ОТЧЁТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

Решение обыкновенных дифференциальных уравнений (Вариант 8)

Выполнил студент 3 курса ПМиИ
Ковшов Максим

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 y_iискомой функции $y = f(x)$ с заданной точностью для некоторых значений аргумента $x_i \in [a, b]$.

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 имеет вид.

$k_1 = hf(x_k, y_k)$,

$k_2 = hf(x_k + h/2, y_k + k_1/2)$,

$k_3 = hf(x_k + h/2, y_k + k_2/2)$,

$k_4 = hf(x_k + h, y_k + k_3)$,

$\Delta y_k=1/6(k_1 + 2k_2 + 2k_3 + k_4)$, $y_{k+1}=y_k + \Delta y_k$, $x_{k+1}=x_k + h$.

Методы Адамса третьего и четвертого порядков имеют вид

$y_{i+1} = y_i + h(23y'_i - 16y'_{i-1} + 5y'_{i-2})/12$;

$y_{i+1} = y_i + h(55y'_i - 59y'_{i-1} + 37y'_{i-2} - 9y'_{i-3})/24$.

Погрешность решения, найденного этими методами, оценивается величиной $O(h^m)$, 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.

Решить <u>уравнение 1</u> методом Эйлера 2-го порядка и методом Рунге-Кутта 4-го порядка. Решить <u>уравнение 2</u> методами Адамса 3-го порядка и 4-го порядка. Погрешность 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 модуль максимальной разности значений функции в совпадающих узлах становится меньше заранее заданной малой величины. Результаты вывести в виде таблиц для последней итерации, в которых первая колонка значения X_k , вторая колонка – значения найденных Y_k.

№ вар.	<u>Уравнение 1</u>	<u>Уравнение 2</u>
8	$y' = 1 + 2{,}2 sinx + 1{,}5y^2$	$y' = \cos(1{,}5x+y)+(x-y)$

$[a;b] = [0;0.5]$, $\varepsilon = 0.001$

$[a; b] = [0; 0.5]$, $\varepsilon = 0.001$

$$\begin{cases} y` = 1 + 2{,}2\sin x + 1{,}5y^2 \\ y(0) = 0 \end{cases}$$

Метод Эйлера 2-ого порядка (1 уравнение):

Рекуррентная формула:

$$y_{\widetilde{i+1}} = y_i + hy_i^{|}$$

$$y_{i+1}^{|} = f(x_{i+1}, y_{\widetilde{i+1}})$$

$$y_{i+1} = y_i + h\frac{y_i^{|} + y_{i+1}^{\sim|}}{2}$$

```
Метод Эйлера
x0 = 0.0000 y0 = 0.0000
x1 = 0.0250 y1 = 0.0257
x2 = 0.0500 y2 = 0.0528
x3 = 0.0750 y3 = 0.0814
x4 = 0.1000 y4 = 0.1116
x5 = 0.1250 y5 = 0.1433
x6 = 0.1500 y6 = 0.1768
x7 = 0.1750 y7 = 0.2122
x8 = 0.2000 y8 = 0.2494
x9 = 0.2250 y9 = 0.2887
x10 = 0.2500 y10 = 0.3302
x11 = 0.2750 y11 = 0.3741
x12 = 0.3000 y12 = 0.4206
x13 = 0.3250 y13 = 0.4699
x14 = 0.3500 y14 = 0.5224
x15 = 0.3750 y15 = 0.5782
x16 = 0.4000 y16 = 0.6378
x17 = 0.4250 y17 = 0.7016
x18 = 0.4500 y18 = 0.7701
x19 = 0.4750 y19 = 0.8440
x20 = 0.5000 y20 = 0.9240
```

Метод Рунге-Кутта 4-го порядка(1 уравнение):

$$k_1 = hf(x_k, y_k)$$

$$k_2 = hf(x_k + h/2, y_k + k_1/2),$$

$$k_3 = hf(x_k + h/2, y_k + k_2/2),$$

$$k_4 = hf(x_k + h, y_k + k_3),$$

$$\Delta y_k = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$y_{k+1} = y_k + \Delta y_k, \quad x_{k+1} = x_k + h$$

```
Метод Рунге-Кутты
x0 = 0.0000 y0 = 0.0000
x1 = 0.0250 y1 = 0.0257
x2 = 0.0500 y2 = 0.0528
x3 = 0.0750 y3 = 0.0814
x4 = 0.1000 y4 = 0.1116
x5 = 0.1250 y5 = 0.1434
x6 = 0.1500 y6 = 0.1769
x7 = 0.1750 y7 = 0.2122
x8 = 0.2000 y8 = 0.2494
x9 = 0.2250 y9 = 0.2888
x10 = 0.2500 y10 = 0.3303
x11 = 0.2750 y11 = 0.3742
x12 = 0.3000 y12 = 0.4207
x13 = 0.3250 y13 = 0.4701
x14 = 0.3500 y14 = 0.5225
x15 = 0.3750 y15 = 0.5784
x16 = 0.4000 y16 = 0.6380
x17 = 0.4250 y17 = 0.7019
x18 = 0.4500 y18 = 0.7705
x19 = 0.4750 y19 = 0.8445
x20 = 0.5000 y20 = 0.9246
```

