A Mathematical Approach Towards Pattern Formation in Passion Flower

Upakul Sarma

A dissertation submitted in the partial fulfilment of
the requirements for the degree of BS-MS in Chemistry

Indian Institute of Science Education and Research Mohali
April 2015

Certificate of Examination

This is to certify that the dissertation titled "**A Mathematical Approach Towards Pattern Formation in Passion Flower**" submitted by **Mr. Upakul Sarma** (Reg. No. MS10008) for the partial fulfilment of BS-MS dual degree programme of the Institute, has been examined by the thesis committee duly appointed by the Institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Professor N. Sathyamurthy (Supervisor)	Dr. P. Balanarayan	Dr. Abhishek Chaudhury

Dated: April 24, 2015

Declaration

The work presented in this dissertation has been carried out by me under the guidance of Professor N. Sathyamurthy at the Indian Institute of Science Education and Research Mohali.
This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Upakul Sarma
Dated: April 24, 2015

In my capacity as the supervisor of the candidate's project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Professor N. Sathyamurthy
(Supervisor)

Acknowledgment

First and foremost I would like to express my deep sense of gratitude to my thesis supervisor Professor N.Sathyamurthy. Really it has been a great experience working under his guidance on such a nice challenging problem. I pretty much appreciate the way he guides a student by providing moral support and freedom. above all he is always very accommodating and Aboveboard person.

I am also very grateful to my group members Ankur Kumar Gupta and Vikash Dhindhwal. Both of them helped me a lot through my master project work. I would also like to thank my friends Aaveg Agarwal, Ankit Kumar Agarwal for sparing time for me and sharing their programming knowledge with me. Also I would like to thank all my friends out there in the campus who stood by me in all my ups and downs. I will always miss your constant company in my whole life.

I also would like to acknowledge DST,Government of India, for their financial support through INSPIRE Fellowship. I am also thankfull to IISER Mohali for providing me with all infrastructure I needed.

Upakul Sarma
MS10008
IISER Mohali

List of Figures

List of Tables

Acronyms

RD	Reaction-Diffusion
PDE	Partial Differential Equation
TDSE	Time-Dependent Schrödinger Equation
DDI	Diffusion-Driven Instability
BZ	Belousov-Zhabotinskii
FD	Finite Difference
FTCS	Forward-Time-Central-Space
BTCS	Backward-Time-Central-Space
IC	Initial Condition

Contents

Abstract

The evolution and development of spatial patterns in both living and non-living objects has been the subject of study for many evolutionary and developmental biologists. Alan Turing's "The Chemical Basis of Morphogenesis" in 1952 was a major breakthrough in which he theorized a system of two different interacting molecules, called morphogens, which could establish chemical gradients through a "reaction-diffusion system." . Here we give a concise description of some of the interesting mathematical aspects of Turing's Reaction-Diffusion (RD) mechanism and give an overview of some popular reaction models incorporated into it. We tried to assimilate the idea of Turing's RD mechanism and utilize it to study the pattern formation in Passiflora Incarnata (Passion Flower) , which has non-uniform alternate bands of violet and white coloured pattern on each of its fibrils. We study the pattern using "Gierer-Meinhardt" model.

Chapter 1

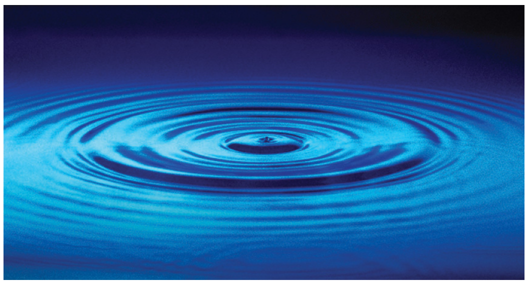
Introduction

Mother nature provides us with several beautiful patterns which can be associated with living as well as non-living objects. These patterns are nothing but a visual recurring of a peculiar form and can be modelled mathematically. These patterns can be thought of as a simple or complex based on their formation. This formation may contain repetition of color, shapes or other variations. In chemistry also there are many patterns which can be categorized in spatial and spatio-temporal pattern. Among the best known pattern the Belousov-Zhabotinskii(BZ) reaction[4] is the finest example of oscillatory reaction. In BZ reaction bromate ions oxidise malonic acid which is catalysed by cerium (Ce^{3+}/Ce^{4+}). Sustained periodic oscillations are observed in cerium ions concentrations. If instead of cerium if one uses the Fe^{2+}/Fe^{3+} and phenanthroline, the pattern is visualized as color changes between reddish-orange and blue.[5] Now question arises that how does the pattern formation occur?

One of the major issues that developmental biology deals with is to understand the morphogenesis i.e. the emergence of structure and shape from an almost uniform mass of dividing cells that constitutes the early embryo. Although genes play a essential role, genetics says nothing about the mechanism responsible for pattern formation. Therefore many questions arise such as: What exactly happens during embryo genesis which leads to an organism's shape? How do living organisms convert the detailed one-dimensional genetic information into a three-dimensional map, the shape of the living organism?

There are many models which explain that how do the different processes come together and generate patterns.They are categorised in two ways such as: the gradient-

(a) Sand dunes


(b) Waves in water

(c) Butterfly wings

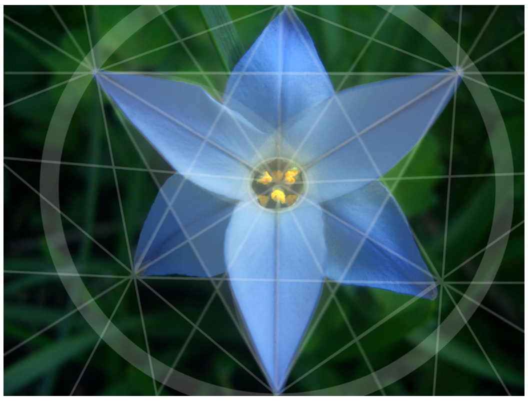

(d) Symmertry in flower

Figure 1.1: Exemplary patterns in nature; Source: Internet

type models which involves a simple source-sink mechanism,[6] the cellular automata models in which the tissue is discretised and rules are introduced as to explain how different elements interact with each other[7] and more complicated models which incorporate more sophisticated chemistry and biology. Here we shall focus on the model from the latter category.

The first person to put forward the kinetic preconception of pattern formation on a firm mathematical basis was "Alan Turing". In 1952 he published a paper entitled "The Chemical Basis Of Morphogenesis".[8] He proposed a mathematical model in which he explained how different morphogens react together and diffuse through the tissues. During this process a spatial pattern is set up which in turn determines the cell differentiation. Thus whatever pattern we observe in nature is because of this pre-pattern formed during cell differentiation. Turing named this model as "Reaction-Diffusion" model. More specifically he considered those morphogens as activator and inhibitor. The activator stimulated and enhanced the production of the inhibitor,

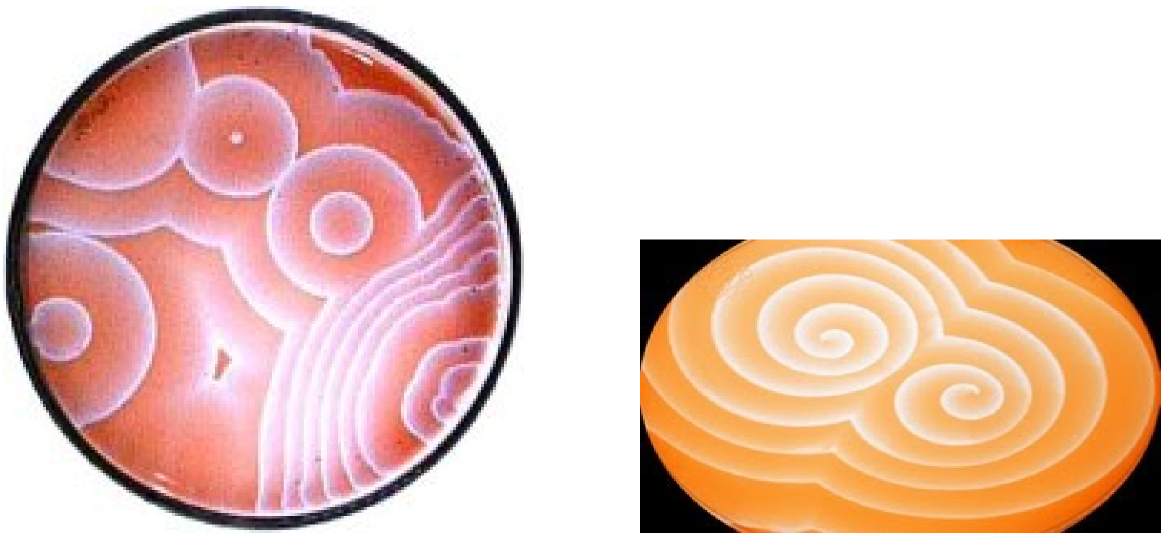

(a) Target patterns (b) Spiral patterns

Figure 1.2: Spatio-temporal patterns in the Belousov-Zhabotinskii reaction

while inhibitor inhibited the formation of the activator. A more general and refined form of the reaction diffusion equation, derived from its original form proposed by Turing is

$$\frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{D}\nabla^2\boldsymbol{u} + \boldsymbol{f}(\boldsymbol{u}, \boldsymbol{p}), \tag{1.1}$$

where $\boldsymbol{u}$ is a vector of chemical concentrations, $\boldsymbol{D}$ is the matrix of diffusion coefficients and $\boldsymbol{f}$ represents chemical coupling with kinetic parameters $\boldsymbol{p}$.

In late 1960s, Prigogine and co-workers[9–11] pointed out clearly that the direct spontaneous transition from a uniform state to a stationary patterned state (the Turing bifurcation) requires that the system involves at least one positive (e.g. auto-catalysis) and one negative feedback (inhibition) process, includes chemical species with appropriately different diffusion coefficients and, last but not the least, operates far from thermodynamic equilibrium. Various scientists extended Turing's approach to pattern formation in the 1970s and 1980s. Among them Hans Meinhardt[12] and James Murray[13,14] are particularly very popular.

The RD model mentioned above (1.1) basically consists of a set of coupled partial differential equations with first order temporal derivatives on the left hand side and second order spatial derivatives on the right hand side. This RD model can contain any number of concentration variables. According to Turing's activator-inhibitor model the RD equation looks like following:

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u + f(u,v) \tag{1.2a}$$
$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + g(u,v) \tag{1.2b}$$

The $f(u,v)$ and $g(u,v)$ are basically kinetic coupling parts which contains coupled kinetic parameters. Turing proposed these two coupling parts to be linear so that it can be analytically solvable. Several non-linear kinetics have been also proposed subsequently. A few proposed non-linear versions of kinetic parts are cited below:

Schnakenberg kinetics[15] :

$$f(u,v) = k_1 - k_2 u + k_3 u^2 v \tag{1.3a}$$
$$g(u,v) = k_4 - k_3 u^2 v \tag{1.3b}$$

Thomas kinetics (adopted by Murray)[16] :

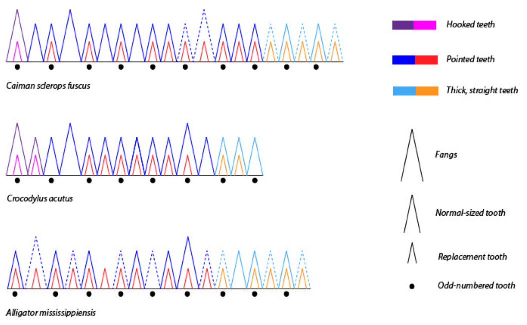
$$f(u,v) = k_1 - k_2 u - h(u,v) \tag{1.4a}$$
$$g(u,v) = k_3 - k_4 v - h(u,v) \tag{1.4b}$$
$$h(u,v) = \frac{k_5 uv}{k_6 + k_7 u + k_8 u^2} \tag{1.4c}$$

Meinhardt kinetics[17] :

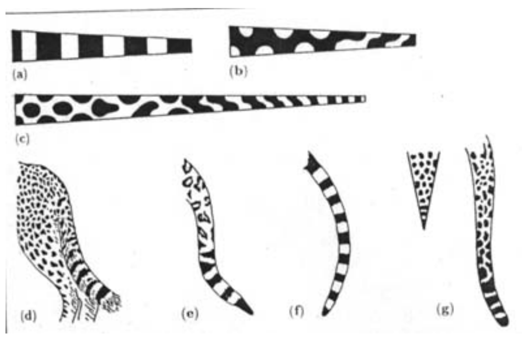
$$f(u,v) = k_1 - k_2 u + \frac{k_3 u^2}{v} \tag{1.5a}$$
$$g(u,v) = k_4 u^2 - k_5 v \tag{1.5b}$$

Out of the above set of three non-linear kinetic equations, kinetics (1.4) and (1.5) are most extensively employed in the RD model for biological pattern generation such as cartilage formation in vertebrate limbs, coloured patterns of butterfly wings, alligator teeth, head regeneration in Hydra, spots of cheetah, stripes of Zebra and many more[1,12,18] . Since the kinetic terms $f(u,v)$ and $g(u,v)$ involve non-linear coupling terms, the equations are not analytically solvable. One has to resort to numerical methods. It is worth mentioning here that the RD equation is analogous to the

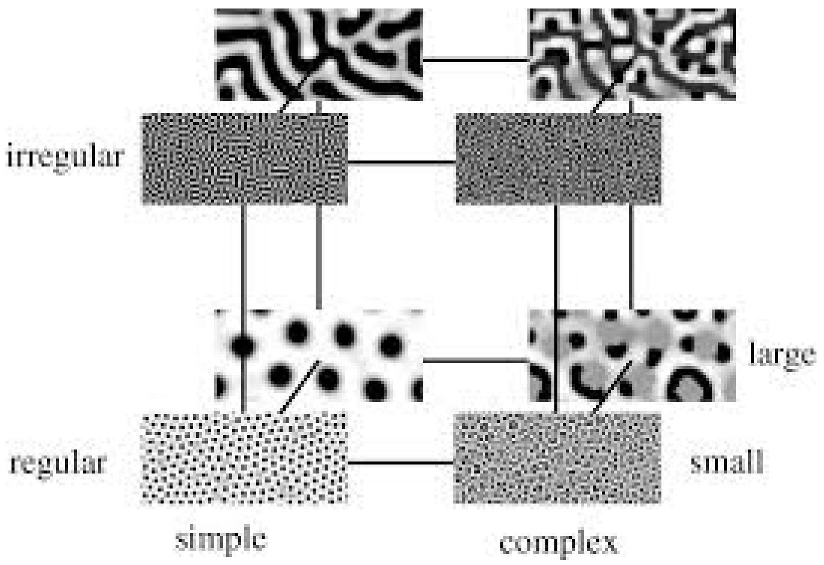
TDSE, except for the imaginary part in the latter.



(a) Computed patterns of alligator teeth



(b) Observed patterns on tails of various organisms, and computed pattern in rectangular domain



(c) Pattern of cheetah spots by mathematical modelling

Figure 1.3: Some of the computed biological patterns using RD equations[1]

The stepwise procedure for solving the RD eq. (1.2) to generate pattern is to first find the uniform steady state values for u and v. It is the solution of $f(u, v) = 0$ and $g(u, v) = 0$ yielding u_0 and v_0, which are constant with respect to time and space. Such a uniform steady state which is stable in the absence of diffusion, can be made unstable in the presence of diffusion and a spatial pattern can evolve. according to Turing this chemical pattern can serve as the required pre-pattern and cells will respond in such a way that a spatial heterogeneous structure would be formed.

