**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5**

**РЕШЕНИЕ ОДУ**

**(Вариант 9)**

*Выполнил студент 3 курса МОиАИС*

*Ходосевич Данила*

***Цель работы***: усвоить сущность и методы решения ***обыкновенных дифференциальных уравнений***. Овладеть технологией решения обыкновенного дифференциального уравнения.

Численное решение дифференциального уравнения предполагает получение числовой таблицы приближенных значений *yi* искомой функции *y* = *f*(*x)* с заданной точностью для некоторых значений аргумента *xi * [*a*, *b*].

Численное решение обыкновенных дифференциальных уравнений возможно методами:

метод Эйлера (первого порядка точности),

модифицированный метод Эйлера-Коши (второго порядка точности)

методы Рунге-Кутты

методы Адамса.

***Метод Рунге-Кутты*** четвёртого порядка точности имеет вид.

*k1* = *hf*(*xk*, *yk*),

*k2* = *hf*(*xk* + *h*/2, *yk* + *k1*/2),

*k3* = *hf*(*xk* + *h*/2, *yk* + *k2*/2),

*k4* = *hf*(*xk* + *h*, *yk* + *k3*),

*yk*=1/6(*k1* + *2k2* + *2k3* + *k4*),    *yk* + 1=*yk* + *yk*,    *xk* + 1=*xk* + *h.*

***Методы Адамса*** третьего и четвертого порядков точности имеют вид

*yi + 1 = yi + h (23y'i - 16y'i-1 + 5y'i-2)/12;*

*yi + 1 = yi + h (55y'i - 59y'i-1 + 37y'i-2 - 9y'i-3)/24.*

Погрешность решения, найденного этими методами, оценивается величиной O(*hm*)*,* где *m* - порядок метода.

Таким образом, метод Рунге-Кутта 4-го порядка и метод Адамса четвертого порядка имеют одинаковую оценку погрешности, но метод Адамса требует примерно вчетверо меньшего объема вычислений.

***Задание.***

**Для всех заданий точность 0,001**

Решить уравнение 1 методом Эйлера 2-го порядка точности (т.е. методом Эйлера-Коши) и методом Рунге-Кутта 4-го порядка точности.

Решить уравнение 2 методами Адамса 3-го порядка точности и 4-го порядка точности. ВНИМАНИЕ! Уравнение 2 - это дифференциальное уравнение 2-го порядка. Подробно расписать как решается уравнение.

Точность вычислений и для первого, и для второго уравнения контролировать методом двойного пересчета.

Сущность метода состоит в последовательных итерациях, каждая следующая из них соответствует удвоению числа точек разбиения. Сравниваются значения в совпадающих узлах. Вычисления прекращаются, когда максимальной модуль разности значений функции в совпадающих узлах для двух итераций становится меньше заранее заданной малой величины.

Результаты вывести в виде таблиц последних 16 точек для последней и 8 точек для предпоследней итераций, в которых первая колонка значения Хk , вторая колонка – значения найденных Yk для предпоследней итерации, третья - значения найденных Yk для последней итерации, четвертая – разность значений из 2-й и 3-й колонок.

Указать число точек разбиения для последней итерации.

|  |  |  |
| --- | --- | --- |
| 9 | *y* = 1 - s*in*(*2x* + *y*) | *y* = 1 + (1 - *x*)s*in y* |

Для всех вариантов и уравнений y(*a*) = 0, [*a*, *b*] = [0; 0,5], для уравнения 2 - y(*a*) = 1. Точность решения 0,001

