1 Problem 1

In this text the questions for problem 1 for assignment 2 is presented, solved and answered. the system that is being analyzed in this problem is the following:

$$\frac{\partial n}{\partial t} = rn\left(1 - \frac{n}{K}\right) - \frac{An}{1 + n/B} + \frac{\partial^2 n}{\partial x^2} \tag{1}$$

1.1 a)

For this first assignment the equation for the system is to be tranformed to dimensionless parameters this is done by the following substitioutions:

$$\tau = At \tag{2}$$

$$\xi = x\sqrt{\frac{A}{D}} \tag{3}$$

$$u(\xi, \tau) = \frac{n(x,t)}{B} \tag{4}$$

To begin the transformation we start with evaluate the following:

$$\frac{\partial u}{\partial \tau} = \frac{\partial u}{\partial t}\frac{\partial t}{\partial \tau} = \frac{1}{BA} \cdot \frac{\partial n}{\partial t} \tag{5}$$

$$\frac{\partial^2 u}{\partial \xi^2} = \frac{\partial^2 u}{\partial x^2}\left(\frac{\partial x}{\partial xi}\right)^2 = \frac{D}{BA} \cdot \frac{\partial^2 n}{\partial x^2} \tag{6}$$

With these two equations we can tranform our system:

$$\frac{\partial u}{\partial \tau} = \frac{1}{BA}\left(rBu(1 - \frac{Bu}{K}) - \frac{ABu}{1+u} + AB\frac{\partial^2 u}{\partial \xi^2}\right) = \rho u(1 - \frac{u}{q}) - \frac{u}{1+u} + \frac{\partial^2 u}{\partial \xi^2} \tag{7}$$

Where $\rho = \frac{r}{A}$ and $q = \frac{K}{B}$.

1.2 b)

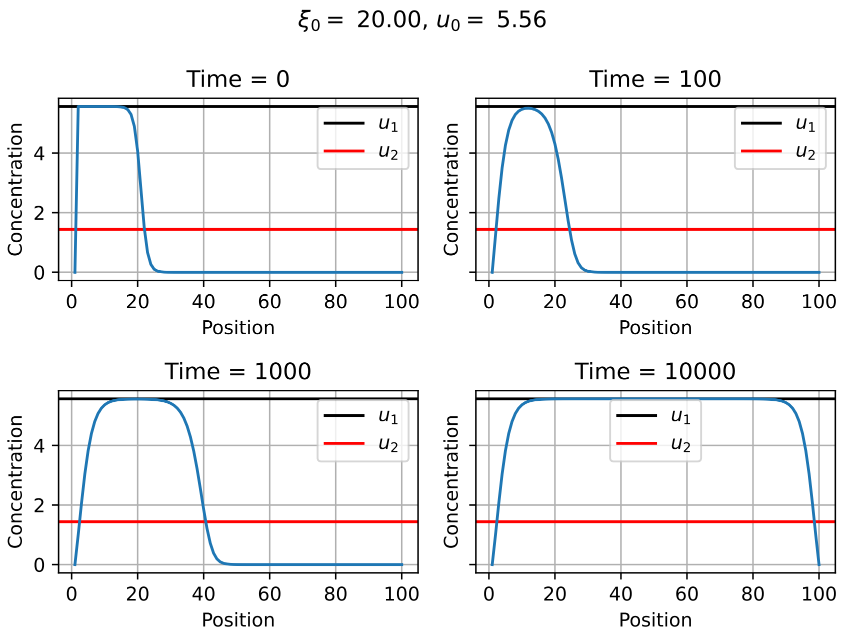

Figure 1:

