**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

# ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине: «Методы математического моделирования сложных процессов и систем»

| | |
|---|---|
| Студент | Косенков Александр Александрович |
| Группа | РК6-12М |
| Тип задания | Лабораторная работа №3 |
| Тема лабораторной работы | Применение библиотек динамической компоновки для разработки программных реализаций вычислительных методов |

Студент _____ **Косенков А.А.**
*подпись, дата* *фамилия, и.о.*

Преподаватель _____ **Соколов А.П.**
*подпись, дата* *фамилия, и.о.*

Оценка _____

*Москва, 2021 г.*

# Оглавление

# Задание на лабораторную работу

## Вариант 2 (задача о падении тела в вязкой среде)

Тело массы $m$ падает вертикально вниз с некоторой высоты. Сила вязкого трения, действующая на тело, пропорциональна величине скорости: $F_{\text{тр}} = -\alpha v$, где $\alpha > 0$ – коэффициент трения. Определить зависимость скорости тела от времени $v(t)$.

**Требуемые для реализации численные методы**: метод Эйлера, метод Рунге-Кутты 4-ого порядка.

**Дополнительное задание**: определить, при каком значении коэффициента трения тело будет двигаться без ускорения.

Требуется:

1. Составить математическую модель процесса в форме ОДУ.
2. Поставить задачу и аналитически решить составленное ОДУ с заданными начальными условиями.
3. Численно решить поставленную задачу определёнными в варианте задания методами и сравнить полученные решения с аналитическим решением.
4. Для решения поставленной задачи разработать программные реализации требуемых методов на языке C++ на основе подключаемого метода решения ОДУ без перекомпиляции исходных кодов. Программная реализация должна обеспечивать возможность подключения других кроме требуемых методов решения ОДУ в будущем.
5. Обеспечить возможность определять неточно один или несколько скалярных исходных параметров.

   Например, для некоторого параметра $P$ программа должна позволять ввести: $M[P]$ (математическое ожидание) и $D[P]$ дисперсию параметра $P$; и/или диапазон изменения: $P \in \left[\underline{P}; \overline{P}\right]$, где $\underline{P}$, $\overline{P}$ – минимальное и максимальное значение $P$ соответственно.

   Предложить и описать в отчете интервальный алгоритм решения поставленной задачи при одном варьируемом параметре. Описать каким

образом возможно доработать предложенный алгоритм для случая нескольких скалярных параметров, заданных в виде их интервальных оценок.

6. Результаты решения и сравнения должны сохраняться в виде текстового файла с разделителями (*.csv* формат).
7. Полученные численно и аналитически зависимости представить графически на одной координатной плоскости и представить в отчете.
8. Вычислить максимально допустимые шаги интегрирования по времени для каждого из применяемых для решения численных методов, обеспечивающие минимальное отклонение от аналитического решения (использовать норму $L_1$ или $L_2$).

## Цель выполнения лабораторной работы

**Цель выполнения лабораторной работы** – изучить принципы создания библиотек динамической компоновки на примере решения ОДУ различными численными методами, реализуемыми независимо в виде функций, экспортируемых из таких библиотек.

## Выполненные задачи

1. Составление математической модели процесса в форме ОДУ.
2. Постановка задачи, аналитическое решение составленного ОДУ с заданными начальными условиями.
3. Настройка среды разработки. Структура проекта. Разработка библиотеки динамической компоновки.
4. Численное решение поставленной задачи, сравнение с аналитическим решением.
5. Реализация интервального алгоритма решения поставленной задачи при одном варьируемом параметре.

6. Определение максимально допустимого шага интегрирования при заданной точности.
7. Исследование влияния коэффициента трения на результирующие значения скорости движения тела.

В ходе лабораторной работы для программной реализации задач был использован язык C++ (с использованием стандарта C++17), а также система автоматизации сборки cmake. Запуск производился на ОС Windows 10.

## 1. Составление математической модели процесса в форме ОДУ.

Обозначим скорость тела $v(t)$ в момент времени t.

На тело действуют две противоположно направленные силы: сила тяжести $F_т = mg$ и сила вязкого трения $F_{тр} = -\alpha v$.

Согласно второму закону Ньютона: $ma = F = F_т + F_{тр}$, таким образом получим:

$$m\frac{dv(t)}{dt} = mg - av. \quad (1)$$

Формула (1) является математической моделью падения тела в вязкой среде в форме ОДУ [1].

## 2. Постановка задачи, аналитическое решение составленного ОДУ с заданными начальными условиями.

В качестве задачи ставится определение зависимости $v(t)$ в случае, если в начальный момент времени тело покоится, что является решением ОДУ (1) с задачей Коши $v(0) = 0$.

Общее решение ОДУ (1) можно найти с помощью метода разделения переменных. Преобразуем (1):

$$\frac{dv}{mg - av} = \frac{dt}{m},$$

После интегрирования обоих частей получаем:

$$\int \frac{dv}{mg - av} = \int \frac{dt}{m},$$

$$-\frac{1}{\alpha}\ln|mg - \alpha v| = \frac{t}{m} + C,$$

$$e^{C - \frac{\alpha}{m}t} = mg - \alpha v,$$

$$v(t) = \frac{mg}{\alpha} - Ce^{-\frac{\alpha}{m}t}. \quad (2)$$

Решая задачу Коши $v(0) = 0$, получаем частное решение:

$$v(t) = \frac{mg}{\alpha}\left(1 - e^{-\frac{\alpha}{m}t}\right). \quad (3)$$

Формула (3) является аналитическим решением ОДУ (1).