**Задание 1**

**Метод Эйлера:**

**Количество разбиений: 128**

|  |  |  |  |
| --- | --- | --- | --- |
| Xk | Yk (последняя итерация) | Yk (предпоследняя итерация) | Разность |
| 0,221 | 0,154 |  |  |
| 0,223 | 0,155 | 0,154 | 0,001 |
| 0,225 | 0,156 |  |  |
| 0,227 | 0,156 | 0,156 | 0 |
| 0,229 | 0,157 |  |  |
| 0,230 | 0,158 | 0,157 | 0,001 |
| 0,232 | 0,159 |  |  |
| 0,234 | 0,160 | 0,159 | 0,001 |
| 0,236 | 0,161 |  |  |
| 0,238 | 0,161 | 0,161 | 0 |
| 0,240 | 0,162 |  |  |
| 0,242 | 0,163 | 0,162 | 0,001 |
| 0,244 | 0,164 |  |  |
| 0,246 | 0,164 | 0,164 | 0 |
| 0,248 | 0,165 |  |  |
| 0,250 | 0,166 | 0,165 | 0,001 |

**Метод Рунге-Кутты:**

**Количество разбиений: 128**

|  |  |  |  |
| --- | --- | --- | --- |
| Xk | Yk (последняя итерация) | Yk (предпоследняя итерация) | Разность |
| 0,221 | 0,154 |  |  |
| 0,223 | 0,155 | 0,154 | 0,001 |
| 0,225 | 0,156 |  |  |
| 0,227 | 0,156 | 0,156 | 0 |
| 0,229 | 0,157 |  |  |
| 0,230 | 0,158 | 0,157 | 0,001 |
| 0,232 | 0,159 |  |  |
| 0,234 | 0,160 | 0,159 | 0,001 |
| 0,236 | 0,161 |  |  |
| 0,238 | 0,161 | 0,161 | 0 |
| 0,240 | 0,162 |  |  |
| 0,242 | 0,163 | 0,162 | 0,001 |
| 0,244 | 0,164 |  |  |
| 0,246 | 0,164 | 0,164 | 0 |
| 0,248 | 0,165 |  |  |
| 0,250 | 0,166 | 0,165 | 0,001 |

**Задание 2**

|  |  |  |
| --- | --- | --- |
| 9 | *y* = 1 - s*in*(*2x* + *y*) | *y* = 1 + (1 - *x*)s*in y* |

Для всех вариантов и уравнений y(*a*) = 0, [*a*, *b*] = [0; 0,5], для уравнения 2 - y(*a*) = 1.

)

)

)

Найдём :

**Метод Адамса 3-го порядка:**

**Количество разбиений: 129**

|  |  |  |  |
| --- | --- | --- | --- |
| Xk | Yk (последняя итерация) | Yk (предпоследняя итерация) | Разность |
| 0,221 | 0,246 |  |  |
| 0,223 | 0,248 | 0,247 | 0,001 |
| 0,225 | 0,251 |  |  |
| 0,227 | 0,253 | 0,252 | 0,001 |
| 0,229 | 0,256 |  |  |
| 0,230 | 0,258 | 0,257 | 0,001 |
| 0,232 | 0,260 |  |  |
| 0,234 | 0,263 | 0,262 | 0,001 |
| 0,236 | 0,265 |  |  |
| 0,238 | 0,268 | 0,267 | 0,001 |
| 0,240 | 0,270 |  |  |
| 0,242 | 0,273 | 0,272 | 0,001 |
| 0,244 | 0,275 |  |  |
| 0,246 | 0,278 | 0,277 | 0,001 |
| 0,248 | 0,280 |  |  |
| 0,250 | 0,283 | 0,282 | 0,001 |

**Метод Адамса 4-го порядка:**

**Количество разбиений: 8193**

|  |  |  |  |
| --- | --- | --- | --- |
| Xk | Yk (последняя итерация) | Yk (предпоследняя итерация) | Разность |
| 0,2482 | 0,5410 |  |  |
| 0,2483 | 0,5413 | 0,5405 | 0,0008 |
| 0,2484 | 0,5416 |  |  |
| 0,2485 | 0,5420 | 0,5412 | 0,0008 |
| 0,2487 | 0,5423 |  |  |
| 0,2488 | 0,5426 | 0,5418 | 0,0008 |
| 0,2489 | 0,5429 |  |  |
| 0,2490 | 0,5433 | 0,5425 | 0,0008 |
| 0,2491 | 0,5436 |  |  |
| 0,2493 | 0,5439 | 0,5431 | 0,0008 |
| 0,2494 | 0,5442 |  |  |
| 0,2495 | 0,5446 | 0,5438 | 0,0008 |
| 0,2496 | 0,5449 |  |  |
| 0,2498 | 0,5452 | 0,5444 | 0,0008 |
| 0,2499 | 0,5455 |  |  |
| 0,2500 | 0,5459 | 0,5451 | 0,0008 |

