Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский университет
"Высшая школа экономики"

Московский институт электроники и математики имени А. Н. Тихонова
Программа "Прикладная математика"

Лабораторная работа №6

ЧИСЛЕННОЕ РЕШЕНИЕ ЗАДАЧИ КОШИ

Группа БПМ213
Вариант 8
Номера выполняемых задач: 7.1.8, 7.4.3, 7.6.3

Выполнила:
Варфоломеева Анастасия Андреевна
Преподаватель:
Токмачев Михаил Геннадьевич

Москва, 2024 г.

1 Задачи Коши для ОДУ 1 порядка

1.1 Формулировка задачи

Найти приближенное решение задачи Коши для обыкновенного дифференциального уравнения (ОДУ) 1 порядка и оценить погрешность решения задачи.

$$\begin{cases} y'(t) = f(t, y(t)), \quad t \in [t_0, T] \\ y(t_0) = y_0 \end{cases} \qquad (1)$$

$$f(t, y) = -\frac{y}{t} + \sin t$$

$$t_0 = \pi, \quad T = \pi + 1, \quad y_0 = \frac{1}{\pi}$$

1.2 Порядок решения задачи

1. Задать исходные данные: функцию f правой части, начальное значение y_0 .
2. Написать функцию $eyler$, реализующую метод Эйлера, и с помощью этой функции найти приближенное решение задачи Коши с шагом $h = 0.1$ по явному методу Эйлера:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

3. Написать функцию $rkfixed$, реализующую метод Рунге-Кутты, и с ее помощью найти приближенное решение задачи Коши с шагом $h = 0.1$ по методу Рунге-Кутты 4 порядка точности:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = f(t_n, y_n)$$

$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$$

$$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$$

$$k_4 = f(t_n + h, y_n + hk_3)$$

4. Найти решение задачи Коши аналитически.
5. Построить таблицы значений приближенных и точного решений. На одном чертеже построить графики приближенных и точного решений.
6. Оценить погрешность приближенных решений двумя способами:
а) по формуле $\varepsilon = \max_{0<i<N} |y(t_i) - y_i|$; здесь $y(t_i)$ и y_i - значения точного и приближенного решений в узлах сетки $t_i, i = 1, ..., N$;
b) по правилу Рунге (по правилу двойного пересчета).

$$y(t_i) - y_i^{h/2} \approx \frac{y_i^{h/2} - y_i^h}{2^p - 1}$$

порядок метода p, $i = 1, \ldots, N$, t_i - узлы интегрирования
7. Выяснить, при каком значении шага $h = h^*$ решение, полученное по методу Эйлера, будет иметь такую же погрешность (см. п. 6а), как решение, полученное с помощью метода Рунге-Кутты с шагом $h = 0.1$.
УКАЗАНИЕ. В п. 7 рекомендуется провести серию вычислений решения по методу Эйлера, дробя шаг h пополам.

1.3 Аналитическое решение

$$\begin{cases} y'(t) = -y/t + \sin t, \quad t \in [\pi, \pi + 1] \\ y(1) = 1/\pi \end{cases}$$

Так как у нас линейное неоднородное ОДУ 1 порядка, решим сначала соответствующее ему однородное уравнение, а затем с помощью метода вариации постоянной и подставлении начальных условий получим решение задачи Коши

$$\frac{dy}{y} = -\frac{dt}{t}$$

$$\ln y = -\ln t + C$$

$$y = C_1 \frac{1}{t}$$

$$y = C_1(t) \frac{1}{t}$$

$$y' = C_1'(t) \frac{1}{t} - C_1(t) \frac{1}{t^2}$$

$$C_1'(t) \frac{1}{t} - C_1(t) \frac{1}{t^2} = -C_1(t) \frac{1}{t^2} + \sin t$$

$$C_1'(t) = t \sin t$$

$$C_1(t) = \int t \sin t = -t \cos t + \sin t + C_2$$

$$y(t) = -\cos t + \frac{\sin t + C_2}{t}$$

$$y(\pi) = 1 + \frac{C_2}{\pi} = \frac{1}{\pi} \quad \Rightarrow \quad C_2 = 1 - \pi$$

