Assignment 4

Name: 陈子蔚 SID: 12332440

1

a

Since $-5 < 0$, the solutions to this ODE is asymptotically stable.

b

Since $|1 + h\lambda| = 1.5 > 1$, Euler's method is not stable for this ODE.

c

$$y_{k+1} \simeq y_k + hy'_k = y_k - 5hy_k = (1 - 5h)y_k \tag{1}$$
$$y(0.5) = (1 - 5 \cdot 0.5)y(0) = -1.5 \tag{2}$$

d

Since $\left|\frac{1}{1-h\lambda}\right| = \frac{2}{7} < 1$, the backward Euler's method is not stable for this ODE.

e

$$y_{k+1} \simeq y_k + hy'_{k+1} = y_k - 5hy_{k+1} \tag{3}$$
$$\Leftrightarrow y_{k+1} = \frac{1}{1 + 5h}y_k \tag{4}$$
$$y(0.5) = \frac{1}{1 + 5 \cdot 0.5}y(0) = \frac{2}{7} \tag{5}$$

2

a

$$y_{k+1} \simeq y_k + hy'_{k+1} = y_k - hy^2_{k+1} \tag{6}$$
$$\Leftrightarrow g(y_{k+1}) := hy^2_{k+1} + y_{k+1} - y_k = 0 \tag{7}$$

b

$$\frac{\partial g(y_{k+1})}{\partial y_{k+1}} = 2hy_{k+1} + 1 \tag{8}$$
$$g(y_{k+1})\left(\frac{\partial g(y_{k+1})}{\partial y_{k+1}}\right)^{-1} = \frac{hy^2_{k+1} + y_{k+1} - y_k}{2hy_{k+1} + 1} \tag{9}$$

To solve (7),

$$y^{\text{new}}_{k+1} \leftarrow y_k - \frac{hy^2_{k+1} + y_{k+1} - y_k}{2hy_{k+1} + 1} \tag{10}$$

c

$$y_1^{\text{init}} \leftarrow y_0 + hy_0' = y_0 - hy_0^2 = 0.9 \tag{11}$$

d

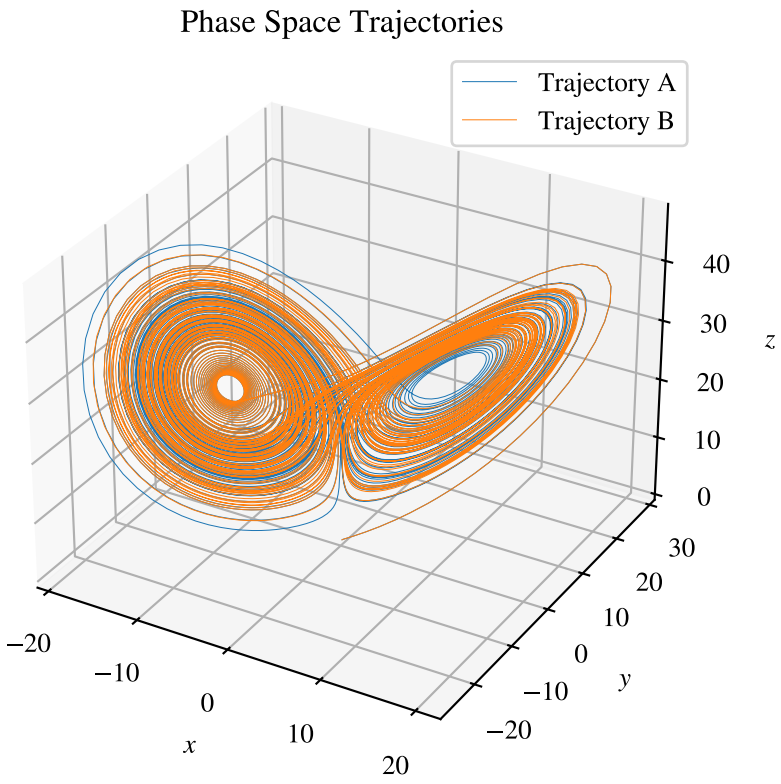
$$y_1^{[1]} \leftarrow y_0 - \frac{hy_1^2 + y_1 - y_0}{2hy_1 + 1} = \tag{12}$$