**Приложения:**

1. **Метод Эйлера**
2. #include <iostream>
3. #include <cmath>
4. #include <map>
5. #include <iomanip>
6. // Определение правой части уравнения
7. double f(double x, double y) {
8. return 1 - sin(2 \* x + y);
9. }
10. // Метод Эйлера-Коши 2-го порядка точности
11. std::map<double, double> eulerCauchy(double a, double b, int n, double y0) {
12. double h = (b - a) / n; // Шаг интегрирования
13. double x = a;
14. double y = y0;
15. std::map<double, double> EC;
16. while (x < b) {
17. double y\_temp = y + h \* f(x, y);
18. y += h \* (f(x, y) + f(x + h, y\_temp)) / 2;
19. x += h;
20. EC[x] = y;
21. }
22. return EC;
23. }
24. // Проверка точности
25. bool checkAccuracy(const std::map<double, double>& y1, const std::map<double, double>& y2, double eps) {
26. for (const auto& pair : y1) {
27. double x = pair.first;
28. if (y2.count(x)) {
29. if (fabs(y1.at(x) - y2.at(x)) >= eps) {
30. return false;
31. }
32. }
33. }
34. return true;
35. }
36. int main() {
37. double a = 0.0;
38. double b = 0.25;
39. int n = 2;
40. double y0 = 0.0; // Начальное условие y(0) = 0
41. double eps = 0.001; // Точность
42. std::map<double, double> not\_checked;
43. std::map<double, double> checked;
44. std::cout << "Метод Эйлера-Коши 2-го порядка:\n";
45. not\_checked = eulerCauchy(a, b, n, y0);
46. while (true) {
47. checked = not\_checked;
48. n \*= 2;
49. not\_checked = eulerCauchy(a, b, n, y0);
50. if (checkAccuracy(checked, not\_checked, eps)) {
51. break;
52. }
53. }
54. int count = not\_checked.size();
55. // Получаем последние 16 точек
56. int last\_points\_count = std::min(16, count);
57. // Вывод данных в консоль
58. std::cout << std::setw(10) << "x"
59. << std::setw(10) << "y1"
60. << std::setw(10) << "y2"
61. << std::setw(12) << "Difference" << '\n';
62. std::cout << std::string(40, '-') << '\n';
63. int i = 0;
64. for (auto it = not\_checked.rbegin(); it != not\_checked.rend() && i < last\_points\_count; ++it, ++i) {
65. double x = it->first;
66. double y1\_value = round(it->second \* 1000) / 1000.0;
67. double y2\_value = checked.count(x) ? round(checked.at(x) \* 1000) / 1000.0 : 0.0;
68. std::cout << std::setw(10) << std::fixed << std::setprecision(3) << x
69. << std::setw(10) << y1\_value;
70. if (checked.count(x)) {
71. double difference = round(fabs(y1\_value - y2\_value) \* 1000) / 1000.0;
72. std::cout << std::setw(10) << y2\_value
73. << std::setw(12) << difference << '\n';
74. } else {
75. std::cout << std::setw(10) << ""
76. << std::setw(12) << "" << '\n';
77. }
78. }
79. std::cout << "Количество точек на последней итерации равно: " << count << '\n';
80. return 0;
81. }

**2. Метод Рунге-Кутты**

#include <iostream>

#include <cmath>

#include <map>

#include <iomanip>

// Определение правой части уравнения

double f(double x, double y) {

    return 1 - std::sin(2 \* x + y);

