МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Рязанский государственный радиотехнический университет

им. В.Ф. Уткина»

Кафедра космических технологий

|  |  |  |
| --- | --- | --- |
|  |  | «К защите»  Заведующий кафедрой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_»\_\_\_\_\_\_\_\_20\_\_\_ г. |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ**

**РАБОТА БАКАЛАВРА**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

на тему

**«Разработка информационной системы регистрации оптических параметров космических объектов»**

Направление подготовки: 02.03.01 «Математика и компьютерные науки»

ОПОП «Математика и компьютерные науки»

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Швецова Д.А.)

Научный руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Наумов Д.А.)

Руководитель ОПОП \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Таганов А.И.)

Рязань, 2021 г.

МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Рязанский государственный радиотехнический университет им. В.Ф. Уткина»

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_» \_\_\_\_\_\_20\_\_\_\_ г.

ЗАДАНИЕ

на дипломное проектирование

Студенту Швецовой Д.А.

1.Тема проекта: Разработка информационной системы регистрации оптических параметров космических объектов

2. Срок сдачи студентом законченного проекта:

3. Руководитель проекта Наумов Дмитрий Анатольевич, РГРТУ им. В.Ф. Уткина, доцент кафедры КТ

(фамилия, имя, отчество полностью, место работы, должность)

4.Исходные данные к проекту:

5. Содержание расчетно-пояснительной записки (технико-экономическое обоснование темы, расчетная, экспериментальная, экономическая часть и др. с расшифровкой задания по каждой части):

Введение.

1. ОПТИЧЕСКИЕ НАБЛЮДЕНИЯ ОБЪЕКТОВ КОСМИЧЕСКОГО ПРОСТРАНСТВА

2. МЕТОДЫ И АЛГОРИТМЫ ОПРЕДЕЛЕНИЯ ТИПА ПОКРЫТИЯ КОСМИЧЕСКИХ ОБЪЕКТОВ

3. ИНФОРМАЦИОННАЯ СИСТЕМА РЕГИСТРАЦИИ ОПТИЧЕСКИХ ПАРАМЕТРОВ КОСМИЧЕСКИХ ОБЪЕКТОВ STOKES-QT

Заключение

Библиографический список

Приложения

Консультанты по проекту (с указанием относящихся к ним разделов проекта):

Консультант от кафедры\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_\_\_» \_\_\_\_\_ 20\_\_\_ г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Наумов Д.А.)

(подпись)

Задание принял к исполнению «\_\_\_\_» \_\_\_\_\_\_ 20\_\_\_ г.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Аннотация

Выпускная квалификационная работа изложена на 95 страницах, состоит из введения, трех глав, заключения, включает 34 рисунка, 10 таблиц, библиографический список из 21 источника литературы, 5 приложений на 56 страницах.

Объектом выпускной квалификационной работы является разработка информационной системы регистрации оптических параметров космических объектов.

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

Цель выпускной квалификационной работы – разработка интерфейса и базы данных информационной системы для ведения базы данных фотометрических изменений и расчетов 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:

* *с++*;
* формат базы данных:*postgres*;
* *Qt*;
* *git*.

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

Содержание

[Введение 8](#_Toc74307361)

[1 наблюдениЕ объектов космического пространства 11](#_Toc74307362)

[1.1 Современные оптические средства наблюдения объектов космического пространства 11](#_Toc74307363)

[1.2 Получение оптическими средствами фотометрической информации о космических объектах 13](#_Toc74307364)

[1.3 Проведение оптических наблюдений и обработки измерений объектов космического мусора 17](#_Toc74307365)

[1.4 Параметры движения и физических характеристик объектов космического мусора в области ГСО 18](#_Toc74307366)

[2 Методы и алгоритмы определения типа покрытия космических объектов 21](#_Toc74307367)

[2.1 Постановка задачи фотометрии 21](#_Toc74307368)

[2.2. Расчет компонент вектора Стокса 22](#_Toc74307369)

[2.3 Вычисление параметров Стокса падающего излучения 23](#_Toc74307370)

[2.4. Алгоритмы нахождения дополнительных параметров поляризации рассеянного светового потока 25](#_Toc74307371)

[2.5 Определение типа покрытия космических объектов 27](#_Toc74307372)

[2.6. Алгоритм вычисления показателя преломления однородной диэлектрической поверхностью 30](#_Toc74307373)

[2.7. Алгоритм вычисления показателя преломления металлических покрытий 32](#_Toc74307374)

[2 Информационная система регистрации оптических параметров космических объектов STOKES-QT 37](#_Toc74307375)

[2.1 Этапы автоматизации регистрации и расчета оптических параметров космических объектов 37](#_Toc74307376)

[2.1.1 Уровень автоматизации I 37](#_Toc74307377)

[2.1.2 Уровень автоматизации II 38](#_Toc74307378)

[2.1.3 Уровень автоматизации III 38](#_Toc74307379)

[2.2 Опредение требований к информационной системе 39](#_Toc74307380)

[2.2.1 Создание функционального описания 39](#_Toc74307381)

[2.2.2 Программные средства 39](#_Toc74307382)

[2.2.3 Метод ввода 40](#_Toc74307383)

[2.2.4 Метод вывода 40](#_Toc74307384)

[2.3 Обоснование выбора средств разработки программной системы 40](#_Toc74307385)

[2.3.1 Язык С++ 40](#_Toc74307386)

[2.3.2 Среда разработки Qt 42](#_Toc74307387)

[2.3.3 База данных PostgresQL 43](#_Toc74307388)

[2.3.4 Система контроля версий git 44](#_Toc74307389)

[3.2 Архитектура программной системы STOKES 45](#_Toc74307390)

[Описание основных классов программы 47](#_Toc74307391)

[3.3 Разработка схемы данных программной системы STOKES 54](#_Toc74307392)

[3.3.1 Спецификация сущностей 54](#_Toc74307393)

[3.3.3 Спецификация связей 59](#_Toc74307394)

[3.3.4 Формализация зависимостей 60](#_Toc74307395)

[3.3.5 Определение таблиц базы данных 64](#_Toc74307396)

[3.3.6 Схема баз данных 66](#_Toc74307397)

[3.3.7 Нормализация баз данных 67](#_Toc74307398)

[3.4 Тестирование программной системы STOKES 70](#_Toc74307399)

[4 Разработка программной документации 77](#_Toc74307400)

[4.1 Описание применения 77](#_Toc74307401)

[4.1.1 Назначение и условия применения информационной   
системы 77](#_Toc74307402)

[4.1.2 Условия применения комплекса 78](#_Toc74307403)

[4.2 Руководство программиста информационной системы STOKES 78](#_Toc74307404)

[4.2.1 Характеристики программной системы 78](#_Toc74307405)

[4.2.2 Настройка программной системы 79](#_Toc74307406)

[4.2.3 Обращение к программной системе 79](#_Toc74307407)

[4.2.4 Входные и выходные данные 80](#_Toc74307408)

[4.3 Руководство оператора 80](#_Toc74307409)

[4.4.1 Основные функции системы 81](#_Toc74307410)

[4.4.2 Выполнение программы 81](#_Toc74307411)

[Заключение 95](#_Toc74307412)

[Библиографический список 96](#_Toc74307413)

[Приложение А. Листинг основных программных модулей 98](#_Toc74307414)

[А.1 Модуль calculation 98](#_Toc74307415)

[А.2 Модуль chisqr 99](#_Toc74307416)

[A.4 Модуль gradient 100](#_Toc74307417)

[A.5 Модуль laplase 101](#_Toc74307418)

[A6. Модуль stokes\_vector 102](#_Toc74307419)

[A7. Модуль task\_12 106](#_Toc74307420)

[Файл task\_12.h 106](#_Toc74307421)

[Файл task12.cpp 106](#_Toc74307422)

[A8. Модуль task3 109](#_Toc74307423)

[Файл task3.h 109](#_Toc74307424)

[Файл task3.cpp 111](#_Toc74307425)

[А.9 Модули, реализующие интерфес пользователя 120](#_Toc74307426)

[Файл main.cpp 120](#_Toc74307427)

[Файл mainwindow.h 121](#_Toc74307428)

[Файл mainwindow.cpp 121](#_Toc74307429)

[Файл database.h 124](#_Toc74307430)

[Файл database.cpp 124](#_Toc74307431)

[Файл experimentswindow.h 125](#_Toc74307432)

[Файл experimentswindow.cpp 126](#_Toc74307433)

[Файл editexperimentdialog.cpp 131](#_Toc74307434)

[Файл roweditingdelegate.cpp 131](#_Toc74307435)

[Файл roweditingdelegate.cpp 132](#_Toc74307436)

[Файл plotwidget.h 133](#_Toc74307437)

[Файл plotwidget.cpp 134](#_Toc74307438)

[Файл constanswindow.h 136](#_Toc74307439)

[Файл constantswindow.cpp 136](#_Toc74307440)

[Файл loaddatadialog.h 137](#_Toc74307441)

[Файл loaddatadialog.cpp 138](#_Toc74307442)

[Файл logfilewidget.h 139](#_Toc74307443)

[Файл logfilewidget.cpp 140](#_Toc74307444)

[Приложение Б. Протокол расчета рассеянного изучения 141](#_Toc74307445)

[Приложение В. Протокол расчета естественного изучения 143](#_Toc74307446)

[Приложение Г. Протокол расчета поляризации 145](#_Toc74307447)

[Приложение Д. Протокол расчета материала покрытия 147](#_Toc74307448)

# Введение

Тема выпускной квалификационной работы – разработка информационной системы регистрации оптических параметров космических объектов.

Современный путь освоения космического пространства приводит к экологическим угрозам атмосфере и околоземному космическому пространству – количество "отходов", "космического мусора" расчет вместе с освоением космического пространства, так как с течением времени запущенные на земную орбиту аппараты заканчивают свой срок функционирования.

Космический мусор – все находящиеся на околоземной орбите или возвращающиеся в атмосферу антропогенные объекты, включая их фрагменты и элементы, которые являются нефункциональными [1].

Дальнейшее освоение околоземного пространства невозможно без объективного анализа текущего состояния загрязнения. Особенно остро этот вопрос стоит по отношению к орбитам до 2000 км., которые называют низкими, а также в геостационарных орбитах, где находится большая масса антропогенный частиц и объектов и где большая вероятность опасных взаимных катастрофических столкновений космических объектов. По мере роста количества спутников на орбите и устаревания существующих, риск лавинообразного развития синдрома Кесслера всё возрастает. Главная особенность синдрома Кесслера заключается в «эффекте домино» [2, 3].

Количество антропогенных частиц и объектов в области низких околоземных орбит составляет приблизительно 77 % от общего числа каталогизированных объектов, в области геостационарной орбиты сосредоточено около 6 % каталогизированных объектов, 10% — в области высокоэллиптических орбит и примерно 7 % — на других орбитах, в том числе в области навигационных спутниковых систем. По своему составу каталогизированные объекты включают 20% космических аппаратов. Среди отслеживаемых космических объектов лишь 7% (около 900 объектов) являются активно функционирующими космическими аппаратами. Порядка 11% составляют ступени ракетоносителей и разгонные блоки, 5 % — операционные элементы, образовавшиеся в процессе выведения космических аппаратов на рабочие орбиты [5, 6, 7].

Основная часть объектов, включенных в каталоги, является продуктами разрушения средств выведения в космос:

* отработанные ступени баллистических ракет;
* верхние ступени ракетоносителей;
* разгонные блоки;
* элементы конструкций ракет;
* космические аппараты различного назначения.

Каталоги этих систем содержат известные оценки курса антропогенных частиц и объектов, и могут быть использованных для прогнозировать их движение [10].

Данная работа посвящена разработке информационной системе регистрации оптических параметров космических объектов и идентификации космических объектов с помощью методов фотометрии.

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", а также выпускная квалификационная работа Фоломеева К.В., гр. 648.

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, 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, четырех разделов, заключения и приложений.

Первый раздел пояснительной записки содержит обзор современных средств наблюдения за космическими объектами, описание программных средств обработки оптических данных.

Во втором разделе описаны математические модели и алгоритмы прямой и обратной задачи фотометрии, описание 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 информационной системы регистрации оптических параметров космических объектов STOKES. В разделе дается краткое описание архитектуры программной системы, ее информационной модели, руководство программиста.

Четвертый раздел содержит описание применения, руководство программиста и руководство пользователя.

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

# 1 наблюдениЕ объектов космического пространства

## 1.1 Современные оптические средства наблюдения объектов космического пространства

Все искусственные космические объекты, находящиеся на околоземных орбитах, нуждаются в постоянном мониторинге их движения, каталогизации и уточнении орбитальных параметров для последующего прогнозирования их положения. В первую очередь контролю на орбите подлежат крупные космические объекты (КО). Информация об их местоположении требует регулярного уточнения с помощью новых наблюдательных данных. Для осуществления этих целей необходима сеть наблюдательных средств - система контроля космического пространства.

В настоящее время в мире существуют две системы контроля космического пространства (СККП), созданные с целью национальной безопасности – принадлежащие Министерствам обороны Российской Федерации и США [11].

Задачи СККП:

* ведение общего каталога искусственных космических объектов;
* обнаружение новых космических объектов;
* определение первоначальных орбит космических объектов;
* уточнение орбит каталогизированных объектов;
* прогнозирование движения космических объектов;
* определение некоординатных характеристик космических объектов;
* оценка космической обстановки.

Объекты космического мусора существуют во всём околоземном пространстве: от высот МКС до ГСО [10]. Классификация орбит:

* низкие околоземные орбиты;
* солнечно-синхронные орбиты;
* cредневысотные орбиты;
* высокоэллиптические орбиты;
* геостационарная орбита;
* геосинхронные орбиты;
* орбиты захоронения.

Существуют два вида средств наземного наблюдения объектов космического мусора:

* оптические средства;
* радиолокационные средства.

Средства наблюдения объектов космического мусора в оптическом диапазоне используются для обнаружения и наблюдения фрагментов на высоких и переходных орбитах. Предельный размер сопровождаемых объектов составляет 10-50 см [10].

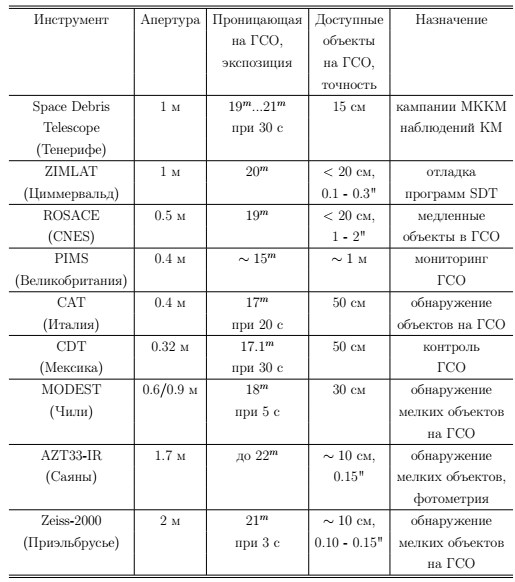
Недостатки оптических наблюдательных:

* зависимость от времени суток, погоды, фаз Луны, фазы освещённости цели;
* доступно малое количество целей одновременно;
* требуется сложное программное обеспечение.

Фрагменты космического мусора обладают изначально неизвестной формой, особенностями изменения блеска, несвойственными функционирующим аппаратам. Из-за отсутствия какой- ибо первоначальной информации об объекте возникает сложность наблюдения и сопровождения неизвестных объектов [10].

В таблице 1.1 приведены характеристики некоторых современных оптических систем, использующихся для наблюдения за космическим мусором [9].

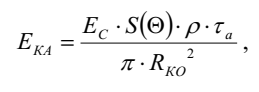
Таблица 1.1



## 1.2 Получение оптическими средствами фотометрической информации о космических объектах

Основными источниками получения фотометрической и астрометрической информации по космическим объектам в околоземном космическом пространстве являются оптические и оптико-электронные средства (ОЭС), принимающие пассивно рассеиваемое космическим объектом солнечное излучение. Блеск, создаваемый КО, зависит от его характеристик (размера, формы, отражательных характеристики поверхности, параметров движения относительно центра масс) и условий его наблюдения (расстояния до КО, коэффициент пропускания атмосферы и взаимного расположения Солнца, КО и ОЭС).

Космический объект простой геометрической формы создает в апертурной плоскости фотометра освещенность [13]:



где *EС* - солнечная постоянная, т.е. освещенность, создаваемая Солнцем в окрестностях Земли и равная 135000 люкс;

*S*(Θ) - видимая освещённая площадь КА, зависящая от фазового угла Θ;

ρ - коэффициент отражения поверхности объекта;

τ*а* - коэффициент пропускания атмосферы;

RКО - расстояние от оптико-электронного средства до КО.

Если КО имеет сложную форму, то создаваемая им освещенность будет зависеть от величины видимых освещенных площадей всех составных частей его конструкции и соответствующих им коэффициентов отражения.

Единицей измерения визуального блеска КО в спутниковой астрономии является звездная величина, которая рассчитывается на входе оптической системы на основании измеренного значения создаваемой КО освещенности по формуле:



Однако, в результате движения КО по орбите и относительно центра масс, а также суточного и годового движения Солнца, блеск КО будет изменяться. В этом случае, при его фотометрировании формируется фотометрическая кривая или функция блеска, получаемая на основе первичной обработки измерений единичных значений блеска КО в заданный интервал времени с заданным периодом дискретизации.

В процессе фотометрирования реальные сигналы претерпевают искажения, обусловленные как факторами детерминированного характера, допускающие их компенсацию (изменение уровня сигнала, влияние нелинейностей тракта регистрации), так и факторами, имеющими стохастическую природу (шумы тракта, фотоприемника и т.д.) [13].

Обработка фотометрических сигналов (ФМС) может состоять в опреде-лении фотометрических признаков КО и включать два этапа (рисунок 1):

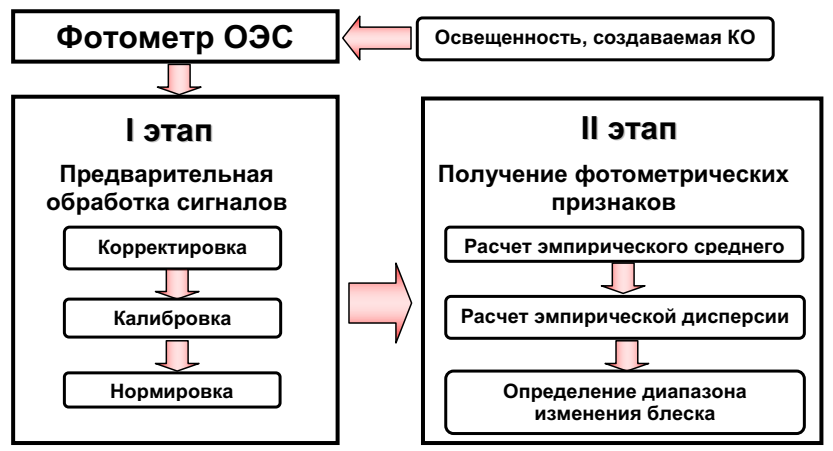


Рис. 1.1 – Последовательность и содержание этапов первичной и вторичной обработки измеренных значений фотометрических сигналов

1. Предварительную обработку ФМС (исключение детерминированных и стохастических факторов).

2. Определение фотометрических признаков, обеспечивающее расчет статистических характеристик ФМС.

Предварительная обработка осуществляется на первом этапе и включает корректировку, калибровку и нормировку измеренных значений ФМС [2, 3].

Корректировка измеренных значений ФМС заключается в исключении влияния срывов измерения, попадания в поле зрения ярких звезд и т.д.

Калибровка состоит в минимизации искажений сигналов за счет нелинейности фотометра и прозрачности атмосферы.

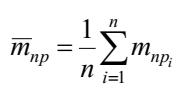
Нормировка обеспечивает приведение сигнала к стандартным условиям наблюдения (например, заданные дальность наблюдения, зенитный угол, фазовый угол освещения) и нормирование сигнала по собственному максимальному значению. Приведение фотометрического сигнала к стандартным условиям наблюдения предполагает устранение зависимости блеска от расстояния до КО, состояния атмосферы во время наблюдения и фазового угла, определяемого взаимным положением Солнца, КО и ОЭС.

Нормированные и фильтрованные сигналы, полученные на этапе предварительной обработки, являются исходной информацией для получения фотометрических признаков (второго этапа).

Второй этап обработки фотометрической кривой предназначен для получения признаков КО, которые могут использоваться при формировании базы данных при мониторинге. В качестве признаков выберем следующие статистические характеристики:

* среднее значение блеска, среднеквадратическое отклонение изменения блеска,
* максимальное и минимальное значения блеска, период изменения блеска.

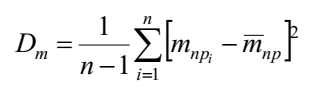
Представив функцию блеска *m(t)* рядом дискретных значений ординат с постоянным шагом *dt* и усредняя значения этой реализации вдоль оси абсцисс по времени, получим оценку математического ожидания случайной функции блеска:



где *n* - число дискретных значений ординат.

Она является интегральной характеристикой величины отражающей поверхности и может характеризовать средний размер КО.

Усредняя квадраты отклонений функции от этого среднего, получим приближенное значение дисперсии:



Дисперсия блеска *Dm* является характеристикой разброса амплитуды отраженного сигнала и определяет изменение блеска КО в зависимости от его конфигурации и возможных ракурсов относительно точки наблюдения и положения Солнца.

Важным информационным признаком КО служит диапазон изменения блеска на интервале времени фотометрирования. Например, резкое изменение блеска за время фотометрирования может свидетельствовать о том, что объект имеет участок поверхности с зеркальным отражением и т. д. Диапазон изменения блеска КО определяется по формуле:



Величина периода функции блеска может свидетельствовать о характере движения КО относительно собственного центра масс.

Фотометрические признаки КО, как наиболее информативные, используются в качестве индивидуальных различительных признаков КО при формировании базы данных по результатам проведенных наблюдений [13].

## 1.3 Проведение оптических наблюдений и обработки измерений объектов космического мусора

Оценка качества измеренных ОЭС значений блеска КО может быть проведена с

* помощью методики, включающей следующие четыре этапа:
* построение по измеренным значениям ФМС кривой блеска;
* приведение кривой блеска к стандартному виду;
* графический анализ измеренных и приведенных значений блеска;
* статистический анализ результатов измерений.

