

BIOEN 537: Final PROJECT

Turing Pattern Simulator

Zihao Song
Dec. 2nd 2024

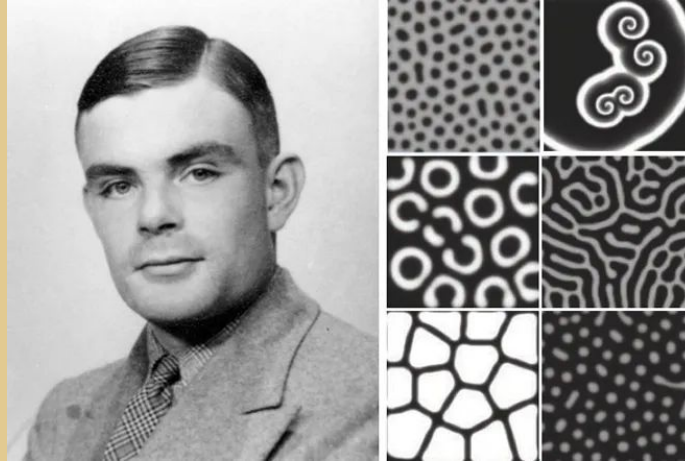


BE BOUNDLESS



Background

Development and Biological Pattern: How do organisms make up these patterns, and how do these patterns break symmetrically from a homogeneous system?[1]



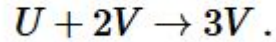
Allen Turing's
'The Chemical Basis of Morphogenesis'[2]

: morphogens diffusion and system instability
cause the symmetry breaking and pattern
formation

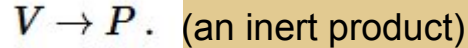


Background

The formation of Turing pattern can be simulated by reaction-diffusion models
Such as Gray-Scott Model[3]



feed rate: F



kill rate: k

$$\frac{\partial U}{\partial t} = D_u \nabla^2 U - UV^2 + F(1 - U)$$

$$\frac{\partial V}{\partial t} = D_v \nabla^2 V + UV^2 - (F + k)V$$



Problem statement



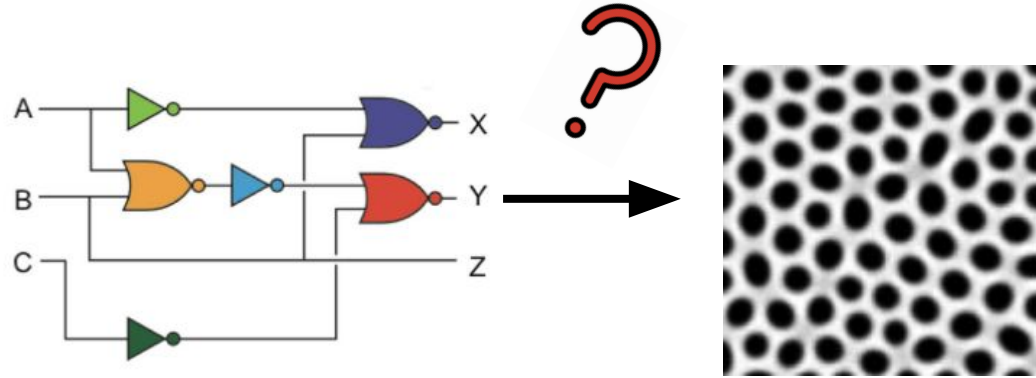
Many developmental biologists are exploring the **molecular mechanisms** under pattern formation. System and synthetic biologists want to use artificial system and circuits to reproduce Turing patterns.

In a reaction diffusion model, small changes in parameters can lead to large changes in output pattern, this makes it difficult for them to make accurate wet lab experiments.

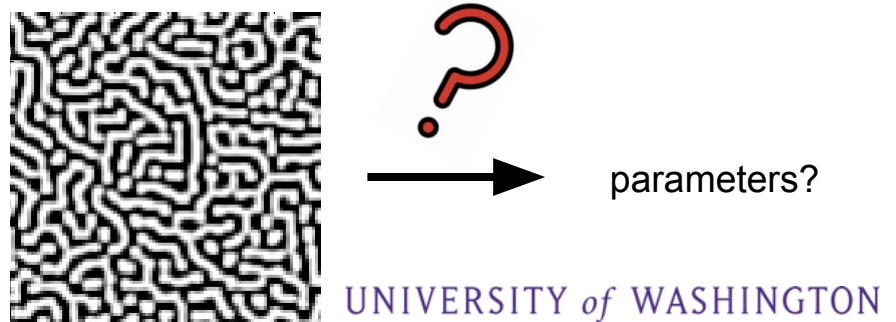
→ **in silico simulator of Turing pattern and a parameter sweeper of all possible patterns with different parameters.**

Use cases

1. a synthetic biologist wants to see if designed genetic circuits (knows all parameters) can form patterns



2. a developmental biologist found a pattern in cell differentiation, and one or two parameters is difficult to measure, and wants to learn the precise range of parameters



Methodology



Developed a python package, for pattern-learning biologists to do pattern simulation and parameter sweep easily in a python environment.