}

// Метод Рунге-Кутта 4-го порядка точности

std::map<double, double> runge\_kutta(double a, double b, int n, double y0) {

    double h = (b - a) / n; // Шаг интегрирования

    double x = a;

    double y = y0;

    std::map<double, double> RK;

    while (x < b) {

        double k1 = h \* f(x, y);

        double k2 = h \* f(x + h / 2, y + k1 / 2);

        double k3 = h \* f(x + h / 2, y + k2 / 2);

        double k4 = h \* f(x + h, y + k3);

        y += (k1 + 2 \* k2 + 2 \* k3 + k4) / 6;

        x += h;

        RK[x] = y;

    }

    return RK;

}

// Проверка точности результатов

bool check\_accuracy(const std::map<double, double>& y1, const std::map<double, double>& y2, double eps) {

    for (const auto& [x, y1\_value] : y1) {

        if (y2.find(x) != y2.end()) {

            if (std::abs(y1\_value - y2.at(x)) >= eps) {

                return false;

            }

        }

    }

    return true;

}

int main() {

    double a = 0;

    double b = 0.25;

    int n = 2;

    double y0 = 0; // Начальное условие y(0) = 0

    double eps = 0.001; // Точность

    std::map<double, double> not\_checked;

    std::map<double, double> checked;

    std::cout << "Метод Рунге-Кутта 4-го порядка:" << std::endl;

    not\_checked = runge\_kutta(a, b, n, y0);

    while (true) {

        checked = not\_checked;

        n \*= 2;

        not\_checked = runge\_kutta(a, b, n, y0);

        if (check\_accuracy(checked, not\_checked, eps)) {

            break;

        }

    }

    int count = not\_checked.size();

    // Получаем последние 16 точек

    int last\_points\_count = std::min(16, count);

    // Вывод данных в консоль

    std::cout << std::setw(10) << "x" << std::setw(10) << "y1" << std::setw(10) << "y2" << std::setw(12) << "Difference" << std::endl;

    std::cout << std::string(40, '-') << std::endl;

    int printed\_points = 0;

    for (auto it = not\_checked.rbegin(); it != not\_checked.rend() && printed\_points < last\_points\_count; ++it) {

        double x = it->first;

        double y1\_value = std::round(it->second \* 1000.0) / 1000.0;

        double y2\_value = checked.count(x) > 0 ? std::round(checked.at(x) \* 1000.0) / 1000.0 : 0;

        if (checked.count(x) > 0) {

            double difference = std::round(std::abs(y1\_value - y2\_value) \* 1000.0) / 1000.0;

            std::cout << std::setw(10) << std::fixed << std::setprecision(3) << x

                      << std::setw(10) << y1\_value

                      << std::setw(10) << y2\_value

                      << std::setw(12) << difference << std::endl;

        } else {

            std::cout << std::setw(10) << std::fixed << std::setprecision(3) << x

                      << std::setw(10) << y1\_value

                      << std::setw(10) << ""

                      << std::setw(12) << "" << std::endl;

        }

        ++printed\_points;

    }

    std::cout << "\nКоличество точек на последней итерации равно: " << count << std::endl;

    return 0;

}

**3.Метод Адамса 3 порядка**

#include <iostream>

#include <cmath>

#include <map>

#include <iomanip>

// Функция f(x, y)

double f(double x, double y) {

    return 1 + (1 - x) \* sin(y);

}

// Метод Рунге-Кутта 4-го порядка

double runge\_kutta(double x, double y, double y0, double h) {

    double k1 = h \* f(x, y);

    double k2 = h \* f(x + h / 2, y + k1 / 2);

    double k3 = h \* f(x + h / 2, y + k2 / 2);

    double k4 = h \* f(x + h, y + k3);

    return y0 + (h / 6) \* (k1 + 2 \* k2 + 2 \* k3 + k4);

}

// Метод Адамса 3-го порядка

std::map<double, double> third\_order\_adams(double a, double b, int n, double y0, double z0) {

    double h = (b - a) / n; // Шаг интегрирования

    std::map<double, double> Y, Z;

    Y[a] = y0;

