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: «Метода математического моделирования сложных процессов и систем»

Студент	Лукаш Сергей Давидович
Группа	РК6-12М
Тип задания	Лабораторная работа №1
Тема лабораторной работы	Применение библиотек динамической компоновки для реализации вычислительных методов

Студент _____ **Лукаш С.Д.**
подпись, дата *фамилия, и.о.*

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

Оценка _____

Москва, 2022 г.

Оглавление

Задание на лабораторную работу

1. Программно реализовать требуемые в варианте задания численные методы решения СОДУ на языке C++, собрать соответствующие библиотеки динамической компоновки, а также использующую их программу. Объяснить каким образом разработанная программа позволит подключать реализации других методов решения СОДУ без перекомпиляции исходных кодов.

2. Численно решить поставленную задачу Коши с помощью реализованного ПО, используя разные числовые методы.

3. Полученные зависимости представить графически на одной координатной плоскости и представить в отчете.

4. Результаты решения различными методами должны сохраняться в виде одного текстового файла с разделителями (.csv формат). Файл должен быть предоставлен среди других сдаваемых к защите материалов.

Вариант 1.1 (моделирование химической реакции)

Дана СОДУ:

$$\begin{cases} \dot{x}_1 = -k_1 x_1; \\ \dot{x}_2 = k_1 - k_2 x_2; \\ \dot{x}_3 = k_2 x_2; \end{cases}$$

Пусть k_1=0.577, k_2=0.422, а также заданы начальные условия:

$$\begin{cases} x_1(0) = 1; \\ x_2(0) = 0; \\ x_3(0) = 0; \end{cases}$$

Требуемые для реализации численные методы: Рунге-Кутты 4-го порядка, Адамса-Башфорта 2-го порядка.

Цель выполнения лабораторной работы

Цель выполнения лабораторной работы - настроить среду разработки, изучить принципы разработки с использованием библиотек динамической компоновки, изучить принципы передачи произвольных параметров от функции к функции.

Ход выполнения лабораторной работы

Базовая часть

1. Настроить среду разработки.
2. Создать структуру каталогов и настроить файлы для сборки.
3. Изучить принципы работы систем контроля версий и настроить доступ к хранилищу системы контроля версий.
4. Изучить принципы разработки на основе библиотек динамической компоновки.

Реализация метода Рунге-Кутты 4-го порядка

Для работы с СОДУ была разработана функция представленная на листинге 1.

```
float* function(float* x) {
    float* f = (float*)malloc(3*sizeof(float));

    f[0] = -kef1*x[0];
    f[1] = kef1*x[0] - kef2*x[1];
    f[2] = kef2*x[1];

    return f;
}
```

Листинг 1. Функция для работы с СОДУ

Данная функция принимает массив *x*, в котором хранятся значения x_1, x_2, x_3, и возвращает указатель на массив значений $\dot{x}_1$, $\dot{x}_2$, $\dot{x}_3$.

На листинге 2 представлена функция, реализующая метод Рунге-Кутты 4-го порядка. Данная функция принимает на массив *x*, в котором содержатся изначальные значения x_1, x_2, x_3, h – шаг по времени. В данной функции последовательно вычисляются значения коэффициентов k_1, k_2, k_3, k_4, затем на их основе вычисляются следующие значения x_1, x_2, x_3 и записываются в массив *x*.

```
void rk4(float* x, float h) {
    float* k1 = function(x);

    float buf[3];
    buf[0] = x[0] + k1[0]/2.0;
    buf[1] = x[1] + k1[1]/2.0;
    buf[2] = x[2] + k1[2]/2.0;
    float* k2 = function(buf);

    buf[0] = x[0] + k2[0]/2.0;
    buf[1] = x[1] + k2[1]/2.0;
    buf[2] = x[0] + k2[0]/2.0;
    float* k3 = function(buf);

    buf[0] = x[0] + k3[0];
    buf[1] = x[1] + k3[1];
    buf[2] = x[0] + k3[0];
    float* k4 = function(buf);

    x[0] = x[0] + 1.0/6.0*h*(k1[0] + 2.0*k2[0] + 2.0*k3[0] + k4[0]);
    x[1] = x[1] + 1.0/6.0*h*(k1[1] + 2.0*k2[1] + 2.0*k3[1] + k4[1]);
    x[2] = x[2] + 1.0/6.0*h*(k1[2] + 2.0*k2[2] + 2.0*k3[2] + k4[2]);

    free(k1);
    free(k2);
    free(k3);
}
```