Used *numpy* to generate the PDE of reaction-diffusion model

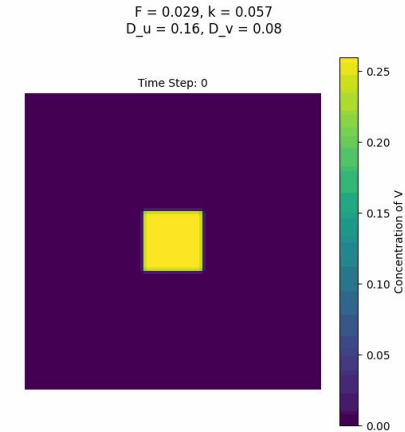
Used *matplotlib* to plot the pattern figures

Designed with several functions to allow single-parameter and multi-parameter sweeps, supporting exploration of pattern dynamics.

simulator

sweeper

Results -simulator demo



```
from Turing_Pattern_Simulator import simulator
```

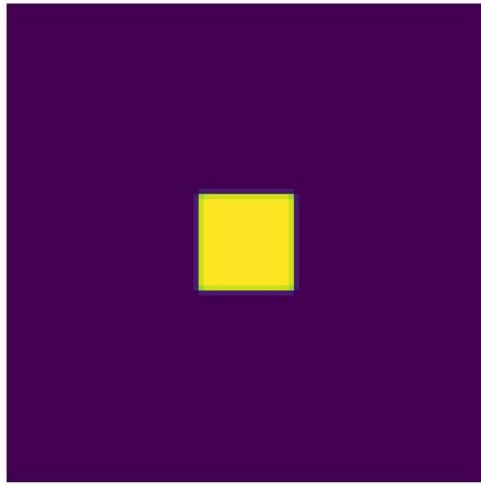
```
simulator(k=0.057,F=0.029,D_u = 0.16, D_v = 0.08, time_steps= 3000,  
grid_size = 100, output_type="gif",fps = 10, frame_interval =  
100,add_noise=False,nois_U_to_V=0.5,nois_V_to_U = 0.05,  
,filename="maze")
```

- parameter definition
- reaction settings
- output type and settings
- noise settings
- filename

Results -Use case1 without noise (gif)

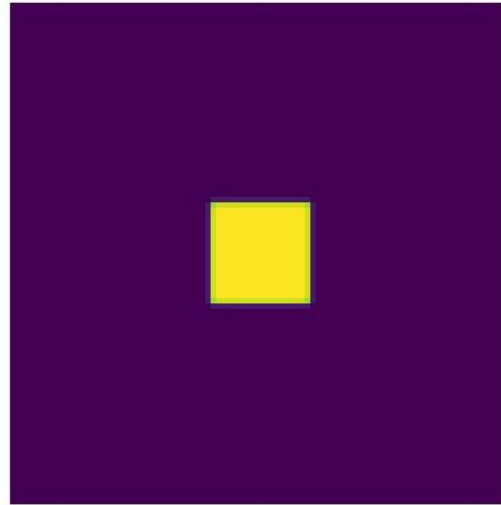
$F = 0.029$, $k = 0.057$
 $D_u = 0.16$, $D_v = 0.08$

Time Step: 0



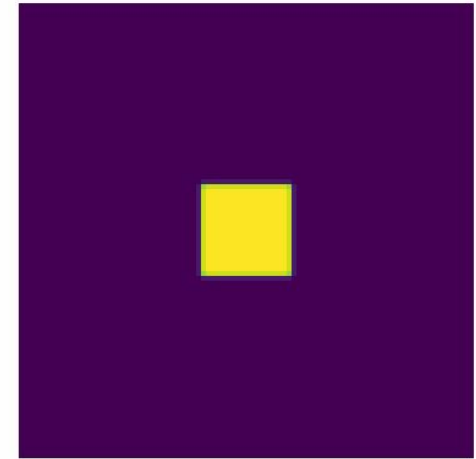
$F = 0.014$, $k = 0.054$
 $D_u = 0.16$, $D_v = 0.08$

