Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
Высшего образования
Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №6 по Вычислительной Математике

Численное решение обыкновенных дифференциальных уравнений

Вариант №5

Группа: P3214
Выполнил:
Минкова Алина Андреевна
Проверил:
Малышева Татьяна Алексеевна

Г. Санкт-Петербург

2024

Оглавление

Цель работы

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

Порядок выполнения работы

1. В программе численные методы решения обыкновенных дифференциальных уравнений (ОДУ) должен быть реализован в виде отдельного класса /метода/функции;
2. Пользователь выбирает ОДУ вида (не менее трех уравнений), из тех, которые предлагает программа;
3. Предусмотреть ввод исходных данных с клавиатуры: начальные условия $y_0 = y(x_0)$, интервал дифференцирования $[x_0, x_n]$, шаг h, точность Ɛ;
4. Для исследования использовать одношаговые методы и многошаговые методы;
По варианту: модифицированный метод Эйлера, метод Рунге-Кутта 4-го порядка, метод Милна;
5. Составить таблицу приближенных значений интеграла дифференциального уравнения, удовлетворяющего начальным условиям, для всех методов, реализуемых в программе;
6. Для оценки точности одношаговых методов использовать правило Рунге;
7. Для оценки точности многошаговых методов использовать точное решение задачи:
$\varepsilon = \max_{0<=i<=n}|y_{i\text{точн}} - y_i|$;
8. Построить графики точного решения и полученного приближенного решения (разными цветами);
9.Программа должна быть протестирована при различных наборах данных, в том числе и некорректных;
10. Проанализировать результаты работы программы.

Рабочие формулы

$$y_{i+1} = y_i + \frac{h}{2}\left[f(x_i, y_i) + f\big(x_{i+1}, y_i + hf(x_i, y_i)\big)\right],\ i = 0, 1 \dots \qquad (10)$$

Данные рекуррентные соотношения описывают новую разностную схему, являющуюся ***модифицированным методом Эйлера***, которая называется методом ***Эйлера с пересчетом***. Метод Эйлера с пересчетом имеет ***второй порядок точности*** $\delta_n = O(h^2)$.

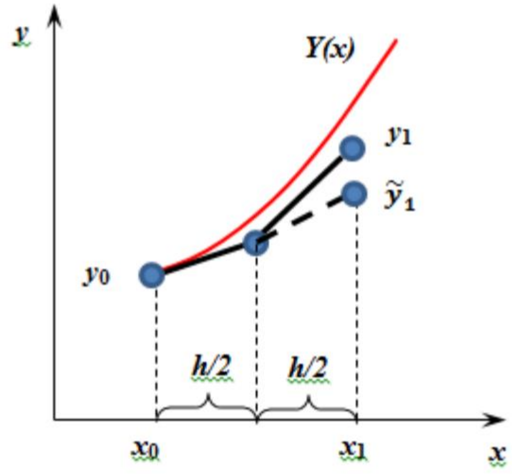


Рис.3 Геометрическая интерпретация метода Эйлера с пересчетом

Широко распространен **метод Рунге-Кутта четвертого порядка**, часто без уточнений называемый просто методом Рунге – Кутты.

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \qquad (13)$$

$k_1 = h \cdot f(x_i, y_i)$

$k_2 = h \cdot f(x_i + \frac{h}{2}, y_i + \frac{k_1}{2})$

$k_3 = h \cdot f(x_i + \frac{h}{2}, y_i + \frac{k_2}{2})$

$k_4 = h \cdot f(x_i + h, y_i + k_3)$

Таким образом, данный метод Рунге-Кутта требует на каждом шаге четырехкратного вычисления правой части $f(x, y)$уравнения (5).

Суммарная погрешность этого метода есть величина $\delta_n = O(h^4)$.

Метод Милна

Метод Милна относится к многошаговым методам и представляет один из методов прогноза и коррекции.

Для получения формул Милна используется первая интерполяционная формула Ньютона с разностями до третьего порядка.

Решение в следующей точке находится в два этапа. На первом этапе осуществляется прогноз значения функции, а затем на втором этапе - коррекция полученного значения. Если полученное значение y после коррекции существенно отличается от спрогнозированного, то проводят еще один этап коррекции. Если опять имеет место существенное отличие от предыдущего значения (т.е. от предыдущей коррекции), то проводят еще одну коррекцию и т.д.

Вычислительные формулы:

а) этап прогноза:

$$y_i^{\text{прогн}} = y_{i-4} + \frac{4h}{3}(2f_{i-3} - f_{i-2} + 2f_{i-1})$$

б) этап коррекции:

$$y_i^{\text{корр}} = y_{i-2} + \frac{h}{3}(f_{i-2} + 4f_{i-1} + f_i^{\text{прогн}})$$

$$f_i^{\text{прогн}} = f(x_i, y_i^{\text{прогн}})$$

Для начала счета требуется задать решения в трех первых точках, которые можно получить одношаговыми методами (например, методом Рунге-Кутта).

