

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

КУРСОВАЯ РАБОТА
по дисциплине «Дифференциальные уравнения»
Тема: Моделирование работы кинескопа

Студент гр. 8382

Студент гр. 8382

Преподаватель

Нечепуренко Н.А.

Терехов А.Е.

Павлов Д.А.

Санкт-Петербург

2021

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студенты: Нечепуренко Н.А., Терехов А.Е.

Группа 8382

Тема работы: Моделирование работы кинескопа

Исходные данные:

Для моделирования работы кинескопа необходимо решить дифференциальное уравнение формулы Лоренца. При этом необходимо сравнить различные методы численного интегрирования и выбрать подходящий. Для корректного заполнения колбы кинескопа частицами, необходимо вывести алгоритм изменения модуля магнитных полей, при котором будет получен желаемый результат.

Содержание пояснительной записки:

«Содержание», «Введение», «Постановка задачи», «Численные методы решения», «Сравнение методов», «Графическое моделирование», «Заключение», «Список использованных источников».

Предполагаемый объём пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 20.04.2021

Дата сдачи работы: 07.06.2021

Студент гр. 8382

Нечепуренко Н.А.

Студент гр. 8382

Терехов А.Е.

Преподаватель

Павлов Д.А.

АННОТАЦИЯ

В работе рассматривается упрощенная модель кинескопа. Движение частиц в электромагнитном поле описывается с помощью формулы Лоренца. Применяв к ней численное интегрирование, можно получить траекторию движения заряженной частицы. В работе происходит визуализация процесса заполнения колбы кинескопа электронами.

СОДЕРЖАНИЕ

Введение	5
Постановка задачи	6
Модель кинескопа	6
Уравнение движения частиц в электромагнитном поле	6
Допущения и проверяемые инварианты системы	7
Численные методы решения	8
Функция правой части	8
Базовый класс метода решения	10
Прямой метод Эйлера	11
Метод Рунге-Кутты порядка 4	11
Метод Адамса-Башфорта порядка 3	12
Сравнение методов	14
Закон сохранения энергии движущейся частицы	14
Аналитическое решение для частного случая	20
Графическое моделирование	30
Начальные данные	30
Принцип изменения модулей магнитных полей	32
Заключение	33
Список использованных источников	34

ВВЕДЕНИЕ

В работе рассматривается упрощенная модель кинескопа. Электронная пушка придает частице начальную скорость, а затем с помощью изменения значений модулей магнитных полей частица попадает в нужное место на колбе. Движение частицы в электромагнитном поле описывается с помощью формулы Лоренца. Применив к ней численное интегрирование, можно получить траекторию движения заряженной частицы. В работе рассмотрены три метода численного интегрирования: метод Эйлера, метод Рунге-Кутты 4 порядка и метод Адамса-Башфорта 3 порядка. Была выполнена визуализация результатов.

1. ПОСТАНОВКА ЗАДАЧИ

1.1. Модель кинескопа

Кинескоп состоит из следующих основных частей:

- электронная пушка, предназначенная для формирования электронного луча; в цветных кинескопах и многолучевых осциллографических трубках несколько пушек объединяются в электронно-оптический прожектор
- экран, покрытый люминофором — веществом, светящимся при попадании на него пучка электронов
- отклоняющая система, управляющая лучом таким образом, что он формирует на экране требуемое изображение

Схематически строение кинескопа приведено на рисунке 1.

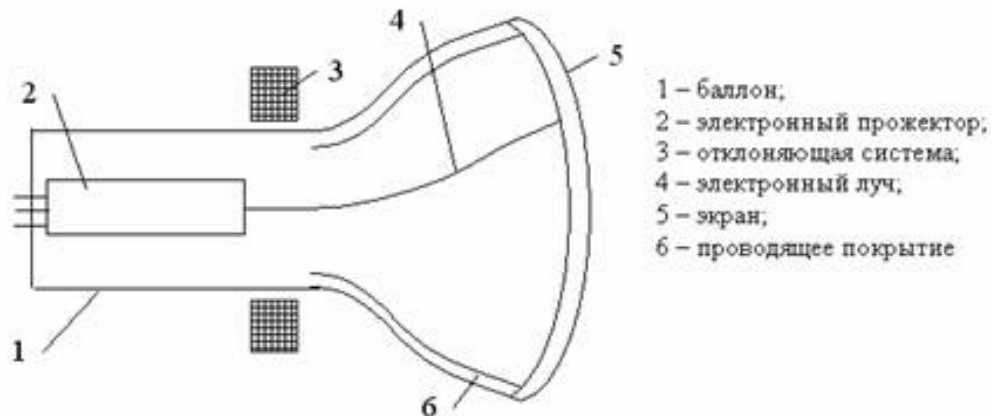


Рисунок 1 – Схема кинескопа

В работе будет рассмотрен кинескоп, способный выдавать черно-белое изображение.

1.2. Уравнение движения частиц в электромагнитном поле

Отклоняющая система кинескопа представлена тремя однородными полями: электрическим и двумя магнитными. Заряженная частица (электрон) движется в электромагнитном поле за счет действующей на нее силы Лоренца.

$$\vec{F} = q(\vec{E} + \vec{v} \times \vec{B})$$

С учетом 2-го закона Ньютона уравнение выше приводит к следующему дифференциальному уравнению:

$$\dot{\vec{v}} = \frac{q}{m}(\vec{E} + \vec{v} \times \vec{B})$$

Эта формула справедлива для однородных полей. Положительно заряженная частица будет ускоряться в том же направлении, что и поле E , но её траектория будет изгибаться перпендикулярно как вектору мгновенной скорости v , так и полю B в соответствии с правилом буравчика. Сила Лоренца — это сила, которую оказывает электромагнитное поле на заряженную частицу, или, другими словами, скорость, с которой передается линейный импульс от электромагнитного поля частице. Магнитное поле не совершает работы, потому что магнитная сила всегда перпендикулярна скорости частицы.

1.3. Допущения и проверяемые инварианты системы

В работе рассматриваются только однородные поля. Положим $q = 1$ и $m = 1$. Моделируемую частицу считаем дискретным носителем заряда. Частицы между собой не взаимодействуют, на них не действуют иные силы (например сила тяжести), отсутствует сопротивление среды (в колбе вакуум). Можно считать силу их взаимодействия пренебрежимо малой.

2. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ

2.1. Функция правой части

Для решения поставленной задачи была написана программа на языке C++ с использованием фреймворка Qt5.

Для представления функции правой части был разработан интерфейс `RhsFunction`.

```
class RhsFunction {  
public:  
    virtual State apply(State state) = 0;  
};
```

Все возможные функции правой части должны выглядеть как черный ящик, принимая состояние в текущий момент времени и возвращая состояние в следующий.

Состояние представляет собой структуру из координаты, скорости и заряда (не используется).

```
class State {  
public:  
    Point3D coordinate;  
    Point3D velocity;  
    double charge;  
    ...
```

Используемая в работе функция правой части от электрических и магнитных полей описана следующим образом:

```
class EMFieldMovingFunction : public RhsFunction {  
private:  
    std::vector<UniformField *> electricFields;  
    std::vector<UniformField *> magneticFields;  
    ...
```


Класс UniformField представляет собой структуру из направления векторов поля и модуля.

```
class UniformField {
public:
    double value;
    Point3D direction;
    ...
}
```

Реализация метода apply представлена ниже

```
State EMFieldMovingFunction::apply(State state) {
    if (terminatePredicate(state.coordinate))
        throw 1;

    Point3D dVelocity(0,0,0);

    for (auto &electricField : electricFields) {
        dVelocity = dVelocity + electricField->value *
            electricField->direction;
    }

    Point3D magneticAccumulator(0,0,0);
    for (auto &magneticField : magneticFields) {
        magneticAccumulator = magneticAccumulator + (
            magneticField->value * magneticField->direction);
    }
    double magneticValue = magneticAccumulator.absoluteValue
        ();
    magneticAccumulator = magneticAccumulator.normalize();

    dVelocity = dVelocity + magneticValue * state.velocity.
        cross(magneticAccumulator);
    return State(state.velocity, dVelocity, state.charge);
}
```

```
}
```

Здесь пользуемся принципом суперпозиции полей. Если частица попала на колбу, бросается исключение и вычисления прерываются.

2.2. Базовый класс метода решения

Для взаимозаменяемости методов решения уравнений был написан интерфейс Solver и базовый абстрактный класс AbstractSolver

```
class Solver {
public:
    virtual void solve(State initialState) = 0;
};

class AbstractSolver : public Solver {
protected:
    std::deque<State> previousStates;
    std::shared_ptr<RhsFunction> rhsFunction;
    std::function<void(State)> onUpdateConsumer;
    double h;
    long long maxIterations = 0;
    virtual State step(double time) = 0;
    ...
}
```

Контракт интерфейса Solver – любой метод должен генерировать решение из заданного начального состояния.

