Diffusion driven instability

Karl Lundgren
Erik Norlin

February, 2023

Problem 2.2

Consider the model for two interacting chemicals undergoing a Belousov-Zhabotinsky reaction

$$\begin{cases} \dfrac{\partial u}{\partial t} = a - (b+1)u + u^2 v + D_u \nabla^2 u \\ \dfrac{\partial v}{\partial t} = bu - u^2 v + D_v \nabla^2 v. \end{cases} \tag{1}$$

Here u denotes the concentration of the activator and v denotes the concentration of the inhibitor. We also have the positive constants, a and b. The diffusion coefficients are denoted by D_u and D_v.

a) **Neglect diffusion and determine the spatially homogeneous steady state(s) of the system (1) and their stability in terms of a and b.**

To obtain the fixed points we neglect the diffusion terms and set

$$\begin{cases} \dfrac{\partial u}{\partial t} = a - (b+1)u + u^2 v = 0 \\ \dfrac{\partial v}{\partial t} = bu - u^2 v = 0. \end{cases} \tag{2}$$

Apart from the trivial solution $u_1^* = 0$ we get

$$\begin{cases} u_2^* = a \\ v^* = \dfrac{b}{a}. \end{cases} \tag{3}$$

To determine their stability, the Jacobian of the system is calculated as

$$\boldsymbol{J} = \begin{bmatrix} b-1 & a^2 \\ -b & -a^2 \end{bmatrix}, \tag{4}$$

with

$$\begin{cases} \operatorname{tr} \boldsymbol{J} = b - a^2 - 1 \\ \det \boldsymbol{J} = a^2. \end{cases} \tag{5}$$

This is stable when $\operatorname{tr} \boldsymbol{J} < 0$ and $\det \boldsymbol{J} > 0$. The second condition will always be fulfilled, i.e., the system will never have saddle points.

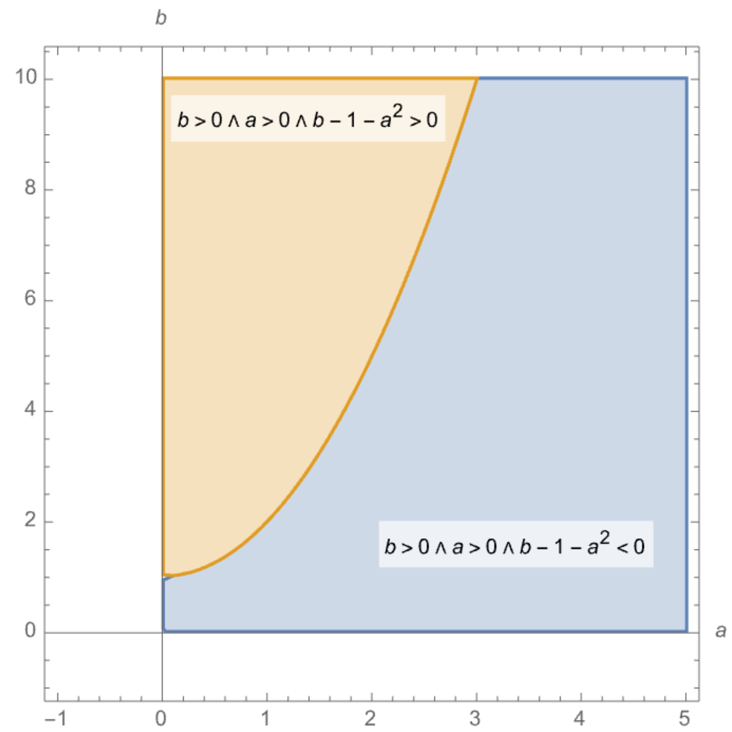

Figure 1: The bifurcation diagram of the system, a plotted against b. The blue region is stable while the orange region is unstable. There is no region containing saddle points.

b) **For which values of D_v does the homogeneous stable steady state(s) of the system (1) exhibit a diffusion-driven instability?**

From now on the parameters take on the values $a = 3$, $b = 8$, $D_u = 1$ and $D_v > 1$. By quickly checking the stability by looking at the bifurcation diagram in Figure 1 we can see that this set of parameters still are located in the stable blue region.

