Problem set 2, Task 2
Diffusion driven instability
Computational Biology
FFR110/FIM740

Mattias Berg, Konstantinos Zakkas

February 26, 2021

$$\frac{\partial u}{\partial t} = a - (b+1)u + u^2 v + D_u \nabla^2 u \tag{1}$$
$$\frac{\partial v}{\partial t} = bu - u^2 v + D_v \nabla^2 v \tag{2}$$

a.)

We want to find the steady states of the system. We neglect the diffusion part (setting D_u and D_v to zero). To do find the fixed point we sett $\frac{\partial u}{\partial t} = \frac{\partial v}{\partial t} = 0$. We start with finding v in terms of u using Equation 2

$$\begin{aligned} 0 &= bu - u^2 v \\ u^2 v &= bu \\ v &= \frac{b}{u} \end{aligned}$$

We put this into Equation 1

$$\begin{aligned} 0 &= a - (b+1)u + u^2 \left(\frac{b}{u}\right) \\ 0 &= a - bu - u + bu \\ u^* &= a \Rightarrow v^* = \frac{b}{a} \end{aligned} \tag{3}$$

The linear stability then is determined by the eigenvalues λ of the linearised Jacobian matrix

$$A = \begin{vmatrix} -1 - b + 2uv & u^2 \\ b - 2uv & -u^2 \end{vmatrix} \tag{4}$$

Thus for $u = u^*$ and $v = v^*$ we get

$$A = \begin{vmatrix} b-1 & a^2 \\ -b & -a^2 \end{vmatrix} \tag{5}$$

The steady state will be stable when the following conditions are satisfied

$$\begin{aligned} trA &< 0 \\ detA &> 0 \end{aligned}$$

For the first condition we have

$$\begin{aligned} b - 1 - a^2 &< 0 \\ b &< 1 + a^2 \end{aligned} \tag{6}$$

and for the second condition

$$\begin{aligned} -(b-1)a^2 - (-ba^2) &> 0 \\ -2ba^2 + a^2 + ba^2 &> 0 \\ a^2 &> 0 \end{aligned} \tag{7}$$

This gives us that we have a stable steady state if $a \neq 0$ and $b < 1 + a^2$, as found in Equation 6 and 7

b.)

We want to find when D_v on the the homogeneous stable steady state have a diffusion-driven instability. We use equation (8) and (9) from lecture note 7 (CompBioLecture7.pdf) to find when the waves become unstable;

$$(dJ_{11} + J_{22})^2 - 4d \det\mathbb{J} \geq 0 \tag{8}$$

$$dJ_{11} + J_{22} > 0 \tag{9}$$

using equation 5 and 7 we get the following parameter values;

$$J_{11} = \frac{\partial \frac{\partial u^*}{\partial t}}{\partial u} = b - 1 \Rightarrow \ b = 8 \ \Rightarrow J_{22} = 7 \tag{10}$$

$$J_{22} = \frac{\partial \frac{\partial v^*}{\partial t}}{\partial v} = -a^2 \Rightarrow \ a = 3 \ \Rightarrow J_{22} = -9 \tag{11}$$

$$\det\mathbb{J} = a^2 \Rightarrow \ a = 3 \ \Rightarrow \ \det\mathbb{J} = 9 \tag{12}$$

inserting the parameters in equation 8, gains the following (point when it becomes unstable);

$$
\begin{aligned}
(d*7-9)^2 - 4d*9 &= 0 \\
49d^2 - 126d + 81 - 36d &= 0 \\
49d^2 - 162d + 81 &= 0 \\
d &= \frac{-(-162) \pm \sqrt{(-162)^2 - 4*49*81}}{2*49} \\
d &= \frac{162 \pm \sqrt{26244 - 15876}}{98} \\
d &= \frac{162 \pm \sqrt{10368}}{98} \\
d &= \frac{162 \pm 101.823}{98} \\
d_1 &\approx 2.692 \\
d_2 &\approx 0.614
\end{aligned}
$$

Using the gained d_1 and d_2 we need to check if they fulfills equation 9 also;

$$
\begin{aligned}
d_1 * 7 - 9 > 0 \Rightarrow 18.844 > 9 \Rightarrow \text{True} \\
d_2 * 7 - 9 > 0 \Rightarrow 4.298 > 9 \Rightarrow \text{Not true}
\end{aligned}
$$