Базовый класс определяет способ взаимодействия с другими компонентами и набор параметров. Для хранения состояний используется deque, что удобно для многошаговых методов. Каждый конкретный метод решения обязан переопределить шаг метода. Базовый класс отвечает за итерирование по сетке решения и использование функции обратного вызова.

```
void AbstractSolver::solve(State initialState) {
```

```

previousStates.push_back(initialState);
for (long long iteration = 0; iteration < maxIterations;
    iteration++) {
    onUpdateConsumer(previousStates.back());
    State nextState = step(iteration);
    previousStates.push_back(nextState);
    previousStates.pop_front();
}
}

```

2.3. Прямой метод Эйлера

Форма пересчета

$$x_{i+1} = x_i + hf(x_i)$$

Прямой метод Эйлера реализован следующим образом

```

State EulerSolver::step(double time) {
    State currentState = previousStates.back();
    State nextState = currentState + h * rhsFunction->apply(
        currentState);
    return nextState;
}

```

Этот метод скорее необходим для сравнения с другими, нежели для непосредственного использования в программе.

2.4. Метод Рунге-Кутты порядка 4

Формула пересчета

$$x_{i+1} = x_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

где

$$k_1 = f(x_i)$$

$$k_2 = f(x_i + \frac{h}{2}k_1)$$

$$k_3 = f(x_i + \frac{h}{2}k_2)$$

$$k_4 = f(x_i + hk_3)$$

Реализация данного метода приведена ниже

```
State RK4Solver::step(double time) {
    State currentState = previousStates.back();
    State k1 = rhsFunction->apply(currentState);
    State k2 = rhsFunction->apply(currentState + h / 2. * k1)
        ;
    State k3 = rhsFunction->apply(currentState + h / 2. * k2)
        ;
    State k4 = rhsFunction->apply(currentState + h * k3);
    return currentState + h * (1/6. * k1 + 1/3. * k2 + 1/3. *
        k3 + 1/6. * k4);
}
```

2.5. Метод Адамса-Башфорта порядка 3

Формула пересчета

$$x_{i+1} = x_i + h(\frac{23}{12}f_i - \frac{4}{3}f_{i-1} + \frac{5}{12}f_{i-2})$$

где $f_n = f(x_n)$

Реализация метода чуть хитрее, так как он требует трех вычисленный состояний системы. Для их вычисления используется метод Ралстона 3 порядка с тем же шагом.

```
State AB3::step(double time) {
    if (previousStates.size() < 3) {
        prepareStates();
    }
}
```

```

    State currentState = previousStates.back();
    int idx = 0;
    for (auto &state : previousStates) {
        currentState = currentState + h * coefficients[idx] *
            rhsFunction->apply(state);
        idx++;
    }
    return currentState;
}

void AB3::prepareStates() {
    std::deque<State> oneStepStates;
    Solver *oneStepSolver = new Ralston3Solver(rhsFunction,
        [&oneStepStates](State state) -> void {
            oneStepStates.push_back(state);
        }, h, 3);
    State currentState = previousStates.back();
    oneStepSolver->solve(currentState);
    oneStepStates.pop_front();
    for (auto &s : oneStepStates) {
        onUpdateConsumer(s);
        previousStates.push_back(s);
    }
    delete oneStepSolver;
}

```

3. СРАВНЕНИЕ МЕТОДОВ

3.1. Закон сохранения энергии движущейся частицы

В электромагнитном поле действует закон сохранения энергии

$$E = K + QU = const$$

где $Q = 1$ (см. раздел 1.3), K – кинетическая энергия, U – потенциальная.

Скорость моделируемой частицы много меньше скорости света, поэтому положим

$$K = \frac{mv^2}{2} = \frac{v^2}{2}$$

так как $m = 1$ (см. раздел 1.3).

Примем потенциал электрического поля в точке $(0, 0, 0)$ за 0. Тогда:

$$U = \int_0^B \vec{E} \cdot d\vec{l}$$

Введем начальные условия для проверки инварианта. Частица находится в точке $(0, 0, 0)$, ее начальная скорость $(1, 0, 0)$. Действует магнитное поле в направлении $(0, 1, 0)$ с модулем 2 и электрическое поле в направлении $(1, 0, 0)$ с модулем 1.

Тогда выражение для U будет иметь вид:

$$U = -B_x \cdot 1$$

где B_x – x координата текущего положения частицы.

Проверка инварианта закона сохранения энергии заключается в проверке следующего тождества

$$E = \frac{v^2}{2} - B_x = const$$

Рассмотрим зависимость модуля разности вычисленной энергии на каждой итерации и аналитического значения 0.5.

Рассмотрим метод Эйлера.

Положим число итераций равным 10000.

Положим шаг равным 0.01. График изменения инварианта от итерации приведен ниже.

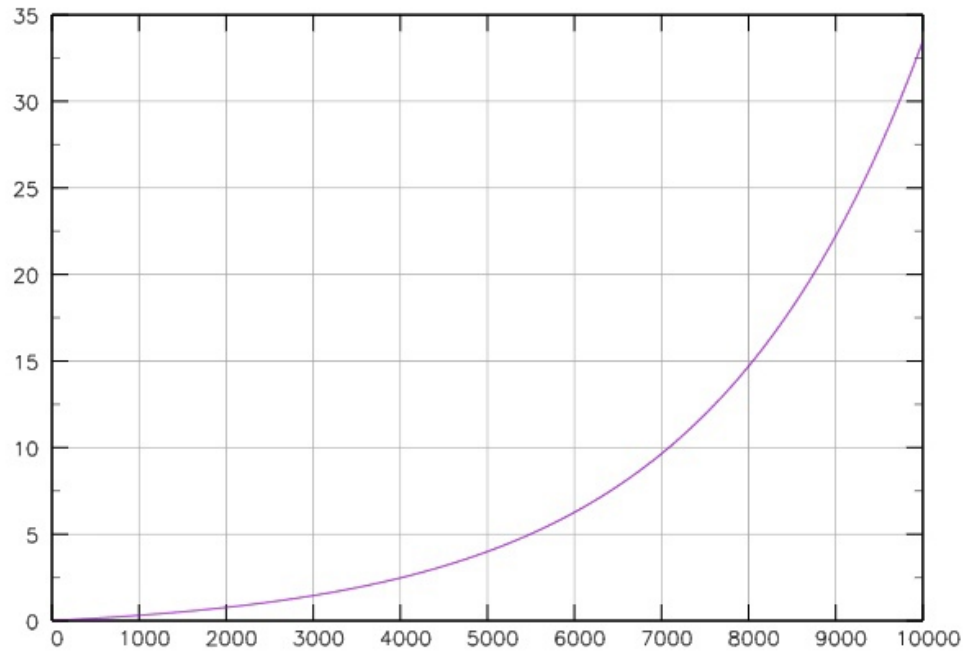


Рисунок 3.1 – Изменение инварианта, метод Эйлера с шагом 0.01

Положим шаг равным 0.001.

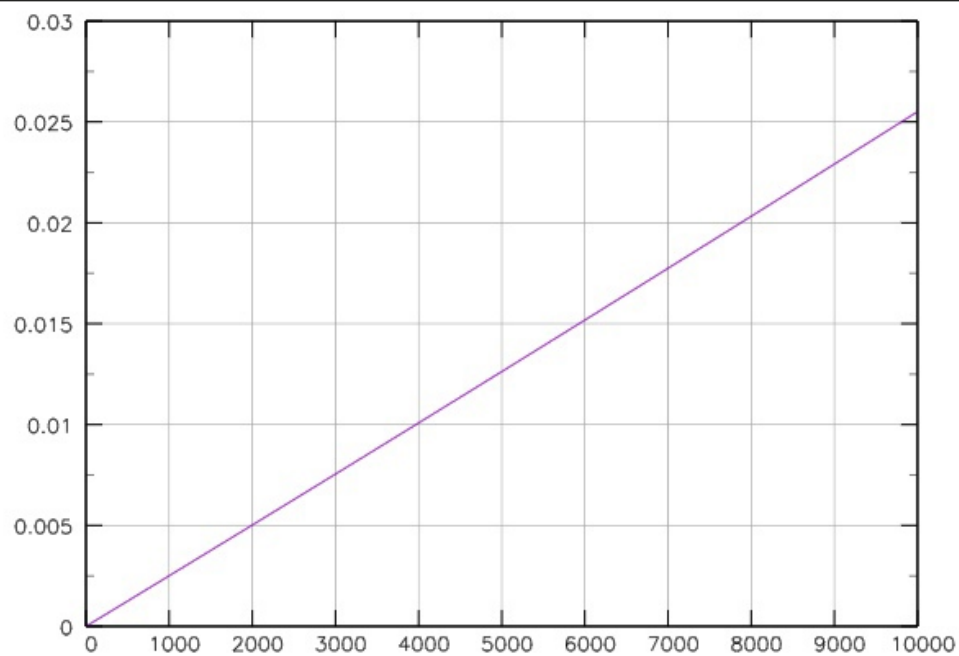


Рисунок 3.2 – Изменение инварианта, метод Эйлера с шагом 0.001

Положим шаг равным 0.0001.

