«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

Лабораторная работа 5. Численное дифференцирование и задача Коши.

Дисциплина «Вычислительная математика»

Автор: Назирджанов Некруз Фарходович P32211

Факультет: ПИиКТ

ITMO UNIVERSITY

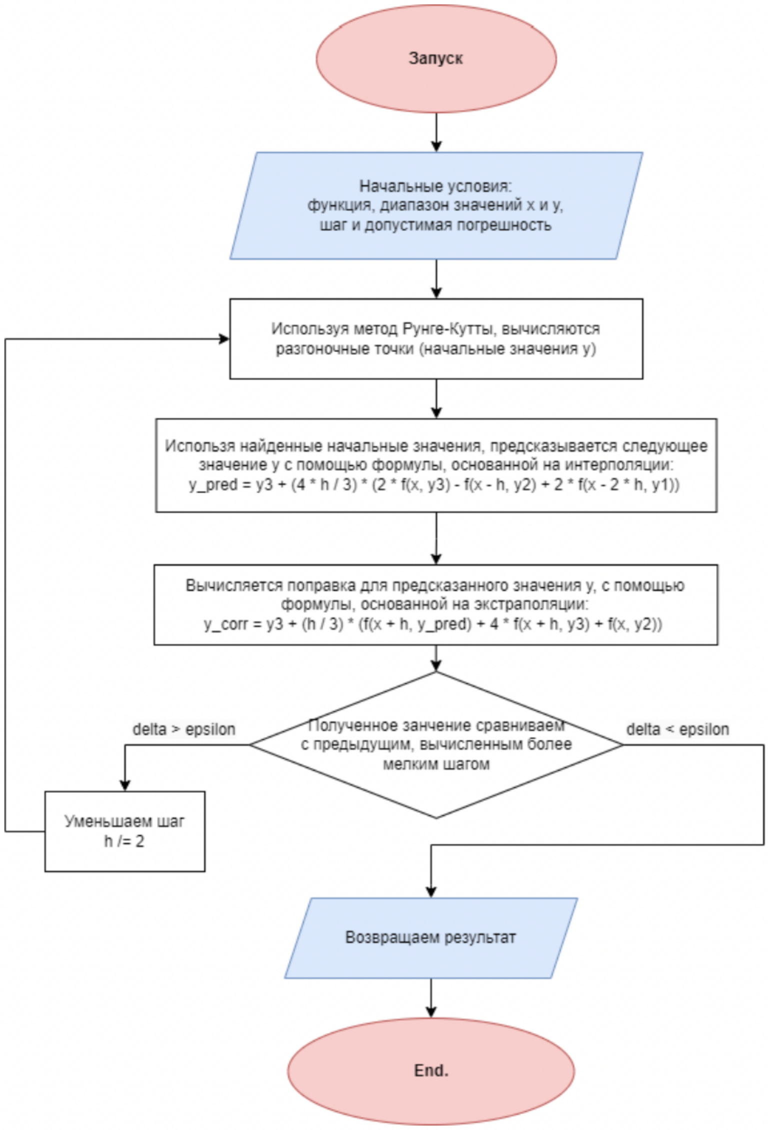
Описание метода Милна

Метод Милна является численным методом решения дифференциальных уравнений. Он использует несколько начальных значений, чтобы предсказать следующее значение функции, а затем корректирует это значение.

Метод Милна подразумевает разбиение интервала, на котором решается уравнение, на несколько под интервалов и нахождение значений функции на каждом из них. Сначала находятся значения функции с помощью метода Рунге-Кутты, а затем эти значения используются для предсказания значения на следующем интервале.

Для этого используется формула, которая позволяет вычислить значение функции на следующем интервале с помощью нескольких значений на текущем и предыдущих интервалах. Затем вычисляется поправка, чтобы учесть погрешности предсказания, и полученное значение используется для корректировки значений функции на текущем и следующих интервалах.