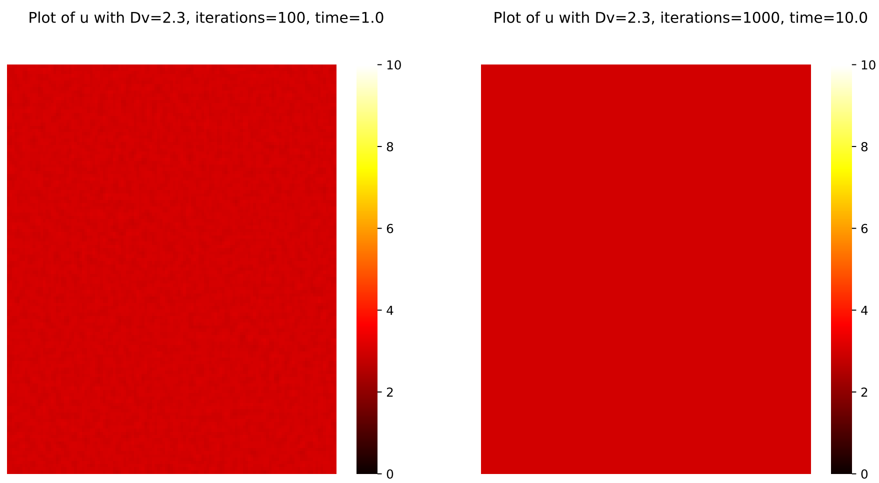
Hence the diffusion-driven instability start at d_1 and goes to ∞. As D_u is set to 1, D_v value is the same as d_1 for this case.

c.)

For $D_v = (2.3, 5, 9)$ iterations numbers 100/1 000 was chosen as transient/stable state for the figures, larger iterations (tests up to 10 000 iteration gave no visible difference), see Figure 1, 3 and 4. $D_v = 3$ have some difference in the pattern between 1000 and 10 000 iterations, and we can see that some part have not yet formed after 1 000 iterations compared to the stable case after 10 000 iterations, and therefor 1 000/10 000 iterations was chosen for the Transient/stable state for Figure 2.

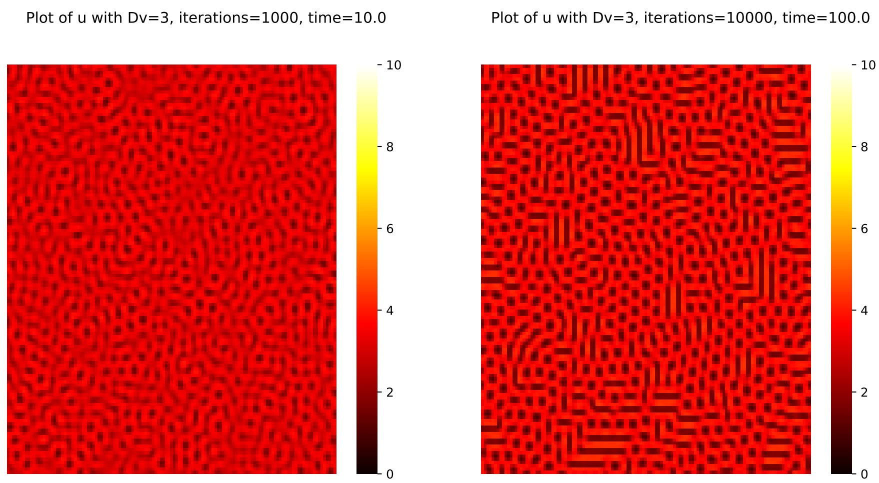
$D_v = 2.3$ in Figure 1 we see a homogeneous distribution, which is expected based on the calculation in b.) where the braking point was found to be $D_v \approx 2.692$. The other simulations. With D_v set above $D_v \approx 2.692$, we can see a clear inhomogeneous distribution in Figure 2, 3 and 4 at their stable state. We can also see that large D_v gives a larger variance between the two chemicals.

The close one is to the stable diffusion, the longer it seems to take for the system to become stable. In Figure 5, showing a D_v value close to where the diffusion-driven instability starts, the stable state pattern wont be clearly seen until iterations somewhere between 10 000 and 100 000, compared to the systems with higher values of D_v that only need somewhere between 100 and 1 000 iterations.