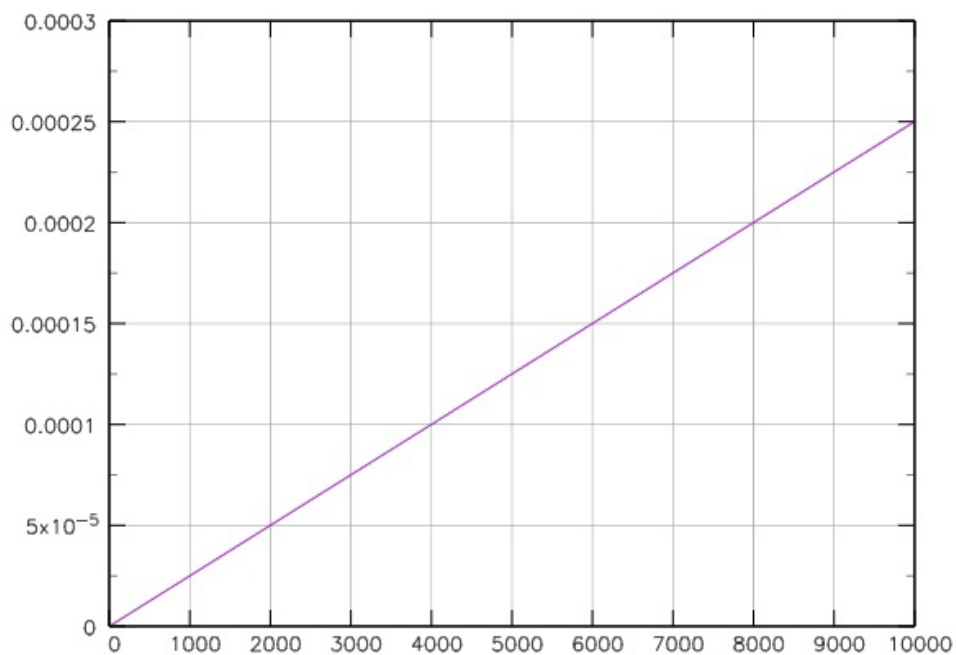


Рисунок 3.3 – Изменение инварианта, метод Эйлера с шагом 0.0001

Ошибка убывает слишком медленно с изменением порядка шага мето-

да.

Рассмотрим метод Адамса-Башфорта порядка 3.

Положим шаг равным 0.1.

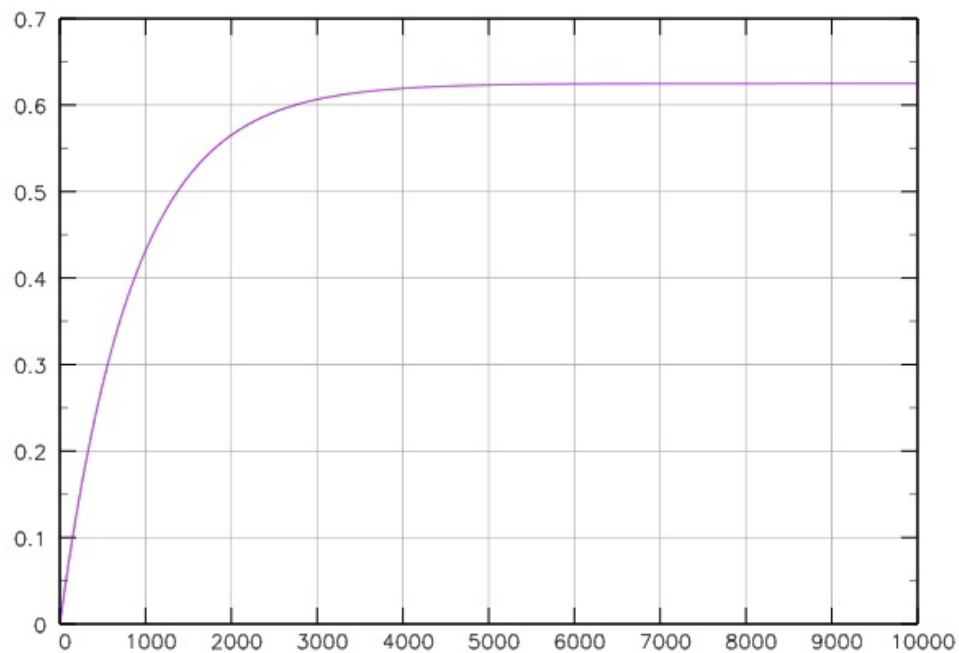


Рисунок 3.4 – Изменение инварианта, метод Адамса-Башфорта с шагом 0.1

Положим шаг равным 0.01.

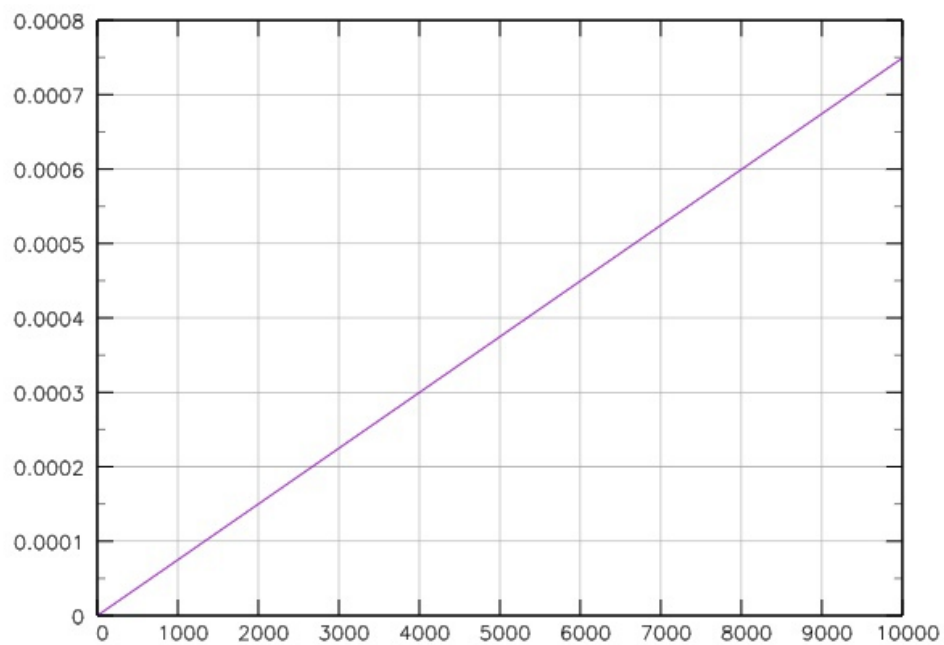


Рисунок 3.5 – Изменение инварианта, метод Адамса-Башфорта с шагом 0.01

Положим шаг равным 0.001.

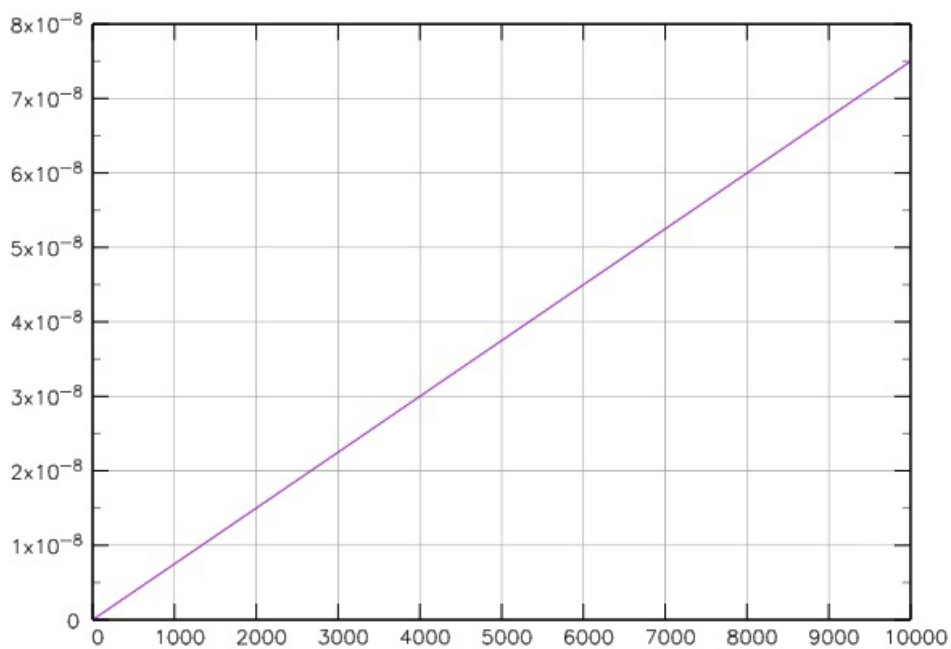


Рисунок 3.6 – Изменение инварианта, метод Адамса-Башфорта с шагом
0.001

За 10000 итераций разность между энергией частицы и аналитическим

значением не превысила $1e-7$.

Рассмотрим метод Рунге-Кутты порядка 4.

Положим шаг равным 0.1.

