CompBio HW2

Axel Eiman - 980626-8414
William Truvé - 980312-4271

February 2022

Problem 1, Travelling waves

Task a)

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

$\tau = At$, $\xi = x\sqrt{A/D}$, $u(\xi, \tau) = \frac{n(x,t)}{B}$, $\rho = r/A$, $q = K/B$, and ignoring diffusion:

$$\frac{AB\partial u}{\partial \tau} = A\rho Bu\left(1 - \frac{Bu}{qB}\right) - \frac{ABu}{1 + \frac{Bu}{B}} \tag{2}$$

$$\frac{\partial u}{\partial \tau} = \rho u\left(1 - \frac{u}{q}\right) - \frac{u}{1 + u} \tag{3}$$

Now we solve:

$$\rho u\left(1 - \frac{u}{q}\right) - \frac{u}{1 + u} = 0 \tag{4}$$

$$\left(\rho u - \frac{\rho u}{q}\right)(1 + u) - u = 0 \tag{5}$$

$$u\left(-\frac{\rho u^2}{q} + \rho u - \frac{\rho u}{q} + \rho - 1\right) = 0 \tag{6}$$

This gives a fixed point $\Rightarrow u_3 = 0$. We will call it u_3 to follow the naming used in the assignment description. u_1, u_2 are then given by:

$$-\frac{\rho u^2}{q} + \rho u - \frac{\rho u}{q} + \rho - 1 = 0 \tag{7}$$

$$u^2 - (q - 1)u + \frac{q(1 - \rho)}{\rho} = 0 \tag{8}$$

$$u_{1,2} = \frac{q - 1}{2} \pm \sqrt{\left(\frac{q - 1}{2}\right)^2 - \frac{q(1 - \rho)}{\rho}} \tag{9}$$

For the next part of the assignment we need to account for the diffusion parameter as well. Dedimensionalizing this goes as follows:

$$D\frac{\partial^2 n}{\partial x^2} \Rightarrow D\frac{\partial^2 Bu}{\partial \frac{\xi^2}{A/D}} = AB\frac{\partial^2 u}{\partial \xi^2} \tag{10}$$

Similarly to what we saw in eq. (2) this has a factor AB which cancels out, so adding this back into the dedimensionalized system we get:

$$\frac{\partial u}{\partial \tau} = \rho u\left(1 - \frac{u}{q}\right) - \frac{u}{1+u} + \frac{\partial^2 u}{\partial \xi^2} \tag{11}$$

Task b)

Now we set $\rho = 0.5$ and $q = 8$ and simulate the dynamics for the three starting cases. The dynamics are visualized in the figures below at a time step when waves have developed.

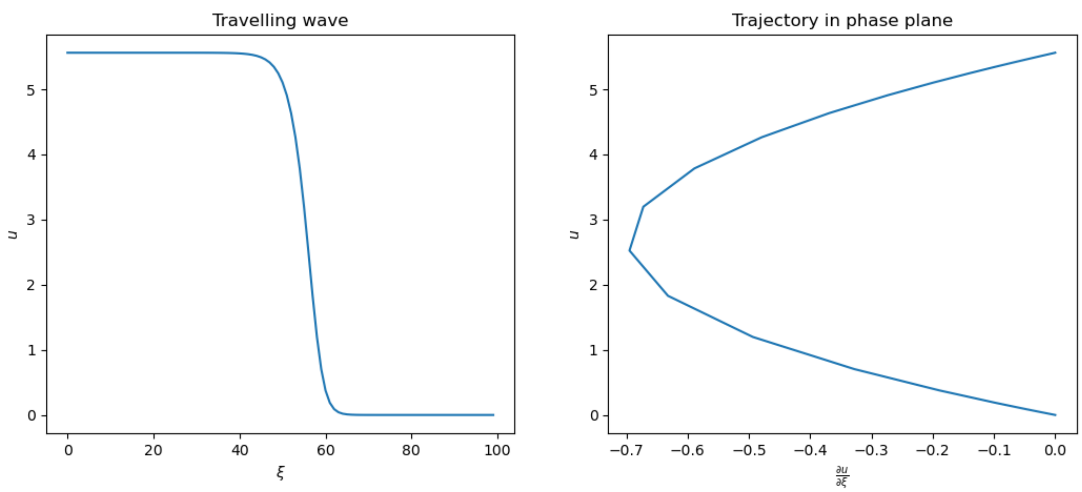

Figure 1: $\xi = 20$ and $u_0 = u_1^*$

With $\xi = 20$ and $u_0 = u_1^*$ we see a clear travelling wave where the population spreads through the habitat. The wave is travelling with a velocity estimated numerically to be $c = 0.17956$. The velocity being positive means that populations move from the fixed point at $u = 0$ in the phase diagram up to the fixed point at $u = u_1^*$. Therefor the fixed point at $u = 0$ is unstable and $u = u_1^*$ is stable.

