SIMULATIONS OF GIERER-MEINHARDT MODEL WITH SOURCE DENSITY

Shourya Verma
Universität Heidelberg
shourya.verma@stud.uni-heidelberg.de

ABSTRACT

This report discusses how the addition of source density affects the original GM model and compare how the activator concentration patterns change when the equations are modified to include saturation of the activator. This is done by running experiments to perform various simulations and visualizing the activator concentration patterns of the systems in two dimensional space.

***K*eywords** Reaction-Diffusion, Activator-Inhibitor, Gierer-Meinhardt, Turing Patterns, Simulations

1 Introduction

One of the most exciting elements of biology is the evolution of a higher creature from a single fertilized egg. How cells differ from one another despite having the same genetic code is a key topic. Inorganic systems frequently exhibit spontaneous pattern development in originally almost homogenous systems. Even though the wind continually redistributes the sand, large sand dunes nevertheless develop. Even though the rain falls more or less uniformly over the land, sharply curved and branching river systems (which are really rather similar to the branching patterns of a nerve) are created by erosion. Introduced by Alan Turing in 1952 [1], 'Morphogenesis' is defined as the biological process which allows an organism to develop its form by a mechanism of distributing cell and tissue spaces. It is one of three fundamental aspects of developmental biology along with the control of tissue growth and patterning of cellular differentiation which allows us to better understand the process of embryonic development. The Gierer-Meinhardt (GM) model introduced in 1972 by Alfred Gierer and Hans Meinhardt [2]. These are a set of activator-inhibitor system rising from the reaction-diffusion equations which serve a role in Turing type pattern formation and morphogenesis observed during development of organisms. The GM model explains many pattern formation process in nature. Through these set of equations we can understand the mechanisms of polarity retention in hydra and the process of congenital cardiac malformations.

2 Background

2.1 Activator-Inhibitor Systems

Activator-inhibitor systems include two substances that act on each other. The localized activator stimulates its own production through autocatalysis and also produces the inhibitor. In-turn, the inhibitor represses the production of the activator. The equation 1 below shows the original GM equations where u is the activator and v is the inhibitor, D_1 and D_2 are diffusion constants, μ and ν are decay rates, a and b are production rates, and Δ is the Laplace operator. As for equation 2, source density S is added which serves as a way to regulate the increase or decrease in the production of activators and inhibitors during cell formation. The source density distribution also determines the polarity of the pattern formation sequence and is expected to survive sectioning and transplantation of tissues. This source density was used by Meinhardt for the hydra model in his 1993 paper [3].

$$\begin{aligned} \partial_t u &= D_1 \Delta u + \frac{a u^2}{v} - \mu u \\ \partial_t v &= D_2 \Delta v + b u^2 - \nu v \end{aligned} \tag{1}$$

$$\begin{aligned} \partial_t u &= D_1 \Delta u + \frac{a u^2}{v} S - \mu u \\ \partial_t v &= D_2 \Delta v + b u^2 S - \nu v \\ \partial_t S &= d \Delta S + u - \delta S \end{aligned} \tag{2}$$

3 Methodology

To perform the simulations on these GM equations, the ordinary differential equation (ODE) solver from scipy library [3] was used. This library solves the initial value problem 3 using the LSODA method which was first introduced by Hindmarsh on FORTRAN [4]. The LSODA solver switches between the stiff BDF 4 and non-stiff Adams method 5. These two methods are implicit methods for solving numerical integration of ODEs which use linear multistep process to solve function for given time interval. The LSODA method performs better than the well known Runge-Kutta 45 method since it better optimizes the time-step function allowing for calculations of more solutions. For the below equations h denotes time step, k denotes step size for order r, a and β are chosen so that method achieves maximum order s, f is the function and y is the solution at time t.

$$y' = f(t, y), \quad y\left(t_0\right) = y_0 \tag{3}$$

$$\sum_{k=0}^{s} a_k y_{n+k} = h \beta f\left(t_{n+s}, y_{n+s}\right) \tag{4}$$

$$y_{n+r} = y_{n+r-1} + h \sum_{k=1}^{r} \lambda_k f_{n+r-k} \tag{5}$$

The code for the simulation experiments was written in python 3 using numpy, scipy, matplotlib and celluloid libraries. These were used to solve the differential equations using LSODA method and for the visualizations of the simulations. To create the visualizations of the simulation solutions, a code was written to create an animation of the 2D solutions of the ODEs. Some of the code snippets are available in the appendix section. The complete code can be found on github.