3

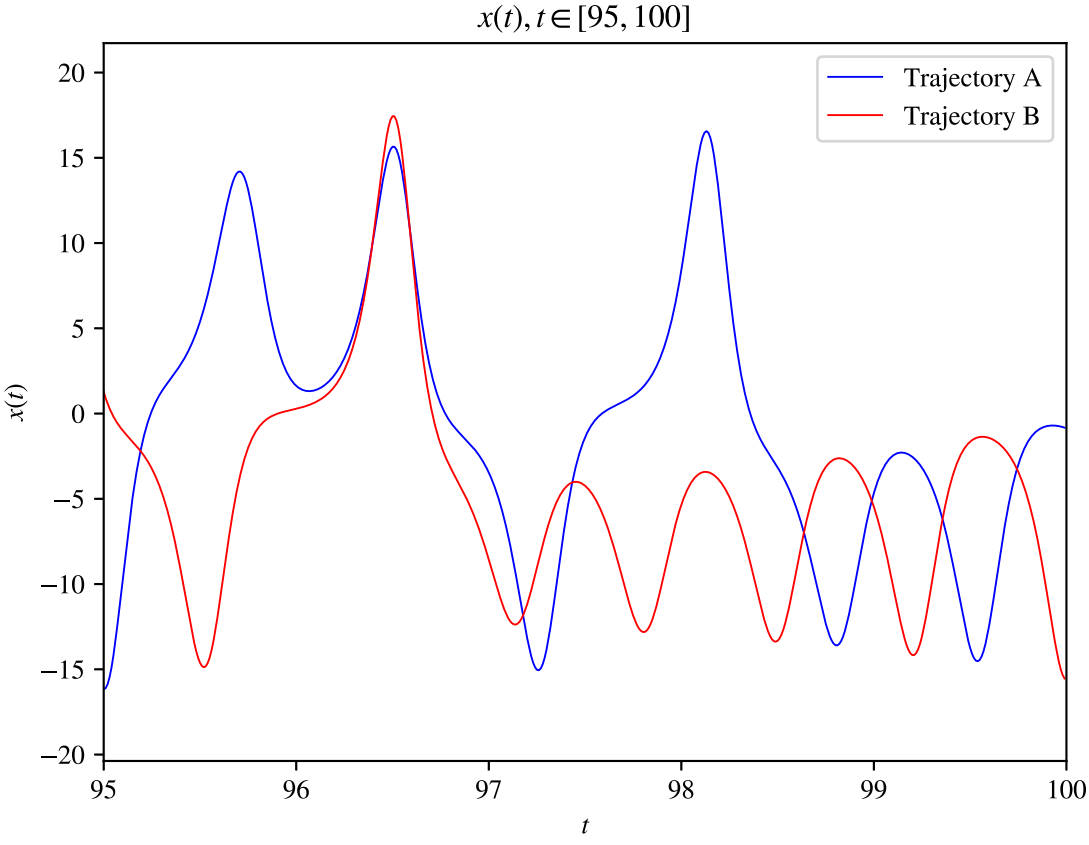
Core Code

```python
# Classical fourth-order Runge-Kutta method
def runge_kutta_4(f, y0, t):
    n = len(t)
    y = np.zeros((n, len(y0)))
    y[0] = y0
    dt = t[1] - t[0]
    for i in range(1, n):
        k1 = dt * f(y[i - 1], t[i - 1])
        k2 = dt * f(y[i - 1] + 0.5 * k1, t[i - 1] + 0.5 * dt)
        k3 = dt * f(y[i - 1] + 0.5 * k2, t[i - 1] + 0.5 * dt)
        k4 = dt * f(y[i - 1] + k3, t[i - 1] + dt)
        y[i] = y[i - 1] + (k1 + 2 * k2 + 2 * k3 + k4) / 6
    return y
```