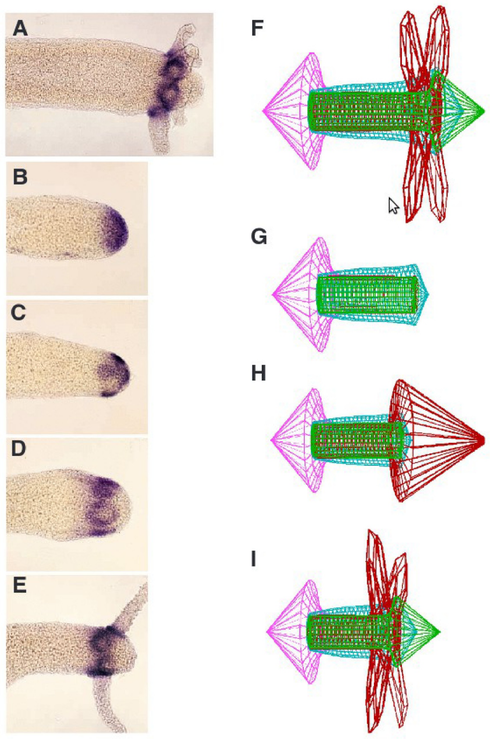


Figure 1.4: *Example of computed patterns by Gierer-Meinhardt kinetics in RD equations*: **(A-E)** Experimental gene expression for tentacle formation in Hydra; **(F-I)** Computed patterns of the same gene expression (brown).[2]

Passion flower (*Passiflora Incarnata*) exhibits a beautiful pattern of alternate violet and white colour on each of its fibrils. Last year, Agastya P. Bhati (MS09010) also worked on this project. Bhati tried to analyse the pattern in one single fibril. But, because of the radially symmetric positions of the coloured bands, this time we have decided to analyse the pattern formation in polar co-ordinates. In this project we have focused on Gierer-Meinhardt model to generate the desired pattern in the Passion Flower using the RD theory. We have used finite difference method as a numerical method.

Chapter 2

Fluid dynamics

2.1 Introduction to the diffusion equation

Let R^n with $n > 1$ be the n^{th} dimensional real space. In particular, we are interested in the cases of $n = 2$ and 3. We assume that Ω is small region in this space. Let *P(t, r)* be the density function of the constituent particles, where t is the time, and $r \in \Omega$ is a point in the n^{th} dimensional space. The dimension of density is number of particles per unit area (if $n = 2$) or unit volume (if $n = 3$).

We need to study the change in function *P(t, r)* with time t and location r. It can happen in two ways: one being diffusion, that is the individual particles move, and the second is production of new particles and/or consumption of existing particles. This may happen due to several reasons, for example, chemical reaction. We model both of these possibilities separately.

We first deal with diffusion. The amount of a substance which passes through a point in space per unit area per unit time is called its **flux** at that point. According to Fick's law of diffusion,[19] the flux of density function *P(t, r)* is a vector pointing from high density region to low density region and with its magnitude proportional to density gradient. Mathematically it can be represented as below:

$$\boldsymbol{J(t, r)} = -D(r)\nabla_r P(t, r), \tag{2.1}$$

where $\boldsymbol{J}$ is the flux of P, *D(r)* is the **diffusion constant** at r, ∇_r is the gradient operator $\nabla_r f(r) = \left(\frac{\partial f}{\partial r_1}, \frac{\partial f}{\partial r_2}, ..., \frac{\partial f}{\partial r_n}\right)$.

The production and/or consumption of constituent particles at any point per unit time, that is the rate of change of the density function, which may occur due to reasons like physical transformation or chemical reactions, is assumed to be given by *f(t, r, P)*, which is called the **reaction rate**. Let O be region in space, then the total number of constituent particles in O is $\int_O P(t,r)dr$, where dr is the infinitesimal volume element. Thus, the rate of change of the total number of particles is

$$\frac{d}{dt}\int_O P(t,r)dr. \tag{2.2}$$

Now we apply the law of mass conservation on this system to derive the Reaction-Diffusion equation. The net production of particles inside the region O is

$$\int_O f(t,r,P(t,r))dr \tag{2.3}$$

and the total out-flux is

$$\int_{\delta O} \boldsymbol{J}(t,r).\boldsymbol{n}(r)dS, \tag{2.4}$$

where δO is the boundary of O and $\boldsymbol{n}(r)$ is the outer normal direction at r. Therefore on conserving the total number of particles we get

$$\frac{d}{dt}\int_O P(t,r)dr = -\int_{\delta O} \boldsymbol{J}(t,r).\boldsymbol{n}(r)dS + \int_O f(t,r,P(t,r))dr. \tag{2.5}$$

From the Divergence Theorem in multi-variable calculus we have

$$\int_{\delta O} \boldsymbol{J}(t,r).\boldsymbol{n}(r)dS = \int_O \boldsymbol{\nabla}.(\boldsymbol{J}(t,r))dr. \tag{2.6}$$

Combining eq. (2.1), (2.5) and (2.6), and interchanging the order of differentiation and integration we obtain

$$\int_O \frac{\partial P(t,r)}{\partial t}dr = \int_0 [\boldsymbol{\nabla}.(D(r)\nabla_r P(t,r)) + f(t,r,P(t,r))]dr \tag{2.7}$$

Since the choice of region O is arbitrary, the differential equation

$$\frac{\partial P(t,r)}{\partial t} = \boldsymbol{\nabla}.(D(r)\nabla_r P(t,r)) + f(t,r,P(t,r)) \tag{2.8}$$

holds for any (t, r). The equation (2.8) is called a **reaction diffusion equation**. Here, $\boldsymbol{\nabla}.(D(r)\nabla_r P(t,r)$ is the diffusion term, which describes the movement of the particles under their density gradient and $f(t,r,P(t,r))$ is the reaction term which describes the reaction occurring in the domain.

The diffusion coefficient *D(r)* may not be a constant always as many systems are heterogeneous. But when the region of the diffusion is approximately homogeneous, we can assume that *D(r)* $\equiv D$, then eq. (2.8) can be simplified to

$$\frac{\partial P}{\partial t} = D \triangle P + f(t, r, P), \tag{2.9}$$

where $\triangle P = \boldsymbol{\nabla}.(\nabla_r P) = \sum_{i=1}^{n} \frac{\partial^2 P}{\partial r_i^2}$ is the Laplacian operator. When there is no reaction, the equation is the **diffusion equation**, as follows

$$\frac{\partial P}{\partial t} = D \triangle P. \tag{2.10}$$

In classical mathematical physics, the equation $T_t = \triangle T$ is called the **heat equation**, where T is the temperature function. Conduction of heat can be considered as a form of diffusion of heat.

2.2 Laplacian (Δ) in polar coordinates

In Cartesian Coordinate, The laplacian is as follows
$\Delta P = \nabla^2 P = \dfrac{\partial^2 P}{\partial x^2} + \dfrac{\partial^2 P}{\partial y^2}$
Now, let's assume, $x = r\cos\theta$ and $y = r\sin\theta$

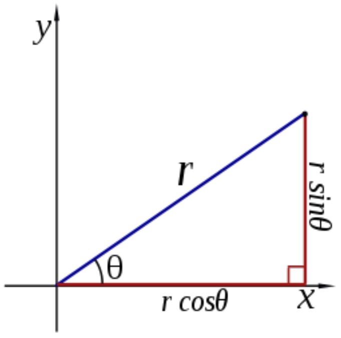


Figure 2.1: polar co-ordinate in 2-D

Then

$$r^2 = x^2 + y^2 \tag{2.11a}$$
$$\cos\theta = \frac{x}{\sqrt{x^2 + y^2}} \tag{2.11b}$$
$$\sin\theta = \frac{y}{\sqrt{x^2 + y^2}} \tag{2.11c}$$

Now if we differentiate the above three equations with respect to r and θ then,
$\dfrac{\partial r}{\partial x} = \cos\theta \quad , \dfrac{\partial r}{\partial y} = \sin\theta$
$\dfrac{\partial \theta}{\partial x} = -\dfrac{\sin\theta}{r} \quad , \dfrac{\partial \theta}{\partial y} = \dfrac{\cos\theta}{r}$
Now calculate separately for $\dfrac{\partial^2 P}{\partial x^2}$ and $\dfrac{\partial^2 P}{\partial y^2}$

$$\frac{\partial P}{\partial x} = \frac{\partial P}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial P}{\partial \theta}\frac{\partial \theta}{\partial x} \tag{2.12a}$$
$$\Rightarrow \frac{\partial P}{\partial x} = \cos\theta(\frac{\partial P}{\partial r}) - \frac{\sin\theta}{r}(\frac{\partial P}{\partial \theta}) = f \tag{2.12b}$$
$$\frac{\partial^2 P}{\partial x^2} = \frac{\partial f}{\partial x} = \frac{\partial f}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial f}{\partial \theta}\frac{\partial \theta}{\partial x} \tag{2.12c}$$

$$\Rightarrow \frac{\partial^2 P}{\partial x^2} = [(\frac{\partial^2 P}{\partial r^2})\cos\theta + \frac{\sin\theta}{r^2}(\frac{\partial P}{\partial \theta}) - \frac{\sin\theta}{r}(\frac{\partial^2 P}{\partial r\,\partial\theta})]\cos\theta + [(\frac{\partial^2 P}{\partial\theta\,\partial r)}\cos\theta - \sin\theta(\frac{\partial P}{\partial r}) - \frac{\cos\theta}{r}(\frac{\partial P}{\partial\theta}) - \frac{\sin\theta}{r}(\frac{\partial^2 P}{\partial\theta^2})](-\frac{\sin\theta}{r}) \quad (2.13a)$$

$$\Rightarrow \frac{\partial^2 P}{\partial x^2} = (\frac{\partial^2 P}{\partial r^2})\cos^2\theta + \frac{\sin 2\theta}{r^2}(\frac{\partial P}{\partial\theta}) - \frac{\sin 2\theta}{r}(\frac{\partial^2 P}{\partial r\,\partial\theta}) + \frac{\sin^2\theta}{r}(\frac{\partial P}{\partial r}) + \frac{\sin^2\theta}{r^2}(\frac{\partial^2 P}{\partial\theta^2}) \quad (2.14)$$

Similarly

$$\frac{\partial^2 P}{\partial y^2} = (\frac{\partial^2 P}{\partial r^2})\sin^2\theta - \frac{\sin 2\theta}{r^2}(\frac{\partial P}{\partial\theta}) + \frac{\sin 2\theta}{r}(\frac{\partial^2 P}{\partial r\partial\theta}) + \frac{\cos^2\theta}{r}(\frac{\partial P}{\partial r}) + \frac{\cos^2\theta}{r^2}(\frac{\partial^2 P}{\partial\theta^2}) \quad (2.15)$$

Now adding eq. (1.4) and eq. (1.5)

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} = \frac{\partial^2 p}{\partial r^2} + \frac{1}{r}(\frac{\partial P}{\partial r}) + \frac{1}{r^2}(\frac{\partial^2 P}{\partial\theta^2}) \quad (2.16)$$

$$\Rightarrow \Delta P = \frac{\partial^2 p}{\partial r^2} + \frac{1}{r}(\frac{\partial P}{\partial r}) + \frac{1}{r^2}(\frac{\partial^2 P}{\partial\theta^2}) \quad (2.17)$$

So, this is how the laplacian (Δ) transform from 2-D Cartesian co-ordinate to polar co-ordintes . This transformation of laplacian (Δ) will be used in our Geirer-Meinhardt model to generate the desired pattern.

Chapter 3

Theory of Reaction-Diffusion model

3.1 The Origin Theory Behind Pattern Formation

In this chapter we will briefly discuss about the theory of pattern formation in living organism. In his seminal paper "The Chemical Basis of Morphogenesis"[8] Turing addressed the issue of how an embryo from its blastula stage, when it is a spherically symmetric mass of cells, gives rise to an organism which is spherically non-symmetric. As he proposed in his paper that the morphogens, which he considered to be activators and inhibitors react together and diffuse through the cells/tissues, they set up a chemical pre-pattern within the uniform homogeneous mass of cells and the cells then differentiate following this pre-pattern. Thus leading to the patterns/forms in the initial uniform mass of cells. Turing proposed a mathematical form to this idea. He used following two equations in his original paper for studying RD of two morphogens X and Y in a linear array of N cells:

$$\frac{dx_r}{dt} = ax_r + by_r + \mu(x_{r+1} - 2x_r + x_{r-1}) \tag{3.1a}$$

$$\frac{dy_r}{dt} = cx_r + dy_r + \nu(y_{r+1} - 2y_r + y_{r-1}) \tag{3.1b}$$

where x_r and y_r are perturbations in the steady state concentrations of X and Y in the r^{th} cell, $a,\ b,\ c,\ d$ are 'marginal reaction rates', μ and ν are cell to cell diffusion constants for X and Y, respectively, $r = 1$ to N.

Using the above set of equations Turing actually demonstrated, under certain conditions of parameters, an initially homogeneous distribution of concentrations of X and y at their steady state could lead to spatially heterogeneous patterns of concentrations stable with time, in response to random perturbations about the steady state. He gave a counter intuitive concept of **diffusion driven instability** which means, the system would be resistant to any random perturbation about the steady state in the absence of diffusion. i.e diffusion has to be there in the system to make the system unstable and to generate pattern. He also suggests that diffusion rate of inhibitor has to be more than that of activator.

Turing actually made **linearity assumption** by which he means that the system never deviated far from the original homogeneous condition. This linearity assumption actually permitted him to replace the general reaction rate by linear ones so that the RD equations become analytically solvable.. He believed that the chemical pre-pattern is formed during the early stages of embryogenesis when such an assumption is valid.He mentioned in his paper, "Its justification lies in the fact that the patterns produced in the early stages when it is valid may be expected to have strong qualitative similarity to those prevailing in the later stages when it is not." (Turing, 1952, p.66). But the linearization led to some stability problems. Turing had also mentioned the possibility of numerically solving non-linear equations using digital computers.

There are various forms of non-linear kinetics have been proposed which differ in their derivation. Each has its own advantage as well as disadvantage and yield different results. Next we are going to discuss few of them in the following chapter.

3.2 Different non-linear kinetics

Since the publication of Turing's paper several RD models have been considered with different kinetic terms. These models are more realistic and are derived in three different ways: (i) empirically, (ii) phenomenologically and (iii) through a hypothetical reaction. Thomas model is based on type(i), Gierer and Meinhardt model is an example of type(ii) and the Schnakenberg model belongs to type(iii). Each of them is discussed one by one.

3.2.1 Thomas Kinetics[3]

In these empirical type models, kinetics are fitted to experimental data. The immobilized-enzyme substrate-inhibition mechanism of Thomas[16] involves the reaction of uric acid (concentration u) with oxygen (concentration v). Both reactants diffuse from a reservoir maintained at constant concentrations u_0 and v_0, respectively, on to a membrane containing the immobilized enzyme uricase. They react in the presence of the enzyme at the empirical rate $\dfrac{k_5uv}{k_6+k_7u+k_8u^2}$ so that

$$f(u,v) = \alpha(u_0-u) - \frac{k_5uv}{k_6+k_7u+k_8u^2} \tag{3.2a}$$

$$g(u,v) = \beta(v_0-v) - \frac{k_5uv}{k_6+k_7u+k_8u^2}, \tag{3.2b}$$

where $k_2=\alpha$, $k_4=\beta$, $k_1=\alpha u_0$, $k_3=\beta v_0$, k_5, k_6, k_7 and k_8 are positive constants.

3.2.2 Meinhardt kinetics[3]

In phenomenological models, the chemicals are considered as activators and inhibitor.

$$f(u,v) = \underbrace{k_1}_{\text{source}} - \underbrace{k_2u}_{\text{linear degradation}} + \underbrace{\frac{k_3u^2}{v}}_{\text{autocatalysis in } u/\text{inhibition from } v} \tag{3.3a}$$

$$g(u,v) = \underbrace{k_4u^2}_{\text{activation by } u} - \underbrace{k_5v}_{\text{linear degradation}} \tag{3.3b}$$

In Gierer-Meinhardt model[17] as shown above, u is the activator; it is produced by autocatalysis and it activates the production of v, which is the inhibitor, inhibiting the production of u.

3.2.3 Schnakenberg Kinetics[3]

Schnakenberg (1979)[15] proposed a series of trimolecular autocatalytic reactions involving two chemicals as follows:

$$X \underset{\alpha_1}{\overset{k_2}{\rightleftharpoons}} A, \qquad B \xrightarrow{\beta_1} Y, \qquad 2X+Y \xrightarrow{k_3} 3X$$

Using the Law of Mass Action, which states that the rate of a reaction is directly proportional to the product of the active concentrations of the reactants, and denoting the concentrations of X, Y, A and B by u, v, α and β, respectively, we have

$$f(u,v) = \alpha_1\alpha - k_2u + k_3u^2v \tag{3.4a}$$
$$g(u,v) = \beta_1\beta - k_3u^2v, \tag{3.4b}$$

where α_1, β_1, k_2 and k_3 are (positive) rate constants, $k_1 = \alpha_1\alpha$ and $k_4 = \beta_1\beta$. Assuming that there is an abundance of A and B, α and β can be considered to be approximately constant, and so are k_1 and k_4.