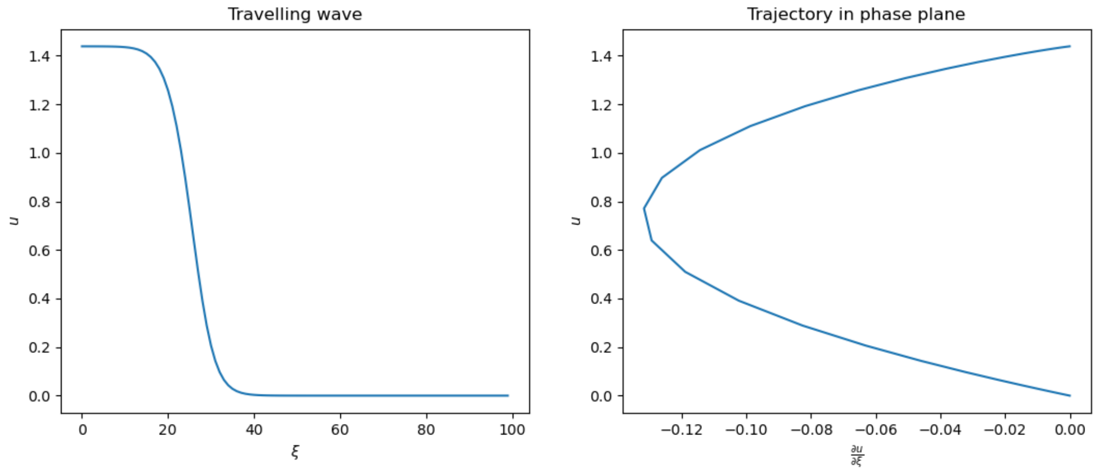

Figure 2: $\xi = 50$ and $u_0 = u_2^*$

With $\xi = 50$ and $u_0 = u_2^*$ we see a fast wave travelling in the opposite direction, so the population is disappearing instead of spreading. The velocity is higher than the first one, at $c = -0.71235$. The velocity being negative means that the stability of the fixed points in the phase plane are opposite to what they were in the first case, so $u = 0$ is stable and $u = u_2^*$ is unstable.

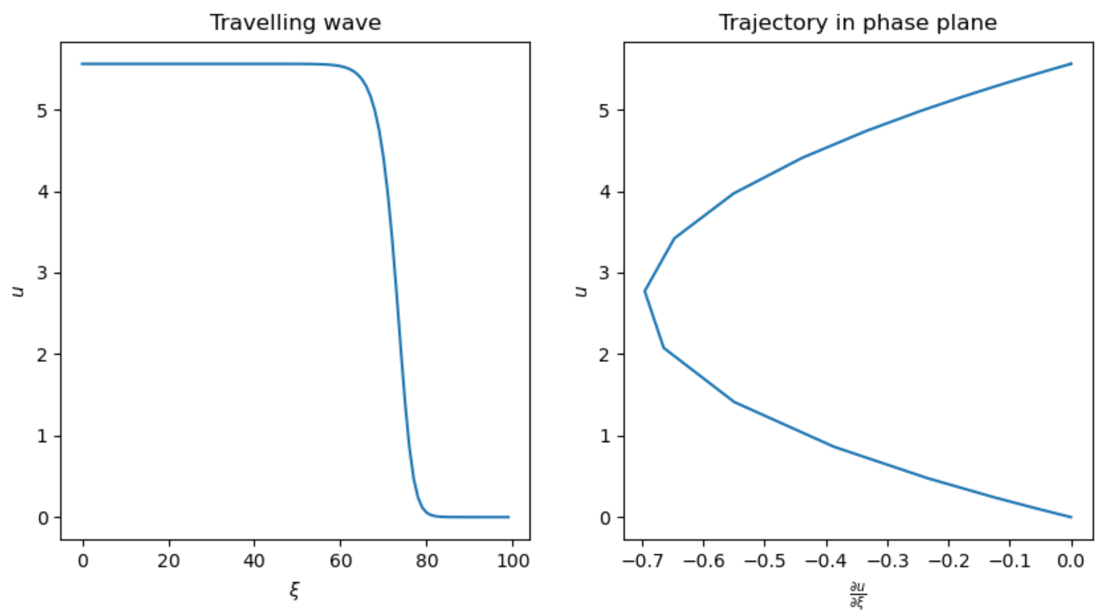

Figure 3: $\xi = 50$ and $u_0 = 1.1 * u_2^*$

With $\xi = 50$ and $u_0 = 1.1 * u_2^*$ we see a little different dynamics. The "top" of the starting wave is above the fixed point u_2, and the whole thing increases toward the other non-zero fixed point. When the wave has reached that height it acts as in the first scenario, travelling with a velocity of $c = 0.17956$. Similarly to the first case, the velocity being positive tells us that $u = 0$ is unstable and $u = u_1^*$ is stable.

