Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Иркутский государственный университет»
(ФГБОУ ВО «ИГУ»)
Институт математики и информационных технологий
Кафедра алгебраических и информационных систем

ОТЧЕТ
ПО УЧЕБНОЙ ПРАКТИКЕ

Магистрант 2 курса очного отделения
Группа 02271–ДМ
Малеев Владислав Викторович

Руководитель практики:
к.ф.-м.н., доцент Рябец Л. В.

Иркутск 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **3**

Раздел 1. Разработка проекта **5**

1.1. Варианты использования 5

1.2. Архитектура проекта 6

Раздел 2. Описание реализации проекта **10**

2.1. Сравнение аналитической и численной моделей 10

2.2. Сравнение двух численных моделей 13

2.3. Оценка рисков 14

Раздел 3. Дальнейшее развитие проекта **17**

3.1. План испытаний 17

3.2. Документирование 19

3.3. Функции для будущих версий 20

ЗАКЛЮЧЕНИЕ **21**

ПРИЛОЖЕНИЕ 1. Метод Рунге-Кутты для системы дифференциальных уравнений высших порядков **22**

ПРИЛОЖЕНИЕ 2. Интерфейс для запуска численных моделей **23**

ВВЕДЕНИЕ

В физике электромагнитного взаимодействия, при разработке модели траектории полёта заряженной частицы в магнитном поле часто используют численные методы, для которых требуется оценка точности этих методов. Причём крайне желательно проводить оценку точности не просто на заранее заготовленных входных данных, а на данных, приближенные к реальным.

В таком случае можно создать программу, принимающая реализацию некоторой численной модели заряженной частицы и оценивающая точность её вычислений с аналитическим решением. Оценка точности должна происходить по модели магнитного поля реального объекта. В качестве такового предлагается взять магнитное поле Земли, так как оно наиболее лучше исследовано.

Результат сравнения аналитической и численной модели должен выражаться через различные статистические характеристики, такие как максимум, минимум, средняя разницы в результатах и так далее. Помимо этого, крайне желательна возможность демонстрации траектории частицы, полученной с помощью моделей.

Решением такого запроса является создание программы, в которой уже реализовано точное аналитическое решение, модель среды, на которой происходит испытания, и интерфейсы для обмена данными и их визуализации. В таком случае пользователю остаётся программно реализовать свою модель полёта заряженной частицы, соединить с интерфейсом программы и получить требуемые результаты.

Альтернативой для этого проекта является реализация собственной аналитической модели полёта заряженной частицы и сравнения с ней. В таком случае возможно использование более лучшего алгоритма, выдающего более точные результаты, по сравнению с предложенной программой. Но в тоже время реализация данного проекта упрощает тестирование новой модели, а так-

же позволяет более объективно сравнивать разные алгоритма на одних и тех же входных данных.

Таким образом, основными функциями описанной выше программы являются:

- Запуск численной модели полёта частицы в магнитном поле на основе данных, переданных моделью магнитного поля.
- Определение точности численной модели на основе аналитического решения на одних и тех же входных данных.
- Отображение на графиках и сохранение в файл траектории моделей полёта частицы.

Раздел 1. Разработка проекта

1.1. Варианты использования

Вариант использования, показанный на рис. 1, соответствует процессу сравнения численной модели пользователя со встроенной аналитической моделью и получения результатов этого сравнения. Сравнение происходит при помощи сопоставления траекторий моделей. Также подразумевается возможность визуализации траектории частицы, полученной как аналитической, так и численной моделями.

Рисунок 1. Сравнение аналитической и численной моделей

Вариант использования, показанный на рис. 2, соответствует процессу сравнения численной модели с другой численной моделью пользователя и демонстрации результатов этого сравнения. Также как и в предыдущем варианте использования, сравнение происходит при помощи сопоставления траекторий моделей. Полученные модели могут быть отображены на графиках.