a & b


Phase Space Trajectories
Trajectory A
Trajectory B
40
30
20
10
0
z
30
20
10
0
−10
−20
y
−20
−10
0
10
20
x

c


$x(t), t \in [95, 100]$
Trajectory A
Trajectory B
20
15
10
5
0
−5
−10
−15
−20
x(t)
95
96
97
98
99
100
t

4

a

$$u_{k+1} = u_k + hu'_k \tag{13}$$
$$u(b) = u(a) + hu'(a) \tag{14}$$
$$\Leftrightarrow \beta = \alpha + (b-a)u'(a) \tag{15}$$

b

$$u'(a) = \frac{\beta - \alpha}{b-a} \tag{16}$$

5

a

$$u_{k+1} = u_k + h_k \cdot \frac{u'_k + u'_{k+1}}{2} \tag{17}$$
$$\Leftrightarrow u(1) = u(0) + 1 \cdot \frac{u'(0) + u'(1)}{2} \tag{18}$$
$$\Leftrightarrow 1 = 1 + 1 \cdot \frac{s_0 + s_1}{2} \tag{19}$$
$$\Leftrightarrow s_0 + s_1 = 0 \tag{20}$$
$$u'_{k+1} = u'_k + h_k \cdot \frac{u''_k + u''_{k+1}}{2} \tag{21}$$
$$\Leftrightarrow u'(1) = u'(0) + 1 \cdot \frac{u''(0) + u''(1)}{2} \tag{22}$$
$$\Leftrightarrow s_1 = s_0 + 1 \cdot \frac{-4 - 4}{2} \tag{23}$$
$$\Leftrightarrow -s_0 + s_1 = -4 \tag{24}$$

(20)(24) could be easily solved,

$$s_0 = 2, s_1 = -2 \tag{25}$$

Then,

$$u'(0.5) = u'(0) + 0.5 \cdot \frac{u''(0) + u''(0.5)}{2} = 2 - 2 = 0 \tag{26}$$
$$u(0.5) = u(0) + 0.5 \cdot \frac{u'(0) + u'(0.5)}{2} = 1 + 0.5 = 1.5 \tag{27}$$

b

$$u'_k = \frac{u_{k+1} - u_{k-1}}{2h} \tag{28}$$
$$u''_k = \frac{u_{k+1} - 2u_k + u_{k-1}}{h^2} \tag{29}$$
$$u''(0.5) = \frac{u(1) - 2u(0.5) + u(0)}{0.5^2} \tag{30}$$
$$= \frac{2 - 2u(0.5)}{0.25} = -4 \tag{31}$$

That is,

$$u(0.5) = 1.5 \tag{32}$$

c

Suppose

$$u(t) = x_1 t^2 + x_2 t + x_3 \tag{33}$$

, then

$$u'(t) = 2x_1 t + x_2 \tag{34}$$
$$u''(t) = 2x_1 \tag{35}$$

The following equations would be obtained,

$$u''(t) = 2x_1 = -4 \tag{36}$$
$$u(0) = x_3 = 1 \tag{37}$$
$$u(1) = x_1 + x_2 + x_3 = 1 \tag{38}$$

