

Задача 7.1. Дана задача Коши для системы двух обыкновенных дифференциальных уравнений

$$\begin{cases} u' = f_0(t, u, v) \\ v' = f_1(t, u, v) \end{cases} \quad u(0) = 1, \quad v(0) = 1$$

Вариант 1:

$$\begin{cases} u' = u + t \\ v' = u \cdot v - 1 \end{cases}$$

Решение задачи:

1. Модифицировать программу решения задачи по явному методу Эйлера из лабораторной работы 6 для решения системы дифференциальных уравнений. Найти приближенное решение задачи Коши с шагом $h=0.01$ на отрезке $[0,1]$. Оценить величину погрешности по правилу Рунге.

формула метода Эйлера:

$$y_{i+1} = y_i + hf(t_i, y_i)$$

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

#7.1
a = 0
b = 1
h = 0.01
n = round((b-a)/h)
u0 = 1
v0 = 1
t0 = 0
t = np.linspace(a, b, n + 1)

def func_du(u, v, t):
    return u + t

def func_dv(u, v, t):
    return u*v - 1

#-----метод Эйлера-----
def Euler(t0, u0, v0, h, n, func_du, func_dv):
    vec_u, vec_v = [u0], [v0]
    buft = t0
    for i in range(1, n + 1):
        vec_u.append(vec_u[i - 1] + h * func_du(vec_u[i - 1], vec_v[i - 1], buft))
        vec_v.append(vec_v[i - 1] + h * func_dv(vec_u[i - 1], vec_v[i - 1], buft))
        buft += h
    return np.array(vec_u), np.array(vec_v)

def ErrorRunge(y_2h, y_1h, p):
    buflist = []
    N = round(n/2)
    for i in range(0, N):
        buflist.append(abs(y_2h[i] - y_1h[2*i]) / (2**p - 1))
    max_number = max(buflist)
    return max_number

u_1h, v_1h = Euler(t0, u0, v0, h, n, func_du, func_dv)
u_2h, v_2h = Euler(t0, u0, v0, 2*h, round(n/2), func_du, func_dv)
```

```

print("-----")
p = 1
print("метод Эйлера")
print("n = ", n)
print("h = ", h)
err_u = ErrorRunge(u_2h, u_1h, p)
print("Погрешность по правилу Рунге для u: ", err_u)
err_v = ErrorRunge(v_2h, v_1h, p)
print("Погрешность по правилу Рунге для v: ", err_u)

```

Результат:

```

-----
метод Эйлера
n = 100
h = 0.01
Погрешность по правилу Рунге для u: 0.02541303630741787
Погрешность по правилу Рунге для v: 0.11144303619077212
-----

```

2. Модифицировать программу решения задачи по индивидуальному варианту из лабораторной работы 6 для решения системы дифференциальных уравнений. Найти приближенное решение задачи Коши с шагом $h=0.01$ на отрезке $[0,1]$. Оценить величину погрешности по правилу Рунге.

Индивидуальный метод:

метод Эйлера-Коши:

$$\bar{y}_{i+1} = y_i + hf(t_i, y_i)$$

$$y_{i+1} = y_i + \frac{h}{2} (f(t_i, y_i) + f(t_{i+1}, \bar{y}_{i+1}))$$

```

def Euler_Cauchy(t0, u0, v0, h, n, func_du, func_dv):
    u_solution = np.zeros(n + 1)
    v_solution = np.zeros(n + 1)
    u_solution[0] = u0
    v_solution[0] = v0
    u_buf = np.copy(u_solution)
    v_buf = np.copy(v_solution)
    buft = t0
    for i in range(1, n + 1):
        u_buf[i] = u_solution[i - 1] + h * func_du(u_solution[i - 1], v_solution[i - 1], buft)
        v_buf[i] = v_solution[i - 1] + h * func_dv(u_solution[i - 1], v_solution[i - 1], buft)

        u_solution[i] = u_solution[i - 1] + 0.5 * h * (func_du(u_solution[i - 1], v_solution[i - 1], buft)
                                                    + func_du(u_buf[i], v_buf[i], buft + h))
        v_solution[i] = v_solution[i - 1] + 0.5 * h * (func_dv(u_solution[i - 1], v_solution[i - 1], buft)
                                                    + func_dv(u_buf[i], v_buf[i], buft + h))

        buft += h
    return u_solution, v_solution

print("-----")

u_1h_Euler_Cauchy, v_1h_Euler_Cauchy = Euler_Cauchy(t0, u0, v0, h, n, func_du, func_dv)
u_2h_Euler_Cauchy, v_2h_Euler_Cauchy = Euler_Cauchy(t0, u0, v0, 2*h, round(n/2), func_du, func_dv)

```

```

print("метод Эйлера-Коши")
print("n = ", n)
print("h = ", h)

p = 2
err_u = ErrorRunge(u_2h_Euler_Cauchy, u_1h_Euler_Cauchy, p)
print("Погрешность по правилу Рунге для u: ", err_u)
err_v = ErrorRunge(v_2h_Euler_Cauchy, v_1h_Euler_Cauchy, p)
print("Погрешность по правилу Рунге для v: ", err_v)
fig, axs = plt.subplots(1, 2, figsize = (10,5))

```

Результат:

```

-----
метод Эйлера-Коши
n = 100
h = 0.01
Погрешность по правилу Рунге для u: 8.55253314406094e-05
Погрешность по правилу Рунге для v: 0.0008189266693716668

```

3. На одном чертеже построить графики первой компоненты $u(t)$ найденного обоими методами решения, а на другом - графики второй компоненты $v(t)$ найденного обоими методами решения.

