Отчет по Лабораторной работе №6
по курсу “вычислительная математика”

Вариант №3

Выполнил:
Студент группы p320820
Дробыш Дмитрий Александрович

Преподаватель:
Машина Екатерина Алексеевна

Санкт-Петербург, 2023

ЛАБОРАТОРНАЯ РАБОТА №6.
«ЧИСЛЕННОЕ РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ»

Цель лабораторной работы: решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

№ варианта задания лабораторной работы определяется как номер в списке группы согласно ИСУ.

1. Порядок выполнения работы

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

2. Требования и содержание отчета

Отчет должен содержать следующие разделы:

- Титульный лист,
- Цель работы,
- Описание алгоритма решения задачи,
- Рабочие формулы используемых методов,

- Листинг программы (по крайней мере, коды используемых методов),
- Скриншоты результатов выполнения программы при различных исходных данных (не менее трех),
- Графики точного решения и полученного приближенного решения,
- Выводы.

3. Варианты задания

Одношаговые методы:

1. Метод Эйлера,
2. Усовершенствованный метод Эйлера,
3. Метод Рунге-Кутта 4- го порядка.

Многошаговые методы:

4. Адамса,
5. Милна.

Таблица 1. Варианты задания для программной реализации задачи

№ варианта	Метод	№ варианта	Метод
1	1, 3, 4	21	1, 3, 4
2	2, 3, 5	22	1, 2, 5
3	1, 3, 5	23	2, 3, 4
4	1, 2, 4	24	1, 3, 4
5	2, 3, 4	25	1, 3, 5
6	1, 3, 5	26	2, 2, 4
7	1, 2, 4	27	1, 3, 4
8	2, 3, 4	28	1, 3, 5
9	1, 2, 5	29	2, 3, 5
10	1, 3, 5	30	1, 2, 4
11	2, 3, 4	31	1, 3, 4
12	1, 3, 4	32	1, 2, 5
13	1, 2, 5	33	2, 3, 4
14	2, 3, 5	34	1, 3, 4
15	1, 3, 4	35	1, 3, 5
16	1, 3, 5	36	2, 3, 4
17	1, 2, 4	37	1, 3, 5
18	1, 3, 4	38	1, 2, 4
19	1, 3, 4	39	2, 3, 4
20	2, 3, 5	40	2, 3, 5

4. Контрольные вопросы

1. Сформулируйте задачу Коши для дифференциального уравнения 1 порядка.
2. Что является решением для дифференциального уравнения 1 порядка?
3. В чем заключается суть метода конечных разностей?
4. Что такое разностная аппроксимация?
5. Геометрический смысл задачи Коши?
6. Что такое интегральная кривая?
7. Какое из условий теоремы существования и единственности решения задачи Коши для ОДУ является условием существования и какое условием единственности?
8. Что должно быть задано для решения ОДУ приближенными методами?
9. Какой порядок точности имеет метод Эйлера? Рунге-Кутта?
10. Перечислите основные одношаговые методы для численного решения ОДУ?
11. Перечислите основные многошаговые методы для численного решения ОДУ?
12. В чем заключается суть методов прогноза и коррекции?
13. Когда в методах прогноза и коррекции можно переходить на следующий этап вычислений?
14. Что такое правило Рунге и как оно используется в данной задаче?
15. Чтобы «запустить» метод Адамса, что необходимо вычислить?

Формула:

$$y_{i+1} = y_i + hf(x_i, y_i)$$

Листинг:

```python
def euler(f, x0, y0, h, xn):
    X = []
    Y = []
    X.append(x0)
    Y.append(y0)
    while x0 < xn:
        y0 = y0 + h * f(x0, y0)
        x0 += h
        X.append(x0)
        Y.append(y0)
    return X, Y
```

