Differential Equations in Bioscience

Simulations of Gierer-Meinhardt Model with Source Density

Shourya Verma
MSc Scientific Computing

[1] Meinhardt, H. (2012) “Turing’s theory of morphogenesis of 1952 and the subsequent discovery of the crucial role of local self-enhancement and long-range inhibition,”*Interface focus*, 2(4), pp. 407–416.

[2] Meinhardt, H. (1993) “A model for pattern formation of hypostome, tentacles, and foot in hydra: how to form structures close to each other, how to form them at a distance,” *Developmental biology*, 157(2), pp. 321–333.

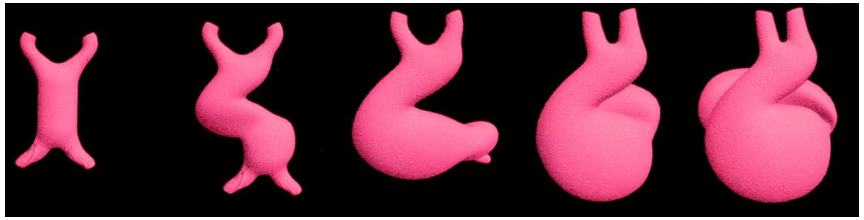
[3] Gierer, A. and Meinhardt, H. (1972) “A theory of biological pattern formation”. *Kybernetik* 12, 30–39

Motivation

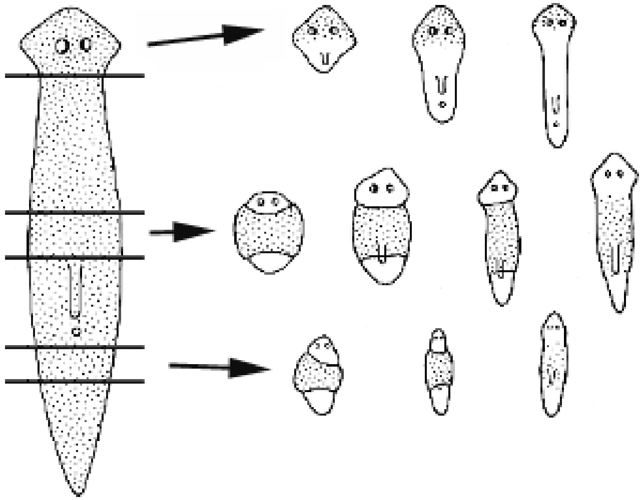
- **Morphogenesis:**
  - Biological process allowing an organism to develop its form
  - Studies mechanism by which distribution of cell space occurs
  - Example: during the process of embryonic development

- **Activator-Inhibitor Systems:**
  - Reaction-diffusion equations
  - Explain many pattern formation processes in nature

- **Why do we study them?**
  - Polarity retention during regeneration in hydra
  - Understanding congenital cardiac malformations



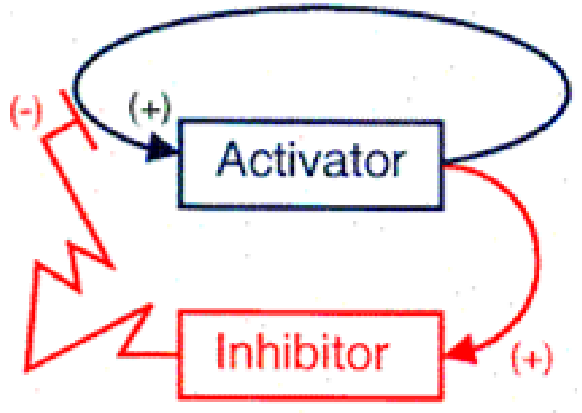
Schleich, Jean-Marc, και Jean-Louis Dillenseger. 'Virtual Imaging for Teaching Cardiac Embryology'. *Circulation* 104.24 (2001): e134–e134. Web.



https://sites.google.com/site/flatwormsjfr2/defense-protection

Gierer-Meinhardt Model 1

- **Activator-Inhibitor System [1]:**
  - Two substances that act on each other
  - Activator stimulates its own production (autocatalysis)
  - Activator also produces the inhibitor
  - Inhibitor represses the production of the activator



Hans Meinhardt (2006) Gierer-Meinhardt model. Scholarpedia, 1(12):1418.

D1-D2 are growth rates, **μ-*ν*** are decay rates, **a-b** are production rates, △ is Laplace operator

$$\partial_t u = D_1 \Delta u + \frac{au^2}{v} - \mu u$$

$$\partial_t v = D_2 \Delta v + bu^2 - \nu v$$

activator **(u)**, inhibitor **(v)** **eq1**

[1] Gierer, A., Meinhardt, H. A theory of biological pattern formation. *Kybernetik* **12,** 30–39 (1972). https://doi.org/10.1007/BF00289234

Gierer-Meinhardt Model 2

- **Source Density:**
  - Meinhardt used source density for hydra model in 1993 [1]
  - It describes ability of cells to perform the autocatalytic reaction
  - Higher source density increases activation concentration
  - Source density gradient can determine polarity of pattern

$$\partial_t u = D_1 \Delta u + \frac{au^2}{v} S - \mu u$$

$$\partial_t v = D_2 \Delta v + bu^2 S - \nu v$$

$$\partial_t S = d \Delta S + u - \delta S$$

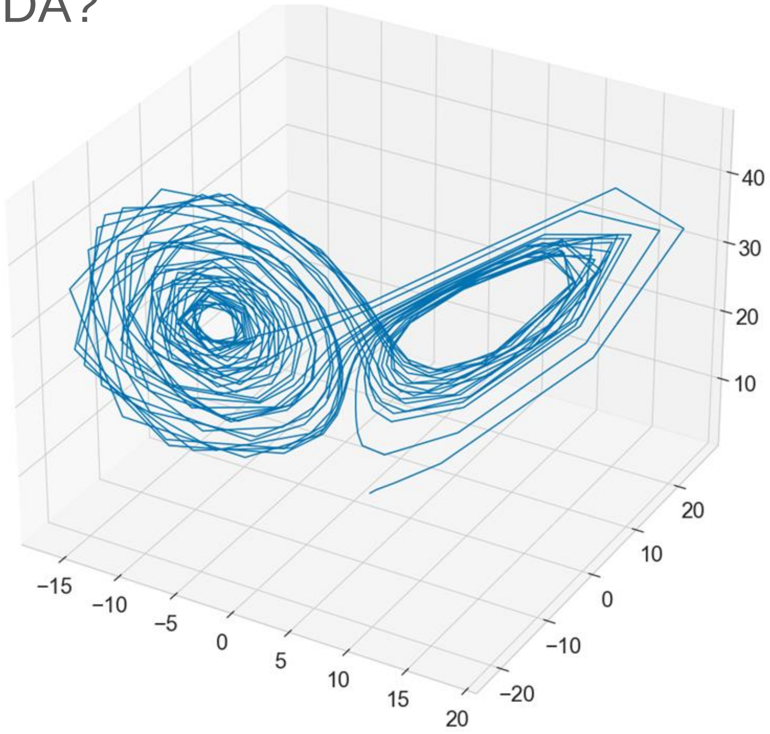
activator **(u)**, inhibitor **(v)**, source density **(S)** **eq2**

