Київський національний університет імені Тараса
Шевченка
Факультет комп’ютерних наук та кібернетики

ЛАБОРАТОРНА РОБОТА №3
З курсу “Комп’ютерне моделювання”

Виконав:
Студент 3 курсу, групи ТТП-31
Спеціальності “Комп’ютерні науки”
Масич Нікіта Дмитрович

Київ - 2023

Завдання в загальній формі:

$$y' = \alpha t^2 + \beta y,\ y(\alpha) = \beta,\ t \in [\alpha, \alpha + 3]$$

Знаємо, що

$$\alpha = 1,\ \beta = 2$$

Отже:

$$y' = t^2 + 2y,\ y(1) = 2,\ t \in [1, 4]$$

Потрібно розв’язати методами:

- Рунге-Кутта-2
- Мілна-Симпсона

З точністю до $\varepsilon = 10^{-4}$

Для розв’язку, напишемо програму мовою python.

Декларуємо вхідні дані:

```python
# Задані початкові умови
y0 = [2]
t_span = [1, 4]
epsilon = 1e-4  # Задана точність

# Функція, що визначає диференціальне рівняння
def func(t, y):
    return t**2 + 2*y
```

За допомогою функції solve_ivp з бібліотеки scipy знаходимо розв’язок методом Рунге-Кутта-2:

```
t_eval = np.linspace(t_span[0], t_span[1], 100)  # Генеруємо 100 точок для побудови графіка

# Розв'язуємо рівняння за допомогою методу Рунге-Кутта-2
sol_rk2 = solve_ivp(func, t_span, y0, method='RK23', t_eval=t_eval)
```

Тут RK23 вказує на необхідний метод відповідно до офіційної документації:

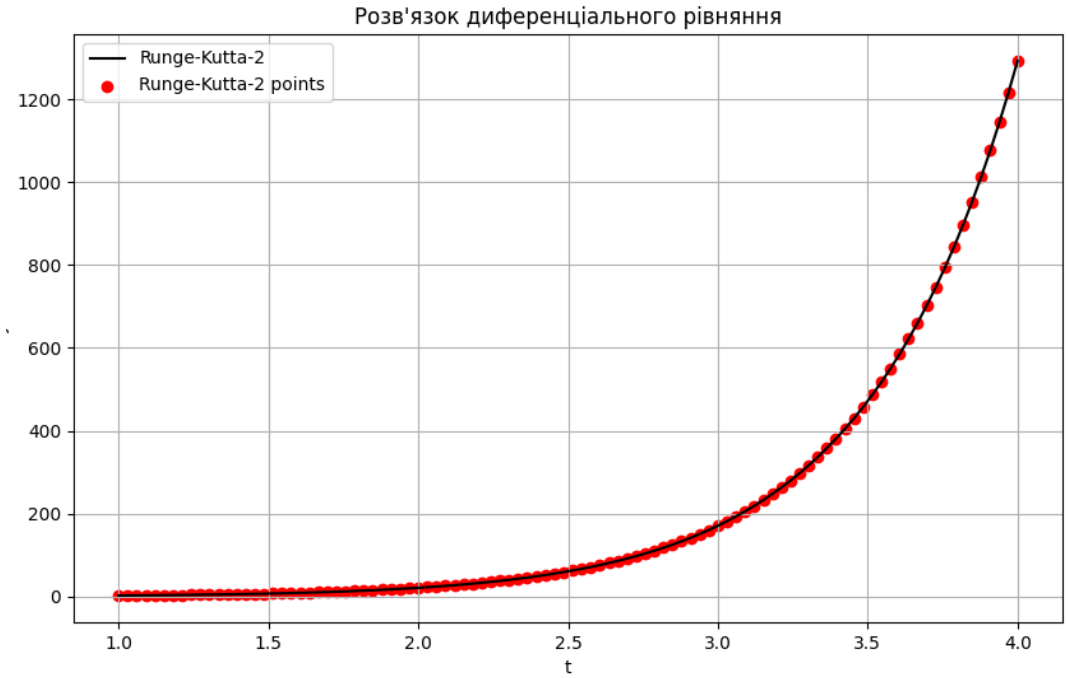
- ‘RK23’: Explicit Runge-Kutta method of order 3(2) [3]. The error is controlled assuming accuracy of the second-order method, but steps are taken using the third-order accurate formula (local extrapolation is done). A cubic Hermite polynomial is used for the dense output. Can be applied in the complex domain.