Листинг 2. Реализация метода Рунге-Кутты 4-го порядка

Для удобного использования динамических библиотек была объявлена единая для всех методов решения функция, представленная на листинге 3.

```
void solve(float* x, float h, float tStart, float tFinish,
const char* outputFileName);
```

Листинг 3. Сигнатура функции для численных методов решения СОДУ

Данная функция принимает массив *x* начальных значений, *h* – щаг по времени, *tStart* – начало отсчета времени, *tFinish* – конец отсчета времени, *outputFileName* – название файла для вывода результатов. На листинге 4 представлена реализация вышеупомянутой функции для метода Рунге-Кутты 4-го порядка.

```
void solve(float* x, float h, float tStart, float tFinish, const char*
outputFileName) {
    fprintf(stdout, "Runge-Kutta method is used\n");
    FILE *out = fopen(outputFileName, "w");

    if (out == NULL) {
        fprintf(stderr, "Error! Could not open file\n");
        exit(-1);
    }

    size_t it = 1;
    for(float t = tStart; t < tFinish; t+=h) {
        int n = fprintf(out, "%zu,\t%f,\t%f,\t%f\n", it++, x[0], x[1],
x[2]);
        if (n < 0) {
            fprintf(stderr, "Error! Could not write to file\n");
            fclose(out);
            exit(-1);
        }

        rk4(x, h);
    }

    fclose(out);
}
```

Листинг 4. Реализация сигнатуры функции для метода Рунге-Кутты 4-го порядка

В данной функции последовательно вычисляются новые значения x_1, x_2, x_3 и записываются в файл.

Реализация метода Адамса-Башфорта 2-го порядка

Для работы с СОДУ была использована функция, представленная ранее на листинге 1.

Метод Адамса-Башфорта – многошаговый. Для использования этого метода необходимо знать 2 значения. В качестве первого значения можно использовать значение точного решения СОДУ в момент времени соответствующий первому шагу. Для нахождения 2-го значения была реализована функция, реализующая метод Эйлера (листинг 5).

```
float* eulerKickStart(float* x1, float h) {
    float* x2 = (float*)malloc(3*sizeof(float));

    float* f = function(x2);

    x2[0] = x1[0] + h*f[0];
    x2[1] = x1[1] + h*f[1];
    x2[2] = x1[2] + h*f[2];

    free(f);

    return x2;
}
```

Листинг 5. Функция, реализующая метод Эйлера

Данная функция принимает массив *x* начальных значений и *h*-шаг по времени. На основе этих данных находится второе значение.

На листинге 6 представлена функция, реализующая метод Адамса-Башфорта 2-го порядка. Данная функция принимает на массив x_1, в котором содержатся изначальные значения x_1, x_2, x_3, и массив x_2, в котором содержатся следующие значения, соответствующие 2-му шагу, h-шаг по времени. В результате работы функции вычисляются значения, соответствующие 3-му шагу, после чего значения полученные во время второго шага записываются в

массив x_1, а значения, полученные во время 3-го шага, записываются в массив x_2. Функция для перезаписи значений в массивах представлена на листинге 7.

```
void ab2(float* x1, float* x2, float h) {
    float x3[3];

    float* f1 = function(x1);
    float* f2 = function(x2);

    x3[0] = x2[0] + 1.5*h*f2[0] - 0.5*h*f1[0];
    x3[1] = x2[1] + 1.5*h*f2[1] - 0.5*h*f1[1];
    x3[2] = x2[2] + 1.5*h*f2[2] - 0.5*h*f1[2];

    move(x1, x2);
    move(x2, x3);

    free(f1);
    free(f2);
}
```