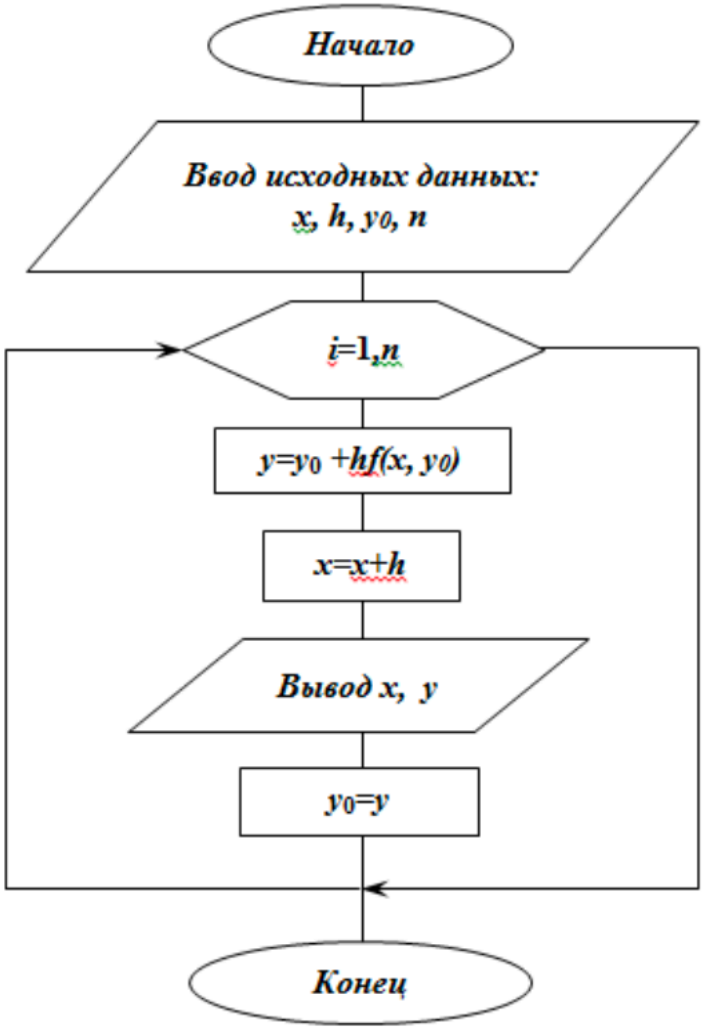


Рис.1 Блок-схема метода Эйлера

Пример работы:

Safari Файл Правка Вид История Закладки Разработка Окно Справка Чт, 18 мая 18:17

atozmath.com

Solve numerical differential equation using Euler method (1st order derivative) calculator | Округление чисел в Python 3: в большую или меньшую сторону — До целого или до сотых (2 заков)

Solution will be displayed step by step (in 2 parts)

Solution

Ad

Find y(1) for $y' = x \cdot x + y, x_0 = 0, y_0 = 0$, with step length 0.2 using Euler method (1st order derivative)

Solution:
Given $y' = x \cdot x + y, y(0) = 0, h = 0.2, y(1) = ?$

Euler method

$$y_1 = y_0 + hf(x_0, y_0) = 0 + (0.2)f(0, 0) = 0 + (0.2) \cdot (0) = 0 + (0) = 0$$

$$y_2 = y_1 + hf(x_1, y_1) = 0 + (0.2)f(0.2, 0) = 0 + (0.2) \cdot (0.04) = 0 + (0.008) = 0.008$$

$$y_3 = y_2 + hf(x_2, y_2) = 0.008 + (0.2)f(0.4, 0.008) = 0.008 + (0.2) \cdot (0.168) = 0.008 + (0.0336) = 0.0416$$

$$y_4 = y_3 + hf(x_3, y_3) = 0.0416 + (0.2)f(0.6, 0.0416) = 0.0416 + (0.2) \cdot (0.4016) = 0.0416 + (0.0803) = 0.1219$$

$$y_5 = y_4 + hf(x_4, y_4) = 0.1219 + (0.2)f(0.8, 0.1219) = 0.1219 + (0.2) \cdot (0.7619) = 0.1219 + (0.1524) = 0.2743$$

$\therefore y(1) = 0.2743$

PyCharm File Edit View Navigate Code Refactor Run Tools VCS Window Help Чт, 18 мая 18:17