Аналитическое решение :

$$y(t) = -\cos t + \frac{\sin t + 1 - \pi}{t}$$

1.4 Код на Python

```
import numpy as np
import matplotlib.pyplot as plt

def f(t, y):
    return -y/t + np.sin(t)

def y_analytic(t):
    return -np.cos(t) + np.sin(t)/t + (1 - np.pi)/t

def eyler(f, y0, t0, T, h=0.1):
    t = np.arange(t0, T + h /2, h)
    y = np.empty((len(t),  1 if isinstance(y0, float) or isinstance(y0, int) else y0.s
    y[0, :] = y0
    for i, t_i in enumerate(t[:-1]):
        y[i + 1, :] = y[i, :] + h * f(t_i, y[i, :])
    return t, y
```

```
def rkfixed(f, y0, t0, T, h=0.1):
    a = [0, h / 2, h / 2, h]
    c = [h / 6, h / 3, h / 3, h / 6]
    B = [
        [h / 2, 0, 0],
        [0, h / 2, 0],
        [0, 0, h],
    ]
    t = np.arange(t0, T + h / 2, h)
    y = np.empty((len(t),  1 if isinstance(y0, float) or isinstance(y0, int) else y0.s
    y[0, :] = y0
    for i, t_i in enumerate(t[:-1]):
        k = [f(t_i + a[0], y[i])]
        for a_i, b_i in zip(a[1:], B):
            k.append(f(t_i + a_i, y[i, :] + sum(k_i * b_i_j for k_i, b_i_j in zip(k, b
        y[i + 1, :]= y[i, :] +  sum(c_i * k_i for c_i, k_i in zip(c, k))
    return t, y

def error_runge(method, p, f, y0, t0, T, h=0.1):
    _, yh = method(f, y0, t0, T, h)
    _, yh_2 = method(f, y0, t0, T, 2 * h)
    return np.max(np.abs(yh[::2] - yh_2)/(2 ** p - 1))


t0, T = np.pi, np.pi + 1
y0 = 1/np.pi
t_points = np.linspace(t0, T, 100)
y_points = y_analytic(t_points)
t_euler, y_euler = eyler(f, y0, t0, T)
y_euler = y_euler[:, 0]
t_rk, y_rk = rkfixed(f, y0, t0, T)
y_rk = y_rk[:, 0]
plt.plot(t_euler, y_euler,"-o", label="Euler")
plt.plot(t_rk, y_rk, "-o", label="Runge-Kutta")
plt.plot(t_points, y_points, label="analytic")
plt.legend()
eps_euler = np.max(np.abs(y_analytic(t_euler) - y_euler))
eps_rk = np.max(np.abs(y_analytic(t_euler) - y_rk))
print("Analytical solution")
with np.printoptions(4):
    print(f't = {t_euler}')
    print(f'y = {y_analytic(t_euler)}')
print('-------------------------')
print("Euler's method")
with np.printoptions(4):
    print(f't = {t_euler}')
    print(f'y = {y_euler}')
print('')
print(f"Error = {eps_euler:.4e}")
```

```
print(f"Runge error = {error_runge(eyler, 1, f, y0, t0, T):.4e}")
print('-------------------------------------------------------------------------------')
print("Analytical solution")
with np.printoptions(4):
    print(f't = {t_euler}')
    print(f'y = {y_analytic(t_euler)}')
print('-------------------------')
print(f"Runge-Kutta method (4th order)")
with np.printoptions(4):
    print(f't = {t_rk}')
    print(f'y = {y_rk}')
print('')
print(f"Error = {eps_rk:.4e}")
print(f"Runge error = {error_runge(rkfixed, 4, f, y0, t0, T):.4e}")
print('-------------------------')
h = 0.1
while eps_euler > eps_rk:
    h /= 2
    t_euler, y_euler = eyler(f, y0, t0, T, h)
    eps_euler = np.max(np.abs(y_analytic(t_euler) - y_euler[:, 0]))
print(f"With h* = {h:.4e} Euler's method reached error = {eps_euler:.4e}")
plt.xlabel("t")
plt.ylabel("y")
plt.show()
```