Метод требует несколько меньшего количества вычислений (достаточно только два раза вычислить $f(x, y)$, остальные берутся с предыдущих этапов).

Суммарная погрешность этого метода есть величина $\delta_n = O(h^4)$.

ОЦЕНКА ПОГРЕШНОСТИ ЧИСЛЕННОГО РЕШЕНИЯ

Оценить погрешность приближенных решений можно двумя способами двумя способами:

1. $\varepsilon = \max_{0 \le i \le n} |y_{i\text{точн}} - y_i|$,

где $y_{i\text{точн}}, y_i$ - значения точного и приближенного решений в узлах сетки x_i, $i = 1, \ldots, n$

2. по правилу Рунге:

$R = \frac{y^h - y^{h/2}}{2^p - 1}$,

где y^h - решение задачи Коши с шагом h в точке $x + h$

$y^{h/2}$ - решение задачи Коши с шагом $h/2$ в точке $x + h$

p – порядок точности метода.

При использовании правила Рунге **контроль точности можно осуществлять:**

1. **На каждом шаге *h*.**

Для этого вычисляем значение y_1 сначала с шагом *h*, затем с шагом *h*/2.

Если $\frac{|y_1^h - y_1^{h/2}|}{2^p - 1} \le \varepsilon$, тогда переходим к следующему узлу сетки $x_2 = x_1 + h$.

Если правило Рунге не работает, уменьшаем шаг и производим вычисления y_1 с шагом *h*/4:

$\frac{|y_1^{h/2} - y_1^{h/4}|}{2^p - 1} \le \varepsilon$ и т.д.

2. **На конце заданного интервала**.

Для этого численно решаем задачу с заданным шагом *h* на всем заданном интервале, затем решаем задачу с шагом *h*/2 на всем заданном интервале.

Сравниваем $\frac{|y_n^h - y_n^{h/2}|}{2^p - 1} \le \varepsilon$. Если точность не достигнута, шаг уменьшаем.

Листинг программы

```
def euler(f, x0, y0, xn, h):
    x = np.arange(x0, xn + h, h)
    y = np.zeros(len(x))
    y[0] = y0
    for i in range(len(x) - 1):
        y[i + 1] = y[i] + h /2 * (f(x[i], y[i]) + f(x[i + 1], y[i] + h *
f(x[i], y[i])))
    return x, y, h

def rk4(f, x0, y0, xn, h):
    x = np.arange(x0, xn + h, h)
    y = np.zeros(len(x))
    y[0] = y0
    for i in range(len(x) - 1):
        k1 = h * f(x[i], y[i])
        k2 = h * f(x[i] + h / 2, y[i] + k1 / 2)
        k3 = h * f(x[i] + h / 2, y[i] + k2 / 2)
        k4 = h * f(x[i] + h, y[i] + k3)
        y[i + 1] = y[i] + (k1 + 2 * k2 + 2 * k3 + k4) / 6
    return x, y, h

def milne(f, x0, y0, xn, h, eps):
    x = np.arange(x0, xn + h, h)
    y = np.zeros(len(x))
    y[0] = y0
    y[1] = y_approx_rk4[np.argmin(np.abs(x_approx_rk4 - (x_approx_rk4[0] +
h)))]
    y[2] = y_approx_rk4[np.argmin(np.abs(x_approx_rk4 - (x_approx_rk4[0] + 2
* h)))]
    y[3] = y_approx_rk4[np.argmin(np.abs(x_approx_rk4 - (x_approx_rk4[0] + 3
* h)))]
    for i in range(3, len(x) - 1):
        y_pred = y[i - 3] + 4 * h * (2 * f(x[i - 2], y[i - 2]) - f(x[i - 1],
y[i - 1]) + 2 * f(x[i], y[i])) / 3
        y_corr = y[i - 1] + h * (f(x[i - 1], y[i - 1]) + 4 * f(x[i], y[i]) +
f(x[i + 1], y_pred)) / 3
        while np.abs(y_corr - y_pred) > eps:
            y_pred = y_corr
            y_corr = y[i - 1] + h * (f(x[i - 1], y[i - 1]) + 4 * f(x[i],
y[i]) + f(x[i + 1], y_pred)) / 3
        y[i + 1] = y_corr
    return x, y
```

Полный код: https://github.com/aulouu/comp_math_labs/tree/main/lab6

Результат выполнения программы

```
Выберите уравнение:
1. y' = x^2 + x - 2y
2. y' = 2x - y + x^2
3. y' = 5x^2 - 2y/x
2
Введите начальное значение x0: 1
Введите начальное условие y0: 1
Введите конечное значение xn: 10
Введите шаг h: 1
Введите точность eps: 0.01

Таблица приближенных значений (метод Эйлера):
+------+------------+--------+----------+
|  x   |  y_approx  | y_true |  Error   |
+------+------------+--------+----------+
| 1.0  |    1.0     |  1.0   |   0.0    |
| 2.0  |  4.005259  |  4.0   | 0.005259 |
| 3.0  |  9.007199  |  9.0   | 0.007199 |
| 4.0  | 16.007915  |  16.0  | 0.007915 |
| 5.0  | 25.008179  |  25.0  | 0.008179 |
| 6.0  | 36.008276  |  36.0  | 0.008276 |
| 7.0  | 49.008312  |  49.0  | 0.008312 |
| 8.0  | 64.008326  |  64.0  | 0.008326 |
| 9.0  |  81.00833  |  81.0  | 0.00833  |
| 10.0 | 100.008332 | 100.0  | 0.008332 |
+------+------------+--------+----------+
Решение с шагом 0.125
```