The conditions for when a system exhibits diffusion driven instability, also called Turing instability, is mentioned in (see Gustafsson 2023, chapter 7.3.4).

$$\begin{cases} d > 1 \\ \dfrac{(dJ_{11} + J_{22})^2}{4d} \geq \det \boldsymbol{J} \\ dJ_{11} + J_{22} > 0. \end{cases} \tag{6}$$

where $d = \dfrac{D_v}{D_u}$. To find the critical value of D_v we set

$$\frac{(dJ_{11} + J_{22})^2}{4d} = \det \boldsymbol{J} \tag{7}$$

and solve this quadratic equation for d which yields.

$$d = \frac{2\det \boldsymbol{J} - J_{11}J_{22} \pm 2\sqrt{(\det \boldsymbol{J})^2 - J_{11}J_{22}\det \boldsymbol{J}}}{J_{11}^2} = \frac{D_v}{D_u} \tag{8}$$

Inserting our values for the parameters yields two solutions

$$\begin{cases} \cancel{D_{v_1} = 0.614} \\ D_{v_2} = 2.692 \end{cases} \tag{9}$$

where only a $D_v > D_{v_2}$ fulfills all the conditions mentioned in (6).

c) **Simulate the dynamics of the system (1) on a square grid of size $L \times L$ with $L = 128$. Initially each grid point takes the value of the spatially homogeneous stable steady state plus a small local random perturbation (of order 10 % of the value of the steady state). Discretize the Laplacian in Eq. (1) and implement periodic boundary conditions. Describe your implementations. For a reliable integration, set the time step to a small value (say, 0.01). Consider four values of D_v: $D_v = 2.3, 3, 5, 9$. For each value of D_v show a snapshot in the form of a heat map of the spatial distribution of u during a transient state (say, after 1000 iterations), as well as when the system has reached a spatially inhomogeneous steady state. Make sure to use the same color range in all heat maps. Describe the patterns observed. What is the effect of increasing D_v?**

Our domain was discretized into a grid of size $L \times L$ with $L = 128$ per the requirement above. The system (1) was then discretized using the Finite-difference method, where, with a five-point stencil, the Laplacian operator is approximated as

$$\frac{c_{i-1,j} + c_{i,j-1} + c_{i+1,j} + c_{i,j+1} - 4c_{i,j}}{h^2} \tag{10}$$

where c denotes the concentration u or v and for the sake of numerical stability the spatial step size between two grid points is $h = 1$ in this case.

The boundary conditions here are periodic Dirichlet boundary conditions:

$$\begin{cases} c(0, j, t) = c(L, j, t) \\ c(i, 0, t) = c(i, L, t). \end{cases} \tag{11}$$

The system (1) was simulated using Python 3 and solved using an iterative approach similar to the Gauss-Seidel method rather than the more explicit way by using, say, Conjugate Gradient Descent for solving this system of partial differential equations on the form $\boldsymbol{Ax} = \boldsymbol{b}$.

With each iteration in time the system was integrated numerically using the Euler method with a small time step of $\mathrm{d}t = 0.01$.

The boundary conditions were enforced using a modulo-approach where simply

$$0 \equiv L \pmod{L} \iff L \equiv 0 \pmod{L}. \tag{12}$$

This approach made using a border around our computational domain not necessary and is also faster as it does not require copying arrays back and forth.

Regarding the results heatmaps were generated using a constant colorbar for better referencing.