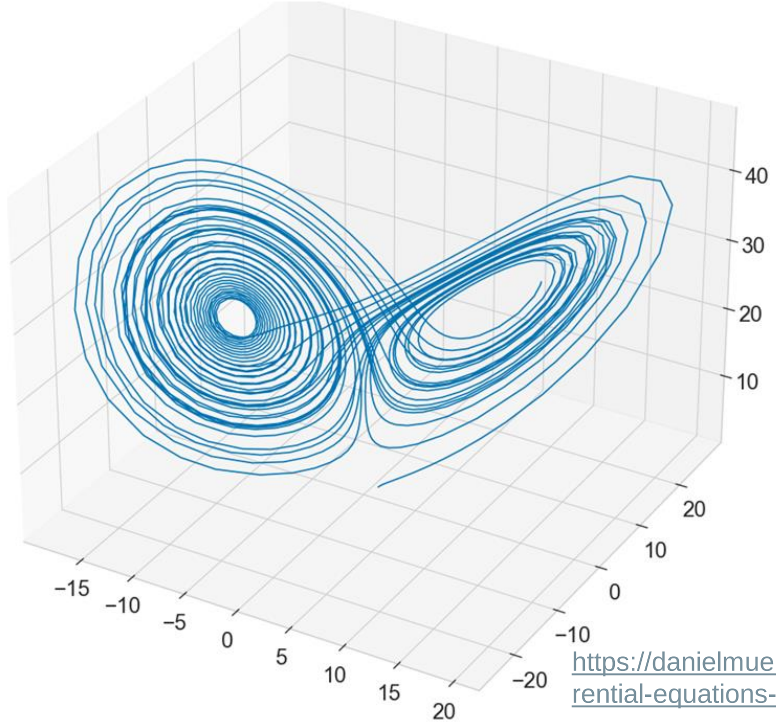
[1] Meinhardt, H. (1993) “A model for pattern formation of hypostome, tentacles, and foot in hydra: how to form structures close to each other, how to form them at a distance,” *Developmental biology*, 157(2), pp. 321–333.

ODE Solver 1: LSODA

- **LSODA**: Solver for Ordinary Differential Equations
    - FORTRAN ODE solver by Hindmarsh **[1]** imported to python by scipy **[3]**
    - Switches between stiff BDF and non-stiff Adams methods **[2]**
    - Why LSODA?

Runge-Kutta 45





LSODA

https://danielmuellerkomorowska.com/2021/02/16/differential-equations-with-scipy-odeint-or-solve_ivp

[1] A. C. Hindmarsh, “ODEPACK, A Systematized Collection of ODE Solvers,” IMACS Transactions on Scientific Computation, Vol 1., pp. 55-64, 1983.
[2] L. Petzold, “Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations”, SIAM Journal on Scientific and Statistical Computing
[3] https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.LSODA.html

ODE Solver 2: Adams

- **Adams:**
  - Implicit method for the numerical integration of ODEs
  - Linear multistep methods, takes function and time
  - Replacing the integrand with polynomial that interpolates $f(t, y)$
  - Coefficients determined by previously calculated data points

$$y' = f(t, y), \quad y(t_0) = y_0.$$

initial value problem

$$y_{n+r} = y_{n+r-1} + h \sum_{k=1}^{r} \lambda_k f_{n+r-k}$$

Adams general formula

$$\sum_{k=1}^{r} \lambda_k = 1$$

- ***h*** denotes time step, ***k*** denotes step size, ***f*** is the function and ***y*** is the solution at time ***t***

ODE Solver 3: BDF

- **Backward differentiation formula** (BDF):
  - Implicit methods for the numerical integration of ODEs
  - Linear multistep method, takes function and time
  - Approximate the derivative of function using previously computed information
  - Increasing the accuracy of the approximation

$$y' = f(t, y), \quad y(t_0) = y_0.$$ initial value problem

$$\sum_{k=0}^{s} a_k y_{n+k} = h\beta f(t_{n+s}, y_{n+s}),$$ BDF general formula

- ***h*** denotes time step, ***k*** denotes step size, **a** and ***β*** are chosen so that method achieves maximum order ***s***, ***f*** is the function and ***y*** is the solution at time ***t***

Code 1: Libraries

- The code was written in python 3.

- Solving differential equations:

```
import numpy as np
from scipy.integrate import solve_ivp
```

  - Solve IVP function solves initial value problem for system of ODEs.

- Visualizations for simulations:

```
import matplotlib.pyplot as plt
from celluloid import Camera
```

  - Animating values at each instance was possible through them.

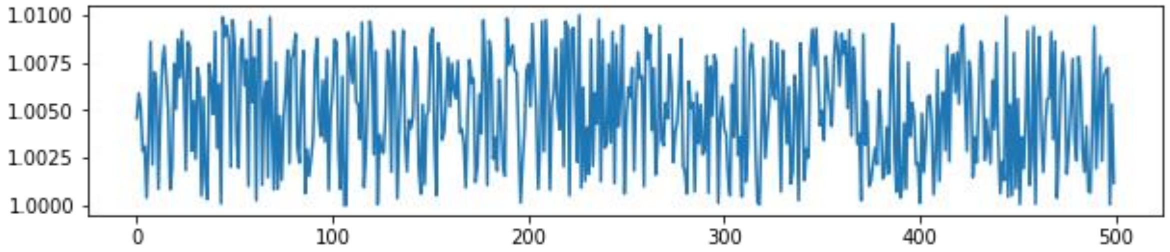
Code 2: Equations

$$\partial_t u = \boxed{D_1 \Delta u} + \frac{au^2}{v} \boxed{S - \mu u}$$
$$\partial_t v = \boxed{D_2 \Delta v} + bu^2 \boxed{S - \nu v}$$
$$\partial_t S = \boxed{d \Delta S} + u - \boxed{\delta S}$$

activator **(u)**, inhibitor **(v)**, source density **(S)** **eq2**

```
#activator eq2
def activator(y):
  activator = D1* delta2d(y[:dim],dx) + a*u(y)**2/v(y) * S(y) - mu*u(y)
  return(activator)

#inhibitor eq2
def inhibitor(y):
  inhibitor = D2* delta2d(y[dim:2*dim],dx) + b*u(y)**2 * S(y) - nu*v(y)
  return(inhibitor)

#source density
def source(y):
  source = d* delta2d(y[2*dim:3*dim],dx) + u(y) - delta * S(y)
  return(source)
```

Code 3: Solve ODEs



- **IC** is vector representation of initial state
  - Uniform Gaussian noise
    - Mean = 1
    - Variance = 0.01
- **tc** is timespan of solver i.e. it integrates from time 0 to 100
- **solve_ivp** takes the function, timespan and initial state and solves ODEs

