Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №8

Методы Эйлера и Рунге-Кутта

Вариант: 19

Выполнил: студент группы 953502

Потейчук Вероника Михайловна

Проверил: Анисимов Владимир Яковлевич

Минск 2021

ЦЕЛЬ ЗАДАНИЯ

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

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

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 $a = x_0, x_1, \ldots, x_n = b$ с шагом $h = (b-a)/n$. Тогда узлы разбиения имеют вид $x_k = x_0 + kh,\ k = \overline{0,n}$.

Пусть $y(x_0), y(x_1), \ldots, y(x_n)$ - значения функции в точках разбиения.

1) Метод Эйлера

Построим рекуррентную последовательность:

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

$$y_0 = y(x_0),$$

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.

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

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

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.

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

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

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.

РЕАЛИЗАЦИЯ

Вариант 1

- Шаг интегрирования: 0.25

Методом Рунге-Кутта: 0, 0.2083459088037499, 0.35280473997149897, 0.4419261313306102, 0.4981503549926386

Точное решение: 0.0, 0.20833216829991202, 0.35282853085646004, 0.4419545435559948, 0.4981770618218474

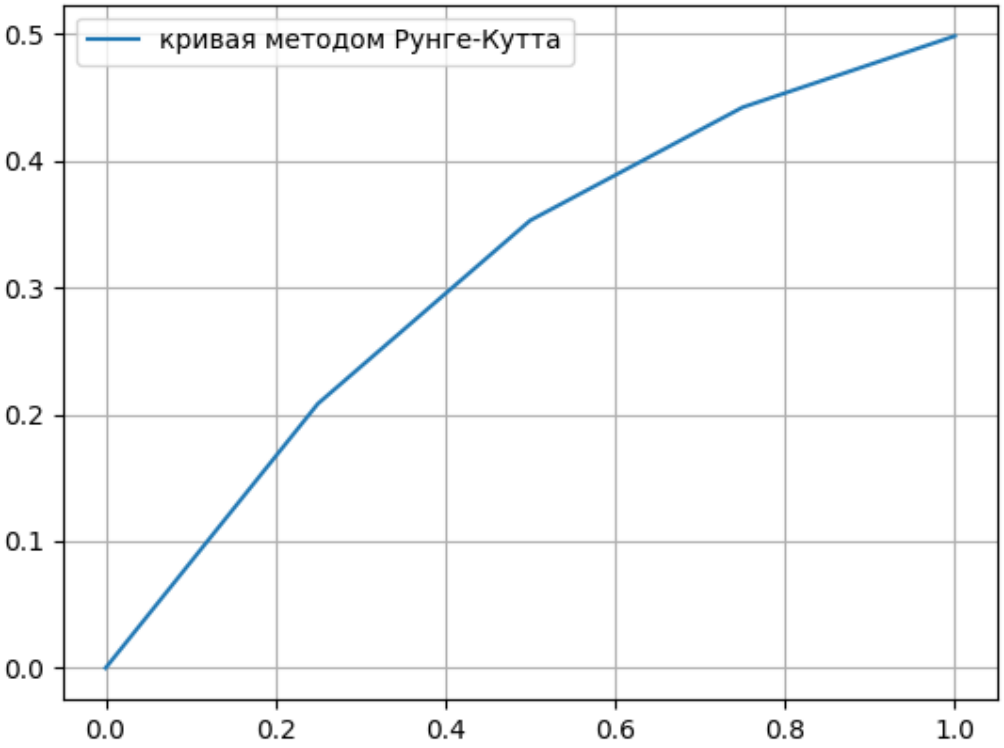
Методом Эйлера: 0, 0.1945945945945946, 0.32477933277418447, 0.404909962744314, 0.45625091329848605