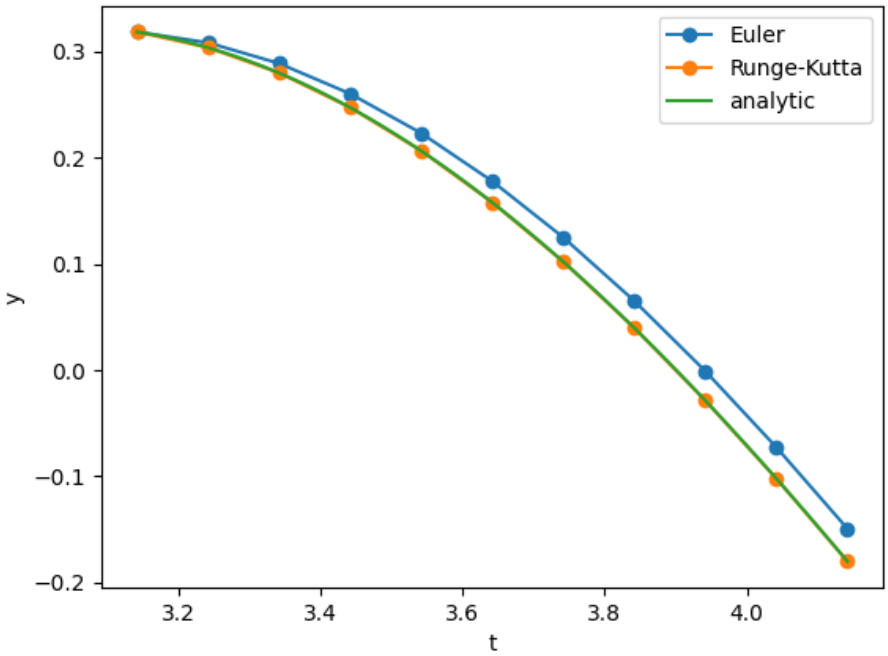
1.5 Таблицы значений

Аналитическое решение		Метод Эйлера		Метод Рунге-Кутта	
t	y	t	y	t	y
3.1416	0.3183	3.1416	0.3183	3.1416	0.3183
3.2416	0.3035	3.2416	0.3082	3.2416	0.3035
3.3416	0.2797	3.3416	0.2887	3.3416	0.2797
3.4416	0.2472	3.4416	0.2602	3.4416	0.2472
3.5416	0.2064	3.5416	0.2231	3.5416	0.2064
3.6416	0.1578	3.6416	0.1778	3.6416	0.1578
3.7416	0.1021	3.7416	0.125	3.7416	0.1021
3.8416	0.0397	3.8416	0.0652	3.8416	0.0397
3.9416	-0.0286	3.9416	-0.0009	3.9416	-0.0286
4.0416	-0.1021	4.0416	-0.0726	4.0416	-0.1021
4.1416	-0.18	4.1416	-0.1492	4.1416	-0.18

1.6 Результат работы программы

```
Analytical solution
t = [3.1416 3.2416 3.3416 3.4416 3.5416 3.6416 3.7416 3.8416 3.9416 4.0416
 4.1416]
y = [ 0.3183  0.3035  0.2797  0.2472  0.2064  0.1578  0.1021  0.0397 -0.0286
 -0.1021 -0.18  ]
------------------------
Euler's method
t = [3.1416 3.2416 3.3416 3.4416 3.5416 3.6416 3.7416 3.8416 3.9416 4.0416
 4.1416]
y = [ 0.3183  0.3082  0.2887  0.2602  0.2231  0.1778  0.125   0.0652 -0.0009
 -0.0726 -0.1492]

Error = 3.0799e-02
Runge error = 3.2387e-02
--------------------------------------------------------------------------------
Analytical solution
t = [3.1416 3.2416 3.3416 3.4416 3.5416 3.6416 3.7416 3.8416 3.9416 4.0416
 4.1416]
y = [ 0.3183  0.3035  0.2797  0.2472  0.2064  0.1578  0.1021  0.0397 -0.0286
 -0.1021 -0.18  ]
------------------------
Runge-Kutta method (4th order)
t = [3.1416 3.2416 3.3416 3.4416 3.5416 3.6416 3.7416 3.8416 3.9416 4.0416
 4.1416]
y = [ 0.3183  0.3035  0.2797  0.2472  0.2064  0.1578  0.1021  0.0397 -0.0286
 -0.1021 -0.18  ]

Error = 1.5638e-08
Runge error = 1.5509e-08
------------------------
With h* = 4.7684e-08 Euler's method reached error = 1.2640e-08
```

1.7 Графики приближенных и точного решений



1.8 Вывод

В программе были реализованы методы Эйлера и Рунге-Кутты (4 порядка) с шагом $h = 0.1$ для поиска приблеженного решения задачи Коши. Также решения были найдены аналитически. Для каждого из приближенных решений было произведено сравнение с аналитическим решением двумя способами: по правилу Рунге и формуле $\varepsilon = \max_{0<i<N} |y(t_i) - y_i|$; здесь $y(t_i)$ и y_i - значения точного и приближенного решений в узлах сетки $t_i, i = 1, ..., N$.

Отклонение от аналитического решения для метода Эйлера составило 3.2387×10^{-2} и 3.0799×10^{-2} соответственно. В то же время отклонение от аналитического решения для метода Рунге-Кутты (4 порядка) составило 1.5509×10^{-8} и 1.5638×10^{-8} соответственно.

В сравнении с методом Рунге-Кутты (4 порядка) метод Эйлера нашел не очень точное приближение, поэтому для того, чтобы достичь точности 1.2640×10^{-8} как в методе Рунге-Кутты (4 порядка) понадобилось уменьшить шаг до $h^* = 4.7684 \times 10^{-8}$.

