Universidade de São Paulo
Instituto de Física de São Carlos
Mathematical-Computational
Modeling

Direction Diffusion Equation

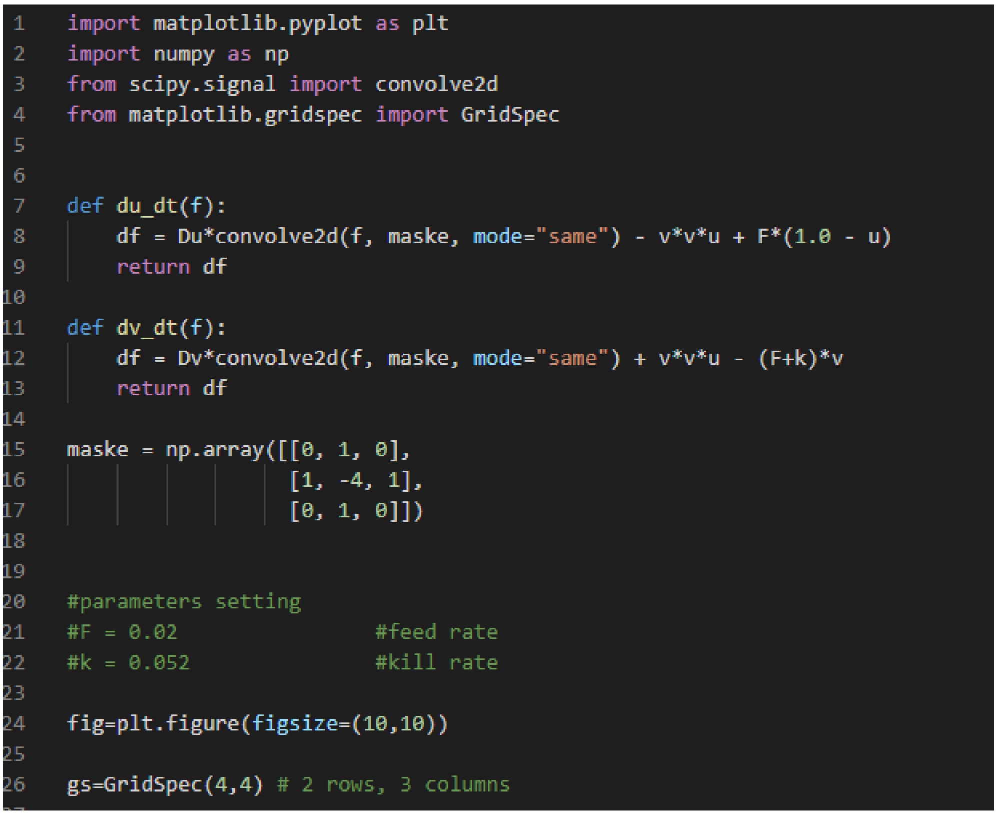
Éverton Luís Mendes da Silva (10728171)

1 Introduction

In this project, new solution patterns of the Gray-Scott equation were approached.

