|  |  |
| --- | --- |
|  | **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 Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

**Лабораторная работа № 5**

«Численное решение краевой задачи для линейного дифференциального уравнения второго порядка методом прогонки»

по курсу «Численные методы»

Выполнил:

студент группы ИУ9-62Б

Марченко А.И.

Проверила:

Домрачева А. Б.

Москва, 2024

## Оглавление

[Постановка задачи 3](#_Toc2956)

[Теоретические сведения 3](#_Toc2957)

[Реализация 3](#_Toc2958)

[Результаты 5](#_Toc2959)

[Выводы 5](#_Toc2960)

# Постановка задачи

**Дано:** краевая задача для линейного дифференциального уравнения второго порядка

**Найти** аналитическое решение задачи Коши:

По найденному решению задачи Коши вычислить

С помощью метода прогонки найти численное решение краевой задачи для того же уравнения с краевыми условиями

Вычислить , найти погрешность численного решения .

**Индивидуальный вариант:**

# Теоретические сведения

Пусть требуется решить краевую задачу на отрезке [0, 1]. Тогда отрезок разбивается на n равных отрезков длины .

Приближенным численным решением краевой задачи для ДУ второго порядка называется сеточная функция , заданная в точках

Обозначим значения коэффициентов уравнения в точках . Применяя разностную аппроксимацию производных по формулам численного дифференцирования получаем приближенную систему уравнений относительно ординат сеточной функции:

После преобразования система имеет вид:

с краевыми условиями . Данная система имеет порядок 𝑛 − 1 и представляет собой трехдиагональную систему линейных алгебраических уравнений. Ее следует решать методом прогонки.

# Реализация

Аналитическое решение задачи Коши:

#include <cmath>  
#include <vector>  
#include <iostream>  
#include <iomanip>  
  
double f(double x) {  
 return 2 \* x;  
}  
  
double check(double x) {  
 return x + std::exp(x) \* std::sin(x) - std::exp(x) \* std::cos(x) + 1;  
}  
  
struct equation {  
 std::vector <double> as;  
 std::vector <double> bs;  
 std::vector <double> cs;  
 std::vector <double> result;  
};  
  
std::vector<double> run(const std::vector<double>& as, const std::vector<double>& bs, const std::vector<double>& cs, const std::vector<double>& d) {  
 int n = d.size();  
 std::vector<double> x(n, 0), v(n, 0), u(n, 0);  
 v[0] = -cs[0] / bs[0];  
 u[0] = d[0] / bs[0];  
 for (int i = 1; i < n; ++i) {  
 v[i] = -cs[i] / (as[i] \* v[i - 1] + bs[i]);  
 u[i] = (d[i] - as[i] \* u[i - 1]) / (as[i] \* v[i - 1] + bs[i]);  
 }  
 x[n - 1] = u[n - 1];  
 for (int i = n - 2; i >= 0; --i) {  
 x[i] = v[i] \* x[i + 1] + u[i];  
 }  
 return x;  
}  
  
equation make\_equation(double h, double p, double q, int n, double x\_0, double a, double b) {  
 std::vector<double> as, bs, cs, result;  
 bs.push\_back(h \* h \* q - 2);  
 cs.push\_back(1 + h / 2 \* p);  
 result.push\_back(h \* h \* f(x\_0 + h) - a \* (1 - h / 2 \* p));  
 as.push\_back(0);  
 for (int i = 2; i < n - 1; ++i) {  
 as.push\_back(1 - h / 2 \* p);  
 bs.push\_back(h \* h \* q - 2);  
 cs.push\_back(1 + h / 2 \* p);  
 result.push\_back(h \* h \* f(x\_0 + h \* i));  
 }  
 as.push\_back(1 - h / 2 \* p);  
 bs.push\_back(h \* h \* q - 2);  
 result.push\_back(h \* h \* f(x\_0 + (n - 1) \* h) - b \* (1 + h / 2 \* p));  
 cs.push\_back(0);  
 return {as, bs, cs, result};  
}  
  
int main() {  
 double p = -2;  
 double q = 2;  
 int n = 10;  
 double x\_0 = 0;  
 double x\_n = 1;  
 double h = (x\_n - x\_0) / n;  
 double a = check(x\_0);  
 double b = check(x\_n);  
  
 auto [as, bs, cs, result] = make\_equation(h, p, q, n, x\_0, a, b);  
 std::vector<double> x\_arr(n + 1), y\_true\_arr(n + 1), y\_ev\_arr, delta, y\_gun, delta\_gun;  
 for (int i = 0; i <= n; ++i) {  
 x\_arr[i] = x\_0 + i \* h;  
 y\_true\_arr[i] = check(x\_0 + i \* h);  
 }  
 y\_ev\_arr.push\_back(a);  
 auto temp = run(as, bs, cs, result);  
 y\_ev\_arr.insert(y\_ev\_arr.end(), temp.begin(), temp.end());  
 y\_ev\_arr.push\_back(b);  
 double max\_err = 0;  
 for (int i = 0; i <= n; ++i) {  
 delta.push\_back(std::abs(y\_true\_arr[i] - y\_ev\_arr[i]));  
 max\_err = std::max(delta[i], max\_err);  
 }  
 std::cout << "Max error:" << '\t' << max\_err << std::endl;  
 std::cout << std::fixed << std::setprecision(6) << "x\t\t" << "Analytical res" << '\t' << "Method's res" << '\t' << "Delta" << std::endl;  
  
 for (int i = 0; i < delta.size(); i++) {  
 std::cout << x\_arr[i] << "\t" << y\_true\_arr[i] << "\t" << y\_ev\_arr[i] << "\t" << delta[i] << std::endl;  
 }  
 std::cout << std::endl;  
 return 0;  
}

# Результаты

x Analytical Method diff

0.000000 0.000000 0.000000 0.000000

0.100000 0.110683 0.110118 0.000565

0.200000 0.245599 0.244482 0.001117

0.300000 0.409341 0.407717 0.001624

0.400000 0.606882 0.604832 0.002051

0.500000 0.843550 0.841198 0.002352

0.600000 1.124986 1.122508 0.002478

0.700000 1.457092 1.454721 0.002371

0.800000 1.845956 1.843986 0.001970

0.900000 2.297759 2.296556 0.001204

1.000000 2.818661 2.818661 0.000000

# Выводы

В процессе выполнения лабораторной работы был использован метод приближенного численного решения краевой задачи для дифференциального уравнения второго порядка: метод прогонки.

Можно заметить, что ошибка увеличивается от нулевого значения в граничных точках до максимального значения в середине отрезка, а общая ошибка не превышает трех тысячных в абсолютном значении, что является хорошим результатом в данной задаче.