Кроме того были построены графики приближенных и точного решений, на которых также видно, что у метода Эйлера достаточно большое отклонение, в то время как у метода Рунге-Кутты (4 порядка) решение накладывается на аналитическое.

2 Задача Коши для ОДУ 3 порядка

2.1 Формулировка задачи

Решить приближенно задачу Коши для ОДУ 3 порядка на отрезке $[A, B]$, используя метод Рунге-Кутты 4 порядка с шагами $h = 0.1$ и $h = 0.05$ для систем ОДУ 1 порядка. Оценить погрешность по правилу Рунге. Построить график решения, найденного с шагом $h = 0.05$.
УКАЗАНИЕ. Эквивалентная задача Коши для системы ОДУ 1 порядка имеет вид:

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ y_3' = (f(t) - a_1 y_3 - a_2 y_2 - a_3 y_1)/a_0 \\ y_1(A) = b_1 \quad y_2(A) = vb_2 \quad y_3(A) = b_3 \end{cases}$$

$$\begin{cases} a_0 y''' + a_1 y'' + a_2 y' + a_3 y = f(t) \\ y(A) = b_1, y'(A) = b_2, y''(A) = b_3 \end{cases}$$

$$A = 0, \quad B = 1.5, \quad b_1 = 1, \quad b_2 = 2.5, \quad b_3 = 6$$

$$a_0 = 1, \quad a_1 = -1.4, \quad a_2 = 0.64, \quad a_3 = 41.52, \quad f(t) = \cos 2t + 3t + 1$$

2.2 Код на Python

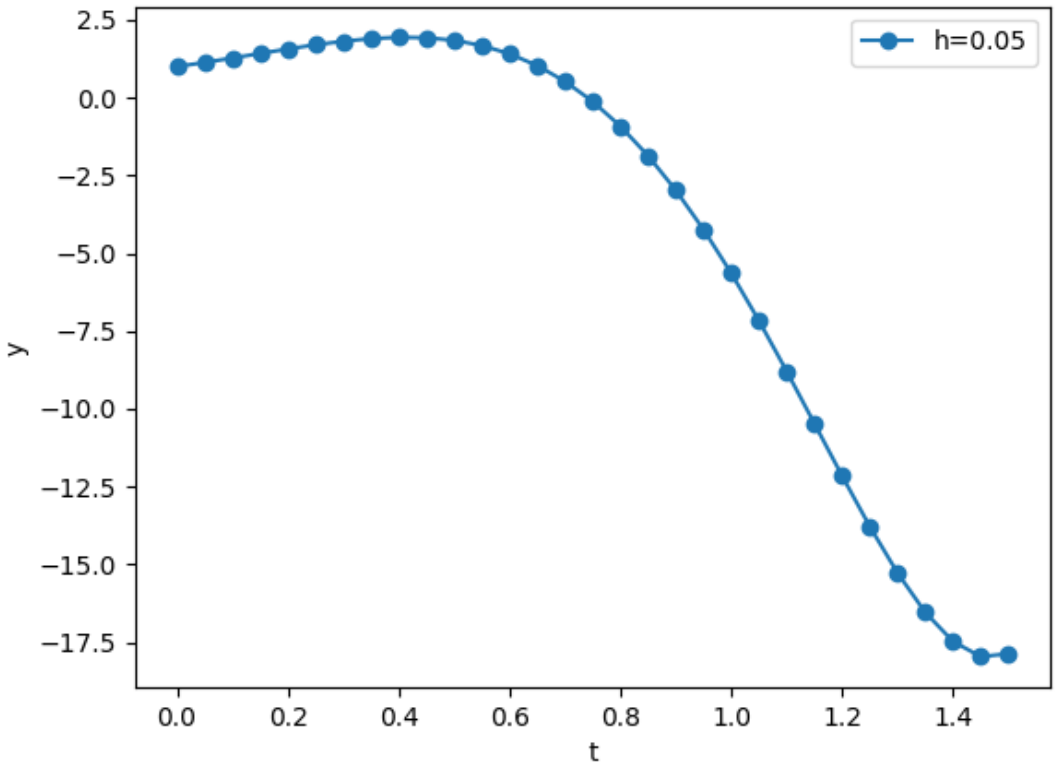
```
import numpy as np
import matplotlib.pyplot as plt
a0 = 1
a1 = -1.4
a2 = 0.64
a3 = 41.52

def f(t):
    return np.cos(2* t) + 3*t + 1
```

```
def F(t, y):
    return np.array(
        [
            y[1],
            y[2],
            (f(t) - a1 * y[2] - a2 * y[1] - a3 * y[0])/ a0
        ]
    )

def rkfixed(f, y0, t0, T, h=0.1):
    a = [0, h / 2, h / 2, h]
    c = [h / 6, h / 3, h / 3, h / 6]
    B = [
        [h / 2, 0, 0],
        [0, h / 2, 0],
        [0, 0, h],
    ]
    t = np.arange(t0, T + h / 2, h)
    y = np.empty((len(t),  1 if isinstance(y0, float) or isinstance(y0, int) else y0.s
    y[0, :] = y0
    for i, t_i in enumerate(t[:-1]):
        k = [f(t_i + a[0], y[i])]
        for a_i, b_i in zip(a[1:], B):
            k.append(f(t_i + a_i, y[i, :] + sum(k_i * b_i_j for k_i, b_i_j in zip(k, l
        y[i + 1, :]= y[i, :] +  sum(c_i * k_i for c_i, k_i in zip(c, k))
    return t, y



A = 0
B = 1.5
b1 = 1
b2 = 2.5
b3 = 6
y0 = np.array([b1, b2, b3])
t_h, y_h = rkfixed(F, y0, A, B, 0.1)
t_h_2, y_h_2 = rkfixed(F, y0, A, B, 0.05)
plt.plot(t_h_2, y_h_2[:, 0], "-o", label="h=0.05")
print(f"Runge error = {np.max(np.abs(y_h - y_h_2[::2]))/(2**4-1):.4e}")
plt.legend()
plt.xlabel("t")
plt.ylabel("y")
plt.show()
```