## 3. Настройка среды разработки. Структура проекта. Разработка библиотеки динамической компоновки

В рамках лабораторной работы была разработана программная реализация решения поставленной задачи с помощью численных методов Эйлера и Рунге-Кутты 4-го порядка, а также ее аналитическое решение. В виду требования по динамическому подключению методов, была применена утилита автоматизации сборки *cmake*. Предварительно была сформирована следующая структура каталогов:

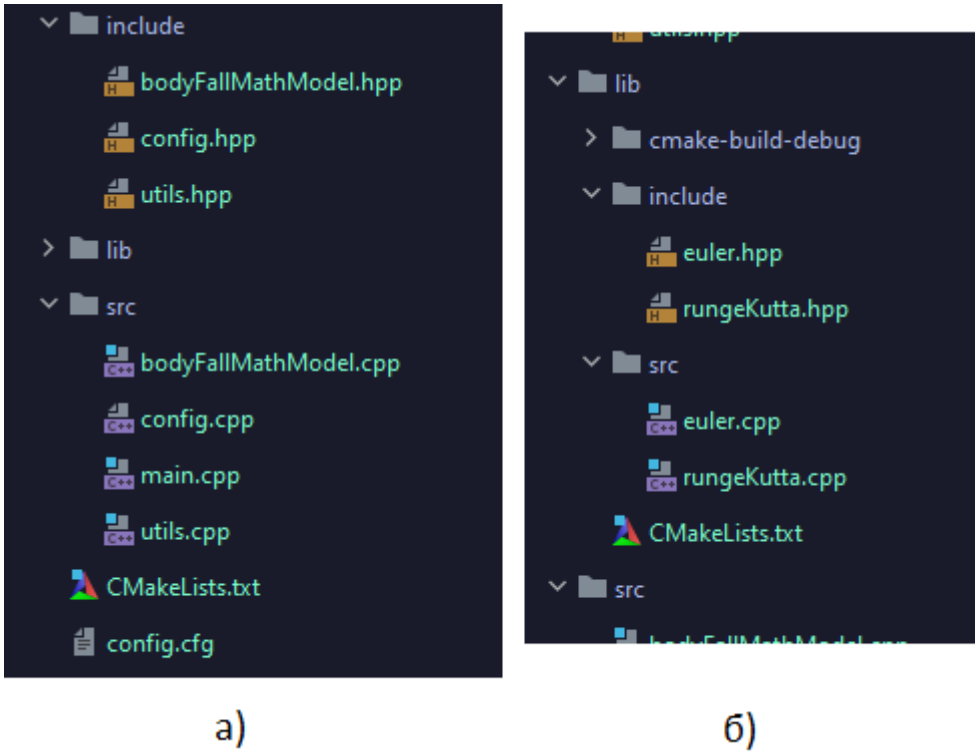


Рис. 1. Структура каталогов проекта (а) и библиотеки с реализацией численных методов (б).

Общая структура предполагает наличие папок *src* и *include*, содержащих файлы с исходным кодом формата *.cpp* и сопоставленные им заголовочные файлы формата *.hpp* соответственно, а также файл сборки *CMakeLists.txt*. Подключаемая динамическая библиотека была вынесена в каталог *lib* и представляет собой вложенный проект со своей структурой и своим файлом сборки.

Содержание подключаемой библиотеки:

- *euler.hpp* – сигнатура численного метода Эйлера для решения поставленной задачи
- *rungeKutta.hpp* – сигнатура численного метода Рунге-Кутты первого порядка для решения поставленной задачи

Содержание основного проекта:

- *bodyFallMathModel.hpp* – сущности, инкапсулирующие в себе параметры задачи и предоставляющие функционал для вычисления точки аналитического решения и шага дифференциального уравнения (1).
- *config.hpp* – интерфейс объекта конфигурации, использующего для работы текстовый файл *config.cfg*, находящийся в корне проекта.
- *utils.hpp* – сигнатуры прикладных методов, необходимых в ходе выполнения лабораторной работы, среди которых присутствует вычисление погрешности измерений, максимального шага интегрирования, построение графиков и т.д.

Листинг 1 содержит инструкции cmake для сборки библиотеки динамической компоновки, что достигается с помощью использования инструкции *add_library* с атрибутом *SHARED*. Кроме того, для более удобной работы с библиотечными методами, и проектом и библиотекой используется общий интерфейс, объявленный в заголовочном файле *bodyFallMathModel.hpp*, что повлекло за собой для корректной работы в ОС Windows 10 добавление инструкций *target_sources*, указывающих на определение данного интерфейса.

Листинг 1. Содержимое файла сборки *CMakeLists.txt* для подключаемой библиотеки.

```
cmake_minimum_required(VERSION 3.17)
project(libs)

set(CMAKE_CXX_STANDARD 17)
set(CMAKE_WINDOWS_EXPORT_ALL_SYMBOLS ON)

include_directories(${CMAKE_SOURCE_DIR}/include ${CMAKE_SOURCE_DIR}/lib/include)

add_library(euler SHARED ${CMAKE_SOURCE_DIR}/lib/src/euler.cpp)
add_library(rungeKutta SHARED ${CMAKE_SOURCE_DIR}/lib/src/rungeKutta.cpp)

target_sources(euler PRIVATE ${CMAKE_SOURCE_DIR}/src/bodyFallMathModel.cpp
${CMAKE_SOURCE_DIR}/src/config.cpp)
target_sources(rungeKutta PRIVATE ${CMAKE_SOURCE_DIR}/src/bodyFallMathModel.cpp
${CMAKE_SOURCE_DIR}/src/config.cpp)
```

Листинг 2 содержит инструкции для сборки основного проекта. При этом инструкции, описанные в Листинге 1 являются внешними и подключаются к общим через инструкцию *add_subdirectory*. Разрешение путей до локальных заголовочных файлов осуществляется через инструкции *include_directories* и *target_include_directories*, подключение динамических библиотек, сформированных ранее, осуществляется с помощью инструкции *target_link_libraries*, а проксирование файла конфигурации *config.cfg* в директорию сборки проекта реализуется с помощью инструкции *configure_file*.

Листинг 2. Содержимое файла сборки *CMakeLists.txt* для проекта.

```
cmake_minimum_required(VERSION 3.17)
project(lab1)
set(CMAKE_CXX_STANDARD 17)
add_subdirectory(lib)

set(SRCS src/main.cpp src/config.cpp src/utils.cpp src/bodyFallMathModel.cpp)
set(LIBS euler rungeKutta)

include_directories(${CMAKE_SOURCE_DIR}/include ${CMAKE_SOURCE_DIR}/lib/include)

target_include_directories(euler PUBLIC ${CMAKE_SOURCE_DIR}/lib/include)
target_include_directories(rungeKutta PUBLIC ${CMAKE_SOURCE_DIR}/lib/include)

add executable(lab1 ${SRCS})
target_link_libraries(lab1 ${LIBS} ${CMAKE_DL_LIBS})

configure_file(config.cfg config.cfg COPYONLY)
```

Для обеспечения требования отсутствия перекомпиляции исходных кодов – был разработан класс *ConfigurationSingleton*, представляющий собой синглтон Майерса [2]. Данный класс позволяет получать значения переменных из конфигурационного файла, изменение которого не влечет за собой повторную компиляцию проекта.