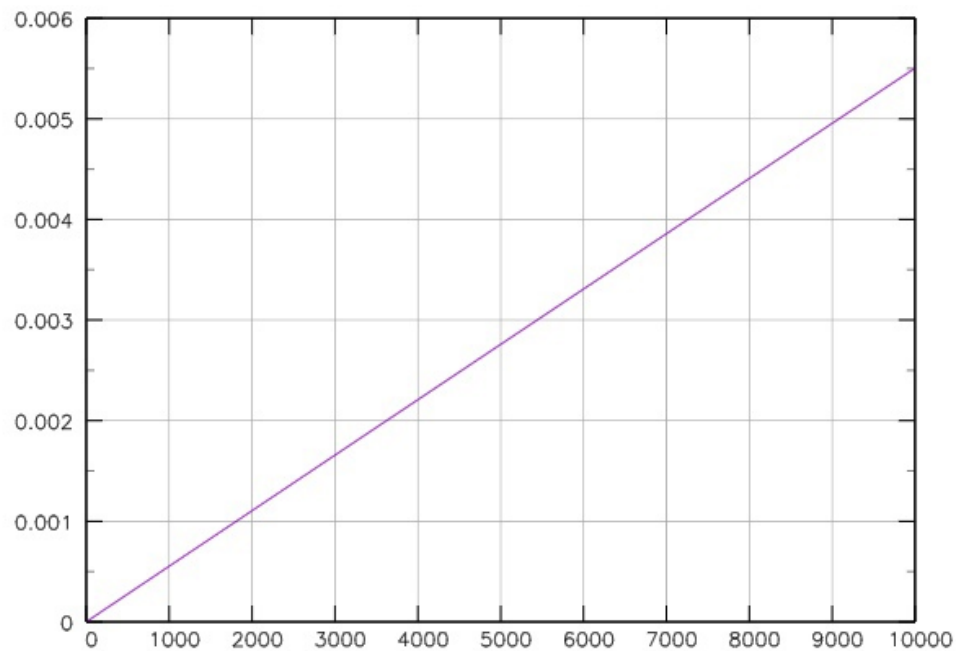


Рисунок 3.7 – Изменение инварианта, метод Рунге-Кутты с шагом 0.1

Положим шаг равным 0.01.

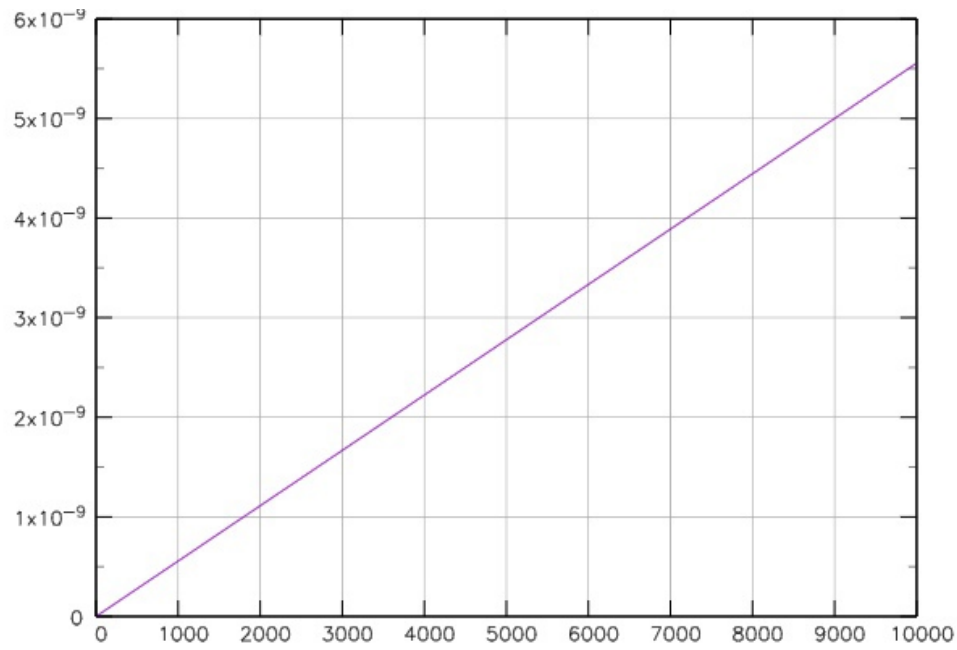


Рисунок 3.8 – Изменение инварианта, метод Рунге-Кутты с шагом 0.01

За 10000 итераций разность между энергией частицы и аналитическим значением не превысила $6e-9$.

3.2. Аналитическое решение для частного случая

Введем начальные условия. Частица находится в точке $(0, 0, 0)$ и имеет нулевую начальную скорость $(0, 0, 0)$. Действует электрическое поле в направлении $(1, 0, 0)$ с модулем 1 и магнитное поле в направлении $(0, 1, 0)$ с модулем 1. Тогда аналитическое уравнение для траектории частицы имеет вид:

$$\begin{cases} x(t) = -\sin(t) + t \\ y(t) = -\cos(t) + 1 \end{cases}$$

Сравним траектории частицы, полученные разными методами с разными шагами. На рисунках (3.9, 3.11, 3.13, 3.15, 3.19) значения $x(t)$ отложены на оси абсцисс, $y(t)$ на оси ординат. Глобальная ошибка – расстояние между полученной точкой траектории и точкой аналитического решения.

Рассмотри метод Эйлера.

Положим шаг равным 0.1.

Траектория движения частицы отображена на рисунке ниже фиолетовым цветом. Зеленая траектория – аналитическое решение.

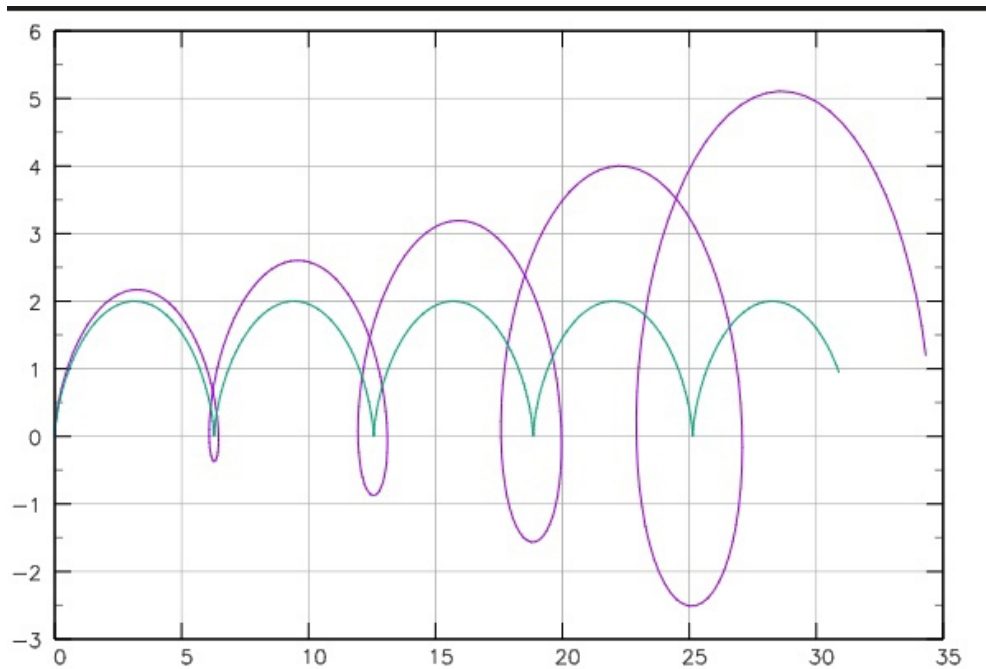


Рисунок 3.9 – Траектория движения с методом Эйлера, шаг 0.1

График глобальной ошибки от времени приведен на рисунке 3.10

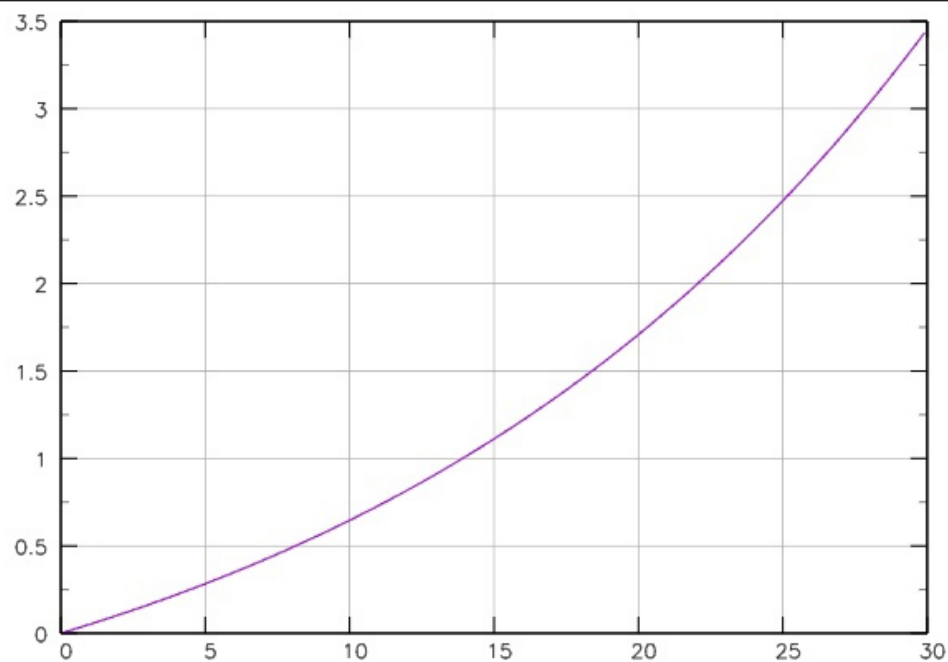


Рисунок 3.10 – Глобальная ошибка метода Эйлера, шаг 0.1

Положим шаг равным 0.01. Траектории приведены на рисунке 3.11, глобальная ошибка на рисунке 3.12.