3.1 Experiment 1

The first experiment focused on equation 2 and created three simulations with different parameter values. This gave us final patterns of the activator concentrations of the system. This allowed us to analyse and observe the differences that appeared due to changing source density while keeping D_1 and D_2 values the same at 0.0002 and 0.01 respectively. The initial state of the system was created by generating random uniform Gaussian noise of mean 1 and variance 0.01, this is visualized in figure 1. Figure 2 shows the final patterns of the systems and table 1 shows the parameters values for each simulation. Here we observe that simulation 1.1 creates horizontal 8 patterns along with slighlty triangular individual patterns, while simulations 1.2 and 1.3 create dots of different sizes. However, the dots in simulation 1.3 were more uniformly arranged compared to simulation 1.2. The L2 norm values shown above the patterns tending to 0 indicate that the simulation is reaching its steady state.

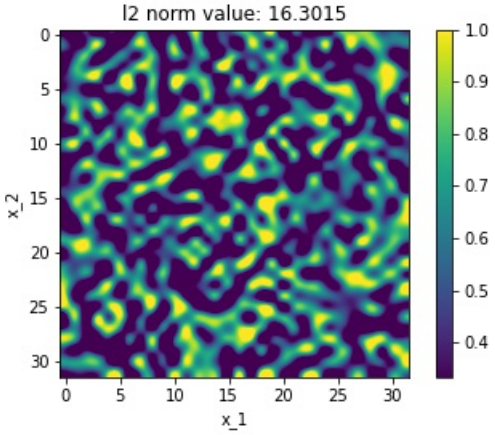

Figure 1: Visualization of Initial State of the Simulation (Uniform Gaussian Noise)

Parameters	sim 1	sim 2	sim 3
a	1	2	3
b	1	1	2
μ	0.5	1	0.75
ν	1	2	4
d	0	2	3
δ	1	1.5	3

Table 1: Simulation Parameters for Experiments 1, 2 and 3

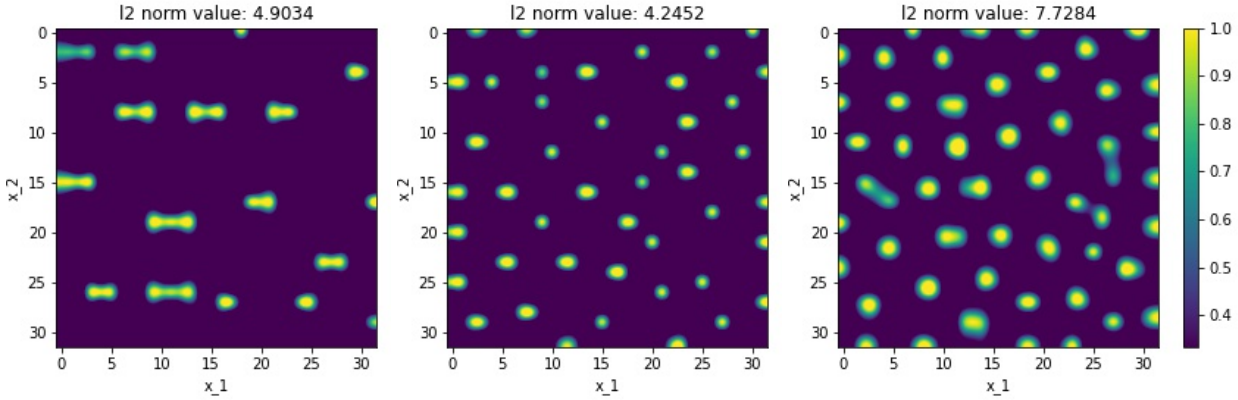

Figure 2: Simulations for Experiment 1 (from left to right: sim 1.1, sim 1.2, sim 1.3)

3.2 Experiment 2

In the second experiment we focus on equation 1 with no source density and use the same parameters for the simulations used above in table 1. We notice that in figure 3 there is no noticeable change observed in the results except for the fact that there are a few more patterns appearing that previously observed in experiment 1.