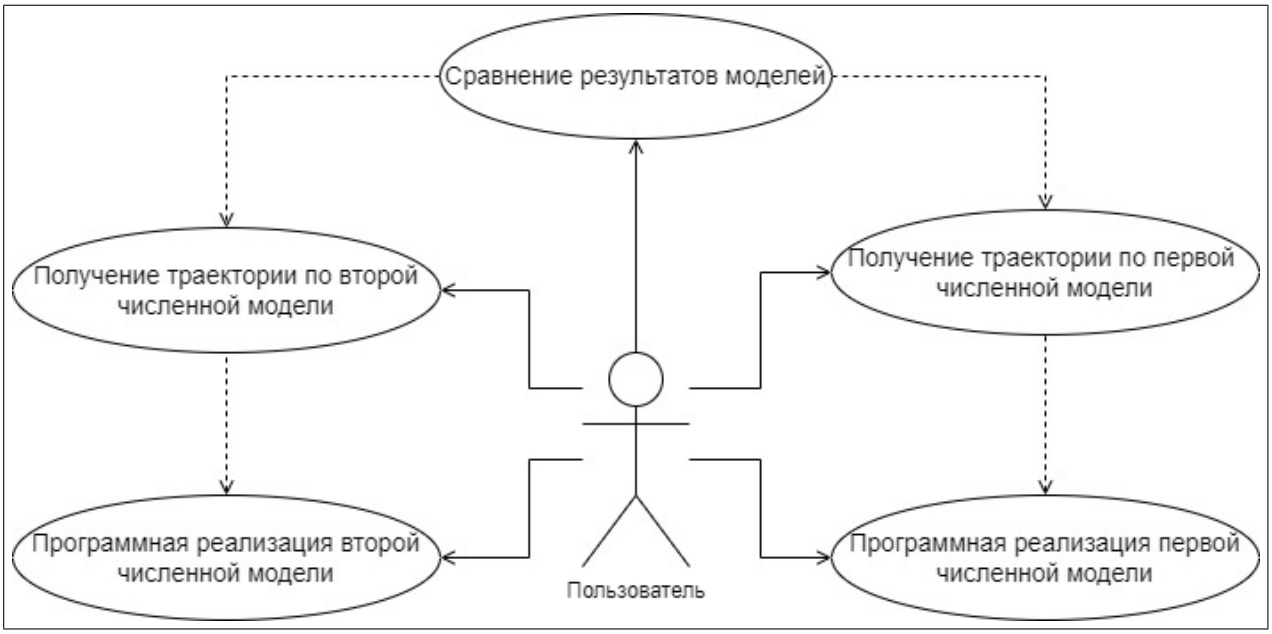

Рисунок 2. Сравнение двух численных моделей

1.2. Архитектура проекта

Программа содержит следующие модули:

- Координаты траектории полёта заряженной частицы.
- Модель магнитного поля.
- Аналитическая модель частицы.
- Численная модель частицы.

Модуль координат траектории полёта заряженной частицы имеет интерфейс для записи положения некоторой частицы в различные моменты времени. Положение частицы отслеживается через последовательные и равные промежутки времени, поэтому запись положения частицы производиться как путём последовательного добавления в список передаваемых координат, так и записи всего вычисленного списка координат целиком. Запись координат возможна как в декартовой, так и в полярной системе координат. Система координат выбирается пользователем самостоятельно перед запуском своей модели и не меняется до её завершения. Имеется возможность перевода координат из одной системы