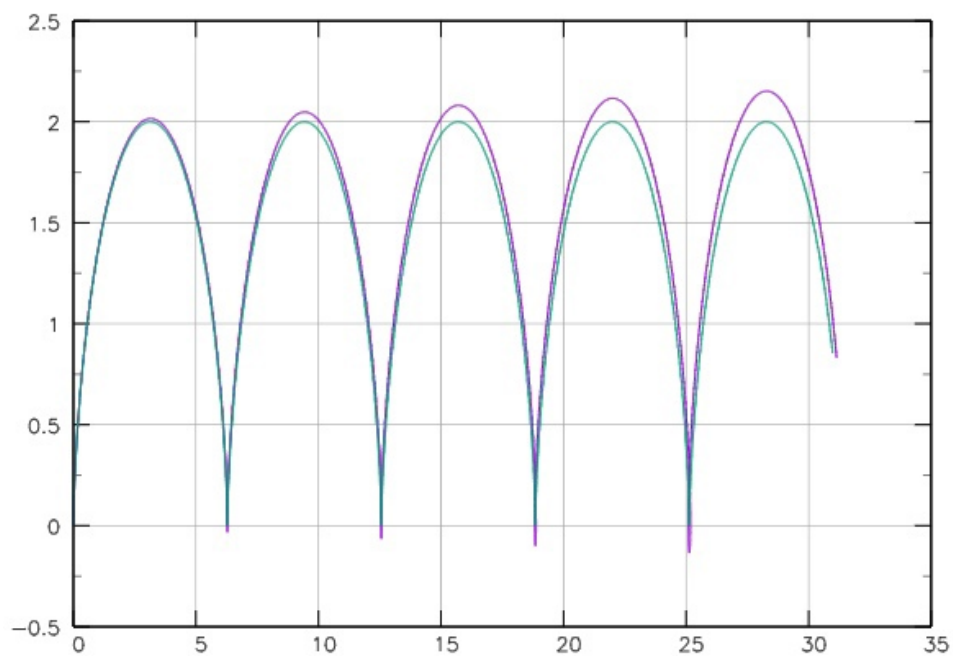


Рисунок 3.11 – Траектория движения с методом Эйлера, шаг 0.01

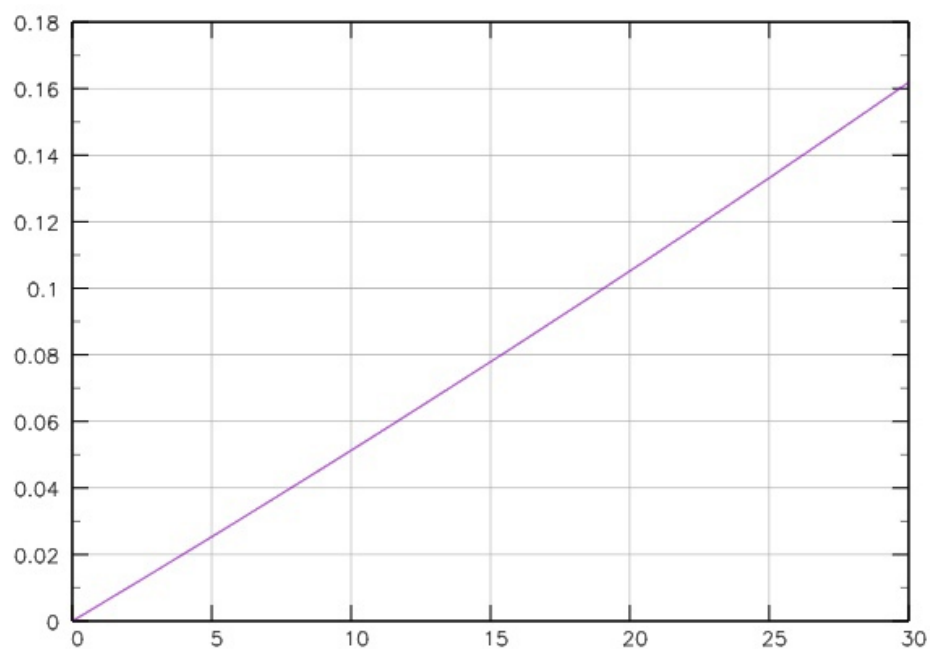


Рисунок 3.12 – Глобальная ошибка метода Эйлера, шаг 0.01

Положим шаг равным 0.001. Траектории приведены на рисунке 3.13, график глобальной ошибки на рисунке 3.14.

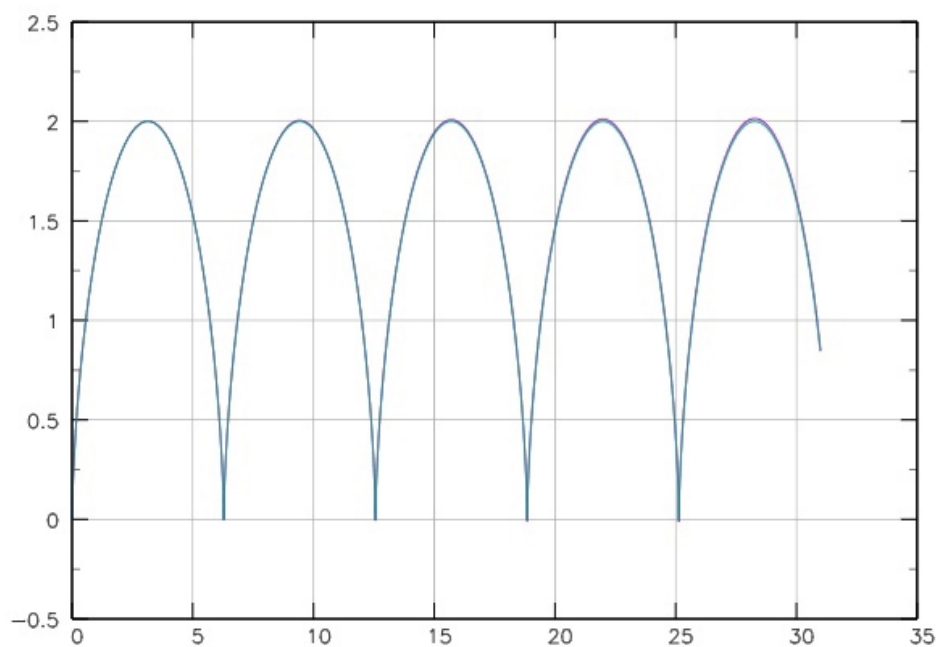


Рисунок 3.13 – Траектория движения с методом Эйлера, шаг 0.001

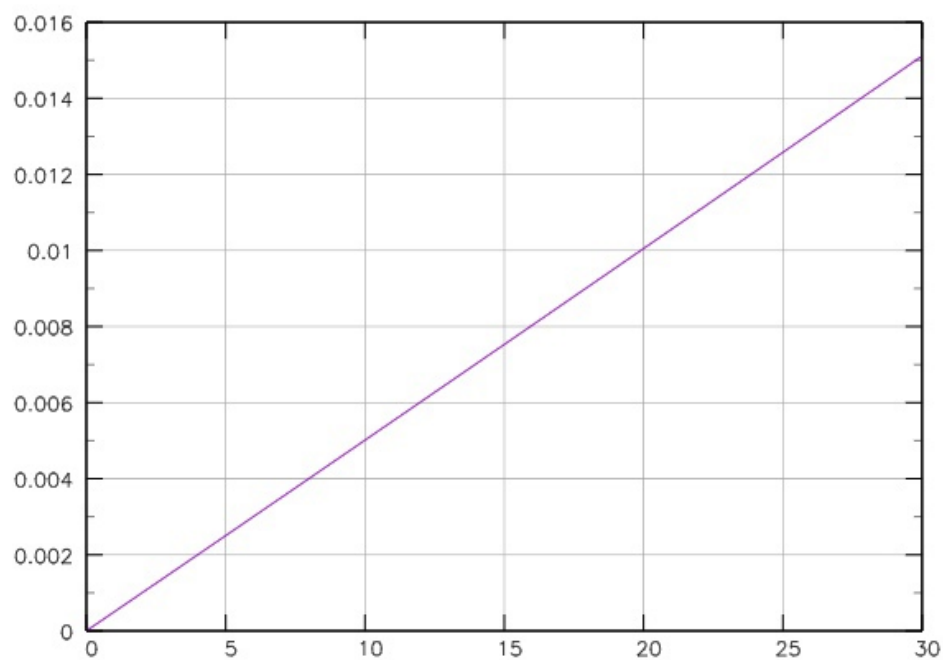


Рисунок 3.14 – Глобальная ошибка метода Эйлера, шаг 0.001

Ошибка с изменением порядка шага убывает слишком медленно.

Рассмотрим метод Адамса-Башфорта порядка 3.

Положим шаг равным 0.1. Траектория частицы приведена на рисунке 3.15.

