МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

**Отчет по заданию №5**

**по дисциплине**

**«Численные методы»**

Выполнил: студент гр. БЭИ2202

Кулешов А. С.

Вариант 16.

Проверил: доц. каф. «Информатика»

Мацкевич А. Г.

Москва, 2023 г.

1. **Задание для численного решения обыкновенных дифференциальных уравнений:**

* дифференциальное уравнение ;
* интервал [0; 0.6];
* начальные условия x0=0, y0=-4;
* шаг интегрирования h0=0.2.

1. **Точное аналитическое решение заданного дифференциального уравнения**

Найдем точное аналитическое решение заданного дифференциального уравнения (решение y=y(x)). методом разделения переменных. Для этого запишем уравнение в виде . и проинтегрируем с учетом начальных условий. Получим . Из начальных условий следует, что .

Аналитическое решение дифференциального уравнения

1. **Значения точного решения ОДУ –y(x)**

Вычислим в сценарии значения полученного решения **y(xi)** на отрезке [0;0.6] с шагом изменения аргумента h=0.2:

|  |  |
| --- | --- |
| **xi** | **y(xi)** |
| 0 | -4 |
| 0.2 | -3.87596 |
| 0.4 | -3.18471 |
| 0.6 | -2.145922 |

1. **Численное решение заданного ОДУ методом Эйлера**

Вычислим в сценарии значения численного решение ОДУ методом Эйлера () в точках отрезка [0;0.6] с шагом h=0.2. Для этого ОДУ записывают в виде y’=f(x,y) . Общая формула для определения очередного значения функции по методу Эйлера имеет вид yi+1=yi+h⋅f(xi,yi), где , :

|  |  |
| --- | --- |
| **xi** |  |
| 0 | -4 |
| 0.2 | -4 |
| 0.4 | -3.616 |
| 0.6 | -2.364 |

1. **Значения погрешностей**

Вычислим в сценарии значения погрешностей для,,:

|  |  |
| --- | --- |
| **xi** | **Ei** |
| 0 | 0 |
| 0.2 | 0.125 |
| 0.4 | 0.432 |
| 0.6 | 0.219 |

1. **Результаты решения ОДУ методом Рунге-Кутта 4-го порядка, дополненным методом автоматического выбора шага, обеспечивающим точность 10-4**

Использую ЯП Python для решения данной задачи

def runge\_kutta(f, x0, y0, h, n):

    result = [y0]

    for i in range(n-1):

        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)

        y = y0 + (k1 + 2\*k2 + 2\*k3 + k4) / 6

        result.append(y)

        x0 += h

        y0 = y

    return result

# Пример использования

def f(x, y):

    return 3\*x\*\*2  \* y\*\*2

x0 = 0  # начальное значение x

y0 = -4  # начальное значение y

h = 0.2  # шаг

n = 4  # количество итераций

result = runge\_kutta(f, x0, y0, h, n)

print(result)

|  |  |
| --- | --- |
| **xi** | **y(xi)** |
| 0 | -4 |
| 0.2 | -3.87595949 |
| 0.4 | -3.1846562 |
| 0.6 | -2.1459317 |

1. **Значения погрешностей**

Вычислим в сценарии значения погрешностей , 

|  |  |
| --- | --- |
| **xi** |  |
| 0 | 0 |
| 0.2 | 0.00000051 |
| 0.4 | 0.000053 |
| 0.6 | 0.0000097 |

Все решения, полученные выше, сведем в табл. результатов 4-2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **xi** | **y(xi)** |  | **Ei** |  |  |
| 0 | -4 | -4 | 0 | -4 | 0 |
| 0.2 | -3.87596 | -4 | 0.125 | -3.87595949 | 0.00000051 |
| 0.4 | -3.18471 | -3.616 | 0.432 | -3.1846562 | 0.000053 |
| 0.6 | -2.145922 | -2.364 | 0.219 | -2.1459317 | 0.0000097 |

– аналитическое решение ОДУ,

 - решение ОДУ, полученное методом Эйлера, ,

- решение ОДУ методом Рунге-Кутты 4-го порядка, .