2.3 Результат работы программы

```
Runge error = 1.1183e-02
```

2.4 График решения задачи Коши с $h = 0.05$



2.5 Вывод

В задаче представлено решение системы методом Рунге-Кутты (4 порядка) с $h = 0.1$ и $h = 0.05$ и посчитана погрешность по правилу Рунге: 1.1183×10^{-2}. Для поиска решения задачи Коши 3 порядка удобно использовать численные метод решения, так как поиск аналитического решения будет достаточно сложным. Также построен график решения системы методом Рунге-Кутты (4 порядка) с $h = 0.05$.

3 Определение жесткой задачи Коши

3.1 Формулировка задачи

Даны две задачи Коши для систем ОДУ 1 порядка с постоянными коэффициентами на отрезке $[0, 1]$, где A и B – заданные матрицы, Y_0 , Z_0 - заданные векторы. Выяснить, какая из задач является жесткой.

$$Y'(t) = AY(t), \quad Y(0) = Y_0$$

$$Z'(t) = BZ(t), \quad Z(0) = Z_0$$

$$A = \begin{pmatrix} -0.717 & -23.827 \\ 114.483 & -640.393 \end{pmatrix}$$

$$B = \begin{pmatrix} -1.905 & -0.015 \\ -0.13 & -2.295 \end{pmatrix}$$

$$Y_0 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$Z_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

3.2 Порядок решения задачи

1. Составить программу-функцию нахождения решения системы ОДУ 1 порядка с постоянными коэффициентами по явному методу Эйлера. Используя составленную программу, решить обе задачи с шагом $h = 0.01$. Определить, для какой из задач явный метод неустойчив при данном шаге h.
2. Используя встроенную функцию для нахождения собственных чисел матриц A и B, найти коэффициенты жесткости обеих систем. Какая из задач является жесткой?
3. Для жесткой задачи теоретически оценить шаг h^*, при котором явный метод Эйлера будет устойчив.
4. Составить программу-функцию нахождения решения системы ОДУ 1 порядка с постоянными коэффициентами по неявному методу Эйлера. Используя составленную программу, найти решение жесткой задачи с шагом $h = 0.01$. Построить графики компонент полученного решения.

$$\frac{y_{n+1} - y_n}{h} = \frac{f(t_n, y_n) - f(t_{n+1}, y_{n+1})}{2}$$

5. Для жесткой задачи экспериментально подобрать шаг h, при котором графики компонент решения, полученного по явному методу Эйлера, визуально совпадают с графиками компонент решения, полученного по неявному методу с шагом $h = 0.01$. Сравнить найденное значение шага с шагом h^*. Объяснить различие поведения явного и неявного методов Эйлера при решении жесткой задачи.