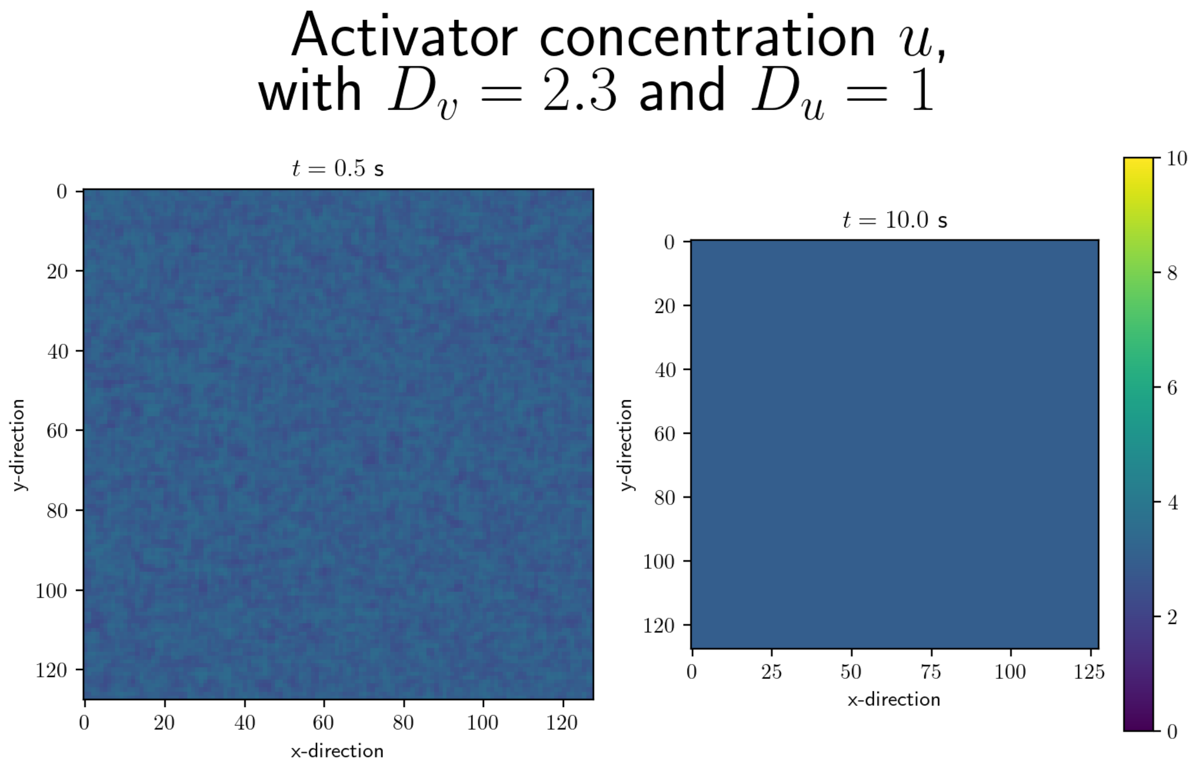

Figure 2: The concentration of the activator u using a diffusion coefficient $D_v = 2.3$.

For the case where $D_v = 2.3$ this is below the critical value for D_v. The system will be stable here. u is initialized with perturbations around the steady state $u^* = a = 3$ as can be seen from the colorbar. After only 10 s the concentration u has homogenized. This can be seen in Figure 2

