**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 | Абидоков Рашид Ширамбиевич |
| Группа | РК6-11М |
| Тип задания | лабораторная работа |
| Тема лабораторной работы | Применение библиотек динамической компоновки для разработки программных реализаций вычислительных методов |

Студент _____ **Абидоков Р. Ш.**
*подпись, дата* *фамилия, и.о.*

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

Оценка _____

*Москва, 2020 г.*

Оглавление

## Задание на лабораторную работу

Тело, имеющее в начальный момент времени $T(0) = T_0$, поместили в среду, температура которой поддерживается неизменной и равна $T_1$.

Экспериментально установлено, что при определенных упрощениях скорость изменения температуры тела пропорциональна разности температур тела и окружающей среды.

Требуемые для реализации численные методы: метод Эйлера, метод Хъюна.

## Цель выполнения лабораторной работы

**Цель выполнения лабораторной работы** – получение практических навыков разработки и применения библиотек динамической компоновки на примере различных реализаций задачи численного решения ОДУ.

## Выполненные задачи

1. По имеющейся математической модели в форме ОДУ получено аналитическое решение
2. На языке C++ написана программа, реализующая возможность численного решения ОДУ с использованием динамически подключаемых библиотек, реализующих методы Ньютона и Хойна. Обеспечена возможность подключения дополнительных методов решения
3. Проведено графическое сравнение методов Ньютона и Хойна с аналитическим решением
4. Вычислены максимально допустимые шаги интегрирования по времени для примененных численных методов
5. Предложен вариант доработки алгоритма для решения с заданием одного из параметров модели в интервальном виде

# 1. Аналитическое решение

Модель процесса представлена в виде ОДУ

$$\frac{dT(t)}{dt} = -\gamma(T(t) - T_1) \tag{1}$$

где $\gamma > 0$ – некоторый коэффициент пропорциональности,

$T(t)$ – температура тела в момент времени $t$

$T_1$ – температура окружающей среды

Аналитическим решением данного уравнения является

$$T(t) = T_1 + (T_0 - T_1)e^{-\gamma t} \tag{2}$$

где $T_0$ – температура тела в начальный момент времени

# 2. Описание программной реализации

Исходный код программы разделен на файлы main.cpp, euler.cpp, heun.cpp, config.hpp и config.cpp. Файлы euler.cpp и heun.cpp являются реализациями методов Эйлера и Хойна соответственно и компилируются в динамические библиотеки euler.dll, heun.dll. Их сходный код приведен в Листингах 1 и 2.

Листинг 1. Метод Эйлера

```cpp
#include <vector>
#include <utility>

using namespace std;

// Принимает функцию производной
extern "C" __declspec(dllexport)
vector<pair<double, double> >& solver(double (*der_func)(double), double y0, double begin, double end, unsigned n_steps) {
  double step = (end - begin) / n_steps;
  vector<pair<double, double> >& ret = *(new vector<pair<double, double> >());

  double y_curr = y0;
  for (unsigned i = 0; i <= n_steps; i++) {
    ret.push_back(pair<double, double>(begin + i * step, y_curr));
    y_curr += der_func(y_curr) * step;
  };

  return ret;
}
```

Листинг 2. Метод Хойна

```cpp
#include <vector>
#include <utility>

using namespace std;

// Принимает функцию производной
extern "C" __declspec(dllexport)
vector<pair<double, double> >& solver(double (*der_func)(double), double y0, double begin, double end, unsigned n_steps) {
  double step = (end - begin) / n_steps;
  vector<pair<double, double> >& ret = *(new vector<pair<double, double> >());

  double y_curr = y0;
  double y_sh = y0;
  for (unsigned i = 0; i <= n_steps; i++) {
    ret.push_back(pair<double, double>(begin + i * step, y_curr));
    y_sh = y_curr + step * der_func(y_curr);
    y_curr += step / 2 * (der_func(y_curr) + der_func(y_sh));
  };

  return ret;
}
```

В main.cpp описана основная логика программы – чтение конфигурации из config.txt (в случае его отсутствия, создается файл по умолчанию), вызов решателей из подключаемых библиотек, запись полученных численных и аналитического результатов в выходной .csv-файл.

В свою очередь, config.hpp и config.cpp являются статической библиотекой, реализующей функционал работы с config-файлом.