3.3 Код на Python

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve

A = np.array([[-0.717, -23.827], [114.483, -640.393]], dtype=np.double)
Y0 = np.array([1., 2.], dtype=np.double)
B = np.array([[-1.905, -0.015], [-0.13, -2.295]], dtype=np.double)
Z0 = np.array([1., 0.], dtype=np.double)

def eyler(f, y0, t0, T, h=0.1):
    t = np.arange(t0, T + h /2, h)
    y = np.empty((len(t),  y0.shape[0] if isinstance(y0, np.ndarray) else 1), dtype=np
    y[0, :] = y0
    for i, t_i in enumerate(t[:-1]):
        y[i + 1, :] = y[i, :] + h * f(y[i, :])
    return t, y

def implicit_eyler(f, y0, t0, T, h=0.1):
```

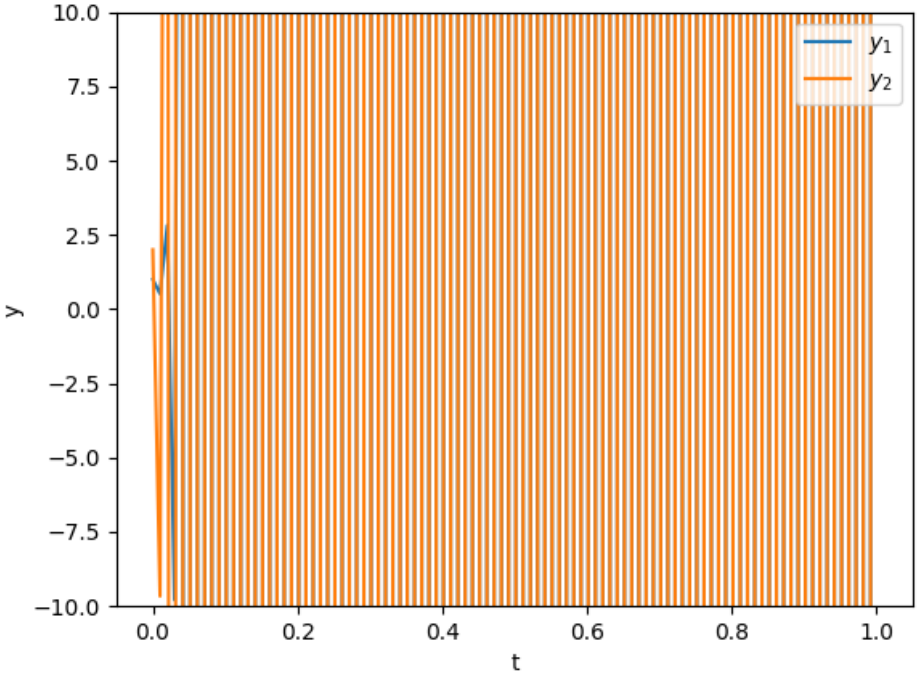
```
    t = np.arange(t0, T + h /2, h)
    y = np.empty((len(t),  y0.shape[0] if isinstance(y0, np.ndarray) else 1), dtype=np
    y[0, :] = y0
    for i, t_i in enumerate(t[:-1]):
        y[i + 1, :] = fsolve(lambda yn: (yn - y[i, :])/h - f(y[i, :])/2 -f(yn)/2, y[i
    return t, y