Time Step: 0



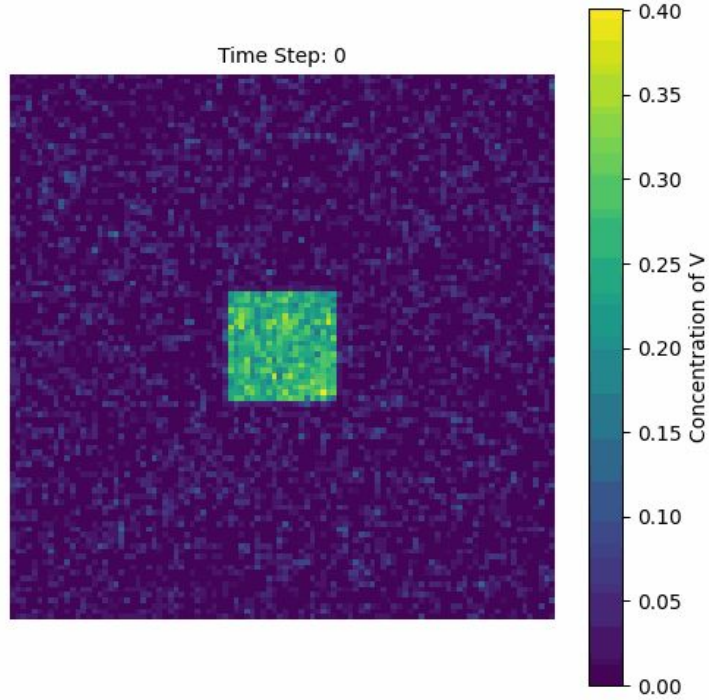
$F = 0.03$, $k = 0.052$
 $D_u = 0.16$, $D_v = 0.08$

Time Step: 0

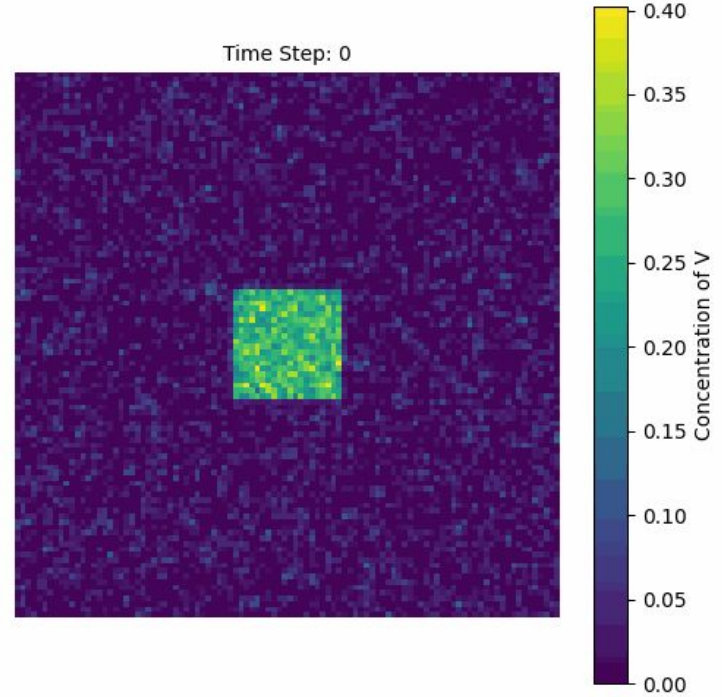


Results -Use case1 - with noise (gif)

$F = 0.029$, $k = 0.057$
 $D_u = 0.16$, $D_v = 0.08$







$F = 0.014$, $k = 0.054$
 $D_u = 0.16$, $D_v = 0.08$



Results -sweeper 1d demo

```
from Turing_Pattern_Simulator import sweeper_1d





sweeper_1d(param_name='F', param_range=(0.02, 0.07, 0.01),
simulator_args={'k': 0.06, 'D_u': 0.16, 'D_v':
0.08}, output_dir="1d_F_sweep")
```

-  parameter needed to sweep
-  parameter range and step length
-  simulator settings
-  output pathway