Easily solved,

$$[x_1, x_2, x_3]^\top = [-2, 2, 1]^\top \tag{39}$$

Thus,

$$u(t) = -2t^2 + 2t + 1 \tag{40}$$
$$u(0.5) = 1.5 \tag{41}$$

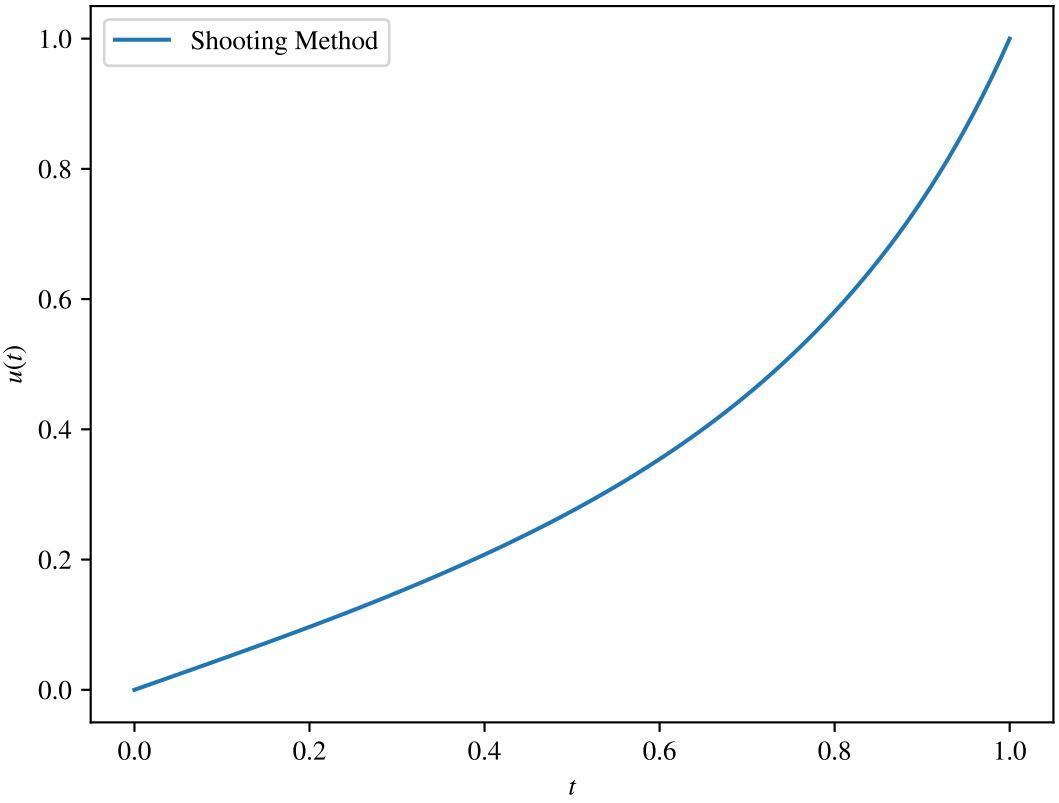
6

a

Core Code

```
def ode_system(t, y):
    u, up = y
    return [up, 10 * u**3 + 3 * u + t**2]

# 定义目标函数，用于寻找合适的初始斜率
def shooting_function(up0):
    if isinstance(up0, np.ndarray):
        up0 = up0[0]
    sol = solve_ivp(ode_system, [0, 1], [0, up0], t_eval=[1])
    return sol.y[0, -1] - 1

# 寻找合适的初始斜率
up0_guess = 1.0  # 初始猜测
sol = root(shooting_function, up0_guess)
up0_optimal = sol.x[0]
```