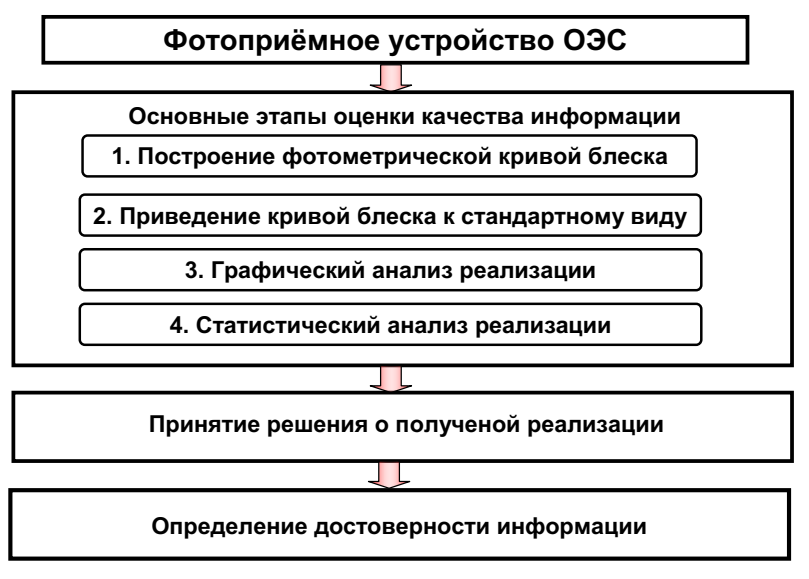


Рис. 1.2 – Возможная последовательность оценки качества ФМР

## 1.4 Параметры движения и физических характеристик объектов космического мусора в области ГСО

Результатами обработки ПЗС-изображений для каждого объекта наблюдений является набор измеренных топоцентрических координат - прямого восхождения и склонения объекта (в стандартной небесной системе координат) и оценка блеска в интегральных звёздных величинах на момент времени наблюдений [11]:

* большая полуось,
* эксцентриситет орбиты,
* угол наклонения,
* долгота восходящего узла,
* количество измеренных положений,
* средняя квадратическая погрешность,
* оценка отношения площади миделева сечения к массе.

Пример изменения блеска космического объекта представлен на рисунке 1.2.

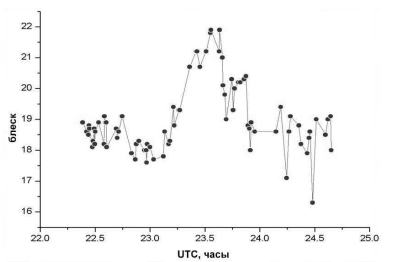


Рис. 1.3

Пример изменения блеска и фазы космического объекта представлен на рисунке 1.4.

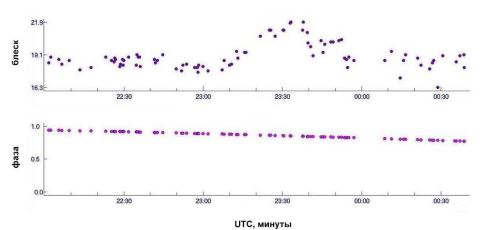


Рис. 1.4

Пример значений отклонения прямого восхождения и склонения объекта:

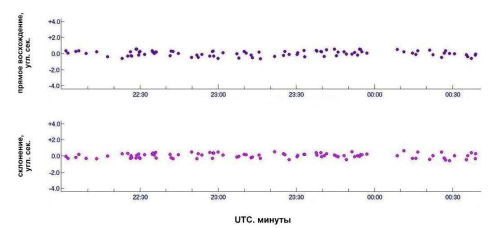
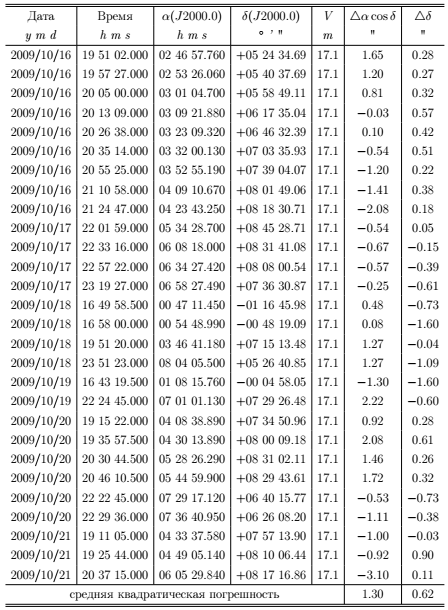


Рис. 1.5

Таблица 1.2 – Фрагмент каталога положений объекта



# 2 Методы и алгоритмы определения типа покрытия космических объектов

## 2.1 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.

## 2.2. Расчет компонент вектора Стокса

Вектор Стокса может быть представлен в виде нормального восходящего телесного конуса[17]. Параметры Стокса могут быть представлены как функции азимутального угла, угла эллиптичности и разности фаз. 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 2.1 – Алгоритм расчета компонент вектора Стокса

## 2.3 Вычисление параметров Стокса падающего излучения

Поскольку естественное солнечное излучение имеет нулевую поляризацию, вектор Стокса для такого излучения имеет параметры  [16, 17]. 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– Алгоритм расчета компонент вектора Стокса   
естественного излучения

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

Рассмотрим алгоритм нахождения дополнительных параметров поляризации рассеянного светового потока [18]:

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: .



Рис. 2.3 – Алгоритм расчета дополнительных параметров   
численными методами

## 2.5 Определение типа покрытия космических объектов

Рассмотрим алгоритмы определения параметров поверхностей космических тел, в том числе и объекты техногенного происхождения. Под параметрами поверхностей понимаются показатели преломления, состояние материала, тип покрытия [17, 18, 19].

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.

*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.

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).

## 2.6. Алгоритм вычисления показателя преломления однородной диэлектрической поверхностью

В [20] 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.



Рис. 2.4– Алгоритм определения коэффициента преломления   
неметаллической поверхности

## 2.7. 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* очках 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.



Рис. 2.5 – Алгоритм определения коэффициента преломления металлических покрытий (часть 1)



Рис. 2.6 – Алгоритм определения коэффициента преломления металлических покрытий (часть 2)

# 2 Информационная система регистрации оптических параметров космических объектов STOKES-QT

## 2.1 Этапы автоматизации регистрации и расчета оптических параметров космических объектов

В настоящее время развитие информационных технологий привело к появлению методов и средств, обеспечивающих интегрированные решения по автоматизации различных процессов, позволяющие автоматизировать ручные операции и поиск документов и данных.

В данной дипломной работе предлагается автоматизировать процесс регистрации и расчета оптических параметром космических объектов.

Предлагается осуществить три уровня автоматизации:

1. Использование *MS Office* и электронных таблиц.

2. Использование локальной версии системы для автоматизации расчетных задач с интерфейсом командной строки.

3. Использование локальной версии системы для автоматизации расчетных задач с графическим интерфейсом пользователя.

4. Использование сетевой версии системы для автоматизации расчетных задач.

### 2.1.1 Уровень автоматизации I

На данном уровне автоматизации предлагается перейти на использование ПЭВМ, и применяя пакет *MS Office* и электронные таблицы и осуществлять процесс расчета и регистрации оптических параметров космических объектов.

Преимущества данного подхода:

* + более высокая оперативность;
  + меньшая трудоемкость обработки информации;
  + более совершенная организация сбора и регистрации исходной информации;
  + возможность оперативного пересчета смет с измененным коэффициентом пересчета с помощью применения макросов;
  + гибкие средства хранения и обработки информации;
  + гибкие средства обмена информацией.

### 2.1.2 Уровень автоматизации II

На данном уровне автоматизации предприятию предлагается перейти на использование ПЭВМ, и применять системы решения расчетных задач используя интерфейс командной строки.

Преимущества данного подхода:

* + более совершенная организация сбора и регистрации исходной информации;
  + возможность оперативного пересчета данных с измененными параметрами расчета;
  + более высокое качество представления документации в печатном виде;
  + гибкие средства обмена информации.

### 2.1.3 Уровень автоматизации III

На данном уровне автоматизации предприятию предлагается перейти на использование ПЭВМ, и применяя версию информационной системы с графическим интерфейсом пользователя.

Преимущества данного подхода:

* + централизованное хранение и управление данными;
  + более быстрый доступ к информации;
  + более высокая оперативность, повышающая качество составления документации;
  + использование понятного пользовательского интерфейса на основе спецификации *GUI Windows*;
  + меньшая трудоемкость обработки информации;
  + более совершенная организация сбора и регистрации исходной информации;
  + более высокое качество представления документации в печатном виде;
  + гибкие средства хранения и обработки информации;
  + гибкие средства обмена информации.

В рамках данной дипломной работы будет реализован III уровень автоматизации.

## 2.2 Определение требований к информационной системе

### 2.2.1 Создание функционального описания

На этапе создания функционального описания необходимо сформулировать принципы, по которым происходит взаимодействие системы с внешним миром. Следует заметить, что создание функционального описания является одним из наиболее критических этапов создания системы и определения пределов ее возможностей.

На основе функционального описания системы в дальнейшем базируются последующие действия, где одним из важнейших является системный анализ.

При создании функционального описания необходимо рассмотреть несколько аспектов, таких как область рассмотрения, аппаратные средства, методы ввода и вывода, возможные пользователи и некоторые другие вопросы.

### 2.2.2 Программные средства

Клиентская часть разрабатываемой программной системы должна работать в операционной системе *Windows*. В качестве среды разработки будет использован язык программирования C++ в среде *Qt*. В качестве сервера баз данных должен использоваться *PostgresQL*

### 2.2.3 Метод ввода

Для ввода информации будет использоваться интерактивный пользовательский интерфейс. По визуальному представлению и принципу использования он представляет собой стандартный интерфейс OC *Windows*. Привычные элементы управления (панели, окна, кнопки) позволят будущему пользователю быстро адаптироваться и научиться работать с программной системой.

### 2.2.4 Метод вывода

Вывод информации будет производиться на экран с помощью интерактивного пользовательского интерфейса, а так же на принтер при необходимости, в текстовый файлы протоколов расчета.

## 2.3 Обоснование выбора средств разработки программной системы

### 2.3.1 Язык С++

C++ – компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

С++ является компилируемым языком, а это значит, что компилятор транслирует исходный код на С++ в исполняемый файл, который содержит набор машинных инструкций. Но разные платформы имеют свои особенности, поэтому скомпилированные программы нельзя просто перенести с одной платформы на другую и там уже запустить. Однако на уровне исходного кода программы на С++ по большей степени обладают переносимостью, если не используются какие-то специфичные для текущей ос функции. А наличие компиляторов, библиотек и инструментов разработки почти под все распространенные платформы позволяет компилировать один и тот же исходный код на С++ в приложения под эти платформы.

Объектно-ориентрованный подход к проектированию программных систем является элементом наукоемких технологий проектирования программ. Использование этого подхода даёт возможность на порядок по сравнению с обычным директивным программированием сократить трудоёмкость отладки программ и внесения изменений в программу при её последующем использовании.

Современные языки программирования, такие как С++, предоставляют в распоряжение программиста обширный арсенал инструментальных средств, позволяющий проектировать мощные и гибкие программы. В объектно-ориентированных программах все данные разбиваются на отдельные группы и строго связываются с функциями, предназначенными для обработки этих данных.

В связи с чем, отметим ряд его достоинств C++:

* он позволяет уменьшить сложность программного обеспечения;
* повысить надежность программного обеспечения;
* обеспечить возможности модификации отдельных компонентов программного обеспечения без изменения остальных его компонентов;
* обеспечить возможности повторного использования отдельных компонентов программного обеспечения. MS Visual C++ 6.0 обладает мощной библиотекой визуальных и не визуальных компонент, что облегчает разработку пользовательского интерфейса и программ в целом;
* язык C++ по-прежнему остается лучшим языком для программирования.

### 2.3.2 Среда разработки Qt

Qt — фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++.

Qt позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем путём простой компиляции программы для каждой системы без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения: классы элементов графического интерфейса, для работы с сетью, базами данных и XML, многопоточного и параллельного программирования. Является полностью объектно-ориентированным, расширяемым и поддерживающим технику компонентного программирования.

Qt широко использует наследование, особенно в модуле Widgets. QObject – это базовый класс. Он предоставляет некоторые очень мощные возможности:

* Можно установить имя как строку для поиска объектов.
* Родительскую систему.
* Сигналы и слоты.
* Управление событиями.

Любой объект, который наследует QObject, может иметь родительские и дочерние элементы. Это дерево иерархии делает работу удобной:

Когда объект уничтожается, все его родительские связи также убираются. Таким образом, вызов delete становится необязательным.

Дочерние виджеты в QWidget автоматически появляются внутри родительского.

Почти все инструменты пользовательского интерфейса имеют механизм для обнаружения действия и реагируют на него.

Сигналы и слоты используются для связи между объектами. Механизм сигналов и слотов является центральной особенностью Qt и, вероятно, той частью, которая больше всего отличается от функций, предоставляемых другими фреймворками. Сигналы и слоты становятся возможными благодаря системе метаобъектов Qt.

Сигнал испускается, когда происходит определенное событие. Виджеты Qt имеют много предопределенных сигналов, но можно создать собственный класс виджетов, чтобы добавить к ним собственные сигналы. Слот - это функция, которая вызывается в ответ на определенный сигнал. Виджеты Qt имеют много предопределенных слотов, но общепринятой практикой является подкласс виджетов и добавление собственных слотов, чтобы вы могли обрабатывать сигналы, которые вас интересуют.

### 2.3.3 База данных PostgresQL

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД).

Разработчики приложений получают в свое распоряжение богатый инструментарий, позволяющий реализовать приложения любого типа:

Возможность использования различных языков для серверного программирования, в том числе встроенного PL/pgSQL (удобного тесной интеграцией с SQL), C для критичных по производительности задач, Perl, Python, Tcl, а также Javascript, Java и другие;

Программные интерфейсы для обращения к СУБД из приложений на любом языке, включая стандартные интерфейсы ODBC и JDBC;

Набор объектов баз данных, позволяющий эффективно реализовать логику любой сложности на стороне сервера: таблицы и индексы, ограничения целостности, представления и материализованные представления, последовательности, секционирование, подзапросы и with-запросы (в том числе рекурсивные), агрегатные и оконные функции, хранимые функции, триггеры и т. д.;

Встроенная гибкая система полнотекстового поиска с поддержкой русского и всех европейских языков, дополненная специально разработанным эффективным индексным доступом;

Поддержка слабоструктурированных данных в духе NoSQL: хранилище пар «ключ-значение» hstore, xml, json (как в текстовом, так и в эффективном двоичном представлении jsonb);

Подключение внешних источников данных, включая все основные СУБД, в качестве таблиц с возможностью их полноценного использования, в том числе для записи и распределенного выполнения запросов (Foreign Data Wrappers).

PostgreSQL работает на операционных системах семейства Unix, включая серверные и клиентские разновидности Linux, FreeBSD, Solaris, Mac OS X, а также на Windows. За счет открытого и переносимого кода на языке C PostgreSQL можно собрать на самых разных платформах, даже если для них отсутствует поддерживаемая сообществом сборка.

Расширяемость — одно из фундаментальных преимуществ системы, лежащее в основе архитектуры PostgreSQL. Пользователи могут самостоятельно, не меняя базовый код системы, добавлять: типы данных, функции и операторы для работы с новыми типами, индексные методы доступа, языки серверного программирования, подключения к внешним источникам (Foreign Data Wrappers), загружаемые расширения.

### 2.3.4 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

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: *Qt*;
* язык программирования: *C++*;
* формат базы данных: *PostgreSQL*.

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



Рис. 3.1 — Связь программных модулей программной системы для вычисления поляризационных характеристик светового потока и расчета оптических параметров покрытий

## 3.3 Описание основных классов программы

В данном разделе описаны основные классы, используемые в проекте. Тексты основных методов данных классов приведены в ***приложении A***. Основные классы проекта:

Класс **Calc12Window** отвечает за отображение и работу с данными задачи расчета излучения. Методы класса:

* void openTable() – получает таблицу с данными из базы и отображает ее.
* void makeCalculations() – выполняет вычисления по всем строкам. Результат записывается в таблицу.
* void saveData() – записывает данные из таблицы в базу.
* void loadData() – перезагружает данные из базы.
* void calculateOne() – выполняет вычисление выделенной строки и записывает результат в таблицу.
* void displayTauPhi() – отвечает за отображение столбцов tau1..tau4 phi1..phi4.
* void displayIntensity() – отвечает за отображение столбцов i1..i4.
* void displayNatRays() – отвечает за отображение столбцов j0 q0 u0 v0.
* void displayDispRays() – отвечает за отображение столбцов j q u v.
* void displayPolar() – отвечает за отображение столбцов alfa1 beta1 re\_hi im\_hi.
* void dropCalculation() – обнуляет результаты расчета.
* void setupToolbar() – создает тулбар окна с кнопками сохранить загрузить.

Класс **Calc3Window** отвечает за отображение и работу данными задачи определения материалов покрытия. Методы:

* void calcC1() – вычисление значений параметра с1.
* void calcC2() – вычисление значений параметра с2.
* void calcC3() – вычисление значений параметра с3.
* void setupToolbar() – создание тулбара с кнопками сохранить загрузить.
* void c1rbtnClicked() – отображает таблицу с данными для параметра с1.
* void c2rbtnClicked() – отображает таблицу с данными для параметра с2.
* void c3rbtnClicked() – отображает таблицу с данными для параметра с3.
* void calcrbtnClicked() – отображает таблицу со статистикой расчета.
* void calcParamPbtnClicked() – вызов функции расчета параметров С в зависимости от активного окна.
* void calcStats() – расчет статистики по параметрам.
* void loadData() – загрузка данных из базы в таблицу.
* void saveData() – запись данных из таблицы в базу.

Класс **ConstansWindow** отвечает за отображение и работу с константными значениями в расчетах. Методы:

* void setupTable() – загрузка данных из базы и отображение таблицы.

Класс **Database** отвечает за взаимодействие с базой данных. Методы:

* void makeConnection() – создает соединение с сервером базы данных.
* bool openDB() – открывает базу данных. Возвращает true при успехе, false в случае ошибки.
* void closeDB() – закрывает соединение с базой.
* Database \* instance() – возвращает указатель на объект класса. Если объект не создан, то от создается.

Класс **EditExperimentDialog** открывает диалог для редактирования записи об экперименте. Методы:

* void initFieldData(QSqlRecord record) – заполняет поля диалога текущими значениями, полученными из record.
* void save() – сохраняет обновленные данные.

Класс **ExperimentsWindow** отвечает за отображение и работу с данными экспериментов. Методы:

* void experimentSelected(QString uuid, QString name) – оповещает главное окно программы о том, какой эксперимент был выбран, и передает его название и идентификатор.
* void setupToolbar() – создает тулбар окна с кнопками создать удалить сохранить.
* void setupTable() – получает из базы и отображает таблицу с информацией об экспериментах.
* void importRaysData(QString expuuid, QString filename) – считывает исходные данные для расчета параметров излучения из файла filename, записывает их в базу и указывает что они относятся к эксперименту с uuid = expuid.
* void importC1Data(QString expuuid, QString filename) - считывает исходные данные для расчета параметра с1 из файла filename, записывает их в базу и указывает что они относятся к эксперименту с uuid = expuid.
* void importC2Data(QString expuuid, QString filename) - считывает исходные данные для расчета параметра с2 из файла filename, записывает их в базу и указывает что они относятся к эксперименту с uuid = expuid.
* void importC3Data(QString expuuid, QString filename) - считывает исходные данные для расчета параметра с3 из файла filename, записывает их в базу и указывает что они относятся к эксперименту с uuid = expuid.
* void openAddDialog() – открыть диалог создания нового эксперимента.
* void deleteExperiment() – удалить выбранный эксперимент.
* void selectExperiment() – выбрать выделенный эксперимент для работы.
* void importExperimentData(QString expuuid, QString frays, QString fc1, QString fc2, QString fc3) – загрузить данные для созданного эксперимента из файлов frays fc1 fc2 fc3.

Класс **LoadDataDialog** позволяет задать файлы исходных данных для эксперимента. Методы:

* void deleteOldData() – удаляет существующие данные.
* void loadNewData() – считывание из файлов новых данных и запись в базу.
* void selectFileRays() – указать файл импорта данных для расчета параметров излучения.
* void selectFileC1() – указать файл импорта данных для расчета параметров с1.
* void selectFileC2() – указать файл импорта данных для расчета параметров с2.
* void selectFileC3() – указать файл импорта данных для расчета параметров с3.

Класс **LogFileWidget** отвечает за отображение лога вычислений. Методы:

* void openFile() – открывает файл и выводит содержимое на экран.

Класс **Logger** отвечает за ведение файлов лога. Методы:

* void logInfo(QString message) – записывает информационное сообщение message в лог.
* void logDebug(QString message) – записывает отладочное сообщение message в лог.
* void logWarning(QString message) – записывает предупреждение message в лог.
* void logError(QString message) – записывает сообщение об ошибке message в лог.

Класс **MainWindow** отвечает за отображение главного окна программы и взаимодействие с остальными окнами. Методы:

* void openConstants() – отображение окна константных параметров.
* void loadPlot() – отображение окна с графиком.
* void openAllExperiments() – отображение окна экспериментов.
* void setMenuEnabled() – активирует меню при выборе эксперимента для работы.
* void currentExperiment(QString uuid, QString name) – сохраняет uuid выбранного эксперимента и отображает его имя name в заголовке окна.
* void openRefraction() – открывает окно таблицы параметров преломления материалов.
* void makeCalc12() – открывает окно задачи расчета параметров излучения.
* void makeCalc3() – открывает окно задачи расчета материала покрытия.
* void openLoadDialog() – открывает диалог выбора файлов исходных данных для текущего эксперимента.
* void openLogRays() – открыть лог расчета параметров излучения.
* void openLogStats() – открыть лог расчета статистики и определения материала покрытия.

Класс **MaterialRefractionTable** отвечает за отображение таблицы показателей преломления материалов. Методы:

* void displayTable() – загружает данные из базы и отображает таблицу.

Класс **NewExperimentDialog** позволяет создавать новые эксперименты. Методы:

* void save() – сохраняет запись об эксперименте и соответствующие ему исходные данные.
* void selectFileRays() – выбор файла исходных данных для расчета параметров излучения.
* void selectFileC1() – выбор файла исходных данных для расчета параметра с1.
* void selectFileC2() – выбор файла исходных данных для расчета параметра с2.
* void selectFileC3() – выбор файла исходных данных для расчета параметра с3.

Класс **PlotWidget** создает график по рассчитанным данным. Методы:

* void plot(QString argument) – построить график для аргумента argument.
* void setupButtonGroup() – объединение кнопок виджета в группу для более удобного управления их сигналами.
* void replot(QAbstractButton \*button) – построение графика от аргумента, выбранного кнопкой button.

