МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
(НИЯУ МИФИ)
Институт интеллектуальных кибернетических систем
Кафедра Кибернетики

Лабораторная работа №2

«Решение системы дифференциальных уравнений неявным методом Рунге-Кутта четвёртого порядка»

Выполнил студент группы Б17-511: Чудновец И.В.

Проверил: Саманчук В.Н.

Москва, 2020

Задание

Написать программу для решения системы дифференциальных уравнений неявным методом Рунге-Кутта четвёртого порядка.

$$\begin{cases} \frac{dy_1}{dx} = -1000 * y_1 + 999 * y_2 \\ \frac{dy_2}{dx} = y_1 - 2 * y_2 \end{cases}$$

Начальные условия:

$$\begin{cases} y_{10} = 10 \\ y_{20} = 20 \end{cases}$$

Описание метода

Метод Рунге — Кутты четвёртого порядка при вычислениях с постоянным шагом интегрирования столь широко распространён, что его часто называют просто методом Рунге — Кутты. Рассмотрим задачу Коши для системы обыкновенных дифференциальных уравнений первого порядка.

(Далее $\mathbf{y}, \mathbf{f}, \mathbf{k}_i \in \mathbb{R}^n$, а $x, h \in \mathbb{R}^1$).

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}), \quad \mathbf{y}(x_0) = \mathbf{y}_0.$$

Тогда приближенное значение в последующих точках вычисляется по итерационной формуле:

$$\overline{\mathbf{y}}_{n+1} = \mathbf{y}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

Вычисление нового значения проходит в четыре стадии:

$$\mathbf{k}_1 = \mathbf{f}(x_n, \mathbf{y}_n),$$
$$\mathbf{k}_2 = \mathbf{f}\left(x_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_1\right),$$
$$\mathbf{k}_3 = \mathbf{f}\left(x_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_2\right),$$
$$\mathbf{k}_4 = \mathbf{f}(x_n + h, \mathbf{y}_n + h\,\mathbf{k}_3).$$

где h — величина шага сетки по x.

Этот метод имеет четвёртый порядок точности. Это значит, что ошибка на одном шаге имеет порядок $O(h^5)$, а суммарная ошибка на конечном интервале интегрирования имеет порядок $O(h^4)$.

Неявный метод Рунге — Кутты:

Неявный метод заключается в реализации схемы предиктор-корректор, где предиктор – метод РК4 (указан выше), а корректор – неявный метод РК4. Он задаётся формулой:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{6}(\overset{n+1}{\mathbf{k}_1} + 2\overset{n+1}{\mathbf{k}_2} + 2\overset{n+1}{\mathbf{k}_3} + \overset{n+1}{\mathbf{k}_4})$$

Вычисление нового значения проходит в четыре стадии:

$$\overset{n+1}{\mathbf{k}_1} = \mathbf{f}\left(x_{n+1}, \overline{\mathbf{y}}_{n+1}\right),$$
$$\overset{n+1}{\mathbf{k}_2} = \mathbf{f}\left(x_{n+1} - \frac{h}{2}, \overline{\mathbf{y}}_{n+1} - \frac{h}{2}\overset{n+1}{\mathbf{k}_1}\right),$$
$$\overset{n+1}{\mathbf{k}_3} = \mathbf{f}\left(x_{n+1} - \frac{h}{2}, \overline{\mathbf{y}}_{n+1} - \frac{h}{2}\overset{n+1}{\mathbf{k}_2}\right),$$
$$\overset{n+1}{\mathbf{k}_4} = \mathbf{f}\left(x_n, \overline{\mathbf{y}}_{n+1} - h\,\overset{n+1}{\mathbf{k}_3}\right).$$

где h — величина шага сетки по x.

На вход программе процедуре поступают параметры: e – точность корректора, h - шаг, N – число шагов, x – начальное значение x0, y – массив начальных значений y0, foo_m – массив функций правой части системы.