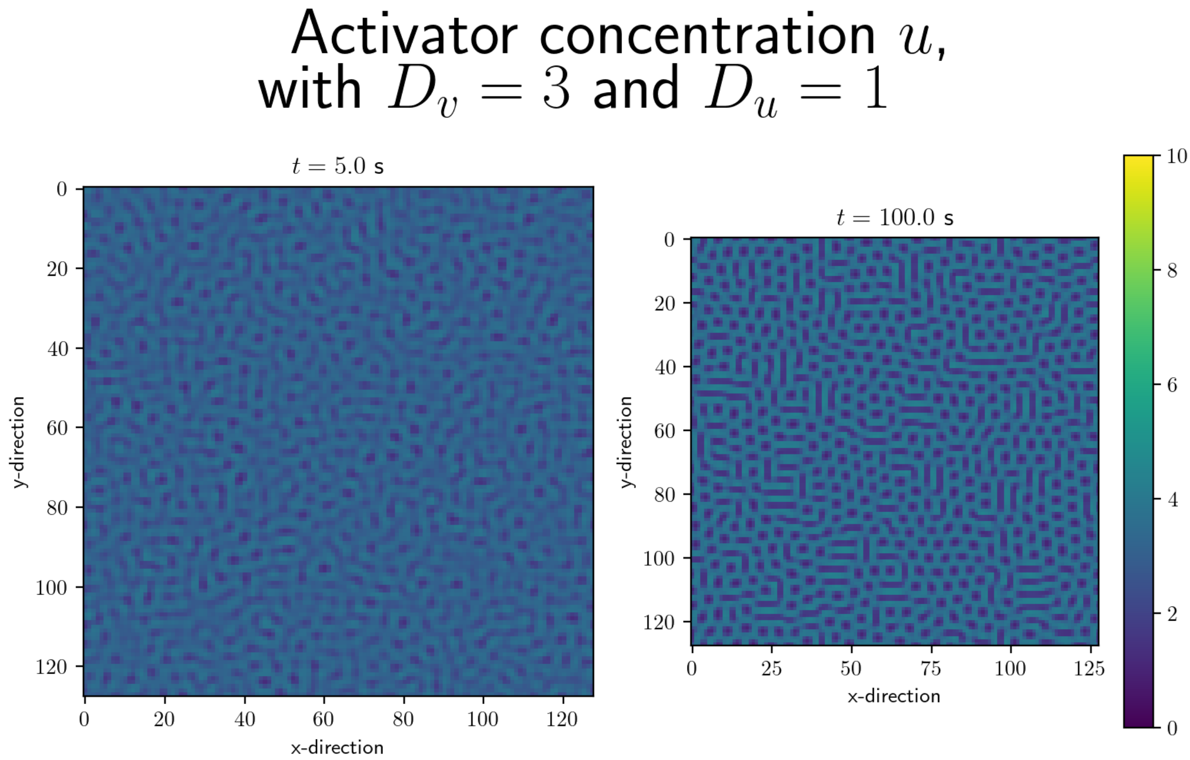

Figure 3: The concentration of the activator u using a diffusion coefficient $D_v = 3$.

When $D_v = 3$ this is now above the critical value for D_v. The system will start exhibiting diffusion-driven instability. However, the reaction rate is slow, but after 100 s the concentration u has formed patterns and reached a spatially inhomogeneous steady state. This can be seen in Figure 3