Листинг 3. Структура config.txt

```
        # Имена библиотек с решателями без .dll
        # В каждом файле должен быть определена функция vector<pair<double, double> >& solver(double (*)(double), doub
solv_names=euler,heun
        # Тип задания коэффициента гамма математической модели
        # 1 для точного значения, 2 для интервала, 3 для M(gam) и D(gam)
gam_type=1
        # Коэффициент гамма математической модели. Для gam_type=1 - одно число, для 2 и 3 - два числа через запятую
gam=0.1
        # Температура окружающей среды
T_s=25.
        # Начальная температура тела
T_beg=150.
        # Начальное время
time_beg=0.
        # Конечное время
time_end=100.
        # Желаемое количество шагов интегрирования
n_steps=1000
        # Имя выходного .csv файла
out_file_name=out.csv
        # Разделитель в выходном файле
csv_dlm=,
```

В config.txt задаются имена используемых файлов решателей, параметры математической модели, имя выходного файла и его разделитель.

## 3. Графическое сравнение решений

Графики построены с использованием библиотеки matplotlib языка python. Здесь и в дальнейшем интервал интегрирования $[0, 100]$ секунд. Приведенные графики построены с количеством шагов 100, шаг интегрирования 1 с для наглядной разницы между решениями. Остальные параметры по умолчанию (см. Листинг 3)

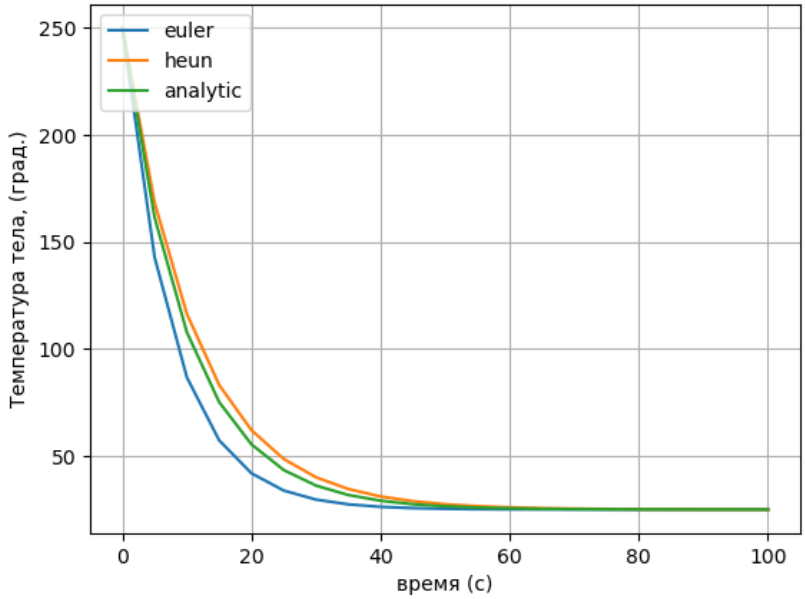


Рис 1.

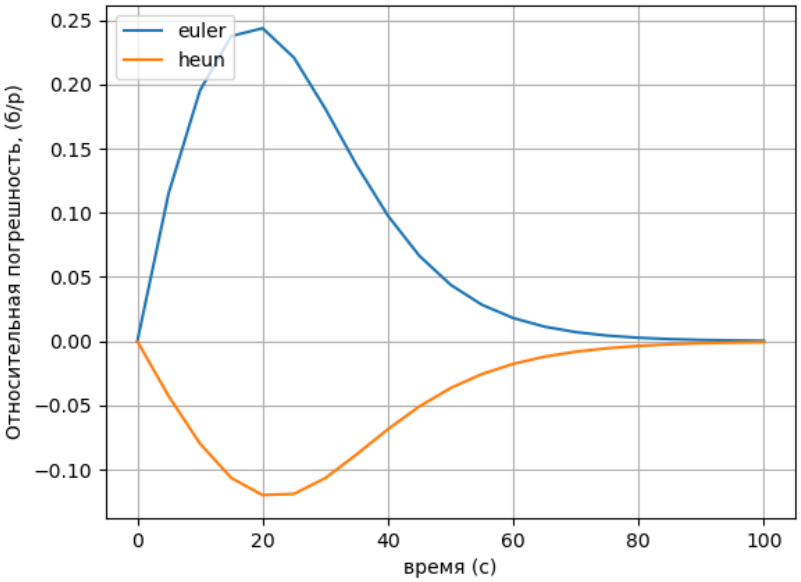


Рис 2.

Как можно видеть, метод Хойна обеспечивает лучшую точность, нежели метод Эйлера.