Листинг программы

```
#include <iostream>
#include <cmath>

using namespace std;

typedef double (*f)(double, double, double);

void predictor(double h, double x, double* y, f array_f[2]){
    double y_[2];
    y_[0] = y[0];
    y_[1] = y[1];

    double y_dy[2];

    double** c_k;
    c_k = new double*[5];
    for(int i = 0; i < 5; i++){
        c_k[i] = new double[2];
    }
    c_k[0][0] = c_k[0][1] = 0;

    double dx;
    double P;

    for(int k = 0; k < 4; k++){
        switch(k){
        case 0:
            dx = 0;
            P = h / 6;
            break;
        case 1:
        case 2:
            dx = h / 2;
            P = h / 3;
            break;
        case 3:
            dx = h;
            P = h / 6;
            break;
        }
        y_dy[0] = y_[0] + dx * c_k[k][0];
```

```
            y_dy[1] = y_[1] + dx * c_k[k][1];

            c_k[k + 1][0] = array_f[0](x + dx, y_dy[0], y_dy[1]);
            c_k[k + 1][1] = array_f[1](x + dx, y_dy[0], y_dy[1]);

            y[0] += P * c_k[k + 1][0];
            y[1] += P * c_k[k + 1][1];
        }
    }

    void corrector(double h, double x, double* y, double* y_1, double* y_2, f
    array_f[2]) {
        y_2[0] = y[0];
        y_2[1] = y[1];

        double y_dy[2];

        double** c_k;
        c_k = new double*[5];
        for(int i = 0; i < 5; i++){
            c_k[i] = new double[2];
        }
        c_k[0][0] = c_k[0][1] = 0;

        double dx;
        double P;

        for(int k = 0; k < 4; k++){
            switch(k){
            case 0:
                dx = 0;
                P = h / 6;
                break;
            case 1:
            case 2:
                dx = -h / 2;
                P = h / 3;
                break;
            case 3:
                dx = -h;
                P = h / 6;
                break;
            }
            y_dy[0] = y_1[0] + dx * c_k[k][0];
            y_dy[1] = y_1[1] + dx * c_k[k][1];

            c_k[k + 1][0] = array_f[0](x + dx, y_dy[0], y_dy[1]);
            c_k[k + 1][1] = array_f[1](x + dx, y_dy[0], y_dy[1]);

            y_2[0] += P * c_k[k + 1][0];
            y_2[1] += P * c_k[k + 1][1];
        }
    }

    bool is_precision(double e, double* y_1, double* y_2){
        return (abs(y_1[0] - y_2[0]) < e) && (abs(y_1[1] - y_2[1]) < e);
    }

    void Runge_Kutta_implicit(double e, double h, int N, double x, double* y, f
    array_f[2]){
        int i = 0;
        cout << i << " " << x << " " << y[0] << " " << y[1] << endl;

        double y_1[2], y_2[2];
        y_1[0] = y[0];
        y_1[1] = y[1];

        for(i = 1; i <= N; i++){
            predictor(h, x, y_1, array_f);
```

```
        x += h;

        corrector(h, x, y, y_1, y_2, array_f);
        while(!is_precision(e, y_1, y_2)){
            y_1[0] = y_2[0];
            y_1[1] = y_2[1];
            corrector(h, x, y, y_1, y_2, array_f);
        }

        y[0] = y_2[0];
        y[1] = y_2[1];
        y_1[0] = y_2[0];
        y_1[1] = y_2[1];
        cout << i << " " << x << " " << y[0] << " " << y[1] << endl;
    }
}

double f1(double x, double y_1, double y_2){
    return -1000 * y_1 + 999 * y_2;
}

double f2(double x, double y_1, double y_2){
    return y_1 - 2 * y_2;
}

int main()
{
    double a = 0;
    double b = 0.001;
    double h = 0.00025;

    int N = (b - a) / h;
    double x = a;
    double *y = new double[2];
    y[0] = 10;
    y[1] = 20;
    f foo_m[] = {f1, f2};
    double e = 0.0001;

    Runge_Kutta_implicit(e, h, N, x, y, foo_m);

    return 0;
}
```

Пример работы программы

```
0 0 10 20
1 0.00025 12.2067 19.9928
2 0.0005 13.9237 19.9861
3 0.00075 15.2595 19.9797
4 0.001 16.2985 19.9737
```