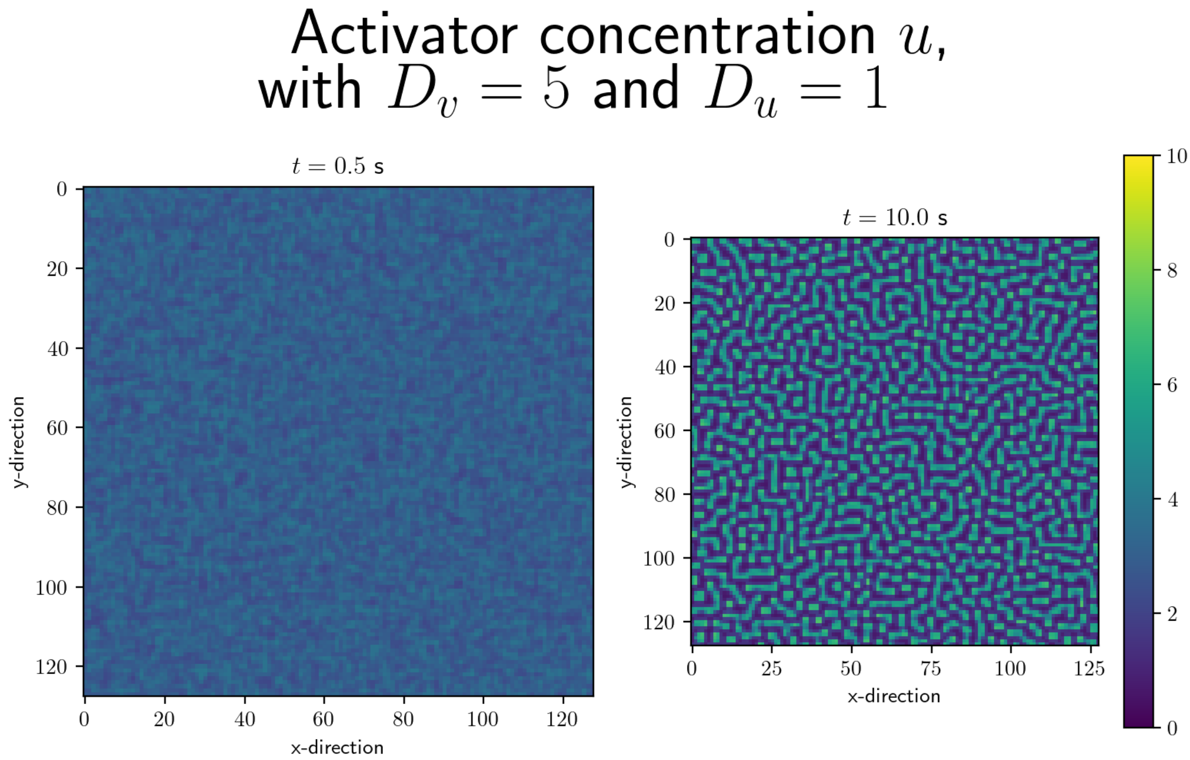

Figure 4: The concentration of the activator u using a diffusion coefficient $D_v = 5$.

With $D_v = 5$ the reaction rate is both much faster and also reaches further into the inhomogeneity. The concentration variance is now larger. This can be seen in Figure 4

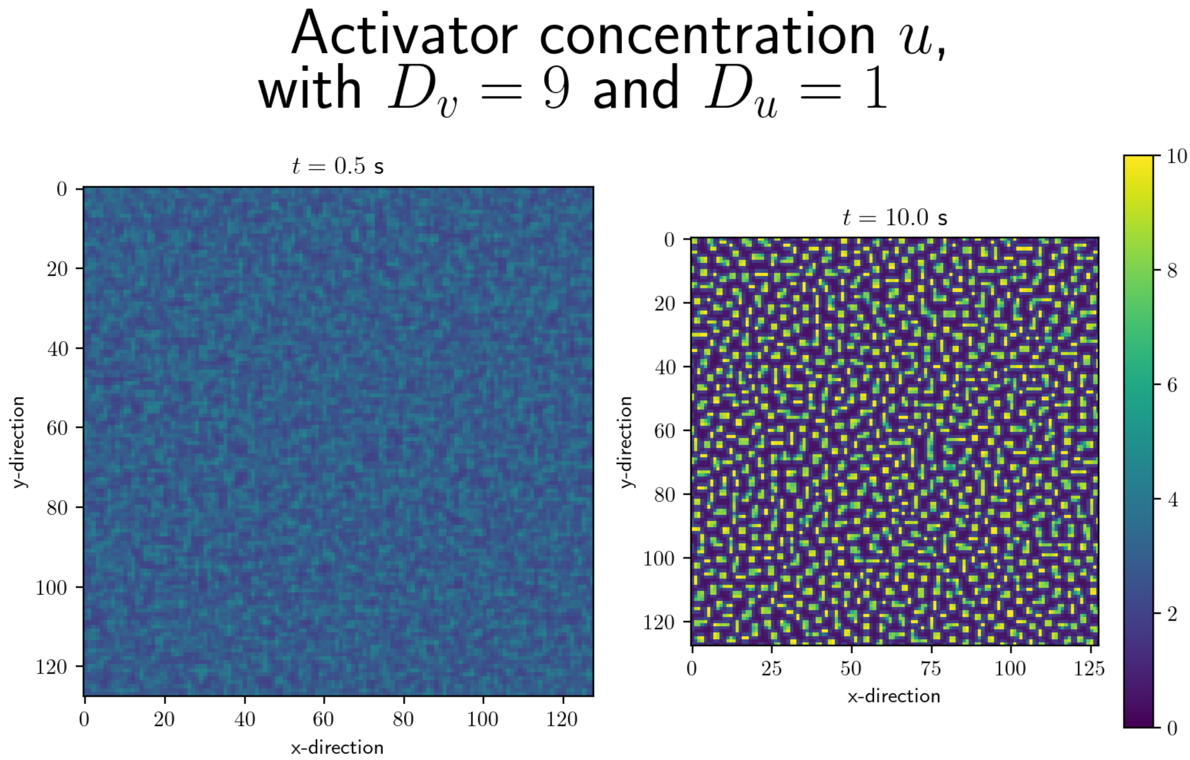

Figure 5: The concentration of the activator u using a diffusion coefficient $D_v = 9$.