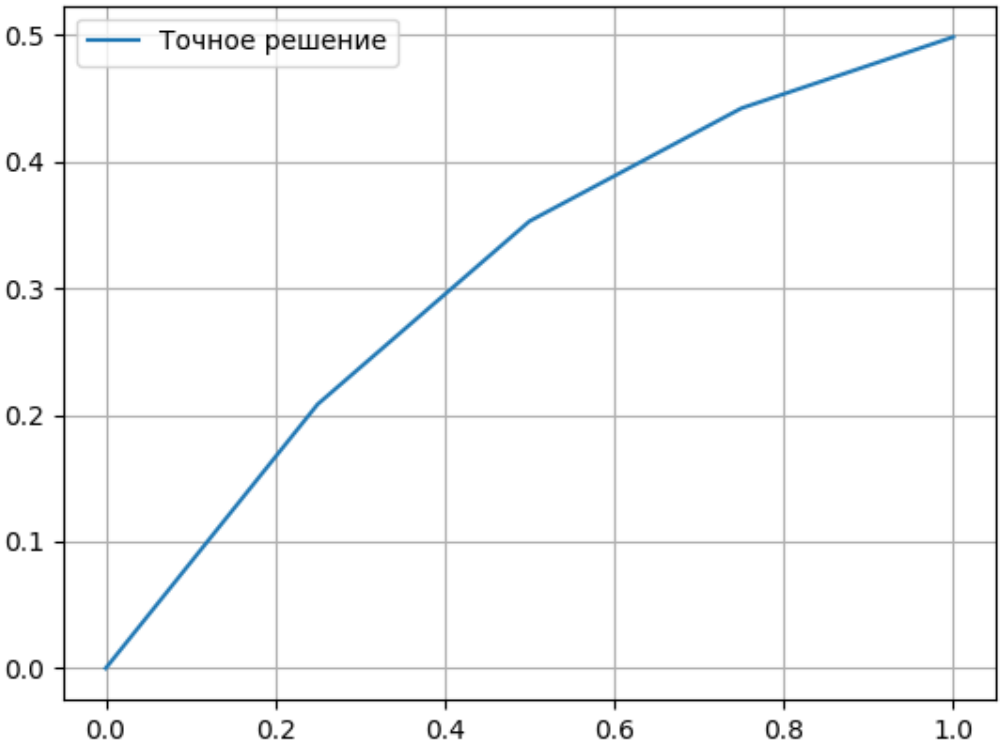
```python
a = 0
b = 1
y0 = 0
epsilon = 0.0001


def f(y, x):
    return (0.9*(1-y**2))/((1+1.5)*(x**2)+(y**2)+1)

def runge_kutta(x, step):
    if x <= a:
        return y0
    prev_x = x - step
    prev_y = runge_kutta(prev_x, step)
    F1 = f(prev_y, prev_x)
    F2 = f(prev_y + (step / 2) * F1, prev_x + step / 2)
    F3 = f(prev_y + (step / 2) * F2, prev_x + step / 2)
    F4 = f(prev_y + step * F3, x)
    return prev_y + step * (F1 + 2 * F2 + 2 * F3 + F4) / 6

def find_step(epsilon):
    h0 = epsilon ** (1/4)
    n = int((b - a) // h0)
    if n % 2 != 0 :
        n += 1
    while check_step(n, epsilon):
        n = n // 4 * 2
    while not check_step(n, epsilon):
        n += 2
    return (b - a) / n

def check_step(n, epsilon):
    h = (b - a) / n
    y2 = runge_kutta(a + 2 * h, h)
    y2e = runge_kutta(a + 2 * h, h * 2)
    eps = (1/15) * abs(y2 - y2e)
    return eps < epsilon
```