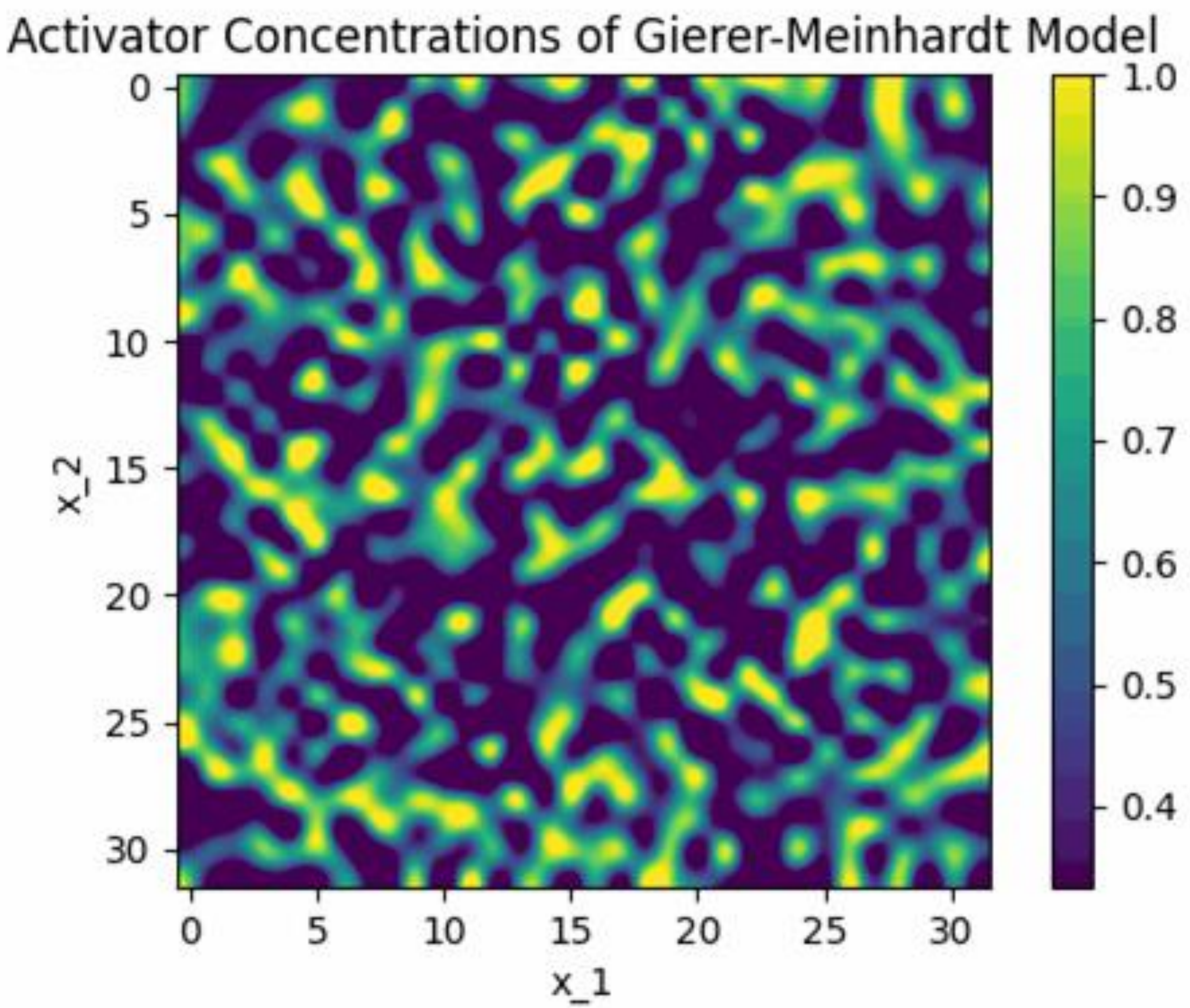
```
IC = 1+0.01*np.random.rand(3*dim)
tc = [0,100]

gms_rhs = lambda t,y: [activator(y),inhibitor(y),source(y)]
sol = solve_ivp(gms_rhs,tc,IC,method='LSODA',vectorized=True)

t = sol.t
y = sol.y.T
```

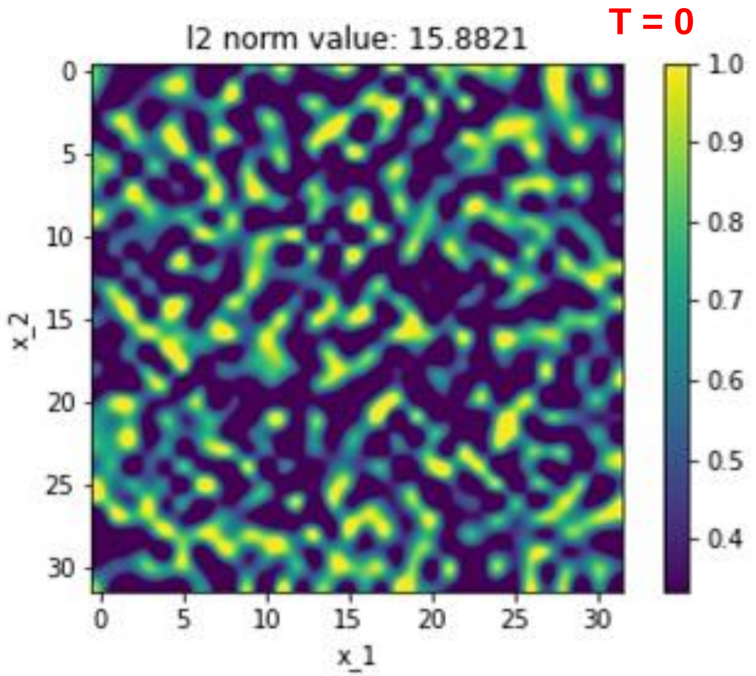
Code 4: Visualization

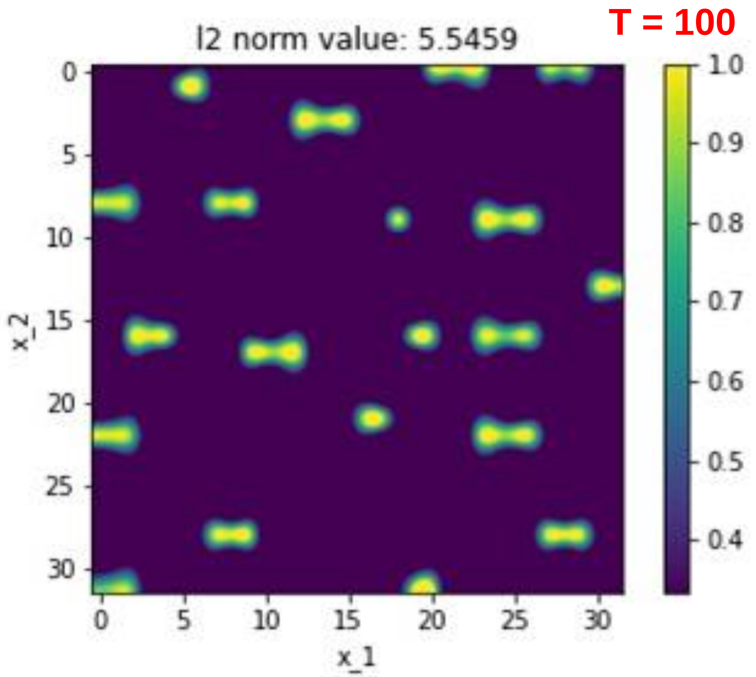
- **Loop** through the solutions for each instance
- **plt.imshow** creates 2D surface plot of solution
- **Camera** captures current slice of result using snap function
- **animate** function arranges all snaps of instances into a gif file

```
fig = plt.figure(dpi=100)
camera = Camera(fig)
for i in range(len(y)):
  image = y[i,:dim].reshape((N,N))
  image = minmax_scale(image)
  plt.imshow(image,interpolation='sinc',vmin=np.max(image)/3)
  plt.xlabel('x_1')
  plt.ylabel('x_2')
  plt.title('Activator Concentrations of Gierer-Meinhardt Model')
  camera.snap()
plt.colorbar()
animation = camera.animate()
animation.save('simulation3.gif', writer='pillow', fps=60)
```