Класс **RowEditingDelegate** отвечает за способ редактирования таблиц. Возможно редактирование не отдельных ячеек, а целой строки таблицы. Методы:

* QWidget \*createEditor(QWidget \*parent, const QStyleOptionViewItem &option, const QModelIndex &index) – создает виджет редактора для редактирования ряда с индексом index, с родительским виджетом parent, опциями отображения option.
* void setEditorData(QWidget \*editor, const QModelIndex &index) – заполняет поля виджета редактирования значениями, расположенными в таблице по индексу index.
* void setModelData(QWidget \*editor, QAbstractItemModel \*model, const QModelIndex &index) – записывает в таблицу по индексу index значения из редактора.
* void commitAndCloseEditor() – подтвердить внесенные изменения и закрыть редактор.

## 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 признаков C1 | Входные данные для задачи 3 по признаку 1 |
| 7 | Данные признаков C2 | Входные данные для задачи 3 по признаку 2 |
| 8 | Данные признаков 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, Iдля 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) |
| 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; |
| 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) |
| 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** | **Первичныйключ** | **Внешнийключ** | **Потенциаль-ныйключ** |
| --- | --- | --- | --- | --- |
| 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 |
|  | 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.

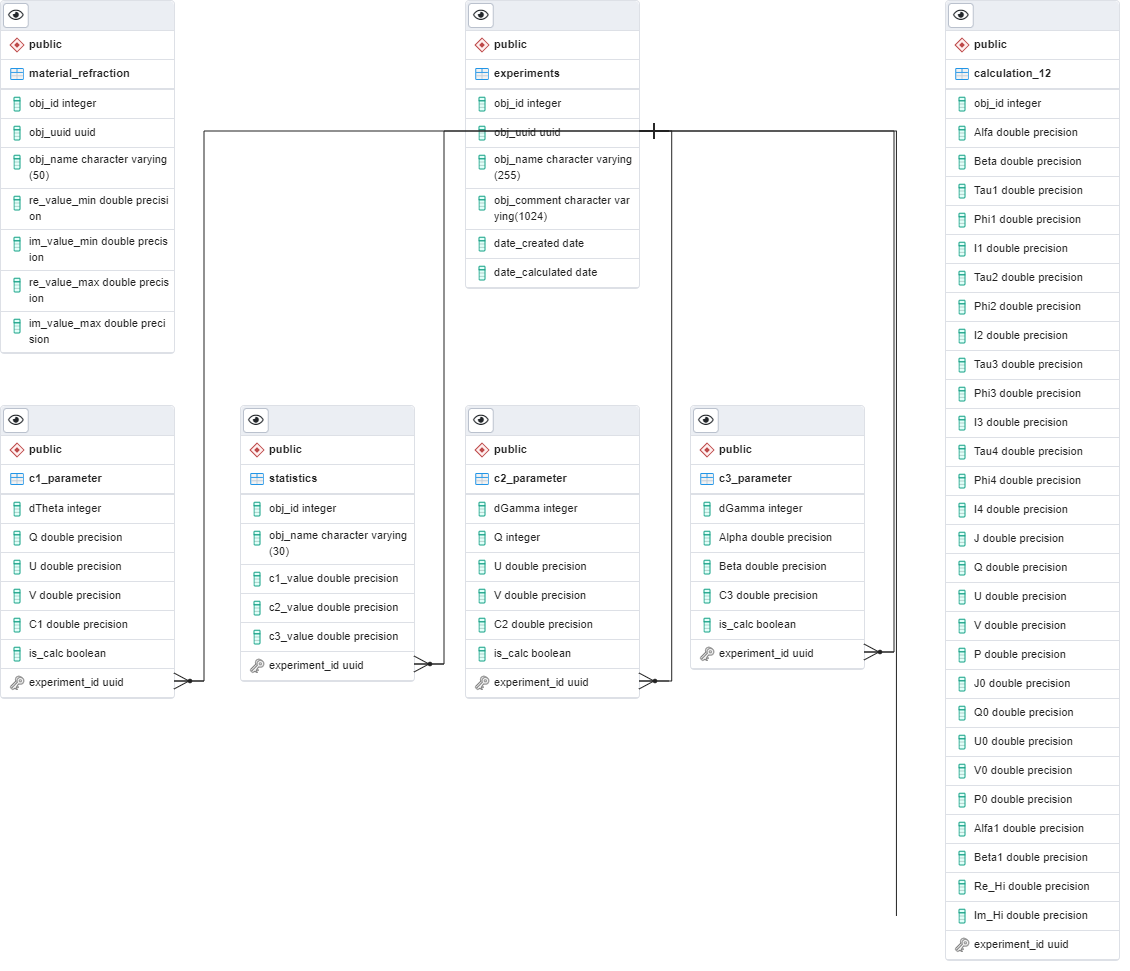


Рисунок 3.3 — Схема базы данных в PostgresQL

### 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 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 (рисунок 3.4).

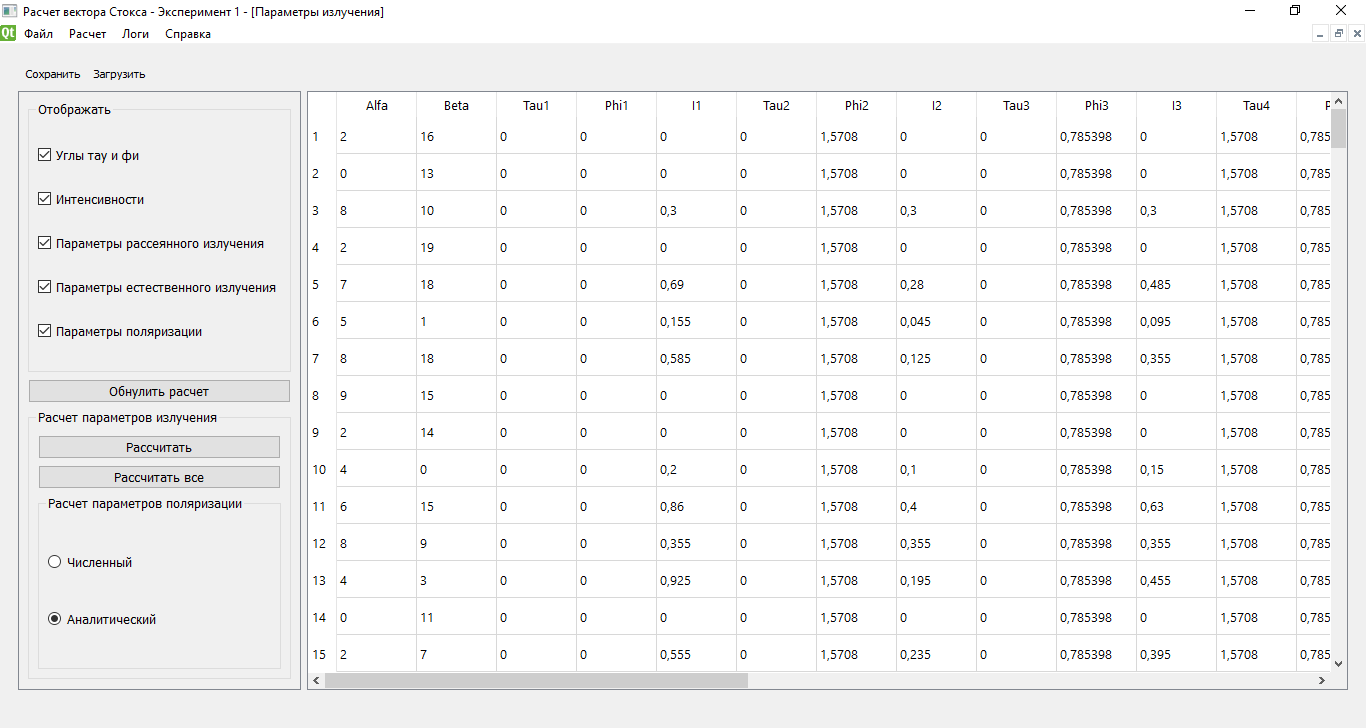


Рис. 3.4 – Тестирование импорта параметров расчета

В качестве исходных данных для тестирования расчета параметров вектора Стокса и расчета параметров рассеянного излучения были использованы данные из [17-20]. 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  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.

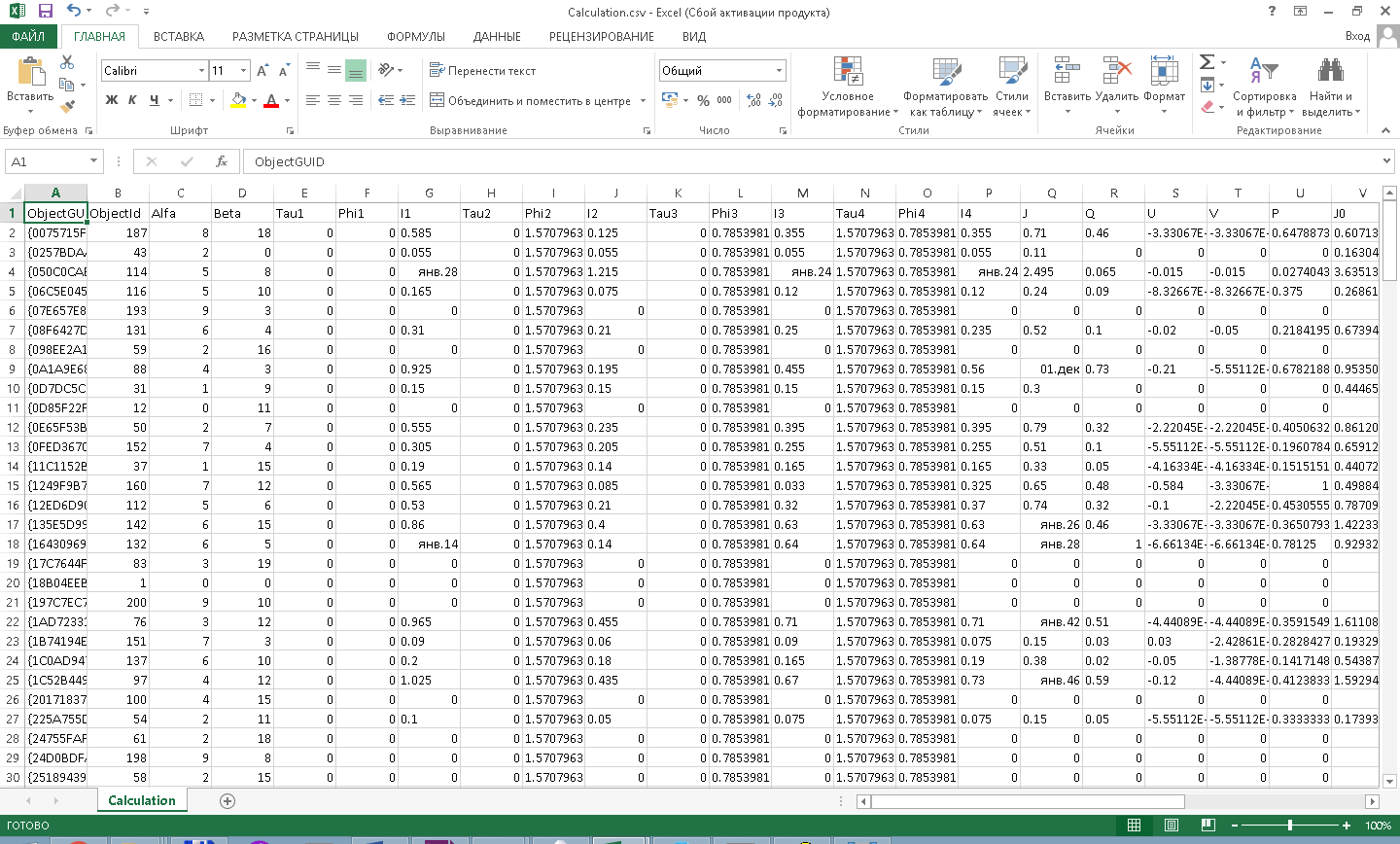


Рис. 3.5 – Тестирование расчета (задачи 1 и 2)

В результате запуска данных задач были получены протоколы расчета (рисунок 3.6, рисунок 3.7, Приложение).

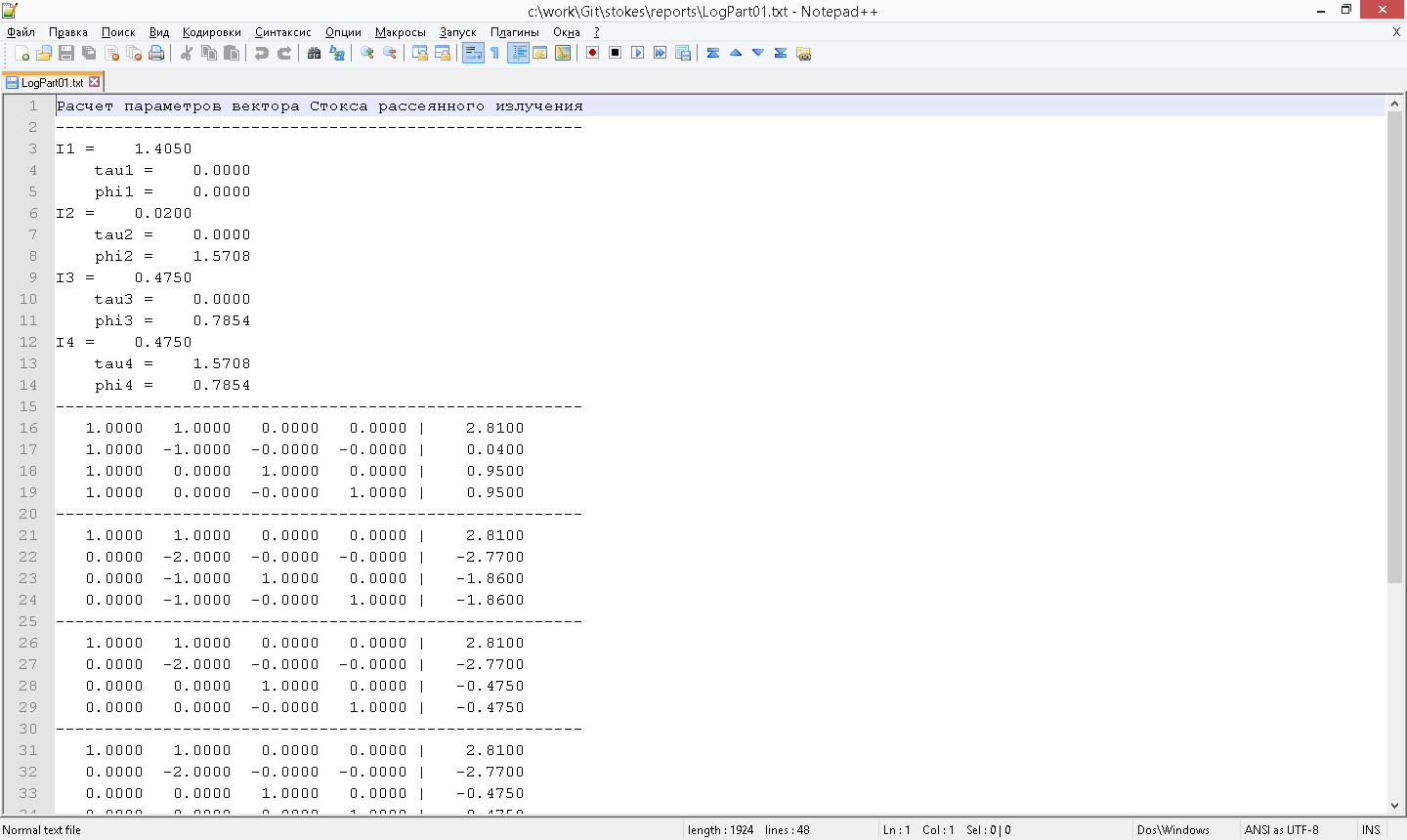


Рис. 3.6 – Протокол расчета (задача 1)

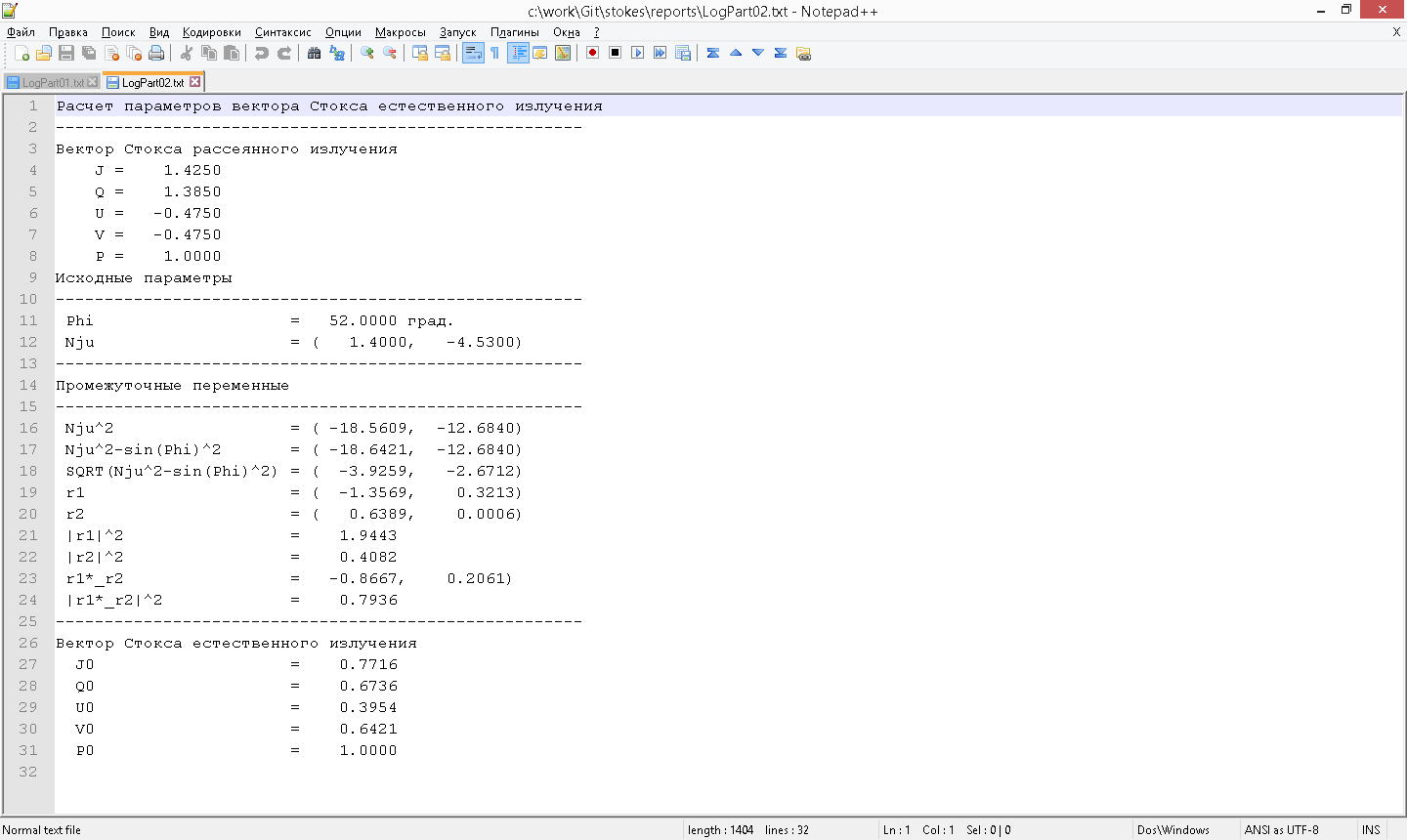


Рис. 3.7 – Протокол расчета (задача 2)

В результате запуска задачи 3 были получены протоколы расчета (рисунок 3.8, рисунок 3.9, Приложение).