Remark: To verify Turing structures, a variation in diffusion coefficients is essentially required. For a general two-species RD system as shown below, the ratio may be changed as follows:[20]

$$\begin{aligned}\frac{\partial u}{\partial t} &= D_u\nabla^2u + f(u,v)\\ \frac{\partial v}{\partial t} &= D_v\nabla^2v + g(u,v).\end{aligned}$$

We additionally assume that the activator is involved in a reaction of the form:

$$U + S \underset{r_2}{\overset{r_1}{\rightleftharpoons}} C\cdot$$

Assuming that both S and C are immobile, the RD system is now modified to:

$$\begin{aligned}\frac{\partial u}{\partial t} &= D_u\nabla^2u + f(u,v) - r_1us + r_2c\\ \frac{\partial v}{\partial t} &= D_v\nabla^2v + g(u,v)\\ \frac{\partial c}{\partial t} &= r_1us - r_2c,\end{aligned}$$

where s and c are the concentrations of S and C, respectively, and r_1, r_2 are rate constants. If r_1 and r_2 are large, then using a singular perturbation, c can be approximated in terms of u by $c \equiv ru$, where $r = \dfrac{s_0r_1}{r_2}$ and we have assumed that the concentration of S remains close to its initial value, s_0.

$$c = ru \qquad \Rightarrow \qquad \frac{\partial c}{\partial t} = r\frac{\partial u}{\partial t}.$$

On the addition of the first and fourth equations above, we obtain the following equation for the activator:

$$(1+r)\frac{\partial u}{\partial t} = D_u \nabla^2 u + f(u,v).$$

Thus when $r >> 1$ the diffusion of the activator is greatly reduced.

This demonstrates one way of reducing the effective diffusion rate of the chemical activator by the formation of an immobile complex.

3.3 Dimensionless RD System

To make our life simpler we can always reduce the number of parameters in a model by using appropriate dimensionless quantities. In general, any RD system can be dimensionless and scaled to take the general form[18]

$$\frac{\partial u}{\partial t} = \gamma f(u,v) + \nabla^2 u \tag{3.5a}$$
$$\frac{\partial v}{\partial t} = \gamma g(u,v) + d\nabla^2 v, \tag{3.5b}$$

where d is the ratio of diffusion coefficients and γ can have any of the following interpretations Based on its definition and appearance in the dimensionless equations.[18]

1. $\gamma^{1/2}$ is proportional to the *linear* size of the spatial domain in one dimension and in two dimensions γ is proportional to the area. That is, it can be used as a handle to increase or decrease the size/volume of the domain.

2. γ represents the relative strength of the reaction terms which means, for example, that an increase in γ may represent an increase in the activity of some rate-limiting step in the reaction sequence.

3.4 Diffusion-Driven Instability Conditions: Linear Stability Analysis

Definition: Any state $(u,v) = (u_0, v_0)$ where u_0 and v_0 are constants in time and space, will be called a uniform steady state if it satisfies the eq. (3.5) and the boundary conditions. We take zero flux boundary conditions.

Zero flux boundary conditions are satisfied by any (u_0, v_0), and eq. (3.5) are satisfied by

$$f(u_0, v_0) = g(u_0, v_0) = 0.$$

As u and v represent chemical concentrations.

Definition: Diffusion-driven instability (DDI), sometimes called Turing instability, occurs when a uniform steady state is stable to small perturbations in the absence of diffusion, but becomes unstable to small spatial perturbations when diffusion is present.

to attain DDI, any steady state has to fulfill certain conditions and those conditions are[3]

$$\begin{aligned} &f_u + g_v < 0, \qquad && f_u g_v - f_v g_u > 0, \\ &df_u + g_v > 0, \qquad && (df_u + g_v)^2 - 4d(f_u g_v - f_v g_u) > 0. \end{aligned} \tag{3.6}$$

where $x_i = \dfrac{\partial x}{\partial i}$ for $x =f$, g and $i = u$, v. and

$$D = \begin{pmatrix} 1 & 0 \\ 0 & d \end{pmatrix}.$$

Remark: The conditions eq. (3.6) for DDI yield that $f_u > 0$ and $g_v < 0$. This further implies that there are two possible cases for f_v and g_u since the only restriction on these terms is that $f_v g_u < 0$. So, we can either have $f_v < 0$ and $g_u > 0$ or the other way round. These correspond to qualitatively different reactions. In the former case, u is the activator, and is also self-activating, while v is inhibitor, which inhibits both u and itself. In the latter case, u is the inhibitor, but is self-activating, while, v is the activator, and self-inhibiting. In both cases, v diffuses more quickly. Another notable point is that in the former case, concentrations of the two species are in phase, that is, both are at high or low density in the same region as the pattern grows, while in the latter case, they are out of phase, that is, u is at a high density where v is low and vice-versa. Figure 3.1 below illustrates these features. An example of the first case is the Geirer-Meinhardt kinetics.

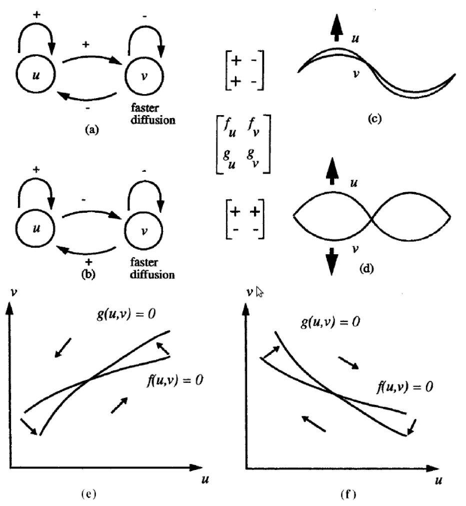

+
+
-
-
u
v
faster diffusion
(a)
+
-
-
+
u
v
faster diffusion
(b)
$\begin{bmatrix} + & - \\ + & - \end{bmatrix}$
$\begin{bmatrix} f_u & f_v \\ g_u & g_v \end{bmatrix}$
$\begin{bmatrix} + & + \\ - & - \end{bmatrix}$
u
v
(c)
u
v
(d)
v
g(u,v) = 0
f(u,v) = 0
u
(e)
v
g(u,v) = 0
f(u,v) = 0
u
(f)

Figure 3.1: Schematic illustration of the two qualitatively different cases of diffusion driven instability. **(a)** self-activating u also activates v, which inhibits both the reactants. The resulting initially growing pattern is shown in **(c)**. **(b)** Here the self-activating u inhibits v but is itself activated by v with the resulting pattern illustrated in **(d)**. The matrices give the signs of f_u , f_v , g_u , g_v evaluated at the steady state. **(e)** and **(f)** The reaction phase planes near the steady state. The arrows indicate the direction of change due to reaction (in the absence of diffusion). Case (e) corresponds to the interactions illustrated in (a) and (c), while that in (f) corresponds to the interactions illustrated in (b) and (d). Reproduced from Murray, J. D. *Mathematical Biology* 2002; Vol. II.

Chapter 4

Measured colour pattern of Passion flower

The *Passiflora Incarnata*, commonly known as Passion flower exhibits a very unique, very fascinating pattern. As shown in figure, the passion flower has a large number of beautiful coloured fibrils. each fibril has alternate bands of violet and white colours and the very uniqueness about the pattern is that each coloured band is non-uniform. If one looks at the flower from top then it can be seen that all the fibrils altogether generate concentric circles of alternate violet and white colours and the radial symmetry that it exhibits is actually the main motive for us to analyse the system in polar co-ordinates. In this project, we have devoted to RD theory, as discussed in the earlier chapters to understand the possible mechanism of the pattern formation in Passion Flower.

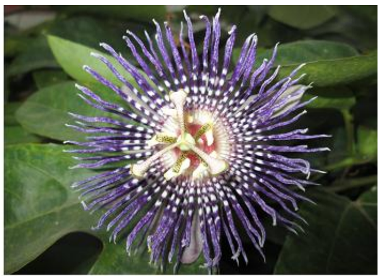


Figure 4.1: Picture of the Passion flower (top view)

Table 4.1: Measured pattern in Passion flower; W: White, V: Violet

Tip of the colour band	Distance from the centre (cm)	Width of the band (cm)
Violet	1.0	1.0
W1	1.6	0.6
V1	1.8	0.2
W2	1.9	0.1
V2	2.1	0.2
W3	2.3	0.2
V3	2.6	0.3
W4	2.9	0.3
V4	3.2	0.3
W5	3.6	0.4
V5	4.1	0.5
W6	4.4	0.3
V6	5.0	0.6
W7	5.2	0.2
V7	8.8	3.6
W8	9.0	0.2

Last year Bhati measured the width of each successive coloured band until the tip of the fibril, using a centimeter ruler (least count = 0.1 cm), for 21 fibrils. The average width of each coloured band was plotted in figure 4.2. Bhati also measured the actual length of 10 fibrils and calculated the average length of the fibrils so as to know the scaling factor of the measured pattern. The average actual length of a fibril is 3.1 cm. The measured average length of a fibril in the printout is 8.8 cm. Measured data are listed in Table 4.1 and a plot of the measured data is shown in Figure 4.2.

$$\text{Scaling factor} = \frac{\text{Actual length}}{\text{Measured length}} = \frac{3.1}{8.8} = 0.35$$

The measured length should be multiplied by 0.35 to get the actual length of the pattern/width of bands.

From the table 4.1 one can see the non-uniformity of the pattern. The width of the violet colour keeps on increasing towards the tip of the fibril and the width of white colour increases initially upto a point , then it starts decreasing and eventually it vanishes on moving towards the tip of the fibril. Beyond a certain point only violet

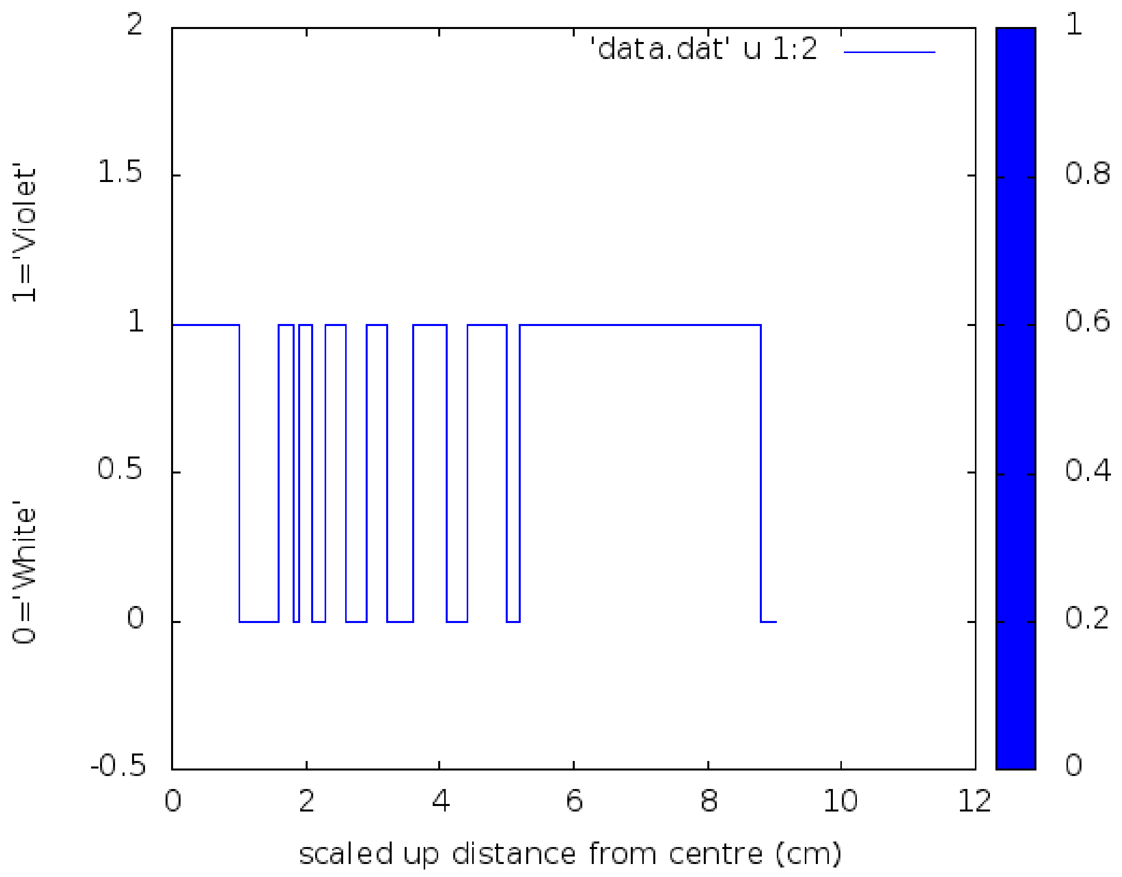


Figure 4.2: Alternation of white and violet colour bands in a Passion flower

colour sustains on the fibril. But the tail end of each fibril exhibits only white colour of very small width.

Chapter 5

Methods and Tools

As mentioned in the previous chapters we studied the system of pattern formation in Passion Flower (*Passiflora Incarnata*) using the "Meinhardt Model" in polar co-ordinate. But before coming to polar co-ordinates here is the overview of Bhati's work.

5.1 Summary of Agastya P. Bhati's Work

Bhati analysed the system in 1-D Cartesian co-ordinate. He considered a linear array of N cells (along X-direction) in a single fibril of the passion flower. He assumed this array of cells to be part of its embryo, during some stage of its embryogenesis, when the chemical pre-pattern is responsible for the colour bands of the fibril is to set up through interactions(reactions and diffusions) of morphogens. The RD model employed to Bhati's system is as follows:

$$\frac{\partial u}{\partial t} = \frac{\rho u^2}{v} - \mu u + D_u \frac{\partial^2 u}{\partial x^2} + \rho_0 \tag{5.1a}$$

$$\frac{\partial v}{\partial t} = \rho^{'} u^2 - \nu v + D_v \frac{\partial^2 v}{\partial x^2} \tag{5.1b}$$

where u is the activator concentration, v is the inhibitor concentration, ρ, μ and ν are first order rate constants, $\rho^{'}$ is a second order rate constant, D_u and D_v are diffusion coefficients of activator and inhibitor respectively.

Bhati then solved the above equations for uniform steady state solutions using zero flux boundary conditions which are satisfied by any (u_0, v_0) and are satisfied by

$$f(u_0, v_0) = g(u_0, v_0) = 0 \tag{5.2a}$$

$$\Rightarrow \frac{\rho u^2}{v} - \mu u + \rho_0 = 0 = \rho^{'} u^2 - \nu v \tag{5.2b}$$

$$\Rightarrow u_0 = \frac{\nu\rho}{\mu\rho^{'}} + \frac{\rho_0}{\mu} = v_0 = \frac{\rho^{'} u_0^2}{\nu}. \tag{5.2c}$$

Bhati found realistic values of the parameters in appropriate units. In a study conducted jointly by the U.S. Fish and Wildlife Service and the U.S. Atomic Energy Commission,[21] he found that the average concentration of Chlorophyll a in Phytoplankton (a type of microalgae) over an year's span was of the order of few μg L^{-1}, which is equivalent to few nmol L^{-1}. The colour in PI is also likely to arise from similar pigments. Therefore, the values of u and v are taken in nmol L^{-1} (nM). The diffusion coefficient of Rhodamine 6G (a dye) in 50/50 methanol/water solution is known to be of the order of 10^{-6} cm^2 s^{-1}.[22] The diffusion coefficient of protein, DNA or other biological molecules in cellular media is expected to be few orders of magnitude smaller than this. The diffusion coefficient of a DNA molecule in *E. coli* is found to be of the order of 10^{-9} cm^2 s^{-1}.[23] he used the values of D_u and D_v of the order of 10^{-2} to 10^{-1} μm^2 s^{-1}. It is worth mentioning here that D_v is taken to be 20 times larger than D_u as the inhibitor needs to diffuse faster than the activator to yield concentration patterns. Experimentally, the value of the first order rate constant for protein-DNA reaction in *E. coli* is known to be of the order of 10^{-2} s^{-1} and that of the second order diffusion-controlled rate constant is of the order of 10^7 M^{-1} s^{-1}.[23] Therefore, we have taken these values in the range of 10^{-2} s^{-1} and 10^{-2} nM^{-1} s^{-1} respectively.