Result



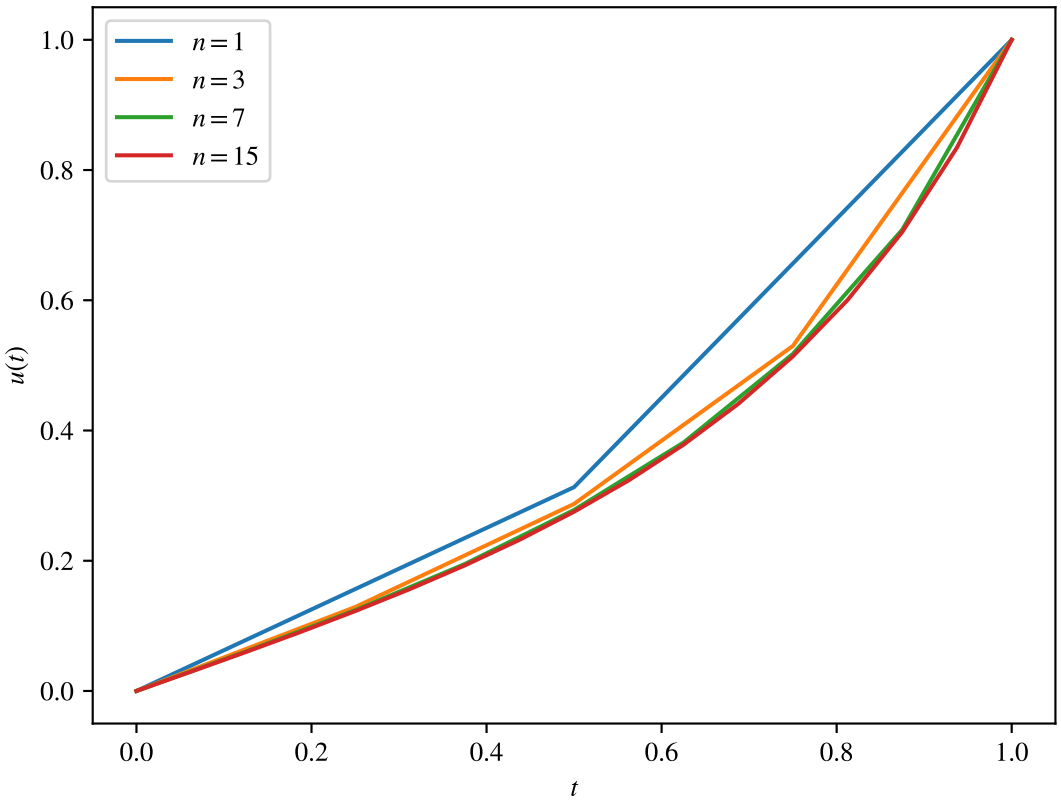
b

Core Code

```
# 牛顿迭代法
for _ in range(max_iter):
    # u_old = u.copy()
    F = np.zeros(n)
    J = np.zeros((n, n))

    for i in range(1, n + 1):
        F[i - 1] = (u[i + 1] - 2 * u[i] + u[i - 1]) / h**2 - (
            10 * u[i] ** 3 + 3 * u[i] + t_values[i] ** 2
        )
        J[i - 1, i - 1] = -2 / h**2 - 30 * u[i] ** 2 - 3
        if i > 1:
            J[i - 1, i - 2] = 1 / h**2
        if i < n:
            J[i - 1, i] = 1 / h**2

    # 求解线性方程组 J * delta_u = -F
    delta_u = np.linalg.solve(J, -F)

    # 更新解
    u[1 : n + 1] += delta_u
```

```
        # 检查收敛性
        if np.linalg.norm(delta_u) < tol:
            break
```