Simulation 1: eq2

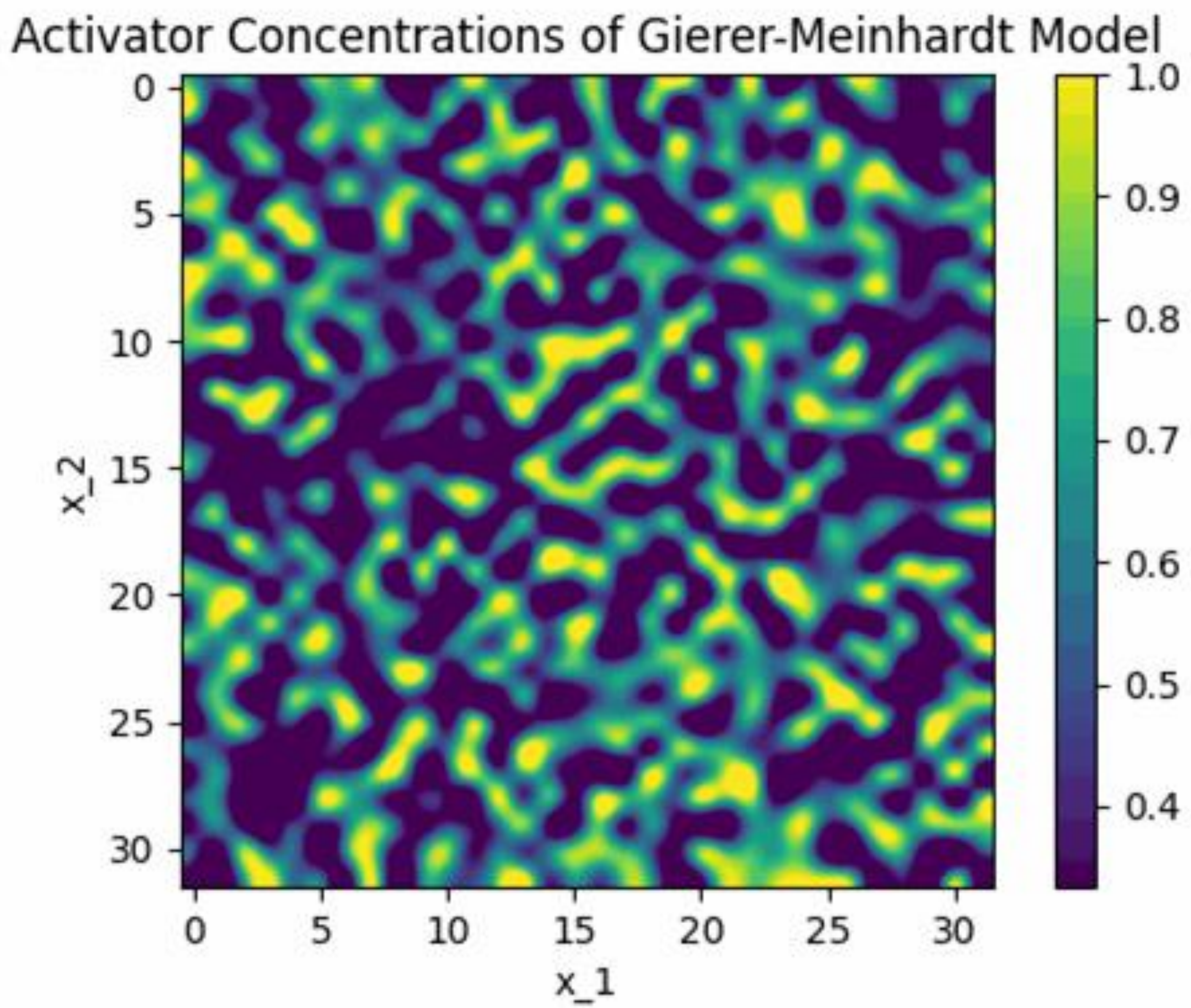




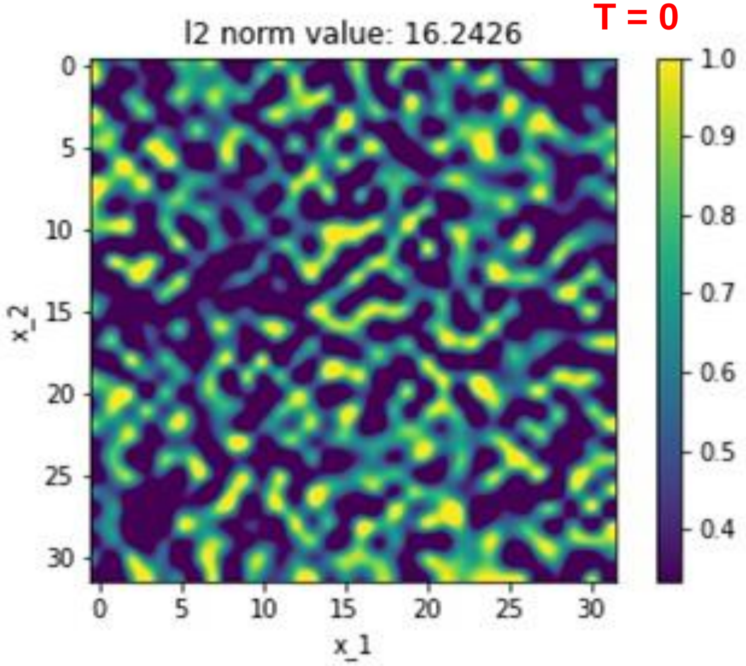


'D1', 0.0002, 'D2', 0.01, 'a', 1, 'b', 1, 'mu', 0.5, 'nu', 1, 'd', 0, 'delta', 1

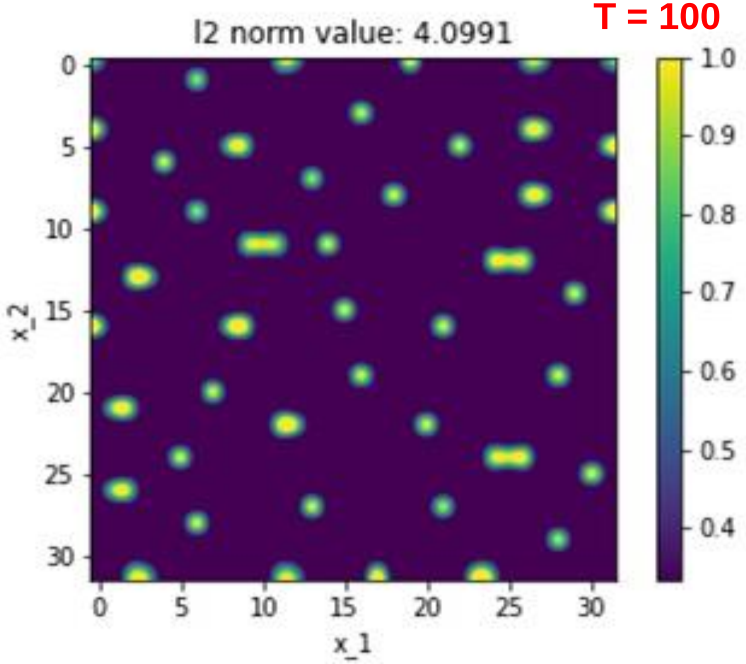
Simulation 2: eq2



T = 0

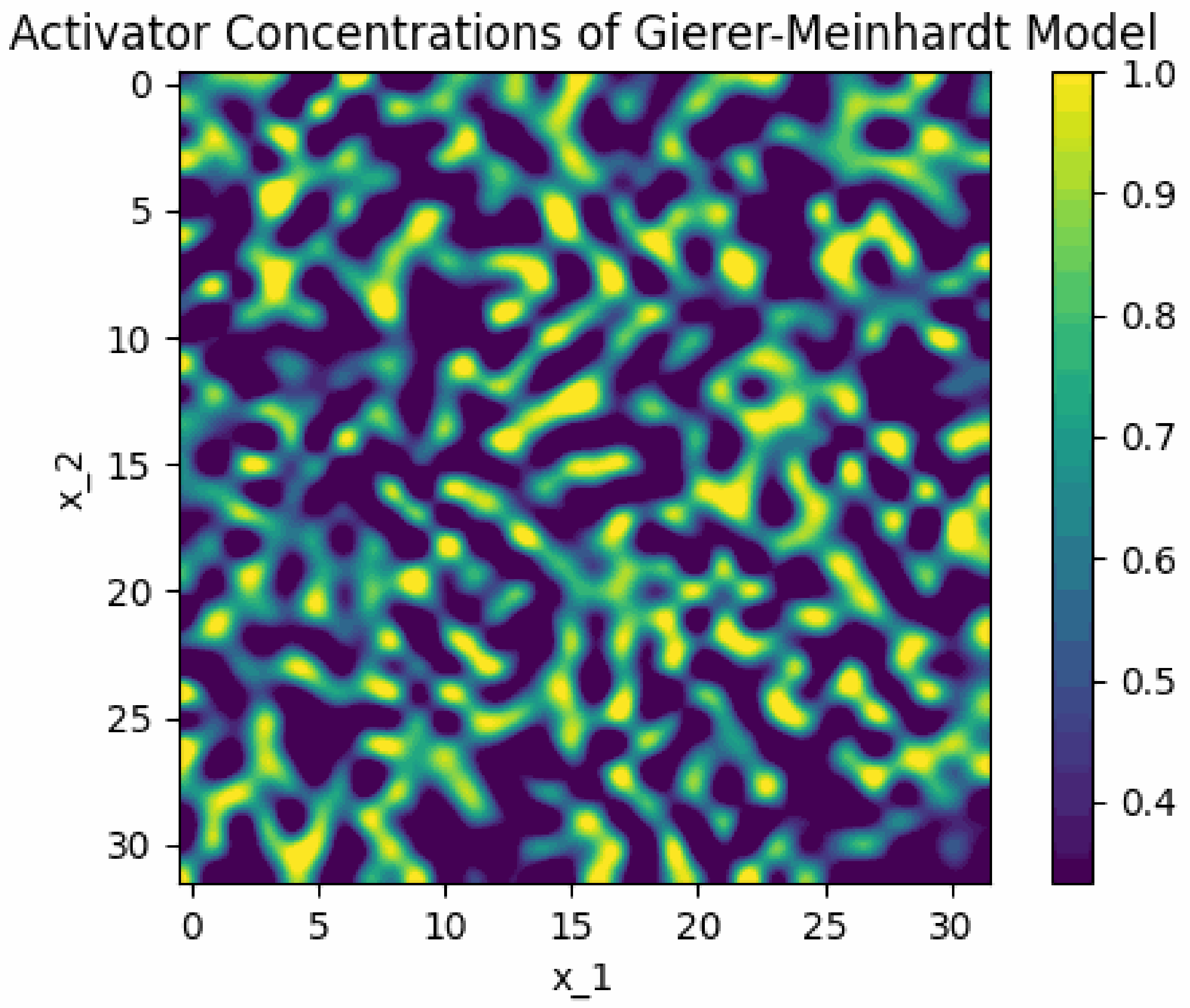