Task c)

For the case $u_0 = u_1^*$ we see the peak disappear quite quickly. The population decreases and spreads slightly outward from the peaks location. With $u_0 = 3 * u_1^*$ however, the peak is large enough to result in a population density strong enough to create travelling waves outward from the initial peak in both directions.

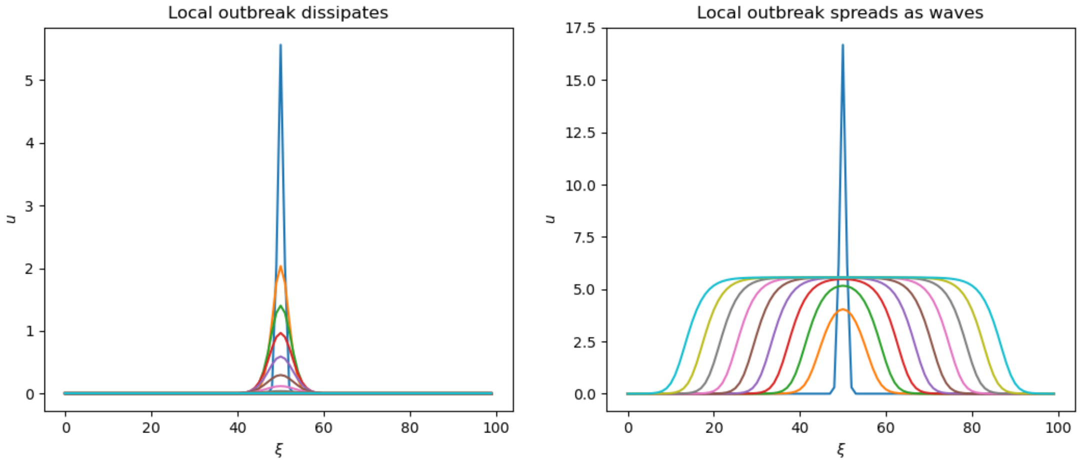

Figure 4: Local outbreak dynamics shown at representative time steps with $u_0 = u_1^*$ (left), and $u_0 = u_1^*$ (right)

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

rho = 0.5
q = 8
L = 100
dt = 1/1000
tmax = 500
tsteps = int(tmax/dt)

h = 1

# (i)
u = np.zeros((L, tsteps + 1))
xi_0 = 20
u_0 = (q - 1) / 2 + np.sqrt(((q - 1) / 2) ** 2 - q * (1 - rho) / rho)

for xi in range(L):
    u[xi, 0] = u_0 / (1 + np.exp(xi - xi_0))

# plt.plot(u[:,0])

for tau in range(tsteps):

    for xi in range(L):
        if xi == 0 or xi == 99:
            u[xi, tau + 1] = u[xi, tau] + dt * (rho * u[xi, tau] * (1 - u[xi, tau] / q) - u
        else:
            u[xi, tau + 1] = u[xi, tau] + dt * (
                        rho * u[xi, tau] * (1 - u[xi, tau] / q) - u[xi, tau] / (1 + u[xi, t
                            u[xi + h, tau] + u[xi - h, tau] - 2 * u[xi, tau]) / h ** 2)

# (ii)
u2 = np.zeros((L, tsteps + 1))
xi_0 = 50
u_0 = (q - 1) / 2 - np.sqrt(((q - 1) / 2) ** 2 - q * (1 - rho) / rho)

for xi in range(L):
    u2[xi, 0] = u_0 / (1 + np.exp(xi - xi_0))

# plt.plot(u2[:,0])

for tau in range(tsteps):

    for xi in range(L):
        if xi == 0 or xi == 99:
```

```
            u2[xi, tau + 1] = u2[xi, tau] + dt * (
                        rho * u2[xi, tau] * (1 - u2[xi, tau] / q) - u2[xi, tau] / (1 + u2[x
        else:
            u2[xi, tau + 1] = u2[xi, tau] + dt * (
                        rho * u2[xi, tau] * (1 - u2[xi, tau] / q) - u2[xi, tau] / (1 + u2[x
                            u2[xi + h, tau] + u2[xi - h, tau] - 2 * u2[xi, tau]) / h ** 2)