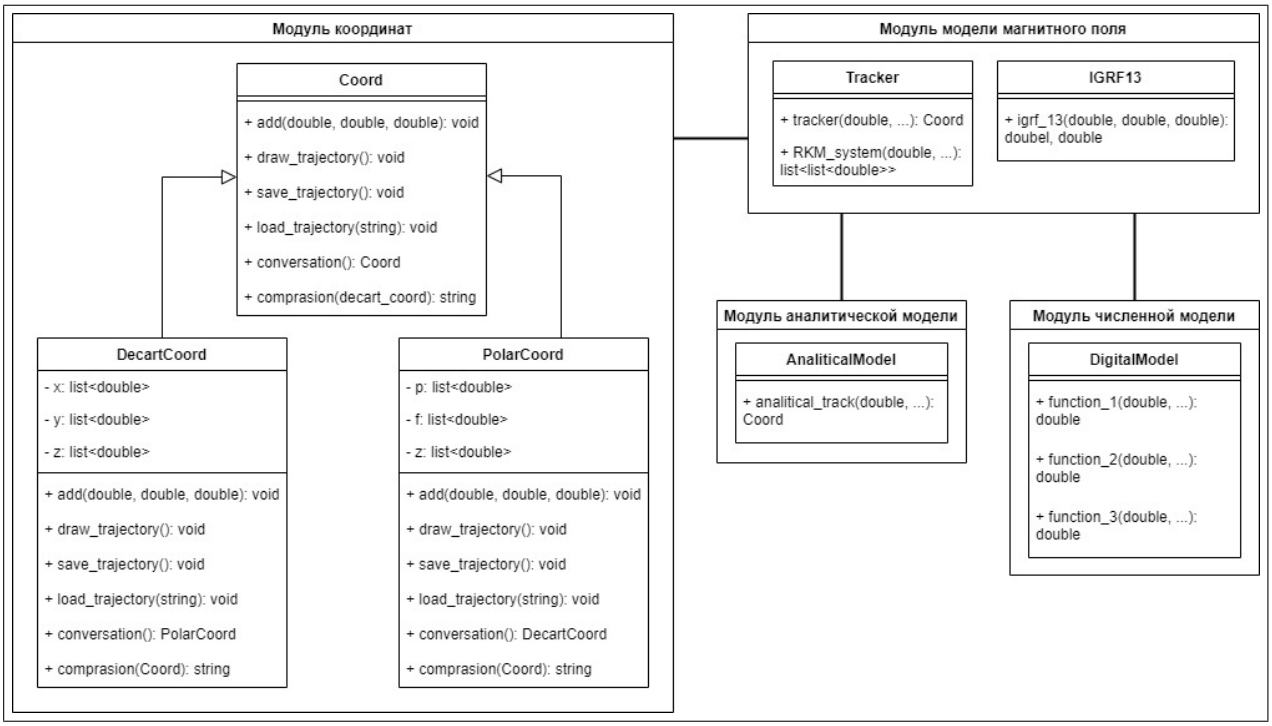

Рисунок 3. UML диаграмма классов

в другую. Помимо этого, в модуле реализована возможность записи в файл и чтения из него траектории полёта частицы и её отрисовки. Эти возможности реализована с помощью уже разработанных библиотек. Также есть возможность сравнить траектории двух различных моделей с одинаковыми промежутками времени отслеживания частицы, возвращая в качестве результата максимальное и среднее значения различий в положениях частицы в одинаковые моменты времени.

Модуль модели магнитного поля реализуется с помощью программы International Geomagnetic Reference Field 13 (IGRF-13), которая является моделью магнитного поля Земли. Данный модуль обеспечивает интерфейс между IGRF-13 и моделями заряженных частиц. IGRF-13 на вход принимает координаты положения частицы и возвращает параметры магнитного поля. Модель частицы принимает на вход текущее параметры частицы и магнитного поля и рассчитывает следующие параметры частицы, включая положение. Таким образом указанный выше интерфейс принимает на вход начальные параметры

частицы, по полученному положению частицы рассчитывает параметры магнитного поля через модель оного, и уже по всем этим параметрам через модель частицы рассчитываются её следующие параметры, после этого алгоритм повторяется. Перечисленные шаги повторяются указанное при запуске количество раз. Все рассчитанные положения частицы с помощью модуля координат записываются в файл.

Модуль аналитической модели реализует по аналитической формуле модель частицы. Эта модель по входным параметрам рассчитывает параметры частицы в следующий момент времени. Так как данный модуль рассчитывает точное положение частицы, то время его выполнения может быть довольно продолжительным.

Численная модель частицы реализуется самим пользователем. Для этого в модуле численной модели реализован интерфейс, с помощью которого пользователь может связать свою модель частицы с модулем магнитного поля. Модель частицы представляет собой систему уравнений, поэтому программной реализацией модели является написание функций, выполняющие вычисление по уравнениям модели пользователя и принимающие на вход необходимые аргументы, такие как значение текущего момента времени, положение, скорость, величина заряда и масса частицы, значение индукции и напряжённость магнитного поля и угол между ними. Для универсальности использования, функции принимают на вход все эти аргументы, даже если они не используются. Эти функции в дальнейшем передаются в качестве аргументов в функцию запуска модели. Для случаев, когда модель частицы описывается системой дифференциальных уравнений, реализован метод Рунге-Кутты для систем дифференциальных уравнений высших порядков.

Все данные программа хранит в файлах с помощью сторонних библиотек.