Таблица приближенных значений (метод Рунге-Кутта 4-го порядка):

x	y_approx	y_true	Error
1.0	1.0	1.0	0.0
2.0	4.001046	4.0	0.001046
3.0	9.001431	9.0	0.001431
4.0	16.001573	16.0	0.001573
5.0	25.001625	25.0	0.001625
6.0	36.001644	36.0	0.001644
7.0	49.001652	49.0	0.001652
8.0	64.001654	64.0	0.001654
9.0	81.001655	81.0	0.001655
10.0	100.001655	100.0	0.001655

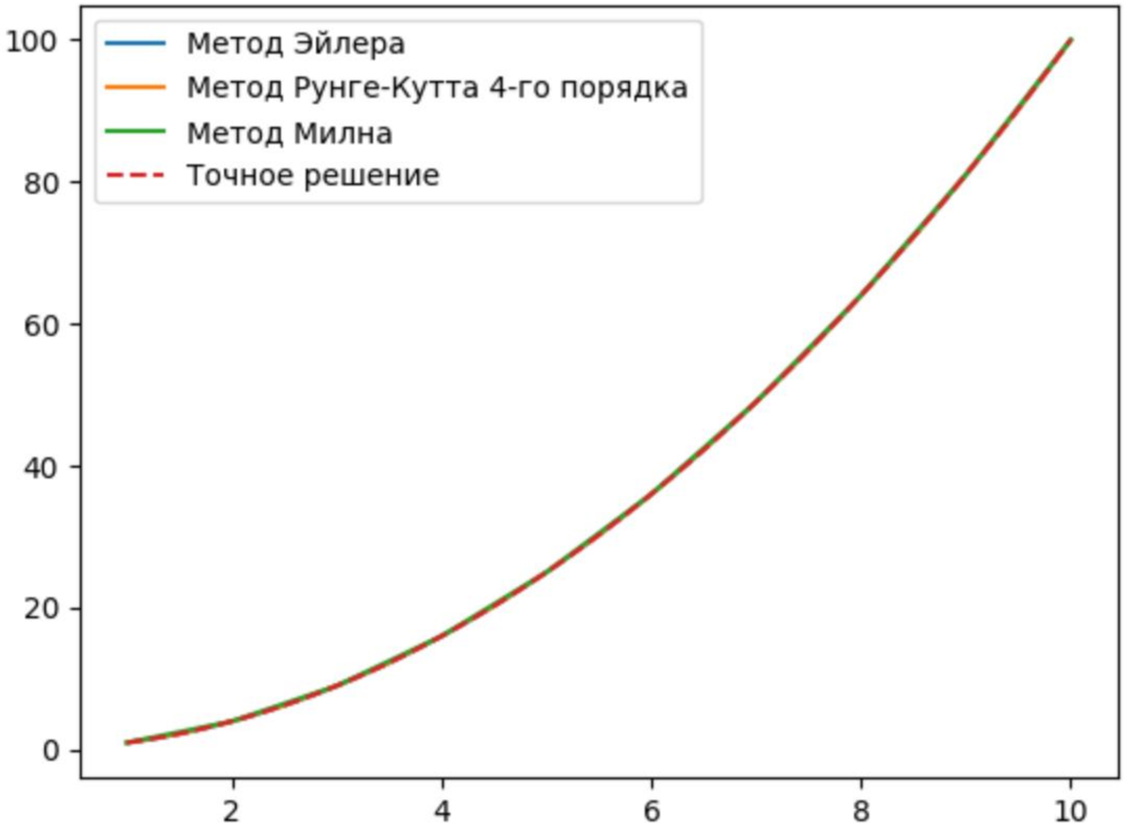
Решение с шагом 0.5

Таблица приближенных значений (метод Милна):

x	y_approx	y_true	Error
1.0	1.0	1.0	0.0
2.0	4.001046	4.0	0.001046
3.0	9.001431	9.0	0.001431
4.0	16.001573	16.0	0.001573
5.0	25.000549	25.0	0.000549
6.0	36.001029	36.0	0.001029
7.0	49.000586	49.0	0.000586
8.0	63.999932	64.0	6.8e-05
9.0	81.000892	81.0	0.000892
10.0	99.999766	100.0	0.000234

Решение с шагом 1.0

Графики точного решения и полученного приближенного решения



Вывод

В ходе данной лабораторной работы я познакомилась с одношаговыми и многошаговыми методами решения обыкновенных дифференциальных уравнений и реализовала методы модифицированного Эйлера, Рунге-Кутта 4-го порядка и Милна.