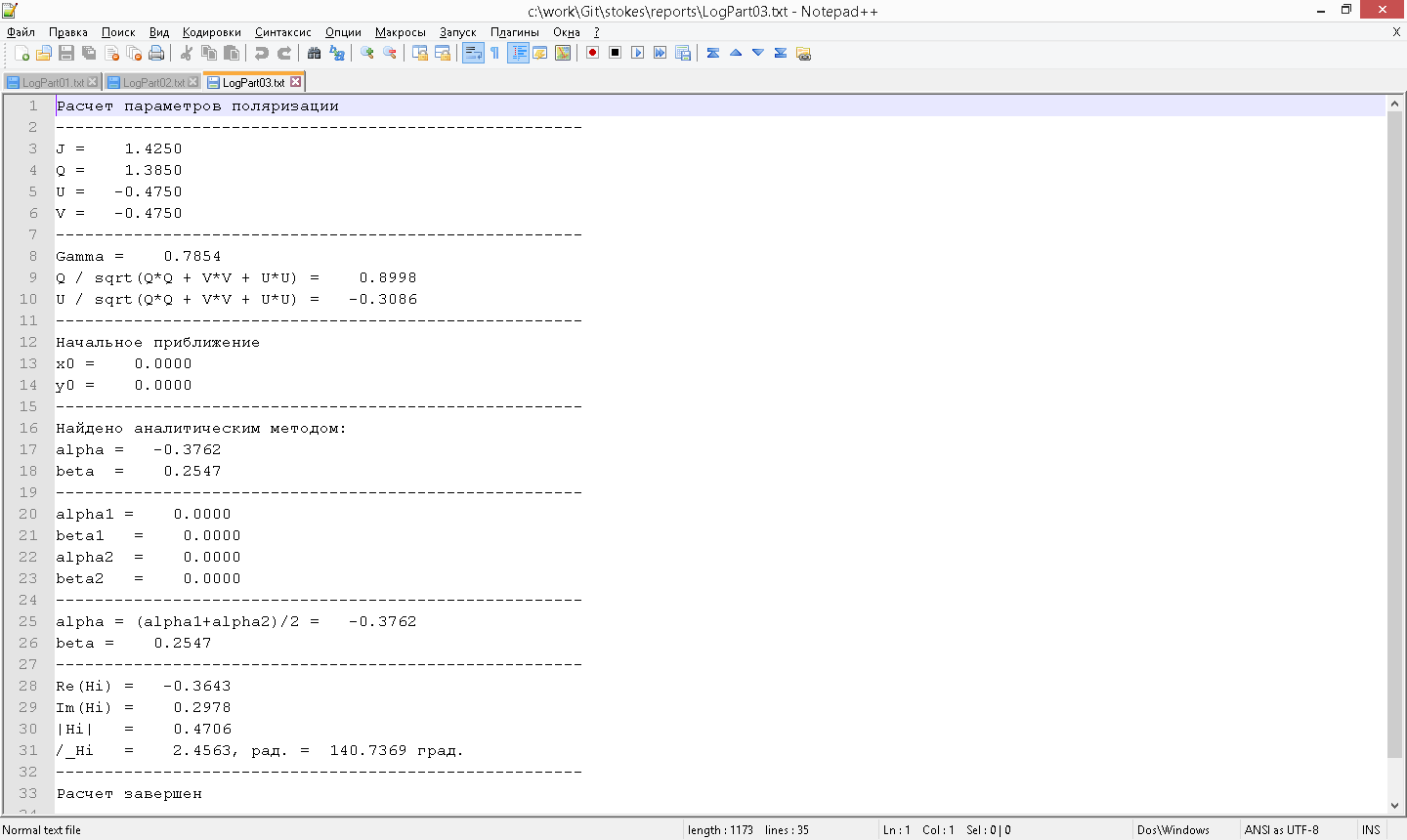


Рис. 3.8 – Протокол расчета 1 (задача 3)

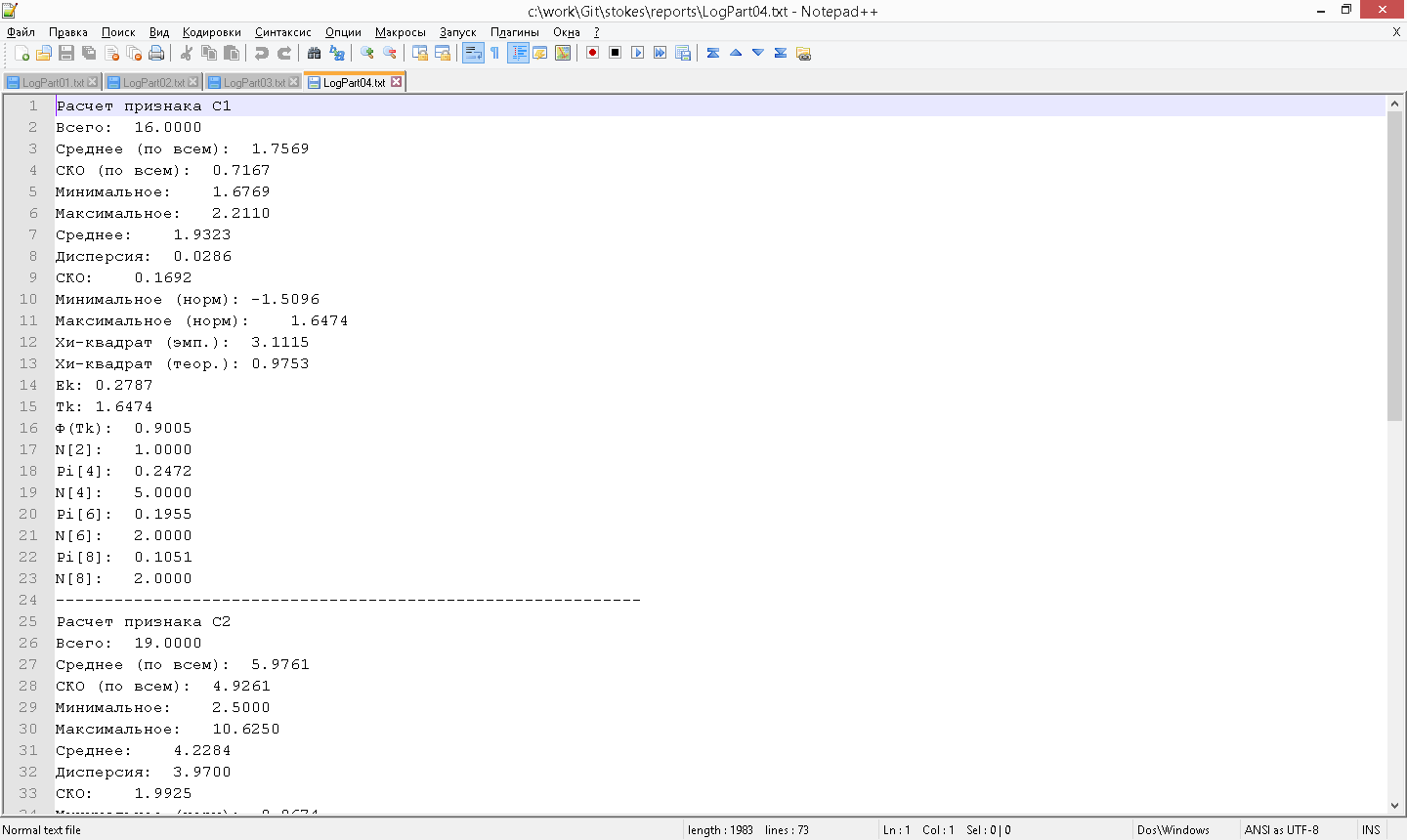


Рис. 3.9 – Протокол расчета 2 (задача 3)

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

1. 3.4. Показатели преломления основных материалов объектов космического мусора в видимой области (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 |

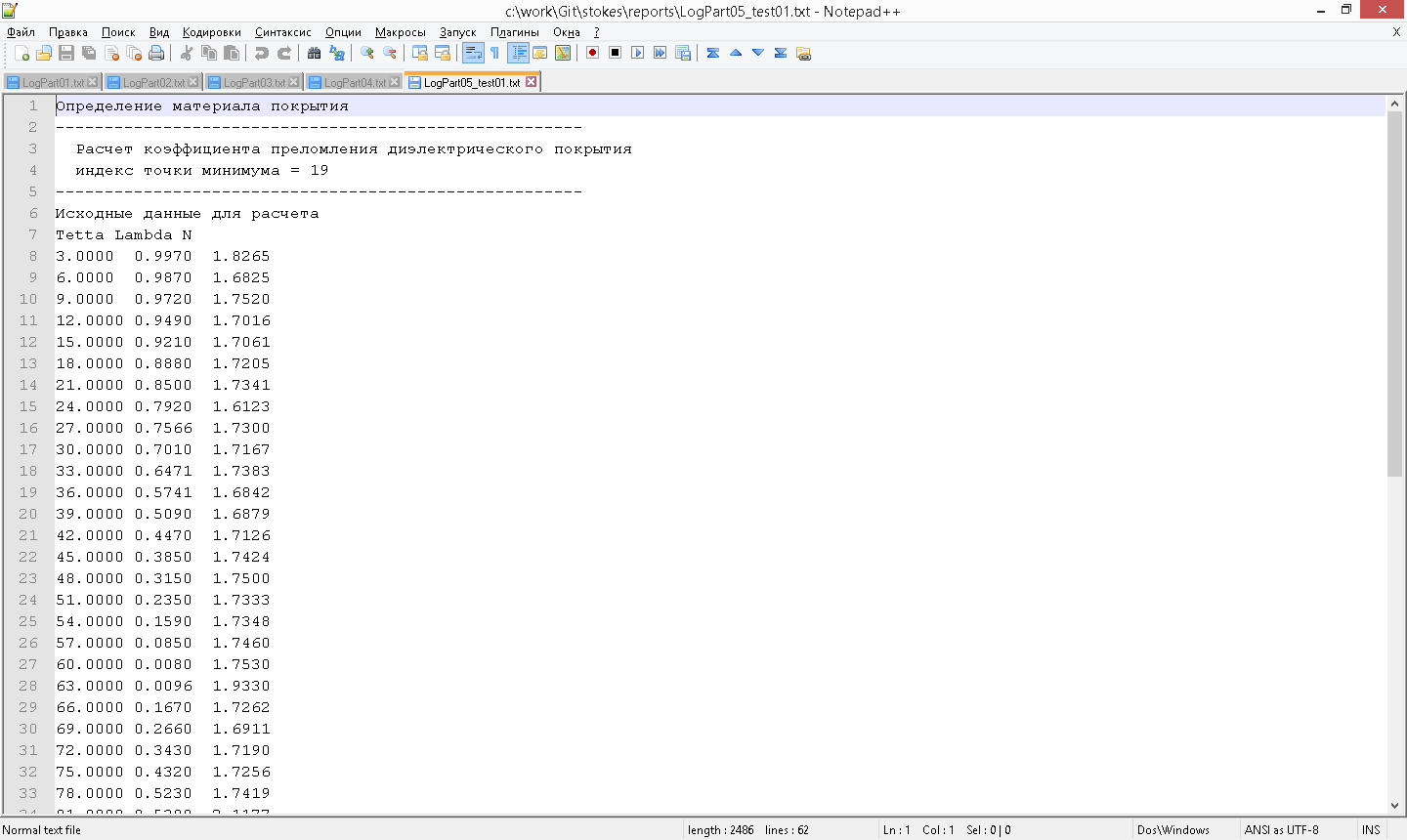


Рис. 3.10 – Протокол расчета 3 (задача 3)

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

# 4 Разработка программной документации

## 4.1 Описание применения

### 4.1.1 Назначение и условия применения информационной системы

Программная система предназначена для решения следующих задач:

* реализация интерфейса пользователя для ввода и обратоки данных, связанных с наблюдениями за космическими объектами;
* реализация интерфейса для расчета параметров Стокса и коэффициента поляризации отраженного светового потока.
* реализация интерфейса для вычисления азимутального угла, угла эллиптичности и комплексного показателя поляризации светового потока;
* реализация интерфейса для расчета показателя преломления и определения материала покрытия;
* сохранение результатов расчета в базу данных;
* формирование отчетов и протоколов расчета.

В состав системы входят:

* модули расчета:
  + модуль расчета параметров вектора Стокса;
  + модуль реализации алгоритма градиентного спуска;
  + модуль вычисления нормированной функции Лапласа;
  + модуль отображения графиков и диаграмм;
  + модуль расчета естественного излучения;
  + модуль расчета рассеянного излучения;
  + модуль расчета поляризации и определения типа покрытия;
* вспомогательные модули
  + модуль работы с базой данных комплекса;
  + модуль импорта данных;
  + модуль формирования протокола расчета;
* модули, реализующие интерфейс пользователя:
  + интерфейс расчета естественного излучения;
  + интерфейс расчета рассеянного излучения;
  + модуль расчета поляризации и определения типа покрытия;
  + интерфейс работы с постоянными параметрами расчета;
  + интерфейс работы с наблюдениями;
  + интерфейс работы с коэффициентами преломления материалов;
  + интерфейс построения графиков и диаграмм;
  + интерфейс главного окна приложения.

### 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*;
* *СУБД PostgreSQL*.

## 4.2 Руководство программиста информационной системы STOKES

### 4.2.1 Характеристики программной системы

Программная система обеспечивает выполнение задачи организации интерфейса пользователя для решения задача ввода и обработки фотометрической информации, выполнения расчетов – определения материала покрытия.

Программная система реализована в виде отдельного приложения с интерфейсом пользователя и взаимодействующая с базой данных.

Программная система работает в интерактивном режиме выполнения оператором необходимых действий посредством работы с использования графического интерфейса пользователя.

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

### 4.2.2 Настройка информационной системы

Для настройки программной системы необходимо выполнить следующие действия:

* создать базу данных в СУБД PostgreSQL с именем stokes\_db;
* настроить права доступа к базе данных для пользователя postgres;
* проверить доступность соединения с базой данных.

### 4.2.3 Обращение к информационной системе

Управление системе передается при ее запуске. Для запуска программы следует вызвать на выполнение приложение *stokes.exe*.

### 4.2.4 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.

## 4.3 Руководство оператора

Настоящий документ является руководством по использованию информационной системы регистрации оптических параметров космических объектов и расчета компонент вектора Стокса рассеянного, естественного излучений и параметров поляризации.

Персональный компьютер пользователя должен работать в операционной системе *Windows*, используя стандартный набор периферийных устройств: клавиатура, мышь.

Системные требования, необходимые для функционирования программной системы:

* *Microsoft Windows.*
* Оперативная память не менее 1 Гб.
* Пространство жесткого диска не менее 150 Мб.
* Монитор с разрешением не менее 1024х768.
* Мышь, клавиатура.

Для работы с информационной системой необходимы быть знакомы следующие понятия и навыки:

* приемы работы с окнами;
* работа с меню;
* использование управляющих элементов диалогов;
* стандартные диалоги;
* работа с программой *Microsoft Excel*.

### 4.4.1 Основные функции системы

Информационная система *Stocks* предназначена для решения следующих задач:

* регистрация наблюдений и параметров оптических объектов;
* расчет параметров рассеянного излучения;
* расчет параметров естественного излучения;
* расчет параметров поляризации.

Исходные данные для расчета могут быть импортированы в систему из файлов формата *csv*, *Microsoft Excel*. После загрузки данные сохраняются в локальной базе данных *PostgreSQL* и при необходимости могут быть изменены пользователем.

Результаты расчета параметров рассеянного излучения, естественного излучения и поляризации могут быть выведены в графическом и табличном видах. Промежуточные этапы процесса расчета сохраняются в текстовом файле.

### 4.4.2 Выполнение программы

Пользователю предлагаются следующие режимы работы с данными:

* форма списка для просмотра сведений обо всех информационных объектах одного типа в виде таблицы (например, список параметров приложения, список интервалов и т.д.);
* форма элемента для просмотра и изменения информации об информационном объекте (например, для редактирования записи параметра приложения).

Для доступа к данным программной системы пользователю предлагается использовать отдельное окно — форму списка, представленную на рисунке:

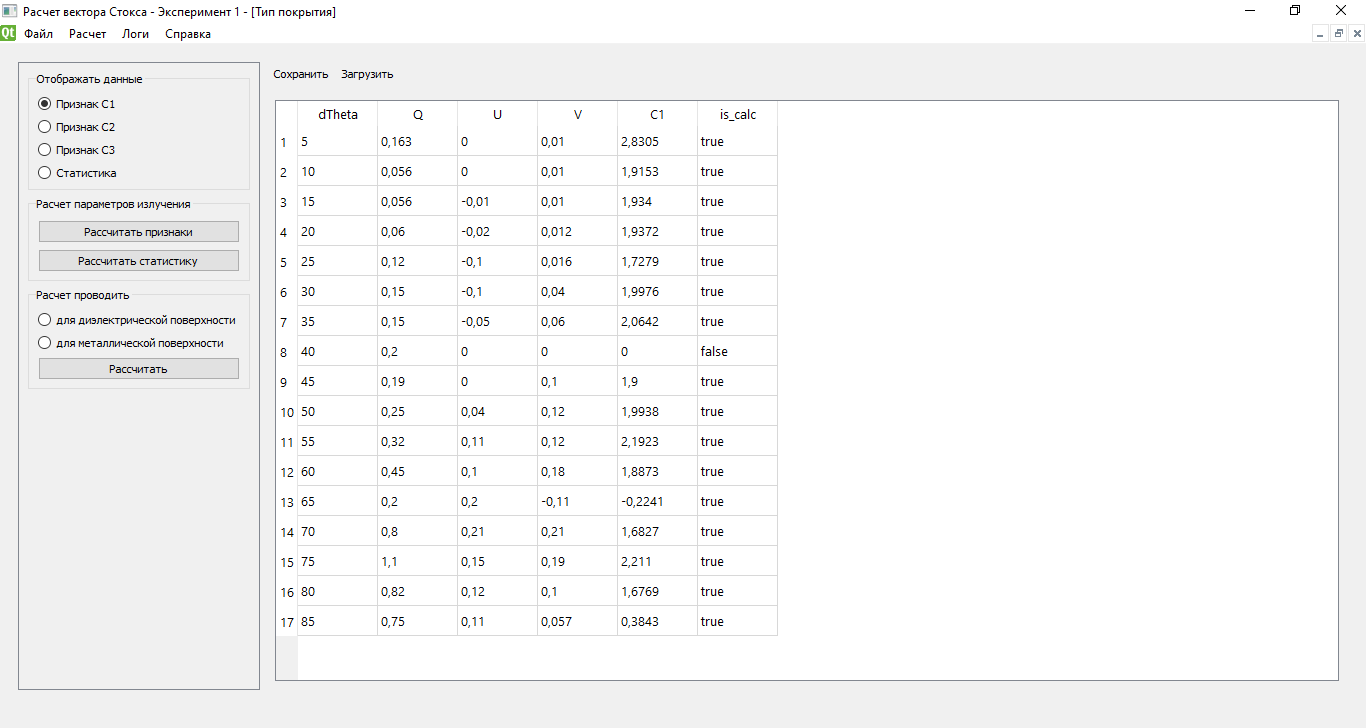


Рис. 4.1 – Пример формы списка

Форма списка состоит из:

* заголовка окна;
* области данных;
* прочих элементов управления.

Заголовок формы списка содержит название окна, системное меню окна и кнопки системного меню. Область данных отображает элементы данных в виде таблицы. Заголовки столбцов таблицы отображают имена атрибутов.

Форма редактирования предназначена для просмотра и редактирования конкретного элемента данных.

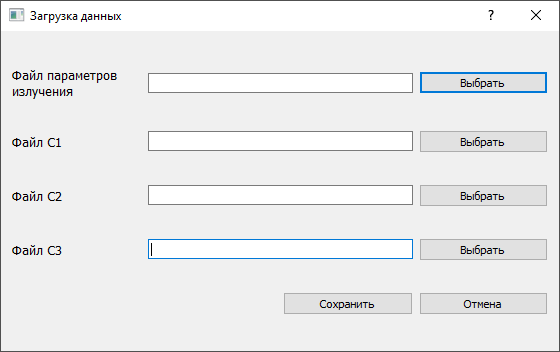


Рис. 4.2 – Пример формы редактирования

Форма редактирования состоит из:

* заголовка;
* рабочей области;
* панели управляющих кнопок.

Рабочая область окна содержит элементы управления, предназначенные для редактирования различных атрибутов элемента. Панель управляющих кнопок содержит кнопки перехода к первому, предыдущему, следующему и последнему элементу данных, а также кнопки:

Запуск программной системы осуществляется при помощи исполняемого файла *stocks.exe*, расположенного в основном каталоге программной системы. После запуска приложения отображается окно главной форма.

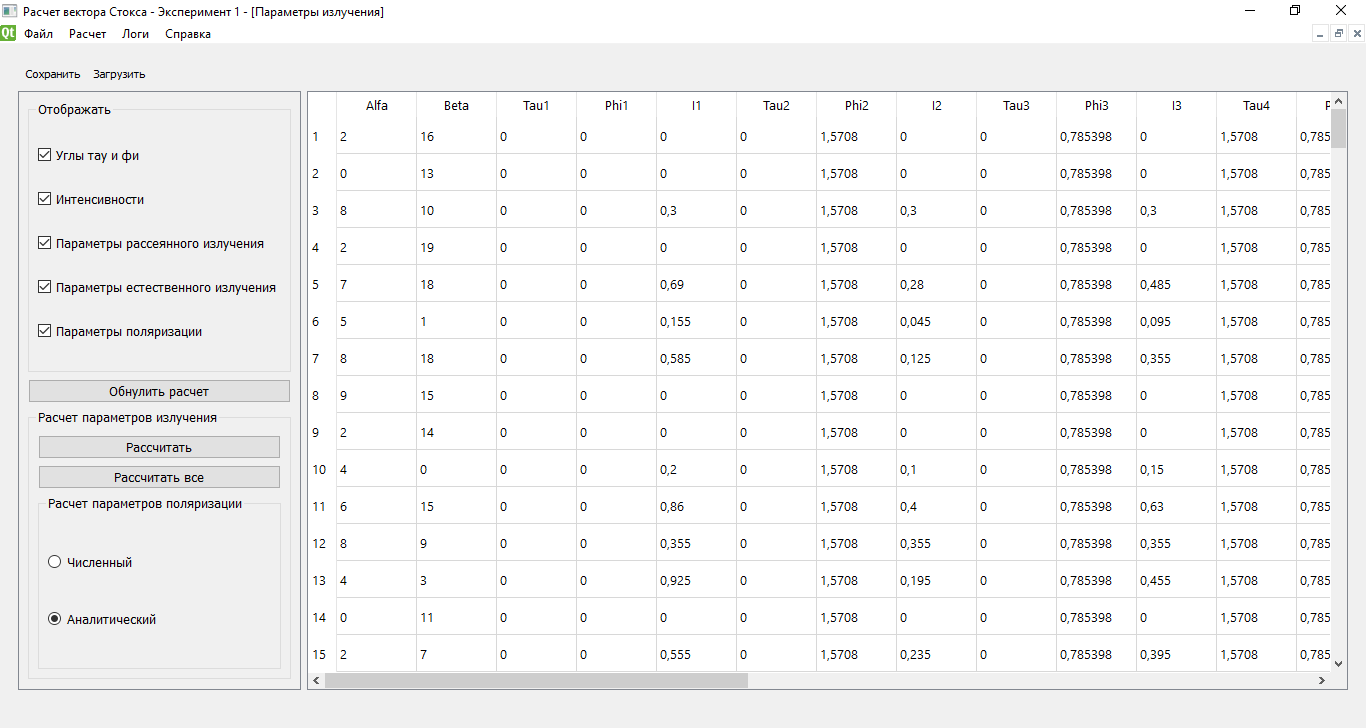


Рис.4.3 – Главная форма

Окно имеет следующие основные элементы:

* заголовок окна;
* текстовое меню;
* рабочая область.

Текстовое меню основного окна представляет собой список задач, которые доступны пользователю.

Для активизации меню нужно щелкнуть левой кнопкой мыши по нужному пункту или нажать клавишу Alt. После нажатия клавиши Alt первый пункт меню будет выделен, для перехода между пунктами меню используются стрелки вправо-влево, для открытия выпадающего списка задач нужно нажать клавишу Enter. Для перемещения в выпадающем списке задач используются клавиши вверх-вниз. Выбор задачи производится нажатием клавиши Enter или щелчком левой кнопки мыши.

Меню содержит следующие пункты:

* Файл — Эксперименты – осуществляет подключение к базе данных программной системы;
* Файл — Выход – происходит закрытие программной системы
* Расчет — Параметры вызывает окно редактирования параметров системы
* Показатели преломления
* Расчет — Загрузить данные вызывает окно импорта данных;
* Расчет — Выполнить расчет вызывает окно расчета параметров;
* Расчет — Определение типа покрытия
* Лог – Протокол расчета параметров поляризации

Для настройки параметров расчета окно «Параметры расчета». При помощи данного окна пользователь может изменять значения следующих параметров:

* угол падения;
* показатель преломления;
* угол γ.

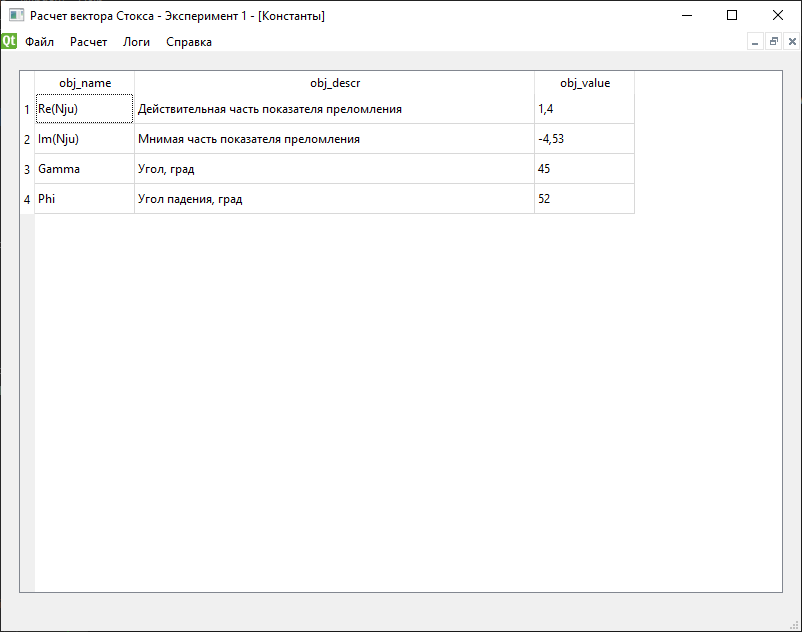


Рис. 4.4 – Форма редактирования параметров

Пользователь не может изменять имена и описание параметров, создавать новые параметры, а также удалять существующие.

