**Министерство образования Республики Беларусь**

**Белорусский государственный университет**

Факультет прикладной математики и информатики

Лабораторная работа №4

По курсу «Численные методы»

**Решение задачи Коши**

Вариант №2

Работу выполнил:

студент 3 курса 7 группы

**Шатерник Артём**

Преподаватель:

**Будник А. М**.

**Минск 2024**

1. **Постановка задачи.**

Требуется найти приближённое решение задачи Коши

на сетке из 10 узлов применяя следующие методы:

* Неявный метод Эйлера. Для его реализации использовать алгоритм Ньютона.
* Метод Рунге-Кутта при
* Экстраполяционный метод Адамса 3-го порядка с началом таблицы, построенным по соответствующему методу последовательного повышения порядка точности.

1. **Алгоритм решения.**

Построим сетку из 10 узлов. Шагом в таком случае будет и мы получим узлы:

**Неявный метод Эйлера.**

В общем виде от задаётся формулой:

Для его разрешения будем использовать метод Ньютона, который в будет задан следующими формулами:

В качестве условия остановки процесса возьмём следующее:

При этом в данном случае функций и её производная будут иметь вид:

**Метод Рунге-Кутта.**

При данных в условии значениях параметров мы получаем метод второго порядка, который задаётся следующими формулами:

**Экстраполяционный метод Адамса 3-го порядка.**

Он задаётся следующими формулами:

Для его построения требуется найти так называемое начало таблицы, в данном случае значения Их будем искать методом последовательного повышения порядка точности 3-го порядка точности, который имеет вид:

**3. Листинг программы.**

**import math  
import numpy as np**

**def func(x, y):  
 return (y\*\*2 \* math.log(x) - y) / x  
  
def func\_der(x, y):  
 return (2 \* y \* math.log(x) - 1) / x  
  
x0, y0 = 1, 1  
a, b = 1, 2  
N = 10  
h = (b - a) / N  
split = [a + h \* i for i in range(N + 1)]  
print(f"Разбиение:\n{split}")**

**# Реальное решение  
def func\_res(x):  
 return 1 / (math.log(x) + 1)  
  
real\_res = []  
for i in split:  
 real\_res.append(func\_res(i))  
 print(f"x = {round(i, 2)}, y = {func\_res(i)}")**

**# Неявный метод Эйлера  
# Метод Ньютона для разрешения  
def newton\_method(x0, y0, f, f\_der, h):  
 y\_new = y0  
 while True:  
 F = y\_new - y0 - h \* f(x0 + h, y\_new)  
 F\_der = 1 - h \* f\_der(x0 + h, y\_new)  
 y\_next = y\_new - F / F\_der  
 if abs(y\_new - y\_next) <= h\*\*3:  
 return y\_next  
 y\_new = y\_next  
  
def euler\_method(split, f, df, y0):  
 h = (split[1] - split[0])  
 y\_new = y0  
 y\_res = [y0]  
 for i in range(len(split) - 1):  
 y\_new = newton\_method(split[i], y\_new, f, df, h)  
 y\_res.append(y\_new)  
 return y\_res  
  
res\_euler = euler\_method(split, func, func\_der, y0)  
for t, y in zip(split, res\_euler):  
 print(f"x = {t:.1f}, y = {y:.6f}")**

**for i in range(len(res\_euler)):  
 print(f"{abs(real\_res[i] - res\_euler[i]):.6e}")**

**# Метод Рунге-Кутта  
def runge\_kutt(split, f, y0):  
 h = (split[1] - split[0])  
 y\_new = y0  
 y\_res = [y0]  
 for i in range(len(split) - 1):  
 p1 = h \* f(split[i], y\_new)  
 p2 = h \* f(split[i] + h, y\_new + p1)  
 y\_new = y\_new + 0.5 \* (p1 + p2)  
 y\_res.append(y\_new)  
 return y\_res  
  
res\_runge\_kutt = runge\_kutt(split, func, y0)  
for t, y in zip(split, res\_runge\_kutt):  
 print(f"x = {t:.1f}, y = {y:.6f}")**