# (iii)
u3 = np.zeros((L, tsteps + 1))
xi_0 = 50
u_0 = 1.1 * ((q - 1) / 2 - np.sqrt(((q - 1) / 2) ** 2 - q * (1 - rho) / rho))

for xi in range(L):
    u3[xi, 0] = u_0 / (1 + np.exp(xi - xi_0))

# plt.plot(u3[:,0])

for tau in range(tsteps):

    for xi in range(L):
        if xi == 0 or xi == 99:
            u3[xi, tau + 1] = u3[xi, tau] + dt * (
                        rho * u3[xi, tau] * (1 - u3[xi, tau] / q) - u3[xi, tau] / (1 + u3[x
        else:
            u3[xi, tau + 1] = u3[xi, tau] + dt * (
                        rho * u3[xi, tau] * (1 - u3[xi, tau] / q) - u3[xi, tau] / (1 + u3[x
                            u3[xi + h, tau] + u3[xi - h, tau] - 2 * u3[xi, tau]) / h ** 2)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=[12.8, 4.8])

ax1.plot(u[:,200000])
ax1.set_ylabel('u')
ax1.set_xlabel('xi')

v = np.diff(u[:,200000])
ax2.plot(v, u[:-1,200000])
ax2.set_xlabel('du/dxi')
ax2.set_ylabel('u')

print(np.argmax(u[60,:]>2))
print(np.argmax(u[50,:]>2))
print(10/((np.argmax(u[60,:]>2)-np.argmax(u[50,:]>2))*dt))

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=[12.8, 4.8])

ax1.plot(u2[:,40000])
```

```
ax1.set_ylabel('u')
ax1.set_xlabel('xi')

v = np.diff(u2[:,40000])
ax2.plot(v,u2[:-1,40000])
ax2.set_xlabel('du/dxi')
ax2.set_ylabel('u')

print(10/((np.argmax(u2[30,:]<0.5)-np.argmax(u2[20,:]<0.5))*dt))

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=[12.8, 4.8])

ax1.plot(u3[:,200000])
ax1.set_ylabel('u')
ax1.set_xlabel('xi')

v = np.diff(u3[:,200000])
ax2.plot(v,u3[:-1,200000])
ax2.set_xlabel('du/dxi')
ax2.set_ylabel('u')

print(10/((np.argmax(u3[70,:]>2)-np.argmax(u3[60,:]>2))*dt))

xi_0 = 50
u4 = np.zeros((L, tsteps + 1))
u5 = np.zeros((L, tsteps + 1))
u_0 = ((q - 1) / 2 + np.sqrt(((q - 1) / 2) ** 2 - q * (1 - rho) / rho))
u_02 = 3 * ((q - 1) / 2 + np.sqrt(((q - 1) / 2) ** 2 - q * (1 - rho) / rho))

for xi in range(L):
    u4[xi, 0] = u_0 * np.exp(-(xi - xi_0) ** 2)
    u5[xi, 0] = u_02 * np.exp(-(xi - xi_0) ** 2)

for tau in range(tsteps):

    for xi in range(L):
        if xi == 0 or xi == 99:
            u4[xi, tau + 1] = u4[xi, tau] + dt * (
                        rho * u4[xi, tau] * (1 - u4[xi, tau] / q) - u4[xi, tau] / (1 + u4[x
        else:
            u4[xi, tau + 1] = u4[xi, tau] + dt * (
                        rho * u4[xi, tau] * (1 - u4[xi, tau] / q) - u4[xi, tau] / (1 + u4[x
                            u4[xi + h, tau] + u4[xi - h, tau] - 2 * u4[xi, tau]) / h ** 2)
for tau in range(tsteps):

    for xi in range(L):
```

```
        if xi == 0 or xi == 99:
            u5[xi, tau + 1] = u5[xi, tau] + dt * (
                        rho * u5[xi, tau] * (1 - u5[xi, tau] / q) - u5[xi, tau] / (1 + u5[x
        else:
            u5[xi, tau + 1] = u5[xi, tau] + dt * (
                        rho * u5[xi, tau] * (1 - u5[xi, tau] / q) - u5[xi, tau] / (1 + u5[x
                            u5[xi + h, tau] + u5[xi - h, tau] - 2 * u5[xi, tau]) / h ** 2)

for i in np.linspace(0,0.2*tmax/dt,50):
    plt.plot(u4[:,int(i)])


for i in np.linspace(0,tmax/dt,30):
    plt.plot(u5[:,int(i)])
```