Значения параметров показателей предназначены для задачи определения типа покрытия. Список параметров преломления отображается и редактируется при помощи формы, представленной на рисунке 4.5

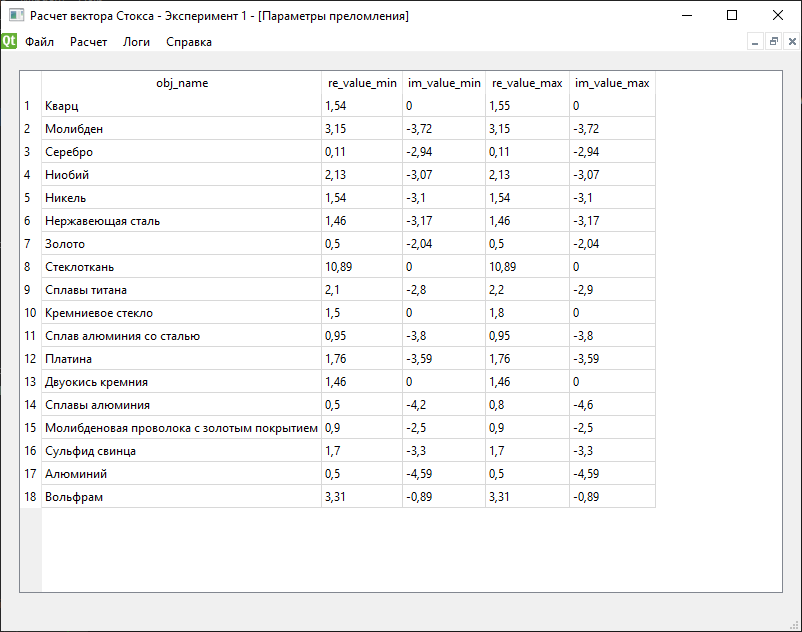


Рис. 4.5 – Показатели преломления материалов

Для загрузки данных из формата *Microsoft Excel* в программную систему используется форма «Импорт данных», представленная на рисунке 4.6.

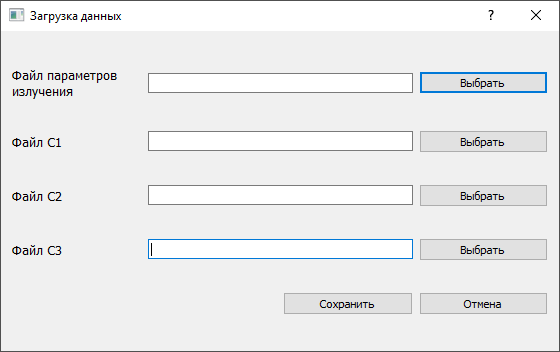


Рис. 4.6 – Форма загрузки данных

Для загрузки данных следует выполнить следующее:

* выбрать файл с данными. После выбора файла данные будут загружены на форму;
* нажать кнопку «Сохранить»»;
* дождаться окончания импорта данных.

Для редактирования наборов данных (исходные данные, статистика, расчетные показатели) используется форма редактирования экспериментов, представленная на рис. 4.7.

С каждым экспериментом связаны данные для расчета задач показателей рассеянного и естественного излучения, показателей поляризации, расчета статистики.

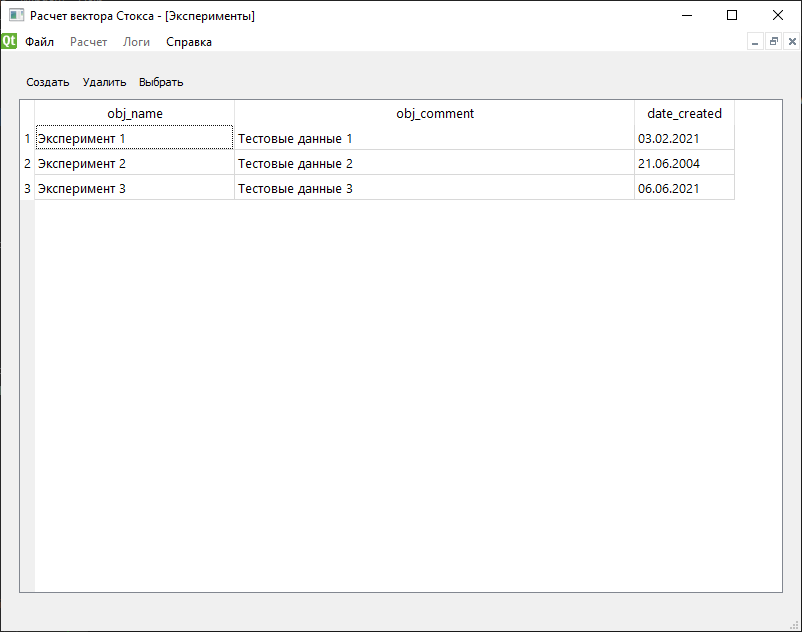


Рис. 4.7 – Форма списка экспериментов

Для выполнения расчетов используется окно «Исходные данные», открываемое командой меню «Расчет — Выполнить расчет». Форма представлена на рисунке 4.8.

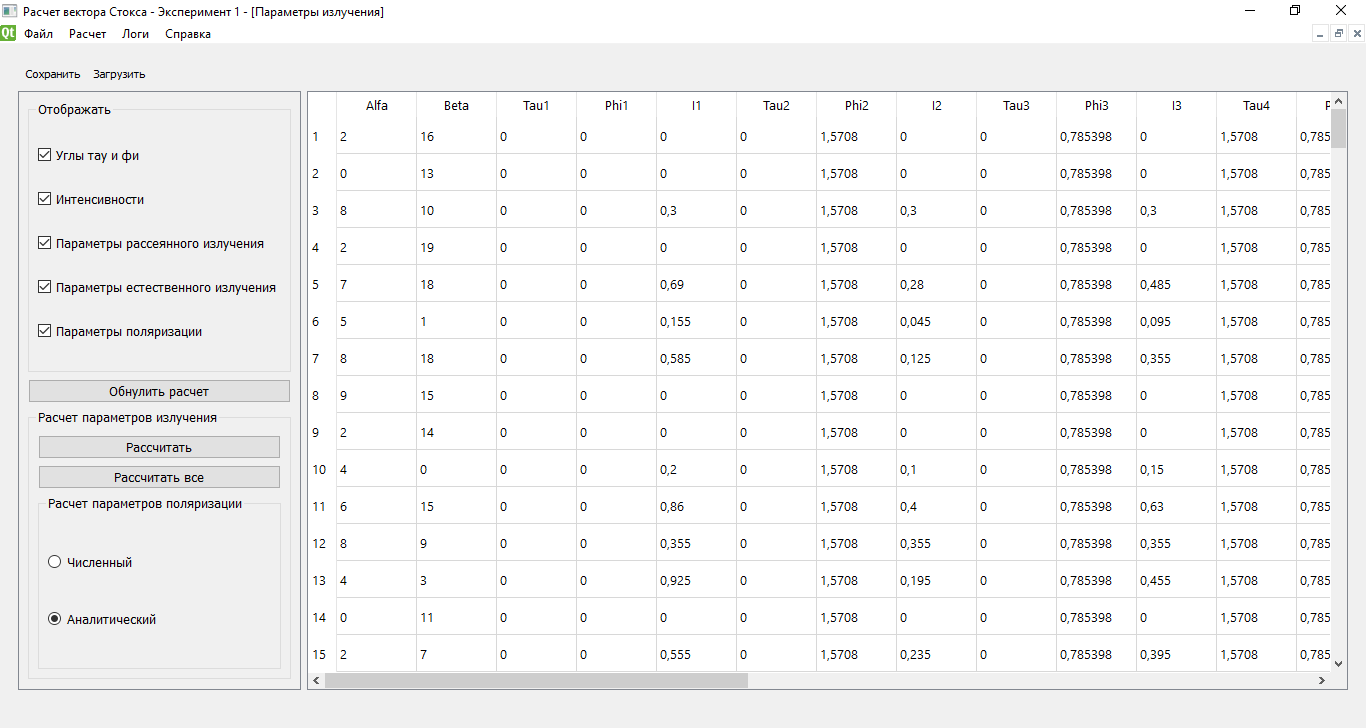


Рис. 4.8 – Расчет параметров излучения

Для просмотра расчета в виде графика используется окно «Рассчитанная поверхность», вызываемая командой меню «Расчет — график». Форма графика представлена на рисунке. Основную часть окна занимает результат расчета, представленный в виде трехмерной поверхности.

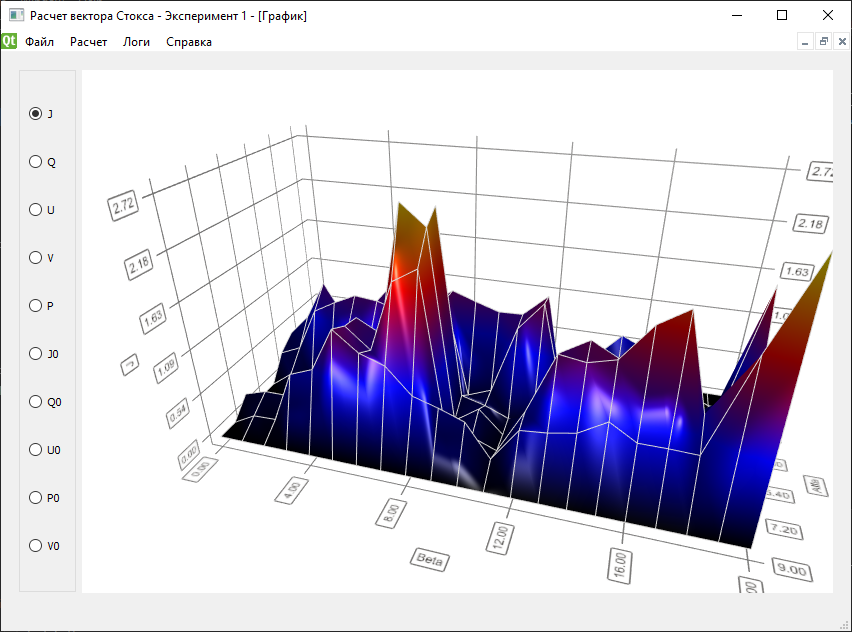


Рис. 4.9 – Пример построения графика

Горизонтальная ось соответствует параметру Alfa в окне расчета, изометрическая ось — параметру Betta. (Из-за особенности вывода графика значения параметров Alfa, Beta должны быть последовательные с шагом 1).

На вертикальной оси выводится значение параметра, выбранного в переключателе «Выбор параметра» на панели инструментов. Доступны для выбора следующие параметры:

* J, Q, U, V, P;
* J0, Q0, U0, V0, P0;
* параметры поляризации (Alfa, Beta);
* значения действительной и мнимой части угла χ.

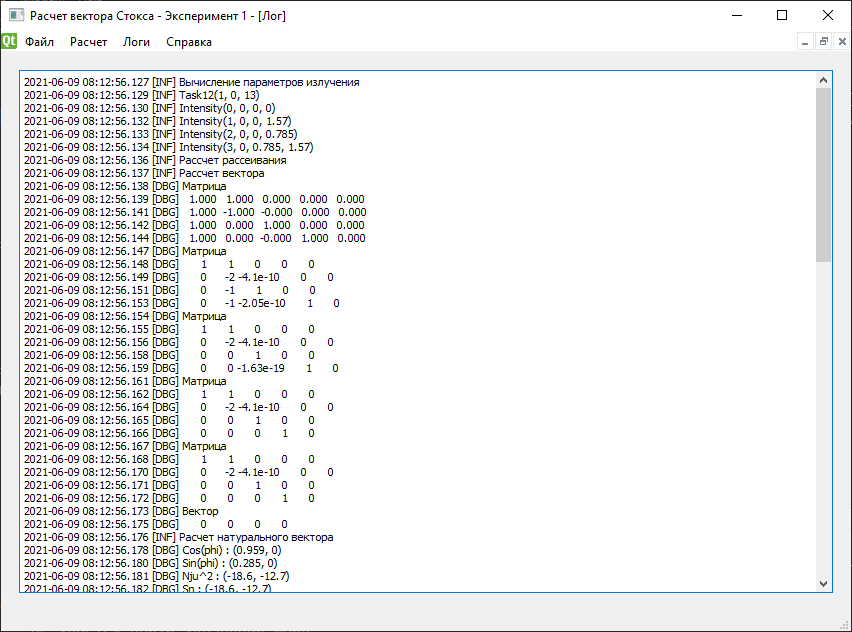


Рис. 4.10 – Пример протокола расчета

Пользователю системы доступны следующие функции вывода информации на печать:

* печать протокола расчета;
* печать расчета рассеянного излучения;
* печать расчета естественного излучения;
* печать параметров поляризации.

Для выполнения классификация объектов по трем признакам необходимо выполнить следующую последовательность действий:

* подготовить исходные данные для расчета;
* выполнить расчет признаков;
* выполнить расчет статистики.

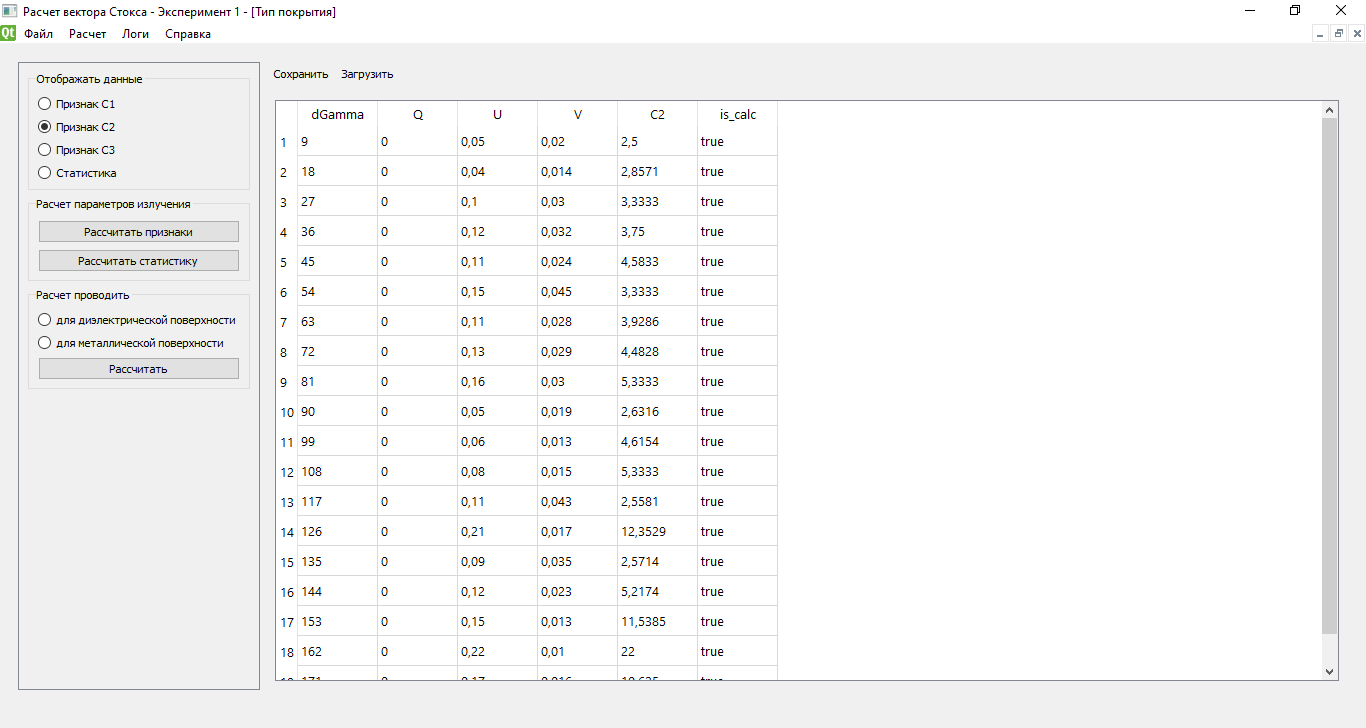


Рис. 4.11 – Окно расчета статистики

Исходные данные для классификации задаются при помощи файлов *Microsoft Excel*.

Для расчета признаков необходимо открыть окно «Определение типа покрытия».

В левой части окна содержаться следующие элементы управления:

* переключатель для выбора отображаемых данных;
* кнопка расчета признаков;
* кнопка расчета статистики.

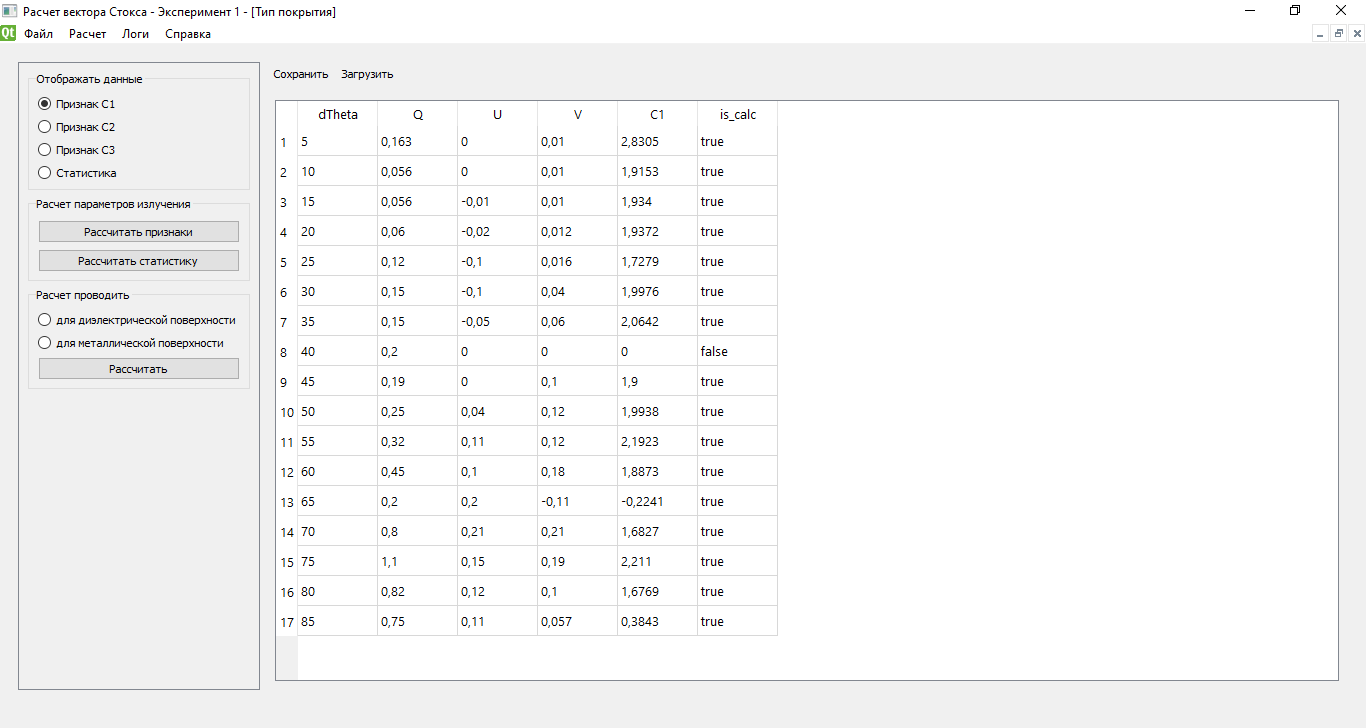


Рис. 4.12 – Определение типа покрытия

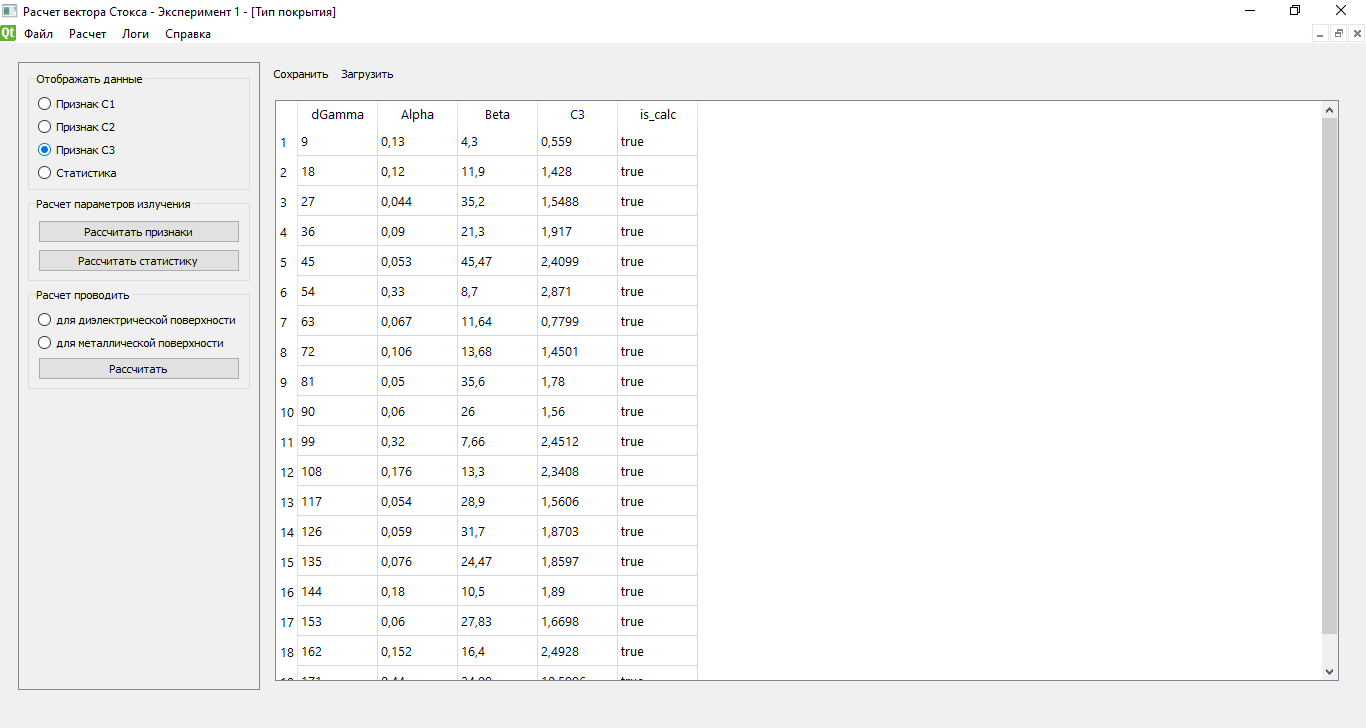


Рис. 4.13 – Исходные данные для расчета статистики

Если выбран режим отображения данных расчета статистики, то в правой части окна отображается таблица рассчитанных параметров для классификации:

Для выполнения расчета необходимо нажать на кнопку «Расчет статистики».