**for i in range(len(res\_runge\_kutt)):  
 print(f"{abs(real\_res[i] - res\_runge\_kutt[i]):.6e}")**

**# Экстраполяционный метод Адамса 3-го порядка  
def inc\_order(x, y, f, h):  
 y13 = y + 1 / 3 \* h \* f(x , y)  
 y23 = y + 2 / 3 \* h \* f(x + h / 3, y13)  
 y\_res = y + h / 4 \* (f(x, y) + 3 \* f(x + h \* 2 / 3, y23))  
 return y\_res**

**def adams\_method(split, f, y0):  
 h = split[1] - split[0]  
 y\_res = [y0]  
 y\_res.append(inc\_order(split[0], y\_res[0], f, h))  
 y\_res.append(inc\_order(split[1], y\_res[1], f, h))  
 for i in range(2, len(split) - 1):  
 y\_res.append(y\_res[i] + h / 12 \* (23 \* f(split[i], y\_res[i]) -  
 16 \* f(split[i - 1], y\_res[i - 1]) + 5 \* f(split[i - 2], y\_res[i - 2])))  
 return y\_res  
  
res\_adams = adams\_method(split, func, y0)  
for t, y in zip(split, res\_adams):  
 print(f"x = {t:.1f}, y = {y:.6f}")**

**# Невязки  
for i in range(len(res\_adams)):  
 print(f"{abs(real\_res[i] - res\_adams[i]):.6e}")**

**4. Результат и его анализ.**

В качестве эталонного решения будем использовать решение, полученное с помощью Wolfram Alpha:

Решения в узлах:

|  |  |  |  |
| --- | --- | --- | --- |
| Точное решение | Эйлер | Рунге-Кутт 2 порядка | Адамс 3 порядка |
| 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| 0.912983 | 0.923440 | 0.912600 | 0.912940 |
| 0.845794 | 0.862847 | 0.845178 | 0.845732 |
| 0.792164 | 0.813621 | 0.791401 | 0.790843 |
| 0.748239 | 0.772775 | 0.747378 | 0.746346 |
| 0.711508 | 0.738290 | 0.710581 | 0.709272 |
| 0.680270 | 0.708749 | 0.679296 | 0.677848 |
| 0.653327 | 0.683131 | 0.652319 | 0.650798 |
| 0.629808 | 0.660680 | 0.628774 | 0.627217 |
| 0.609068 | 0.640825 | 0.608015 | 0.606440 |
| 0.590616 | 0.623126 | 0.589547 | 0.587966 |

Невязки в узлах:

|  |  |  |
| --- | --- | --- |
| Эйлер | Рунге-Кутт 2 порядка | Адамс 3 порядка |
| 0.000000e+00 | 0.000000e+00 | 0.000000e+00 |
| 1.045618e-02 | 3.833324e-04 | 4.346047e-05 |
| 1.705365e-02 | 6.159653e-04 | 6.124054e-05 |
| 2.145662e-02 | 7.635166e-04 | 1.321260e-03 |
| 2.453675e-02 | 8.607882e-04 | 1.892705e-03 |
| 2.678155e-02 | 9.271712e-04 | 2.236644e-03 |
| 2.847846e-02 | 9.739510e-04 | 2.422140e-03 |
| 2.980452e-02 | 1.007947e-03 | 2.528343e-03 |
| 3.087280e-02 | 1.033416e-03 | 2.590199e-03 |
| 3.175782e-02 | 1.053089e-03 | 2.627134e-03 |
| 3.250997e-02 | 1.068760e-03 | 2.649764e-03 |

* Неявный метод Эйлера.

Он имеет локальную погрешность 2 порядка, все невязки также имеют второй порядок, что соответствует теории при

При этом даже на последних узлах погрешность не успела накопиться, чтобы превзойти

* Метод Рунге-Кутта 2 порядка.

Он имеет локальную погрешность 3 порядка. В начале он давал более точные значения, но потом погрешность накопилась и стала как раз 3 порядка.

* Метод Адамса 3 порядка.

Второй и третий узел был найден с помощью метода ПППТ 3 порядка, они имеют невязки порядка , что соответствует точности метода. Остальные узлы были найдены уже методом Адамса, он имеет локальную погрешность 4 порядка и глобальную погрешность 3 порядка, а также является многошаговым. Все невязки на этих узлах имеют порядок , что соответствует глобальной погрешности метода. Ожидалось, что значения, полученные этим методом, будут иметь наилучшую точность, однако он дал такую же точность, как метод 2 порядка, (хотя точность всё ещё соответствует теории), это может быть связанно с самим видом функции