Bhati took a linear array of 3000 cells (along x direction with x ranging from 1 to 3000 microns), with cell number 1 considered to be at the stalk (centre) of the flower. He considered constantly growing domain, starting with 300 microns, increasing it by 300 microns after every 7000 s, up to maximum of 3000 microns. The same domain is considered for polar co-ordinates also.

Bhati studied the evolution of activator and inhibitor concentrations for two different sets of initial conditions (see below). Values of parameters taken were as follows: $\mu = 0.01$ s^{-1}, $\nu = 0.02$ s^{-1}, $\rho = 0.01$ s^{-1}, $\rho^{'} = 0.01$ s^{-1} nM^{-1}, $\rho_0 = 0.00001$ nM s^{-1}, $D_u = 0.02$ μm^2 s^{-1}, $D_v = 0.4$ μm^2 s^{-1} (20 times D_u). Using equation (5.2) the uniform steady state for these set of parameters is $u_0 = 2.001$ nM and $v_0 = 2.002$ nM.

Bhati, hereafter, denote the activator and inhibitor concentrations in cell number i as u[i] and v[i] respectively. Initial Condition 1 (IC1) has a very high concentration at one end of the array of cells with a uniform steady state over the rest of the domain as follows: $u[1] = 2001$ nM ($1000u_0$), u[i] $= u_0$ (i = 2 to 3000), v[i] $= v_0$ (i = 1 to 3000). Initial condition 2 (IC2) has a slightly higher concentration at one end, gradually approaching the uniform steady state with increasing cell number as follows: $u[1] = v$[i] $= 15$ nM, thereafter gradually decreasing in steps of 1 nM with increasing cell number up to $u[13] = v[13] = 3$ nM, and then uniform steady state u[i] $= u_0$, v[i] $= v_0$ (i = 14 to 3000). He used 3-point finite difference formula to evaluate the second order spatial derivative and crank-Nicholson scheme for time evolution.

5.2 Meinhardt's Model In Polar Co-ordinates

$$\frac{\partial u_{(r,\theta)}}{\partial t} = \frac{\rho u^2}{v} - \mu u + D_u\left[\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2}\right] + \rho_0 \tag{5.3a}$$

$$\frac{\partial v_{(r,\theta)}}{\partial t} = \rho^{'} u^2 - \nu u + D_v\left[\frac{\partial^2 v}{\partial r^2} + \frac{1}{r}\frac{\partial v}{\partial r} + \frac{1}{r^2}\frac{\partial^2 v}{\partial \theta^2}\right] \tag{5.3b}$$

Then, we made few assumptions as following:

$$u_{(r,\theta)} = u_r\, u_\theta \qquad v_{(r,\theta)} = v_r\, v_\theta$$
$$u_\theta = c\, e^{i\, m\, \theta} \qquad v_\theta = c^{'}\, e^{i\, m^{'}\, \theta}$$

Here c and $c^{'}$ are normalizing constants. Now putting all these assumptions in equations (5.3), we get

$$\frac{\partial u_r}{\partial t} + iu_r m\frac{\partial \theta}{\partial t} = \frac{\rho u_r^2 (ce^{im\theta})}{v_r(c^{'}e^{im^{'}\theta})} - \mu u_r + D_u\left[\frac{\partial^2 u_r}{\partial r^2} + \frac{1}{r}\frac{\partial u_r}{\partial r} - \frac{m^2 c u_r}{r^2}\right] + \frac{\rho_0}{ce^{im\theta}} \tag{5.4a}$$

$$\frac{\partial v_r}{\partial t} + iv_r m^{'}\frac{\partial \theta}{\partial t} = \rho^{'} u_r^2 (ce^{im\theta}) - \nu v_r + D_v\left[\frac{\partial^2 v_r}{\partial r^2} + \frac{1}{r}\frac{\partial v_r}{\partial r} - \frac{m^{'2} c^{'} v_r}{r^2}\right] \tag{5.4b}$$

Now we have taken two cases such as $m = m^{'} = 0$ and $m = m^{'} = 1$. Therefore

- FOR $m = m' = 0$

In this case $c = c' = 1$

$$\frac{\partial u_r}{\partial t} = \frac{\rho\, u_r^2}{v_r} - \mu\, u_r + D_u \left[\frac{\partial^2 u_r}{\partial r^2} + \frac{1}{r}\frac{\partial u_r}{\partial r} \right] + \rho_0 \tag{5.5a}$$

$$\frac{\partial v_r}{\partial t} = \rho'\, u_r^2 - \nu\, v_r + D_v \left[\frac{\partial^2 v_r}{\partial r^2} + \frac{1}{r}\, \frac{\partial v_r}{\partial r} \right] \tag{5.5b}$$

- FOR $m = m' = 1$

In this case $c = c' = \dfrac{1}{\sqrt{2\pi}}$

$$\frac{\partial u_r}{\partial t} + i u_r \frac{\partial \theta}{\partial t} = \frac{\rho u_r^2}{v_r} - \mu u_r + D_u \left[\frac{\partial^2 u_r}{\partial r^2} + \frac{1}{r}\frac{\partial u_r}{\partial r} - \frac{c\, u_r}{r^2} \right] + \frac{\rho_0}{c e^{i\theta}} \tag{5.6a}$$

$$\frac{\partial v_r}{\partial t} + i v_r \frac{\partial \theta}{\partial t} = \rho' u_r^2 (c e^{i\theta}) - \nu v_r + D_v \left[\frac{\partial^2 v_r}{\partial r^2} + \frac{1}{r}\frac{\partial v_r}{\partial r} - \frac{c' v_r}{r^2} \right] \tag{5.6b}$$

Because of the i factor we have to separate the equations into real part and imaginary part. So,

- REAL PART

$$\frac{\partial u_r}{\partial t} = \frac{\rho\, u_r^2}{v_r} - \mu\, u_r + D_u \left[\frac{\partial^2 u_r}{\partial r^2} + \frac{1}{r}\frac{\partial u_r}{\partial r} - \frac{c\, u_r}{r^2} \right] + \frac{\rho_0\, \cos\theta}{c} \tag{5.7a}$$

$$\frac{\partial v_r}{\partial t} = \rho'\, u_r^2\, c\, \cos\theta - \nu\, v_r + D_v \left[\frac{\partial^2 v_r}{\partial r^2} + \frac{1}{r}\, \frac{\partial v_r}{\partial r} - \frac{c'\, v_r}{r^2} \right] \tag{5.7b}$$

- IMAGINARY PART

$$\frac{\partial \theta}{\partial t} = - \frac{\rho_0\, \sin\theta}{c\, u_r} \tag{5.8a}$$

$$\frac{\partial \theta}{\partial t} = \frac{\rho'\, u_r^2\, c\, \sin\theta}{v_r} \tag{5.8b}$$

Now both $m = m' = 0$ and $m = m' = 1$ parts are solved to get uniform steady state value for $(u, v) = (u_0, v_0)$ where u_0 and v_0 are constants in time and space. we take zero flux boundary conditions, which are satisfied by any (u_0, v_0).

- For $m = m' = 0$

$$f(u_0, v_0) = g(u_0, v_0) = 0 \tag{5.9a}$$

$$\Rightarrow \frac{\rho u^2}{v} - \mu u + \rho_0 = 0 = \rho' u^2 - \nu v \tag{5.9b}$$

$$\Rightarrow u_0 = \frac{\nu\rho}{\mu\rho'} + \frac{\rho_0}{\mu} = v_0 = \frac{\rho' u_0^2}{\nu}. \tag{5.9c}$$

- For $m = m' = 1$

$$f(u_0, v_0) = g(u_0, v_0) = 0 \tag{5.10a}$$

$$\Rightarrow \frac{\rho u_r^2}{v} - \mu u_r + \frac{\rho_0 \cos\theta}{c} = 0 = \rho' u_r^2 \cos\theta - \nu v_r \tag{5.10b}$$

$$\Rightarrow u_0 = \frac{\nu\rho}{\mu\rho' c\cos\theta} + \frac{\rho_0 \cos\theta}{\mu c} = v_0 = \frac{\rho' u_0^2 c\cos\theta}{\nu} \tag{5.10c}$$

Here also we have considered the same values for the parameters as Bhati did. In our case, the uniform steady state values (u_0, v_0) are:

- For $m = m' = 0$, $u_0 = 2.001$ nM and $v_0 = 2.002$ nM.
- For $m = m' = 1$, $u_0 = 5.791$ nM and $v_0 = 5.793$ nM.

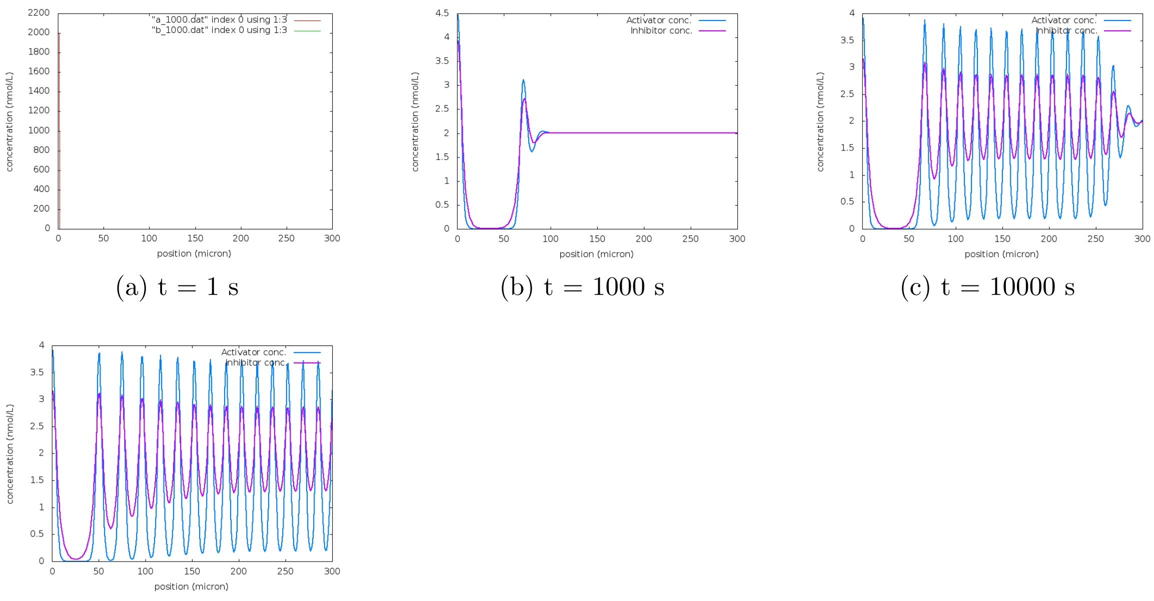
These equations are same as the heat equation, mentioned as (2.10) in chapter 2, except for the additional non-linear coupling terms. The standard heat equation is analytically solvable with known solutions for Dirichlet, Neumann as well as mixed boundary conditions. But because of the non-linear coupling terms the heat equation is not analytically solvable. Therefore we have to resort to numerical methods for solving our equations. We chose finite difference method here. a description of FD method is provided in appendix A. 3-point finite difference formula was used to evaluate the second order spatial derivative and Crank-Nicholson scheme was employed for time evolution.

Chapter 6

Results and Discussions

We evolve our system both spatially and temporally using "Finite-Difference" method for 150000 time steps ($\triangle t = 1$). Lets look at summary of Bhati's results.

6.1 Overview of Agastya P. Bhati's results:



(a) t = 1 s

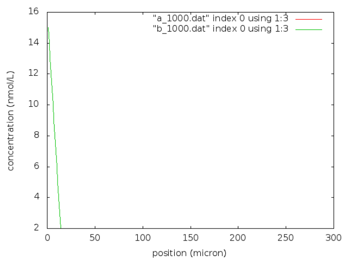
(b) t = 1000 s

(c) t = 10000 s

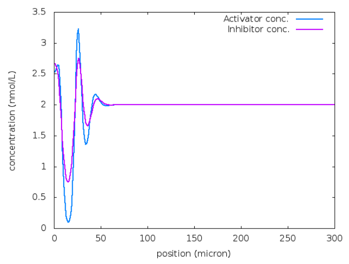
(d) t = 80000 s

Figure 6.1: Evolution of the system along x-axis for initial condition IC1 with concentration (nM) along y-axis

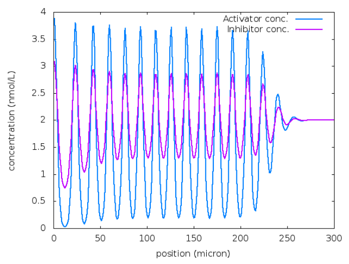
In the case of IC1, the concentration of the activator in cell number 1 is 1000 times larger than that in other cells. Due to such a high concentration of the activator, more activator is produced due to self activation and The concentration of the inhibitor also increases rapidly. Due to the higher diffusivity of the inhibitor, it inhibits the production of the activator, and itself in the vicinity. This feature is clearly visible in figure 6.1 as a long valley between the first two peaks in the concentration profile.



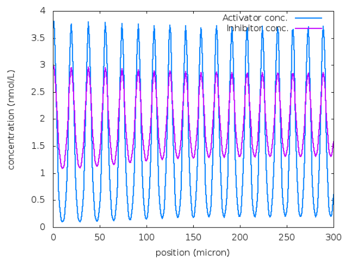
(a) t = 1 s



(b) t = 1000 s



(c) t = 10000 s

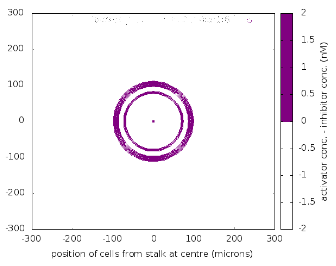


(d) t = 80000 s

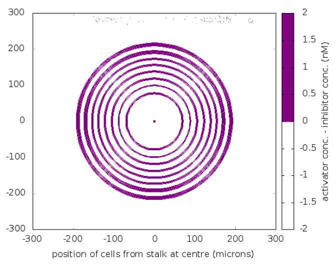
Figure 6.2: Evolution of the system along x-axis for initial condition IC2 with concentration (nM) along y-axis

In the case of IC2, the feature of the valley between the first two peaks he got is not much pronounced as the initial concentration in the cells at one end is not very high as compared to nearby cells. In general, the higher initial concentration at one end quickly spreads over the whole width. This feature is expected due to coupling terms in the Meinhardt's model. Another consequence of such a coupling is the concentration pattern oscillating in space, which is a characteristic of activator-inhibitor models.