Параметры математической модели были инкапсулированы в классе *BodyFallParams*, который, в свою очередь, является агрегатом класса *BodyFallMathModel*.

Реализация метода вычисления скорости на следующем шаге относительно предыдущего с помощью дифференциального уравнения (1), который используется в численных методах, и метода получения точки аналитического решения, который используется для получения аналитической зависимости $V(t)$ по формуле (3), представлена в Листинге 3:

Листинг 3. Реализация методов класса *BodyFallMathModel*.

```
double BodyFallMathModel::getAnalyticalSolutionPerTime(double time) {
    return this->params.m * BodyFallParams::G / this->params.alpha *
          (1 - exp(-this->params.alpha * time / this->params.m));
}

double BodyFallMathModel::getDiffSolutionPerTime(double vPrev) {
    return BodyFallParams::G - this->params.alpha * vPrev / this->params.m;
}
```

Дальнейшие вычисления графика зависимости $V(t)$ подразумевает использование функций-решателей с унифицированной сигнатурой, определенной в заголовочном файле *utils.hpp*. Сигнатура представлена в Листинге 4.

Листинг 4. Сигнатура функции-решателя ОДУ (1).

```
using SolveFunction = void (*) (BodyFallMathModel, std::map<double, double>&);
```

Данной сигнатурой обладают разрабатываемые методы библиотеки динамической компоновки, а также функция получения аналитического решения ОДУ *solveAnalytic*, реализующая итерационное вычисление точек аналитического решения (3) в цикле.

С помощью разработанной функции *solveAnalytic* была получена аналитическая зависимость $V(t)$ поставленной задачи (рис. 2).

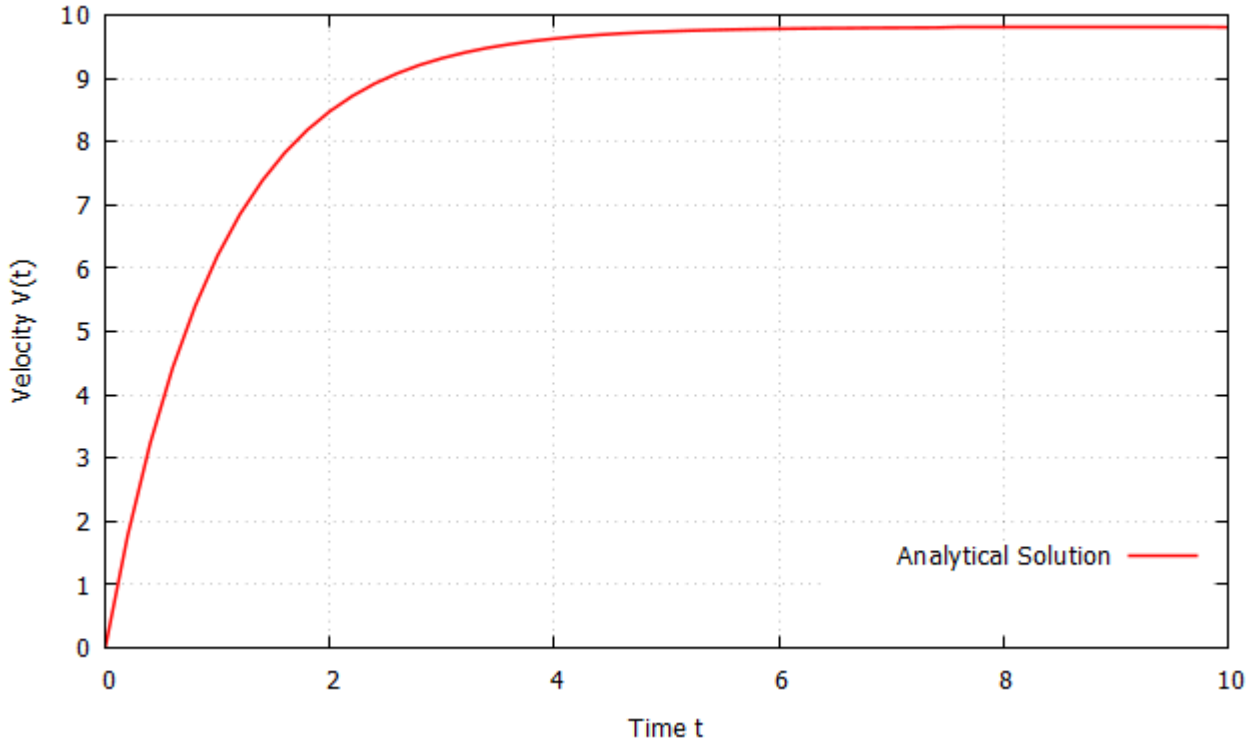


Рис. 2. Аналитическое решение ОДУ ($m = 10$ кг., $\alpha = 10$)

## 4. Численное решение поставленной задачи, сравнение с аналитическим решением.

В рамках разработки библиотеки динамической компоновки с целью удовлетворения требования отсутствия перекомпиляции исходных кодов были разработаны функции численного решения ОДУ (1) с помощью методов Эйлера (Листинг 5) и Рунге-Кутты 4-го порядка (Листинг 6).

Метод Эйлера представляет собой явный, одношаговый численный метод первого порядка точности для решения систем ОДУ, основанный на аппроксимации интегральной кривой кусочно-линейной функцией.

Для ОДУ общего вида:

$$\frac{dy}{dt} = f(t, y),$$

где $t \in [a; b]$, а $y(a) = \alpha$ – задача Коши. Тогда формулировка метода Эйлера имеет вид [3]:

$$\begin{gathered} \omega_0 = \alpha, \\ \omega_{i+1} = \omega_i + hf(t_i, \omega_i), \quad i = 0, 1, \dots, m-1\,, \end{gathered} \tag{4}$$

где предполагается, что $\omega_i \approx y(t_i)$.