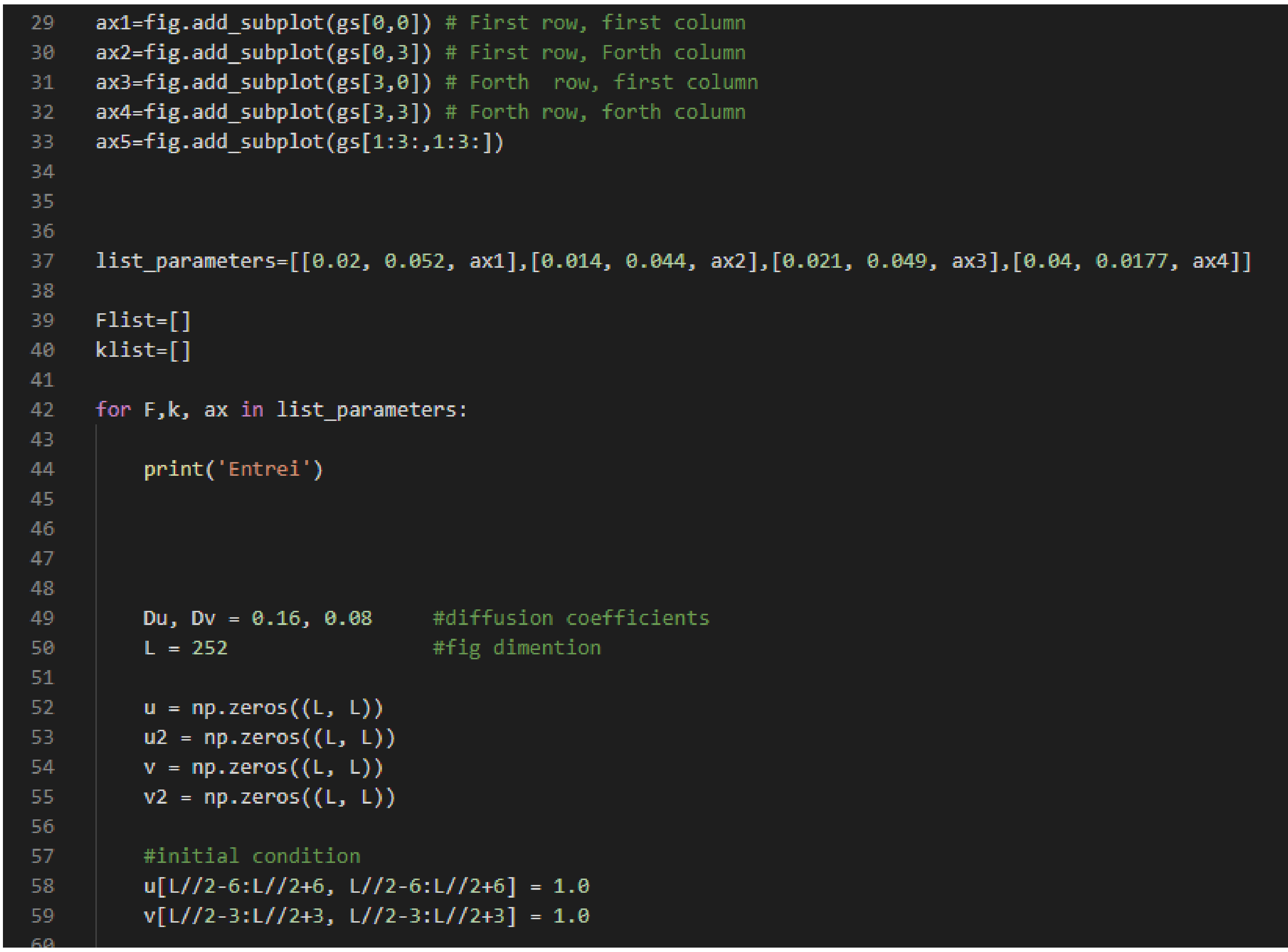
2 Part Two

To make such visualization, some changes were made to the code referenced in the CDT-21.



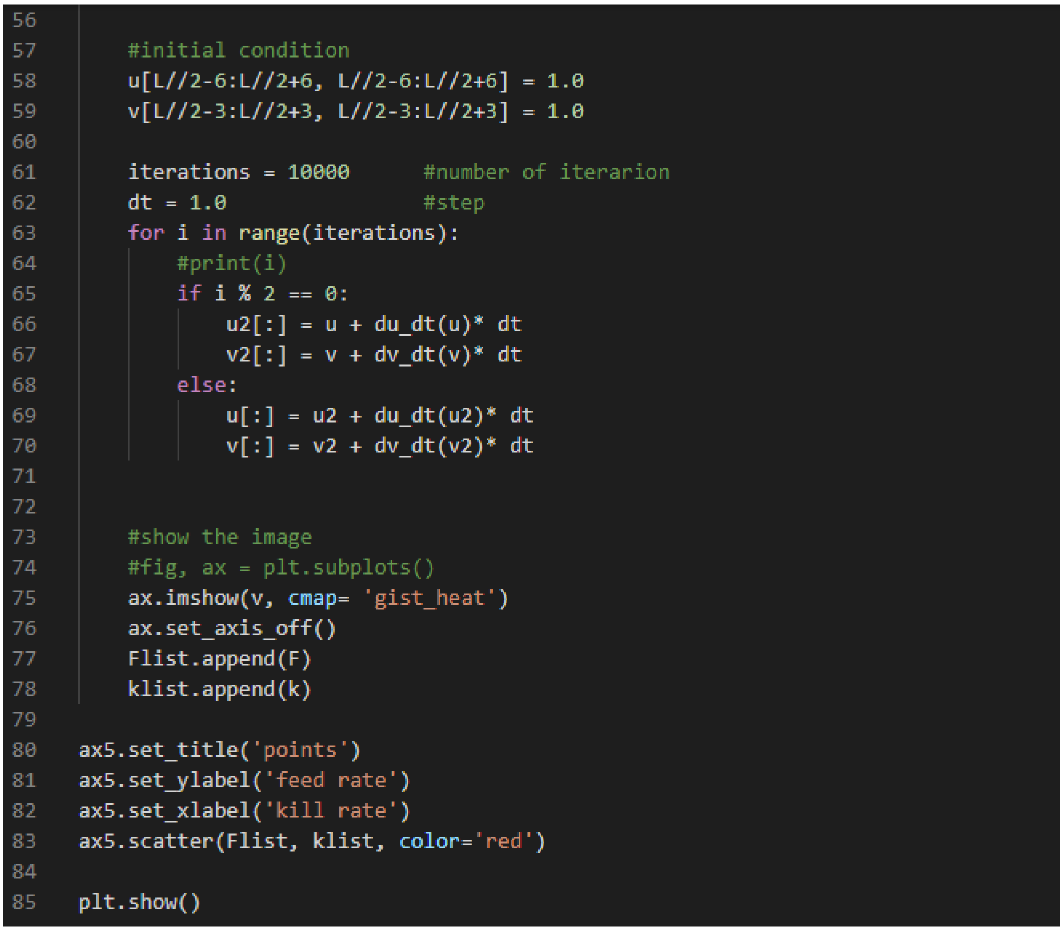
```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import convolve2d
from matplotlib.gridspec import GridSpec


def du_dt(f):
    df = Du*convolve2d(f, maske, mode="same") - v*v*u + F*(1.0 - u)
    return df

def dv_dt(f):
    df = Dv*convolve2d(f, maske, mode="same") + v*v*u - (F+k)*v
    return df

maske = np.array([[0, 1, 0],
                  [1, -4, 1],
                  [0, 1, 0]])


#parameters setting
#F = 0.02                #feed rate
#k = 0.052               #kill rate

fig=plt.figure(figsize=(10,10))

gs=GridSpec(4,4) # 2 rows, 3 columns
```