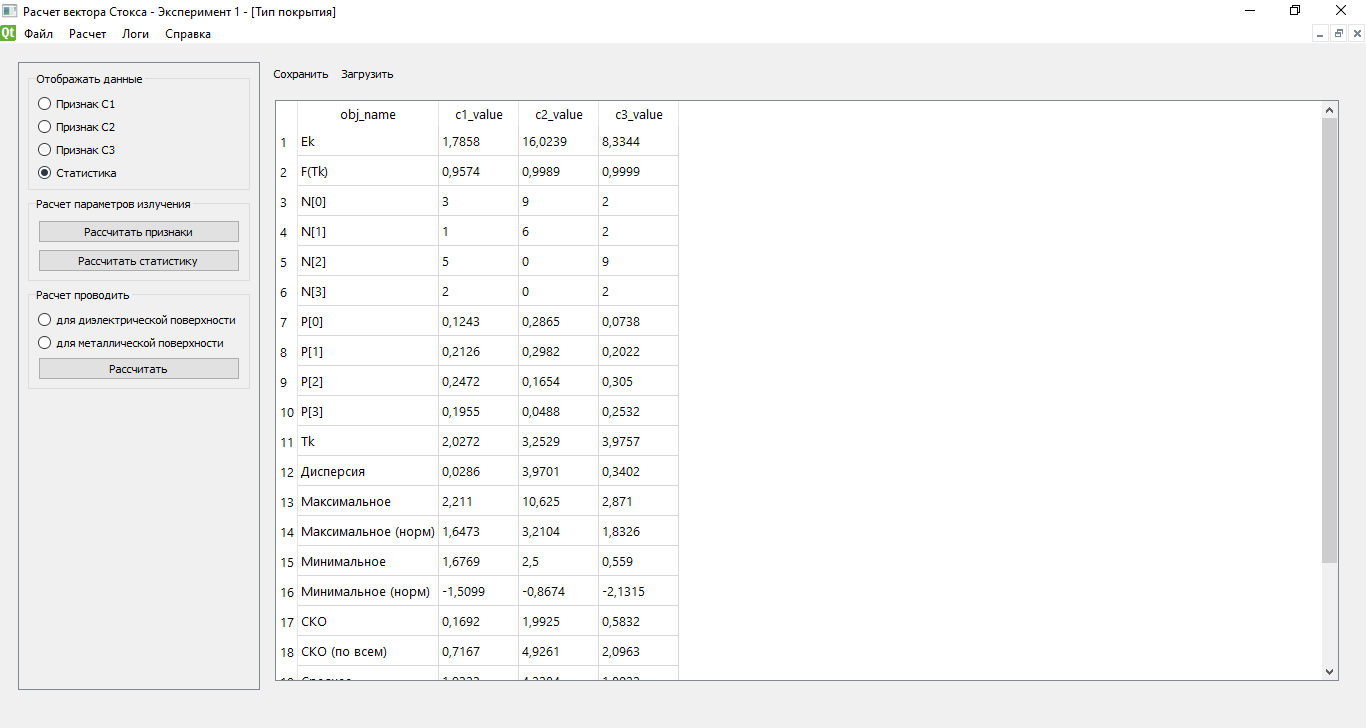


Рис. 4.14 – Расчет критериев

# Заключение

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

* изучены информационные источники по теме регистрации оптических параметров космических объектов, исследованы существующие модели и алгоритмы определения типа покрытий и расчета оптических параметров покрытий космических объектов;
* реализован на языке С++ алгоритм для расчета поляризационных характеристик отраженной световой волны и алгоритм определения типа покрытий и расчета оптических параметров покрытий космических объектов на основе вероятностного метода распознавания образов, методов аналитического и статистического моделирования;
* разработана информационная система Stokes-QT на основе свободно-распространяемого ПО (Qt, PostresQL) для решения поставленных задач.

Разработанная система 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 интерфейса пользователя;
* апробация разработанной программной системы на новых наборах выходных данных;
* внедрение разработанной системы в учебный процесс РГРТУ.

# Библиографический список

1. Руководящие принципы Комитета по использованию космического пространства в мирных целях по предупреждению образования космического мусора, 2007.
2. Адушкин В.В., Вениаминов С.С., Козлов С.И., Сильников М.В. О техногенном засорении космоса и некоторых его последствиях // Вопросы оборонной техники. Серия 16: Технические средства противодействия терроризму. 2015. № 7-8. С. 16-21.
3. Stansbery G. Orbital Debris Research at NASA // Johnson Space Center. URL: <http://aero.tamu.edu/sites/default/files/faculty/alfriend/S2.2%20Stansbery.pdf> (дата обращения: 12.5.2017).
4. Солодов А.В. Инженерный справочник по космической технике. М.: Воениздат, 1977.
5. Берлин А.А., Ассовский И.Г. Перспективные материалы и технологии для. ТОРУС ПРЕСС, 2007.
6. Панасюк М.И., Новиков Л.С. Модель космоса. Т.1: Физические условия в космическом пространстве. М.: КДУ, 2007.
7. Назаренко А.И. Моделирование космического мусора. Москва: ИКИ РАН, 2013. 216 с.
8. Райкунов Г.Г. Методы наблюдения и модели космического мусора. Кн. 1. М.: Физматлит, 2014.
9. Назаренко А. И. Моделирование космического мусора. М.: ИКИ РАН, 2013. 216 с. (Серия «Механика, управление и информатика»).
10. Зыков И.И. Методы и алгоритмы обнаружения антропогенных частиц в сложных фоновых условиях функционирования оптикоэлектронных систем [Текст]: дис. канд.техн. наук: 05.13.01: / Автор Зыков И.И. – Тверь., 2018.
11. Левкина П.А. Физические и орбитальные характеристики объектов космического мусора по данным оптических наблюдений [Текст]: дис. канд. физ.-мат. наук: 01.03.01: / Автор Левкина П.А., – Москва, 2016.
12. Дрига В.В., Колинько В.И. Разработка методики оценки качество фотометрической информации в спутниковой астрономии [Электр.] URL: <http://www.contenant.ru/1electron_mag/pdf/35.pdf>. Дата просмотра 02.03.2021.
13. Курышев В.И. Оптические наблюдения космических объектов. Учебное пособие. – М.: МО СССР, 1973
14. Здор С.Е., Широков В.Б. Оптический поиск и распознавание. – М.: Наука, 1973.
15. Астрономический календарь. Постоянная часть. Отв. редактор П.И.Бакулин. – М: Наука, 1973
16. Разработка методики оценки качества фотометрической информации в спутниковой астрономии. Дрига В.В., Колинько В.И., Электронный научно-технический журнал «Контенант». Электрон. журн. 2015. № 1. С. 13-21
17. Бодров О.А, Бодрова И.В., Наумов Д.А. Расчет матриц рассеяния светового пучка металлической и диэлектрической поверхностями техногенных космических объектов – Вестник Рязанского радиотехнического университета, № 3 2015, Рязань, 2015
18. Бодров О.А, Бодрова И.В., Наумов Д.А. Методика расчета компонентов вектора Стокса отраженного и падающего излучения при фотометрических исследованиях техногенных космических объектов – Авиакосмическое приборостроение №9 2015, Научтехлитиздат, Москва, 2015
19. Бодров О.А, Бодрова И.В., Наумов Д.А. – Модифицированная методика расчета дополнительных поляризационных параметров для поляриметрических исследований – Авиакосмическое приборостроение, №6 2016, Научтехлитиздат, Москва, 2016
20. Бодров О.А, Бодрова И.В., Наумов Д.А. – Алгоритм расчета показателя преломления металлических покрытий удаленных объектов фотометрическими методами – Инженерная физика, №5 2017, Научтехлитиздат, Москва, 2017
21. Муртазов А.К., Бодрова И.В., Наумов Д.А – Алгоритм расчета показателей преломления диэлектрических покрытий удаленных поверхностей фотометрическими методами – Авиакосмическое приборостроение, №4 2017, Научтехлитиздат, Москва, 2017