**8. Решение ОДУ при помощи ЯП Python**

def runge\_kutta(f, x0, y0, h, n):

    result = [y0]

    for i in range(n-1):

        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)

        y = y0 + (k1 + 2\*k2 + 2\*k3 + k4) / 6

        result.append(y)

        x0 += h

        y0 = y

    return result

# считаем погрешность для маленького отрезочка

def runge\_double\_prosch(f, x0, y0, h, n):

  m1 = runge\_kutta(f,x0,y0,h,n)

  m2 = runge\_kutta(f,x0,y0,h/2,n\*2-1)

  return abs(m1[-1] - m2[-1]), m1

def runge\_double\_prosch\_big(f, x0, y0, h, n):

  y\_prev = y0

  y\_vals = [y0]

  for i in range(n-1):

    error = 10\*\*9

    t\_h = h

    t\_n = 1

    m1 = []

    while (error > 10\*\*-5):

      error, m1 = runge\_double\_prosch(f,x0 + i\*h, y\_prev ,t\_h,t\_n+1)

      t\_h /= 2

      t\_n \*= 2

    print(f"На отрезке [{x0+i\*h};{x0+(i+1)\*h}] было использовано {t\_n//2} точек")

    y\_prev = m1[-1]

    y\_vals.append(m1[-1])

  return y\_vals

def euler\_method(f, x0, y0, h, n):

    result = [y0]

    for i in range(n-1):

        y = y0 + h \* f(x0, y0)

        result.append(y)

        x0 += h

        y0 = y

    return result

def f(x, y):

    return 3\*x\*\*2  \* y\*\*2

x0 = 0  # начальное значение x

y0 = -4  # начальное значение y

h = 0.2  # шаг

n = 4  # количество итераций

error = 10\*\*9

result\_kuuta = runge\_double\_prosch\_big(f,x0,y0,h,n)

#print(result\_kuuta)

result\_euler = euler\_method(f,x0, y0, h, n)

#print(result\_euler)

def exact(val):

  return 1/(-0.25-val\*\*3 )

result\_exact = []

for i in range(n):

  result\_exact.append(exact( x0 + (i)\*h))

#print(result\_exact)

error\_max = 0

for i in range(n):

  error\_max = max(error\_max, abs(result\_exact[i]-result\_kuuta[i]))

print('Реальная ошибка метода Куута: ', error\_max )

import matplotlib.pyplot as plt

import numpy as np

x\_100 = np.linspace(0, 0.6, 100)

y\_100 = exact(x\_100)

plt.plot(x\_100, y\_100, label='Функция')

x = [0, 0.2, 0.4, 0.6]

plt.plot(x, result\_euler, label='Методом Эйлера')

#print(x)

#print(result\_kuuta)

plt.plot( x , result\_kuuta, label='Методом Рунге-Кутта ')

# Настройка осей и легенды

plt.xlabel('Индекс элемента')

plt.ylabel('Значение в точке')

plt.legend()

# Отображение графика

plt.show()

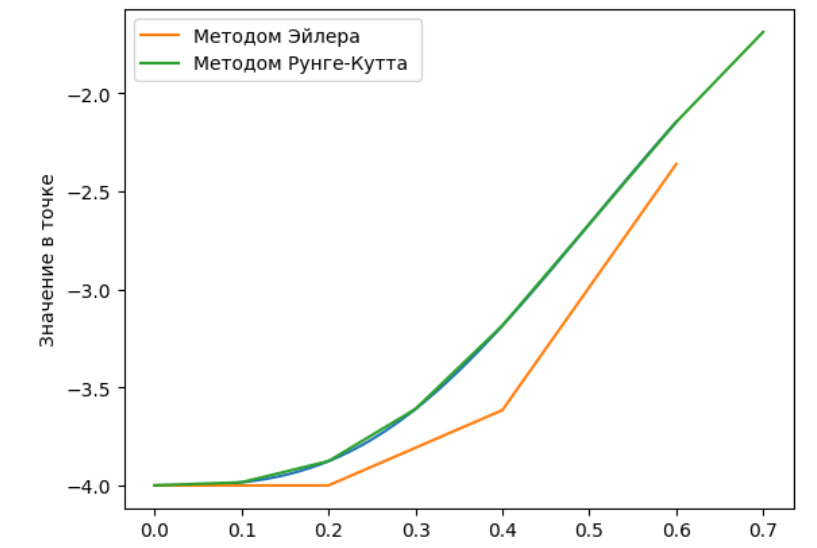
****

Рисунок 1 – результаты расчёта ОДУ в графическом виде

**Вывод**: В данном примере значения решений ОДУ методом Рунге-Кутта 4-го порядка (y4(x))и аналитическим методом (y(x)), практически совпадают. В решении ОДУ методом Эйлера(y1(x)), по мере удаления от начальной точки, погрешность накапливается за счет допущений, принятых в методе.