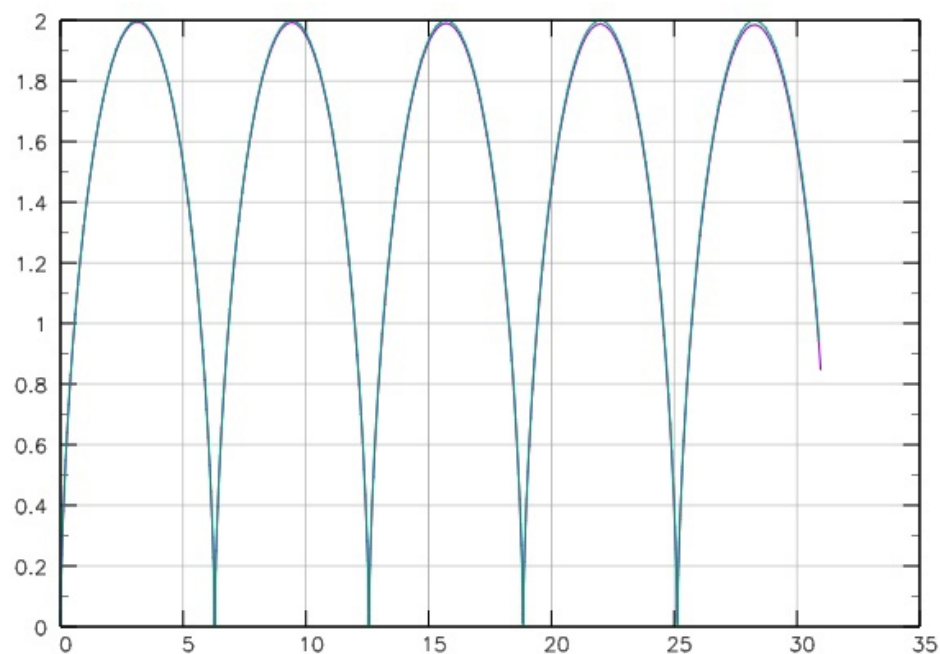


Рисунок 3.15 – Траектория движения с методом Адамса, шаг 0.1

Визуально траектории совпали, в дальнейшем будут приводиться только графики глобальной ошибки. Графики глобальной ошибки от времени при разных значениях шага отображены на рисунках 3.16, 3.17, 3.18.

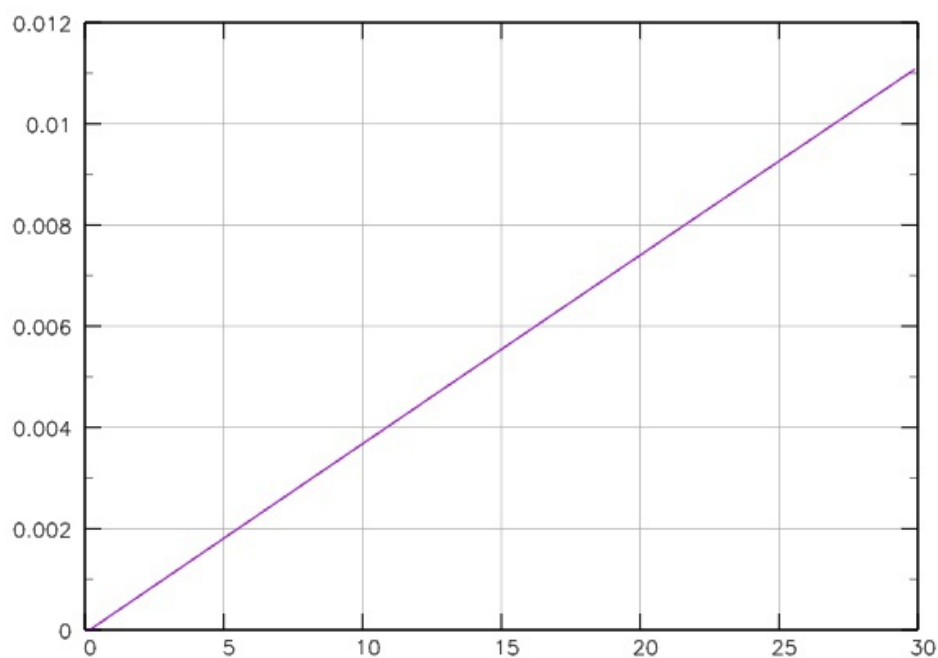


Рисунок 3.16 – Глобальная ошибка метода Адамса, шаг 0.1

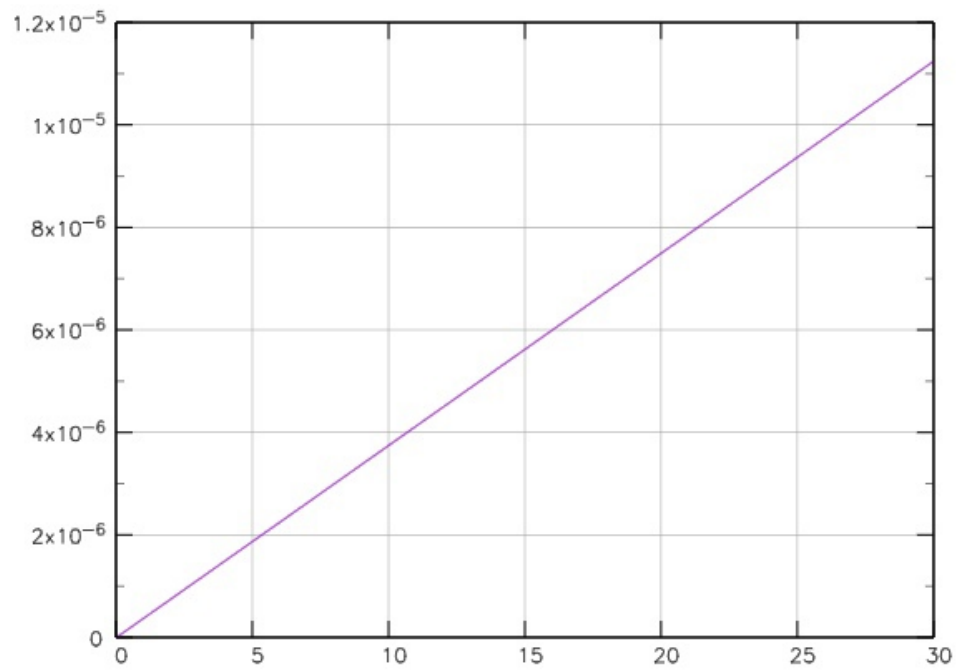


Рисунок 3.17 – Глобальная ошибка метода Адамса, шаг 0.01

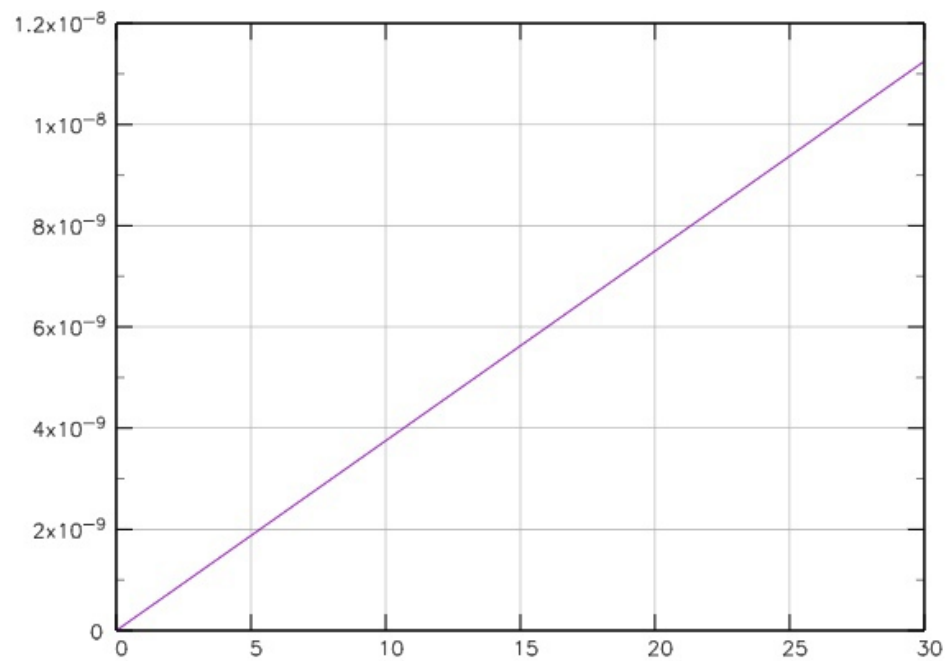


Рисунок 3.18 – Глобальная ошибка метода Адамса, шаг 0.001

Глобальная ошибка при использовании метода Адамса с шагом 0.001 (расстояние между i -той точкой траектории и i -той точкой аналитического решения) не превысило $12e-9$.

Рассмотрим метод Рунге-Кутты порядка 4.

Положим шаг равным 0.1 Траектория движения частицы и график глобальной ошибки приведены на рисунках 3.19 и 3.20 соответственно.

