1.

import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import CubicSpline

def original\_function(x):

return (5\*np.sin(3\*x/8) + 3) \* np.cos(-x\*\*2/9)

x\_data = np.linspace(0, 10, num=21, endpoint=True)

y\_data = original\_function(x\_data)

cs = CubicSpline(x\_data, y\_data)

x = np.linspace(0, 10, num=100)

plt.plot(x, original\_function(x), label='Original Function')

plt.plot(x, cs(x), label='Interpolation Curve')

plt.scatter(x\_data, y\_data, color='red', label='Data Points')

plt.legend()

plt.title('Cubic Spline Interpolation')

plt.show()

2.

﻿from scipy.optimize import fsolve

import numpy as np

def equations(vars):

x, y = vars[0], vars[1]

eq1 = y \* np.sin(x) - 4

eq2 = x \* y - x - 5

return [eq1, eq2]

initial\_guess = [1, 1]

result = fsolve(equations, initial\_guess)

solution\_x = result[0]

solution\_y = result[1]

print("解:")

print(f"x = {solution\_x}")

print(f"y = {solution\_y}")

Out:

﻿解:

x = 1.6581954959797305

y = 4.015326004758085

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

def ode\_func(x, y):

return [y[1], 2 \* y[1] - 3 \* y[0] + 4]

initial\_condition = [16, 18]

t\_eval = np.linspace(0, 10, 1000)

solution = solve\_ivp(ode\_func, [0, 10], initial\_condition, t\_eval=t\_eval)

plt.scatter(solution.t, solution.y[0], color='red', label='Numerical Solution')

def analytical\_solution(x):

return (16 \* np.cos(x) + 2 \* np.sin(x) + 8 \* np.exp(x)) / 6

plt.plot(t\_eval, analytical\_solution(t\_eval), label='Analytical Solution')

plt.legend()

plt.title('Solution of the Differential Equation')

plt.show()