Листинг 6. Реализация метода Адамса-Башфорта 2-го порядка

```
void move(float* left, float* right) {
    left[0] = right[0];
    left[1] = right[1];
    left[2] = right[2];
}
```

Листинг 7. Функция для перезаписи значений в массивах

На листинге 8 представлена реализация сигнатуры функции, представленной на листинге 3.

```
void solve(float* x1, float h, float tStart, float tFinish, const char*
outputFileName) {
    fprintf(stdout, "Adams–Bashforth method is used\n");
    FILE *out = fopen(outputFileName, "w");

    if (out == NULL) {
        fprintf(stderr, "Error! Could not open file\n");
        exit(-1);
    }

    float* x2 = eulerKickStart(x1, h);

    size_t it = 1;
    for(float t = tStart; t < tFinish; t+=h) {
        int n = fprintf(out, "%zu,\t%f,\t%f,\t%f\n", it++, x1[0],
x1[1], x1[2]);
        if (n < 1) {
            fprintf(stderr, "Error! Could not write to file\n");
            fclose(out);
            free(x2);
            exit(-1);
        }

        ab2(x1, x2, h);
    }

    free(x2);
    fclose(out);
}
```

Листинг 8. Реализация сигнатуры функции для метода Адамса-Башфорта 2-го порядка

Реализация программы, использующей библиотеки динамической компановки

На листинге 9 представлена программа, использующая библиотеки динамической компоновки.

```
#include <stdio.h>
#include <string.h>
#include "dlfcn.h"

const char* ab2Method = "ab2";
const char* ab2LibLocation = "ab2/libab2.dylib";

const char* rk4Method = "rk4";
const char* rk4LibLocation = "rk4/librk4.dylib";

int main(int argc, char *argv[]) {
    if (argc < 3) {
        fprintf(stderr, "You need to pass two arguments to function\n");
        return -1;
    }

    const char* method = argv[1];
    const char* libLocation;
    if (strcmp(method, ab2Method) == 0) {
        libLocation = ab2LibLocation;
    }
    if (strcmp(method, rk4Method) == 0) {
        libLocation = rk4LibLocation;
    }
    if (strcmp(libLocation, "") == 0) {
        fprintf(stderr, "You need to chose one of two methods: ab2 or rk4\n");
        return -1;
    }

    const char* fileName = argv[2];

    void* lib;
    void (*solve)(float* x, float h, float tStart, float tFinish, const char* outputFileName);

    lib = dlopen(libLocation, RTLD_LAZY);
    if (!lib) {
        fprintf(stderr, "Error: %s\n", dlerror());
        return -1;
```

```
    }

    *(void**)(&solve) = dlsym(lib, "solve");
    if (!solve) {
        fprintf(stderr, "Error: %s\n", dlerror());
        dlclose(lib);
        return -1;
    }

    float x[3];
    // x1(0)=1;
    x[0] = 1;
    // x2(0)=0;
    x[1] = 0;
    // x3(0)=0;
    x[2] = 0;

    float tStart = 0;
    float tFinish = 20;
    float h = 0.001;

    solve(x, h, tStart, tFinish, fileName);
    dlclose(lib);

    return 0;
}
```

Листинг 9. Программа использующая библиотеки динамической компановки

Для загрузки библиотек в адресное пространство используется функция *dlopen*, которая принимает имя библиотеки. Функция *dlsym* позволяет найти функцию в динамической библиотеке, эта функция принимает название функции и возвращает указатель на эту функцию. Для использования функции необходимо преобразовать ее к типу, который указан в динамической библиотеке.

Сборка библиотек динамической компоновки и использующей их программы

Структура каталогов проекта представлена на рисунке 1.