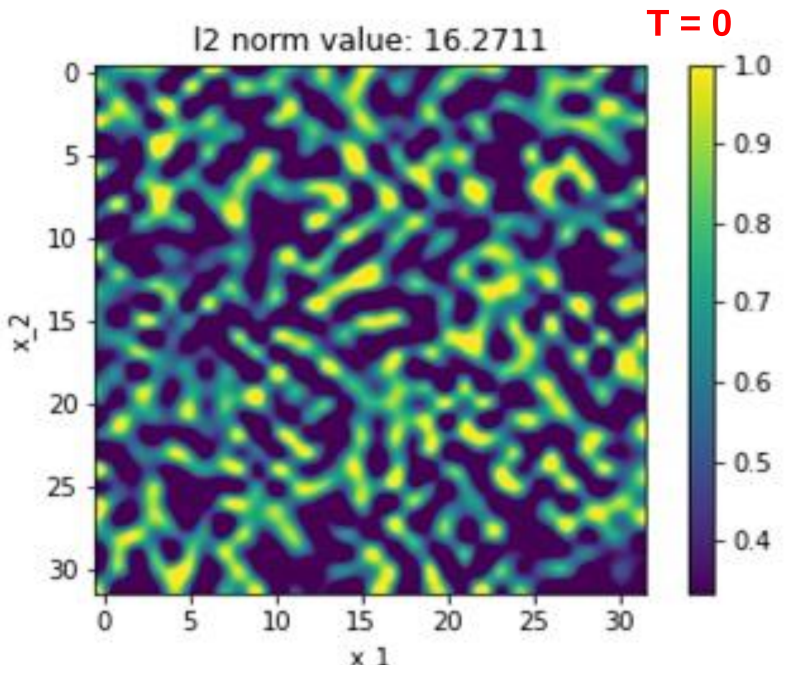
T = 100

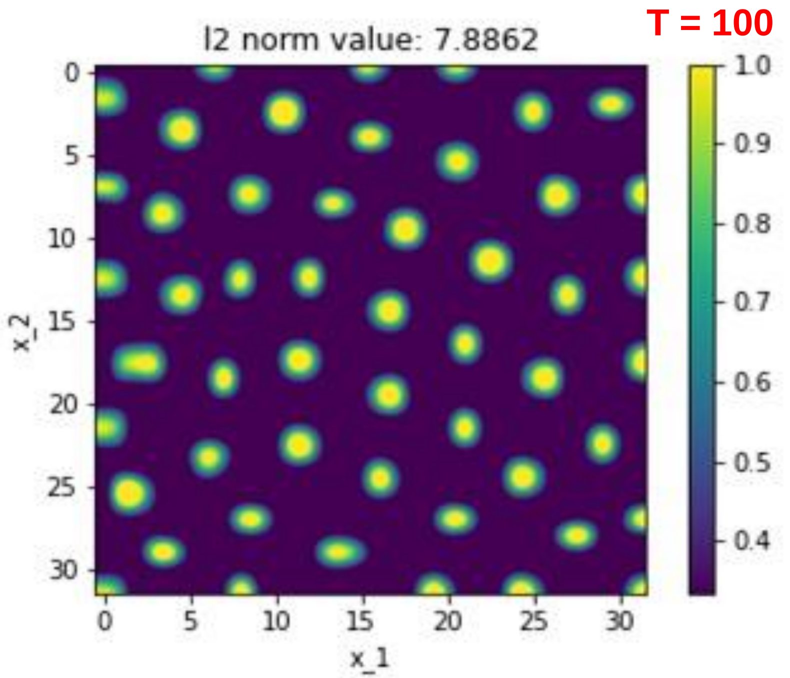


'D1', 0.0002, 'D2', 0.01, 'a', 2, 'b', 1, 'mu', 1, 'nu', 2, 'd', 2, 'delta', 1.5

Simulation 3: eq2



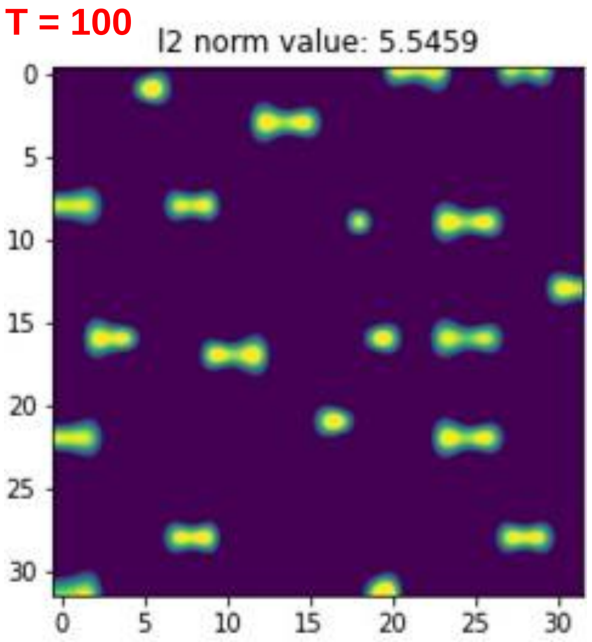




'D1', 0.0002, 'D2', 0.01, 'a', 3, 'b', 2, 'mu', 0.75, 'nu', 4, 'd', 3, 'delta', 3

