Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы численного анализа

ОТЧЁТ

к лабораторной работе
на тему

Методы Эйлера и Рунге-Кутта

Выполнил: студент группы 053502
Ермаков Антон Владимирович

Проверил: Анисимов Владимир Яковлевич

Минск 2022

Оглавление

Вариант 11

Цели выполнения задания

Изучить решение задачи Коши для обыкновенных дифференциальных уравнений методом Эйлера и методом Рунге-Кутта.

Теоретические сведения

■

9. ЧИСЛЕННОЕ РЕШЕНИЕ ЗАДАЧИ КОШИ ДЛЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Рассмотрим дифференциальное уравнение $y' = f(x, y)$ с начальным условием $y(x_0) = y_0$. Будем предполагать, что *f(x,y)* непрерывная и непрерывно дифференцируемая по y функция в окрестности замкнутой области

$$D = \{(x, y) \mid a \le x \le b, c \le y \le d\},$$

содержащей внутри себя точку (x_0, y_0).

Требуется решить задачу Коши: найти непрерывно дифференцируемую функцию *y=y(x)*, такую что $y'(x) = f(x, y(x))$ при всех $x \in [a,b]$ и $y(x_0) = y_0$.

Разобьем отрезок *[a, b]* с помощью точек разбиения $a = x_0, x_1, ..., x_n = b$ с шагом $h = (b-a)/n$. Тогда узлы разбиения имеют вид $x_k = x_0 + kh,\ k = \overline{0,n}$.

Пусть $y(x_0), y(x_1), ..., y(x_n)$ − значения функции в точках разбиения.

9.1. Метод ломаных Эйлера

Пусть $y = y(x)$ искомое решение задачи Коши. В точке (x_0,y_0) построим касательную (см. рис. 9.1) к графику $y = y(x)$.

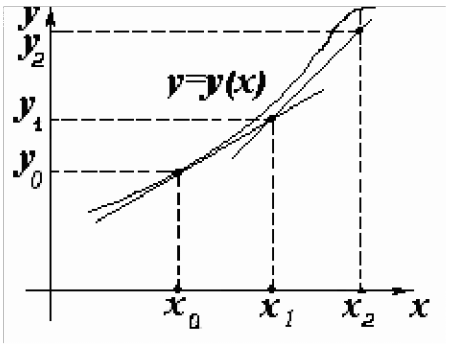

Рис. 9.1

Запишем уравнение касательной:

$$y = y_0 + y'(x_0)(x - x_0) = y_0 + f(x_0, y_0)(x - x_0).$$

и найдем точку пересечения этой касательной с прямой $x = x_1$:

$$y_1 = y_0 + hf(x_0, y_0).$$

Запишем уравнение прямой

$$y = y_1 + f(x_1, y_1)(x - x_1)$$

и найдем точку ее пересечения с прямой с $x = x_2$:

$$y_2 = y_1 + hf(x_1, y_1).$$

Продолжая процесс, получим рекуррентную последовательность:

$$y_{k+1} = y_k + hf(x_k, y_k), \qquad k = 0,1... \tag{9.1}$$

$$y_0 = y(x_0),$$

которую называют последовательностью Эйлера. Соединяя ломаными все точки (x_k, y_k), полученные из рекуррентной последовательности Эйлера, получим ломаную линию, приближающую график решения $y = y(x)$. Функция, график которой совпадает с ломаной Эйлера, принимается за приближенное решение задачи Коши.