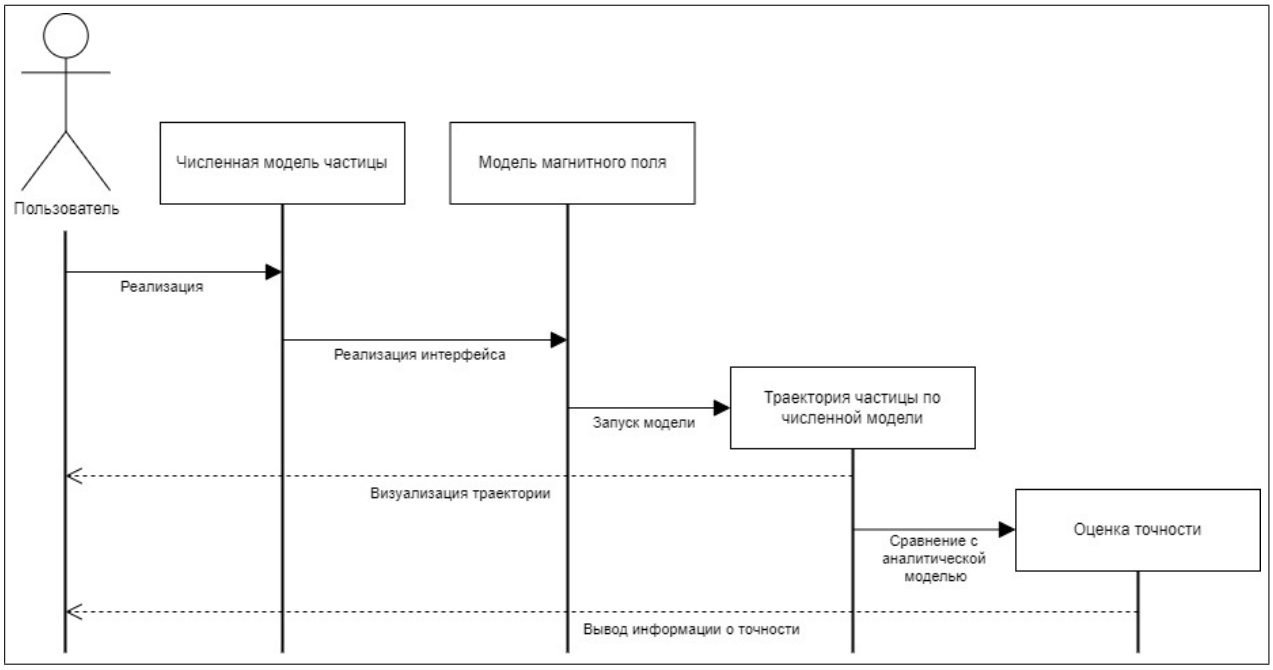

Рисунок 4. UML диаграмма последовательности оценки точности модели

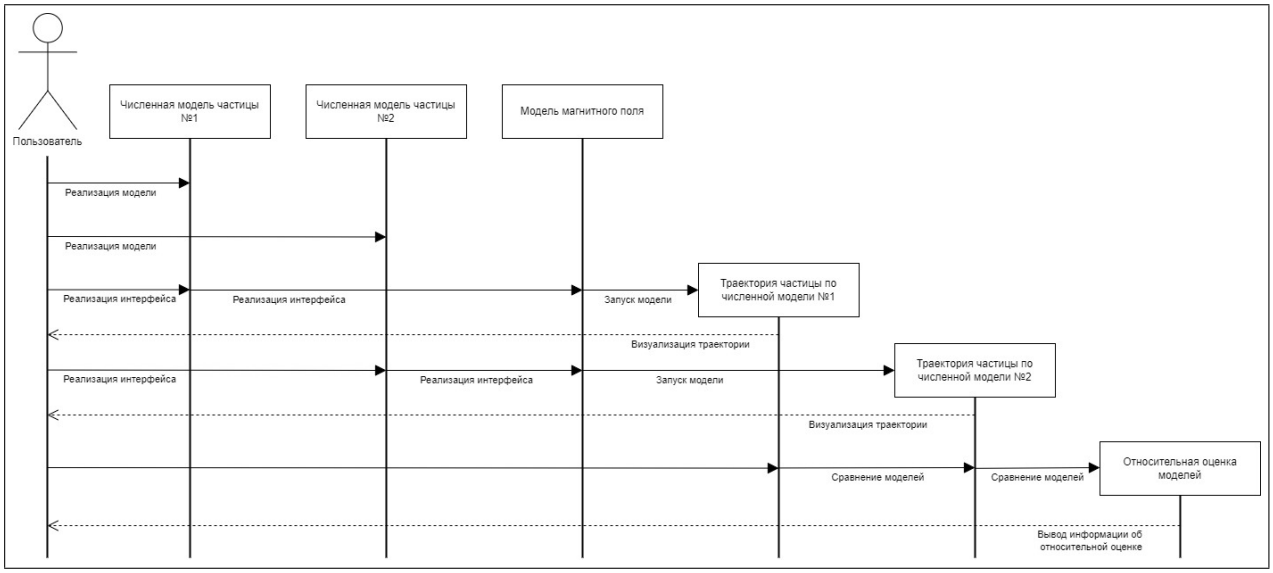

Рисунок 5. UML диаграмма последовательности сравнения двух моделей

Раздел 2. Описание реализации проекта

Документом по стилю кодирования выступает руководство PEP8[1].

2.1. Сравнение аналитической и численной моделей

Вначале пользователю требуется программно реализовать свою численную модель движения заряженной частицы в магнитном поле. Обычно, модель движения частицы описывается системой уравнений, где каждое уравнение соответствует изменению одного из параметров выбранной системы координат при движении частицы. Для примера, возьмём модель в декартовой системе координат, представленную системой уравнений (1).

$$\begin{cases} x'' = \dfrac{q}{m}(y' B \sin\alpha - z' B \cos\alpha) \\ y'' = \dfrac{q}{m} E - \dfrac{q}{m} x' B \sin\alpha \\ z'' = \dfrac{q}{m} x' B \cos\alpha \end{cases} \tag{1}$$

Реализацией является создание отдельной функции для каждого вышеописанного уравнения. Для универсальности, каждая такая функция должна принимать в качестве аргументов все аргументы, такие как значение текущего момента времени, положение, скорость, величина заряда и масса частицы, значение индукции и напряжённость магнитного поля и угол между ними, даже если они не используются. Реализация модели (1) представлена на листинге 1.

