4320 1.7256

78.0000 0.5230 1.7419

81.0000 0.5380 2.1177

84.0000 0.6360 2.3284

87.0000 0.8650 1.7029

Коэффициент преломления: (1.8268+0.0000j)

------------------------------------------------------

Материалы, Коэффициент

Кварц, (1.54..1.55)+(0.00..0.00)j

Молибден, (3.15..3.15)+(-3.72..-3.72)j

Серебро, (0.11..0.11)+(-2.94..-2.94)j

Ниобий, (2.13..2.13)+(-3.07..-3.07)j

Никель, (1.54..1.54)+(-3.10..-3.10)j

Нержавеющая сталь, (1.46..1.46)+(-3.17..-3.17)j

Золото, (0.50..0.50)+(-2.04..-2.04)j

Стеклоткань, (10.89..10.89)+(0.00..0.00)j

Сплавы титана, (2.10..2.20)+(-2.80..-2.90)j

Кремниевое стекло, (1.50..1.80)+(0.00..0.00)j

Сплав алюминия со сталью, (0.95..0.95)+(-3.80..-3.80)j

Платина, (1.76..1.76)+(-3.59..-3.59)j

Двуокись кремния, (1.46..1.46)+(0.00..0.00)j

Сплавы алюминия, (0.50..0.80)+(-4.20..-4.60)j

Молибденовая проволока с золотым покрытием, (0.90..0.90)+(-2.50..-2.50)j

Сульфид свинца, (1.70..1.70)+(-3.30..-3.30)j

Алюминий, (0.50..0.50)+(-4.59..-4.59)j

Вольфрам, (3.31..3.31)+(-0.89..-0.89)j

------------------------------------------------------

Наиболее близкий по характеристике материал к Re=1.826807401084923, Im=0.0:

Кремниевое стекло

(1.50..1.80)+(0.00..0.00)j