Выясним точность метода Эйлера. Сравним значения точного решения y(x) задачи Коши в узловых точках со значениями, полученными методом Эйлера:

$$y(x_{k+1}) = y(x_k) + y'(x_k)h + O(h^2),$$

$$y_{k+1} = y_k + hf(x_k, y_k) + O(h^2).$$

Поскольку

$$y'(x_k) = f(x_k, y(x_k)),$$

то $y(x_{k+1}) - y_{k+1} = O(h^2)$ при условии, что $y_k = y(x_k)$. То есть, точность метода на отдельном отрезке $[x_k, x_{k+1}]$ совпадает с $O(h^2)$. Тогда, очевидно, точность метода Эйлера на всем отрезке [*a, b*] будет *O(h).*

Для повышения точности вычислений иногда используется модифицированный метод Эйлера, в котором рекуррентная последовательность Эйлера вычисляется по формулам

$$y_{k+1} = y_k + hf\left(x_k + \frac{h}{2}, y_k + \frac{h}{2} f(x_k, y_k)\right), \qquad k = 0,1..., n-1. \tag{9.2}$$

Модифицированный метод Эйлера обычно дает более точное приближение решения.

Пример. Пусть требуется решить задачу Коши:

$$\begin{cases} y' = -y, \ x \in [0,1] \\ y(0) = 1. \end{cases}$$

Полагая $h = 0{,}2$ и используя метод Эйлера, получим, как легко убедиться, из формулы Эйлера (9.1)

$$y_{k+1} = y_k + 0.2 \cdot (-y_k) = 0.8 \cdot y_k .$$

С другой стороны, используя модифицированный метод Эйлера, получим в силу формулы (2) рекуррентную последовательность

$$y_{k+1} = y_k + 0.2 \cdot (-y_k) = 0.82 \cdot y_k \ .$$

Поскольку точным решением задачи Коши, как легко проверить, является функция $y = e^{-x}$, можно сравнить точность обоих методов.

	0	1	2	3	4	5
x_k	0	0.2	0.4	0.6	0.8	1
y_k	1	0.8	0.64	0.572	0.4086	0.3277
$y_k^{модиф}$	1	0.82	0.6724	0.5514	0.4521	0.3708
e^{-x}	1	0.8187	0.6703	0.5488	0.4493	0.3679

Общепризнанным недостатком метода Эйлера является его не достаточно высокая точность. Несомненным достоинством метода Эйлера является его простота.

9.2. Методы Рунге-Кутта

1. Метод Рунге-Кутта второго порядка (или метод типа «предиктор-корректор»).

Метод состоит из двух этапов. Сначала находят по методу Эйлера грубое решение:

$$y^*_{\kappa+1} = y_k + hf(x_k, y_k) \ .$$

На следующем шаге это грубое решение сглаживается:

$$y_{k+1} = y_k + h\frac{f(x_k, y_k) + f(x_{k+1}, y^*_{k+1})}{2}, \qquad k = 0,1\ldots, n-1 .$$

Выясним точность метода. Преобразуя y_{k+1}, получаем:

$$y_{k+1} = y_k + \frac{h}{2} f(x_k, y_k) + \frac{h}{2} f(x_k + h, y_k + h \cdot f(x_k, y_k)) =$$

$$= y_k + \frac{h}{2} f(x_k, y_k) + \frac{h}{2} f(x_k, y_k) + \frac{h}{2}[f'_x(x_k, y_k) +$$

$$+ h \cdot f'_y(x_k, y_k) \cdot f(x_k, y_k) + O(h^2)] =$$

$$= y_k + hf(x_k, y_k) + \frac{h^2}{2}[f'_x(x_k, y_k) + + f'_y(x_k, y_k) \cdot f(x_k, y_k)] + O(h^3).$$

С другой стороны, разложим точное решение *y(x)* по формуле Тейлора. Получим

$$y(x_{k+1}) = y(x_k + h) = y(x_k) + y'(x_k)h + y''(x_k)\frac{h^2}{2} + O(h^3) = y(x_k) + hf(x_k, y(x_k)) +$$

$$+ \frac{h^2}{2}[f'_x(x_k, y(x_k)) + f'_y(x_k, y(x_k))f(x_k, y(x_k)) + O(h^3).$$

Полагая $y(x_k) = y_k$, получаем погрешность на отдельном шаге равную $O(h^3)$. Тогда на всем отрезке погрешность составит $O(h^2)$.

Достоинство метода: его точность превосходит точность метод Эйлера.