Блок-схема

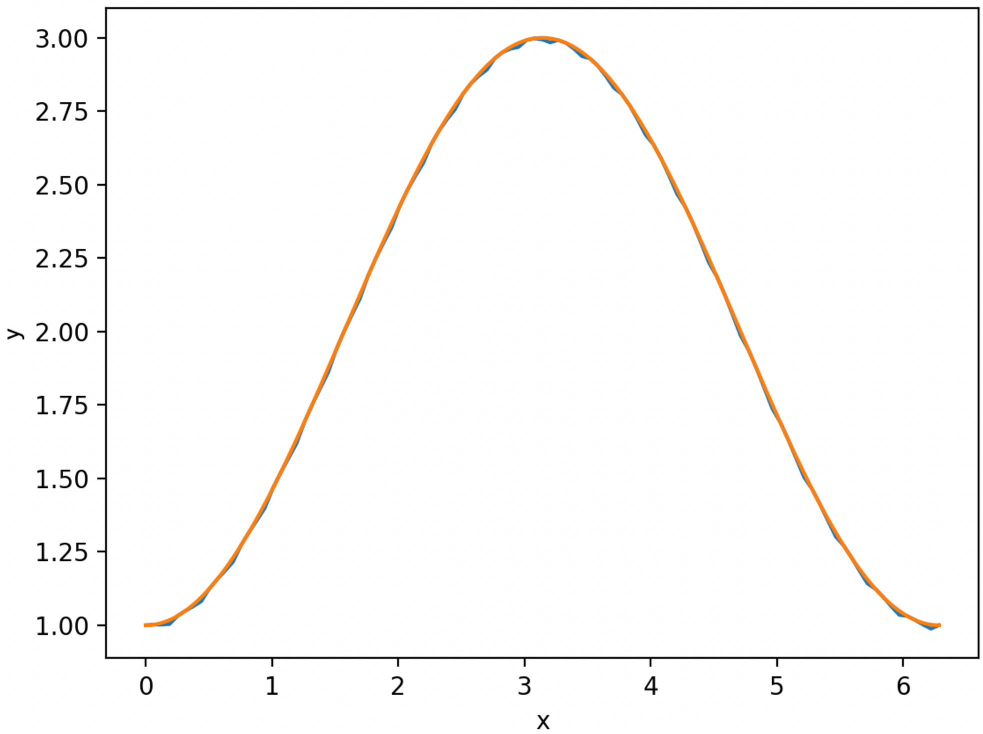

Запуск
Начальные условия:
функция, диапазон значений x и y,
шаг и допустимая погрешность
Используя метод Рунге-Кутты, вычисляются
разгоночные точки (начальные значения y)
Используя найденные начальные значения, предсказывается следующее
значение y с помощью формулы, основанной на интерполяции:
y_pred = y3 + (4 * h / 3) * (2 * f(x, y3) - f(x - h, y2) + 2 * f(x - 2 * h, y1))
Вычисляется поправка для предсказанного значения y, с помощью
формулы, основанной на экстраполяции:
y_corr = y3 + (h / 3) * (f(x + h, y_pred) + 4 * f(x + h, y3) + f(x, y2))
delta > epsilon
Полученное занчение сравниваем
с предыдущим, вычисленным более
мелким шагом
delta < epsilon
Уменьшаем шаг
h /= 2
Возвращаем результат
End.