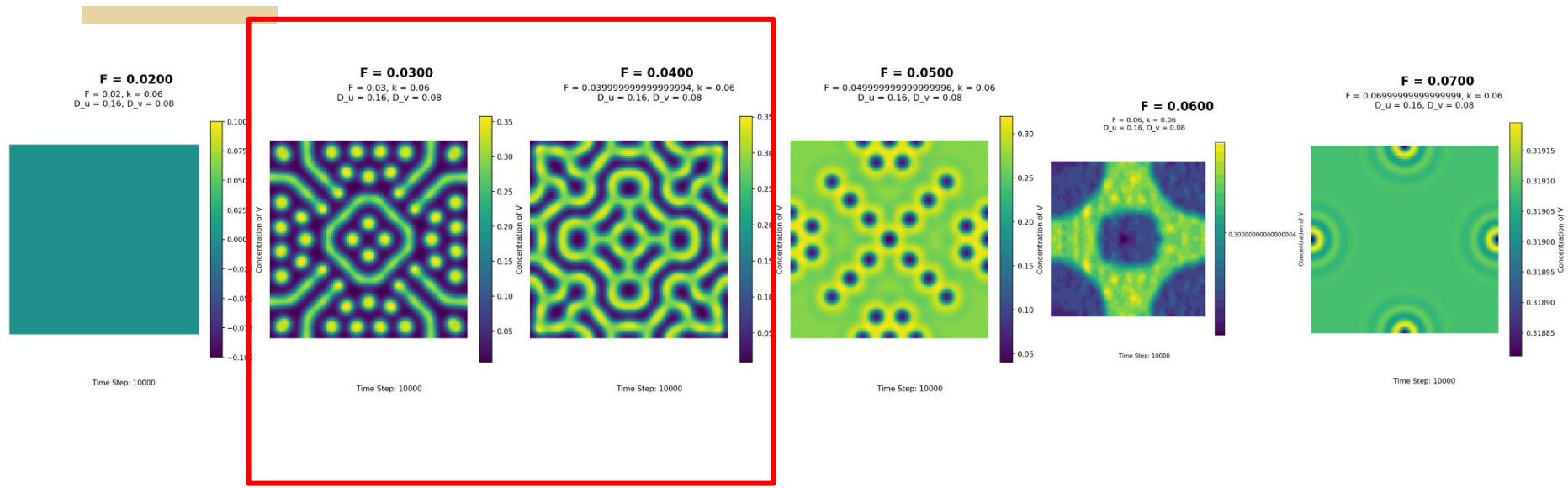
Results -sweeper 2d demo

```
from Turing_Pattern_Simulator import sweeper_2d
```

```
sweeper_2d(param1_name='F', param1_range=(0.02, 0.06,  
0.01),param2_name='k',param2_range=(0.02, 0.06, 0.01),  
simulator_args={'D_u': 0.16,'D_v': 0.08,'time_steps':  
3000},output_dir="2d_F_k_sweep")
```

-  parameter1 settings
-  parameter2 settings
-  simulator settings
-  output pathway