Lab6 – euler.py

Lab6 › methods_for_solving_ordinary_differential_equat... › euler.py

Project: Lab6 ~/Desktop/Вычмат/Lab6
- methods_for_solving_ordinary_differentia
  - euler.py
  - fourth_order_runge_kutta.py
  - milna.py
  - runge_rule.py
- sys_util_lib
  - parsers_io.py
- venv
- main.py
- test.csv
- External Libraries
- Scratches and Consoles

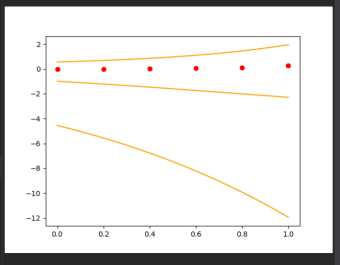
main.py | parsers_io.py | euler.py | fourth_order_runge_kutta.py | milna.py

```
import math


def euler(f, x0, y0, h, xn):
    X = []
    Y = []
    while x0 <= xn:
        X.append(x0)
        Y.append(y0)
        y0 = y0 + h * f(x0, y0)
        x0 += h
    return X, Y

```

SciView: Data Plots — 4-bit color) 15,85 kB



Run: main

```
/Users/demetrius/Desktop/Вычмат/Lab6/venv/bin/python /Users/demetrius/Desktop/Вычмат/Lab6/main.py
[0, 0.2, 0.4, 0.6000000000000001, 0.8, 1.0]
 [0, 0.0, 0.008000000000000002, 0.04160000000000001, 0.12192000000000004, 0.2743040000000001]

Process finished with exit code 0
```

Version Control | Run | Debug | Python Packages | TODO | Python Console | Problems | Terminal | Services

13:1 LF UTF-8 4 spaces Python 3.11 (Lab6)

Runge_Kutta:

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

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

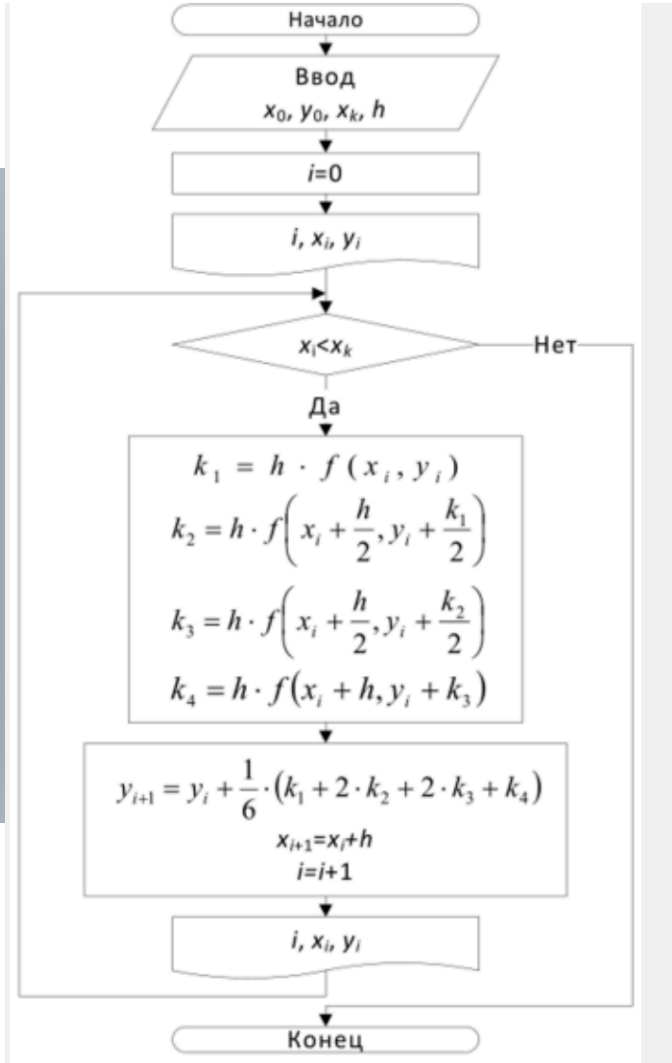
$$k_1 = h \cdot f(x_i, y_i)$$

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

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

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

```python
def fourth_order_runge_kutta(f, x0, y0, h, xn):
    X, Y = [], []
    X.append(x0)
    Y.append(y0)
    while x0 < xn:
        k1 = h*f(x0, y0)
        k2 = h*f(x0+h/2, y0+k1/2)
        k3 = h*f(x0+h/2, y0+k2/2)
        k4 = h*f(x0+h, y0+k3)
        x0 += h
        y0 = y0 + 1/6 * (k1 + 2 * k2 + 2 * k3 + k4)
        X.append(x0)
        Y.append(y0)
    return X, Y
```



Milna:

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).

```python
def milna(f, x, y, h, xn, eps):
    xnn = x + h*4
    X, Y = fourth_order_runge_kutta(f, x, y, h, xnn)
    F = []
    for i in range(4):
        F.append(f(X[i], Y[i]))

    # Prediction part:

    while X[-1] < xn:
        X.append(X[-1] + h)
        Y.append(Y[-4] + 4*h/3 * (2 * F[-3] - F[-2] + 2 * F[-1]))
        Fi = f(X[-1], Y[-1])
        y_correct = Y[-2] + h/3 * (F[-2] + 4*F[-1] - Fi)

        # Correction part:
        while abs(y_correct-Y[-1]) >= eps:
            Y[-1] = y_correct
            Fi = f(X[-1], y_correct)
            y_correct = Y[-2] + h/3 * (F[-2] + 4*F[-1] - Fi)
        Y[-1] = y_correct
        F[-1] = Fi
    return X, Y
```

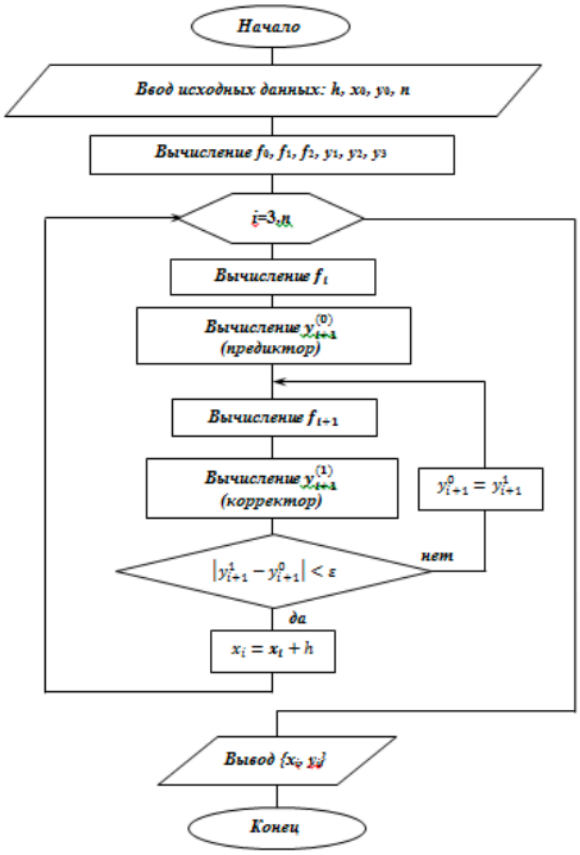


Рис. 3. Блок-схема метода предиктор-корректор

Вывод: В ходе лабораторной работы я попробовал различные методы решения ОДУ при помощи вычислительных методов на языке программирования Python3. Должен отметить, что выявил для себя наиболее оптимальным метод Милна.НО. Во-первых, сами методы имеют слишком большую погрешность, во-вторых, из-за разных порядков функций могут

быть проблемы с графиками, в-третьих, сами методы решения ОДУ на компьютере имеют слишком большую погрешность и не учитывают коэффициент С.
Таким образом, лабораторная имеет смысл лишь с точки зрения математики, но для компьютера это слишком слабо.