    Z[a] = z0;

    // Начальные условия с помощью метода Рунге-Кутта

    Z[a + h] = runge\_kutta(a, Y[a], Z[a], h);

    Y[a + h] = Y[a] + h \* Z[a];

    Z[a + 2 \* h] = runge\_kutta(a + h, Y[a + h], Z[a + h], h);

    Y[a + 2 \* h] = Y[a + h] + h \* Z[a + h];

    // Метод Адамса-Башфорта для последующих точек

    for (int i = 3; i <= n; ++i) {

        double x\_i = a + i \* h;

        double x\_i1 = a + (i - 1) \* h;

        double x\_i2 = a + (i - 2) \* h;

        double x\_i3 = a + (i - 3) \* h;

        Z[x\_i] = Z[x\_i1] + h \* (23 \* f(x\_i1, Y[x\_i1]) - 16 \* f(x\_i2, Y[x\_i2]) + 5 \* f(x\_i3, Y[x\_i3])) / 12;

        Y[x\_i] = Y[x\_i1] + h \* (23 \* Z[x\_i1] - 16 \* Z[x\_i2] + 5 \* Z[x\_i3]) / 12;

    }

    return Y;

}

// Проверка точности

bool check\_accuracy(const std::map<double, double>& y1, const std::map<double, double>& y2, double eps) {

    for (const auto& [x, y] : y1) {

        if (y2.find(x) != y2.end() && std::abs(y - y2.at(x)) >= eps) {

            return false;

        }

    }

    return true;

}

int main() {

    double a = 0;

    double b = 0.25;

    double y0 = 0; // Начальное условие y(0) = 0

    double z0 = 1; // Начальное условие y'(0) = z = 1

    double eps = 0.001; // Точность

    std::map<double, double> not\_checked;

    std::map<double, double> checked;

    int n = 16;

    std::cout << "Интегрирование методом Адамса третьего порядка:\n";

    not\_checked = third\_order\_adams(a, b, n, y0, z0);

    while (true) {

        checked = not\_checked;

        n \*= 2;

        not\_checked = third\_order\_adams(a, b, n, y0, z0);

        if (check\_accuracy(checked, not\_checked, eps)) {

            break;

        }

    }

    int count = not\_checked.size();

    // Подготовка данных для вывода в консоль

    std::cout << std::setw(10) << "x" << std::setw(10) << "y1" << std::setw(10) << "y2" << std::setw(10) << "разность" << "\n";

    std::cout << std::string(40, '-') << "\n";

    auto it\_checked = checked.rbegin();

    for (auto it\_not\_checked = not\_checked.rbegin(); it\_not\_checked != not\_checked.rend(); ++it\_not\_checked) {

        double x = it\_not\_checked->first;

        double y1\_value = std::round(it\_not\_checked->second \* 1000.0) / 1000.0;

        double y2\_value = (it\_checked != checked.rend() && it\_checked->first == x) ? std::round(it\_checked->second \* 1000.0) / 1000.0 : 0;

        if (it\_checked != checked.rend() && it\_checked->first == x) {

            double difference = std::round(std::abs(y1\_value - y2\_value) \* 1000.0) / 1000.0;

            std::cout << std::setw(10) << x << std::setw(10) << y1\_value << std::setw(10) << y2\_value << std::setw(10) << difference << "\n";

            ++it\_checked;

        } else {

            std::cout << std::setw(10) << x << std::setw(10) << y1\_value << std::setw(10) << "" << std::setw(10) << "" << "\n";

        }

    }

    std::cout << "\nКоличество точек на последней итерации равно: " << count << "\n";

    return 0;

}

**4. Метод Адамса 4 порядка**

#include <iostream>

#include <cmath>

#include <map>

#include <iomanip>

// Функция f(x, y)

double f(double x, double y) {

    return 1 + (1 - x) \* sin(y);