```

fig, axs = plt.subplots(1, 2, figsize = (10,5))

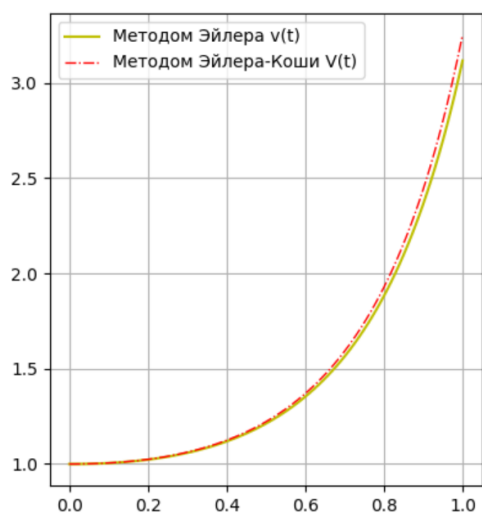
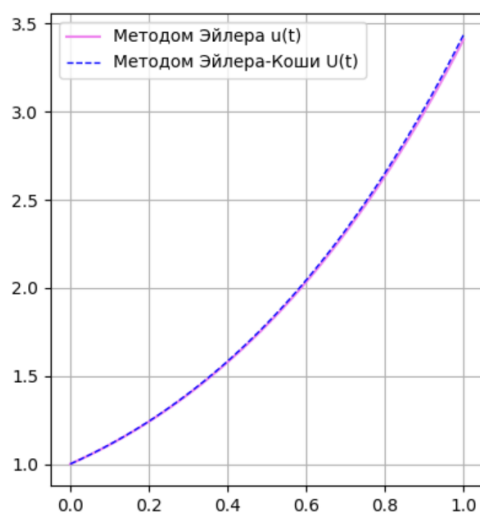
axs[0].plot(t, u_1h, label='Методом Эйлера u(t)', color = 'violet')
axs[0].plot(t, u_1h_Euler_Cauchy, label='Методом Эйлера-Коши U(t)', color = 'b', linestyle='dashed', linewidth=1)
axs[0].grid()
axs[0].legend()

axs[1].plot(t, v_1h, label='Методом Эйлера v(t)', color = 'y')
axs[1].plot(t, v_1h_Euler_Cauchy, label='Методом Эйлера-Коши V(t)', color = 'red', linestyle='dashdot', linewidth=1)
axs[1].grid()
axs[1].legend()

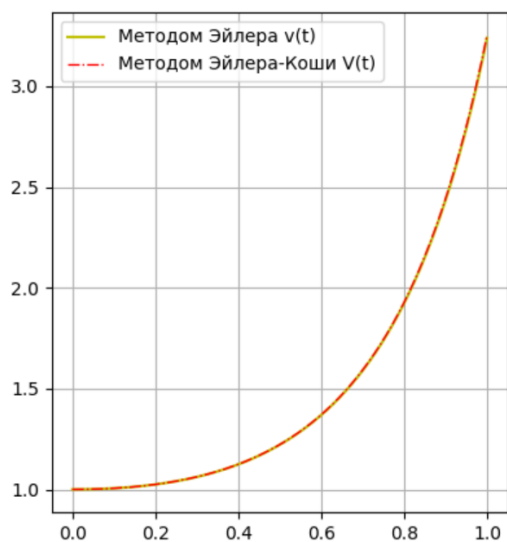
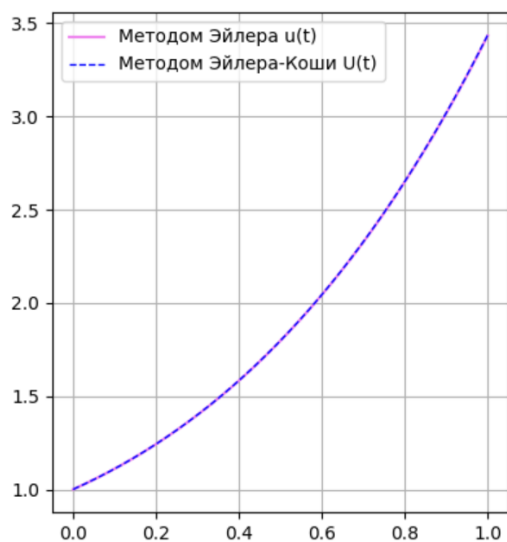
plt.show()

```

При шаге $h = 0.01$



При $h = 0.001$:



4. Сравнить полученные результаты.

При шаге $h = 0.01$ мы видим, что для $u(t)$ графики совпали, а для $v(t)$ практически совпали. При уменьшении шага, графики совпадут. Это значит, что приближенное решение задачи Коши с шагом $h=0.01$ на отрезке $[0, 1]$ для метода Эйлера и Эйлера-Коши совпадают. Погрешность по правилу Рунге для соответствующих компонент метода Эйлера-Коши много меньше, чем для метода Эйлера, так и должно быть согласно теории (так как метод Эйлера-Коши имеет более высокий порядок точности).

Задача 7.2. Дана задача Коши для двух систем обыкновенных дифференциальных уравнений с постоянными коэффициентами

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

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

где A и B – заданные матрицы, Y_0, Z_0 – заданные векторы. Исследовать поведение решения систем уравнений

N	A	Y_0	B	Z_0
7.2.1	$\begin{pmatrix} -269.264 & -47.169 & -33.887 \\ -56.642 & -18.312 & 115.518 \\ 12.843 & -119.698 & -8.424 \end{pmatrix}$	$\begin{pmatrix} 1.6 \\ 4.4 \\ 2.4 \end{pmatrix}$	$\begin{pmatrix} -11.485 & 63.977 & -62.832 \\ -64.673 & -11.25 & 86.831 \\ 62.115 & -87.346 & -11.265 \end{pmatrix}$	$\begin{pmatrix} 0.8 \\ 4.4 \\ 1.6 \end{pmatrix}$

Решение задачи

1. Используя встроенную функцию пакета NUMPY для нахождения собственных чисел матриц A и B , найти коэффициенты жесткости обеих систем. Установить какая задача является жесткой.