Finally, when $D_v = 9$ the system reaches a strong inhomogeneous steady state with a very large variance of concentration in the domain. This can be seen in Figure 5

References

Gustafsson, Kristian (2023). *Lecture notes - FFR110 - Computational Biology.*

Appendix

Python code for simulations of section c)

```
import numpy as np
import scipy.sparse as sp
import matplotlib.pyplot as plt
import matplotlib as mpl
from tqdm.notebook import tqdm
plt.rcParams['text.usetex'] = True
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

L = 256
dt = 0.01
oneOverHSq = 1**-2
nIterations = 1000
Dvs = np.array([2.3, 3, 5, 9])
```

```
Du = 1
a, b = 3, 8
pertSize = 0.1
perturbation = 1 - np.random.uniform(low=-pertSize, high=pertSize, size=(
    ↪ L, L))
u, v = a * perturbation, b / a * perturbation



def RunSolver(Dv, nIterations, L, oneOverHSq, u, v, t1, t2):
    laplacianPreAllocation = np.zeros((L, L))

    def Laplacian(oldC, newL, L):
        for i in range(L):
            for j in range(L):
                newL[i, j] = (
                    oldC[(i + 1) % L, j]
                    + oldC[(i - 1) % L, j]
                    + oldC[i, (j + 1) % L]
                    + oldC[i, (j - 1) % L]
                    - 4 * oldC[i, j]
                )
        return newL
    figsize = 5
    fig, axes = plt.subplots(nrows=1, ncols=2, tight_layout=True, figsize
        ↪ =(figsize * 1.61, figsize))
    def Plot(index, fig, axes):
        plt.tight_layout()
        axes[index].set_xlabel(r'x-direction')
        axes[index].set_ylabel(r'y-direction')
        normalize = mpl.colors.Normalize(vmin=0, vmax=10)
        im = axes[index].imshow(u, norm=normalize)
        if index == 1:
            fig.suptitle(r'Activator concentration $u$, \\ with $D_v={}$
                ↪ and $D_u={}$'.format(Dvs[0], Du), fontsize=30)
            axes[index].set_title(r'$t={}$ s'.format(t2))
            fig.colorbar(im)
        else:
            axes[index].set_title(r'$t={}$ s'.format(t1))

    for t in tqdm(range(nIterations)):
        du_dt = a - (b + 1) * u + u ** 2 * v + Du * Laplacian(u,
            ↪ laplacianPreAllocation, L) * oneOverHSq
        dv_dt = b * u - u ** 2 * v + Dv * Laplacian(v,
            ↪ laplacianPreAllocation, L) * oneOverHSq
        u = u + du_dt * dt
        v = v + dv_dt * dt
        if t == int(0.05 * nIterations):
```

```
            Plot(0, fig, axes)
    Plot(1, fig, axes)

Dvs = np.array([2.3, 3, 5, 9])
Dvs = [2.3]
nIterations = 1000
t1 = nIterations * 0.05 * dt
t2 = nIterations * dt

RunSolver(Dvs, nIterations, L, oneOverHSq, u, v, t1, t2)

Dvs = np.array([2.3, 3, 5, 9])
Dvs = [3]
nIterations = 10000
t1 = nIterations * 0.05 * dt
t2 = nIterations * dt
RunSolver(Dvs, nIterations, L, oneOverHSq, u, v, t1, t2)

Dvs = np.array([2.3, 3, 5, 9])
Dvs = [5]
nIterations = 1000
t1 = nIterations * 0.05 * dt
t2 = nIterations * dt
RunSolver(Dvs, nIterations, L, oneOverHSq, u, v, t1, t2)

Dvs = np.array([2.3, 3, 5, 9])
Dvs = [9]
nIterations = 1000
t1 = nIterations * 0.05 * dt
t2 = nIterations * dt
RunSolver(Dvs, nIterations, L, oneOverHSq, u, v, t1, t2)
```