[1] https://pep8.org

Листинг 1. Реализация численной модели

```
def x_function(t, x, y, z, vx, vy, vz,  x_0, y_0, z_0, vx_0, vy_0, vz_0,
               q, m, E, B, alpha, angle_in_degrees = True):
  alpha = degree_to_rad(alpha, angle_in_degrees)
  return (q*B)/m * ( vy * math.sin(alpha) - vz * math.cos(alpha) )

def y_function(t, x, y, z, vx, vy, vz, x_0, y_0, z_0, vx_0, vy_0, vz_0,
               q, m, E, B, alpha, angle_in_degrees = True):
  alpha = degree_to_rad(alpha, angle_in_degrees)
  return q/m * E - q/m *vx * B * math.sin(alpha)

def z_function(t, x, y, z, vx, vy, vz, x_0, y_0, z_0, vx_0, vy_0, vz_0,
               q, m, E, B, alpha, angle_in_degrees = True):
  alpha = degree_to_rad(alpha, angle_in_degrees)
  return q/m * vx * B * math.cos(alpha)
```

Далее, необходимо запустить модель. Вычисление траектории происходит с помощью функции RKM_system(), реализующая метод Рунге-Кутты для системы дифференциальных уравнений высших порядков и представленная в приложении 1. Значения переменных магнитного поля вычисляются с помощью модели IGRF-13 по координатам в декартовой системе координат. Чтобы запустить модель нужно объединить все три компонента. Это происходит с помощью функции tracker(). Данная функция создаёт оболочку над функциями модели частицы таким образом, чтобы принимать на вход переменные, используемые в методе Рунге-Кутты, преобразовывать их в параметры частицы, а также подставлять параметры модели магнитного поля по IGRF-13. Полный код функции tracker() представлен в приложении 2.

После выполнения функции tracker(), пользователю возвращается координаты траектории движения частицы по переданной модели. Эта траектория представлена классом DcardCoord или PolarCoord, в зависимости от выбора пользователя. Далее пользователь может сохранить траекторию в файл,

отобразить на графике, конвертировать в другую систему координат или сравнить с другой моделью. Пример отображения траектории частицы на графике является рис. 6.