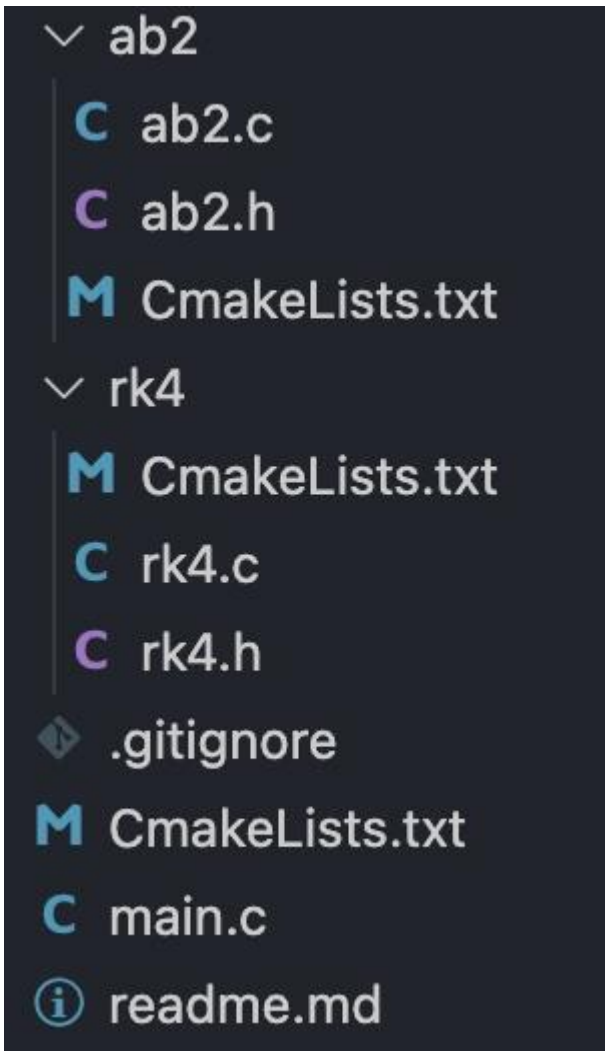

Рисунок 1. Структура каталогов проекта

На листинге 10 представлен один из файлов сборки библиотек.

```
cmake_minimum_required(VERSION 3.24.2)
project(lab1_ab2)
set(AB2_SOURCE_LIB ab2.c)
add_library(ab2 SHARED ${AB2_SOURCE_LIB})
```

Листинг 10. Файл сборки динамической библиотеки

В данном файле указываются файлы, которые необходимо скомпилировать в виде динамической библиотеки. Аналогичный файл используется для второго метода.

На листинге 11 представлен файл сборки программы, использующей библиотеки динамической компоновки.

```
cmake_minimum_required(VERSION 3.24.2)
project(lab1)
set(SOURCE main.c)
add_subdirectory(ab2)
add_subdirectory(rk4)
add_executable(main ${SOURCE})
```

Листинг 11. Файл сборки программы, использующей библиотеки динамической компоновки

Результаты работы программы

Результатом работы программы является файл с расширением .csv, в котором находятся координаты полученных точек. Пример файла представлен на листинге 12.

```
0,1,0,0
1,0.9999,0.0001,0.0001
2,0.9998,0.0002,0.0001
3,0.9997,0.0003,0.0002
```

Листинг 12. Пример файла, содержащего результаты работы программы

На рисунке 2 представлен результат работы метода Рунге-Кутты 4-го порядка.