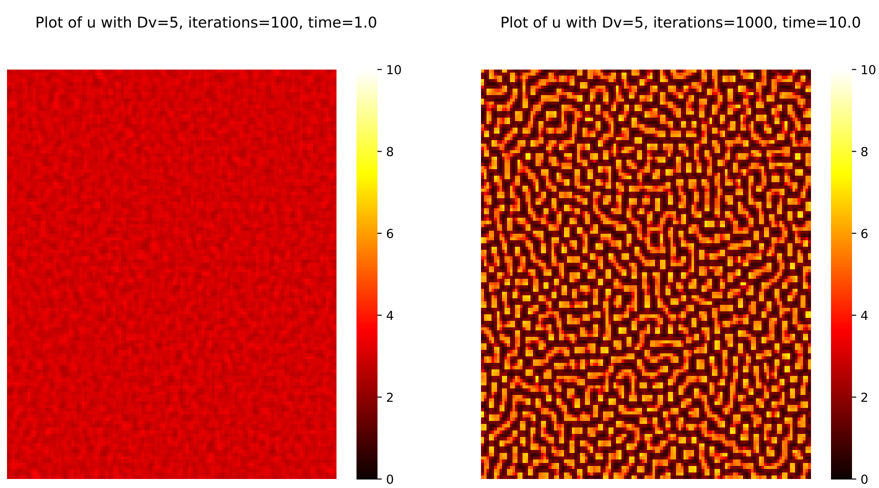
(a) Transient (b) Stable state

Figure 1: Spatial distribution of u in form of heatmaps for $D_v = 2.3$



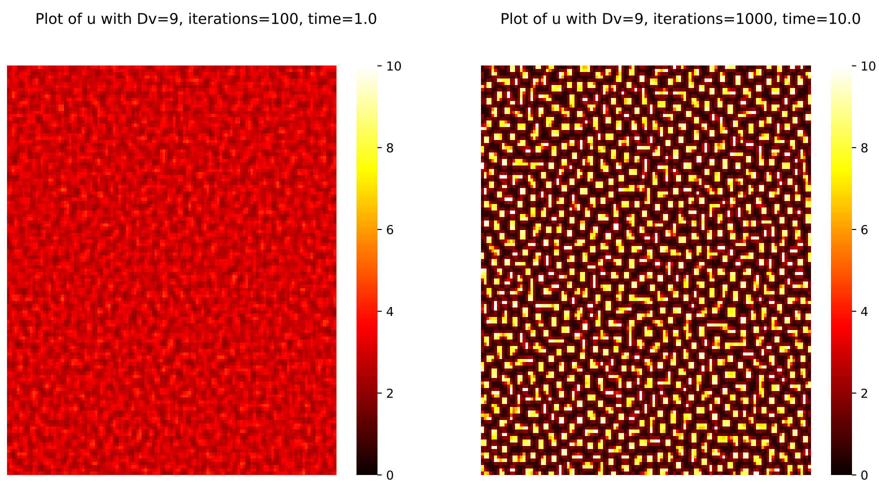
(a) Transient (b) Stable state

Figure 2: Spatial distribution of u in form of heatmaps for $D_v = 3$



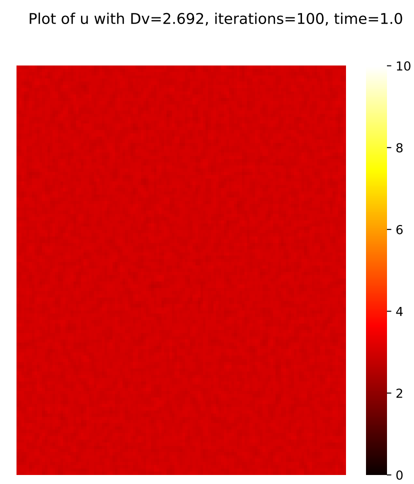
(a) Transient (b) Stable state

Figure 3: Spatial distribution of u in form of heatmaps for $D_v = 5$

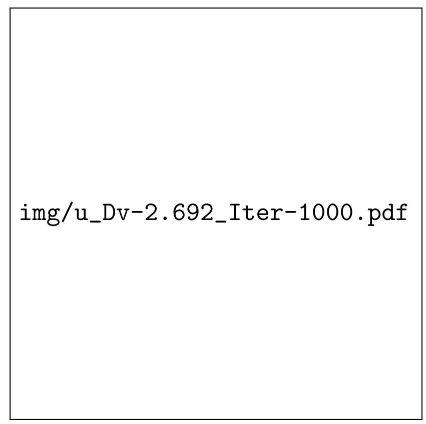


(a) Transient (b) Stable state

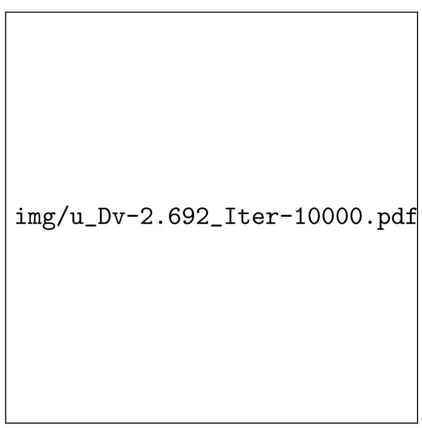
Figure 4: Spatial distribution of u in form of heatmaps for $D_v = 9$



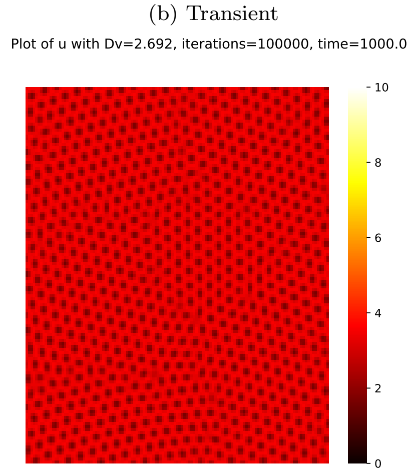
(a) Transient



(b) Transient



(c) Transient



(d) Stable state

Figure 5: Spatial distribution of u in form of heatmaps at point close to diffusion-driven instability ($D_v = 2.692$)