def f1(y):
    return A @ y

def f2(y):
    return B @ y

t0 = 0
T = 1
h = 0.01
t_euler_A, y_euler_A = eyler(f1, Y0, t0, T, h)
t_euler_B, y_euler_B = eyler(f2, Z0, t0, T, h)
plt.plot(t_euler_A, y_euler_A[:, 0], label = "$y_1$")
plt.plot(t_euler_A, y_euler_A[:, 1], label = "$y_2$")
plt.ylim(-10, 10)
plt.legend()
plt.xlabel("t")
plt.ylabel("y")
plt.show()
plt.plot(t_euler_B, y_euler_B)
plt.plot(t_euler_B, y_euler_B[:, 0], label = "$z_1$")
plt.plot(t_euler_B, y_euler_B[:, 1], label = "$z_2$")
plt.ylim(-10, 10)
plt.legend()
plt.xlabel("t")
plt.ylabel("z")
plt.show()
l_A = np.linalg.eigvals(A)
l_B = np.linalg.eigvals(B)
with np.printoptions(5):
    print("Eigenvalues of A :", l_A)
    a_stiff = np.max(np.abs(l_A))/np.min(np.abs(l_A))
    print(f"Stiffness coef of A : {a_stiff:.5f}")
    print('---------------------')
    print("Eigenvalues of B :", l_B)
    b_stiff = np.max(np.abs(l_B))/np.min(np.abs(l_B))
    print(f"Stiffness coef of B : {b_stiff:.5f}")
print('---------------------')
print("A is stiffer" if a_stiff > b_stiff else "B is stiffer")
print(f"h* theoretical: {2/np.max(np.abs(l_A)):.5f}")
print(f"h visial: {h/3.7:.5f}")
t_ieuler_A, y_ieuler_A = implicit_eyler(f1, Y0, t0, T, h)
plt.plot(t_ieuler_A, y_ieuler_A[:, 0], label = "$y_1$")
plt.plot(t_ieuler_A, y_ieuler_A[:, 1], label = "$y_2$")
```

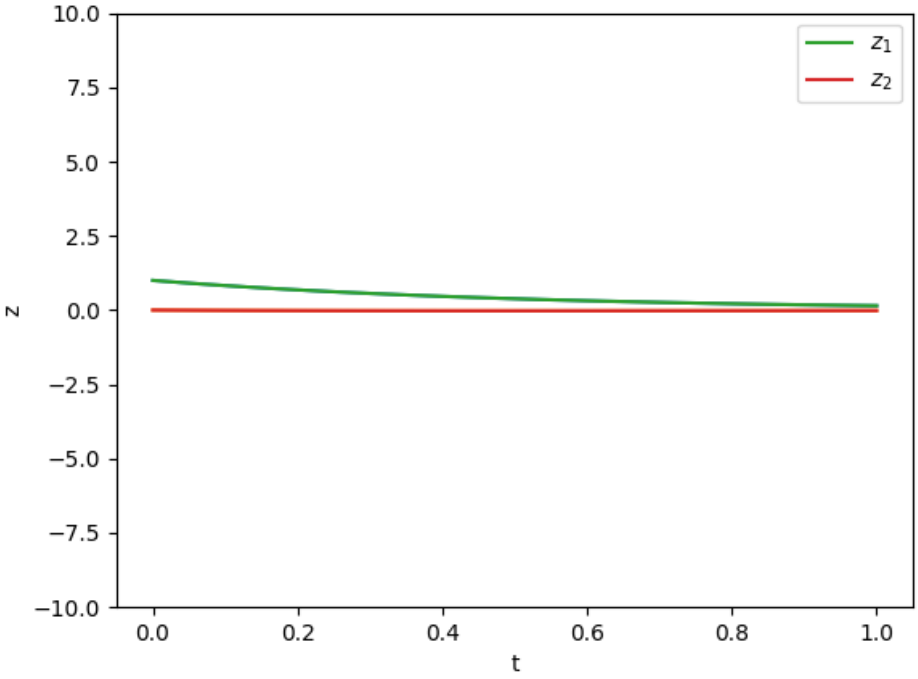
```
plt.legend()
plt.xlabel("t")
plt.ylabel("y")
plt.show()
t_euler_A, y_euler_A = eyler(f1, Y0, t0, T, h/3.7)
plt.plot(t_euler_A, y_euler_A[:, 0], label = "$y_1$")
plt.plot(t_euler_A, y_euler_A[:, 1], label = "$y_2$")
plt.legend()
plt.xlabel("t")
plt.ylabel("y")
plt.show()
```

3.4 Результат работы программы

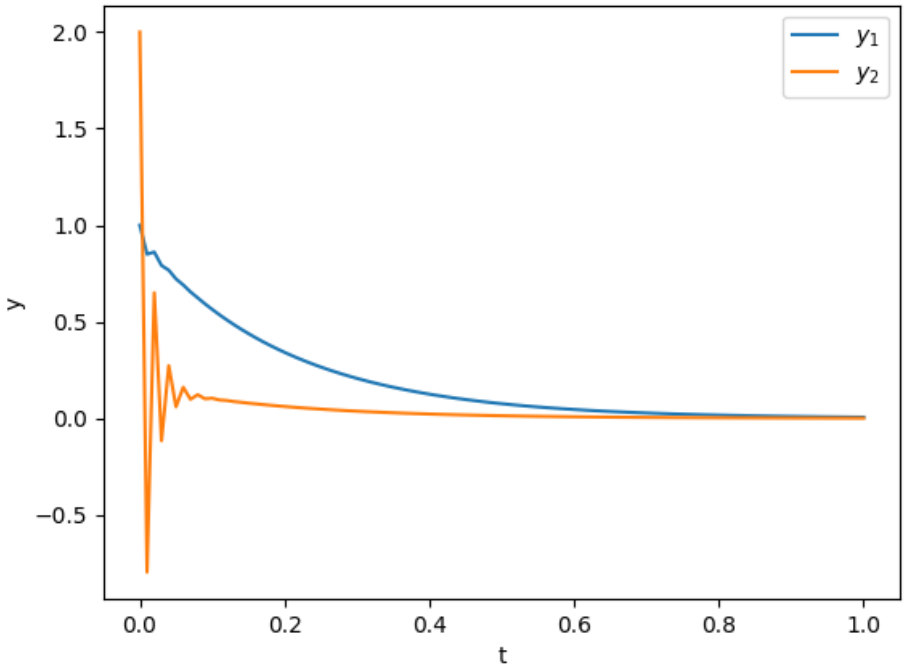
```
Eigenvalues of A : [  -5.01014 -636.09986]
Stiffness coef of A : 126.96254
---------------------
Eigenvalues of B : [-1.90006 -2.29994]
Stiffness coef of B : 1.21045
---------------------
A is stiffer
h* theoretical: 0.00314
h visial: 0.00270
```

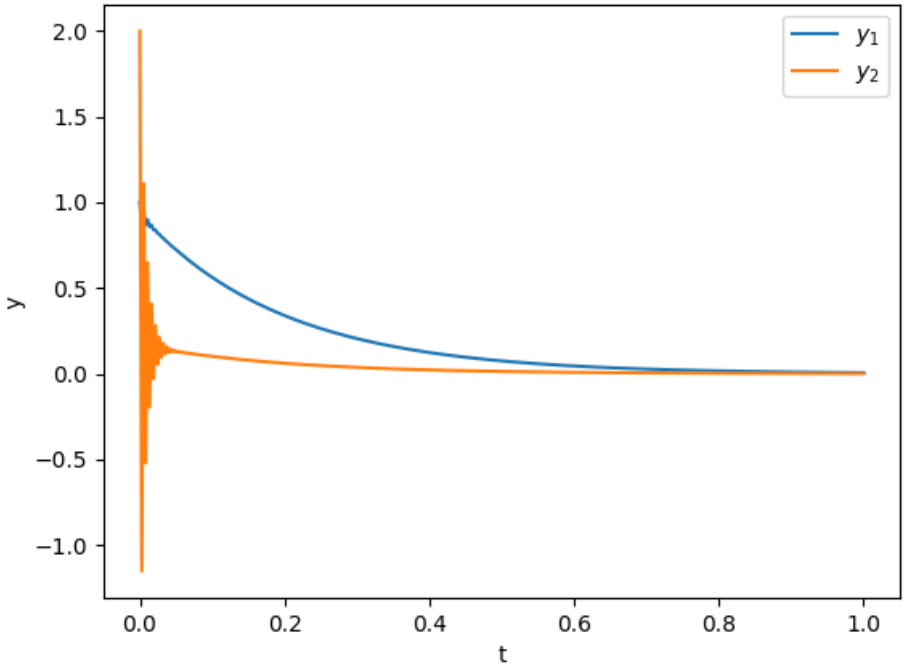