```
import numpy as np
A = np.array([
    [-269.264, -47.169, -33.887],
    [-56.642, -18.312, 115.518],
    [12.843, -119.698, -8.424],
])
Y0 = np.array([1.6, 4.4, 2.4])
B = np.array([
    [-11.485, 63.977, -62.832],
    [-64.673, -11.25, 86.831],
    [62.115, -87.346, -11.265],
])
Z0 = np.array([0.8, 4.4, 1.6])

A_lambda = np.linalg.eig(A)[0]
print("Массив из собственных чисел матрицы A:", A_lambda)

B_lambda = np.linalg.eig(B)[0]
print("Массив из собственных чисел матрицы B:", B_lambda)
```

```
Массив из собственных чисел матрицы A: [-279.99984898 +0.j          -8.00007551+119.99999717j
      -8.00007551-119.99999717j]
Массив из собственных чисел матрицы B: [-11.00004638+125.00020264j -11.00004638-125.00020264j
      -11.99990724 +0.j          ]
```

Так как $Re\lambda_i < 0$ для всех собственных чисел матрицы, то определяем жесткость системы

С помощью формулы $s = \frac{|Re\lambda_i|}{|Re\lambda_i|}$

```
def system_stiffness(x):
    return np.max(np.abs(x.real))/np.min(np.abs(x.real))
print("Жесткость системы A:", system_stiffness(A_lambda.real))
print("Жесткость системы B:", system_stiffness(B_lambda.real))
```

Результат:

```
Жесткость системы A: 34.99965076468353
Жесткость системы B: 1.0908960586817704
```

Так как число жесткости системы A $s = 34.99965076468353 \gg 1$, то система A жесткая.

Так как число жесткости системы B $s = 1.0908960586817704$ немного больше 1, то система B не является жесткой.

2. Численно решить обе задачи на отрезке $[0,1]$ с шагом $h=0.01$ явным методом Эйлера. Определить, для какой из задач явный метод неустойчив при данном шаге h . Построить графики компонент полученного решения.

```
#-----метод Эйлера-----
def system_Euler(matrix, y0, h, n):
    y_t = [y0]
    for i in range(1, n + 1):
        y_t.append(y_t[i - 1] + h * (matrix @ y_t[i - 1]))
    print(y_t)
    return np.array(y_t)

def output(arr, t):
    fig, axes = plt.subplots(3, 1, figsize = (21, 10))
    #print("00", arr[0:, 0])
    axes[0].plot(t, arr[0:, 0], label='1 компонента', color="r")
    axes[0].legend()
    #print("01", arr[0:, 1])
    axes[1].plot(t, arr[0:, 1], label='2 компонента', color="g")
    axes[1].legend()

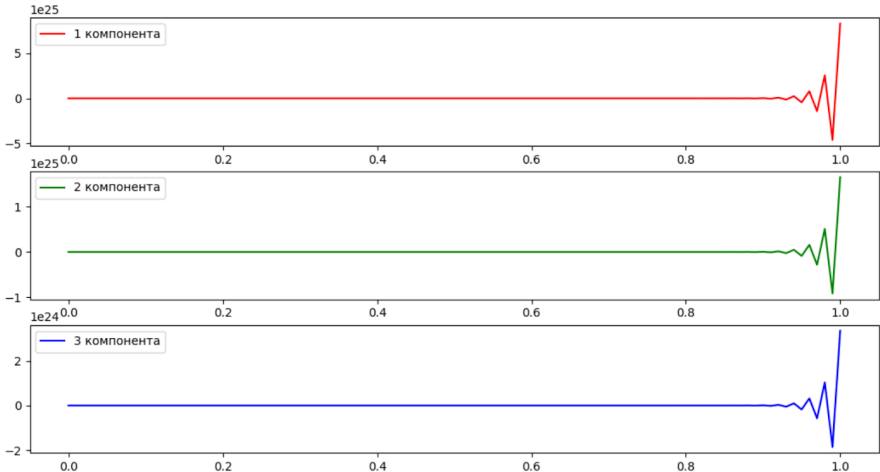
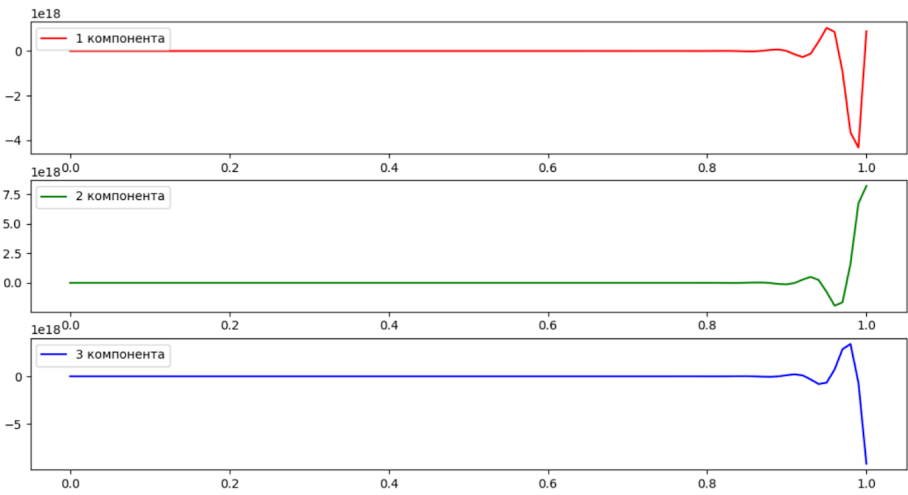
    axes[2].plot(t, arr[0:, 2], label='3 компонента', color="b")
    axes[2].legend()

    plt.show()

t = np.linspace(a, b, n + 1)
AY = system_Euler(A, Y0, h, n)
print(AY)
output(AY, t)
#print(AY.shape)