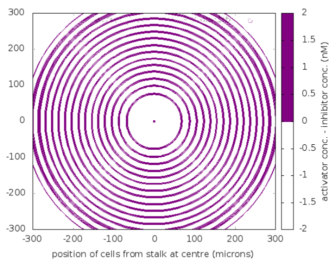
In general, Bhati got oscillating concentration values for both activator and inhibitor. Notably, the amplitude of the oscillation of the activator is higher than that



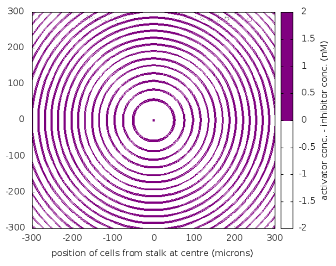
(a) t = 1000 s



(b) t = 5000 s

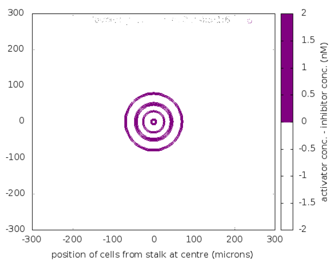


(c) t = 10000 s

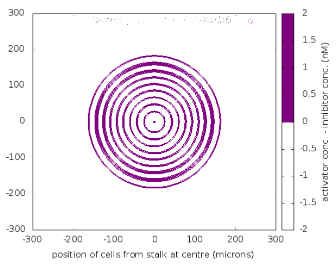


(d) t = 80000 s

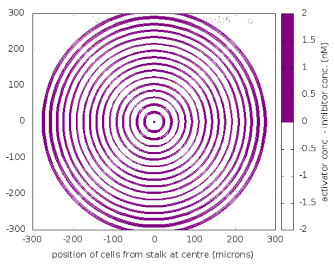
Figure 6.3: Evolution of the system in (x,y) axis for initial condition IC1



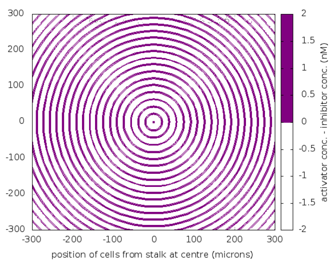
(a) t = 1000 s



(b) t = 5000 s



(c) t = 10000 s



(d) t = 80000 s

Figure 6.4: Evolution of the system in (x,y) axis for initial condition IC2

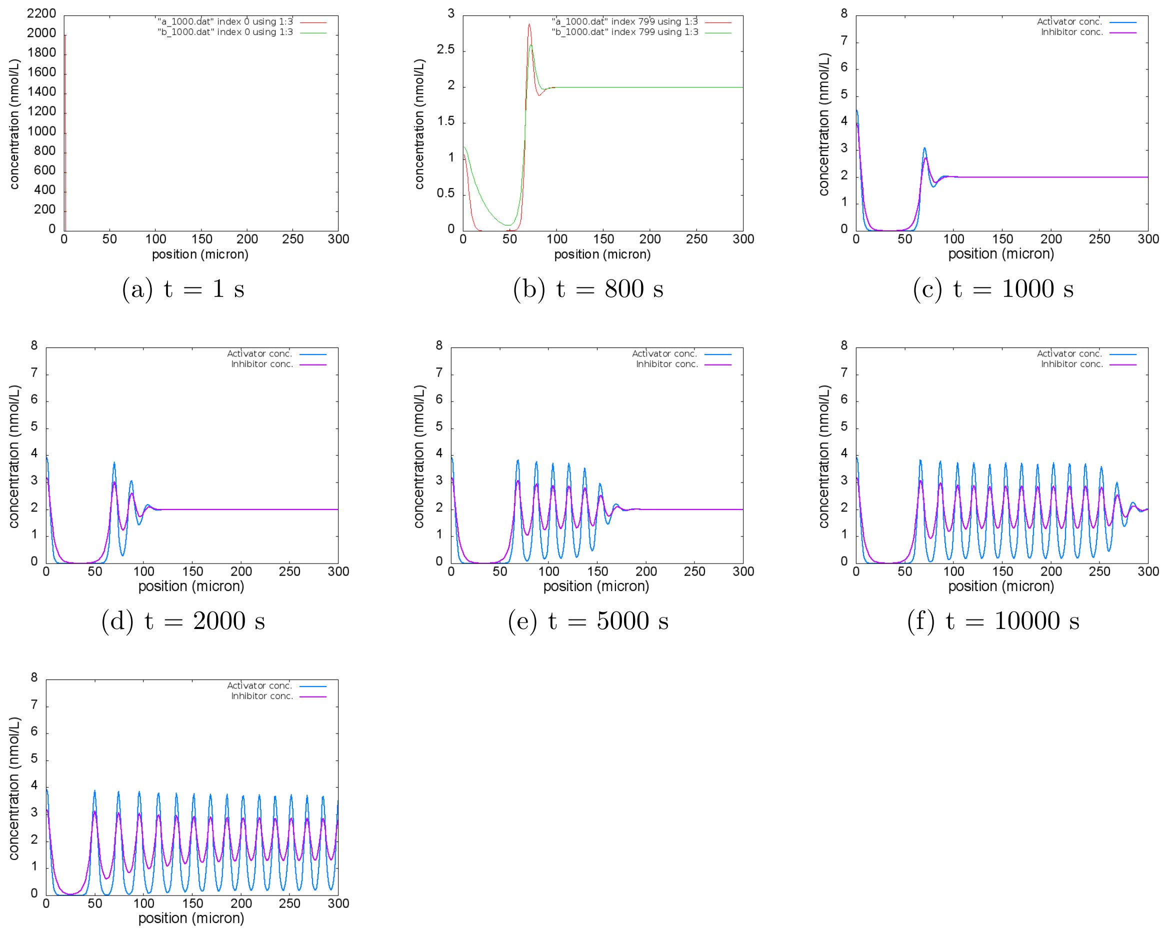
of the inhibitor. This is the reason of activator concentration being higher at crests and lower at troughs than inhibitor concentration. He represented excess activator as violet in colour and excess inhibitor as white in colour.

Bhati was able to get concentric ring pattern from his calculations but of uniform width, which is not the exact pattern that the passion flower exhibits.

6.2 Meinhardt's Model in Polar co-ordinates

In polar co-ordinates also we evolve our system for the same time steps as Bhati did. But, this time we stick to the "initial condition 1" only. First we will look at the system for $m = m' = 0$ and then for $m = m' = 1$.

6.2.1 $m = m' = 0$



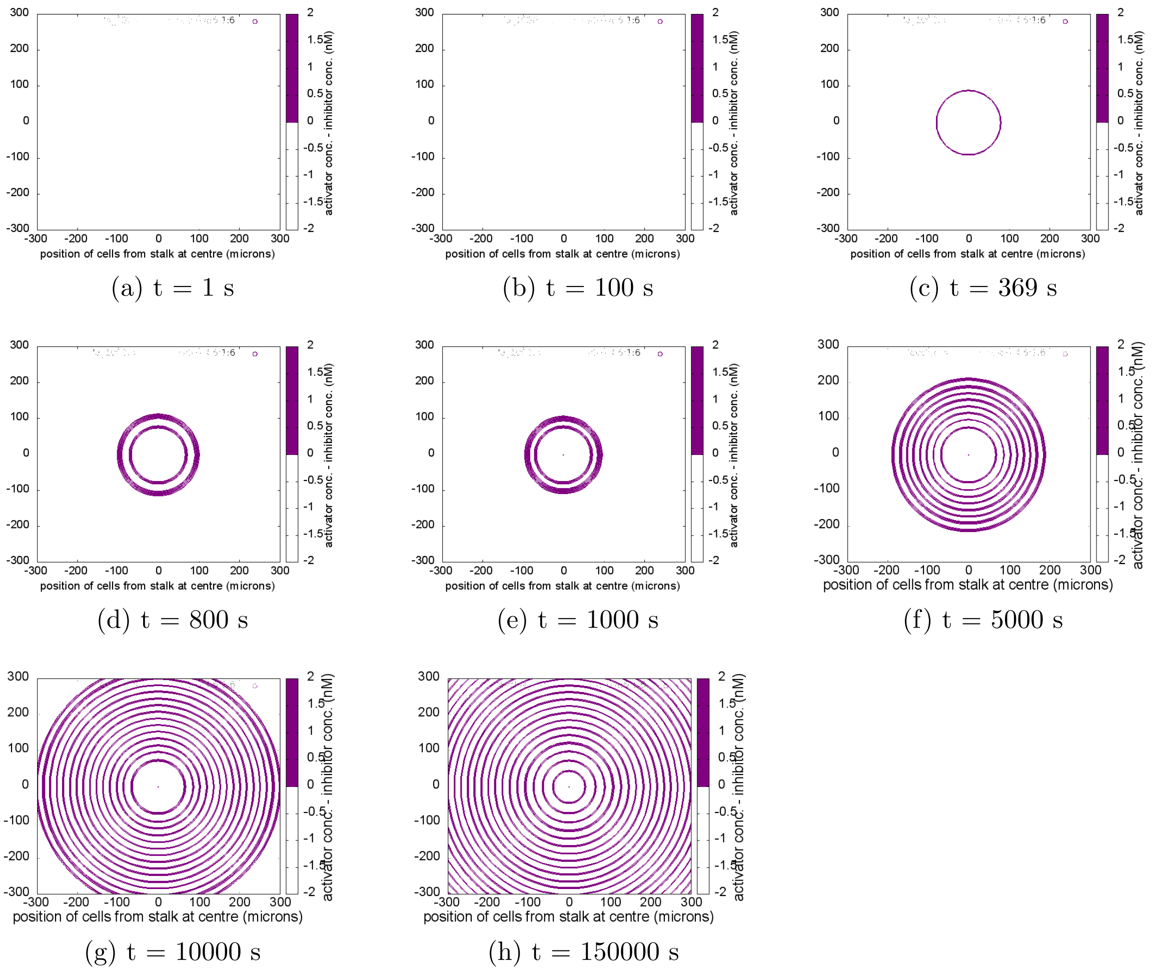
(a) t = 1 s (b) t = 800 s (c) t = 1000 s

(d) t = 2000 s (e) t = 5000 s (f) t = 10000 s

(g) t = 80000 s

Figure 6.5: Evolution of the system along x-axis with concentration (nM) along y-axis for $m = m' = 0$

As we can see from 6.5, there is not much different in the oscillation pattern of both activator and inhibitor concentration from Bhati's result (6.1). But we can see the non-uniformity in the widths of concentric bands (6.6) as the system evolves.



(a) t = 1 s (b) t = 100 s (c) t = 369 s (d) t = 800 s (e) t = 1000 s (f) t = 5000 s (g) t = 10000 s (h) t = 150000 s

Figure 6.6: Evolution of concentric circular pattern for $m = m^{'} = 0$, violet = activator and white = inhibitor

We also have changed the values of the parameters and tried to generate the pattern. We did see a change in the pattern while varying the values of μ, ν, ρ and $\rho^{'}$.

On increasing the values of μ and ν by 10 percent we can see form 6.7 that the concentric rings evolve in a different way. At $t = 3000s$ we can see a circular ring emerging from boundary, merges with the concentric rings evolving from center and then evolve like the usual way. Same effect we can also see if we change the values of

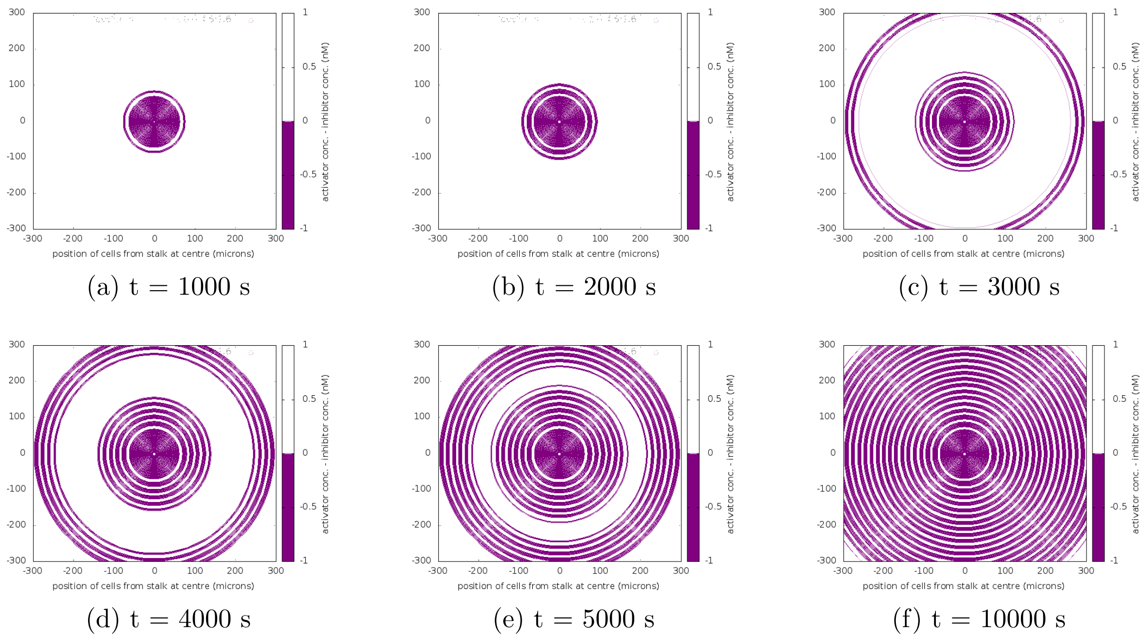


Figure 6.7: Evolution of concentric circular pattern for $m = m' = 0$, $\mu = 0.0110, \nu = 0.0220$, violet = activator and white = inhibitor

ρ and ρ'.

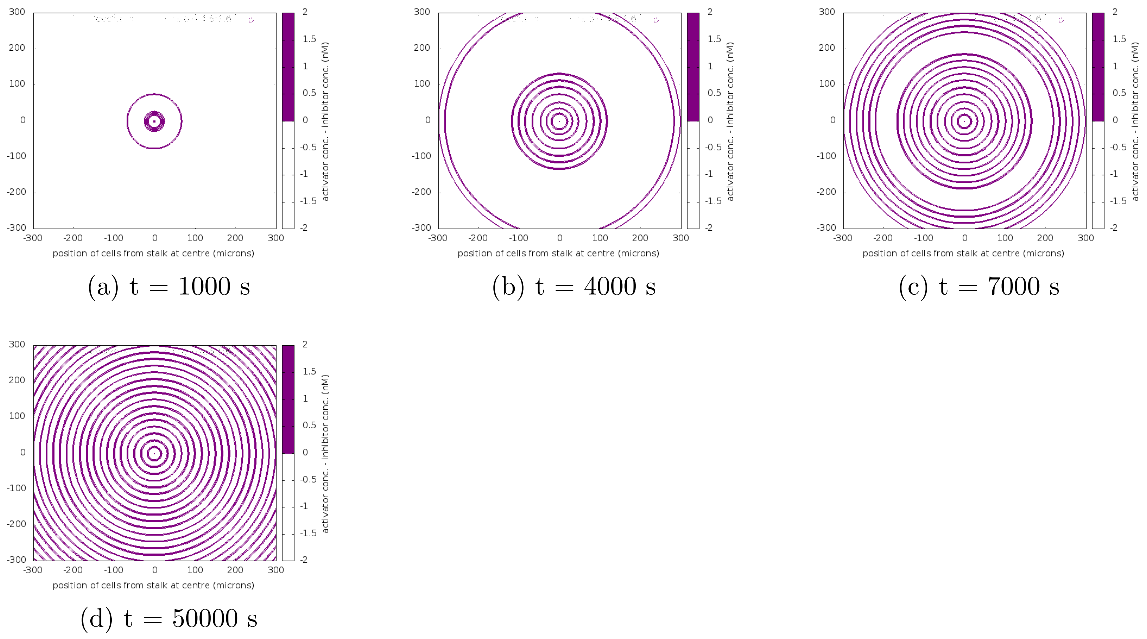
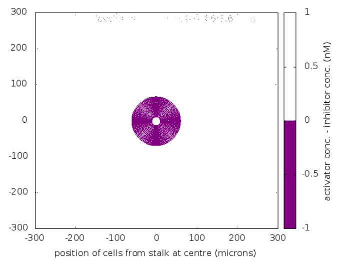
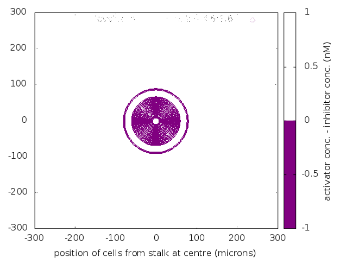


Figure 6.8: Evolution of concentric circular pattern for $m = m' = 0$, $\rho = 0.0100, \rho' = 1.0000$, violet = activator and white = inhibitor

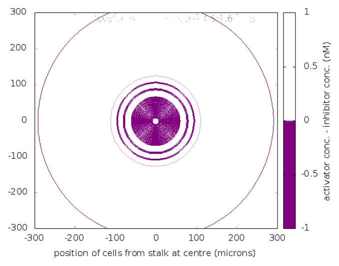
Here also the pattern evolves the same way as on changing the values of μ and ν (6.7) upto $t = 7000s$. But after that we can see uniformity in the widths.