3.5 Графики решения системы ОДУ 1 порядка с постоянными коэффициентами по явному методу Эйлера для двух задач





3.6 Графики решения системы ОДУ 1 порядка с постоянными коэффициентами по неявному и явному методам Эйлера для жесткой задачи





3.7 Вывод

В работе представлены реализации явного и неявного методов Эйлера нахождения решения системы ОДУ 1 порядка с постоянными коэффициентами и шагом $h = 0.01$.

Было дано две задачи с соответствующими матрицами A и B. Сначала было проверено, для какой из задач явный метод неустойчив при данном шаге h. Это видно из приведенных выше графиков: для задачи с матрицей A метод вляется неустойчивым.

Затем были найдены собственные числа матриц (через встроенные функции) и посчитаны коэффициенты жесткости обеих систем S, где

$$S = \max_i |Re\lambda_i| / \min_i |Re\lambda_i|,$$

$\lambda_i, \quad i = 1, \ldots, n$, - собственные числа матрицы порядка n. Задача A оказалась жесткой ($126.96254 >> 1$). Жёсткой системой ОДУ называется такая система, численное решение которой явными методами является неудовлетворительным из-за резкого увеличения числа вычислений (при малом шаге интегрирования) или из-за резкого возрастания погрешности при недостаточно малом шаге. Для жёстких систем характерно то, что для них неявные методы дают лучший результат, чем явные методы.

Для нее был теоретически оценен шаг $h^* = 0.00314$, при котором явный метод Эйлера будет устойчив по формуле: $h \leq 2/\max_i |Re\lambda_i|, \quad \lambda_i, \quad i = 1, \ldots, n$, - собственные числа матрицы порядка n.

Далее с помощью программы-функции нахождения решения системы ОДУ 1 порядка с постоянными коэффициентами по неявному методу Эйлера было найдено решение жесткой задачи с шагом $h = 0.01$ и построены соответствующие графики. Затем был произведен экспериментальный подбор шага h, при котором графики компонент решения, полученного по явному методу Эйлера, визуально совпадают с графиками компонент решения, полученного по неявному методу с шагом $h = 0.01$. Значение подобранного параметра: $h = 0.00270$, что немного отличается от найденного теоретически значения.