Для рассматриваемой модели (1) формулировка (4) принимает частный вид при $y(t_i) = v(t_i)$:

$$\begin{gathered} v_0 = 0, \\ v_{i+1} = v_i + h\left(g - \frac{\alpha v_i}{m}\right). \end{gathered} \tag{5}$$

Реализация (5) представлена в функции *eulerSolution* (Листинг 5).

Листинг 5. Реализация метода Эйлера.

```
void eulerSolution(BodyFallMathModel &solution, std::map<double, double>
&resultEuler) {
    BodyFallParams params = solution.getParams();
    double timestamp = params.timeStart;
    double vPrev = 0.0;
    double vCurrent = 0.0;

    resultEuler.insert(std::pair<double, double>(timestamp, vCurrent));

    for (int i = 1; i <= (int) (params.timeEnd / params.step); i++) {
        vCurrent = vPrev + params.step * solution.getDiffSolutionPerTime(vPrev);
        vPrev = vCurrent;
        timestamp += params.step;
        resultEuler.insert(std::pair<double, double>(timestamp, vCurrent));
    }
}
```

Метод Рунге-Кутты 4-го порядка основан на аппроксимации ряда Тэйлора [3] и имеет следующую формулировку:

$$\omega_{i+1} = \omega_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \qquad i = 0, 1, \dots, m-1, \tag{6}$$

где:

$$\begin{gathered} k_1 = hf(t_i, \omega_i), \\ k_2 = hf\left(t_i + \frac{h}{2}, \omega_i + \frac{1}{2}k_1\right), \\ k_3 = hf\left(t_i + \frac{h}{2}, \omega_i + \frac{1}{2}k_2\right), \\ k_4 = hf(t_i + h, \omega_i + k_3). \end{gathered} \tag{7}$$

Для рассматриваемой модели (1) формулировка (6) - (7) принимает частный вид при $\omega_i \approx y(t_i) = v(t_i)$:

$$v_{i+1} = v_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \qquad i = 0,1,\dots,m-1, \tag{8}$$

где:

$$\begin{aligned} k_1 &= h\left(g - \frac{\alpha}{m}v_i\right), \\ k_2 &= h\left(g - \frac{\alpha}{m}\left(v_i + \frac{1}{2}k_1\right)\right), \\ k_3 &= h\left(g - \frac{\alpha}{m}\left(v_i + \frac{1}{2}k_2\right)\right), \\ k_4 &= h\left(g - \frac{\alpha}{m}(v_i + k_3)\right). \end{aligned} \tag{9}$$

Реализация (8) – (9) представлена в функции *rungeKuttaSolution* (Листинг 6).

Листинг 6. Реализация метода Рунге-Кутты 4-го порядка.

```
void rungeKuttaSolution(BodyFallMathModel &solution, std::map<double, double>&
resultRungeKutta) {
    BodyFallParams params = solution.getParams();

    double timestamp = params.timeStart;
    double vCurrent = 0.0;
    double vPrev = 0.0;

    resultRungeKutta.insert(std::pair<double, double>(timestamp, vCurrent));

    for (int i = 1; i <= (int) (params.timeEnd / params.step); i++) {
        double k1 = params.step * (BodyFallParams::G - params.alpha * (vPrev) /
params.m);
        double k2 = params.step * (BodyFallParams::G - params.alpha * (vPrev + 1
/ 2 * k1) / params.m);
        double k3 = params.step * (BodyFallParams::G - params.alpha * (vPrev + 1
/ 2 * k2) / params.m);
        double k4 = params.step * (BodyFallParams::G - params.alpha * (vPrev +
k3) / params.m);

        vCurrent = vPrev + (k1 + 2. * k2 + 2. * k3 + k4) / 6;
        vPrev = vCurrent;
        timestamp += params.step;
        resultRungeKutta.insert(std::pair<double, double>(timestamp, vCurrent));
    }
}
```

Результаты вычислений с помощью разработанных функций представлены на графике сравнения аналитического и численных решений ОДУ (1) на рис.3.

Анализ результатов на рис.3 дает основание полагать о корректности работы разработанных численных методов.

Также была реализована возможность записи результатов в текстовый .csv файл для каждого случая решения. Содержимое данных файлов, являющееся набором пар $t - v(t)$ представлено на рис.4