Appendix

```python
#Course : FFR110/FIM740 Computational Biology
#Problem : Problem set 2, Task 2. Diffusion driven instability
#Code :  Python 3.8.5

import numpy as np
import matplotlib.pyplot as plt
import math
import seaborn as sns
from numpy import genfromtxt

import time

def Calc_BelousovZhabotinsky(a, b, Du, Dv, u, v, dTime = 0.01):
    yLenght, xLenght = u.shape
    u_new = np.zeros((yLenght,xLenght))
    v_new = np.zeros((yLenght,xLenght))

    h = 1

    for y in range(1,yLenght-1):

        for x in range(1,xLenght-1):

            LaplacianU=(u[y,x+h]+u[y,x-h]+u[y+h,x]+u[y-h,x]-4*u[y,x])/(h**2)
            LaplacianV=(v[y,x+h]+v[y,x-h]+v[y+h,x]+v[y-h,x]-4*v[y,x])/(h**2)

            dudt=a-(b+1)*u[y,x]+v[y,x]*u[y,x]**2+Du*LaplacianU
            dvdt=b*u[y,x]-v[y,x]*u[y,x]**2+Dv*LaplacianV


            u_new[y,x] = u[y,x]+dTime*dudt
            v_new[y,x] = v[y,x]+dTime*dvdt

            if x == 1:
                u_new[y,-1] = u_new[y,x]
                v_new[y,-1] = v_new[y,x]
            elif x == yLenght-2:
                u_new[y,0] = u_new[y,x]
                v_new[y,0] = v_new[y,x]

            if y == 1:
                u_new[-1,x] = u_new[y,x]
                v_new[-1,x] = v_new[y,x]
            elif y == yLenght-2:
                u_new[0,x] = u_new[y,x]
                v_new[0,x] = v_new[y,x]



    return u_new, v_new

def SaveU(Dv, u, Iteration, dTime = 0.01):
    fileName="u_Dv-%s_Iter-%s.csv" % (Dv, Iteration)
    np.savetxt(fileName, u, delimiter=";")


    vmin=0
    vmax=10

    fig, ax = plt.subplots(figsize=(6,6))

    titleString = "Plot of u with Dv=%s, iterations=%s, time=%s" % (Dv, Iteration, Iteration*dTime)
    fig.suptitle(titleString)
    ax = sns.heatmap(u[1:-2,1:-2],cmap="hot", center=(vmin + vmax) / 2.,vmin=vmin, vmax=vmax, yticklabels=False, xtick

    graphName="u_Dv-%s_Iter-%s.pdf" % (Dv, Iteration)
    plt.savefig(graphName,  bbox_inches = 'tight')

def SaveV(Dv, v, Iteration, dTime = 0.01):
    fileName="v_Dv-%s_Iter-%s.csv" % (Dv, Iteration)
    np.savetxt(fileName, v, delimiter=";")


    vmin=0
    vmax=10

    fig, ax = plt.subplots(figsize=(6,6))

    titleString = "Plot of v with Dv=%s, iterations=%s, time=%s" % (Dv, Iteration, Iteration*dTime)
    fig.suptitle(titleString)
    ax = sns.heatmap(v[1:-2,1:-2],cmap="jet", center=(vmin + vmax) / 2.,vmin=vmin, vmax=vmax, yticklabels=False, xtick

    graphName="v_Dv-%s_Iter-%s.pdf" % (Dv, Iteration)
    plt.savefig(graphName,  bbox_inches = 'tight')

def RunProgram(Dv, Iteration):
    xLenght = 128
    yLenght = 128
    a=3
    b=8
```

```
    Du=1

    dTime = 0.01

    #Add two to the lenght to create space on each side of the matrix (for edge cases)
    u = a+(np.random.rand(yLenght+2,xLenght+2)*2-1)*0.1
    v = b/a+(np.random.rand(yLenght+2,xLenght+2)*2-1)*0.1
    for t in range(Iteration):
        u, v = Calc_BelousovZhabotinsky(a, b, Du, Dv, u, v, dTime=dTime)
        if not t%10:
            tString = "Dv=%s, iterations %s out of %s" % (Dv ,t ,Iteration)
            print(tString)
    SaveU(Dv, u, Iteration)

Iteration = 100
Dv=2.3
RunProgram(Dv, Iteration)

Iteration = 1000
Dv=2.3
RunProgram(Dv, Iteration)

Iteration = 10000
Dv=2.3
RunProgram(Dv, Iteration)
```