2. Метод Рунге-Кутта четвертого порядка.

На каждом шаге производится вычисление коэффициентов K_1, K_2, K_3, K_4:

$K_1 = hf(x_k, y_k)$;

$K_2 = hf(x_k + \frac{h}{2}, y_k + \frac{K_1}{2})$;

$K_3 = hf(x_k + \frac{h}{2}, y_k + \frac{K_2}{2})$;

$K_4 = hf(x_k + h, y_k + K_3)$.

Затем вычисляем

$y_{k+1} = y_k + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$.

Данный метод имеет точность $O(h^4)$ на $[a,b]$.

Рассмотрим пример, который мы использовали для иллюстрации точности метода Эйлера.

Пример. Требуется решить задачу Коши:

$\begin{cases} y' = -y \\ y(0) = 1 \end{cases}$ на отрезке [0, 1].

Выберем шаг $h = 0{,}2$. Результат вычислений поместим в таблицу.

	0	1	2	3	4	5
x_k	0	0.2	0.4	0.6	0.8	1
y_k	1	0.8187	0.6703	0.5487	0.4493	0.3678
e^{-x}	1	0.8187	0.6703	0.5488	0.4493	0.3679

Таким образом, метод Рунге-Кутта 4-го порядка отличается очень высокой точностью. К определенным его недостаткам относится большая сложность и трудоемкость (на каждом шаге необходимо четырежды вычислять значения функции f вместо одного раза в методе Эйлера).

Отметим, что на практике выбирают начальную длину шага h таким образом, чтобы $h^4 < \varepsilon$, где ε − заданная точность вычисления решения. Затем шаг выбирают вдвое меньшим и останавливают вычисления, если разность полученных значений y_k со значениями, полученными при начальном выборе шага меньше ε. В противном случае шаг еще раз уменьшают вдвое и т.д.

Задание

Вариант 11

ЗАДАНИЕ. С помощью метода Эйлера, а затем метода Рунге-Кутта найти с точностью до 0.001 решения следующих уравнений на отрезке [0; 1].

$$y' = \frac{a\left(1 - y^2\right)}{(1+m)x^2 + y^2 + 1},\ y(0) = 0,$$

где значения параметров a и m принимают следующие значения для вариантов k.

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14
m	1.0	1.5	2.0	1.0	1.5	2.0	1.0	1.5	2.0	1.0	1.5	2.0	1.0	2.0
a	0.5	0.7	0.9	1.1	1.3	0.5	0.7	0.9	1.1	1.3	0.5	0.7	0.9	1.0

Шаг интегрирования h, обеспечивающий требуемую точность, выбирать в процессе вычисления из сравнения результатов, полученных с h и $h/2$. В случае необходимости шаг h должен быть уменьшен.

Сравнить результаты.