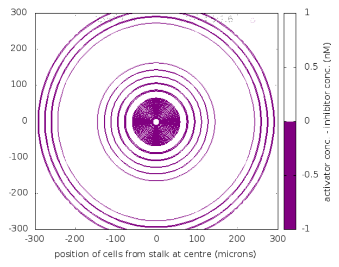
(a) t = 1000 s



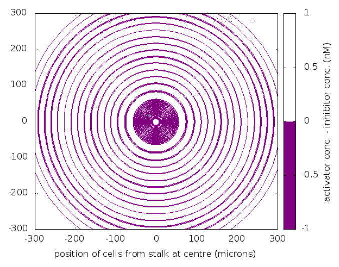
(b) t = 3000 s



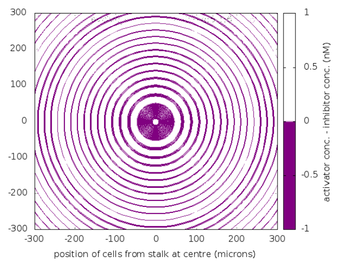
(c) t = 4000 s



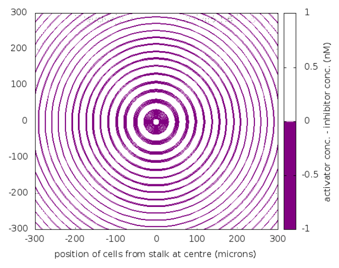
(d) t = 6000 s



(e) t = 10000 s



(f) t = 50000 s

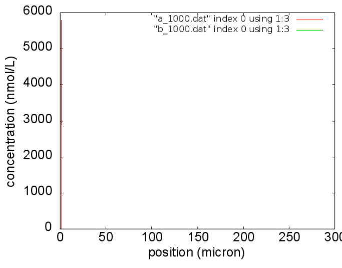


(g) t = 150000 s

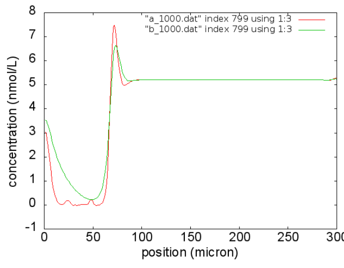
Figure 6.9: Evolution of concentric circular pattern for $m = m' = 0$, $\rho = 0.0020, \rho' = 0.0010$, violet = activator and white = inhibitor

In this case the evolution is somewhat different. As we can see in (6.9) at $t = 1000s$ there is violet colour in the center which is not the case in former case (6.8). Also we have non-uniformity in the widths as the system evolves with time.

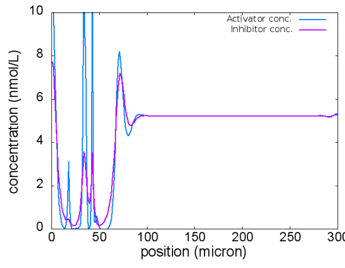
6.2.2 $m = m' = 1$



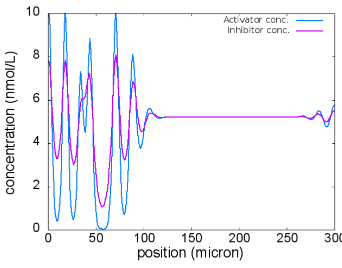
(a) t = 1 s



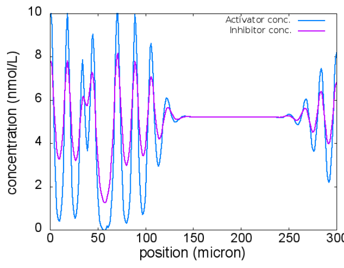
(b) t = 800 s



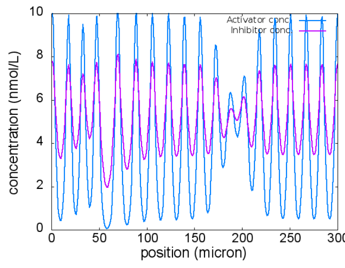
(c) t = 1000 s



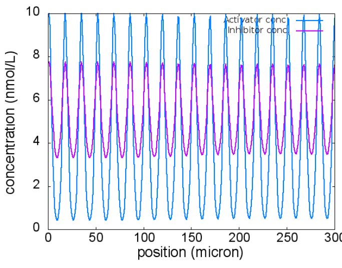
(d) t = 2000 s



(e) t = 3000 s



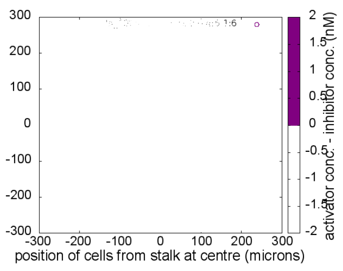
(f) t = 6000 s



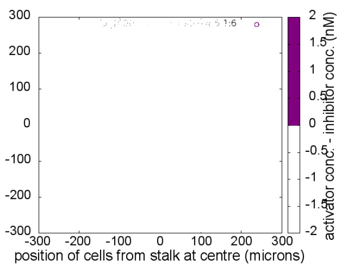
(g) t = 150000 s

Figure 6.10: Evolution of the system along x-axis with concentration (nM) along y-axis for $m = m' = 1$

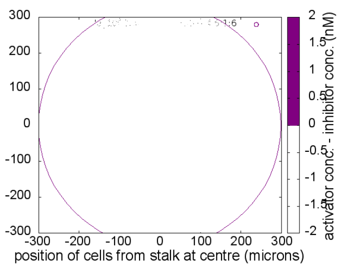
For $m = m' = 1$ we can see lot of fluctuation in the oscillation of activator-inhibitor concentration (6.10). Also at $t = 2000s$ it starts to oscillate from the other end and merge with each other at $t = 6000s$, then move forward, which is very different from the earlier cases. Also in the evolution of circular pattern we can see lots of fluctuations (see 6.11). We can see bright violet and white colour in center at $t = 268s$ and at $t = 522s$ it disappears. May be we can attribute this bright violet colour to that part of the fibrils where only violet colour sustains (see 4.1) (4.2). This fluctuations continue upto $t = 1000s$ and after that it evolves as usual.



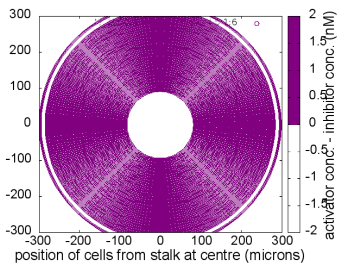
(a) t = 1 s



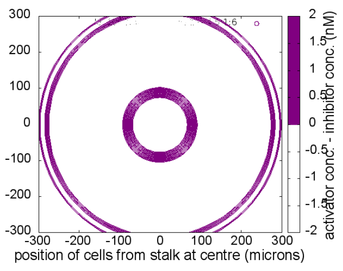
(b) t = 100 s



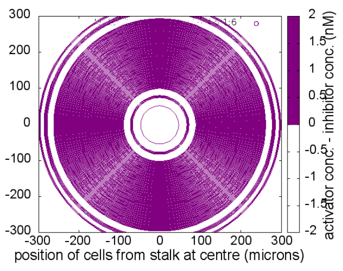
(c) t = 260 s



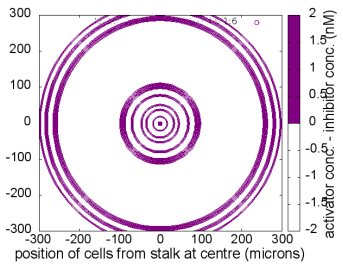
(d) t = 268 s



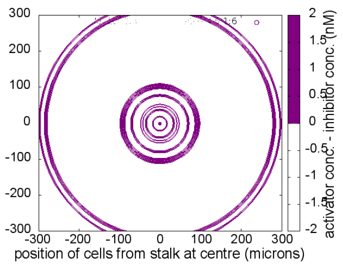
(e) t = 522 s



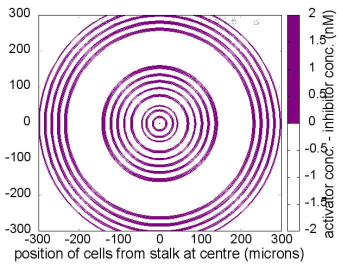
(f) t =808 s



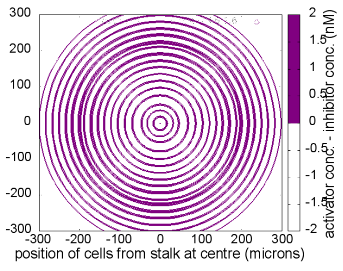
(g) t = 979 s



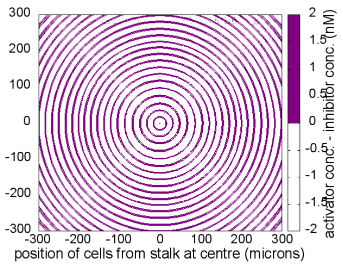
(h) t = 1000 s



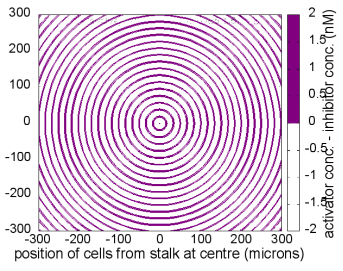
(i) t = 3000 s



(j) t = 5000 s



(k) t = 10000 s



(l) t = 150000 s

Figure 6.11: Evolution of concentric circular pattern for $m = m' = 1$, violet = activator and white = inhibitor

Chapter 7

Conclusions and Outlook

We have discussed about the Gierer-Meinhardt Model kinetics in detail by employing the model successfully in Turing's RD model and tried to apply the model to get the beautiful but complicated pattern of the Passion flower. There are lots of fluctuations we can see in the circular pattern in first few hundred time steps, but Yet we are not able to generate our measured pattern. but still qualitatively we have been able to generate the concentric bands of alternate violet and white colour of non-uniform widths using this activator-inhibitor model.

We have also varied the values of μ, ν, ρ, $\rho^{'}$ for $m = m^{'} = 0$. Here also we have seen the non-uniformity in the concentric coloured pattern but not the exact pattern of the flower. One possibility may be the initial condition that we have considered is not sufficient enough to make the system unstable. May be we can play with the initial condition also. Another possibility to generate the measured pattern may be the selection of right numerical method. We can also resort to other numerical methods like "Runge-Kutta". We can also try Fast-Fourier transform.

We plan to continue our work to explore the above mentioned possibilities. We first need to properly analyse their effects and come up with a mathematical framework for the same. Then using it we hope to be able to logically choose the parameter values and vary them accordingly with time and/or space so as to arrive at our desired pattern.

Bibliography

[1] Murray, J. D. *J. Theo. Bio.* **1981**, *88*, 161–199.

[2] Meinhardt, H. *Int. J. Dev. Biol.* **2012**, *56*, 447–462.

[3] Bhati, A. P. *Understanding the Mechanism of Pattern Formation in Passion Flower. MS thesis*; IISER Mohali: Mohali, 2014.

[4] Field, R. J.; Burger, M. *Oscillations and Travelling Waves in Chemical Systems*; Wiley: New York, 1985.

[5] Field, R. J.; Koros, E.; Noyes, R. M. *J. Am. Chem. Soc* **1972**, *94*, 8649.

[6] Wolpert, L. *J. Theo. Bio* **1969**, *25*, 1–47.

[7] Bard, J. *J. Theo. Bio* **1981**, *93*, 363–385.

[8] Turing, A. M. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* **1952**,

[9] Lefever, R.; Prigogine, I. *48*, 1695–1700.

[10] Nicolis, G.; Prigogine, I. *Self-organization in nonequilibrium systems*; Wiley: New York, 1977.

[11] Glandsdorff, P.; Prigogine, I. *Thermodynamic Theory of Structure, stability and fluctuations*; Wiley: New York, 1971.

[12] Meinhardt, H. *Models of Biological Pattern Formation*; Academic Press: New York, 1982.

[13] Winfree, A. T. *Science* **1973**, *181*, 937–939.

[14] Zaikin, A. N.; Zhabotinskii, A. M. *Nature* **1970**, *225*, 535–537.

[15] Schnackenberg, J. *J. Theo. Bio* **1979**, *81*, 389–400.

[16] Thomas, D.; Kernezev, J. P. *Analysis and Control of Immobilized Enzyme Systems*; Springer-Verlag: Berlin-Heidelberg-New York, 1975.

[17] Gierer, A.; Meinhardt, H. *Kybernetik* **1972**, *12*, 30–39.

[18] Murray, J. D. *Mathematical Biology*; Springer: Berlin, Heidelberg, 2002; Vol. II.

[19] Fick, A. *Phil. Mag.* **1855**, *10*, 30.

[20] Karpal, R. *Physica D* **1995**, *86*, 149.

[21] Williams, R. B.; Murdoch, M. B. *Assosiation for the Sciences of Limnology and Oceanography* **2003**,

[22] Culbertson, C. T. *Talanta* **2002**, *56*, 365–373.

[23] Winter, R. B.; Berg, O. G.; Hippe, P. H. *Biochemistry* **1981**, *20*, 6961–6977.

Appendices

Appendix A

Finite Difference methods

Let's consider a rectangular domain D: $0 \leq x \leq a$ and $0 \leq y \leq b$ and draw straight lines parallel to x-axis and y-axis as shown in the figure A.1 such that $x_i = i * \triangle x$ for $i = 1, 2, 3, \cdots n\text{-}1$ and $y_j = j * \triangle y$ for $j = 1, 2, 3, \cdots m\text{-}1$ where $\triangle x$ and $\triangle y$ are small positive steplengths obtained by $\triangle x = \dfrac{a}{n}$ and $\triangle y = \dfrac{b}{m}$.

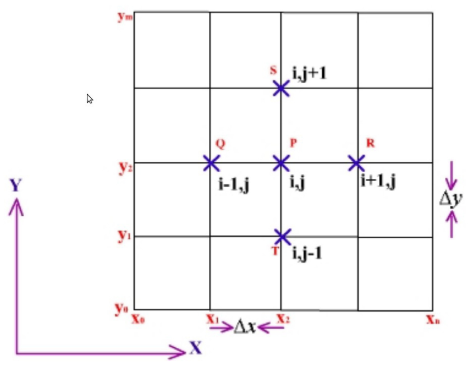


Figure A.1: Discretised rectangular domain

Let $P_{i,j} = P(x_i, y_j)$ be any point in the region D then the co-ordinates x_i and y_j

can be obtained by

$$\begin{aligned} x_i &= x_0 + i * \triangle x \\ y_j &= y_0 + j * \triangle y, \end{aligned} \tag{A.1}$$

where (x_0, y_0) are the coordinates of the left bottom most point of the rectangle, that is (0,0) in the present case. If $u(x, y)$ is any continuous function with all necessary derivatives existing in D then

$$u_{i\pm1,j} = u(x_i \pm \triangle x, y_j) = u_{i,j} \pm \triangle x \frac{\partial u_{i,j}}{\partial x} + \frac{\triangle x^2}{2!}\frac{\partial^2 u_{i,j}}{\partial x^2} \pm \cdots\cdots \tag{A.2a}$$

$$u_{i,j\pm1} = u(x_i, y_j \pm \triangle y) = u_{i,j} \pm \triangle y \frac{\partial u_{i,j}}{\partial y} + \frac{\triangle y^2}{2!}\frac{\partial^2 u_{i,j}}{\partial y^2} \pm \cdots\cdots \tag{A.2b}$$

From eq. (A.2), partial derivatives can be approximated as follows:

$$\begin{aligned} \frac{\partial u_{i,j}}{\partial x} &= \frac{u_{i+1,j} - u_{i-1,j}}{2 \triangle x} + O(\triangle x^2), \qquad \text{(Central difference)} \\ &= \frac{u_{i+1,j} - u_{i,j}}{\triangle x} + O(\triangle x), \qquad \text{(Forward difference)} \\ &= \frac{u_{i,j} - u_{i-1,j}}{\triangle x} + O(\triangle x). \qquad \text{(Backward difference)} \\ \frac{\partial u_{i,j}}{\partial y} &= \frac{u_{i,j+1} - u_{i,j-1}}{2 \triangle y} + O(\triangle y^2), \qquad \text{(Central difference)} \\ &= \frac{u_{i,j+1} - u_{i,j}}{\triangle y} + O(\triangle y), \qquad \text{(Forward difference)} \\ &= \frac{u_{i,j} - u_{i,j-1}}{\triangle y} + O(\triangle y). \qquad \text{(Backward difference)} \\ \frac{\partial^2 u_{i,j}}{\partial x^2} &= \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\triangle x^2} + O(\triangle x^4). \\ \frac{\partial^2 u_{i,j}}{\partial y^2} &= \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\triangle y^2} + O(\triangle y^4). \end{aligned} \tag{A.3}$$

Now we can convert the partial differential equations into difference equations by using these approximations and the resultant system of algebraic equations can be solved using any direct or iterative method. Since the analytical methods for finding solution of second order partial differential equations depend on the type of PDE, the numerical schemes also depend on the type of PDE. For example now we will try to solve a PDE as shown below which is similar to the differential part in Meinhardt model in polar co-ordinates for $m = m^{'} = 0$ and $m = m^{'} = 1$. (see 5.5 and 5.6)

$$\frac{\partial u}{\partial t} = C[\frac{\partial^2 u}{\partial x^2} + \frac{\partial u}{\partial r}] \tag{A.4}$$

A.1 Forward Time and Central Space (FTCS) Scheme

In this method the time derivative term in the one-dimensional heat eq. (A.4) is approximated with forward difference and space derivatives are approximated with second order central differences. Thus, it gives:

$$\frac{u_i^{n+1} - u_i^n}{\triangle t} = C[\frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\triangle x^2} + \frac{u_{i+1}^n - u_{i-1}^n}{2 \triangle x}] \tag{A.5}$$

where $x_i = i \triangle x$ ($i = 0, 1, 2, 3, \cdots N$) and $t_n = n \triangle t$ ($n = 0, 1, 2, 3, \cdots\cdots$). To distinguish between space and time coordinates superscript index n is used for the time coordinate whereas a subscript i is used to represent the space position along x direction. N is the number of points along the x-direction excluding zeroth point.