Figure 1



```
ax1=fig.add_subplot(gs[0,0]) # First row, first column
ax2=fig.add_subplot(gs[0,3]) # First row, Forth column
ax3=fig.add_subplot(gs[3,0]) # Forth  row, first column
ax4=fig.add_subplot(gs[3,3]) # Forth row, forth column
ax5=fig.add_subplot(gs[1:3:,1:3:])



list_parameters=[[0.02, 0.052, ax1],[0.014, 0.044, ax2],[0.021, 0.049, ax3],[0.04, 0.0177, ax4]]

Flist=[]
klist=[]

for F,k, ax in list_parameters:

    print('Entrei')




    Du, Dv = 0.16, 0.08     #diffusion coefficients
    L = 252                 #fig dimention

    u = np.zeros((L, L))
    u2 = np.zeros((L, L))
    v = np.zeros((L, L))
    v2 = np.zeros((L, L))

    #initial condition
    u[L//2-6:L//2+6, L//2-6:L//2+6] = 1.0
    v[L//2-3:L//2+3, L//2-3:L//2+3] = 1.0
```

Figure 2



```
    #initial condition
    u[L//2-6:L//2+6, L//2-6:L//2+6] = 1.0
    v[L//2-3:L//2+3, L//2-3:L//2+3] = 1.0

    iterations = 10000      #number of iterarion
    dt = 1.0                #step
    for i in range(iterations):
        #print(i)
        if i % 2 == 0:
            u2[:] = u + du_dt(u)* dt
            v2[:] = v + dv_dt(v)* dt
        else:
            u[:] = u2 + du_dt(u2)* dt
            v[:] = v2 + dv_dt(v2)* dt


    #show the image
    #fig, ax = plt.subplots()
    ax.imshow(v, cmap= 'gist_heat')
    ax.set_axis_off()
    Flist.append(F)
    klist.append(k)

ax5.set_title('points')
ax5.set_ylabel('feed rate')
ax5.set_xlabel('kill rate')
ax5.scatter(Flist, klist, color='red')

plt.show()
```

Figure 3

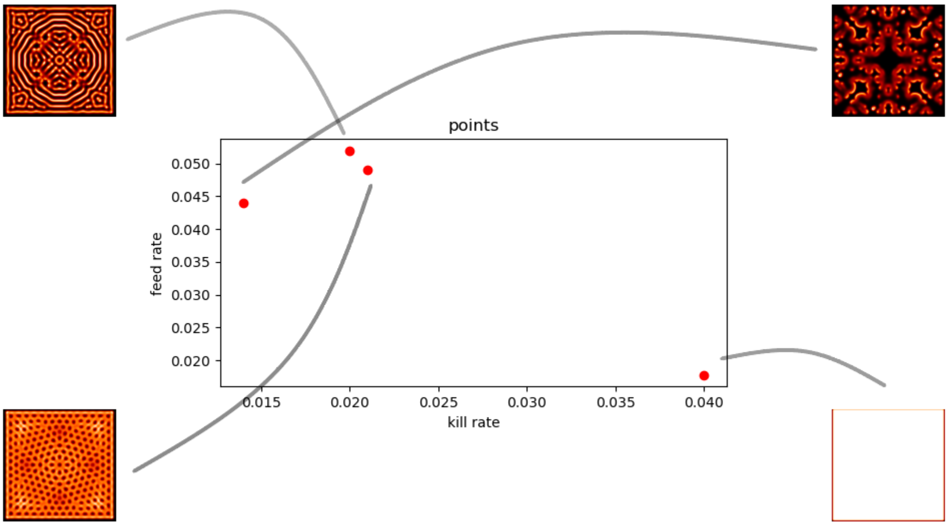


Figure 4

3 References

[1] da Silva, Éverton Luís Mendes. Code and Images used is this PDF. ¡https://github.com/evertonmendes/Mathematical-Computational-Modeling¿