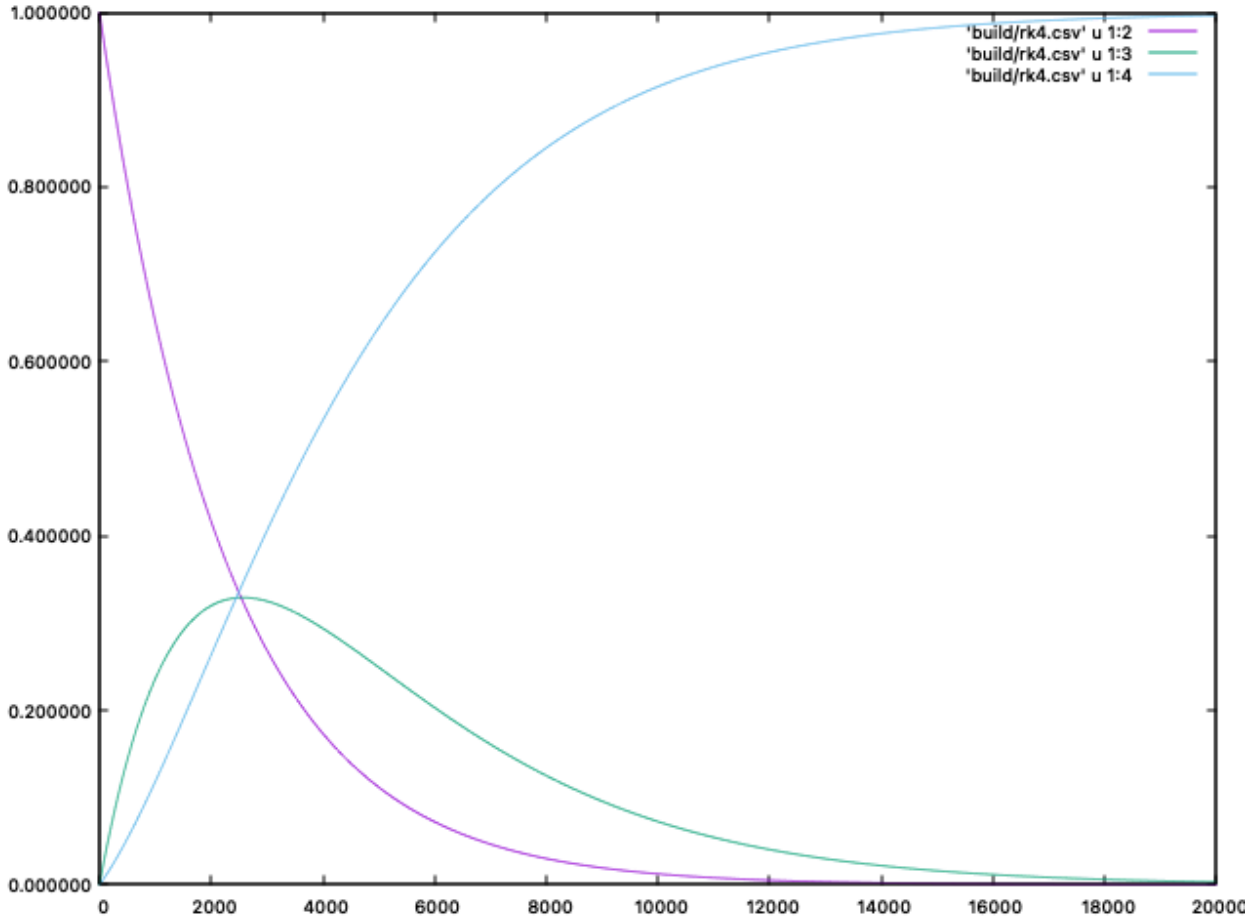

Рисунок 2. Графическое отображение результата работы метода Рунге-Кутты 4-го порядка

На рисунке 3 изображен результат работы метода Адамса-Башфорта 2-го порядка.