Results -sweeper 1d Use case2



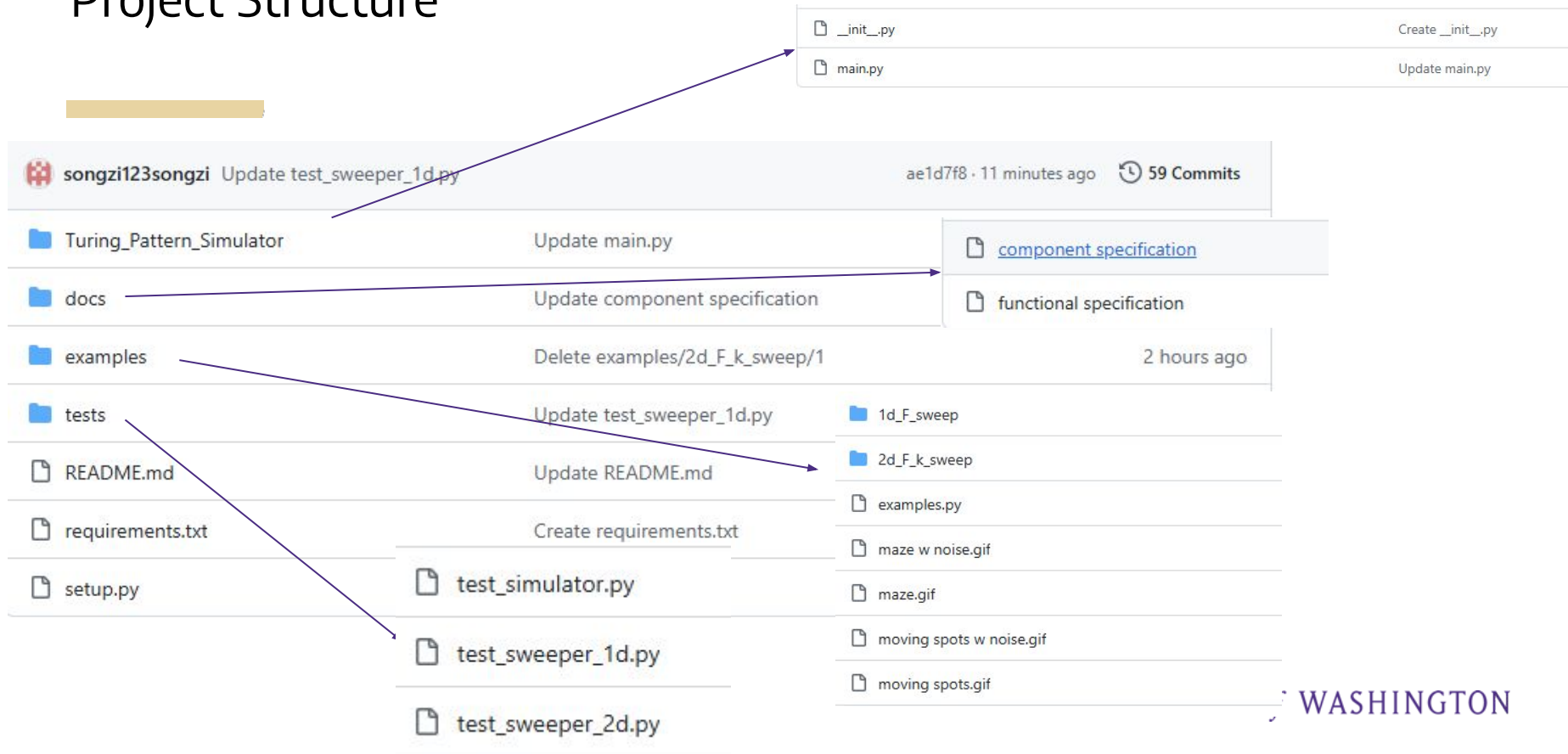
sweep of F, from 0.02 to 0.07, step length 0.01

K



UNIVERSITY of WASHINGTON

Project Structure



```
pip install Turing_Pattern_Simulator
```

Project Structure - readme

Turing-Pattern-Simulator

This model is used for simulating Turing pattern by reaction diffusion model.

A Turing pattern is a spatially organized, stable pattern that emerges from the interaction between two or more chemical species undergoing reaction and diffusion. It was first described by Alan Turing in his 1952 paper on morphogenesis. The system typically involves an activator and an inhibitor, where:

1. Activator promotes both its own production and that of the inhibitor.
2. Inhibitor suppresses the activator's production.
3. Diffusion plays a key role: the inhibitor diffuses faster than the activator, leading to localized concentrations of the activator. Mathematically, Turing patterns emerge when specific parameter conditions cause the homogeneous steady state of the reaction-diffusion system to become unstable, giving rise to periodic spatial patterns.

The Turing pattern appears in chemical reactions, in the formation and development of biological forms. Understanding the Turing pattern is important for understanding biomolecular interactions, biological development, the construction of self-forming systems, and systems biology.

There are many reaction-diffusion models that describe the generation of Turing patterns. A classic example is the Gray-Scott (GS) model, which is widely used to study Turing patterns due to its simplicity and ability to produce complex, self-organizing structures. Details about the function can be seen

John E. Pearson ,Complex Patterns in a Simple System.Science261,189-192(1993).DOI:10.1126/science.261.5118.189

Introduction

Turing-pattern-simulation

This part is for users to generate their own Turing pattern by indicating parameters

How to use: `from Turing_Pattern_Simulator import simulator`

```
filename="your_file_name"

#Simulation part:
k = # removal rate of V, better give from 0.01 to 0.1,
F = # feed rate of U into the system, better from 0 to 0.1,
D_u = # diffusion coefficients for U, default = 0.16,
D_v = # diffusion coefficients for V, default = 0.08,
time_steps= # time steps to do the reaction, default = 10000,
add_noise= # Add the noise of U and V on the simulation, True or False, default = false,
noise_U_to_V = # noise intensity of U, default = 0.05,
noise_V_to_U = # noise intensity of V, default = 0.05,

#Visualization part
grid_size = # default = 100
output_type= # "gif" or "png", gif can give you a motion graph about the diffusion steps, png can give
frame_interval= # frame interval of time steps, default = 100,
fps= # default = 10
)
```