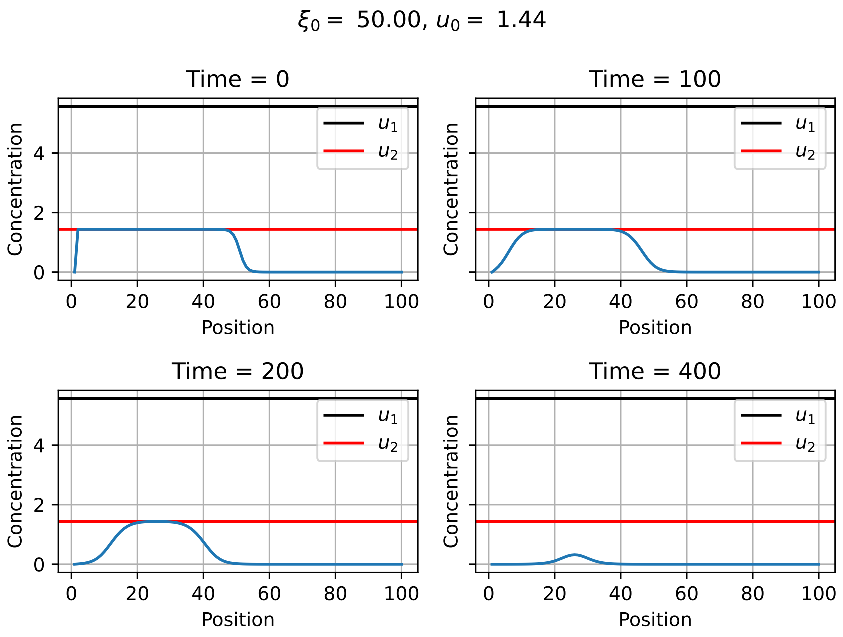

Figure 2:

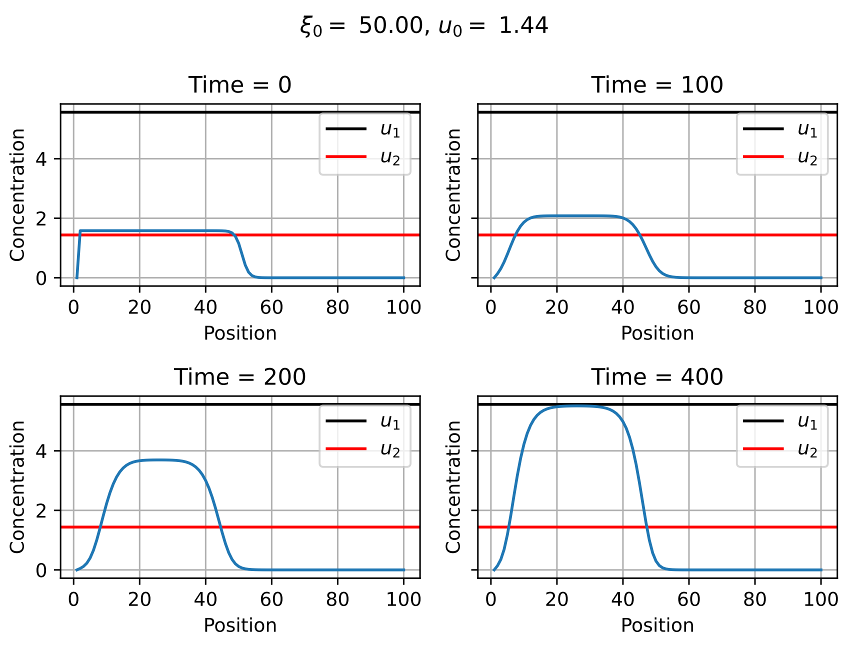

Figure 3:

2 code

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


def euler_second_derivative(f, h):
    second_derivative = (np.roll(f, 1) + np.roll(f, -1) - 2 * f) / (h**2)
    return second_derivative


def system(u: np.ndarray, rho: float, q: float, d2udxi2: np.ndarray) -> np.ndarray:
    dudt = rho * u * (1 - u / q) - u / (1 + u) + d2udxi2
    return dudt


def start_concentration(u_0: float, xi_0: float, habitat_size: int) -> np.ndarray:
    index = np.arange(0, habitat_size, dtype=int)
    start_concentration = u_0 / (1 + np.exp(index - xi_0))
    start_concentration[0] = 0.0
    start_concentration[-1] = 0.0
    return start_concentration


def start_concentration2(u_0: float, xi_0: float, habitat_size: int) -> np.ndarray:
    index = np.arange(0, habitat_size, dtype=int)
    start_concentration = u_0 * np.exp(-((index - xi_0) ** 2))
    start_concentration[0] = 0.0
    start_concentration[-1] = 0.0
    return start_concentration