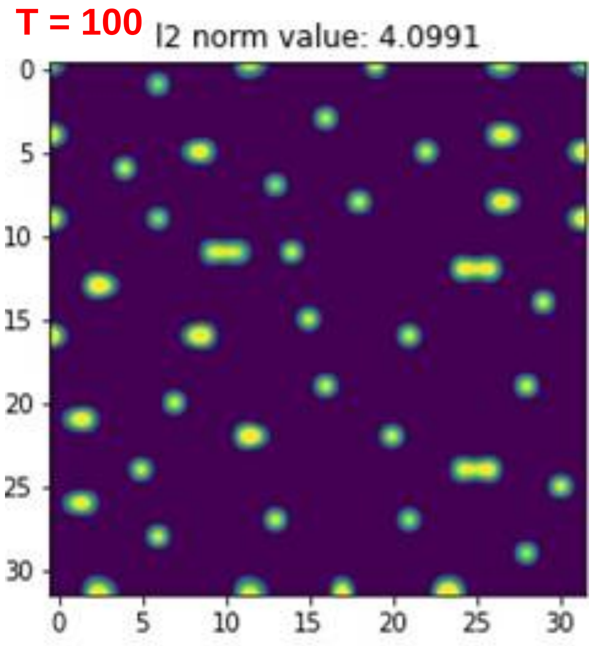
Results 1: eq2

a	1
b	1
mu	0.5
nu	1
d	0
delta	1



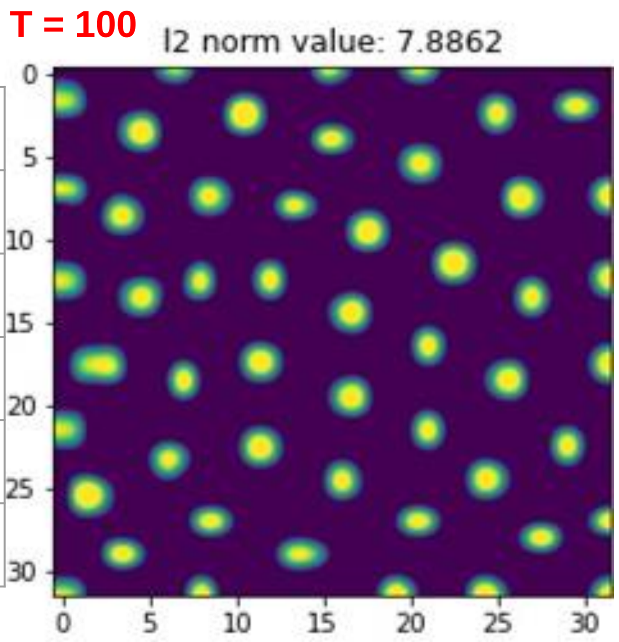
Simulation 1

a	2
b	1
mu	1
nu	2
d	2
delta	1.5



Simulation 2

a	3
b	2
mu	0.75
nu	4
d	3
delta	3



Simulation 3

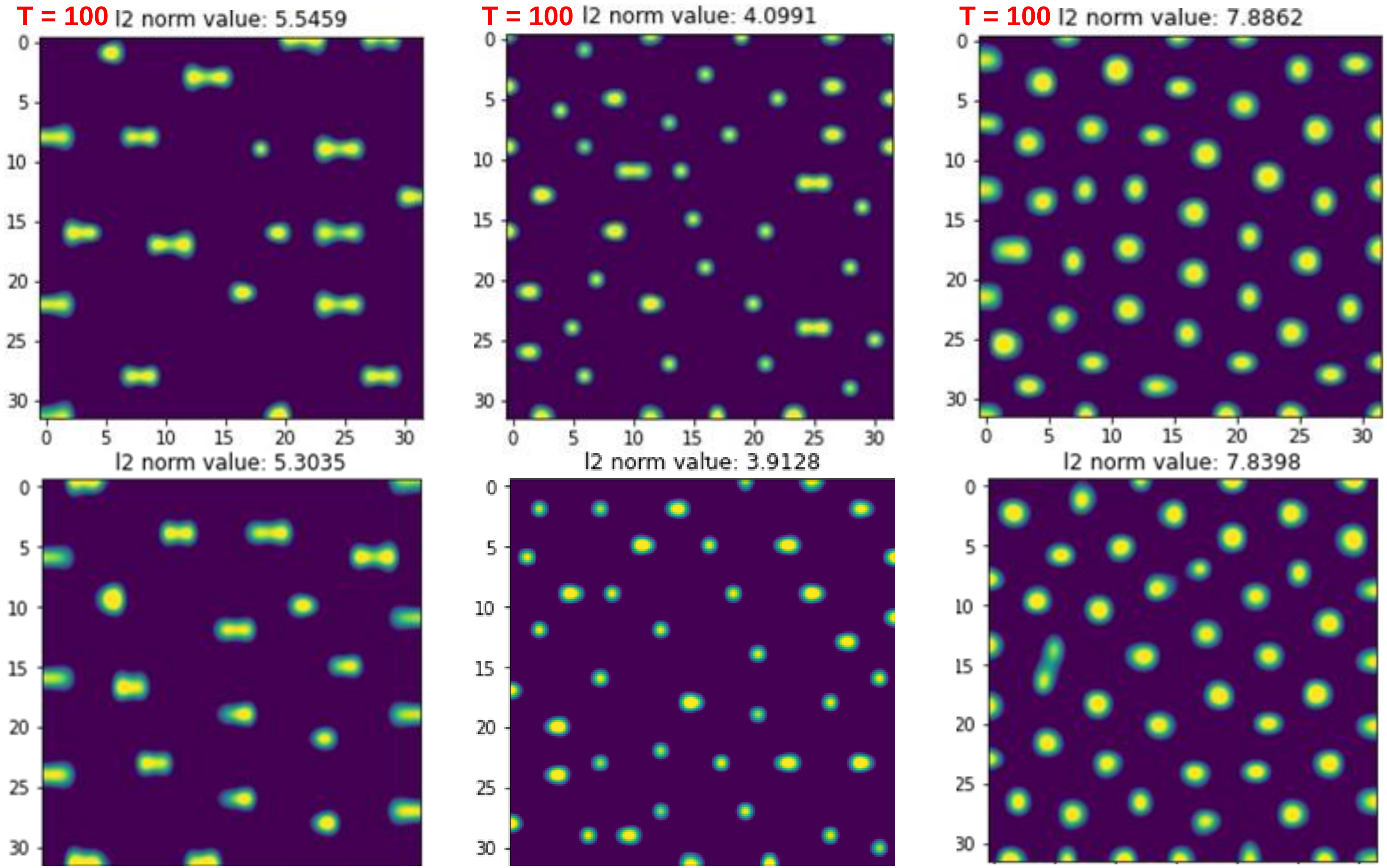
- Comparison of the systems at their final pattern
- **D1** and **D2** values were **0.0002** and **0.01** for all
- **L2 norm** tending to 0 indicates reaching the steady state
    - values did not go lower when increasing **tf** from **100** to **300**

Results 2: Comparison

With source density
(eq2)

Same parameters

No source density
(eq1)



Results 3: Slight changes

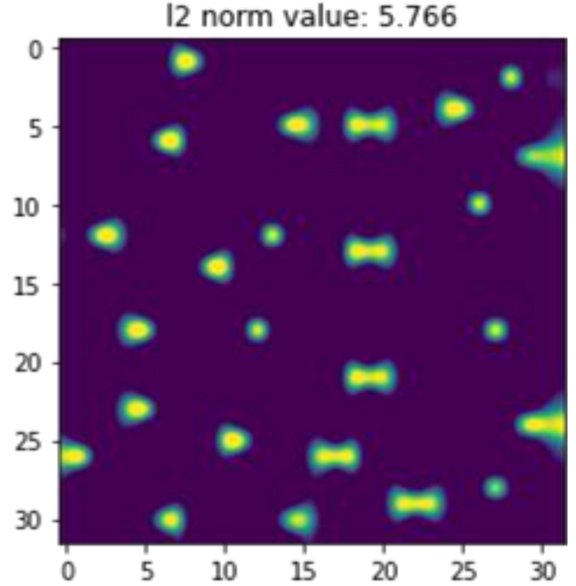
- Experimenting with slightly different equation:
  - Source density impact is higher
  - However the difference is not major
  - Compared to **eq1** and **eq2**