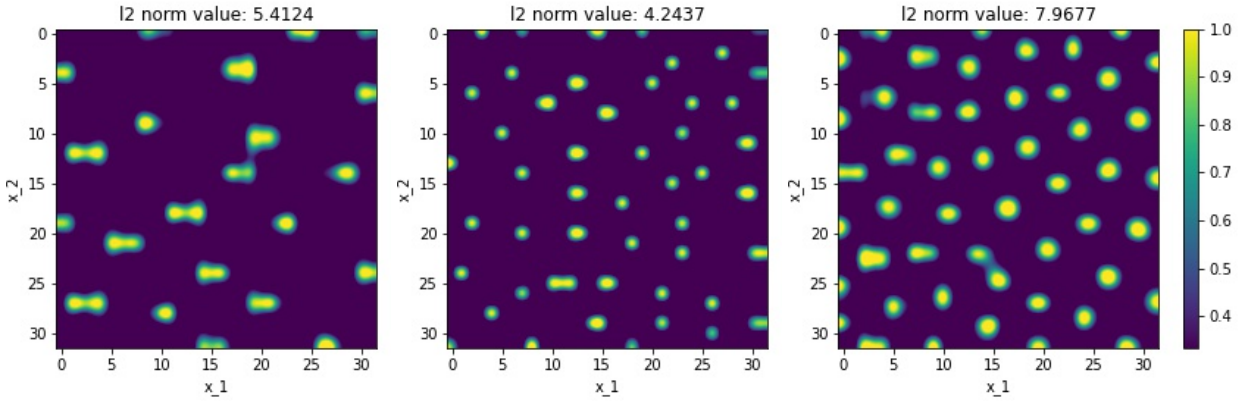

Figure 3: Simulations for Experiment 2 (from left to right: sim 2.1, sim 2.2, sim 2.3)

3.3 Experiment 3

Since no concrete difference was found between patterns emerging from equations 2 and 1, a slightly modified equation 6 was used for experiment 3. Here the source density directly affects the μ and ν variables for both activator and inhibitor. The simulations were ran for the same parameters as above in table 1. Here we notice that in simulation 3.1 the horizontal 8 patterns are more sharp along with the unique funnel type patterns appearing on the edges. Simulation 3.2 and 3.3 look quite similar, however the latter is more dense and has more dots appearing.

$$\begin{aligned} \partial_t u &= D_1 \Delta u + (\frac{au^2}{v} - \mu u) * S \\ \partial_t v &= D_2 \Delta v + (bu^2 - \nu v) * S \\ \partial_t S &= d\Delta S + u - \delta S \end{aligned} \tag{6}$$

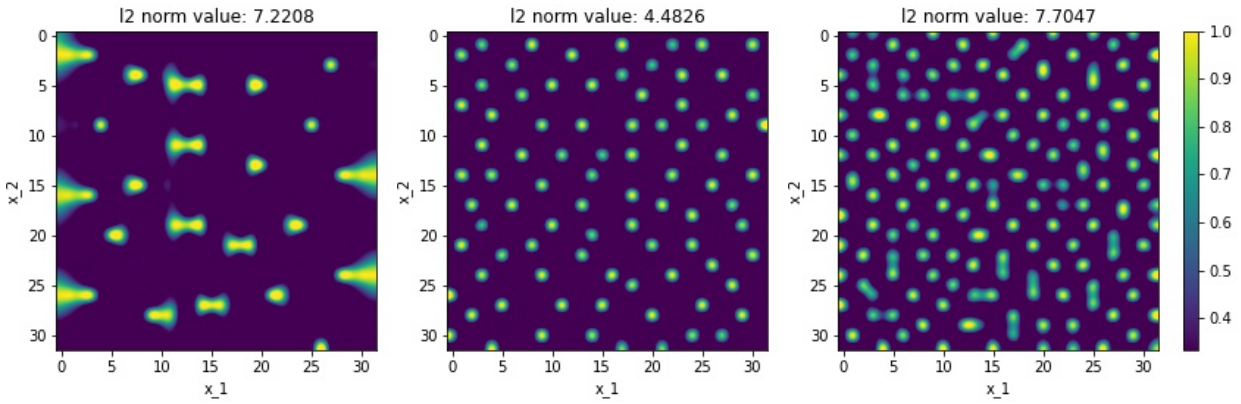

Figure 4: Simulations for Experiment 3 (from left to right: sim 3.1, sim 3.2, sim 3.3)

3.4 Experiment 4

The GM model originally introduced explicit limitation of the activator to achieve constant activation for a given range. This was done by assuming a saturation of the activator production, multiplying $1/(1 + ku^2)$ to $\frac{au^2}{v}$. This will allow the activator to saturate when a^2 is approximately eqaul to k. This allows the activator concentration system to produce striped and checkered patterns. The parameter values for experiment 4 were set as: $D_1 : 0.0002$, $D_2 : 0.01$, $a : 2$, $b : 1$, $\mu : 1$, $\nu : 4$, $k : 0.002$. Only the d and δ values were changed between the three simulations, these values were: $d : 2$, $\delta : 1.5$ for simulation 4.1, $d : 3$, $\delta : 3$ for simulation 4.2, and $d : 4$, $\delta : 5$ for simulation 4.3. Figures 5 shows the activator concentration patterns formed by this system of equations and parameters. We can clearly notice a checkered pattern for simulation 4.1 and stripe patterns for simulation 4.2 and 4.3. The L2 norm values are still quite high which represents that the system has not yet reached its steady state.