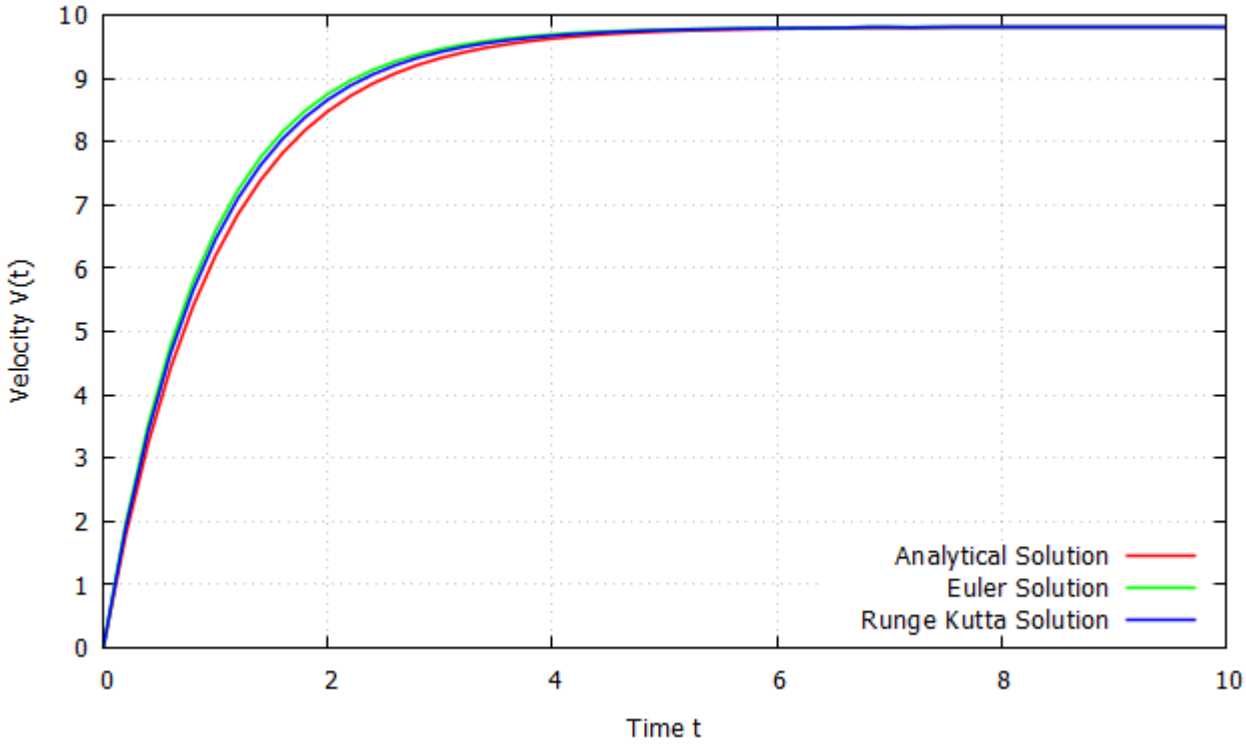


Рис. 3. Графики $v(t)$ для аналитического и численных решений ($m = 10$ кг., $\alpha = 10$, шаг = 0.2с.)

analytical_solution.csv

| t | V(t) |
|---|---|
| 0 | 0 |
| 0.2 | 1.77764 |
| 0.4 | 3.23306 |
| 0.6 | 4.42465 |
| 0.8 | 5.40024 |
| 1 | 6.19899 |
| 1.2 | 6.85294 |
| 1.4 | 7.38836 |
| 1.6 | 7.82672 |
| 1.8 | 8.18562 |
| 2 | 8.47946 |
| 2.2 | 8.72004 |
| 2.4 | 8.91701 |
| 2.6 | 9.07828 |
| 2.8 | 9.21031 |

euler_solution.csv

| t | V(t) |
|---|---|
| 0 | 0 |
| 0.2 | 1.96133 |
| 0.4 | 3.53039 |
| 0.6 | 4.78565 |
| 0.8 | 5.78985 |
| 1 | 6.59321 |
| 1.2 | 7.2359 |
| 1.4 | 7.75005 |
| 1.6 | 8.16137 |
| 1.8 | 8.49042 |
| 2 | 8.75367 |
| 2.2 | 8.96427 |
| 2.4 | 9.13274 |
| 2.6 | 9.26752 |
| 2.8 | 9.37535 |

rungeKutta_solution.csv

| t | V(t) |
|---|---|
| 0 | 0 |
| 0.2 | 1.89595 |
| 0.4 | 3.42535 |
| 0.6 | 4.65907 |
| 0.8 | 5.65427 |
| 1 | 6.45706 |
| 1.2 | 7.10465 |
| 1.4 | 7.62704 |
| 1.6 | 8.04843 |
| 1.8 | 8.38835 |
| 2 | 8.66256 |
| 2.2 | 8.88375 |
| 2.4 | 9.06218 |
| 2.6 | 9.20611 |
| 2.8 | 9.32221 |

Рис. 4. Содержимое выходных .csv файлов после выполнения вычислений ($m = 10$ кг., $\alpha = 10$, шаг = 0.2с.).

# 5. Реализация интервального алгоритма решения поставленной задачи при одном варьируемом параметре.

Вариация параметров математической модели возможна с применением интервального анализа [4], который применяется для получения приближенного решения (его оценки) при неявно заданных параметрах. Основной сущностью являются интервалы, которыми задаются границы значений параметров для оценки. В случае вариации нескольких параметров (т.е. при наличии нескольких интервалов) может быть применена интервальная арифметика.

В рамках данной лабораторной работы была предложена реализация вариации одного параметра $P$ с помощью интервала $P \in [\underline{P}; \overline{P}]$ или с помощью математического ожидания $M(P)$ и дисперсии $D(P)$.

Для этого предусмотрена возможность указания параметров *INTERVAL_PARAM* и *INTERVAL_MODE*, отвечающих за варьируемую переменную (*alpha* или *mass*) и за тип указания интервала (*interval* или *dispersion*) соответственно, а также сами интервальные параметры в формате *<переменная>_<LEFT/RIGHT>* для режима типа *interval* и *<переменная>_<MATH_EXPECTED/DISPERSION>* для режима типа *dispersion*. Пример приведен на рис. 5:

```
INTERVAL_PARAM = alpha
INTERVAL_MODE = dispersion

ALPHA_MATH_EXPECTED = 30.
ALPHA_DISPERSION = 400
```

*Рис. 5. Часть содержимого файла config.cfg, отвечающая за интервальный анализ*

При этом в случае режима interval интервал для параметра $P$ задается как $[L_i, R_i]$, где $L_i$ и $R_i$ принимают значения переменных из конфигурационного файла, а в случае режима dispersion интервал задается как $[L_D, R_D]$, где $L_D = M(P) - \sigma(P)$, $R_D = M(P) + \sigma(P)$. $\sigma(P)$ в свою очередь является квадратичным отклонением и вычисляется как квадратный корень из заданной в конфигурационном файле дисперсии [5]: $\sigma(P) = \sqrt{D(P)}$.

Вычисление данных интервалов в соответствии с параметрами, указанными в конфигурационном файле, осуществляется с помощью разработанной функции *getIntervalParams*.

В свою очередь для оценки всех возможных вариаций зависимости $v(t)$ при неявно заданном параметре сам интервальный анализ подразумевает получение решения только на границах рассматриваемого интервала, поскольку, как видно из аналитического решения (3) и его графика (рис. 2) – зависимость $v(t)$ является монотонной, а значит, для оценки поведения зависимости решения от вариации параметра решения на границах достаточно.