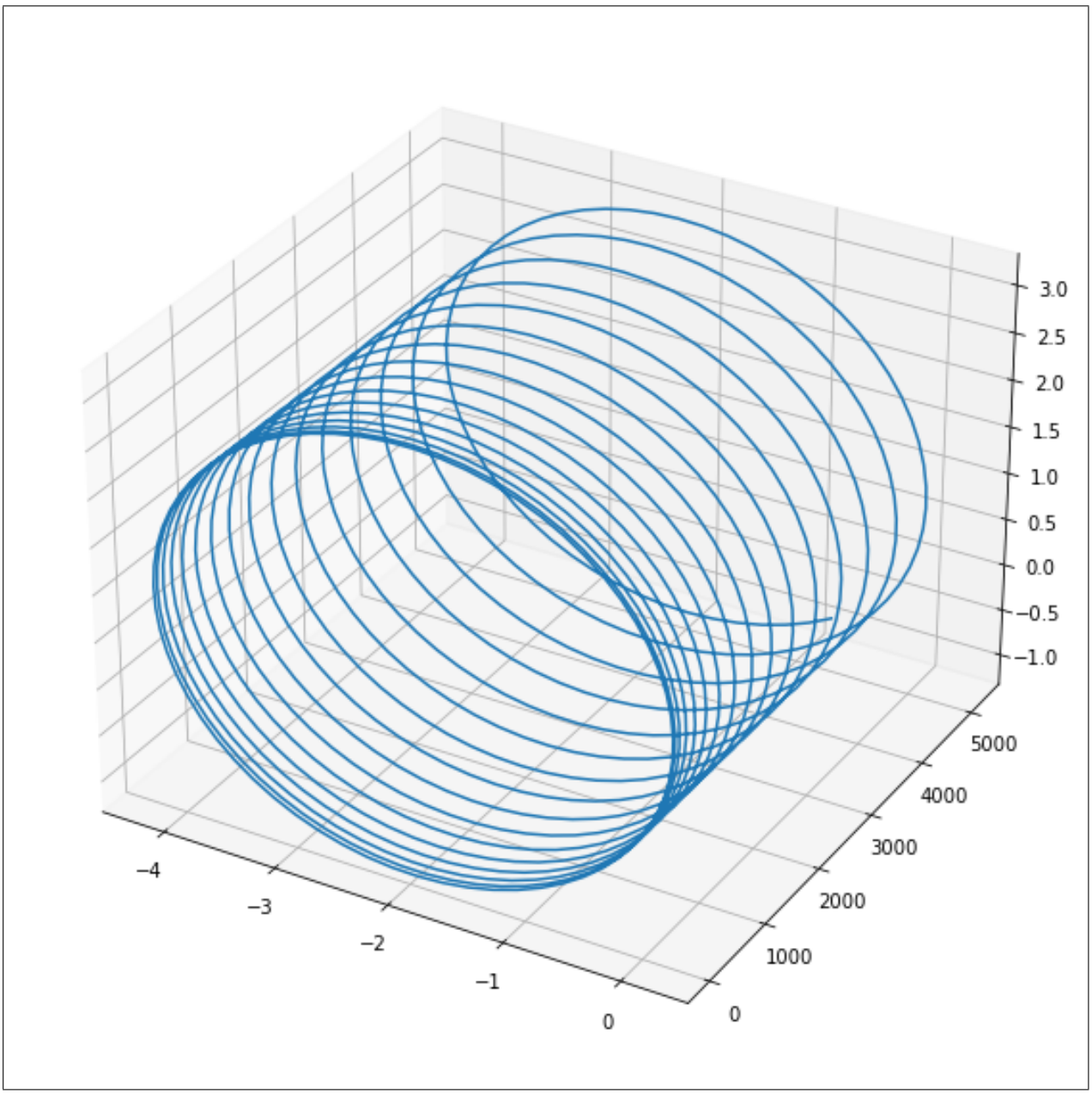

Рисунок 6. График траектории заряженной частицы

Чтобы получить точность численной модели её необходимо сравнить с аналитической моделью. Для этого требуется получить траекторию частицы по аналитической модели, вызвав функцию analitical_track() с аналогичными аргументами, использовавшимися для численной модели. Далее необходимо ис-

пользовать функцию сравнения comprasion() из класса Coord для получения оценки. Пример результата сравнения показан на рис. 7.

max deviation		
x	y	z
-0.0001854	1.728e-11	0.0001823
average deviation		
x	y	z
-8.054e-07	6.108e-12	1.691e-06

Рисунок 7. Результат сравнения численной модели с аналитической

2.2. Сравнение двух численных моделей

Сравнение двух численных моделей происходит схожим образом, описанным выше. В начале требуется реализовать численные модели, которые пользователь хочет сравнить. Модели также реализуются в виде трёх отдельных функций, по одному на каждый параметр системы счисления. Далее выполняется запуск этих моделей с помощью функции tracker() и получение траекторий частиц. Если вычисление проводились в разных системах координат, то траектории приводятся к одной системе, причём следует учитывать, что при приведении к другой системе координат возможна потеря точности. В конце необходимо использовать функцию сравнения comprasion() из класса Coord для получения относительной оценки моделей. Результатом сравнения выглядит аналогичным образом, что и для сравнения с аналитической моделью и показан на рис. 7.

2.3. Оценка рисков

Р1. *Ошибка в аналитической модели.* Ошибка в аналитической модели может способствовать уменьшению демонстрируемой точности на хороших численных моделях и, наоборот, увеличению на плохих. Ввиду того, что аналитическая модель состоит из множества различных элементов, в которых легко допустить ошибку, данный риск имеет среднюю вероятность возникновения. В результате возникновения данного риска одна из основных функций данного проекта будет не доступна, поэтому риск имеет высокое влияние на проект. Для того, чтобы уменьшить вероятность возникновения данного риска следует более тщательно реализовывать аналитическую модель и проводить большее тестирование данной области. В случае реализации риска следует как можно скорее найти и исправить ошибку.

Р2. *Программная несовместимость модулей.* Возможна возникновение ситуации, когда параметры модели частицы и модели магнитного поля различаются. Так как параметры движения заряженной частицы в магнитном поле для подавляющего большинства физических моделей одни и те же, то вероятность возникновения такого риска довольно низкая. Однако при его возникновении программа не сможет выполнять все свои функции для данной модели, а потому влияние данного риска высоко. Для уменьшения вероятности возникновения такого риска следует изучить новые современные модели магнитного поля и заряженных частиц в нём и включить дополнительные параметры в работу модулей. При возникновении данного риска для минимизации последствий следует в наикратчайшие сроки добавить в модули недостающие параметры.

Р3. *Ошибка в сохранении данных.* При сохранении данных в файл или чтении из него может возникнуть какая-либо ошибка. Поскольку для записи данных используются сторонние библиотеки, то вероятность возникновения риска мала. Помимо этого, без этой функции программа, так или иначе, сможет вы-

полнять свои основные функции, поэтому влияние такого риска низкое. Для уменьшения вероятности данного риска следует провести тестирование записи и чтения данных, а для уменьшения влияния следует использовать альтернативные дублирующие библиотеки.

Р4. *Пользователь может не разобраться в документации.* Так как для работы программы необходима численная модель заряженной частицы, которую реализует сам пользователь, появляется риск того, что пользователь не разберётся в документации к программе и не сможет правильно создать свою модель. Программа рассчитана в первую очередь на людей с необходимыми навыками, но также может попасть в руки к неопытным пользователям, поэтому риск имеет среднюю вероятность возникновения. Для уменьшения вероятности возникновения можно добавить соответствующий раздел в руководство пользователя.