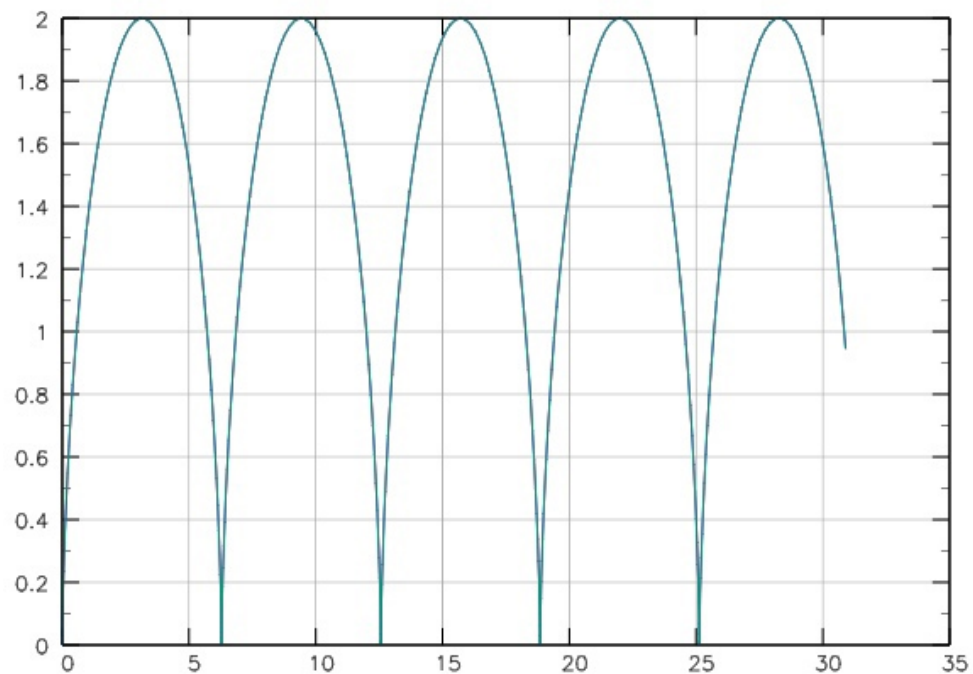


Рисунок 3.19 – Траектория движения с методом Рунге-Кутты, шаг 0.1

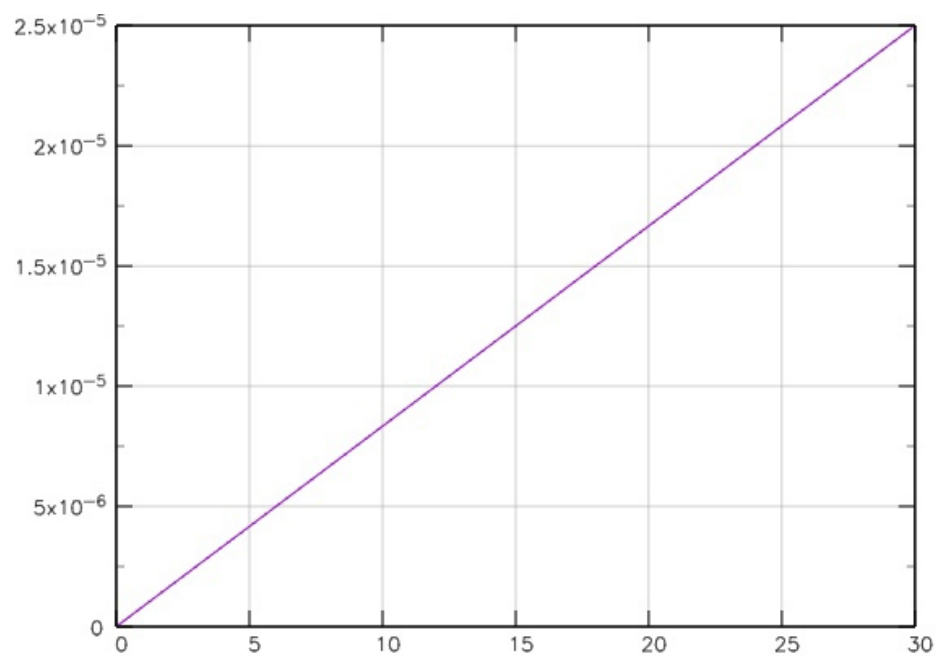


Рисунок 3.20 – Глобальная ошибка метода Рунге-Кутты, шаг 0.1

Рассмотрим графики глобальной ошибки при шаге 0.01 и 0.001.

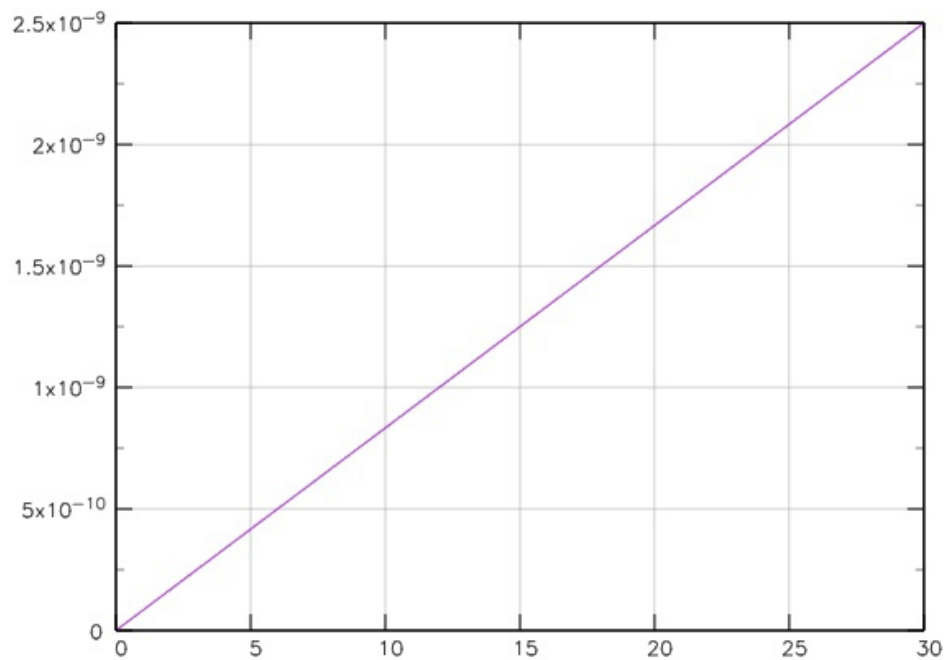


Рисунок 3.21 – Глобальная ошибка метода Рунге-Кутты, шаг 0.01

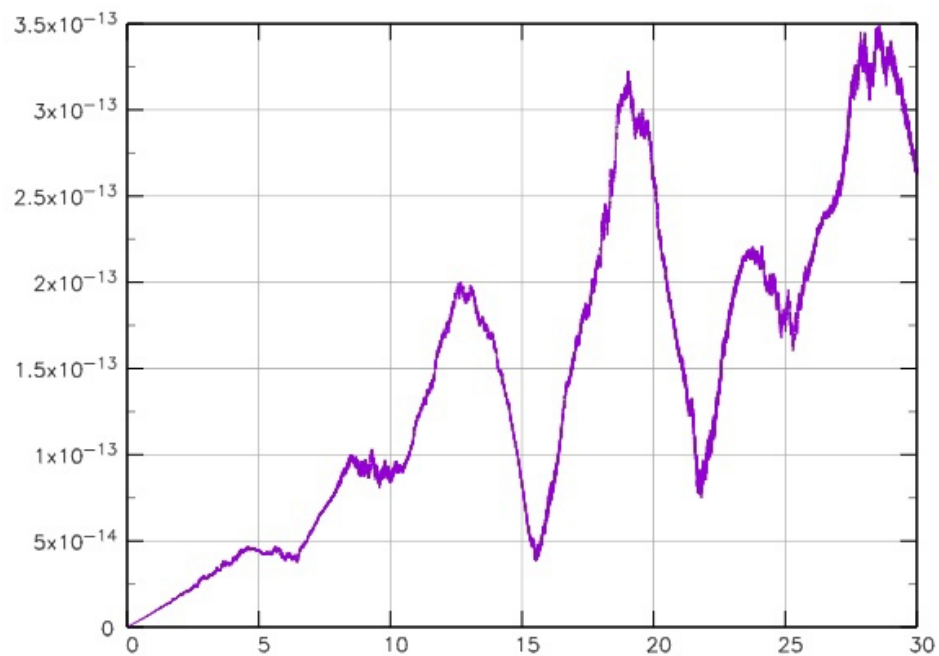


Рисунок 3.22 – Глобальная ошибка метода Рунге-Кутты, шаг 0.001

Изменение инварианта при использовании метода с шагом 0.01 не превышает $3e-9$, что удовлетворяет условиям моделирования.

При моделировании будет использован метод Рунге-Кутты порядка 4 с шагом 0.01.