$$\partial_t u = D_1 \Delta u + \left\{ \frac{a u^2}{v} - \mu u \right\} S$$
$$\partial_t v = D_2 \Delta v + \left\{ b u^2 - \nu v \right\} S$$
$$\partial_t S = d \Delta S + u - \delta S$$

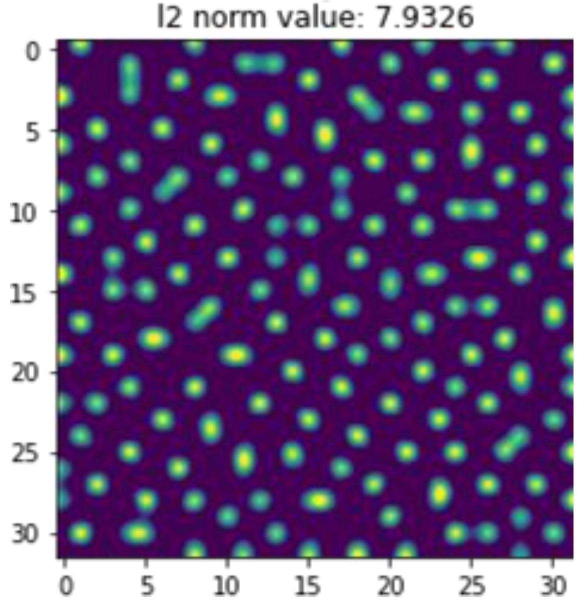
activator **(u)**, inhibitor **(v)**, source density **(S)** **eq3**

T = 100

a	1
b	1
mu	0.5
nu	1
d	0
delta	1



a	3
b	2
mu	0.75
nu	4
d	3
delta	3



Results 4: Saturation

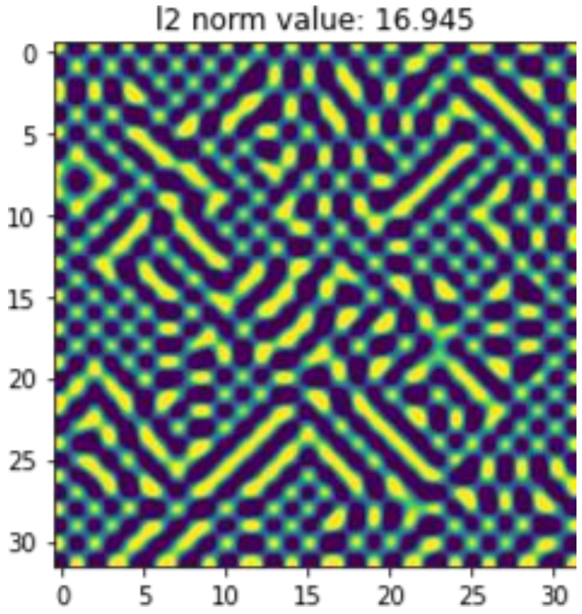
- For the case of saturation of the activator **[1]**:
  - Replacing $\mathbf{u^2}$ with $\mathbf{u^2/(1+ku^2)}$
  - K = 0.002
  - Gives rise to striped and checkered patterns
  - Steady state not reached within T=200

$$\partial_t u = D_1 \Delta u + \left\{\frac{au^2}{v(1+ku^2)} - \mu u\right\} S$$
$$\partial_t v = D_2 \Delta v + \left\{bu^2 - \nu v\right\} S$$
$$\partial_t S = d \Delta S + u - \delta S$$