def run_iteration(
    n_i: np.ndarray, rho: float, q: float, time_step: float, habitat_step: float = 1.0
) -> np.ndarray:
    d2dxi2 = euler_second_derivative(n_i, habitat_step)
    dudt = system(n_i, rho, q, d2dxi2)
    dudt[0] = 0.0
    dudt[-1] = 0.0
    n_new = dudt * time_step + n_i
    return n_new


def run_simulation(
    u_0: float,
    xi_0: float,
    rho: float,
    q: float,
    habitat_size: int,
    sim_time: int,
    time_step: float,
    second_init=False,
):
    sim_steps = int(sim_time / time_step)
    simulation_matrix = np.zeros((habitat_size, sim_steps))
    if second_init:
        simulation_matrix[:, 0] = start_concentration2(u_0, xi_0, habitat_size)
    else:
        simulation_matrix[:, 0] = start_concentration(u_0, xi_0, habitat_size)

    for idx in range(0, sim_steps - 1):
        simulation_matrix[:, idx + 1] = run_iteration(
            simulation_matrix[:, idx], rho, q, time_step
        )
    return simulation_matrix


def plot1():
    rho = 0.5
    q = 8.0
    u_01 = 1 / 2 * (q - 1 + np.sqrt((q - 1) ** 2 - 4 * q * (1 / rho - 1)))
    u_02 = 1 / 2 * (q - 1 - np.sqrt((q - 1) ** 2 - 4 * q * (1 / rho - 1)))
    xi_0 = 20
    time_step = 0.1
    sim_time = 10000
    habitat_size = 100

    sim_matrix = run_simulation(u_01, xi_0, rho, q, habitat_size, sim_time, time_step)

    plot_times = np.array([0, 100, 1000, 10000])
    x = np.arange(1, habitat_size + 1, 1)
    fig, ax = plt.subplots(2, 2, sharey=True)
    ax = ax.flatten()
    for idx, time in enumerate(plot_times):
        ax[idx].axhline(u_01, label=r"$u_1$", c="black")
        ax[idx].axhline(u_02, label=r"$u_2$", c="red")
        ax[idx].plot(x, sim_matrix[:, time])
        ax[idx].set_xlabel("Position")
        ax[idx].set_ylabel("Concentration")
        ax[idx].set_title(f"Time = {time}")
        ax[idx].grid()
        ax[idx].legend()

    fig.suptitle(rf"$\xi_0 =$ {xi_0:.2f}, $u_0 =$ {u_01:.2f}")
    fig.tight_layout()
    fig.savefig("../report/figures/1b1.pdf")


def plot2():
    rho = 0.5
    q = 8.0
    u_01 = 1 / 2 * (q - 1 + np.sqrt((q - 1) ** 2 - 4 * q * (1 / rho - 1)))
    u_02 = 1 / 2 * (q - 1 - np.sqrt((q - 1) ** 2 - 4 * q * (1 / rho - 1)))
    xi_0 = 50.0
    time_step = 0.1
    sim_time = 10000
    habitat_size = 100

    sim_matrix = run_simulation(u_02, xi_0, rho, q, habitat_size, sim_time, time_step)

    plot_times = np.array([0, 100, 200, 400])
    x = np.arange(1, habitat_size + 1, 1)
    fig, ax = plt.subplots(2, 2, sharey=True)
    ax = ax.flatten()

    for idx, time in enumerate(plot_times):
        ax[idx].axhline(u_01, label=r"$u_1$", c="black")
        ax[idx].axhline(u_02, label=r"$u_2$", c="red")
        ax[idx].plot(x, sim_matrix[:, time])
        ax[idx].set_xlabel("Position")
        ax[idx].set_ylabel("Concentration")
        ax[idx].set_title(f"Time = {time}")
        ax[idx].grid()
        ax[idx].legend()

    fig.suptitle(rf"$\xi_0 =$ {xi_0:.2f}, $u_0 =$ {u_02:.2f}")
    fig.tight_layout()
    fig.savefig("../report/figures/1b2.pdf")


def plot3():
    rho = 0.5
    q = 8.0
    u_01 = 1 / 2 * (q - 1 + np.sqrt((q - 1) ** 2 - 4 * q * (1 / rho - 1)))
    u_02 = 1 / 2 * (q - 1 - np.sqrt((q - 1) ** 2 - 4 * q * (1 / rho - 1)))
    xi_0 = 50.0
    time_step = 0.1
    sim_time = 10000
    habitat_size = 100