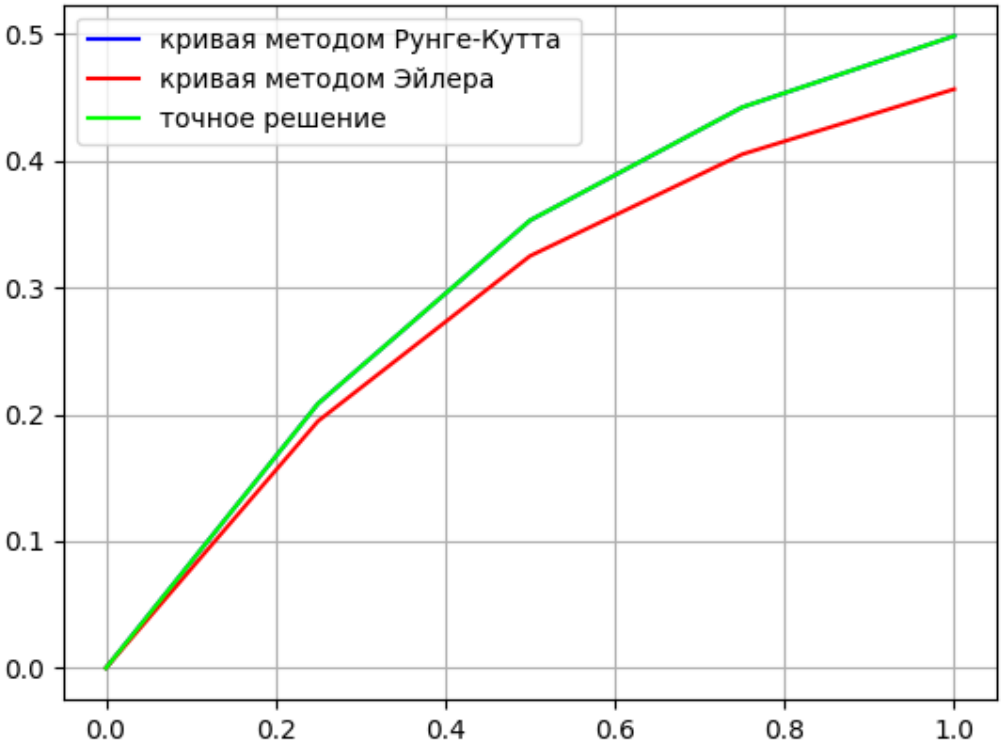
```
def accurate_solution(x):
    sol = odeint(f, y0, [a, x])
    return sol[1][0]

def show_curve(method, step, text):
    pylab.cla()
    xlist = numpy.arange(a, b + step, step)
    ylist = []
    x = a
    while x <= b:
        r = method(x, step)
        ylist.append(r)
        x += step
    pylab.plot (xlist, ylist, label = text)
    print(f"Методом Рунге-Кутта: {ylist}")
    pylab.grid(True)
    pylab.legend()
    pylab.savefig("1.png")
    ylist2 = ylist
    ylist = []
    x = a
    while x <= b:
        r = accurate_solution(x)
        ylist.append(r)
        x += step
    pylab.cla()
    pylab.plot (xlist, ylist, label = "Точное решение")
    print(f"Точное решение: {ylist}")
    pylab.grid(True)
    pylab.legend()
    pylab.savefig("2.png")
    rez = []
"""    for item in range(len(ylist)):
        rez.append(abs(ylist[item]-ylist2[item]))
    print(f"Погрешность между зачениями: {rez}")"""

show_curve(runge_kutta, step, "кривая методом Рунге-Кутта")
```

```
def euler(x, step):
    if x <= a:
        return y0
    prev = euler(x - step, step)
    return prev + step * f(prev, x)

xlist = numpy.arange(a, b + step, step)
runge_kutta_points = []
euler_points = []
accurate_solution_points = []
x = a
while x <= b:
    r = runge_kutta(x, step)
    r1 = euler(x, step)
    r2 = accurate_solution(x)
    runge_kutta_points.append(r)
    euler_points.append(r1)
    accurate_solution_points.append(r2)
    x += step

print(f"Методом Эйлера: {euler_points}")
"""rez = []
for item in range(len(euler_points)):
    rez.append(abs(euler_points[item]-runge_kutta_points[item]))"""

print(f"Погрешность между зачениями: {rez}")
pylab.cla()
pylab.plot (xlist, runge_kutta_points,label = "кривая методом Рунге-Кутта ", color = (0, 0, 1))
pylab.plot (xlist, euler_points,label = "кривая методом Эйлера", color = (1, 0, 0))
pylab.plot (xlist, accurate_solution_points,label = "точное решение", color = (0, 1, 0))
pylab.grid(True)
pylab.legend()
pylab.savefig("combine.png")
```

Графики


кривая методом Рунге-Кутта
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Точное решение
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
1.0



Рассмотрим погрешности при уменьшении шага интегрирования

- Шаг интегрирования: 0.125

Погрешность между значениями Эйлера и Рунге-Кутта:
0.017933339502368417

- Шаг интегрирования: 0.0625

Погрешность между значениями Эйлера и Рунге-Кутта:
0.008207735330544197

ВЫВОДЫ

В ходе выполнения лабораторной работы были изучены методы Эйлера и Рунге-Кутта для решения задачи Коши обыкновенных дифференциальных уравнений. Также было наглядно продемонстрировано, что не смотря на простоту метода Эйлера, он является не точным и чем больше было взято значений, тем сильнее расходились кривые в сравнении с методом Рунге-Кутта, который оказался более точным и довольно близок к точному решению. Также при уменьшении шага в 2 раза наблюдается уменьшении разницы между методами в 2 раза (приблизительно), что означает линейную зависимость.