Для розв’язку методом Мілна-Сімпсона запрограмуємо наступні дії:

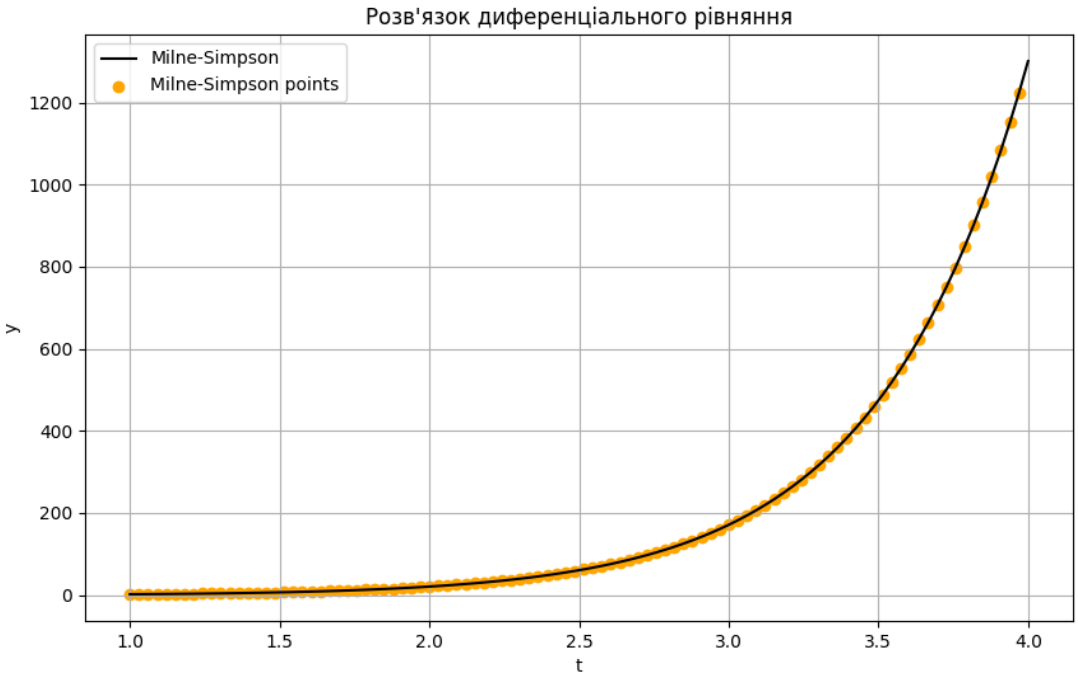
```
# Розв'язуємо рівняння та отримуємо значення y за допомогою методу Мілна-Сімпсона
y_milne = np.zeros_like(t_eval)
y_milne[0] = y0[0]

milne_points = []  # Зберігаємо точки, де обчислюємо Мілн

for i in range(1, len(t_eval)):
    h = t_eval[i] - t_eval[i-1]
    k1 = h * func(t_eval[i-1], y_milne[i-1])
    k2 = h * func(t_eval[i-1] + h/2, y_milne[i-1] + k1/2)
    k3 = h * func(t_eval[i-1] + h/2, y_milne[i-1] + k2/2)
    k4 = h * func(t_eval[i], y_milne[i-1] + k3)

    y_milne[i] = y_milne[i-1] + (k1 + 2*k2 + 2*k3 + k4) / 6

    milne_points.append([t_eval[i-1], y_milne[i-1]])  # Зберігаємо точки для відображення

    # Перевірка точності і зупинка, якщо досягнута
    if np.abs(y_milne[i] - y_milne[i-1]) < epsilon:
        break
```

Результати:

Маємо наступний графік, відповідний до методу Рунге-Кутта-2:



І аналогічний для методу Мілне-Сімпсона:



Результати цих методів демонструють, як апроксимована функція $y(t)$ змінюється з t відповідно до вказаного диференціального рівняння та початкової умови.

Код розв’язку можна побачити за посиланням:

https://github.com/NikitaMasych/km-lab3