Метод Адамса 3-ого и 4-ого порядка (2уравнение):

Методы Адамса третьего и четвертого порядков имеют вид

$$y_{i+1} = y_i + h\,(23y'_i - 16y'_{i-1} + 5y'_{i-2})/12;$$
$$y_{i+1} = y_i + h\,(55y'_i - 59y'_{i-1} + 37y'_{i-2} - 9y'_{i-3})/24.$$

Известно:

$$\begin{cases} y'' = \cos(1{,}5x + y) + (x - y) \\ y(0) = 0 \\ y'(0) = 1 \end{cases}$$

Так как эти методы применимы только для ОДУ первого порядка, сделаем замену:

$y`(x, y) = g(x, y)$, тогда $y``(x, y) = g`(x, y)$ и можем записать

$$\begin{cases} g(x,y) = \cos(1,5x + y) + (x - y) \\ y`(x,y) = g(x,y) \\ g(0) = 1 \\ y(0) = 0 \end{cases}$$

Для метода Адамса 3-го порядка найдем первые три значения: g_0 – известно, g_1 , g_2 найдем методом Рунге-Кутты, а y_1 y_2 найдем методом Эйлера:

$$y_i = y_{i-1} + hg_{i-1}$$

$$g_i = g_{i-1} + h(\cos(1{,}5x_{i-1} + y_{i-1}) + (x_{i-1} - y_{i-1})$$

Таким образом, получим первые три приближения, а затем можно будет применять метод Адамса третьего порядка:

$$g_i = g_{i-1} + \frac{h(23f(x_{i-1}, y_{i-1}) - 16f(x_{i-2}, y_{i-2}) + 5f(x_{i-3}, y_{i-3})}{12}$$

$$y_i = y_{i-1} + \frac{h(23g_2 - 16g_1 + 5g_0)}{12}$$

Аналогично для метода Адамса 4-ого порядка, только находим не 3, а 4 начальных приблежения.

```
Метод Адамса 3-его порядка
x0 = 0.0000 y0 = 0.0000
x1 = 0.0375 y1 = 0.0725
x2 = 0.0750 y2 = 0.1376
x3 = 0.1125 y3 = 0.1970
x4 = 0.1500 y4 = 0.2503
x5 = 0.1875 y5 = 0.2976
x6 = 0.2250 y6 = 0.3389
x7 = 0.2625 y7 = 0.3745
x8 = 0.3000 y8 = 0.4047
x9 = 0.3375 y9 = 0.4299
x10 = 0.3750 y10 = 0.4507
x11 = 0.4125 y11 = 0.4675
x12 = 0.4500 y12 = 0.4807
x13 = 0.4875 y13 = 0.4907
x14 = 0.5250 y14 = 0.4981
```

```
Метод Адамса 4-ого порядка
x0 = 0.0000 y0 = 0.0000
x1 = 0.0375 y1 = 0.0371
x2 = 0.0750 y2 = 0.0771
x3 = 0.1125 y3 = 0.1158
x4 = 0.1500 y4 = 0.1534
x5 = 0.1875 y5 = 0.1899
x6 = 0.2250 y6 = 0.2249
x7 = 0.2625 y7 = 0.2583
x8 = 0.3000 y8 = 0.2898
x9 = 0.3375 y9 = 0.3193
x10 = 0.3750 y10 = 0.3468
x11 = 0.4125 y11 = 0.3721
x12 = 0.4500 y12 = 0.3953
x13 = 0.4875 y13 = 0.4164
x14 = 0.5250 y14 = 0.4353
```

ПРИЛОЖЕНИЕ

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <tuple>
#include <iomanip>

#define a 0
#define b 0.5
#define eps 0.001

double f1(double x, double y) {
    return 1 + 2.2 * sin(x) + 1.5 * pow(y, 2);
}

double f2(double x, double y) {
    return cos(1.5 * x + y) + (x - y);
}

double euler_second_oeder(double y_initial, double x, double h) {
    double y_half = y_initial + 0.5 * h * f1(x, y_initial);
    double y_new = y_initial + h * f1(x + 0.5 * h, y_half);
    return y_new;
}

std::vector<std::tuple<double, double>> euler_method(double y_initial, double h) {
    std::vector<std::tuple<double, double>> points;
    double x = a;
    double y = y_initial;
    points.emplace_back(x, y);
    while (x < b - eps) {
        double y_prev = y;
        y = euler_second_oeder(y, x, h);
        x += h;
        points.emplace_back(x, y);
        if (abs(y - y_prev) < eps)
            break;
    }
    return points;
}