BZ = system_Euler(B, Z0, h, n)
print(BZ)
output(BZ, t)
```

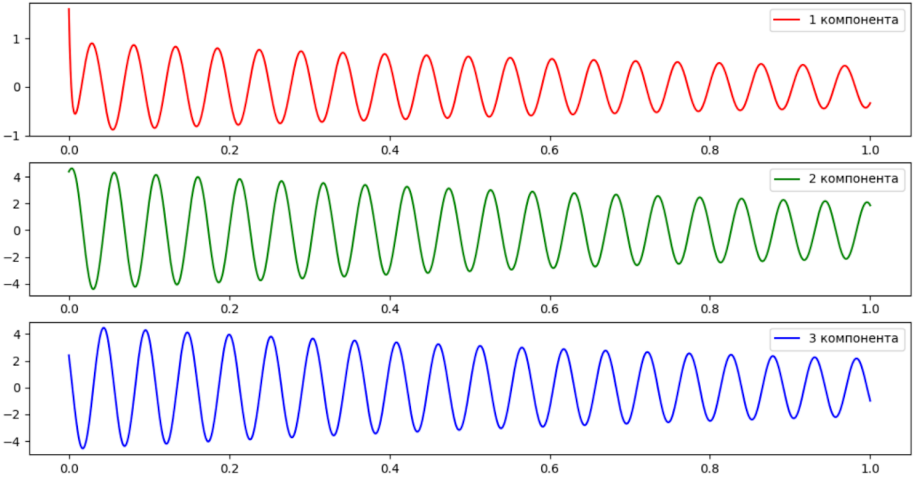
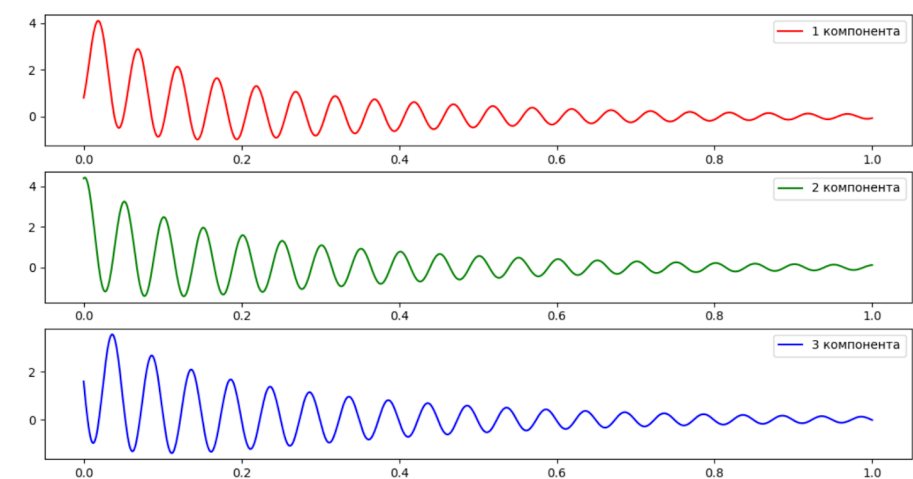
метод Эйлера		
шаг	Задача	График

$h=0.01$	$Y'(t) = AY(t),$ $Y(0) = Y_0$	
$h=0.01$	$Z'(t) = BZ(t),$ $Z(0) = Z_0$	

На графиках наблюдаем, что для обеих задач явный метод Эйлера неустойчив при данном шаге $h=0.01$

В качестве вычислительного эксперимента, уменьшим шаг.

метод Эйлера		
шаг	Задача	График

h=0.001	$Y'(t) = AY(t),$ $Y(0) = Y_0$	
h=0.001	$Z'(t) = BZ(t),$ $Z(0) = Z_0$	

Таким образом, видно, что при уменьшении шага (при $h=0.001$) метод сходится. Значит, изначально не хватало аппроксимации, шаг $h=0.01$ достаточно большой.

3. Численно решить обе задачи на отрезке $[0,1]$ с шагом $h=0.01$ по индивидуальному варианту из лабораторной работы. Определить, для какой из задач к $h=0.01$. Построить графики компонент полученного решения.


```

#-----метод Эйлера-Коши-----
def system_Euler_Cauchy(matrix, y0, h, n):
    buf_y = [y0]
    y_t = [y0]

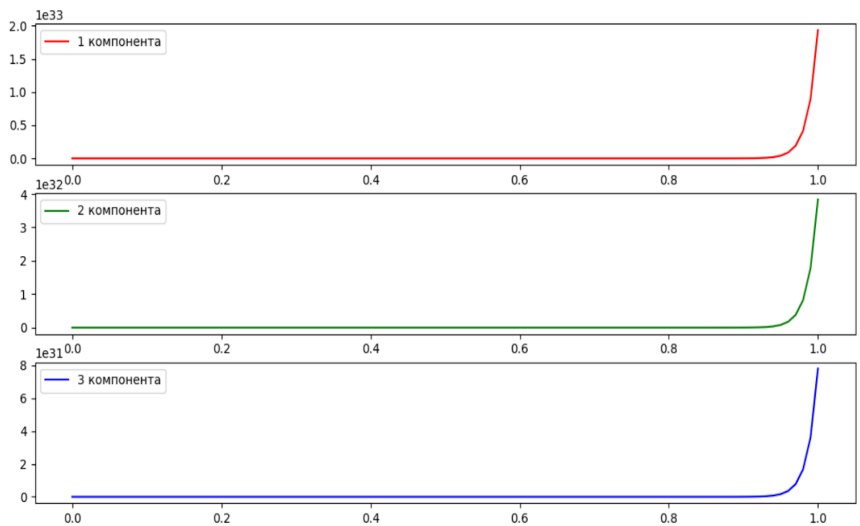
    for i in range(1, n + 1):
        buf_y.append(y_t[i - 1] + h * (matrix @ y_t[i - 1]))
        y_t.append(y_t[i - 1] + 0.5 * h * (matrix @ buf_y[i-1]))
    return np.array(y_t)