Result



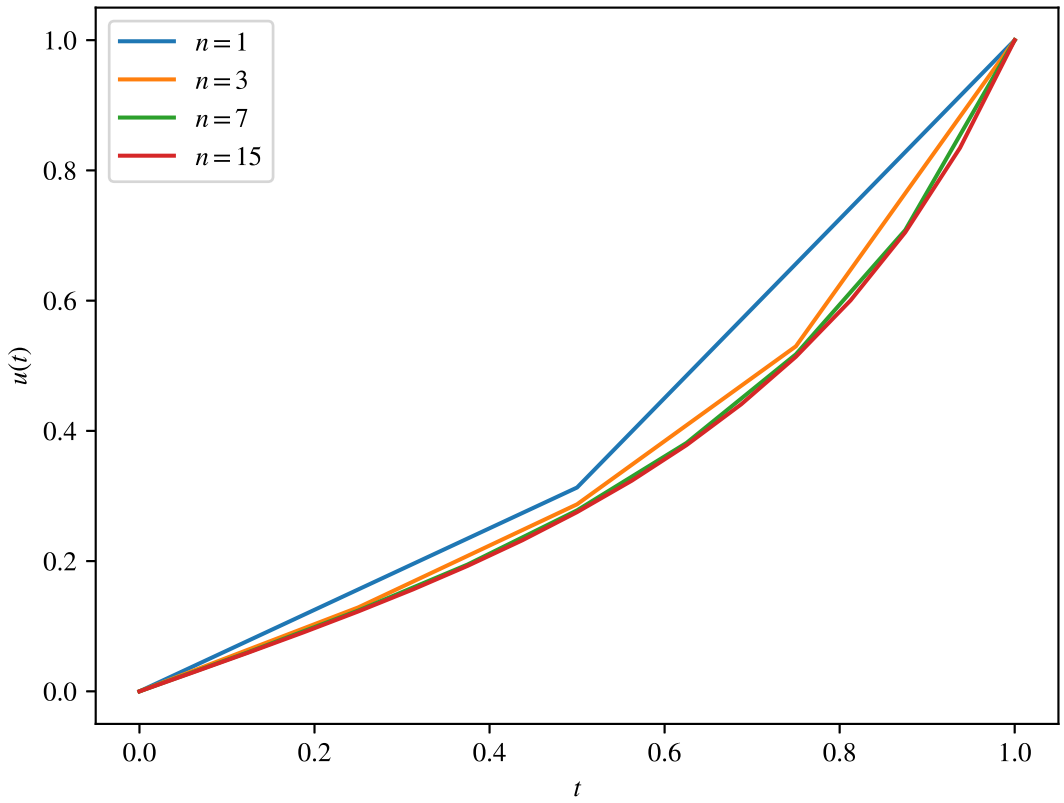
c

Core Code

```
        def func_u(t, x):
            _u = 0
            for i in range(n):
                _u += x[i] * t**i
            return _u

        def func_ddu_dtt(t, x):
            _ddu_dtt = 0
            for i in range(2, n):
                _ddu_dtt += i * (i - 1) * x[i] * t ** (i - 2)
            return _ddu_dtt

        def residuals(x):
            res = (
                [func_u(0, x) - 0]
                + [
```

```
                func_ddu_dtt(t, x)
                - 10 * func_u(t, x) ** 3
                - 3 * func_u(t, x)
                - t**2
                for t in t_values[1:-1]
            ]
            + [func_u(1, x) - 1]
        )
        return res

    result = root(residuals, np.linspace(0, 1, n))
    x = result.x
    u = [func_u(t, x) for t in t_values]

    return t_values, u
```

Result



7

Let $h = 0.5$,

$$u_{xx} + u_{yy} = \frac{u(x+h,y) - 2u(x,y) + u(x-h,y)}{h^2} \tag{42}$$

$$+ \frac{u(x,y+h) - 2u(x,y) + u(x,y-h)}{h^2} \tag{43}$$

$$= x + y \tag{44}$$

Thus,

$$\frac{u(1,0.5)-2u(0.5,0.5)+u(0,0.5)}{0.25}+\frac{u(0.5,1)-2u(0.5,0.5)+u(0.5,0)}{0.25}=0.5+0.5 \quad (45)$$

$$\Leftrightarrow 4(u(1,0.5)-2u(0.5,0.5)+u(0,0.5)+u(0.5,1)-2u(0.5,0.5)+u(0.5,0))=1 \quad (46)$$

From the figure,

$$u(1,0.5)=1, u(0,0.5)=0, u(0.5,1)=1, u(0.5,0)=0 \quad (47)$$

Thus,

$$4(2-4u(0.5,0.5))=1 \Leftrightarrow u(0.5,0.5)=\frac{7}{16}=0.4375 \quad (48)$$