## 4. Вычисление максимально допустимых шагов интегрирования

Расчеты проводились при помощи скриптов на языке python с использованием библиотек numpy и pandas. Интервал интегрирования везде одинаковый, изменялось количество шагов интегрирования. Для каждого случая вычислялась величина шага и следующая метрика, основанных на L2-норме:

$$Q(h) = \frac{1}{N_h}\left[\sum_{i=1}^{N_h}\left(\frac{T_{an_i} - T_{num_i}}{T_{an_i}}\right)^2\right]^{\frac{1}{2}} \quad (3)$$

т.е., фактически, значение Евклидовой длины вектора относительных ошибок, разделенное на размерность этого вектора.

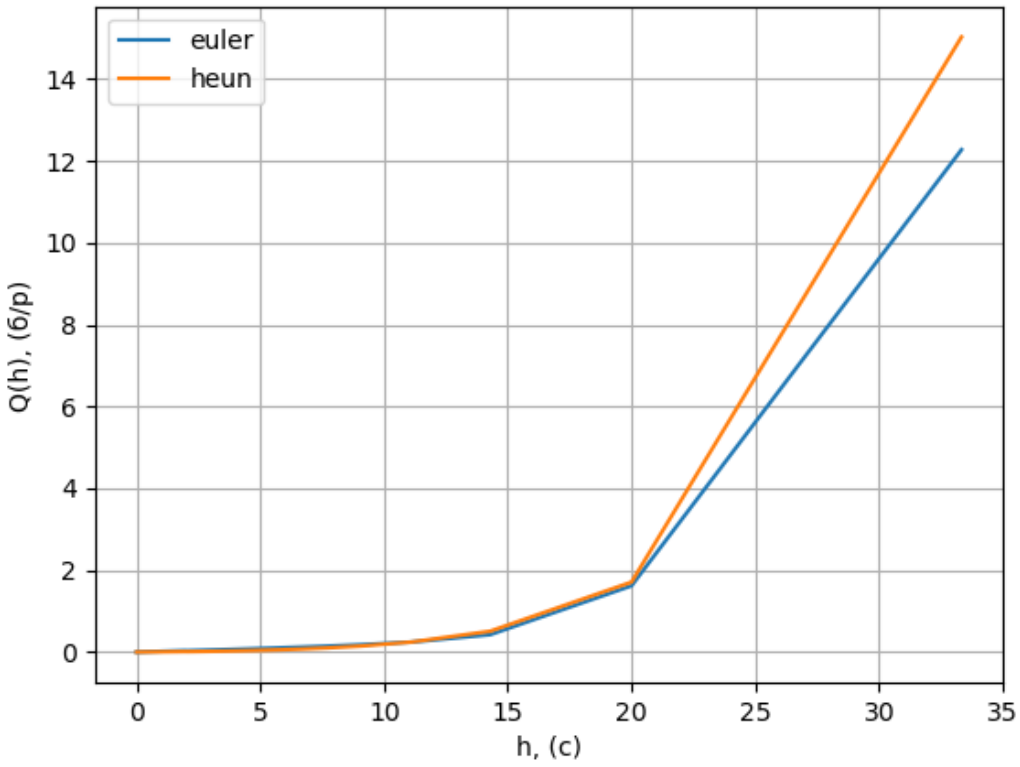


Рис 3. Большой масштаб

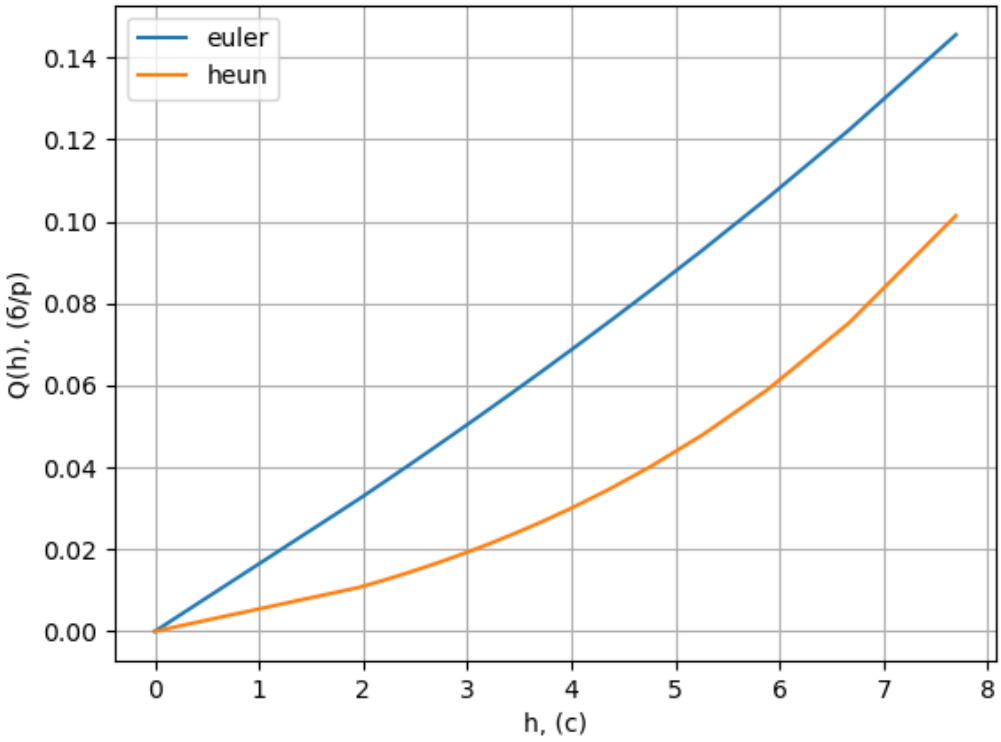


Рис 3. Малый масштаб

Откуда видно, что для обеспечения точности порядка 5%, максимально допустимыми шагами интегрирования являются 3 секунды для метода Эйлера и порядка 5.5 секунд для метода Хойна соответственно.

## 5. Вариант реализации решения задачи с интервально заданными параметрами

Данный раздел является размышлениями на тему "что можно было бы сделать, получив на вход параметр, заданный в интервальном виде либо матожиданием и дисперсией".

*Один из параметров задан интервалом.* Например, $\gamma \in [\gamma_1,\ \gamma_2]$. В этом случае интерес может представлять интервал значений некоторой выходной функции $F\big(T(t,\gamma)\big)$, где $T(t,\gamma)$ – решение исходного уравнения. Т.е. фактически задача будет представлять собой поиск $F_{min}\big(T(t,\gamma)\big)$ и $F_{max}\big(T(t,\gamma)\big)$, в простейшем случае просто минимальное и максимальное возможные значения $T(t,\gamma)$.

Здесь возможно применение методов оптимизации, к примеру, градиентных. Саму производную можно было бы вычислить численно, если предположить, что уравнение может быть интерполировано как по $t$, так и по $\gamma$.