$$\begin{aligned} \partial_t u &= D_1 \Delta u + (\frac{au^2}{v(1 + ku^2)} - \mu u) * S \\ \partial_t v &= D_2 \Delta v + (bu^2 - \nu v) * S \\ \partial_t S &= d\Delta S + u - \delta S \end{aligned} \tag{7}$$

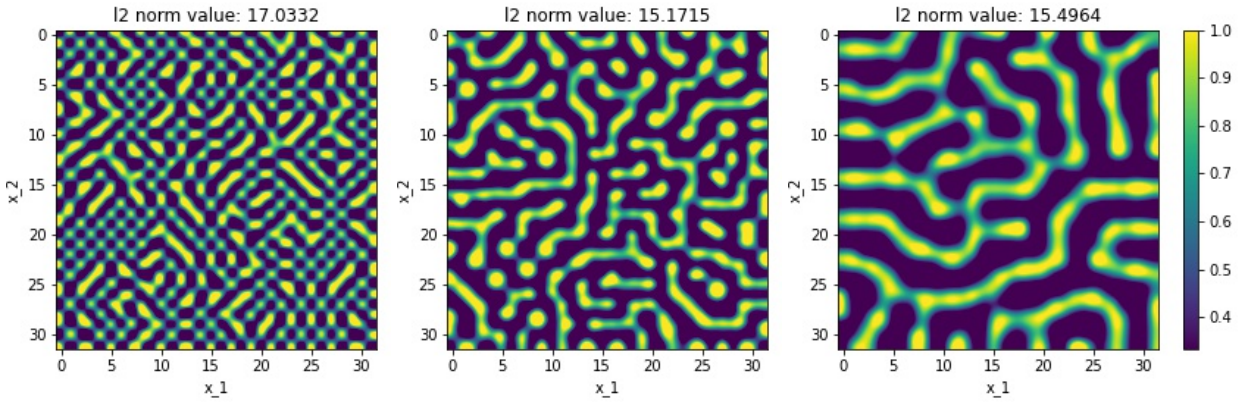

Figure 5: Simulations for Experiment 4 (from left to right: sim 4.1, sim 4.2, sim 4.3)

4 Conclusion

For all the experiments above, the time overhead was around 30-60 seconds per simulation when time t was set to 100. To conclude from the findings, we found that there was not much noticeable difference in activator concentration pattern between equations 1 and 2. Some difference was noticed in equation 6 as we saw more dense patterns in simulations 3.2 and 3.3, and the funnel type patterns on the edges of simulation 3.1. Major difference was noticed in the patters for equation 7 when activator was saturated using $k : 0.002$, giving us checkered and stripe patterns. These patterns however had not yet reached their stable state which would mean they could further change their patterns. What we can conclude about the impact of source density is that it did not seem very effective for equations 1 and 2, however it was effective for equations 6 and 7. The systems were most affected by change of μ and ν values.

References

[1] Alan Mathison Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, 1952.

[2] Alfred Gierer and Hans Meinhardt. A theory of biological pattern formation. kybernetik 12, 30- 39. *Biological Cybernetics*, 12:30–39, 01 1972.

[3] Hans Meinhardt. 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):321–333, 1993.

[4] Alan C Hindmarsh. Odepack: A systemized collection of ode solvers. *Scientific computing*, pages 55–64, 1983.

5 Appendix

The source density, activator and inhibitor equations 2 were written as follows:

```
#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)
```

The code for solving the equations using LSODA included the initial state which is uniform gaussian noise with mean 1 and variance 0.01, the time steps and the solver which gave solutions for each value of t. This was written as follows:

```
#initial state
IC = np.array(1+0.01*np.random.rand(3*dim))
#time steps
tc = [0,200]

#ode solver using scipy.integrate.solve.ivp for gierer-meinhardt model with source density
gms_rhs = lambda t,y: [activator(y),inhibitor(y),source(y)]
sol = solve_ivp(gms_rhs,tc,IC,method='LSODA',vectorized=True)

#solutions
t = sol.t
y = sol.y.T
```

A delta function was created to act as the Laplace operator which computed the finite difference approximation of the Laplace operator with Neumann boundary conditions in the 2D squared domain. This function was written as follows:

```
#   INPUT
#   y : input function values, stored in 1D vector
#   dx : spatial step of the grid (scalar value)
#   OUTPUT
#   dy : output function values, stored in 1D vector

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)
```