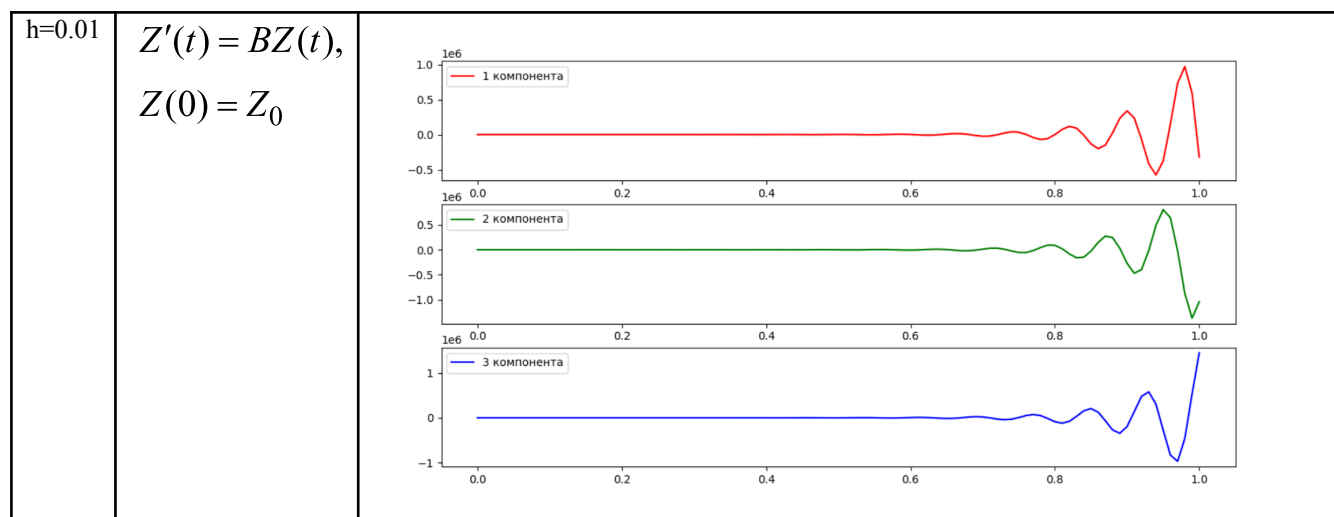
AY = system_Euler_Cauchy(A, Y0, h, n)
output(AY, t)