Программная реализация

```
task_m = 1.5
task_a = 0.5


def f_der(x: float, y: float) -> float:
    return task_a * (1 - y**2) / ((1 + task_m) * x**2 + y**2 + 1)


def euler_method(f_der, begin: float, end: float, num_of_steps: int, y0: float) -> list:
    h = (end - begin) / (num_of_steps - 1)
    ydots = []
    x = begin
    y = y0
    while x <= end:
        ydots.append(y)
        y_next = y + h * f_der(x, y)
        x += h
        y = y_next
    return ydots


def modified_euler_method(f_der, begin: float, end: float, num_of_steps: int, y0: float) -> list:
    h = (end - begin) / (num_of_steps - 1)
    ydots = []
    x = begin
    y = y0
    while x <= end:
        ydots.append(y)
        y_next = y + h * f_der(x + h / 2, y + h / 2 * f_der(x, y))
        x += h
        y = y_next
    return ydots


def runge_kutta_method(f_der, begin: float, end: float, num_of_steps: int, y0: float) -> list:
    h = (end - begin) / (num_of_steps - 1)
    ydots = []
    x = begin
    y = y0
    while x <= end:
        ydots.append(y)
        k1 = h * f_der(x, y)
        k2 = h * f_der(x + h / 2, y + k1 / 2)
        k3 = h * f_der(x + h / 2, y + k2 / 2)
        k4 = h * f_der(x + h, y + k3)
        y_next = y + (k1 + 2 * k2 + 2 * k3 + k4) / 6
        x += h
        y = y_next
    return ydots


def solve_with_eps(method, f_der, begin: float, end: float, y0: float, start_num_of_steps: int, eps: float) -> list:
    cur_num_of_steps = start_num_of_steps
    old_y_dots = method(f_der, begin, end, cur_num_of_steps, y0)
    while True:
        is_suit = True
        new_y_dots = method(f_der, begin, end, 2 * cur_num_of_steps, y0)
        for i in range(len(old_y_dots)):
            if abs(old_y_dots[i] - new_y_dots[i * 2]) > eps:
                cur_num_of_steps *= 2
                old_y_dots = new_y_dots
                is_suit = False
                break
        if is_suit:
            return new_y_dots


def print_result(y_dots: list, begin: float, end: float, num_of_steps: int):
```

```
    h = (end - begin) / (len(y_dots) - 1)
    print(f"Шаг интегрирования h = {{:.4f}'.format(h)}")

    step_y = len(y_dots) // num_of_steps
    step_x = (end - begin) / num_of_steps
    x_dots = []
    x = begin
    print("x:", end=" ")
    while x <= end:
        x_dots.append(x)
        print("{:.4f}".format(x), end=" ")
        x += step_x
    print()
    print("y:", end=" ")

    for i in range(0, len(y_dots), step_y):
        print("{:.4f}".format(y_dots[i]), end=" ")
    print()


def main():
    begin = 0
    end = 1
    eps = 0.001
    y0 = 0
    start_num_of_steps = 4

    print(f"Метод Эйлера:")
    euler_y_dots = solve_with_eps(euler_method, f_der, begin, end, y0, start_num_of_steps, eps)
    print_result(euler_y_dots, begin, end, 10)

    print()
    print("Модифицированный метод Эйлера:")
    modified_euler_y_dots = solve_with_eps(modified_euler_method, f_der, begin, end, y0, start_num_of_steps, eps)
    print_result(modified_euler_y_dots, begin, end, 10)

    print()
    print("Метод Рунге-Кутта 4-ого порядка:")
    runge_kutta_y_dots = solve_with_eps(runge_kutta_method, f_der, begin, end, y0, start_num_of_steps, eps)
    print_result(runge_kutta_y_dots, begin, end, 10)


if __name__ == '__main__':
    main()
```

Полученные результаты

Задание по варианту:

```
Метод Эйлера:
Шаг интегрирования h = 0.0020
x: 0.0000 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000
y: 0.0000 0.0494 0.0961 0.1382 0.1749 0.2062 0.2326 0.2549 0.2737 0.2897 0.3034

Модифицированный метод Эйлера:
Шаг интегрирования h = 0.0039
x: 0.0000 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000
y: 0.0000 0.0486 0.0945 0.1360 0.1723 0.2034 0.2298 0.2521 0.2710 0.2870 0.3008

Метод Рунге-Кутта 4-ого порядка:
Шаг интегрирования h = 0.0039
x: 0.0000 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000
y: 0.0000 0.0486 0.0945 0.1360 0.1723 0.2034 0.2298 0.2521 0.2710 0.2870 0.3008
```

Выводы

В ходе выполнения этой лабораторной работы были изучены метод Эйлера и метод Рунге-Кутта 4-го порядка для решения задачи Коши для обыкновенных дифференциальных уравнений. Была написана программа для решения задачи Коши для обыкновенных дифференциальных уравнений этими методами. Видно, что для метода Эйлера потребовался меньший шаг интегрирования, чем для модифицированного метода Эйлера и метода Рунге-Кутта 4-го порядка. Это объясняется тем, что метод Рунге-Кутта 4-го порядка обладает большей точностью.