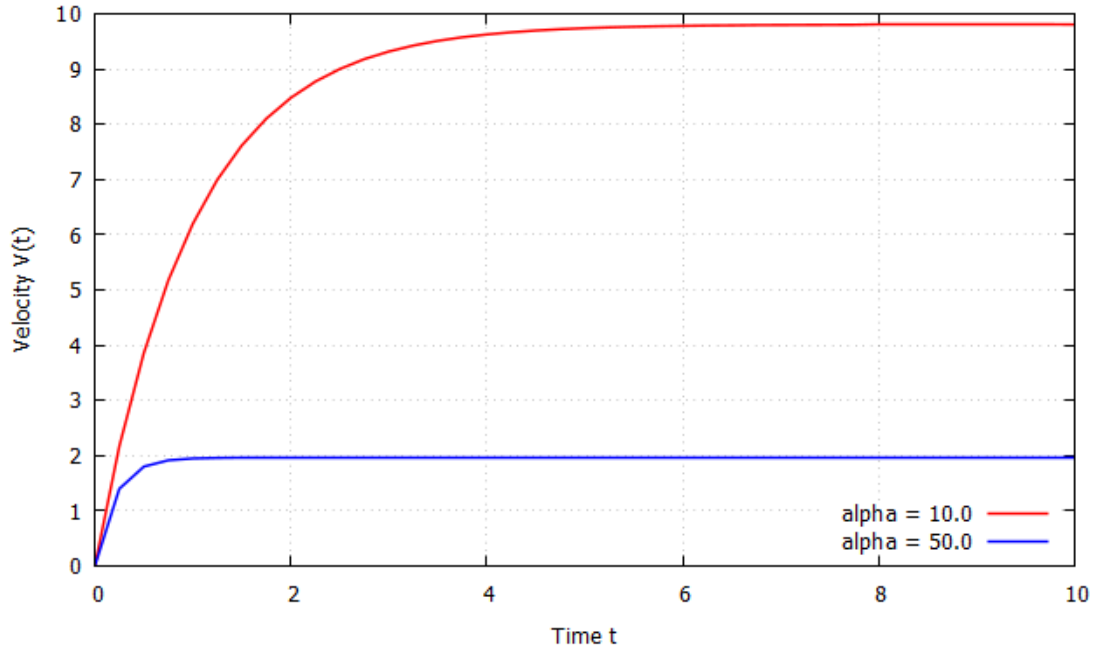


Рис. 6. Графики зависимостей $v(t)$ при неявно заданном параметре $\alpha$. Интервал задан значениями границ $[10, 50]$.

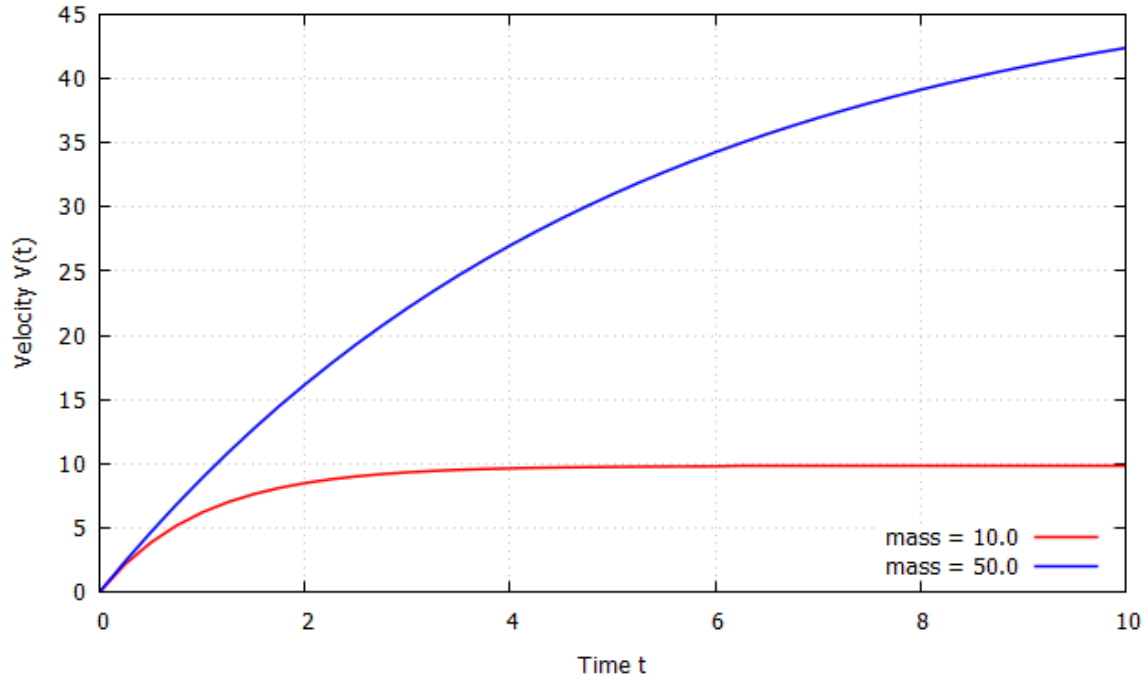


Рис. 7. Графики зависимостей $v(t)$ при неявно заданном параметре $m$. Интервал задан значениями $M(P) = 30$ *и* $D(P) = 400$.

Получение граничных решений реализовано в функции *solveInterval*. На рис. 6-7 приведены примеры результатов для разных режимов интервального анализа.

## 6. Определение максимально допустимого шага интегрирования при заданной точности.

Определение максимально допустимого шага интегрирования при заданной точности необходимо для нахождения оптимального соотношения между вычислительными затратами и точностью. В рамках данной работы точность задается в конфигурационном файле *config.cfg*, а вычисление максимально допустимого шага осуществляется с помощью функции *findMaxStep*.

Данная функция определяет удовлетворение шага точности решения путем вычисления относительной погрешности численного метода по отношению к аналитическому решению с помощью нормы $L_2$, определяемой в нашем случае как:

$$L_2 = \sqrt{\frac{1}{H}\sum_{i=0}^{H-1}\left(\frac{v_i - v_i^*}{v_i}\right)^2},$$

где $H$ – количество шагов, $v_i$– значение скорости на $i$-ом шаге, полученное при аналитическом решении, $v_i^*$ – значение скорости на $i$-ом шаге, полученное при решении с помощью численного метода. Вычисление данной нормы реализовано в функции *findNormL2*.