}

// Метод Рунге-Кутта 4-го порядка

double runge\_kutta(double x, double y, double y0, double h) {

    double k1 = h \* f(x, y);

    double k2 = h \* f(x + h / 2, y + k1 / 2);

    double k3 = h \* f(x + h / 2, y + k2 / 2);

    double k4 = h \* f(x + h, y + k3);

    return y0 + (h / 6) \* (k1 + 2 \* k2 + 2 \* k3 + k4);

}

// Метод Адамса-Бэшфорта 4-го порядка

std::map<double, double> forth\_order\_adams(double a, double b, int n, double y0, double z0) {

    double h = (b - a) / n;

    std::map<double, double> Y, Z;

    Y[a] = y0;

    Z[a] = z0;

    // Начальные условия с помощью метода Рунге-Кутта

    Z[a + h] = runge\_kutta(a, Y[a], Z[a], h);

    Y[a + h] = Y[a] + h \* Z[a];

    Z[a + 2 \* h] = runge\_kutta(a + h, Y[a + h], Z[a + h], h);

    Y[a + 2 \* h] = Y[a + h] + h \* Z[a + h];

    Z[a + 3 \* h] = runge\_kutta(a + 2 \* h, Y[a + 2 \* h], Z[a + 2 \* h], h);

    Y[a + 3 \* h] = Y[a + 2 \* h] + h \* Z[a + 2 \* h];

    // Метод Адамса-Башфорта для последующих точек

    for (int i = 4; i <= n; ++i) {

        double x = a + i \* h;

        Z[x] = Z[x - h] + h \* (55 \* f(x - h, Y[x - h]) -

                               59 \* f(x - 2 \* h, Y[x - 2 \* h]) +

                               37 \* f(x - 3 \* h, Y[x - 3 \* h]) -

                               9 \* f(x - 4 \* h, Y[x - 4 \* h])) / 24.0;

        Y[x] = Y[x - h] + h \* (55 \* Z[x - h] -

                               59 \* Z[x - 2 \* h] +

                               37 \* Z[x - 3 \* h] -

                               9 \* Z[x - 4 \* h]) / 24.0;

    }

    return Y;

}

// Проверка точности

bool check\_accuracy(const std::map<double, double>& y1, const std::map<double, double>& y2, double eps) {

    for (const auto& [x, val1] : y1) {

        auto it = y2.find(x);

        if (it != y2.end() && std::abs(val1 - it->second) >= eps) {

            return false;

        }

    }

    return true;

}

int main() {

    double a = 0;

    double b = 0.25;

    double y0 = 0;  // Начальное условие y(0) = 0

    double z0 = 1;  // Начальное условие y'(0) = z = 1

    double eps = 0.001;  // Точность

    std::map<double, double> not\_checked, checked;

    int n = 16;

    std::cout << "Интегрирование методом Адамса четвертого порядка:\n";

    not\_checked = forth\_order\_adams(a, b, n, y0, z0);

    while (true) {

        checked = not\_checked;

        n \*= 2;

        not\_checked = forth\_order\_adams(a, b, n, y0, z0);

        if (check\_accuracy(checked, not\_checked, eps)) {

            break;

        }

    }

    int count = not\_checked.size();

    // Подготовка данных для вывода в консоль

    std::cout << std::setw(10) << "x" << std::setw(10) << "y1" << std::setw(10) << "y2" << std::setw(10) << "разность" << "\n";

    std::cout << std::string(40, '-') << "\n";

    for (auto it = not\_checked.rbegin(); it != not\_checked.rend() && count-- > 16; ++it) {

        double x = it->first;

        double y1\_value = round(it->second \* 10000) / 10000.0;

        double y2\_value = checked.count(x) > 0 ? round(checked.at(x) \* 10000) / 10000.0 : 0.0;

        double difference = std::abs(y1\_value - y2\_value);

        std::cout << std::setw(10) << std::fixed << std::setprecision(4) << x

                  << std::setw(10) << y1\_value

                  << std::setw(10) << y2\_value

                  << std::setw(10) << difference << "\n";

    }

    std::cout << "\nКоличество точек на последней итерации равно: " << not\_checked.size() << "\n";

    return 0;

}