Задаемся некоторым начальным приближением $(t_0,\ \gamma_0)$, численно решаем уравнение с различными $\gamma$ вокруг $\gamma_0$ (кол-во зависит от выбранного выражения производной), находим градиент искомой ф-ии, сдвигаемся и т.д. до тех пор, пока не найдем min/max либо не упремся в границу.

*Один из параметров задан математическим ожиданием и дисперсией.* Т.е. задано распределение вероятности параметра (предполагается, что оно имеет вид нормального распределения). В этом случае может представлять интерес распределение вероятности некоторой выходной функции $F\big(T(t,\gamma)\big)$.

В этом случае можно было бы выбрать некоторый интервал, например, $[M_\gamma - 3\sigma_\gamma,\ M_\gamma + 3\sigma_\gamma]$ и, равномерно разбив его на участки, для каждого найти численное решение и значение искомой функции

$$F_i = F\big(T(t,\gamma_i)\big) \tag{4}$$

И получить таким образом распределение $\hat{f}_F$ (предположив его вид, например, также нормальное распределение), если бы $\gamma$ имела постоянную вероятность внутри выбранного интервала. Т.е. фактически у нас имелось бы правило, по которому можно было преобразовать исходное распределение $f_\gamma$ и получить итоговое $f_F$, но для предложения конкретного алгоритма вопрос требуется изучить более подробно.

## Заключение

Реализована программа, осуществляющая численное решение ОДУ и сравнение полученных решений с аналитическим. Показано, что метод Хойна обеспечивает лучшую по сравнению с методом Эйлера точность. Получен практический опыт использования динамических библиотек позднего связывания.

Найдены предельно допустимые шаги интегрирования по времени, составляющие 3 секунды для метода Эйлера и порядка 5.5 секунд для метода Хойна соответственно.

Приведены возможные способы решения задачи с интервально заданным параметром.