Алгоритм представляет собой уменьшение шага в цикле, пока значение погрешности не будет ниже значения точности, после чего происходит увеличение шага при удовлетворении точности с целью поиска максимально допустимого шага. Дробление реализовано с коэффициентом $0.5$, увеличение – с коэффициентом $1.2$ (коэффициенты выбраны эмпирически)

Для параметров $m = 10$ кг, $\alpha = 10$ были определены оптимальные шаги интегрирования:

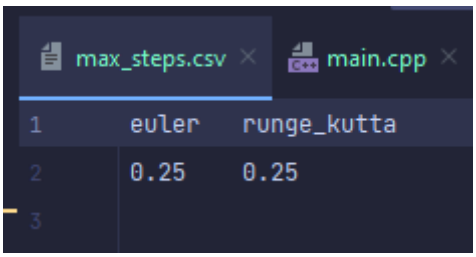


Рис. 8. csv-файл со значениями максимально допустимых шагов интегрирования

## 7. Исследование влияния коэффициента трения на результирующие значения скорости движения тела.

Дополнительным заданием данной лабораторной работы является определение того, при каком значении коэффициента трения тело будет двигаться без ускорения.

Строгий анализ аналитического решения (3) позволяет понять, что данная ситуация невозможна при любом значении коэффициента трения, который представляет собой константу при переменной $t$. Для доказательства рассмотрим изменяющуюся компоненту уравнения: $1 - e^{-\frac{\alpha}{m}t}$. Можно заметить, что с течением времени (при увеличении $t$) происходит уменьшение компонента $e^{-\frac{\alpha}{m}t}$, а значит увеличение $1 - e^{-\frac{\alpha}{m}t}$ в целом. Однако, данное изменение является ассимптотически монотонным, поскольку правая часть стремится к значению $0$, но не принимает его. Следовательно, изменение скорости, выраженное зависимостью $v(t)$ монотонно и постоянно, т.е. ускорение всегда присутствует и не принимает значение $0$. Однако, в предельном случае данная ситуация возможна, поскольку $\lim\limits_{t\to\infty} e^{-\frac{\alpha}{m}t} = 0$. Аналогично возможно рассмотрение предела для коэффициента трения: $\lim\limits_{\alpha\to\infty} e^{-\frac{\alpha}{m}t} = 0$. И, таким образом:

$$a(t) = \dot{v}(t) = \lim_{\alpha\to\infty} g e^{-\frac{\alpha}{m}t} = 0. \qquad (10)$$

Действительно, с определенной погрешностью возможно наблюдение подобной ситуации при большом значении коэффициента трения:

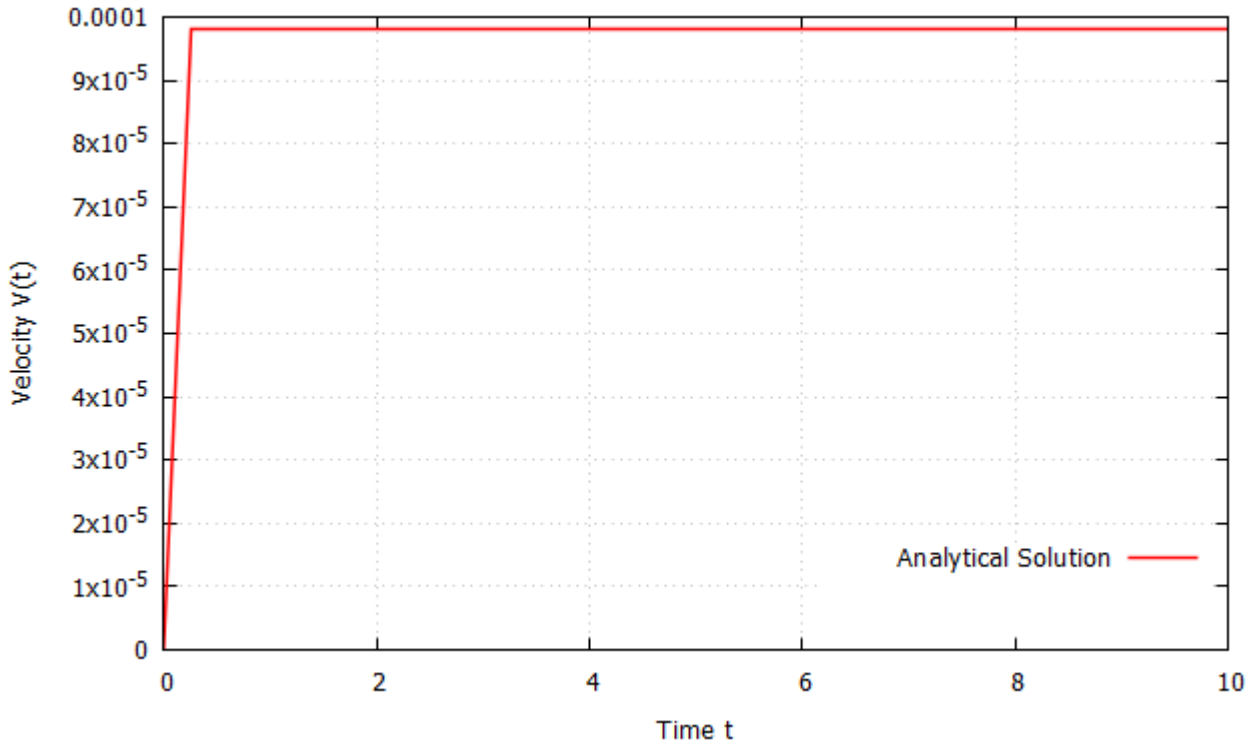


Рис. 9. Аналитическое решение ОДУ ($m = 1$ кг., $\alpha = 10^5$)

Подобное поведение можно сравнить с тем, как тело лежит на твердой поверхности. При этом бесконечно малый прирост скорости падения тела, наблюдаемый в аналитической модели, можно наблюдать и в реальности в виде диффузии твердых тел.

Однако, для обеспечения требования $a = 0$ присутствует другая проблема – по задаче Коши из условия тело начинает движение с нулевой скоростью, откуда при сколь угодно большом значении коэффициента $\alpha$ будет возникать скачок в начале графика зависимости $v(t)$, наблюдаемый на рис. 6.