4. ГРАФИЧЕСКОЕ МОДЕЛИРОВАНИЕ

4.1. Начальные данные

Интерфейс программы выглядит следующим образом

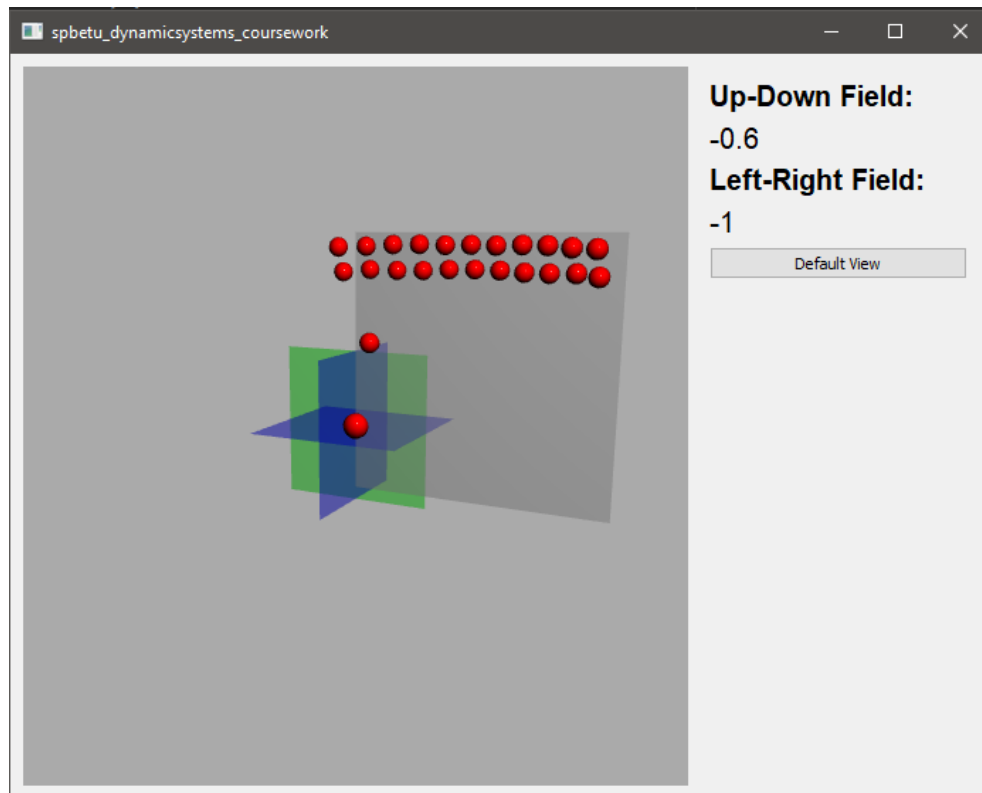


Рисунок 4.1 – Интерфейс программы

Сцена состоит из следующих элементов:

- Три пересеченных плоскости, ортогональные векторам соответствующих полей (синий – магнитные, зеленый – электрическое)
- Серая плоскость, касательная к сферической части колбы (не удалось изобразить колбу)
- Столбец со значениями модулей соответствующих магнитных полей
- Кнопка «default view»

Частицы начинают двигаться из точки $(0, 0, 0)$. Электронная пушка не показана. Начальная скорость частицы равна нулю, ее радиус 1.7 у.е.

Электрическое поле не изменяется, его линии сонаправлены с вектором $(1, 0, 0)$, модуль равен 10.

Направления линий магнитных полей не изменяются, они сонаправлены с векторами $(0, 0, 1)$ и $(0, 1, 0)$ соответственно. Их модули изменяются, и эта комбинация своя для каждой частицы. Изначальные значения равны -1. Алгоритм изменения модулей полей будет рассмотрен ниже.

При нажатии на кнопку «default view» камера перемещается в точку $(-20, 0, 0)$, что дает следующий вид на колбу

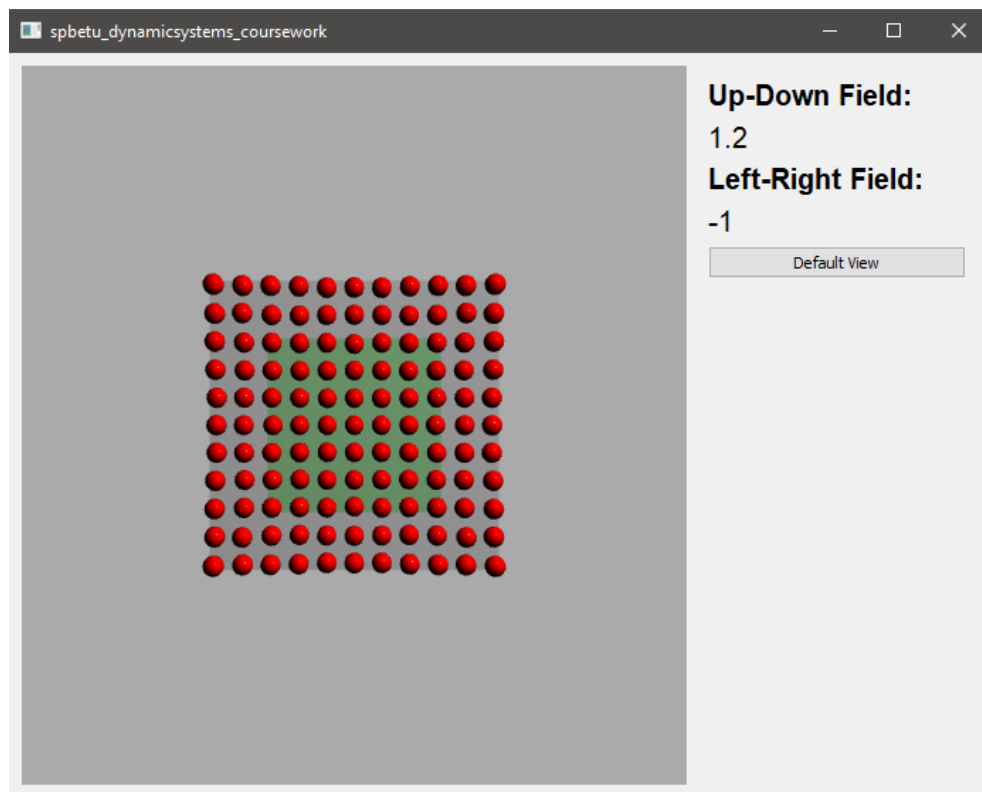


Рисунок 4.2 – Default view

Электроны на колбе, как и ожидалось, формируют часть сферы

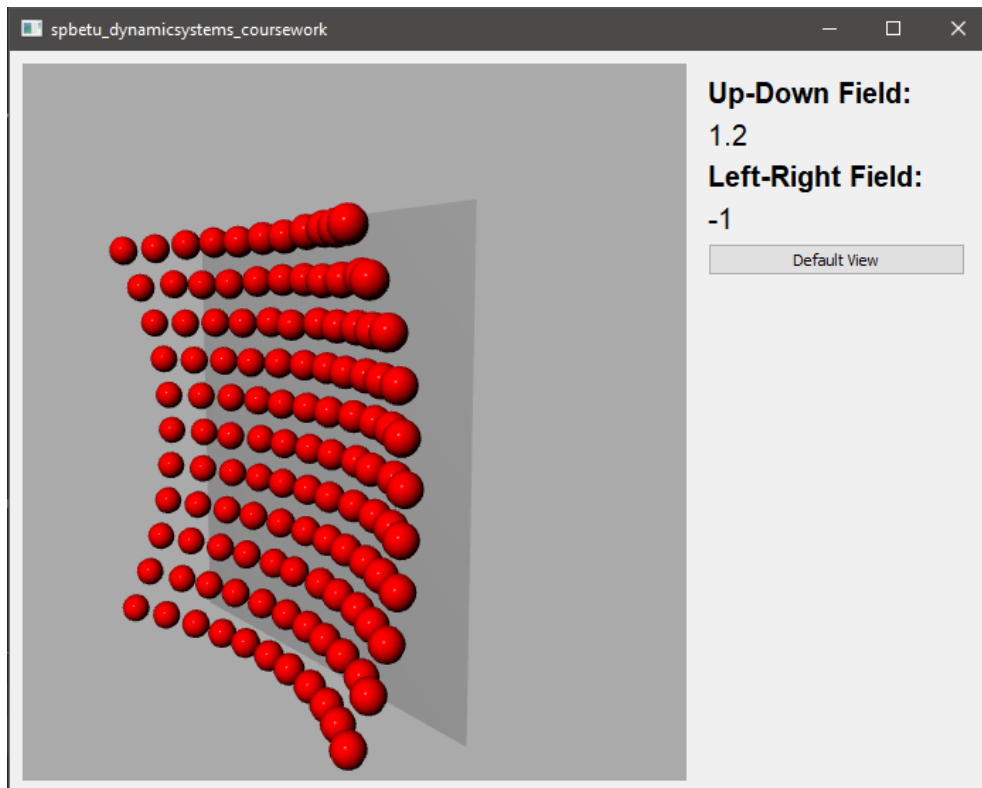


Рисунок 4.3 – Вид на колбу сбоку

4.2. Принцип изменения модулей магнитных полей

За изменение комбинации модулей магнитных полей для направления каждой частицы в отведенное место описан класс `MagneticGunManager` с методом `next`.

```
void next() {  
    secondField->value += d1;  
    iteration++;  
    if (iteration >= 11){  
        secondField->value = secondFieldValue;  
        firstField->value += d2;  
        iteration = 0;  
    }  
}
```

где $d_1 = 0.2, d_2 = 0.2$.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы была реализована упрощенная модель кинескопа. С учетом допущений (см. раздел 1.3.) для каждого электрона была рассчитана траектория движения согласно формуле Лоренца. Для расчета траектории было решено дифференциальное уравнение (см. раздел 1.2.). Решение производилось с помощью численных методов: прямого метода Эйлера, метода Рунге-Кутты порядка 4 и метода Адамса-Башфорта порядка 3. В разделе 3 было проведено сравнение методов. В разделе 4 были приведены рисунки с интерфейсом реализованной программы для моделирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сила Лоренца: https://ru.wikipedia.org/wiki/Сила_Лоренца
2. Электричество и магнетизм:
<https://online.mephi.ru/courses/physics/electricity/data/course/5/5.3.html>
3. Как работает телевизор: <https://cxem.net/beginner/beginner49.php>
4. Метод Адамса: https://ru.wikipedia.org/wiki/Метод_Адамса