At any typical node (i, n), the finite difference eq. (A.5) can be rearranged as

$$u_i^{n+1} = u_i^n + r(u_{i-1}^n - 2u_i^n + u_{i+1}^n) + R(u_{i+1}^n - u_{i-1}^n) \tag{A.6}$$

where $r = \frac{c \triangle t}{\triangle x^2}$ and $R = \frac{c \triangle t}{2 \triangle x}$. It gives a formula to compute the unknown concentrations in the domain at various positions at various times. For n=1, the unknown u is first calculated using the initial conditions at t=0 and boundary values at x=0 and x=L (where L is the length of the domain). Once the solution at time step 1 is obtained, the solution at n=2 is calculated in the same manner by making use of the solution at n=1 and the boundary conditions at x=0 and x=L. The same procedure is repeated until the solution reaches a steady state or until the desired time step.

Since eq. (A.6) has only one unknown for any i and n, it is called an explicit scheme. The FTCS scheme is illustrated in Figure A.2.

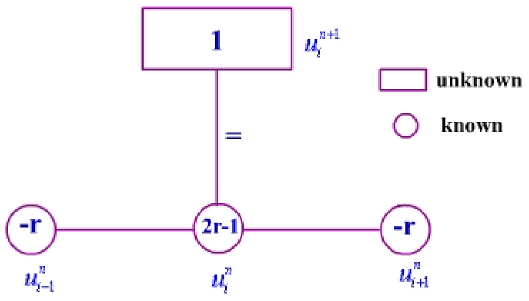


Figure A.2: Sketch for the FTCS scheme

A.2 Backward Time Central Space (BTCS) scheme

If the forward difference approximation for time derivative in the one dimensional heat eq. (A.4) is replaced with the backward difference and the central difference approximation for space derivative is used, then eq. (A.4) can be written as

$$\frac{u_i^n - u_i^{n-1}}{\triangle t} = C[\frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\triangle x^2} + \frac{u_{i+1}^n - u_{i-1}^n}{2 \triangle x} \tag{A.7}$$

where $i = 1, 2, 3, \cdots N$ and $n = 1, 2, 3, \cdots\cdots$
Alternatively,

$$\frac{u_i^{n+1} - u_i^n}{\triangle t} = C[\frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{\triangle x^2} + \frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{2 \triangle x} \tag{A.8}$$

for $i = 1, 2, 3, \cdots N$ and $n = 0, 1, 2, \cdots\cdots$
Rearranging the above equation, we obtain

$$\begin{aligned} u_i^{n+1} - u_i^n &= r(u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}) + R(u_{i+1}^{n+1} - u_{i-1}^{n+1}) \\ \Rightarrow \quad (R-r)u_{i-1}^{n+1} + (1+2r)u_i^{n+1} &- (R+r)u_{i+1}^{n+1} = u_i^n \end{aligned} \tag{A.9}$$

for $i = 1, 2, 3, \cdots N$ and $n = 0, 1, 2, \cdots\cdots$

Since there are three unknown terms in eq. (A.9), the scheme so obtained is referred to as an implicit method. The main drawback of having more than one unknown coefficient in any equation, unlike FTCS method, is that the value of the dependent variable at any typical node say (i, n) cannot be obtained from a single finite difference equation of the node (i, n), and one has to generate a system of equations for each time step separately by varying i. Then for each time step there will be a system of equations equivalent to the number of unknowns in that time step (say N in the present case). This linear system of algebraic equations in N unknowns has to be solved to obtain the solution for each time step . This process has to be repeated until the desired time step is reached. The scheme (A.9) is called the fully implicit method.

A.3 Crank-Nicholson

Schemes (A.5) and (A.8) are two different methods to solve the one dimensional heat equation (A.4). Crank-Nicholson scheme is obtained by taking an average of these two schemes, that is

$$\frac{u_i^{n+1}-u_i^n}{\triangle t}=\frac{C}{2}\left[\frac{u_{i-1}^{n+1}-2u_i^{n+1}+u_{i+1}^{n+1}}{\triangle x^2}+\frac{u_{i+1}^n-u_{i-1}^n}{2\triangle x}\right]+\frac{C}{2}\left[\frac{u_{i-1}^n-2u_i^n+u_{i+1}^n}{\triangle x^2}+\frac{u_{i+1}^{n+1}-u_{i-1}^{n+1}}{2\triangle x}\right]$$

for $i = 1, 2, 3, \cdots N$ and $n = 0, 1, 2, \cdots\cdots$

$$\Rightarrow \quad (\frac{R}{2}-\frac{r}{2})u_{i-1}^{n+1}+(1+r)u_i^{n+1}-(\frac{R}{2}+\frac{r}{2})u_{i+1}^{n+1}=(\frac{r}{2}-\frac{R}{2})u_{i-1}^n+(1-r)u_i^n+(\frac{r}{2}+\frac{R}{2)}u_{i+1}^n \tag{A.10}$$

Where $r=\frac{c\triangle t}{\triangle x^2}$ and $R=\frac{c\triangle t}{2\triangle x}$.

Since more than one unknown is involved for each i in eq. (A.10) Crank-Nicholson scheme is also an implicit scheme. Therefore, one has to solve a system of linear algebraic equations for every time step to get the field variable u. The Crank-Nicholson scheme is illustrated in Figure A.3.

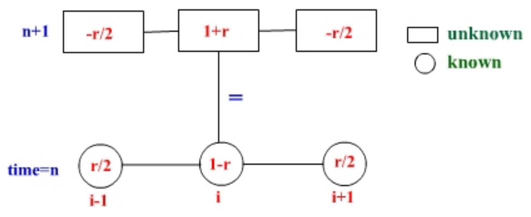


Figure A.3: Sketch for the Crank-Nicholson scheme

The linear algebraic system of equations generated by the Crank-Nicholson method for the time step t^{n+1} are sparse because the finite difference equation obtained at any space node, say i and at time step t^{n+1} has only three unknown coefficients involving space nodes i-1 , i and i+1 at t^{n+1}. In matrix notation, these equations can be written as **AU=B** , where **U** is the unknown vector of order N at any time level t^{n+1} , **B** is the known vector of order N, which involves the values of **U** at the n^{th} time step and **A** is the coefficient square matrix of order N $\times$ N with a tri-diagonal structure as follows:

$$\mathbf{A} = \begin{bmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & 0 & \cdots & 0 & 0 & a_N & b_N \end{bmatrix}$$

Such a matrix is called a tri-diagonal matrix and the system of equations with tridiagonal coefficient matrix is called tridiagonal system. Though direct solvers like Gauss elimination and LU decomposition can be used to solve these systems there are some special schemes available to solve tridiagonal systems. One of them is Thomas algorithm which exploits the tridiagonal nature of the coefficient matrix. Thomas algorithm is similar to Gauss elimination. However, the novelty in the method is that the forward elimination and back substitution parts of Gauss elimination are used only for the non-zero positions of the system **AU=B**.

A.4 Thomas algorithm for tridiagonal system of equations

Let us call the three non-zero diagonals of the above coefficient matrix **A** as a, b and c, where b is the element of the principal diagonal, a is the element of the diagonal before the principal diagonal with zero as the first element and c is the element of the diagonal that lies after the principal diagonal with a zero as the last element. Then the order of a, b and c is equal to the number of unknowns for any time step with the known vector B (with elements d_i). Then the Thomas algorithm can be written as:

```
Do $i = 2$ to $N$ (if $N$ is the number of unknowns)
        $b_i = b_i - a_i \frac{c_{i-1}}{b_{i-1}}$
        $d_i = d_i - a_i \frac{d_{i-1}}{b_{i-1}}$
end Do
        $u_N = \frac{d_N}{b_N}$
    Do $i = N$-1 to 1
        $u_i = \frac{d_i - c_i u_{i+1}}{b_i}$
    end Do
```

A.5 Alternate Direction Implicit method

The heat equation in two dimensions is as follows:

$$\frac{\partial u}{\partial t} = C\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{A.11}$$

Applying Crank-Nicholson scheme to the above equation we get:

$$\begin{aligned}
\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\triangle t} &= \frac{C}{2}\left[\frac{u_{i-1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i+1,j}^{n+1}}{\triangle x^2} + \frac{u_{i-1,j}^{n} - 2u_{i,j}^{n} + u_{i+1,j}^{n}}{\triangle x^2}\right] \\
&+ \frac{C}{2}\left[\frac{u_{i,j-1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j+1}^{n+1}}{\triangle y^2} + \frac{u_{i,j-1}^{n} - 2u_{i,j}^{n} + u_{i,j+1}^{n}}{\triangle y^2}\right] \\
\Rightarrow \quad & -\frac{r_1}{2}\left(u_{i-1,j}^{n+1} + u_{i+1,j}^{n+1}\right) + (1 + r_1 + r_2)u_{i,j}^{n+1} - \frac{r_2}{2}\left(u_{i,j-1}^{n+1} + u_{i,j+1}^{n+1}\right) \\
&= \frac{r_1}{2}\left(u_{i-1,j}^{n} + u_{i+1,j}^{n}\right) + (1 + r_1 + r_2)u_{i,j}^{n} - \frac{r_2}{2}\left(u_{i,j-1}^{n} + u_{i,j+1}^{n}\right)
\end{aligned}$$

where $r_1 = \frac{C \triangle t}{\triangle x^2}$ and $r_2 = \frac{C \triangle t}{\triangle y^2}$,
for $i = 1, 2, 3, \cdots N$, $j = 1, 2, 3, \cdots M$ and $n = 0, 1, 2, \cdots\cdots$

This is certainly a viable scheme; the problem arises in solving the coupled linear equations. Whereas in one dimension the system was tridiagonal, that is no longer true, though the matrix is still very sparse.

Alternate Direction Implicit (ADI) provides a slightly different way of generalizing the Crank-Nicholson algorithm. It is still second-order accurate in time and space, and unconditionally stable, but the equations are easier to solve than in the above case. Here, the idea is to divide each timestep into two steps of size $\frac{\triangle t}{2}$. In each substep, a different dimension is treated implicitly. The equations can be written as follows:
First half time step: implicit along x direction

$$\frac{u_{i,j}^{n+\frac{1}{2}} - u_{i,j}^{n}}{\triangle t/2} = C\left[\frac{u_{i-1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i+1,j}^{n+\frac{1}{2}}}{\triangle x^2} + \frac{u_{i,j-1}^{n} - 2u_{i,j}^{n} + u_{i,j+1}^{n}}{\triangle y^2}\right] \tag{A.12}$$

Second half time step: implicit along y direction

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n+\frac{1}{2}}}{\triangle t/2} = C\left[\frac{u_{i-1,j}^{n+\frac{1}{2}} - 2u_{i,j}^{n+\frac{1}{2}} + u_{i+1,j}^{n+\frac{1}{2}}}{\triangle x^2} + \frac{u_{i,j-1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j+1}^{n+1}}{\triangle y^2}\right] \tag{A.13}$$

for $i = 1, 2, 3, \cdots N$, $j = 1, 2, 3, \cdots M$ and $n = 0, 1, 2, \cdots\cdots$
Rewriting eq. (A.12):

$$-\frac{r_1}{2}u_{i-1,j}^{n+\frac{1}{2}} + (1+r_1)u_{i,j}^{n+\frac{1}{2}} - \frac{r_1}{2}u_{i+1,j}^{n+\frac{1}{2}} = \frac{r_2}{2}u_{i,j-1}^{n} + (1-r_2)u_{i,j}^{n} + \frac{r_2}{2}u_{i,j+1}^{n} \tag{A.14}$$

This is a tridiagonal system which can be solved using Thomas algorithm for the unknown $u_{i,j}$ at the time step $n + \frac{1}{2}$. Similarly eq. (A.13) can be rewritten as:

$$-\frac{r_2}{2}u_{i,j-1}^{n+1} + (1+r_2)u_{i,j}^{n+1} - \frac{r_2}{2}u_{i,j+1}^{n+1} = \frac{r_1}{2}u_{i-1,j}^{n+\frac{1}{2}} + (1-r_1)u_{i,j}^{n+\frac{1}{2}} + \frac{r_1}{2}u_{i+1,j}^{n+\frac{1}{2}} \tag{A.15}$$

Since u terms on the right hand side of eq. (A.15) have already been calculated by solving eq. (A.14), eq. (A.15) is again a tridiagonal system which can also be solved using Thomas algorithm for $u_{i,j}$ at time step n+1. This completes one iteration in time direction and the same is repeated until the desired time step is reached. The advantage of this method is that each substep requires only the solution of a simple tridiagonal system.

Appendix B

C-programs

In this chapter we list our C-programs used for solving the RD equation numerically (both in Cartesian co-ordinate[3] and polar co-ordinates)

B.1 Crank-Nicholson scheme on Meinhardt's model in 1d Cartesian co-ordinate[3]

Following are the programmes for numerical solution of Bhati's one-dimensional Meinhardt's model(both ic1 and ic2) using Crank-Nicholson scheme.

B.1.1 Initial condition 1

ic1.c

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define Da 0.02
#define Db 20*Da
#define nx 3000
//#define ny 100
#define dx 1.0
//#define dy 0.01
#define dt 1.0
#define timeSteps 150000

//void dmatrix_initial (double A[nx][ny]);
void fprintmatrix (double p[nx+2], FILE *f);
void pattern (double p[nx+2], double q[nx+2], FILE *f);
void tridag(double a[], double b[], double c[], double r[], double ↩
    u[], int n);
void neumannbc(double a[nx+2]);
double r1, r3 ;// r2, r4;

int main(void)
{
  /*defining required parameters and variables*/

  int i, n, nxx=300;
  double ra, rb, a0, b0, rho, rho1, ba;// bb; gamma, a, b, alpha, u0, v0;
  double a[nx+2], b[nx+2], f[nx], g[nx];
  double a1[nx], b1[nx], c1[nx], d1[nx], t1[nx];
//  double a2[ny], b2[ny], c2[ny], d2[ny], t2[ny];
  double x, t;
  FILE *fp, *fpo, *fpa, *fpb;

  /*initializing the parameters and variables*/

  ba = 0.00001; //rho0 constant term activator
  //bb = 0.0000; //rho' constant term inhibitor
  ra = 0.0100; //mu
  rb = 0.0200; //nu
  rho = 0.0100; //source density activator
  rho1 = 0.0100; //source density inhibitor

  a0 = (rb*rho)/(ra*rho1) + ba/ra;
  b0 = (a0*a0*rho1)/rb;

  r1 = Da*dt/(dx*dx); //r2 = dt/(dy*dy);
  r3 = Db*dt/(dx*dx); //r4 = d*dt/(dy*dy);