Р5. *Ошибки в модели магнитного поля.* В самой модели магнитного поля возможно возникновение каких-либо ошибок, из-за которых модель возвращает неверные данные. Ввиду того, что разработкой модели IGRF-13 занимается профессиональное сообщество, вероятность возникновение в ней ошибки, а, следовательно, и риска низкая. При возникновении такой ошибки, и численная модель, и аналитическая будут иметь её в своих выходных данных, возможно, компенсируя друг друга при получении сравнительных данных, поэтому данный риск может оказать среднее воздействие на проект. Для уменьшения вероятности риска следует провести предварительное тестирования модели на заранее подготовленных данных. При возникновении риска для смягчения последствий следует как можно быстрее сообщить о проблеме разработчикам модели.

Р6. *Модуль координаты траектории частицы будет давать дополнительную ошибку.* Учитывая особенности хранения вещественных чисел в цифровом виде, при записи координат траектории частицы в соответствующий модуль возможна потеря точности. Так как вычисления происходят на веществен-

ных числах с высоким порядком, вероятность возникновения риска высока. Тем не менее, на большинстве вычислений такие потери отражаются слабо, поэтому влияние данного риска низкое. Для уменьшения влияния риска для вычислений можно использовать вещественные числа с максимально доступным порядком.

Раздел 3. Дальнейшее развитие проекта

3.1. План испытаний

Для создания тестов следует использовать библиотеку unittest.

Юнит-тесты

T1.1. Тест аналитической модели.

Что проверять: Корректность результатов аналитической модели.

Как проверять: Необходимо сравнить результаты на тестовых данных с предварительно подготовленными результатами, полученными подсчётом вручную.

Когда реализовать тесты: До реализации аналитического модуля.

Когда запускать тесты: При каждом внесении изменении в часть аналитического модели, ответственную за вычисления результата.

T1.2 Тест модуля координат. Корректность добавления координат.

Что проверять: Правильно ли добавляются новые значения координат.

Как проверять: Следует создать тестовый набор координат, которые необходимо будет последовательно добавлять в экземпляр модуля координат, запросить обратно набор координат из этого экземпляра и сравнить с исходным набором.

Когда реализовать тесты: До реализации функций добавления и получения координат в модуле координат.

Когда запускать тесты: Единожды, после реализации функций добавления и получения координат.

T1.3 Тест модуля координат. Отрисовка траектории.

Что проверять: Правильно ли модуль координат отрисовывает траекторию.

Как проверять: Следует предварительно создать тестовый набор координат траектории частицы и отобразить его на графике, после чего передать этот же набор в модуль координат и так же отобразить на графике, после чего сравнить изображения.

Когда реализовать тесты: До реализации функции отрисовки траектории в модуле координат.

Когда запускать тесты: Единожды, после реализации функции отрисовки траектории.

Т1.4 Тест модуля координат. Сохранение координат.

Что проверять: Правильно ли модуль координат сохраняет координаты.

Как проверять: Следует предварительно создать тестовый набор координат и сохранить в файл, после чего передать этот же набор в модуль координат и так же сохранить в файл, после чего сравнить файлы.

Когда реализовать тесты: До реализации функции сохранение координат в модуле координат.

Когда запускать тесты: Каждый раз, после реализации очередной функции для сохранения координат.

Интеграционные тесты

Т2.1 Тест вычисления, визуализации и сохранения аналитического решения.

Что проверять: Корректно ли программа находит аналитическое решение, отображает его и сохраняет в файл.

Как проверять: Используя соответствующую функцию модуля аналитической модели, запустить поиск и сохранения аналитического решения. Так как корректность отдельных частей этого теста должная проверятся при юнит-тестировании, задача этого теста состоит в проверке правильной реализации интерфейсов между разными модулями.

Когда реализовать тесты: После окончательной реализации сигнатур всех необходимых функции.

Когда запускать тесты: Единожды, в конце проекта для проверки интерфейсов.

Т2.2 Тест получения, визуализации и сохранения численного решения.

Что проверять: Корректно ли программа использует пользовательские функции численной модели, отображает результаты и сохраняет в файл траекторию.

Как проверять: Для проведения этого теста необходимо реализовать модуль численной модели. Для этого можно либо реализовать простую модель, либо использовать аналитическую модель в качестве отдельного модуля. Далее реализованный модуль следует соединить с основной программой с помощью интерфейсов и запустить поиск решения по модели, сравнения с аналитической моделью, визуализации и сохранения траектории в файл.

Когда реализовать тесты: После окончательной реализации сигнатур всех необходимых функции.

Когда запускать тесты: Единожды, в конце проекта для проверки интерфейсов.

Юзабилити-тесты

Т3.1 Тест удобности использования интерфейсов для численной модели.

Что проверять: Понятно и удобно ли пользователь использовать интерфейсы программы.

Как проверять: Реализовать простую численную модель и попытаться связать её с различными методами программы. Оценить, насколько быстро находится нужное описание и реализуется.

Когда реализовать тесты: После реализации всех компонентов проекта.

Когда запускать тесты: Единожды, в конце проекта.

3.2. Документирование

В качестве документации к программе необходимо подготовить руководство пользователя, в которой расписаны все методы, необходимые для получе-

ния требуемых результатов пользователем.