double runge_kutta(double y_initial, double x, double h) {
    double k1 = h * f1(x, y_initial);
    double k2 = h * f1(x + 0.5 * h, y_initial + 0.5 * k1);
    double k3 = h * f1(x + 0.5 * h, y_initial + 0.5 * k2);
    double k4 = h * f1(x + h, y_initial + k3);
    double y_new = y_initial + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
    return y_new;
}

std::vector<std::tuple<double, double>> runge_kutta_method(double y_initial, double h) {
    std::vector<std::tuple<double, double>> points;
    double x = a;
    double y = y_initial;
    points.emplace_back(x, y);
    while (x < b - eps) {
        double y_prev = y;
        y = runge_kutta(y, x, h);
        x += h;
        points.emplace_back(x, y);
        if (abs(y - y_prev) < eps)
            break;
    }
    return points;
}
```

```cpp
double adams_third_step(double y, double y_prev, double y_prev2, double h, double x) {
    return y + h * (23.0 / 12.0 * f2(x, y) - 4.0 / 3.0 * y_prev + 5.0 / 12.0 * y_prev2);
}

std::vector<std::tuple<double, double>> adams_third_order(double y_initial, double y_prime_initial, double h) {
    std::vector<std::tuple<double, double>> points;
    double x = a;
    double y = y_initial;
    double y_prev = y + h * y_prime_initial;
    points.emplace_back(x, y);
    while (x < b - eps) {
        double y_prev2 = y_prev;
        y_prev = y;
        y = adams_third_step(y, y_prev, y_prev2, h, x);
        x += h;
        points.emplace_back(x, y);
        if (abs(y - y_prev) < eps)
            break;
    }
    return points;
}

double adams_fourth_step(double y, double y_prev, double y_prev2, double y_prev3, double h, double x) {
    return y + h * (55.0 / 24.0 * f2(x, y) - 59.0 / 24.0 * f2(x - h, y_prev) +
        37.0 / 24.0 * f2(x - 2 * h, y_prev2) - 9.0 / 24.0 * f2(x - 3 * h, y_prev3));
}

```

```cpp
std::vector<std::tuple<double, double>> adams_fourth_order(double y_initial, double y_prime_initial, double h) {
    std::vector<std::tuple<double, double>> points;
    double x = a;
    double y = y_initial;
    double y_prev = y + h * y_prime_initial;
    double y_prev2 = y_prev + h * f2(x,y);
    double y_prev3 = y_prev2 + h * f2(x+h,y_prev);
    points.emplace_back(x, y);
    while (x < b - eps) {
        y_prev3 = y_prev2;
        y_prev2 = y_prev;
        y_prev = y;
        y = adams_fourth_step(y, y_prev, y_prev2, y_prev3, h, x);
        x += h;
        points.emplace_back(x, y);
        if (abs(y - y_prev) < eps)
            break;
    }
    return points;
}

```

```cpp
int main()
{
    setlocale(LC_ALL, "Russian");
    std::cout << std::setprecision(4);
    std::cout << std::fixed;

    double y_initial = 0;
    double h = 0.0375;
    double y_prime_initial = 1;
    //В данных векторах хранятся итоговые значения вычислений
    std::vector<std::tuple<double, double>> euler =
        euler_method(y_initial, h);
    std::vector<std::tuple<double, double>> runge_kutta =
        runge_kutta_method(y_initial, h);
    std::vector<std::tuple<double, double>> adams_third =
        adams_third_order(y_initial, y_prime_initial, h);
    std::vector<std::tuple<double, double>> adams_fourth =
        adams_fourth_order(y_initial, y_prime_initial, h);

    std::cout << "Метод Эйлера" << std::endl;
    for (int i = 0; i < euler.size(); i++)
        std::cout << "x" << i << " = " << std::get<0>(euler[i]) << " "
            << "y" << i << " = " << std::get<1>(euler[i]) << std::endl;

    std::cout << "Метод Рунге-Кутты" << std::endl;
    for (int i = 0; i < runge_kutta.size(); i++)
        std::cout << "x" << i << " = " << std::get<0>(runge_kutta[i]) << " "
            << "y" << i << " = " << std::get<1>(runge_kutta[i]) << std::endl;

    std::cout << "Метод Адамса 3-его порядка" << std::endl;
    for (int i = 0; i < adams_third.size(); i++)
        std::cout << "x" << i << " = " << std::get<0>(adams_third[i]) << " "
            << "y" << i << " = " << std::get<1>(adams_third[i]) << std::endl;

    std::cout << "Метод Адамса 4-ого порядка" << std::endl;
    for (int i = 0; i < adams_fourth.size(); i++)
        std::cout << "x" << i << " = " << std::get<0>(adams_fourth[i]) << " "
            << "y" << i << " = " << std::get<1>(adams_fourth[i]) << std::endl;
}
```