|  |
| --- |
|  |
| Приложение А. Листинг основных программных модулейА.1 Модуль calculation #ifndef CALCULATION\_H  #define CALCULATION\_H  #include <functional>  #include <cmath>  #include <complex>  namespace Calculation  {  const double PI = 3.141592;  class Gradient  {  public:  double Gamma,  W,  V;  public:  Gradient(double gamma=PI/4, double w=0.77, double v=-0.3);  double Fxy(double x, double y);  double dFdx(double x, double y);  double dFdy(double x, double y);  std::tuple<double, double, bool> gradientMethod(double x0, double y0, double eps);  };  typedef struct  {  double i,  tau,  phi;  } Intensity;  class StokesVector  {  public:  double J,  Q,  U,  V;  public:  StokesVector(double j=0, double q=0, double u=0, double v=0);  double p();  void calculate(Intensity I1, Intensity I2, Intensity I3, Intensity I4);  std::complex<double> squareRoot(std::complex<double> v, int index);  private:  void swap(int k, int n, double A[4][5]);  };  class NaturalStokesVector : public StokesVector  {  public:  NaturalStokesVector(double j=0, double q=0, double u=0, double v=0);  void calculateNatural(StokesVector &vector, std::complex<double> nju, double phi);  };  double simpson(std::function<double(double)> f, double a, double b, double eps);  double func(double t);  double laplace(double x);  std::complex<double> squareRoot(std::complex<double> v, int index);  }  #endif // CALCULATION\_H А.2 Модуль chisqr #ifndef TASK12\_H  #define TASK12\_H  #include "calculation.h"  #include <array>  #include <utility>  typedef struct  {  double alfa,  alfa1,  beta,  beta1,  re\_hi,  im\_hi;  } Reflection;  class Task12  {  public:  int Idx;  double Alfa, Beta;  Calculation::StokesVector SV;  Calculation::NaturalStokesVector NSV;  std::array<Calculation::Intensity, 4> I;  Task12(int idx, double alfa, double beta);  void loadIntensities(int idx, double i, double tau, double phi);  std::pair<Calculation::StokesVector, Calculation::NaturalStokesVector>  calcRadiation(std::complex<double> nju, double phi);  Reflection calcReflection(std::complex<double> nju, double phi,  Calculation::Gradient &gradient, bool isanalytic);  };  #endif // TASK12\_H  #ifndef CHISQR\_H  #define CHISQR\_H  double factorial(double n);  double gamma(double n);  double chi\_square\_distribution(double x, double v);  double \_chi\_square\_distribution(double x, double v);  double search\_chi\_square(double x, double h, double p, double v);  double chi\_square\_inverse(double p, double v);  #endif // CHISQR\_H A.4 Модуль gradient #include "calculation.h"  using std::sin;  using std::cos;  using std::sqrt;  using std::abs;  Calculation::Gradient::Gradient(double gamma, double w, double v)  {  Gamma = gamma;  W = w;  V = v;  }  double Calculation::Gradient::Fxy(double x, double y)  {  double a = cos(2\*x) \* cos(2\*y) - cos(Gamma) \* sin(2\*x) \* sin(2\*y) - W;  double b = sin(2\*x) \* cos(2\*y) + cos(Gamma) \* cos(2\*x) \* sin(2\*y) - V;  return a \* a + b \* b;  }  double Calculation::Gradient::dFdx(double x, double y)  {  double a = 4 \* cos(2\*x) \* cos(Gamma) \* sin(2\*y) \* W;  double b = 4 \* cos(Gamma) \* sin(2\*x) \* sin(2\*y) \* V;  double c = -4 \* cos(2\*x) \* cos(2\*y) \* V;  double d = 4 \* cos(2\*y) \* sin(2\*x) \* W;  return a + b + c + d;  }  double Calculation::Gradient::dFdy(double x, double y)  {  double a = -4 \* cos(2\*x) \* cos(2\*y) \* cos(Gamma) \* V;  double b = 4 \* cos(2\*y) \* cos(Gamma) \* cos(Gamma) \* sin(2\*y);  double c = 4 \* cos(2\*y) \* cos(Gamma) \* sin(2\*x) \* W;  double d = 4 \* cos(2\*x) \* sin(2\*y) \* W;  double e = 4 \* sin(2\*x) \* sin(2\*y) \* V;  double f = -4 \* cos(2\*y) \* sin(2\*y);  return a + b + c + d + e + f;  }  std::tuple<double, double, bool> Calculation::Gradient::gradientMethod(double x0, double y0, double eps)  {  double t = PI / 8;  double x = x0, y = y0;  int i = 0, n = 1000;  while (1){  if (i > n){  x = x0;  y = y0;  return std::make\_tuple(x, y, false);  }  double xt = x - t \* dFdx(x, y);  double yt = y - t \* dFdy(x, y);  if ( abs(sqrt((x-xt) \* (x-xt)) + sqrt((y-yt) \* (y-yt))) < eps ){  x = xt;  y = yt;  return std::make\_tuple(x, y, true);  }  if (Fxy(x ,y) < Fxy(xt, yt)){  t /= 2;  }  else {  x = xt;  y = yt;  }  i++;  }  } A.5 Модуль laplase #include "calculation.h"  double Calculation::simpson(std::function<double (double)> f, double a, double b, double eps)  {  int n = 1;  double s = 0, s0 = eps + 1;  while (std::abs(s - s0) > eps){  n \*= 2;  double h = (b - a) / n;  s0 = s;  s = f(a) + f(b);  for (int i = 1; i < n; i++){  s += 2 \* (1 + i % 2) \* func(a + i\*h);  }  s \*= (b - a) / (3 \* n);  }  return s;  }  double Calculation::func(double t)  {  return std::exp(-t \* t/2);  }  double Calculation::laplace(double x)  {  return simpson(func, 0, x, 1e-12) / std::sqrt(2 \* 3.141592);  } A6. Модуль stokes\_vector #include "calculation.h"  #include <vector>  #include <algorithm>  #include <QDebug>  using std::cos;  using std::sin;  using std::sqrt;  using std::abs;  Calculation::StokesVector::StokesVector(double j, double q, double u, double v)  {  J = j;  Q = q;  U = u;  V = v;  }  double Calculation::StokesVector::p()  {  if (abs(J) < 1e-8){  return 0.0;  }  else {  double result = sqrt(Q\*Q + U\*U + V\*V) / J;  if (result > 1){  result = 1;  }  return result;  }  }  void Calculation::StokesVector::swap(int k, int n, double A[4][5])  {  double z = A[k][k];  int i = k;  for (int j = k+1; j < n; j++){  if (abs(A[j][k]) > z){  z = A[j][k];  i = j;  }  }  if (i > k){  for (int j = k; j < n; j++){  z = A[i][j];  A[i][j] = A[k][j];  A[k][j] = z;  }  }  }  void Calculation::StokesVector::calculate(Intensity I1, Intensity I2, Intensity I3, Intensity I4)  {  double A[4][5];  double X[4] = {0, 0, 0, 0};  J = 0.0;  Q = 0.0;  U = 0.0;  V = 0.0;  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;  for (int k = 0; k < 4; k++){  if (abs(A[k][k]) < 1e-10){  std::vector<double> list;  list.reserve(4);  for (int q = 0; q < 4; q++){  list.push\_back(abs(A[q][k]));  }  auto m = std::max\_element(list.begin(), list.end());  auto idx = std::distance(list.begin(), m);  if (idx != k){  swap(k, idx, A);  }  else {  return;  }  }  for (int j = k+1; j < 4; j++){  double r = A[j][k] / A[k][k];  for (int i = 0; i < 5; i++){  A[j][i] -= r \* A[k][i];  }  }  } // end for (int k = 0; k < 4; k++)  for (int k = 3; k > -1; k--){  double r = 0;  for (int j = k+1; j < 4; j++){  double g = A[k][j] \* X[j];  r += g;  }  if (A[k][k] == 0){  return;  }  X[k] = (A[k][4] - r) / A[k][k];  }  J = X[0];  Q = X[1];  U = X[2];  V = X[3];  }  std::complex<double> Calculation::squareRoot(std::complex<double> v, int index)  {  double a = sqrt(sqrt(v.real()\*v.real() + v.imag()\*v.imag()));  double phi = std::atan(v.imag() / v.real());  if (v.real() < 0){  phi += PI;  }  if ( (index < 0) || (index > 1) ){  index = 0;  }  if (index == 1){  phi += PI;  }  return std::complex<double> {a \* cos(phi), a \* sin(phi)};  }  Calculation::NaturalStokesVector::NaturalStokesVector(double j, double q, double u, double v)  : StokesVector(j, q, u, v)  {  }  void Calculation::NaturalStokesVector::calculateNatural(Calculation::StokesVector &vector,  std::complex<double> nju, double phi)  {  double cphi = cos(phi / 180.0), sphi = sin(phi / 180.0);  std::complex<double> nju2 = nju \* nju,  cosphi(cphi, 0.0),  sinphi(sphi, 0.0),  sinphi2 = sinphi \* sinphi,  sn = nju2 - sinphi2,  sq = Calculation::squareRoot(sn, 0);  if (sq.imag() > 0){  sq = Calculation::squareRoot(sn, 1);  }  std::complex<double> ch1 = cosphi - sq,  zn1 = cosphi + sq,  nju2cosphi = nju2 \* cosphi,  ch2 = nju2cosphi - sq,  zn2 = nju2cosphi + sq,  r1 = ch1 / zn1,  r2 = ch2 / zn2,  r1r2\_ = r1 \* std::conj(r2);  double r1\_2 = abs(r1) \* abs(r1),  r2\_2 = abs(r2) \* abs(r2),  r1r2\_2 = abs(r1r2\_) \* abs(r1r2\_);  J = (vector.J \* (r1\_2 + r2\_2) - vector.Q \* (r1\_2 - r2\_2)) / (2 \* r1\_2 \* r2\_2);  Q = (vector.Q \* (r1\_2 + r2\_2) - vector.J \* (r1\_2 - r2\_2)) / (2 \* r1\_2 \* r2\_2);  U = (vector.U \* r1r2\_.real() + vector.V \* r1r2\_.imag()) / r1r2\_2;  U = (vector.V \* r1r2\_.real() - vector.U \* r1r2\_.imag()) / r1r2\_2;  } A7. Модуль task\_12Файл task\_12.h #ifndef TASK12\_H  #define TASK12\_H  #include "calculation.h"  #include <array>  #include <utility>  typedef struct  {  double alfa,  alfa1,  beta,  beta1,  re\_hi,  im\_hi;  } Reflection;  class Task12  {  public:  int Idx;  double Alfa, Beta;  Calculation::StokesVector SV;  Calculation::NaturalStokesVector NSV;  std::array<Calculation::Intensity, 4> I;  Task12(int idx, double alfa, double beta);  void loadIntensities(int idx, double i, double tau, double phi);  std::pair<Calculation::StokesVector, Calculation::NaturalStokesVector>  calcRadiation(std::complex<double> nju, double phi);  Reflection calcReflection(std::complex<double> nju, double phi,  Calculation::Gradient &gradient, bool isanalytic);  };  #endif // TASK12\_H Файл task12.cpp #include "task12.h"  using std::sin;  using std::cos;  using std::sqrt;  using std::abs;  using std::tan;  Task12::Task12(int idx, double alfa, double beta)  {  Idx = idx;  Alfa = alfa;  Beta = beta;  SV = Calculation::StokesVector();  NSV = Calculation::NaturalStokesVector();  for (int i = 0; i < 4; i++){  I[i] = Calculation::Intensity{0, 0, 0};  }  }  void Task12::loadIntensities(int idx, double i, double tau, double phi)  {  I[idx] = Calculation::Intensity {i, tau, phi};  }  std::pair<Calculation::StokesVector, Calculation::NaturalStokesVector>  Task12::calcRadiation(std::complex<double> nju, double phi)  {  SV.calculate(I[0], I[1], I[2], I[3]);  NSV.calculateNatural(SV, nju, phi);  return std::make\_pair(SV, NSV);  }  Reflection Task12::calcReflection(std::complex<double> nju, double phi,  Calculation::Gradient &gradient, bool isanalytic)  {  auto rad = calcRadiation(nju, phi);  SV = rad.first;  NSV = rad.second;  double J = SV.J;  double Q = SV.Q;  double U = SV.U;  double V = SV.V;  if (Q\*Q + V\*V + U\*U < 1e-5){  gradient.W = 0;  gradient.V = 0;  }  else {  gradient.W = Q / sqrt(Q\*Q + V\*V + U\*U);  gradient.V = U / sqrt(Q\*Q + V\*V + U\*U);  }  double x = 0, x0 = 0, y = 0, y0 = 0, x1 = 0, x2 = 0;  double Alfa, Beta;  if (isanalytic){  // bool done = true;  double cg = cos(gradient.Gamma);  double a = Q\*Q + U\*U + cg\*cg \* (Q\*Q + V\*V + U\*U);  Beta = abs(std::atan(V / sqrt(a)));  if (gradient.Gamma \* V > 0){  Beta = -Beta;  }  Alfa = abs(std::atan(sin(2\*Beta) / cos(2\*Beta) \* cg));  if (abs(U) < 1e-5){  Alfa = 0;  }  else if (U < 0){  Alfa = -Alfa;  }  }  else {  auto gr = gradient.gradientMethod(x0, y0, 1e-5);  x = std::get<0>(gr);  y = std::get<1>(gr);  bool done = std::get<2>(gr);  if (!done){  std::exit(-1);  }  if (abs(gradient.Fxy(x, y)) > 1e-5){  std::exit(-1);  }  double a = cos(-2\*y) + gradient.W;  double b = sqrt(2) \* sin(-2\*y);  double c = gradient.W - cos(-2\*y);  double d = b \* b - 4 \* c \* a;  if (d < 0){  std::exit(-1);  }  x1 = std::atan( (-b + sqrt(d)) / (2 \* a) );  x2 = std::atan( (-b - sqrt(d)) / (2 \* a) );  if (abs(x1 - x) > abs(x2 - x)){  x2 = x1;  }  Alfa = (x + x2) / 2;  if (V > 0){  Beta = -abs(y);  }  else {  Beta = abs(y);  }  }  auto res = std::complex<double> {tan(Alfa), tan(Beta)} /  std::complex<double> {1, -tan(Alfa) \* tan(Beta)};  return Reflection {Alfa, x, Beta, y, res.real(), res.imag()};  } A8. Модуль task3Файл task3.h #ifndef TASK3\_H  #define TASK3\_H  #include <vector>  #include <map>  #include <string>  namespace Task3  {  enum class PlaneTypes  {  IS\_DIEL,  IS\_METAL  };  class ItemC  {  public:  bool iscalc;  double value;  double value\_norm;  ItemC() = default;  virtual double calc() {return value;}  };  class ItemC1 : public ItemC  {  public:  double dTetta, Q, U, V;  ItemC1(double dtetta, double q, double u, double v);  double calc() override;  };  class ItemC2 : public ItemC  {  public:  double dGamma, Q, U, V;  ItemC2(double dgamma, double q, double u, double v);  double calc() override;  };  class ItemC3 : public ItemC  {  public:  double dGamma, Alpha, Beta;  ItemC3(double dgamma, double alpha, double beta);  double calc() override;  };  class Interval  {  public:  static int INTERVAL\_COUNT;  int number, value\_count;  double low, high, laplaceValueLow, laplaceValueHigh, pi, npi, ni\_npi,  ni\_npi\_norm;  };  struct Sample  {  double tetta, lambda, n;  };  struct Param  {  double tetta1, a1, v1, u1, tetta2, a2, v2, u2;  };  struct Material  {  std::string name, label;  double re\_min, re\_max, im\_min, im\_max, length;  };  inline double det2(double a11, double a21, double a12, double a22)  {  return a11 \* a22 - a21 \* a12;  }  double len2rect(double re, double im, double re1, double re2, double im1, double im2);  double chi2P(double chi, int df);  std::vector<Interval> getIntervals(int count, double min, double max);  std::map<std::string, double> calcStat(std::vector<ItemC> &data);  std::pair<int, Material>  calcPlaneMaterial(std::vector<Sample> data, std::vector<Material> materials, PlaneTypes type);  }  #endif // TASK3\_H Файл task3.cpp #include "task3.h"  #include <cmath>  #include <algorithm>  #include <QDebug>  #include "calculation.h"  #include "chisqr.h"  #include "logger.h"  using std::sqrt;  using std::min;  using std::max;  using std::abs;  using std::sin;  using std::cos;  using std::vector;  using Calculation::PI;  double Task3::len2rect(double re, double im, double re1, double re2, double im1, double im2)  {  double x1 = min(re1, re2);  double x2 = max(re1, re2);  double y1 = min(im1, im2);  double y2 = max(im1, im2);  /\* 1 | 2 | 3  \* 4 | - | 5  \* 6 | 7 | 8  \*/  if ( (re >= x1) && (re <= x2) && (im >= y1) && (im <= y2) ){  return 0;  }  else if (re < x1){  if (im > y2)  return sqrt( (re-x1)\*(re-x1) + (im-y2)\*(im-y2) );  else if (im < y2)  return sqrt( (re-x1)\*(re-x1) + (im-y1)\*(im-y1) );  else  return abs(re - x1);  }  else if (re > x2){  if (im > y2)  return sqrt( (re-x2)\*(re-x2) + (im-y2)\*(im-y2) );  else if (im < y1)  return sqrt( (re-x2)\*(re-x2) + (im-y1)\*(im-y1) );  else  return abs(re - x2);  }  else {  return min( abs(im - y1), abs(im - y2) );  }  }  double Task3::chi2P(double chi, int df)  {  /\*\* Return prob(chisq >= chi, with df degrees of freedom).  \* df must be even.  \*/  // If chi is very large, exp(-m) will underflow to 0.  double m = chi / 2.0;  double term = std::exp(-m), sum = term;  for (int i = 1; i < df / 2; i++){  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);  }  Task3::ItemC1::ItemC1(double dtetta, double q, double u, double v)  {  dTetta = dtetta;  Q = q;  U = u;  V = v;  }  double Task3::ItemC1::calc()  {  if (V != 0){  iscalc = true;  return ( Q \* sin(2\*dTetta\*PI/180) + U \* cos(2\*dTetta\*PI/180) ) / V;  }  else {  iscalc = false;  qDebug() << "ItemC1::calc() : V == 0\n";  return 0.0; // throw something  }  }  Task3::ItemC2::ItemC2(double dgamma, double q, double u, double v)  {  dGamma = dgamma;  Q = q;  U = u;  V = v;  }  double Task3::ItemC2::calc()  {  if (V != 0){  iscalc = true;  return U / V;  }  else {  qDebug() << "ItemC2::calc() : V == 0\n";  iscalc = false;  return 0.0; // throw something  }  }  Task3::ItemC3::ItemC3(double dgamma, double alpha, double beta)  {  dGamma = dgamma;  Alpha = alpha;  Beta = beta;  }  double Task3::ItemC3::calc()  {  iscalc = true;  return Alpha \* Beta;  }  vector<Task3::Interval> Task3::getIntervals(int icount, double imin, double imax)  {  double len = (imax - imin) / icount;  vector<Task3::Interval> intervals;  intervals.reserve(icount);  for (int i = 0; i < icount; i++){  Interval item;  item.number = i;  item.value\_count = 0;  item.low = imin + i \* len;  item.high = imin + (i + 1) \* len;  item.laplaceValueLow = Calculation::laplace(item.low);  item.laplaceValueHigh = Calculation::laplace(item.high);  item.pi = item.laplaceValueHigh - item.laplaceValueLow;  intervals.push\_back(item);  }  return intervals;  }  double mean(vector<Task3::ItemC> data)  {  double sum = 0;  for (const auto &val : data){  sum += val.value;  }  return sum / data.size();  }  double stdev(vector<Task3::ItemC> data)  {  double m = mean(data);  double s = 0;  for (const auto &x : data){  s += (x.value - m) \* (x.value - m);  }  return sqrt(s / (data.size() - 1));  }  std::map<std::string, double> Task3::calcStat(std::vector<ItemC> &data)  {  std::map<std::string, double> stat;  Logger logger("statistics.log");  vector<ItemC> data\_c;  data\_c.reserve(data.size());  for (auto &x : data){  if (x.iscalc){  data\_c.push\_back(x);  }  }  double countP = data\_c.size(),  midP = mean(data\_c),  sdP = stdev(data\_c);  logger.logInfo(QString("Всего: %1").arg(countP, 0, 'g',3));  logger.logInfo(QString("Среднее (по всем): %1").arg(midP, 0, 'g',3));  logger.logInfo(QString("СКО (по всем): %1").arg(sdP, 0, 'g',3));  stat["Всего"] = countP;  stat["Среднее (по всем)"] = midP;  stat["СКО (по всем)"] = sdP;  vector<ItemC> data\_n;  data\_n.reserve(data\_c.size());  for (auto &x : data\_c){  x.iscalc = x.iscalc && (abs(x.value - midP) < sdP);  if (x.iscalc){  data\_n.push\_back(x);  }  }  double mid = mean(data\_n);  double sd = stdev(data\_n);  auto compare = [](const ItemC &x1, const ItemC &x2)->bool{return x1.value < x2.value;};  double min\_v = (\*std::min\_element(data\_n.begin(), data\_n.end(), compare)).value;  double max\_v = (\*std::max\_element(data\_n.begin(), data\_n.end(), compare)).value;  for (auto &x : data\_n){  x.value\_norm = (x.value - mid) / sd;  }  logger.logInfo(QString("Минимальное: %1").arg(min\_v, 0, 'g',3));  logger.logInfo(QString("Максимальное: %1").arg(max\_v, 0, 'g',3));  logger.logInfo(QString("Среднее: %1").arg(mid, 0, 'g',3));  logger.logInfo(QString("Дисперсия: %1").arg(sd \* sd, 0, 'g',3));  logger.logInfo(QString("СКО: %1").arg(sd, 0, 'g',3));  stat["Минимальное"] = min\_v;  stat["Максимальное"] = max\_v;  stat["Среднее"] = mid;  stat["Дисперсия"] = sd \* sd;  stat["СКО"] = sd;  int count\_n = data\_n.size();  auto compare\_n = [](const ItemC &x1, const ItemC &x2)->bool{return x1.value\_norm < x2.value\_norm;};  double min\_n = (\*std::min\_element(data\_n.begin(), data\_n.end(), compare\_n)).value\_norm;  double max\_n = (\*std::max\_element(data\_n.begin(), data\_n.end(), compare\_n)).value\_norm;  logger.logInfo(QString("Минимальное (норм): %1").arg(min\_n, 0, 'g',3));  logger.logInfo(QString("Максимальное (норм): %1").arg(max\_n, 0, 'g',3));  stat["Минимальное (норм)"] = min\_n;  stat["Максимальное (норм)"] = max\_n;  logger.logInfo("Вычисление интервалов");  auto intervals = getIntervals(Interval::INTERVAL\_COUNT, min\_n, max\_n);  double SumProb = 0;  for (const auto &x : intervals){  SumProb += x.pi;  logger.logInfo(QString("Interval(%1, %2)").arg(x.low, 0, 'g',3).arg(x.high, 0, 'g',3));  }  for (const auto &x : data\_n){  double value = std::round(x.value\_norm \* 100) / 100;  int index = -1;  for (int j = 0; j < static\_cast<int>(intervals.size()); j++){  if (abs(value - intervals[j].high) < 1e-3){  if (j < Interval::INTERVAL\_COUNT / 2)  index = j + 1;  else  index = j;  break;  }  if (abs(value - intervals[j].low) < 1e-3){  if (j < Interval::INTERVAL\_COUNT / 2)  index = j;  else  index = j - 1;  break;  }  if ( (value > intervals[j].low) && (value < intervals[j].high+1e-3) ){  index = j;  break;  }  } // for (int j = 0; j < intervals.size(); j++)  if (index == -1){  if (value <= min\_n + 1e-2)  index = 0;  else  index = Interval::INTERVAL\_COUNT - 1;  }  logger.logInfo(QString("Значение : %1 Интервал : %2")  .arg(value, 0, 'g',3).arg(index, 0, 'g',3));  index = std::clamp(index, 0, Interval::INTERVAL\_COUNT-1);  intervals[index].value\_count++;  } // for (const auto &x : data\_n)  double pis[5] = {0}, ns[5] = {0};  int i = 0;  for (auto &x : intervals){  x.npi = count\_n \* x.pi / SumProb;  x.ni\_npi = (x.value\_count - x.npi) \* (x.value\_count - x.npi);  x.ni\_npi\_norm = x.ni\_npi / x.npi;  pis[i] = x.pi;  ns[i] = x.value\_count;  logger.logInfo(QString("P[%1] : %2 N[%1] : %3").arg(i, 0, 'g',3)  .arg(x.pi, 0, 'g',3).arg(x.value\_count, 0, 'g',3));  i++;  }  stat["P[0]"] = pis[0];  stat["P[1]"] = pis[1];  stat["P[2]"] = pis[2];  stat["P[3]"] = pis[3];  stat["P[4]"] = pis[4];  stat["N[0]"] = ns[0];  stat["N[1]"] = ns[1];  stat["N[2]"] = ns[2];  stat["N[3]"] = ns[3];  stat["N[4]"] = ns[4];  double SumHi = 0;  for (const auto &x : intervals)  SumHi += x.ni\_npi\_norm;  double SumHiTeor = chi\_square\_inverse(0.05, Interval::INTERVAL\_COUNT-3);  logger.logInfo(QString("Хи-квадрат (эмп.) : %1").arg(SumHi, 0, 'g', 3));  logger.logInfo(QString("Хи-квадрат (теор.) : %1").arg(SumHiTeor, 0, 'g', 3));  stat["Хи-квадрат (эмп.)"] = SumHi;  stat["Хи-квадрат (теор.)"] = SumHiTeor;  data\_c.clear();  data\_c.reserve(data\_n.size());  for (auto &x : data\_n){  data\_c.push\_back(x);  }  double midE = mean(data\_c),  sdE = stdev(data\_c),  d = (\*std::max\_element(data\_c.begin(), data\_c.end(),  [=](const auto &x1, const auto &x2){  return abs(x1.value - midE) < abs(x2.value - midE);})).value,  Ek = abs(d - midE),  Tk = Ek / sdE,  FTk = 2 \* Calculation::laplace(Tk);  logger.logInfo(QString("Ek : %1").arg(Ek, 0, 'g', 3));  logger.logInfo(QString("Tk : %1").arg(Tk, 0, 'g', 3));  logger.logInfo(QString("F(Tk) : %1").arg(FTk, 0, 'g', 3));  stat["Ek"] = Ek;  stat["Tk"] = Tk;  stat["F(Tk)"] = FTk;  return stat;  }  std::pair<int, Task3::Material>  Task3::calcPlaneMaterial(std::vector<Task3::Sample> data, std::vector<Task3::Material> materials,  Task3::PlaneTypes type)  {  int idx = 0;  double val = data[0].lambda;  for (int i = 0; i < static\_cast<int>(data.size()); i++){  if (data[i].lambda < val){  idx = i;  val = data[i].lambda;  }  }  for (int i = 0; i < static\_cast<int>(data.size()); i++){  double Tetta = Calculation::PI \* data[i].tetta / 180.0,  Lambda = data[i].lambda;  switch (type){  case PlaneTypes::IS\_DIEL:  if (i < idx){  data[i].n = sin(Tetta) / cos(Tetta)  \* sqrt(1 + Lambda \* Lambda - 2 \* Lambda \* cos(2 \* Tetta))  / (1 - Lambda);  }  else{  data[i].n = sin(Tetta) / cos(Tetta)  \* sqrt(1 + Lambda \* Lambda + 2 \* Lambda \* cos(2 \* Tetta))  / (1 + Lambda);  }  break;  case PlaneTypes::IS\_METAL:  if (i < idx){  data[i].n = sin(Tetta) \* sin(Tetta) / cos(Tetta) \* (1 + Lambda) / (1 - Lambda);  }  else{  data[i].n = sin(Tetta) \* sin(Tetta) / cos(Tetta) \* (1 - Lambda) / (1 + Lambda);  }  break;  default:  break;  }  } // for (int i = 0; i < data.size(); i++)  double a = 0;  for (auto x : data){  a += x.n;  }  double nmid = a / (data.size() - 1);  std::complex<double> coef;  int ParamCount, i2;  switch (type) {  case PlaneTypes::IS\_DIEL:  coef = std::complex(nmid, 0.0);  break;  case PlaneTypes::IS\_METAL:  {  int count = data.size() - 1;  if (count / 2 == 1){  ParamCount = count / 2 + 1;  i2 = count - 1;  }  else{  ParamCount = count / 2;  i2 = count / 2;  }  vector<Param> params;  params.reserve(ParamCount);  for (int i = 0; i < ParamCount; i++){  params.push\_back(Param {0, 0, 0, 0, 0, 0, 0, 0});  }  for (int i = 0; i < static\_cast<int>(data.size()); i++){  if (i < ParamCount){  params[i].tetta1 = data[i].tetta;  params[i].a1 = data[i].n;  }  if (i >= i2){  params[i].tetta2 = data[i].tetta;  params[i].a2 = data[i].n;  }  }  double u\_mid = 0, v\_mid = 0, c\_mid = 0;  for (auto &x : params){  double d1 = det2(  std::pow(x.a1, 4) + x.a1\*std::pow(sin(x.tetta1\*Calculation::PI/180), 2),  std::pow(x.a2, 4) + x.a2\*std::pow(sin(x.tetta2\*Calculation::PI/180), 2),  x.a1 \* x.a1,  x.a2 \* x.a2  );  double d2 = det2(1, 1, x.a1\*x.a1, x.a2\*x.a2);  double d4 = 0, u = 0, v = 0;  if (d1 / d2 < 0){  // все хуйня  }  else{  v = sqrt(d1 / d2);  double d3 = det2(  1,  1,  std::pow(x.a1, 4) + x.a1\*std::pow(sin(x.tetta1\*Calculation::PI/180), 2),  std::pow(x.a2, 4) + x.a2\*std::pow(sin(x.tetta2\*Calculation::PI/180), 2)  );  d4 = det2(1, 1, x.a1\*x.a1, x.a2\*x.a2);  u = d3 / d4;  }  if ( (abs(d2) > 1e-3) && (abs(d4) > 1e-3) ){  v\_mid += v;  u\_mid += u;  c\_mid += 1;  }  else {  // все хуйня  }  } // for (auto &x : params)  v\_mid /= c\_mid;  u\_mid /= c\_mid;  double re = sqrt( (u\_mid + sqrt(u\_mid\*u\_mid + 4\*v\_mid\*v\_mid)) / 2 );  coef = std::complex(re, -v\_mid/re);  break;  }  default:  break;  }  idx = 0;  int length = 1e+8;  for (int i = 0; i < static\_cast<int>(materials.size()); i++){  materials[i].length = len2rect(coef.real(), coef.imag(), materials[i].re\_min,  materials[i].re\_max, materials[i].im\_min, materials[i].im\_max);  if (materials[i].length < length){  idx = i;  length = materials[i].length;  }  }  return std::pair {idx, materials[idx]};  }  int Task3::Interval::INTERVAL\_COUNT = 5; А.9 Модули, реализующие интерфес пользователяФайл main.cpp #include "mainwindow.h"  #include <QApplication>  int main(int argc, char \*argv[])  {  QApplication a(argc, argv);  MainWindow w;  