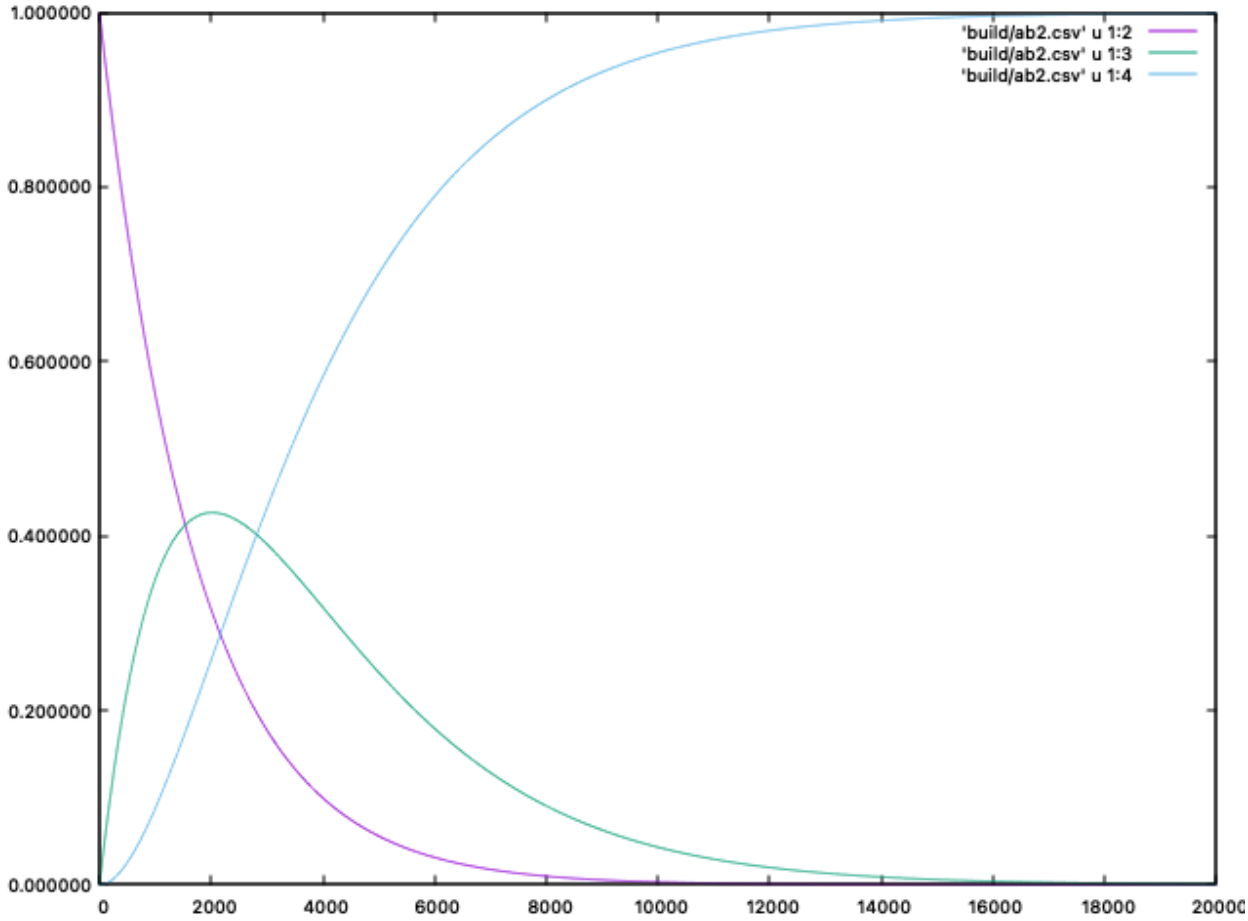

Рисунок 3. Графическое отображение результата работы метода Адамса-Башфорта 2-го порядка

На рисунке 4 изображено сравнение графиков, полученных разными методами.