#example:

```
from Turing_Pattern_Simulator import simulator
simulator(k=0.057,F=0.029,output_type="gif",filename="maze",time_steps= 3000 ,add_noise=False)
```


Project Structure - readme

🔗 Turing-pattern parameter sweep

In reaction-diffusion models like the Gray-Scott model, the emergent patterns (e.g., spots, stripes, or oscillations) are highly sensitive to the choice of parameters. Small changes in parameters such as feed rate(F), kill rate (k), and diffusion coefficients can lead to different patterns. Performing a parameter sweep allows us to explore the model's behavior across a range of values, and also estimate some underlying function in molecular developmental biology.

How to use Here, we can do 1d (one parameter) and 2d (two parameters sweep) and automatically generate a matrix for those parameters.

For 1d sweeper:

```
from Turing_Pattern_Simulator import sweeper_1d
sweeper_1d(
    param_name='F', #parameter you want to sweep
    param_range=(0.01, 0.1, 0.03), #(start point, end point, step size)
    simulator_args={'k': 0.06, 'D_u': 0.16, 'D_v': 0.08}, # other parameters you use for sweep
    output_dir=output_dir, #the output folder for your sweep figures)
```



For 2d sweeper:

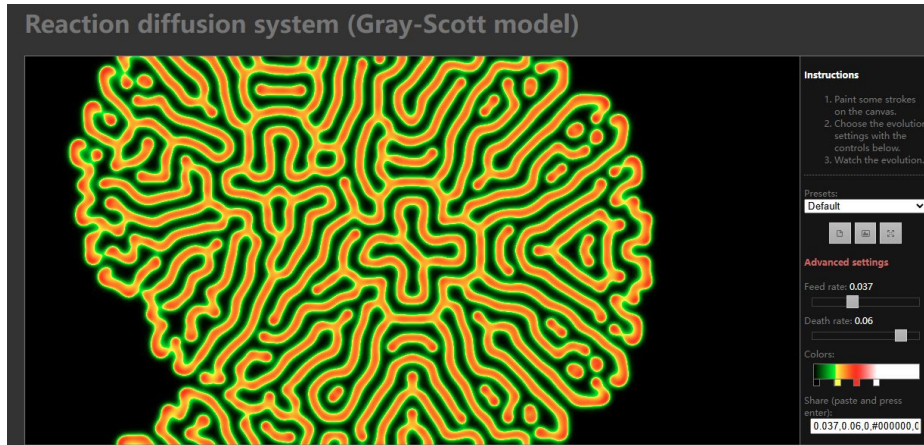
```
from Turing_Pattern_Simulator import sweeper_2d
sweeper_2d(
    param1_name='F', #parameter1 you want to sweep
    param1_range=(0.02, 0.07, 0.01), #(start point, end point, step size)
    param2_name='k', #parameter2 you want to sweep
    param2_range=(0.02, 0.07, 0.01), #(start point, end point, step size)
    simulator_args={'k': 0.06, 'D_u': 0.16, 'D_v': 0.08}, # other parameters you use for sweep
    output_dir=output_dir, #the output folder for your sweep figures)
```



Challenges/Issues/Future work

In some systems, diffusion happens in multi sites, and probably randomly start.
— need to add seed option - select start in 1 site, 2 sites or random sites.

Interactive interface: better to play with and easy to access.[4]



Challenges/Issues/Future work



Multiple models: Not only Gray-scott model.

Integration with Experimental Data: make users to input real biological data

Advanced Parameter Sweeps: Implement Monte Carlo or optimization-based parameter search

Reference



- [1] Turing, Alan. "Turing patterns, 70 years later." *Interface* 18 (2021): 20210034.
- [2] Turing, Alan Mathison. "The chemical basis of morphogenesis." *Bulletin of mathematical biology* 52 (1990): 153-197.
- [3] Pearson, John E. "Complex patterns in a simple system." *Science* 261.5118 (1993): 189-192.
- [4] Neila, Pablo M. *Gray-Scott Model Explorer*. Accessed 2 Dec. 2024, <https://pmneila.github.io/jsexp/grayscott>.