BZ = system_Euler_Cauchy(B, Z0, h, n)
output(BZ, t)

```

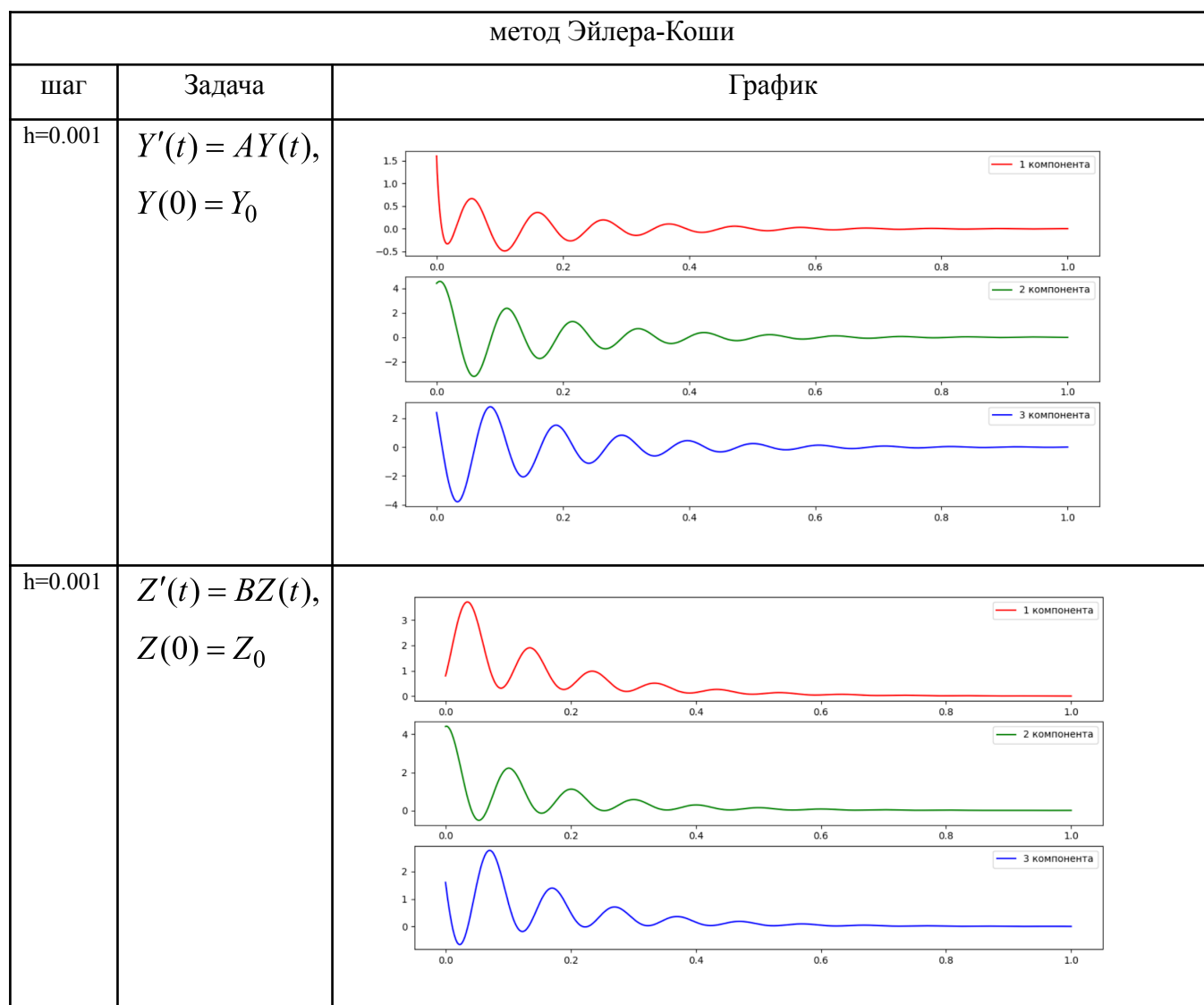
Графики для задачи $Y'(t) = AY(t), Y(0) = Y_0$

метод Эйлера-Коши		
шаг	Задача	График
h=0.01	$Y'(t) = AY(t),$ $Y(0) = Y_0$	



Оба метода неустойчивы на данном шаге, так как шаг $h=0.01$ достаточно мал.

Уменьшим шаг, $h=0.001$:



При уменьшении шага мы видим, что получено решение, оно находится в начале временного промежутка, в конце временного промежутка наблюдается стабилизация, то есть график устроен как график константы. Хотя задача A является жесткой, с числом жесткости $s = 34.99965076468353$, и метод Эйлера-Коши является явным, как и метод Эйлера в пункте 2, уменьшение шага приводит к тому, что метод устойчив при $h = 0.001$.

Теория гласит, что при решении жестких задач лучше применять неявные методы.

Применим неявный метод Эйлера.

$$Y_{i+1} = Y_i + hAY_{i+1}$$

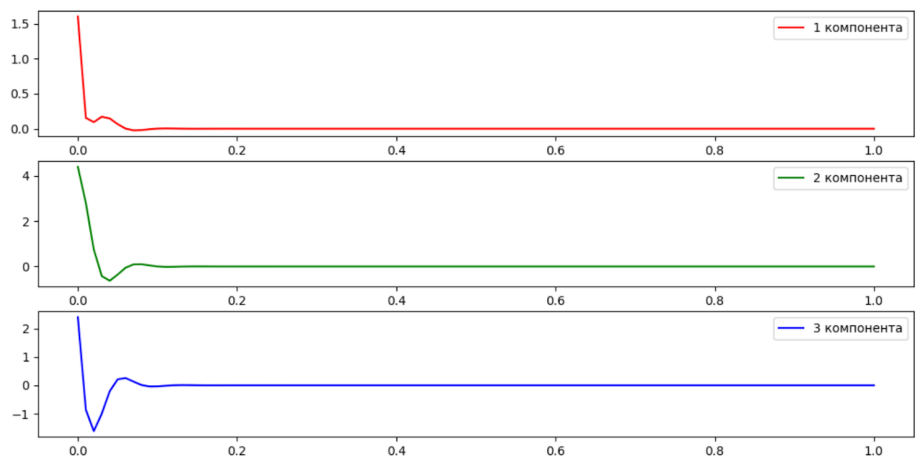
$$Y_{i+1}(E - hA) = Y_i$$

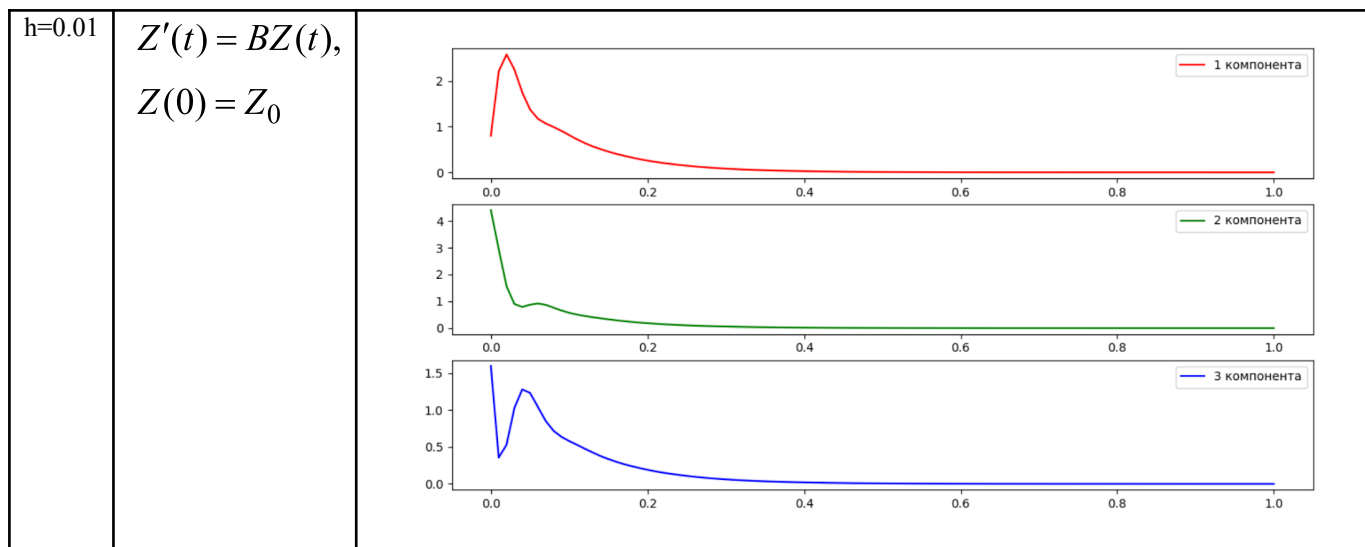
$$Y_{i+1} = (E - hA)^{-1}Y_i$$