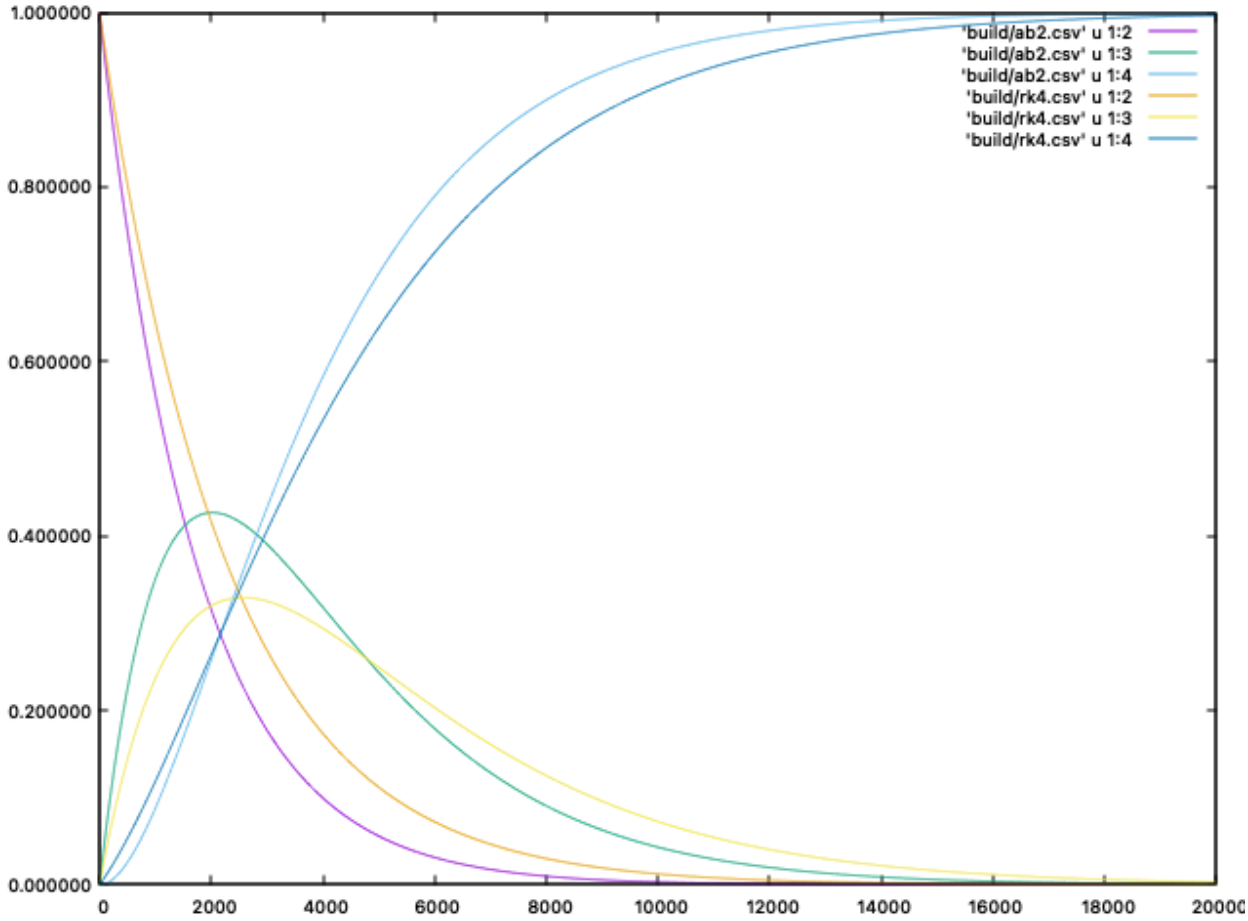

Рисунок 4. Сравнение графиков, полученных методами Рунге-Кутты 4-го порядка и Адамса-Башфорта 2-го порядка

Выводы

В данной работе были изучены принципы работы библиотек динамической компоновки на примере решения СОДУ различными численными методами.

1. Были реализованы численные методы решения СОДУ.
2. Были разработаны библиотеки динамической сборки.
3. Были представлены графически результаты работы методов решения СОДУ.

Список использованных источников

1. Документация функции dlsym // Электронный ресурс https://linux.die.net/man/3/dlsym
   (Дата обращения: 04.10.2022).
2. Документация функции dlopen // Электронный ресурс https://linux.die.net/man/3/dlsym
   (Дата обращения: 04.10.2022).