3.3. Функции для будущих версий

Графический редактор формул численной модели полёта частиц

Так как для программной реализации моделей заряженных частиц требуется некоторая компетенция в сфере программирования, то существует запрос на создание графического редактора для ввода формул, описывающих движение частицы. Такой редактор должен давать возможность пользователю ввести формулы либо в математическом видел, либо с помощью специальных операторов, достаточно простых для понимания неподготовленному пользователю. На основе этих формул программа должна создавать функции на языке Python и использовать в качестве функций модели частицы, ранее создаваемых пользователем вручную.

Сохранение промежуточных результатов при выполнении вычислений

Вычисления траектории частицы может занимать очень большое количество времени, а потому имеется ненулевая вероятность того, что при незапланированном завершении выполнения кода, вследствие тех или иных событий, будут потерянны результаты уже выполненных вычислений. Поэтому, чтобы избежать подобного, необходимо создать возможность сохранения промежуточных результатов и начала выполнения вычислений с последней точки сохранения. Промежуток времени, через которое будет выполняться промежуточное сохранение, должно задаваться по умолчанию, с возможностью изменения пользователем. Также для этих сохранений должны быть возможность загрузки их в качестве координат траекторий частиц и выполнении всех соответствующих функций.

ЗАКЛЮЧЕНИЕ

В результате прохождения учебной практики были получены следующие результаты:

- реализована аналитическая модель движения частиц в магнитном поле;
- реализован класс для хранения координат траекторий частиц и манипуляций над ними;
- реализован интерфейс для запуска моделей частиц с моделью магнитного поля Земли IGRF-13;
- создан документ, описывающий архитектуру проекта.

ПРИЛОЖЕНИЕ 1.

Метод Рунге-Кутты для системы дифференциальных уравнений высших порядков

```
def RKM_system(functions, t_0, Cs_init, step, steps_count):
  Cs = np.array(Cs_init)
  Ks = np.zeros(Cs.shape + (4,))

  def append_empty(array, element):
    res = np.empty(array.shape[0] + 1)
    res[:-1] = array
    res[-1] = element
    return res

  def f(t_loc, Cs_loc):
    return np.array([ append_empty(C_loc[1:], function(t_loc, Cs_loc))
                      for C_loc, function in zip(Cs_loc, functions) ])

  res = [0] * (steps_count+1)
  res[0] = Cs_init.copy()
  t = t_0
  for i in range(steps_count):
    Ks[:, :, 0] = step * f(t,          Cs                )
    Ks[:, :, 1] = step * f(t + step/2, Cs + Ks[:, :, 0]/2)
    Ks[:, :, 2] = step * f(t + step/2, Cs + Ks[:, :, 1]/2)
    Ks[:, :, 3] = step * f(t + step,   Cs + Ks[:, :, 2]  )
    Cs = Cs + (Ks[:, :, 0] + 2*Ks[:, :, 1] + 2*Ks[:, :, 2] + Ks[:, :, 3])/6
    t += step
    res[i+1] = Cs.copy()

  return np.array(res)
```

ПРИЛОЖЕНИЕ 2.

Интерфейс для запуска численных моделей

```
def tracker(function_x, function_y, function_z, time_0, time_step,
            time_steps_count, x_0, y_0, z_0, vx_0, vy_0, vz_0,
            q, m):

  def f_x(t, Cs):
    E, B, alpha, angle_in_degrees = igrg_13(Cs[0, 0], Cs[1, 0], Cs[2, 0])
    return function_x(t, Cs[0, 0], Cs[1, 0], Cs[2, 0],
                      Cs[0, 1], Cs[1, 1], Cs[2, 1], x_0, y_0, z_0,
                      vx_0, vy_0, vz_0, q, m, E, B,
                      alpha, angle_in_degrees)
  def f_y(t, Cs):
    E, B, alpha, angle_in_degrees = igrg_13(Cs[0, 0], Cs[1, 0], Cs[2, 0])
    return function_y(t, Cs[0, 0], Cs[1, 0], Cs[2, 0],
                      Cs[0, 1], Cs[1, 1], Cs[2, 1], x_0, y_0, z_0,
                      vx_0, vy_0, vz_0, q, m, E, B,
                      alpha, angle_in_degrees)
  def f_z(t, Cs):
    E, B, alpha, angle_in_degrees = igrg_13(Cs[0, 0], Cs[1, 0], Cs[2, 0])
    return function_z(t, Cs[0, 0], Cs[1, 0], Cs[2, 0],
                      Cs[0, 1], Cs[1, 1], Cs[2, 1], x_0, y_0, z_0,
                      vx_0, vy_0, vz_0, q, m, E, B,
                      alpha, angle_in_degrees)

  res = RKM_system([f_x, f_y, f_z], time_0,
                   [[x_0, vx_0], [y_0, vy_0], [z_0, vz_0]],
                   time_step, time_steps_count)

  return Coord(res[:,0], res[:,1], res[:,2])
```