```
#-----неявный метод Эйлера-----
def implicit_Euler(matrix, y0, h, n):
    y_t = [y0]
    for i in range(1, n + 1):
        y_t.append(np.linalg.inv(np.eye(y0.shape[0]) - h * matrix) @ y_t[i - 1])
    return np.array(y_t)

AY = implicit_Euler(A, Y0, h, n)
output(AY, t)

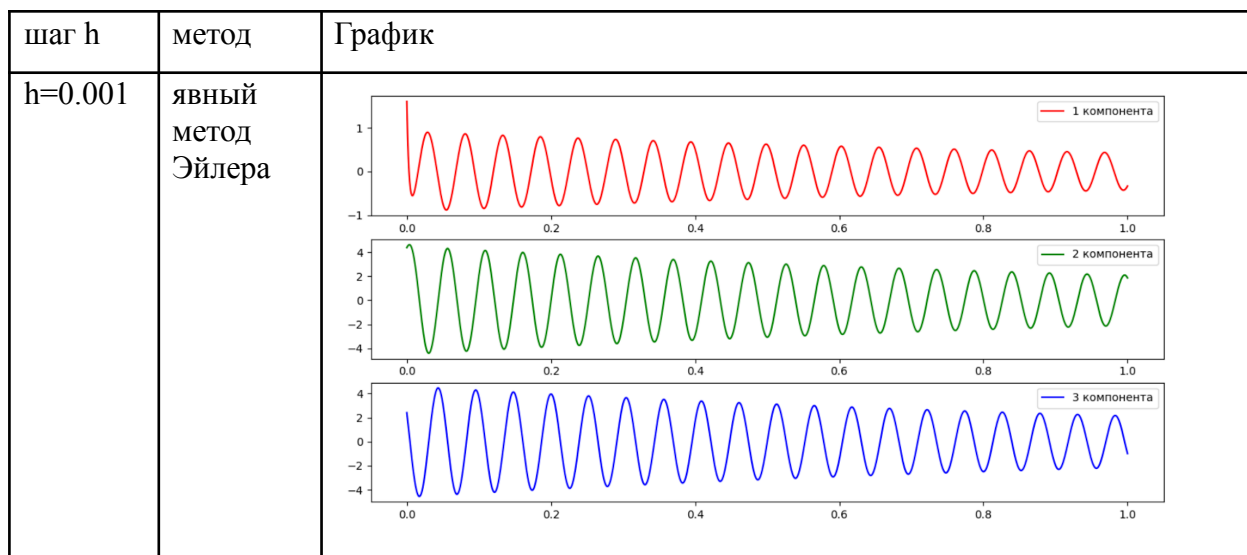
BZ = implicit_Euler(B, Z0, h, n)
output(BZ, t)
```

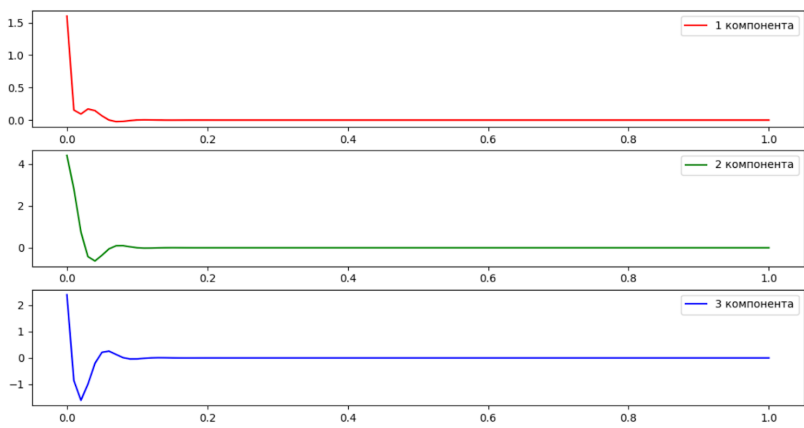
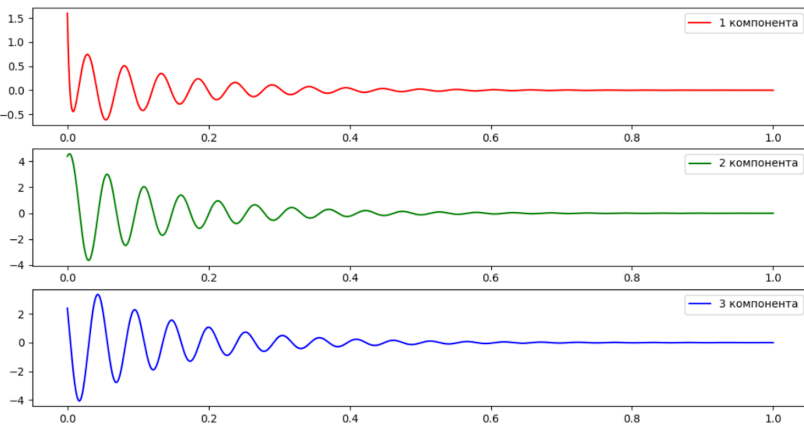
Неявный метод Эйлера		
шаг	Задача	График
h=0.01	$Y'(t) = AY(t),$ $Y(0) = Y_0$	



Мы видим, что метод устойчив при шаге $h=0.01$ и неявный метод действительно дает лучшие результаты по сравнению с явными при решении жестких задач.

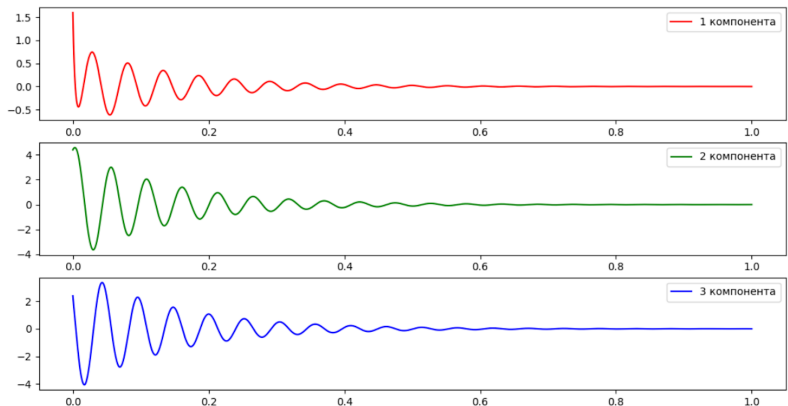
4. Для жесткой задачи экспериментально подобрать шаг h , при котором графики компонент решения, полученного по явному методу Эйлера, визуально совпадают с графиками компонент решения, полученного по неявному методу с шагом $h=0.01$. Сравнить найденное значение шага с теоретическим значением шага, при котором явный метод Эйлера для жестких задач должен быть устойчивым.

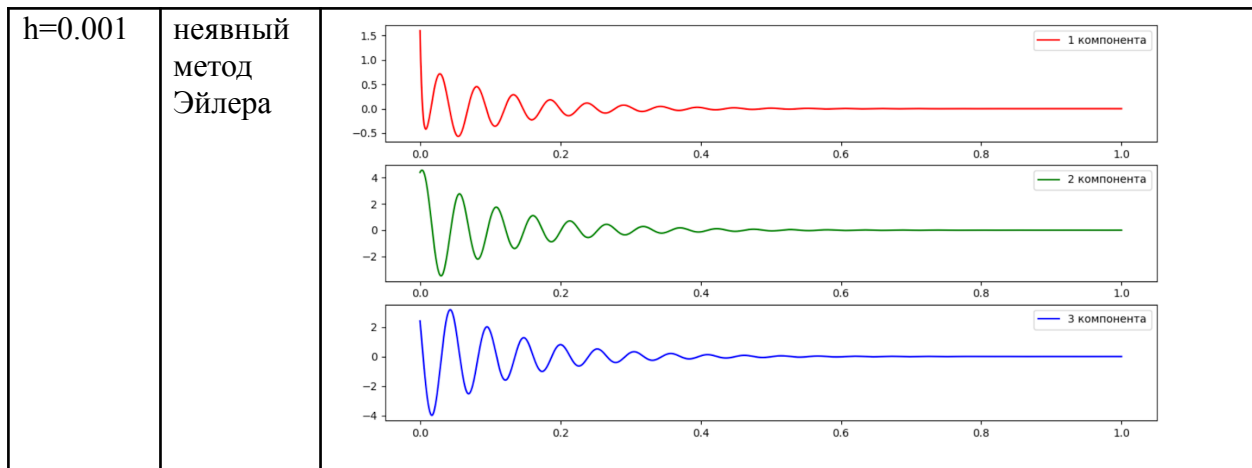


$h=0.01$	неявный метод Эйлера	
$h = 0.0001$	явный метод Эйлера	

Визуального совпадения графиков компонент решения, полученного по явному методу Эйлера, и компонент решения, полученного по неявному методу с шагом $h=0.01$, мы не наблюдаем.