Реализация

```cpp
#include <utility>
#include "MilneMethod.h"
#include "vector"
#include "iostream"
#include "cmath"
using namespace std;
std::pair<std::vector<double>, std::vector<double>> MilneMethod::solveByMilne(std::function<double(double, double)>
&f, double epsilon, double a, double y_a, double b) {
    double h = (b - a) / 10;
    while (true) {
        std::vector<double> y(4);
        std::vector<double> x_vec;

        double k1, k2, k3, k4;

        y[0] = y_a;
        k1 = f(a, y[0]);
        k2 = f(a + h / 2, y[0] + h / 2 * k1);
        k3 = f(a + h / 2, y[0] + h / 2 * k2);
        k4 = f(a + h, y[0] + h * k3);
        y[1] = y[0] + h / 6 * (k1 + 2 * k2 + 2 * k3 + k4);

        k1 = f(a, y[0]);
        k2 = f(a + h / 2, y[1]);
        k3 = f(a + h / 2, y[1]);
        k4 = f(a + h, y[1]);
        y[2] = y[0] + h / 6 * (k1 + 2 * k2 + 2 * k3 + k4);

        k1 = f(a, y[1]);
        k2 = f(a + h / 2, y[2]);
        k3 = f(a + h / 2, y[2]);
        k4 = f(a + h, y[2]);
        y[3] = y[1] + h / 6 * (k1 + 2 * k2 + 2 * k3 + k4);

        x_vec.push_back(a);
        x_vec.push_back(a + h);
        x_vec.push_back(a + 2 * h);
        x_vec.push_back(a + 3 * h);


        int iter = 4;
        double x = a + iter * h;
        bool success = true;
        while (x <= b) {
            x_vec.push_back(x);
            double y_i_1 = y[iter - 4] + 4 * h / 3 * (2 * f(x - 3 * h, y[iter - 3]) -
                                                     f(x - 2 * h, y[iter - 2]) +
                                                     2 * f(x - h, y[iter - 1]));
            y.push_back(y_i_1);
            double y_i_2 =
                    y[iter - 2] + h / 3 * (f(x, y[iter]) + 4 * f(x - h, y[iter - 1]) +
                                           f(x - 2 * h, y[iter - 2]));

            double epsilon_m = std::abs(y_i_1 - y_i_2) / 29;
            if (epsilon_m > epsilon) {
                success = false;
                h /= 2;
                break;
            }
            x += h;
            iter++;
        }
        if (success) {
            return {x_vec, y};
        }
    }
}
```

Примеры и результаты работы программы

```
if you want to solve y` = sin(x) type 1
if you want to solve y` = xy/2 type 2
if you want to solve y` = x + y type 3
if you want to solve y` = 0 type 4
1
enter accuracy
0.01
enter left and right border
0
6.28318
enter initial condition
1
```



Вывод:

Многошаговый Метод Милна обладает высокой точностью и низкой численной ошибкой при решении дифференциальных уравнений. Однако, данный метод может быть менее применим к некоторым видам задач, так как требует начальных значений на нескольких шагах.

Метод Эйлера является одним из самых простых и быстрых методов, но при этом имеет низкую точность и высокую численную ошибку, особенно при большом шаге интегрирования. Усовершенствованный метод Эйлера имеет высокую точность и более низкую ошибку, но также может быть менее применим к некоторым задачам из-за необходимости дополнительных вычислений.

Метод Рунге-Кутты 4-го порядка обеспечивает высокую точность и более низкую ошибку, чем Метод Эйлера, но при этом является более сложным с вычислительной точки зрения.

Многошаговый метод Адамса является более точным, чем Метод Эйлера и Усовершенствованный метод Эйлера, и менее сложным, чем Метод Рунге-Кутты 4-го порядка. Однако, он также может быть менее применим к некоторым задачам, так как требует начальных значений на нескольких шагах.

Таким образом, выбор конкретного метода должен зависеть от конкретной задачи, которую необходимо решить, а также от желаемой точности и вычислительной сложности.

Для каждой конкретной задачи можно выбрать наиболее подходящий метод, учитывая требуемую точность и вычислительную сложность.

Однако, необходимо учитывать, что в реальных задачах могут возникать дополнительные ограничения и условия, такие как необходимость обеспечения устойчивости метода или требование соблюдения определенных ограничений на шаг интегрирования. Поэтому, выбор конкретного метода интегрирования должен производиться с учетом всех этих условий и ограничений.

В целом, проведенный анализ позволил установить, что многошаговые методы, такие как Метод Милна и методы Адамса, могут обеспечить более высокую точность при решении дифференциальных уравнений, чем одношаговые методы, но при этом могут быть менее применимы к некоторым задачам из-за дополнительных требований к начальным условиям. Выбор конкретного метода интегрирования должен быть основан на конкретной задаче и учитывать все условия и ограничения.