w.show();  return a.exec();  } Файл mainwindow.h #ifndef MAINWINDOW\_H  #define MAINWINDOW\_H  #include <QMainWindow>  #include "database.h"  QT\_BEGIN\_NAMESPACE  namespace Ui { class MainWindow; }  QT\_END\_NAMESPACE  class MainWindow : public QMainWindow  {  Q\_OBJECT  public:  MainWindow(QWidget \*parent = nullptr);  ~MainWindow();  private:  Ui::MainWindow \*ui;  Database \*db;  QString defaultTitle;  QString selectedExperimentUUID;  signals:  void logfile(QString filename);  private slots:  void openConstants();  void loadPlot();  void openAllExperiments();  void setMenuEnabled();  void currentExperiment(QString uuid, QString name);  void openRefraction();  void makeCalc12();  void makeCalc3();  void openLoadDialog();  void openLogRays();  void openLogStats();  };  #endif // MAINWINDOW\_H Файл mainwindow.cpp #include "mainwindow.h"  #include "ui\_mainwindow.h"  #include <QToolButton>  #include <QFileDialog>  #include <QPushButton>  #include "experimentswindow.h"  #include "newexperimentdialog.h"  #include "materialrefractiontable.h"  #include "calc12window.h"  #include "calc3window.h"  #include "plotwidget.h"  #include "constantswindow.h"  #include "loaddatadialog.h"  #include "logfilewidget.h"  MainWindow::MainWindow(QWidget \*parent)  : QMainWindow(parent)  , ui(new Ui::MainWindow)  {  ui->setupUi(this);  db = Database::instance();  db->makeConnection();  db->openDB();  defaultTitle = this->windowTitle();  connect(ui->actionParameters, &QAction::triggered, this, &MainWindow::openConstants);  connect(ui->actionPlot, &QAction::triggered, this, &MainWindow::loadPlot);  connect(ui->actionExperiments, &QAction::triggered, this, &MainWindow::openAllExperiments);  connect(ui->actionRefraction, &QAction::triggered, this, &MainWindow::openRefraction);  connect(ui->actionCalculate12, &QAction::triggered, this, &MainWindow::makeCalc12);  connect(ui->actionCalculate3, &QAction::triggered, this, &MainWindow::makeCalc3);  connect(ui->actionLoadData, &QAction::triggered, this, &MainWindow::openLoadDialog);  connect(ui->actionLogRays, &QAction::triggered, this, &MainWindow::openLogRays);  connect(ui->actionLogPolarization, &QAction::triggered, this, &MainWindow::openLogStats);  }  MainWindow::~MainWindow()  {  db->closeDB();  delete ui;  }  void MainWindow::openConstants()  {  ConstantsWindow \*cw = new ConstantsWindow();  ui->mdiArea->addSubWindow(cw);  cw->showMaximized();  }  void MainWindow::loadPlot()  {  PlotWidget \*pw = new PlotWidget(selectedExperimentUUID);  ui->mdiArea->addSubWindow(pw);  pw->showMaximized();  }  void MainWindow::openAllExperiments()  {  ExperimentsWindow \*ew = new ExperimentsWindow();  ui->mdiArea->addSubWindow(ew);  ew->showMaximized();  connect(ew, &ExperimentsWindow::experimentSelected, this, &MainWindow::currentExperiment);  connect(ew, SIGNAL(experimentSelected(QString, QString)), this, SLOT(setMenuEnabled()));  }  void MainWindow::setMenuEnabled()  {  ui->menuCalculate->setEnabled(true);  ui->menuLog->setEnabled(true);  }  void MainWindow::currentExperiment(QString uuid, QString name)  {  this->setWindowTitle(defaultTitle + " - " + name);  this->selectedExperimentUUID = uuid;  }  void MainWindow::openRefraction()  {  MaterialRefractionTable \*table = new MaterialRefractionTable();  ui->mdiArea->addSubWindow(table);  table->showMaximized();  table->displayTable();  }  void MainWindow::makeCalc12()  {  Calc12Window \*calc = new Calc12Window(selectedExperimentUUID);  ui->mdiArea->addSubWindow(calc);  calc->showMaximized();  }  void MainWindow::makeCalc3()  {  Calc3Window \*calc = new Calc3Window(selectedExperimentUUID);  ui->mdiArea->addSubWindow(calc);  calc->showMaximized();  }  void MainWindow::openLoadDialog()  {  LoadDataDialog \*lddlg = new LoadDataDialog();  lddlg->exec();  }  void MainWindow::openLogRays()  {  LogFileWidget \*lfw = new LogFileWidget("rays.log");  ui->mdiArea->addSubWindow(lfw);  lfw->showMaximized();  }  void MainWindow::openLogStats()  {  LogFileWidget \*lfw = new LogFileWidget("statistics.log");  ui->mdiArea->addSubWindow(lfw);  lfw->showMaximized();  } Файл database.h #ifndef DATABASE\_H  #define DATABASE\_H  #include <QObject>  #include <QtSql>  class Database : public QObject  {  Q\_OBJECT  public:  static Database \* instance();  void makeConnection();  bool openDB();  void closeDB();  private:  static Database \*pdb;  QSqlDatabase conn;  explicit Database(QObject \*parent = nullptr);  };  #endif // DATABASE\_H Файл database.cpp #include "database.h"  #include <QMessageBox>  Database::Database(QObject \*parent) : QObject(parent)  {  }  void Database::makeConnection()  {  conn = QSqlDatabase::addDatabase("QPSQL", "stokes\_db");  if (!conn.isValid()){  QMessageBox msgb;  msgb.setText(conn.lastError().text());  msgb.setStandardButtons(QMessageBox::Cancel);  msgb.exec();  }  conn.setDatabaseName("stokes\_db");  conn.setHostName("127.0.0.1");  conn.setPort(5432);  conn.setUserName("postgres");  conn.setPassword("postgres");  }  bool Database::openDB()  {  if (!conn.isOpen()){  if (!conn.open()){  QMessageBox msgb;  msgb.setText(conn.lastError().text());  msgb.setStandardButtons(QMessageBox::Cancel);  msgb.exec();  }  }  return true;  }  void Database::closeDB()  {  if (conn.isOpen()){  conn.close();  }  }  Database \* Database::instance()  {  if (!pdb){  pdb = new Database();  }  return pdb;  }  Database\* Database::pdb = nullptr; Файл experimentswindow.h #ifndef EXPERIMENTSWINDOW\_H  #define EXPERIMENTSWINDOW\_H  #include <QWidget>  #include <QSqlTableModel>  #include <QToolBar>  namespace Ui {  class ExperimentsWindow;  }  class ExperimentsWindow : public QWidget  {  Q\_OBJECT  public:  explicit ExperimentsWindow(QWidget \*parent = nullptr);  ~ExperimentsWindow();  signals:  void experimentSelected(QString uuid, QString name);  private:  Ui::ExperimentsWindow \*ui;  QSqlTableModel \*modelExp;  QToolBar \*tb;  void setupToolbar();  void setupTable();  void importRaysData(QString expuuid, QString filename);  void importC1Data(QString expuuid, QString filename);  void importC2Data(QString expuuid, QString filename);  void importC3Data(QString expuuid, QString filename);  private slots:  void openAddDialog();  void deleteExperiment();  void selectExperiment();  void importExperimentData(QString expuuid, QString frays, QString fc1, QString fc2, QString fc3);  // void openLoadDialog();  // void openUpdateDialog();  };  #endif // EXPERIMENTSWINDOW\_H Файл experimentswindow.cpp #include "experimentswindow.h"  #include "ui\_experimentswindow.h"  #include <QSqlRecord>  #include <QSqlQuery>  #include <QFile>  #include <QTextStream>  #include <QMessageBox>  #include "roweditingdelegate.h"  #include "newexperimentdialog.h"  ExperimentsWindow::ExperimentsWindow(QWidget \*parent) :  QWidget(parent),  ui(new Ui::ExperimentsWindow),  modelExp(new QSqlTableModel(this, QSqlDatabase::database("stokes\_db")))  {  ui->setupUi(this);  setupTable();  setupToolbar();  }  ExperimentsWindow::~ExperimentsWindow()  {  delete ui;  }  void ExperimentsWindow::setupToolbar()  {  tb = new QToolBar(this);  // QAction \*actSave = new QAction("Save", this);  // tb->addAction(actSave);  // connect(actSave, &QAction::triggered, ...)  // QAction \*actLoad = new QAction("Load", this);  // tb->addAction(actLoad);  // connect(actLoad, &QAction::triggered, ...)  QAction \*actAdd = new QAction("Создать", this);  tb->addAction(actAdd);  connect(actAdd, &QAction::triggered, this, &ExperimentsWindow::openAddDialog);  QAction \*actDelete = new QAction("Удалить", this);  tb->addAction(actDelete);  connect(actDelete, &QAction::triggered, this, &ExperimentsWindow::deleteExperiment);  QAction \*actSelect = new QAction("Выбрать", this);  tb->addAction(actSelect);  connect(actSelect, &QAction::triggered, this, &ExperimentsWindow::selectExperiment);  ui->toolLayout->addWidget(tb);  }  void ExperimentsWindow::setupTable()  {  modelExp->setTable("experiments");  modelExp->setEditStrategy(QSqlTableModel::OnManualSubmit);  modelExp->setSort(modelExp->fieldIndex("obj\_id"), Qt::AscendingOrder);  modelExp->select();  ui->exprimentsTable->setModel(modelExp);  ui->exprimentsTable->hideColumn(modelExp->fieldIndex("obj\_id"));  ui->exprimentsTable->hideColumn(modelExp->fieldIndex("obj\_uuid"));  ui->exprimentsTable->setColumnWidth(modelExp->fieldIndex("obj\_name"), 200);  ui->exprimentsTable->setColumnWidth(modelExp->fieldIndex("obj\_comment"), 400);  ui->exprimentsTable->setSelectionBehavior(QAbstractItemView::SelectRows);  ui->exprimentsTable->setItemDelegate(new RowEditingDelegate());  ui->exprimentsTable->resizeRowsToContents();  }  void ExperimentsWindow::importRaysData(QString expuuid, QString filename)  {  QSqlTableModel \*model = new QSqlTableModel(this, QSqlDatabase::database("stokes\_db"));  model->setTable("calculation\_12");  QSqlRecord record = model->record();  record.remove(model->fieldIndex("obj\_id"));  QFile file = QFile(filename);  if (!file.open(QFile::ReadOnly | QFile::Text)){  return;  }  QTextStream in(&file);  in.readLine(); // skip header  while (!in.atEnd()){  QString line = in.readLine();  QStringList values = line.split(";");  record.setValue("Alfa", values[0]);  record.setValue("Beta", values[1]);  record.setValue("Tau1", values[2]);  record.setValue("Phi1", values[3]);  record.setValue("I1", values[4]);  record.setValue("Tau2", values[5]);  record.setValue("Phi2", values[6]);  record.setValue("I2", values[7]);  record.setValue("Tau3", values[8]);  record.setValue("Phi3", values[9]);  record.setValue("I3", values[10]);  record.setValue("Tau4", values[11]);  record.setValue("Phi4", values[12]);  record.setValue("I4", values[13]);  record.setValue("experiment\_id", expuuid);  model->insertRecord(-1, record);  }  file.close();  model->submitAll();  }  void ExperimentsWindow::importC1Data(QString expuuid, QString filename)  {  QSqlTableModel \*model = new QSqlTableModel(this, QSqlDatabase::database("stokes\_db"));  model->setTable("c1\_parameter");  QSqlRecord record = model->record();  QFile file = QFile(filename);  if (!file.open(QFile::ReadOnly | QFile::Text)){  return;  }  QTextStream in(&file);  in.readLine(); // skip header  while (!in.atEnd()){  QString line = in.readLine();  QStringList values = line.split(";");  record.setValue("dTheta", values[0]);  record.setValue("Q", values[1]);  record.setValue("U", values[2]);  record.setValue("V", values[3]);  record.setValue("experiment\_id", expuuid);  model->insertRecord(-1, record);  }  file.close();  model->submitAll();  }  void ExperimentsWindow::importC2Data(QString expuuid, QString filename)  {  QSqlTableModel \*model = new QSqlTableModel(this, QSqlDatabase::database("stokes\_db"));  model->setTable("c2\_parameter");  QSqlRecord record = model->record();  QFile file = QFile(filename);  if (!file.open(QFile::ReadOnly | QFile::Text)){  return;  }  QTextStream in(&file);  in.readLine(); // skip header  while (!in.atEnd()){  QString line = in.readLine();  QStringList values = line.split(";");  record.setValue("dGamma", values[0]);  record.setValue("Q", values[1]);  record.setValue("U", values[2]);  record.setValue("V", values[3]);  record.setValue("experiment\_id", expuuid);  model->insertRecord(-1, record);  }  file.close();  model->submitAll();  }  void ExperimentsWindow::importC3Data(QString expuuid, QString filename)  {  QSqlTableModel \*model = new QSqlTableModel(this, QSqlDatabase::database("stokes\_db"));  model->setTable("c3\_parameter");  QSqlRecord record = model->record();  QFile file = QFile(filename);  if (!file.open(QFile::ReadOnly | QFile::Text)){  return;  }  QTextStream in(&file);  in.readLine(); // skip header  while (!in.atEnd()){  QString line = in.readLine();  QStringList values = line.split(";");  record.setValue("dGamma", values[0]);  record.setValue("Alpha", values[1]);  record.setValue("Beta", values[2]);  record.setValue("experiment\_id", expuuid);  model->insertRecord(-1, record);  }  file.close();  model->submitAll();  }  void ExperimentsWindow::openAddDialog()  {  NewExperimentDialog nedlg;  connect(&nedlg, &NewExperimentDialog::importFiles, this, &ExperimentsWindow::importExperimentData);  if (nedlg.exec() == NewExperimentDialog::Accepted){  modelExp->select();  ui->exprimentsTable->reset();  }  }  void ExperimentsWindow::deleteExperiment()  {  if (ui->exprimentsTable->selectionModel()->selectedRows().isEmpty()){  return;  }  int ret = QMessageBox::warning(this, "Удаление", "Удалить эксперимент?",  QMessageBox::Ok | QMessageBox::Cancel);  if (ret == QMessageBox::Ok){  int row = ui->exprimentsTable->selectionModel()->selectedRows().back().row();  modelExp->removeRow(row);  modelExp->submitAll();  modelExp->select();  ui->exprimentsTable->reset();  }  }  void ExperimentsWindow::selectExperiment()  {  if (ui->exprimentsTable->selectionModel()->selectedRows().isEmpty()){  return;  }  int row = ui->exprimentsTable->selectionModel()->selectedRows().back().row();  QSqlRecord rec = modelExp->record(row);  emit experimentSelected(rec.value("obj\_uuid").toString(), rec.value("obj\_name").toString());  }  void ExperimentsWindow::importExperimentData(QString expuuid, QString frays, QString fc1, QString fc2, QString fc3)  {  importRaysData(expuuid, frays);  importC1Data(expuuid, fc1);  importC2Data(expuuid, fc2);  importC3Data(expuuid, fc3);  } Файл editexperimentdialog.cpp #include "editexperimentdialog.h"  #include "ui\_editexperimentdialog.h"  #include <QDebug>  EditExperimentDialog::EditExperimentDialog(QWidget \*parent) :  QDialog(parent),  ui(new Ui::EditExperimentDialog)  {  ui->setupUi(this);  this->setFixedWidth(300);  this->setModal(true);  this->is\_accepted = false;  connect(ui->buttonSave, &QPushButton::clicked, this, &EditExperimentDialog::save);  connect(ui->buttonCancel, &QPushButton::clicked, this, &EditExperimentDialog::close);  }  EditExperimentDialog::~EditExperimentDialog()  {  delete ui;  }  void EditExperimentDialog::initFieldData(QSqlRecord record)  {  ui->editName->setText(record.value("obj\_name").toString());  ui->editComment->setPlainText(record.value("obj\_comment").toString());  rec = record;  }  void EditExperimentDialog::save()  {  rec.setValue("obj\_name", ui->editName->text());  rec.setValue("obj\_comment", ui->editComment->toPlainText());  is\_accepted = true;  accept();  } Файл roweditingdelegate.cpp #ifndef ROWEDITINGDELEGATE\_H  #define ROWEDITINGDELEGATE\_H  #include <QStyledItemDelegate>  class RowEditingDelegate : public QStyledItemDelegate  {  public:  RowEditingDelegate();  QWidget \*createEditor(QWidget \*parent,const QStyleOptionViewItem &option,  const QModelIndex &index) const;  void setModelData(QWidget \*editor, QAbstractItemModel \*model, const QModelIndex &index) const;  void setEditorData(QWidget \*editor, const QModelIndex &index) const;  private slots:  void commitAndCloseEditor();  };  #endif // ROWEDITINGDELEGATE\_H Файл roweditingdelegate.cpp #include "roweditingdelegate.h"  #include "editexperimentdialog.h"  #include <QSqlRecord>  #include <QSqlTableModel>  RowEditingDelegate::RowEditingDelegate()  {  }  QWidget \*RowEditingDelegate::createEditor(QWidget \*parent,  const QStyleOptionViewItem &option,  const QModelIndex &index) const  {  Q\_UNUSED(parent);  Q\_UNUSED(option);  Q\_UNUSED(index);  EditExperimentDialog \*dlg = new EditExperimentDialog();  return dlg;  }  void RowEditingDelegate::setEditorData(QWidget \*editor, const QModelIndex &index) const  {  QSqlRecord rec = qobject\_cast<const QSqlTableModel \*>(index.model())->record(index.row());  EditExperimentDialog \*dlg = qobject\_cast<EditExperimentDialog \*>(editor);  dlg->initFieldData(rec);  }  void RowEditingDelegate::setModelData(QWidget \*editor,  QAbstractItemModel \*model,  const QModelIndex &index) const  {  EditExperimentDialog \*dlg = qobject\_cast<EditExperimentDialog \*>(editor);  if (dlg->is\_accepted){  QSqlTableModel \*mod = static\_cast<QSqlTableModel \*>(model);  mod->setRecord(index.row(), dlg->rec);  mod->submitAll();  }  }  void RowEditingDelegate::commitAndCloseEditor()  {  EditExperimentDialog \*editor = qobject\_cast<EditExperimentDialog \*>(sender());  emit commitData(editor);  emit closeEditor(editor);  } Файл plotwidget.h #ifndef PLOTWIDGET\_H  #define PLOTWIDGET\_H  #include <QWidget>  #include <QButtonGroup>  #include <QAbstractButton>  #include <QtDataVisualization/Q3DSurface>  namespace Ui {  class PlotWidget;  }  class PlotWidget : public QWidget  {  Q\_OBJECT  public:  explicit PlotWidget(QString experiment\_id, QWidget \*parent = nullptr);  ~PlotWidget();  private:  Ui::PlotWidget \*ui;  QWidget \*container;  QButtonGroup \*group;  QString experiment;  void plot(QString argument);  void setupButtonGroup();  private slots:  void replot(QAbstractButton \*button);  };  #endif // PLOTWIDGET\_H Файл plotwidget.cpp #include "plotwidget.h"  #include "ui\_plotwidget.h"  #include <QSqlQueryModel>  #include <QSqlRecord>  #include <QDebug>  #include <QSqlError>  #include <QSqlQuery>  using namespace QtDataVisualization;  PlotWidget::PlotWidget(QString experiment\_id, QWidget \*parent) :  QWidget(parent),  ui(new Ui::PlotWidget),  experiment(experiment\_id)  {  ui->setupUi(this);  setupButtonGroup();  plot("J");  }  PlotWidget::~PlotWidget()  {  delete ui;  }  void PlotWidget::plot(QString argument)  {  QSurfaceDataArray \*data = new QSurfaceDataArray();  QSqlQueryModel model = QSqlQueryModel();  model.setQuery(QString("select distinct \"Alfa\" from calculation\_12 where \"experiment\_id\" = '%1' order by \"Alfa\" asc").arg(experiment),  QSqlDatabase::database("stokes\_db"));  for (int i = 0; i < model.rowCount(); i++){  QSurfaceDataRow \*row = new QSurfaceDataRow();  QSqlRecord rec = model.record(i);  double alfa = rec.value("Alfa").toDouble();  QSqlQuery q = QSqlQuery(QString("select \"Beta\", \"%1\" from calculation\_12 "  "where \"Alfa\" = %2 and \"experiment\_id\" = '%3' "  "order by \"Beta\" asc;")  .arg(argument)  .arg(alfa)  .arg(experiment),  QSqlDatabase::database("stokes\_db"));  while(q.next()){  double beta = q.value("Beta").toDouble();  double arg = q.value(argument).toDouble();  qDebug() << alfa << beta << arg;  QSurfaceDataItem item1 = QSurfaceDataItem();  item1.setX(beta);  item1.setY(arg);  item1.setZ(alfa);  row->append(item1);  }  data->append(row);  }  QSurface3DSeries \*series = new QSurface3DSeries();  series->dataProxy()->resetArray(data);  QLinearGradient gr;  gr.setColorAt(0.0, Qt::black);  gr.setColorAt(0.33, Qt::blue);  gr.setColorAt(0.67, Qt::red);  gr.setColorAt(1.0, Qt::yellow);  series->setBaseGradient(gr);  series->setColorStyle(Q3DTheme::ColorStyleRangeGradient);  series->setFlatShadingEnabled(false);  Q3DSurface \*surface = new Q3DSurface();  surface->addSeries(series);  surface->axisX()->setTitle("Beta");  surface->axisX()->setTitleVisible(true);  surface->axisY()->setTitle(argument);  surface->axisY()->setTitleVisible(true);  surface->axisZ()->setTitle("Alfa");  surface->axisZ()->setTitleVisible(true);  surface->axisZ()->setReversed(true);  surface->setShadowQuality(Q3DSurface::ShadowQualityNone);  container = QWidget::createWindowContainer(surface);  ui->horizontalLayout->addWidget(container);  container->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);  container->show();  }  void PlotWidget::setupButtonGroup()  {  group = new QButtonGroup();  group->addButton(ui->rbJ);  group->addButton(ui->rbJ0);  group->addButton(ui->rbP);  group->addButton(ui->rbP0);  group->addButton(ui->rbQ);  group->addButton(ui->rbQ0);  group->addButton(ui->rbU);  group->addButton(ui->rbU0);  group->addButton(ui->rbV);  group->addButton(ui->rbV0);  connect(group, SIGNAL(buttonClicked(QAbstractButton\*)), this, SLOT(replot(QAbstractButton \*)));  }  void PlotWidget::replot(QAbstractButton \*button)  {  delete container;  QString parameter = button->text();  plot(parameter);  } Файл constanswindow.h #ifndef CONSTANTSWINDOW\_H  #define CONSTANTSWINDOW\_H  #include <QWidget>  #include <QSqlTableModel>  namespace Ui {  class ConstantsWindow;  }  class ConstantsWindow : public QWidget  {  Q\_OBJECT  public:  explicit ConstantsWindow(QWidget \*parent = nullptr);  ~ConstantsWindow();  private:  Ui::ConstantsWindow \*ui;  QSqlTableModel \*model;  void setupTable();  };  #endif // CONSTANTSWINDOW\_H Файл constantswindow.cpp #include "constantswindow.h"  #include "ui\_constantswindow.h"  ConstantsWindow::ConstantsWindow(QWidget \*parent) :  QWidget(parent),  ui(new Ui::ConstantsWindow),  model(new QSqlTableModel(this, QSqlDatabase::database("stokes\_db")))  {  ui->setupUi(this);  setupTable();  }  ConstantsWindow::~ConstantsWindow()  {  delete ui;  }  void ConstantsWindow::setupTable()  {  model->setEditStrategy(QSqlTableModel::OnManualSubmit);  model->setTable("constants");  model->select();  ui->table->setModel(model);  ui->table->setColumnWidth(model->fieldIndex("obj\_descr"), 400);  ui->table->setSelectionBehavior(QAbstractItemView::SelectRows);  ui->table->setEditTriggers(QAbstractItemView::NoEditTriggers);  ui->table->hideColumn(0);  } Файл loaddatadialog.h #ifndef LOADDATADIALOG\_H  #define LOADDATADIALOG\_H  #include <QDialog>  namespace Ui {  class LoadDataDialog;  }  class LoadDataDialog : public QDialog  {  Q\_OBJECT  public:  explicit LoadDataDialog(QWidget \*parent = nullptr);  ~LoadDataDialog();  private:  Ui::LoadDataDialog \*ui;  void deleteOldData();  private slots:  void loadNewData();  void selectFileRays();  void selectFileC1();  void selectFileC2();  void selectFileC3();  };  #endif // LOADDATADIALOG\_H Файл loaddatadialog.cpp #include "loaddatadialog.h"  #include "ui\_loaddatadialog.h"  #include <QMessageBox>  #include <QFileDialog>  #include <QFile>  #include <QTextStream>  LoadDataDialog::LoadDataDialog(QWidget \*parent) :  QDialog(parent),  ui(new Ui::LoadDataDialog)  {  ui->setupUi(this);  connect(ui->buttonCancel, &QPushButton::clicked, this, &LoadDataDialog::close);  connect(ui->buttonC1, &QPushButton::clicked, this, &LoadDataDialog::selectFileC1);  connect(ui->buttonC2, &QPushButton::clicked, this, &LoadDataDialog::selectFileC2);  connect(ui->buttonC3, &QPushButton::clicked, this, &LoadDataDialog::selectFileC3);  connect(ui->buttonRays, &QPushButton::clicked, this, &LoadDataDialog::selectFileRays);  }  LoadDataDialog::~LoadDataDialog()  {  delete ui;  }  void LoadDataDialog::deleteOldData()  {  }  void LoadDataDialog::loadNewData()  {  int ret = QMessageBox::warning(this, "Загрузка", "Старые данные будут удалены\nПродолжить?",  QMessageBox::Ok | QMessageBox::Cancel);  if (ret == QMessageBox::Ok){  }  }  void LoadDataDialog::selectFileRays()  {  QString filename = QFileDialog::getOpenFileName(this, "Open", "C:\\Users\\User\\Documents\\",  "CSV (\*.csv) ;; Text (\*.txt) ;; All files (\*.\*)");  ui->editRays->setText(filename);  }  void LoadDataDialog::selectFileC1()  {  QString filename = QFileDialog::getOpenFileName(this, "Open", "C:\\Users\\User\\Documents\\",  "CSV (\*.csv) ;; Text (\*.txt) ;; All files (\*.\*)");  ui->editC1->setText(filename);  }  void LoadDataDialog::selectFileC2()  {  QString filename = QFileDialog::getOpenFileName(this, "Open", "C:\\Users\\User\\Documents\\",  "CSV (\*.csv) ;; Text (\*.txt) ;; All files (\*.\*)");  ui->editC2->setText(filename);  }  void LoadDataDialog::selectFileC3()  {  QString filename = QFileDialog::getOpenFileName(this, "Open", "C:\\Users\\User\\Documents\\",  "CSV (\*.csv) ;; Text (\*.txt) ;; All files (\*.\*)");  ui->editC3->setText(filename);  } Файл logfilewidget.h #ifndef LOGFILEWIDGET\_H  #define LOGFILEWIDGET\_H  #include <QWidget>  namespace Ui {  class LogFileWidget;  }  class LogFileWidget : public QWidget  {  Q\_OBJECT  public:  explicit LogFileWidget(QString file, QWidget \*parent = nullptr);  ~LogFileWidget();  private:  Ui::LogFileWidget \*ui;  QString filename;  void openFile();  };  #endif // LOGFILEWIDGET\_H Файл logfilewidget.cpp #include "logfilewidget.h"  #include "ui\_logfilewidget.h"  #include <QFile>  #include <QTextStream>  #include <QDebug>  LogFileWidget::LogFileWidget(QString file, QWidget \*parent) :  QWidget(parent),  ui(new Ui::LogFileWidget),  filename(file)  {  ui->setupUi(this);  openFile();  }  LogFileWidget::~LogFileWidget()  {  delete ui;  }  void LogFileWidget::openFile()  {  QFile file(filename);  ui->plainTextEdit->clear();  if (!file.open(QIODevice::ReadOnly | QIODevice::Text)){  qDebug() << file.errorString();  }  QTextStream in(&file);  ui->plainTextEdit->setPlainText(in.readAll());  file.close();  } |
| Приложение Б. Протокол 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 В. Протокол 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 Г. Протокол 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 Д. Протокол 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