Уменьшим шаг и для неявного метода Эйлера, $h = 0.0001$

шаг h	метод	График
$h=0.0001$	явный метод Эйлера	



Экспериментально получаем, что при шаге $h = 0.0001$ графики компонент решения, полученного по явному методу Эйлера, визуально совпадают с графиками компонент решения, полученного по неявному методу с шагом $h=0.001$. Но при шаге $h=0.01$ для неявного метода Эйлера такого совпадения добиться не удалось.

Теоретическая оценка: $h \leq \frac{2}{|\lambda|}$

```
abs_lambda = np.max(abs(A_lambda.real))
print("|lambda|:", abs_lambda)
h_t = 2/abs_lambda
print("Теоретическая оценка: h<= ", h_t)
```

Результат:

```
|lambda|: 279.99984897789705
Теоретическая оценка: h<= 0.007142860995463888
```

Экспериментальный шаг $h = 0.0001 \leq 0.00716$

Таким образом экспериментальный шаг не противоречит теоретической оценке.

5. Сравнить полученные результаты.

Вывод: Применение неявных методов дает лучшие результаты по сравнению с применением явных при решении жестких задач. Вычислительные эксперименты показали, что при некотором шаге (особенно если он большой), жесткая задача не решается явным методом, но неявным решается. Также при решении методом Эйлера можно подобрать шаг h , что метод Эйлера будет устойчивым и для решения жестких задач. (Метод Эйлера является устойчивым при $T \rightarrow \infty$ при шаге $h \leq \frac{2}{|\lambda|}$)