Во избежание подобного скачка возможно изменение задачи Коши на:

$$v(0) = \frac{mg}{\alpha}. \tag{11}$$

В таком случае:

$$\frac{mg}{\alpha} = \frac{mg}{\alpha} - Ce^{-\frac{\alpha}{m}0},$$

$$0 = -C * 1$$

$$\mathrm{C} = 0,$$

откуда

$$v(t) = \frac{mg}{\alpha} - 0e^{-\frac{\alpha}{m}t} = \frac{mg}{\alpha} = const, \quad (12)$$

что удовлетворяет условию $a = 0$ при строгом аналитическом анализе.

Более того, решение ОДУ (1) с начальным условием (11) методом Эйлера дает следующий график:

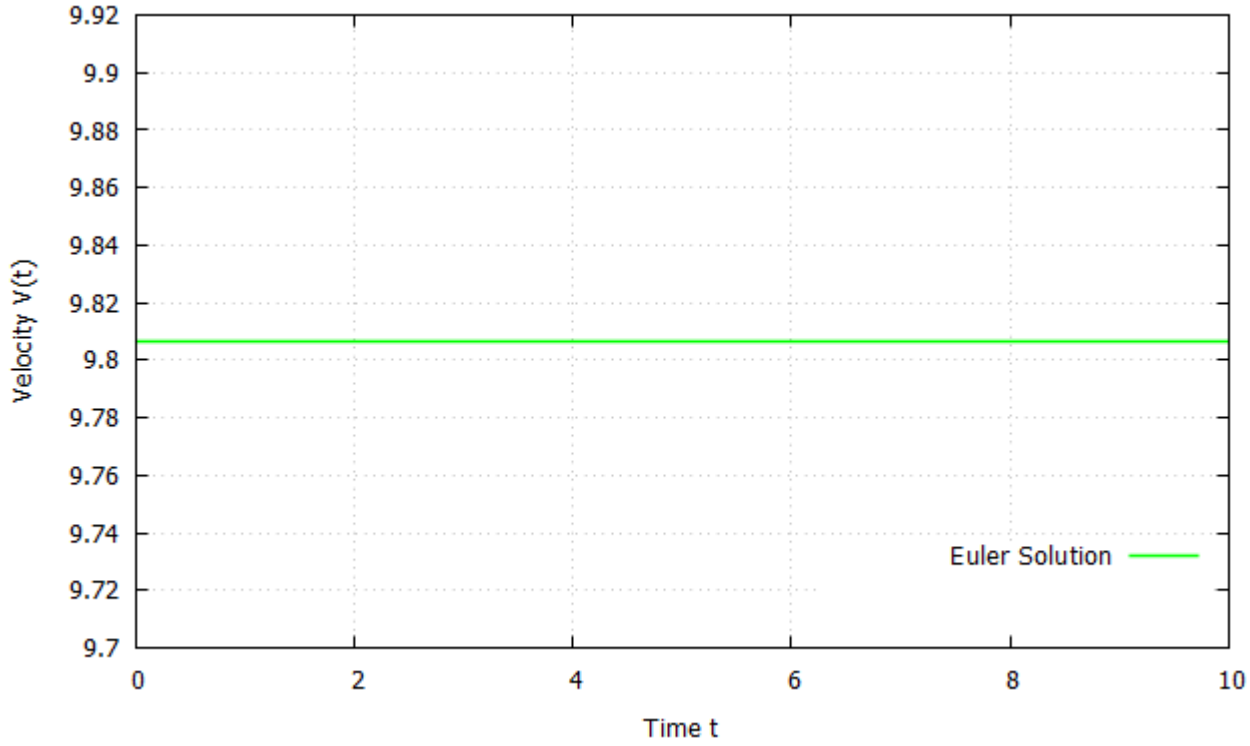


Рис. 10. График зависимости $v(t)$, найденный методом Эйлера с начальным условием (11) при ($m = 10$ кг., $\alpha = 10$)

Значение скорости на рис. 7 по графику: $v = 9.80665$, что соответствует $\frac{mg}{\alpha} = \frac{10*9.80665}{10} = 9.80665$, а поведение соответствует требованию задания.

При этом, если рассмотреть зависимость $v(t)$ при тех же параметрах, но с начальным условием, заданным в постановке задачи (рис. 2), и определить значение времени, при котором скорость начинает изменяться меньше заданной точности (например, $\varepsilon = 10^{-9}$), для чего был реализован метод *findStableVelocity*, то наблюдается схождение значения скорости к $9.80665$ при $t = 21.5$ с.

Таким образом, возможность движения тела без ускорения возможно при начальном условии (11) при любом значении коэффициента трения $\alpha$, или при бесконечно большом значении $\alpha$ в предельном рассмотрении в случае начального условия, при котором тело покоится.

# Заключение

В ходе лабораторной работы было разработано программное обеспечение вычисления зависимости $v(t)$ для задачи о падении тела в вязкой среде на основе ОДУ (1) аналитически, а также с помощью численных методов. Была разработана и использована библиотека динамической компоновки реализации численных методов Эйлера и Рунге-Кутты 4-го порядка.

В ходе разработки было удовлетворено требование отсутствия перекомпиляции исходных кодов в случае изменения основного проекта или параметров модели.

Также была предусмотрена возможность неявного задания параметров с помощью интервала или математического ожидания и дисперсии.

Помимо этого, была реализована возможность поиска максимально допустимого шага интегрирования для численных методов.

Была проанализирована зависимость коэффициента трения на итоговую зависимость $v(t)$.

# Список использованных источников


1. **Соколов А.П.** Методы математического моделирования сложных процессов и систем. Лабораторные работы. // Кафедра РК6 (Системы автоматизированного проектирования). МГТУ им. Н.Э. Баумана, Москва, 2018 г., 25 с.
2. **Meyers S**. Effective Modern C++ / Scott Meyers – OReilly, 2014. P. 316
3. **Першин А.Ю.** *Лекции по вычислительной математике (черновик)*. [Электронный ресурс] // Кафедра РК6 (Системы автоматизированного проектирования). МГТУ им. Н.Э. Баумана, Москва, 2020 г., 142 с.
4. **Глазачев В. А.** Использование методов интервального анализа в некоторых задачах линейной алгебры / Глазачев Владимир Александрович // СПБГУ – 2016. 33 с.
5. Среднеквадратичное отклонение. [Электронный ресурс] // https://ru.wikipedia.org/wiki/Среднеквадратичное_отклонение - (Дата обращения: 13.10.2021)