    sim_matrix = run_simulation(
        u_02 * 1.1, xi_0, rho, q, habitat_size, sim_time, time_step
    )

    plot_times = np.array([0, 100, 200, 400])
    x = np.arange(1, habitat_size + 1, 1)
    fig, ax = plt.subplots(2, 2, sharey=True)
    ax = ax.flatten()

    for idx, time in enumerate(plot_times):
        ax[idx].axhline(u_01, label=r"$u_1$", c="black")
        ax[idx].axhline(u_02, label=r"$u_2$", c="red")
        ax[idx].plot(x, sim_matrix[:, time])
        ax[idx].set_xlabel("Position")
        ax[idx].set_ylabel("Concentration")
        ax[idx].set_title(f"Time = {time}")
        ax[idx].grid()
        ax[idx].legend()

    fig.suptitle(rf"$\xi_0 =$ {xi_0:.2f}, $u_0 =$ {u_02:.2f}")
    fig.tight_layout()
    fig.savefig("../report/figures/1b3.pdf")


def plot4():
    rho = 0.5
    q = 8.0
    u_01 = 1 / 2 * (q - 1 + np.sqrt((q - 1) ** 2 - 4 * q * (1 / rho - 1)))
    u_02 = 1 / 2 * (q - 1 - np.sqrt((q - 1) ** 2 - 4 * q * (1 / rho - 1)))
    xi_0 = 50.0
    time_step = 0.1
    sim_time = 10000
    habitat_size = 100

    sim_matrix = run_simulation(
        u_01, xi_0, rho, q, habitat_size, sim_time, time_step, second_init=True
    )

    plot_times = np.array([0, 50, 100, 150])
    x = np.arange(1, habitat_size + 1, 1)
    fig, ax = plt.subplots(2, 2, sharey=True)
    ax = ax.flatten()

    for idx, time in enumerate(plot_times):
        ax[idx].axhline(u_01, label=r"$u_1$", c="black")
        ax[idx].axhline(u_02, label=r"$u_2$", c="red")
        ax[idx].plot(x, sim_matrix[:, time])
        ax[idx].set_xlabel("Position")
        ax[idx].set_ylabel("Concentration")
        ax[idx].set_title(f"Time = {time}")
        ax[idx].grid()
        ax[idx].legend()

    fig.suptitle(rf"$\xi_0 =$ {xi_0:.2f}, $u_0 =$ {u_01:.2f}")
    fig.tight_layout()
    fig.savefig("../report/figures/1c1.pdf")


def plot5():
    rho = 0.5
    q = 8.0
    u_01 = 1 / 2 * (q - 1 + np.sqrt((q - 1) ** 2 - 4 * q * (1 / rho - 1)))
    u_02 = 1 / 2 * (q - 1 - np.sqrt((q - 1) ** 2 - 4 * q * (1 / rho - 1)))
    xi_0 = 50.0
    time_step = 0.1
    sim_time = 10000
    habitat_size = 100

    sim_matrix = run_simulation(
        u_01 * 3, xi_0, rho, q, habitat_size, sim_time, time_step, second_init=True
    )

    plot_times = np.array([0, 100, 400, 2000])
    x = np.arange(1, habitat_size + 1, 1)
    fig, ax = plt.subplots(2, 2, sharey=True)
    ax = ax.flatten()

    for idx, time in enumerate(plot_times):
        ax[idx].axhline(u_01, label=r"$u_1$", c="black")
        ax[idx].axhline(u_02, label=r"$u_2$", c="red")
        ax[idx].plot(x, sim_matrix[:, time])
        ax[idx].set_xlabel("Position")
        ax[idx].set_ylabel("Concentration")
        ax[idx].set_title(f"Time = {time}")
        ax[idx].grid()
        ax[idx].legend()

    fig.suptitle(rf"$\xi_0 =$ {xi_0:.2f}, $u_0 =$ {u_01 *3:.2f}")
    fig.tight_layout()
    fig.savefig("../report/figures/1c2.pdf")


def main():
    plot1()
    plot2()
    plot3()
    plot4()
    plot5()


if __name__ == "__main__":
    main()
```