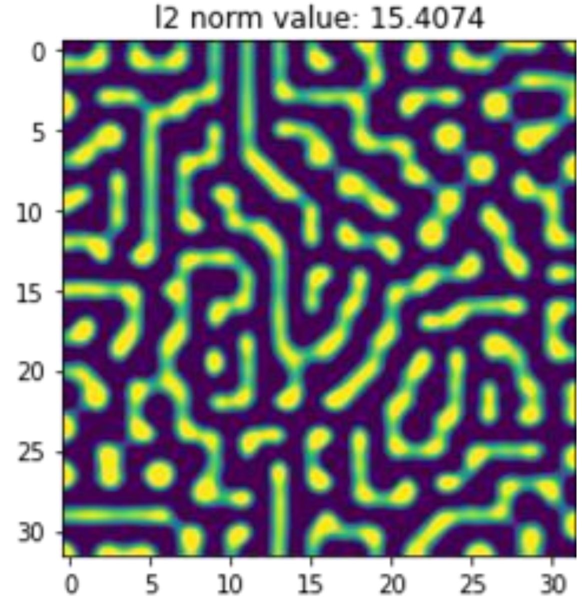
activator **(u)**, inhibitor **(v)**, source density **(S)** **eq4**

T = 200

a	2
b	1
mu	1
nu	4
d	2
delta	1.5

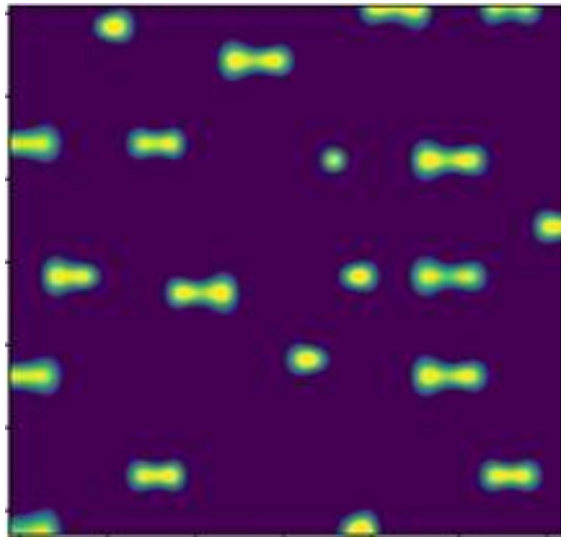


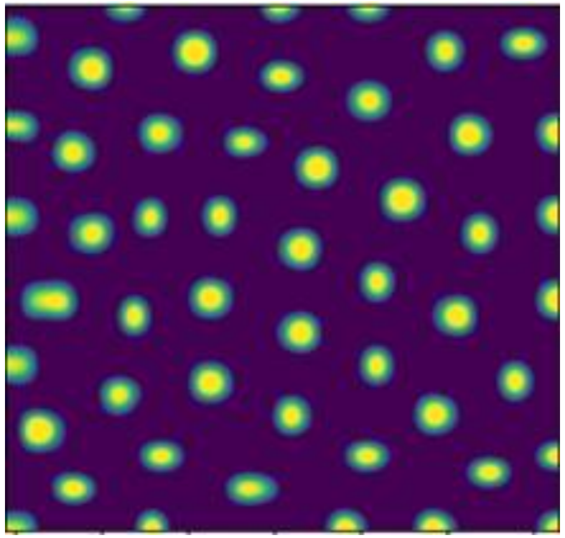
a	2
b	1
mu	1
nu	4
d	3
delta	3

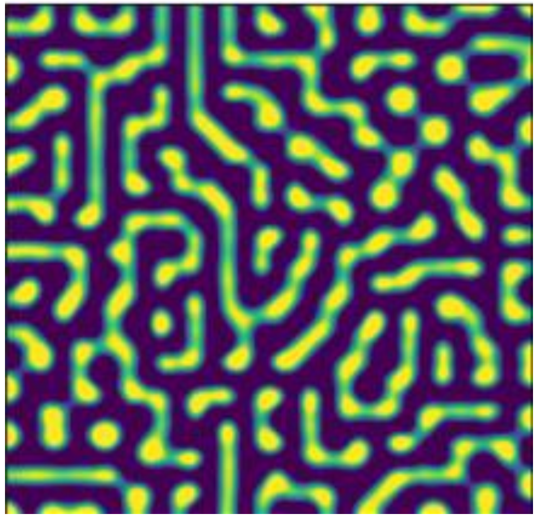


[1] Li, Y., Wang, J., & Hou, X. (2017). Stripe and spot patterns for the Gierer–Meinhardt model with saturated activator production. In Journal of Mathematical Analysis and Applications (Vol. 449, Issue 2, pp. 1863–1879)

Patterns in Nature







https://www.pinterest.com/pin/red-eyes--10062799137806377/

https://wildlifesafari.info/cheetah.html

https://nl.pinterest.com/pin/418553359100200002/

Discussion

- Time overhead (google colab):
  - Simulations took between 30 to 60 seconds when T=100
  - Animation took approximately 30 seconds

- Not much noticeable difference between **eq1** and **eq2**
- Some difference was noticed in **eq3**
- Major difference in **eq4** when activator saturated

- Source density in simulations:
  - Seems not very effective for **eq1, eq2**
  - However very effective in case of **eq3**, **eq4**
  - Systems were most affected by $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$

Thank you for your time!

Questions?

Github: link to code

Appendix: delta2d function and params

```
N=32
dim = N**2
dx = 1/(N-1)

D1 = 0.0002
D2 = 0.01

a = 3
b = 2
mu = 0.75
nu = 4

d = 3
delta = 3
```

params

```
def delta2d(y,dx):
  y = y.reshape(-1,1)
  N = int(np.sqrt(len(y)))
  U = np.reshape(y,(N,N))

  Ur = np.hstack([U[:,1:],U[:,-1].reshape(-1,1)])
  Ul = np.hstack([U[:,0].reshape(-1,1), U[:,:-1]])
  Ut = np.vstack([U[0,:].reshape(-1,1).T, U[:-1,:]])
  Ub = np.vstack([U[1:,:], U[-1,:].reshape(-1,1).T])

  dU = (Ur+Ul+Ut+Ub - np.multiply(4,U))/dx**2
  dy = dU.ravel().reshape(-1,1)
  return(dy)
```

delta2D computes the finite difference approximation of the Laplace operator with Neumann boundary conditions in the 2D squared domain.