  /*Initialize concentration arrays and coupling terms*/

  for(i=1; i<nx+1; i++)
   {
       //s[i-1] = 0.01*(1. + qc*((double) rand()/RAND_MAX));
       a[i] = a0;// + 0.1*((double) rand()/(RAND_MAX/2) - 1.0);
       b[i] = b0;// + 0.1*((double) rand()/(RAND_MAX/2) - 1.0);
    }

  a[1] = 1000*a[1];
 // a[nx] = 1000*a[nx];
  /*initializing boundary values of concentrations using neumann boundary conditions*/

  a[0] = a[2]; a[nxx+1] = a[nxx-1];
  b[0] = b[2]; b[nxx+1] = b[nxx-1];
 //neumannbc(a);
 //neumannbc(b);

  fp=fopen("a_initial.dat","w");

      if(fp==NULL)
        {
          puts("cannot open file\n");
          exit(1);
        }

  fpo=fopen("b_initial.dat","w");

      if(fpo==NULL)
         {
            puts("cannot open file\n");
            exit(1);
         }

  x = 1.0;
  for(i=1; i<nx+1; i++)
    {
      fprintf(fp,"%d\t%f\n",(int) x, a[i]);
      fprintf(fpo,"%d\t%f\n",(int) x, b[i]);
      x += dx;
    }

  fclose(fp);
  fclose(fpo);

 for(i=1; i<nxx+1; i++)
   {
       // h[i-1] = rho*u[i][j]*v[i][j]/(1+u[i][j]+K*u[i][j]*u[i][j]);
       f[i-1] = rho*a[i]*a[i]/b[i] - ra*a[i] + ba;
       g[i-1] = rho1*a[i]*a[i] - rb*b[i];// + bb;
  }

 fp=fopen("evolve_a.dat","w");

       if(fp==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpo=fopen("evolve_b.dat","w");

       if(fpo==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpa=fopen("a_1000.dat","w");

       if(fpa==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpb=fopen("b_1000.dat","w");

       if(fpb==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 /*Begin time loop*/

 for (n=1; n<=timeSteps; n++)
  {
   t = n*dt;
   /*Printing concentrations at intermediate times*/

 if(n%1000==0)
   {
      x = 1.0;
        for(i=1; i<nx+1; i++)
          {
             fprintf(fp,"%d\t%d\t%f\t0\t0\t%f\n",(int) x,(int) ↩
                t,a[i],a[i]-b[i]);
             x += dx;
          }

     fprintf(fp,"\n\n");

      x = 1.0;
        for(i=1; i<nx+1; i++)
          {
             fprintf(fpo,"%d\t%d\t%f\t0\t0\n",(int) x,(int) t,b[i]);
             x += dx;
          }

    fprintf(fpo,"\n\n");
    }

//Printing concentrations at first 1000 timesteps

  if(n <= 1000)
   {
      x = 1.0;
        for(i=1; i<nxx+1; i++)
          {
             fprintf(fpa,"%d\t%d\t%f\t0\t0\t%f\n",(int) x,(int) ↩
                t,a[i],a[i]-b[i]);
             x += dx;
          }

       fprintf(fpa,"\n\n");

        x = 1.0;
          for(i=1; i<nxx+1; i++)
            {
                fprintf(fpb,"%d\t%d\t%f\t0\t0\n",(int) x,(int) t,b[i]);
                x += dx;
            }

      fprintf(fpb,"\n\n");
      }


   /*Thomas algorithm to update concentration values*/


       /*Initializing tridiagonal coefficients*/

       for(i=0; i<nxx; i++)
        {
         a1[i] = -0.5*r1;
         b1[i] = 1 + r1;
         c1[i] = -0.5*r1;
         d1[i] = 0.5*r1*a[i] + (1-r1)*a[i+1] + 0.5*r1*a[i+2] + ↩
            dt*f[i];
        }

    /*update boundary coefficients for neumann bc*/

         c1[0] = -r1;
         a1[nxx-1] = -r1;

        /*Solving for tridiagonal matrix equation*/

       tridag(a1,b1,c1,d1,t1,nxx);

       /*Back substitution: updating concentration a*/

       for(i=0; i<nxx; i++)
          a[i+1] = t1[i];

       /*repeating the above procedure for concentration b*/

       for(i=0; i<nxx; i++)
        {
         a1[i] = -0.5*r3;
         b1[i] = 1 + r3;
         c1[i] = -0.5*r3;
         d1[i] = 0.5*r3*b[i] + (1-r3)*b[i+1] + 0.5*r3*b[i+2] + ↩
            dt*g[i];
        }

        c1[0] = -r3;
        a1[nxx-1] = -r3;

       tridag(a1,b1,c1,d1,t1,nxx);

       /*Back substitution: Updating concentrations b*/

       for(i=0; i<nxx; i++)
          b[i+1] = t1[i];

     /*End of first half time step*/

   /*Update coupling terms*/


       for(i=1; i<nxx+1; i++)
        {
          //q = s[i-1]*a[i]*a[i];
          f[i-1] = rho*a[i]*a[i]/b[i] - ra*a[i] + ba;
          g[i-1] = rho1*a[i]*a[i] - rb*b[i];// + bb;
        }

   /*Update boundary values*/

```

```
  a[0] = a[2]; a[nxx+1] = a[nxx-1];
  b[0] = b[2]; b[nxx+1] = b[nxx-1];
  //neumannbc(a);
  //neumannbc(b);

  if(n%7000==0 && nxx < nx)
     nxx += 300;

  }

  fclose(fp);
  fclose(fpo);

  fpo=fopen("finala.dat","w");

     if(fpo==NULL)
          {
            puts("cannot open file");
            exit(1);
          }

  fp=fopen("finalb.dat","w");

     if(fp==NULL)
          {
            puts("cannot open file");
            exit(1);
          }

   fprintmatrix(a,fpo);
   fprintmatrix(b,fp);

   fclose(fpo);
   fclose(fp);

  fp=fopen("pattern_final.dat","w");

     if(fp==NULL)
          {
            puts("cannot open file");
            exit(1);
          }

   pattern(b,a,fp);

   fclose(fp);

  return 0;
}

void fprintmatrix (double p[nx+2], FILE *f)
{
 int i;
 double x;

     x = 1.0;
        for(i=1; i<nx+1; i++)
          {
            fprintf(f,"%d\t%f\t0\t0\n",(int) x,p[i]);
            x += dx;
          }
}

void pattern (double p[nx+2], double q[nx+2], FILE *f)
{
 int i;
 double x, y;

     x = 1.0;
        for(i=1; i<nx+1; i++)
          {
            y = p[i] - q[i];
            if(y > 0)
              fprintf(f,"%d\t%f\t0\t0\n",(int) x, y);
            else
```

```
                fprintf(f,"%d\t0.000000\t0\t0\n",(int) x);

            x += dx;
        }
}

void tridag(double a[], double b[], double c[], double r[], double ↩
    u[], int n)
{
  int j;

  double bet, gam[n];

//  gam=dvector(1,n);
  if (b[0] == 0.0) printf("Error 1 in tridag\n");
  u[0]=r[0]/(bet=b[0]);
  for (j=1;j<n;j++) {
    gam[j]=c[j-1]/bet;
    bet=b[j]-a[j]*gam[j];
    if (bet == 0.0) printf("Error 2 in tridag\n");
    u[j]=(r[j]-a[j]*u[j-1])/bet;

  }

  for (j=(n-2);j>=0;j--)
    u[j] -= gam[j+1]*u[j+1];
//  free_dvector(gam,1,n);
}

void neumannbc(double a[nx+2])
{

       a[0]=a[2];
       a[nx+1]=a[nx-1];
}
```

B.1.2 Initial condition 2

ic2.c

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define Da 0.02
#define Db 20*Da
#define nx 3000
//#define ny 100
#define dx 1.0
//#define dy 0.01
#define dt 1.0
#define timeSteps 150000

//void dmatrix_initial (float A[nx][ny]);
void fprintmatrix (float p[nx+2], FILE *f);
void pattern (float p[nx+2], float q[nx+2], FILE *f);
void tridag(float a[], float b[], float c[], float r[], float u[], ↩
    int n);
void neumannbc(float a[nx+2]);
float r1, r3;// r2, r4;

int main(void)
{
  /*defining required parameters and variables*/

  int i, n, nxx=300;
  float ra, rb, a0, b0, rho, rho1, ba;// bb; gamma, a, b, alpha, ↩
      u0, v0;
  float a[nx+2], b[nx+2], f[nx], g[nx];
  float a1[nx], b1[nx], c1[nx], d1[nx], t1[nx];
```

```
//  float a2[ny], b2[ny], c2[ny], d2[ny], t2[ny];
  float x, t;
  FILE *fp, *fpo, *fpa, *fpb;

  /*initializing the parameters and variables*/

  ba = 0.00001; //rho0 constant term activator
  //bb = 0.0000; //rho' constant term inhibitor
  ra = 0.0100; //mu
  rb = 0.0200; //nu
  rho = 0.0100; //source density activator
  rho1 = 0.0100; //source density inhibitor

  a0 = (rb*rho)/(ra*rho1) + ba/ra;
  b0 = (a0*a0*rho1)/rb;

  r1 = Da*dt/(dx*dx); //r2 = dt/(dy*dy);
  r3 = Db*dt/(dx*dx); //r4 = d*dt/(dy*dy);

  /*Initialize concentration arrays and coupling terms*/

  for(i=1; i<nx+1; i++)
   {
       //s[i-1] = 0.01*(1. + qc*((float) rand()/RAND_MAX));
       a[i] = a0;// + 0.1*((float) rand()/(RAND_MAX/2) - 1.0);
       b[i] = b0;// + 0.1*((float) rand()/(RAND_MAX/2) - 1.0);
    }

  n=15;
  for(i=1; n > (int) a0; i++)
    {
     a[i] = (float) n;
     n--;
    }

  n=15;
  for(i=1; n > (int) b0; i++)
     {
      b[i] = (float) n;
      n--;
     }
  //a[1] = 1000*a[1];
 // a[nx] = 1000*a[nx];
  /*initializing boundary values of concentrations using neumann boundary conditions*/

  a[0] = a[2]; a[nxx+1] = a[nxx-1];
  b[0] = b[2]; b[nxx+1] = b[nxx-1];
 //neumannbc(a);
 //neumannbc(b);

  fp=fopen("a_initial.dat","w");

      if(fp==NULL)
        {
          puts("cannot open file\n");
          exit(1);
        }

  fpo=fopen("b_initial.dat","w");

      if(fpo==NULL)
         {
            puts("cannot open file\n");
            exit(1);
         }

  x = 1.0;
  for(i=1; i<nx+1; i++)
    {
      fprintf(fp,"%d\t%f\n",(int) x, a[i]);
      fprintf(fpo,"%d\t%f\n",(int) x, b[i]);
      x += dx;
    }

 fclose(fp);
 fclose(fpo);

for(i=1; i<nxx+1; i++)
  {
      // h[i-1] = rho*u[i][j]*v[i][j]/(1+u[i][j]+K*u[i][j]*u[i][j]);
      f[i-1] = rho*a[i]*a[i]/b[i] - ra*a[i] + ba;
      g[i-1] = rho1*a[i]*a[i] - rb*b[i];// + bb;
  }

 fp=fopen("evolve_a.dat","w");

       if(fp==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpo=fopen("evolve_b.dat","w");

       if(fpo==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpa=fopen("a_1000.dat","w");

       if(fpa==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpb=fopen("b_1000.dat","w");

       if(fpb==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 /*Begin time loop*/

 for (n=1; n<=timeSteps; n++)
  {
   t = n*dt;
   /*Printing concentrations at intermediate times*/

 if(n%1000==0)
   {
      x = 1.0;
        for(i=1; i<nx+1; i++)
          {
             fprintf(fp,"%d\t%d\t%f\t0\t0\t%f\n",(int) x,(int) t,a[i],a[i]-b[i]);
             x += dx;
          }

     fprintf(fp,"\n\n");

      x = 1.0;
        for(i=1; i<nx+1; i++)
          {
             fprintf(fpo,"%d\t%d\t%f\t0\t0\n",(int) x,(int) t,b[i]);
             x += dx;
          }

    fprintf(fpo,"\n\n");
    }

//Printing concentrations at first 1000 timesteps

  if(n <= 1000)
   {
          x = 1.0;
            for(i=1; i<nxx+1; i++)
              {
                 fprintf(fpa,"%d\t%d\t%f\t0\t0\t%f\n",(int) x,(int) ↩
                    t,a[i],a[i]-b[i]);
                 x += dx;
              }

         fprintf(fpa,"\n\n");

          x = 1.0;
            for(i=1; i<nxx+1; i++)
              {
                 fprintf(fpb,"%d\t%d\t%f\t0\t0\n",(int) x,(int) t,b[i]);
                 x += dx;
              }

        fprintf(fpb,"\n\n");
        }


    /*Thomas algorithm to update concentration values*/


        /*Initializing tridiagonal coefficients*/

        for(i=0; i<nxx; i++)
         {
          a1[i] = -0.5*r1;
          b1[i] = 1 + r1;
          c1[i] = -0.5*r1;
          d1[i] = 0.5*r1*a[i] + (1-r1)*a[i+1] + 0.5*r1*a[i+2] + ↩
             dt*f[i];
         }

     /*update boundary coefficients for neumann bc*/

          c1[0] = -r1;
          a1[nxx-1] = -r1;

         /*Solving for tridiagonal matrix equation*/

        tridag(a1,b1,c1,d1,t1,nxx);

        /*Back substitution: updating concentration a*/

        for(i=0; i<nxx; i++)
           a[i+1] = t1[i];

        /*repeating the above procedure for concentration b*/

        for(i=0; i<nxx; i++)
         {
          a1[i] = -0.5*r3;
          b1[i] = 1 + r3;
          c1[i] = -0.5*r3;
          d1[i] = 0.5*r3*b[i] + (1-r3)*b[i+1] + 0.5*r3*b[i+2] + ↩
             dt*g[i];
         }

         c1[0] = -r3;
         a1[nxx-1] = -r3;

        tridag(a1,b1,c1,d1,t1,nxx);

        /*Back substitution: Updating concentrations b*/

        for(i=0; i<nxx; i++)
           b[i+1] = t1[i];

      /*End of first half time step*/

    /*Update coupling terms*/


        for(i=1; i<nxx+1; i++)
```

```
          {
            //q = s[i-1]*a[i]*a[i];
            f[i-1] = rho*a[i]*a[i]/b[i] - ra*a[i] + ba;
            g[i-1] = rho1*a[i]*a[i] - rb*b[i];// + bb;
          }

    /*Update boundary values*/

    a[0] = a[2]; a[nxx+1] = a[nxx-1];
    b[0] = b[2]; b[nxx+1] = b[nxx-1];
    //neumannbc(a);
    //neumannbc(b);

    if(n%7000==0 && nxx < nx)
       nxx += 300;

    }

    fclose(fp);
    fclose(fpo);

    fpo=fopen("finala.dat","w");

       if(fpo==NULL)
            {
              puts("cannot open file");
              exit(1);
            }

    fp=fopen("finalb.dat","w");

       if(fp==NULL)
            {
              puts("cannot open file");
              exit(1);
            }

     fprintmatrix(a,fpo);
     fprintmatrix(b,fp);

     fclose(fpo);
     fclose(fp);

    fp=fopen("pattern_final.dat","w");

       if(fp==NULL)
            {
              puts("cannot open file");
              exit(1);
            }

     pattern(b,a,fp);

     fclose(fp);

    return 0;
}

void fprintmatrix (float p[nx+2], FILE *f)
{
 int i;
 float x;

     x = 1.0;
        for(i=1; i<nx+1; i++)
          {
             fprintf(f,"%d\t%f\n",(int) x,p[i]);
             x += dx;
          }
}

void pattern (float p[nx+2], float q[nx+2], FILE *f)
{
 int i;
 float x, y;
```

```

      x = 1.0;
        for(i=1; i<nx+1; i++)
          {
            y = p[i] - q[i];
            if(y > 0)
              fprintf(f,"%d\t%f\t0\t0\n",(int) x,y);
            else
              fprintf(f,"%d\t0.000000\t0\t0\n",(int) x);

            x += dx;
          }
}

void tridag(float a[], float b[], float c[], float r[], float u[], ↩
    int n)
{
  int j;

  float bet, gam[n];

//  gam=dvector(1,n);
  if (b[0] == 0.0) printf("Error 1 in tridag\n");
  u[0]=r[0]/(bet=b[0]);
  for (j=1;j<n;j++) {
    gam[j]=c[j-1]/bet;
    bet=b[j]-a[j]*gam[j];
    if (bet == 0.0) printf("Error 2 in tridag\n");
    u[j]=(r[j]-a[j]*u[j-1])/bet;

  }

  for (j=(n-2);j>=0;j--)
    u[j] -= gam[j+1]*u[j+1];
//  free_dvector(gam,1,n);
}

void neumannbc(float a[nx+2])
{

       a[0]=a[2];
       a[nx+1]=a[nx-1];
}
```

B.2 Crank-Nicolson scheme on Meinhardt's model in polar co-ordinates

C-programmes to numerically solve the Meinhardt's model in polar co-ordinates using Finite difference methods are listed below:

B.2.1 $m = m' = 0$

m0.c

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define Da 0.02
#define Db 20*Da
#define nx 3000
//#define ny 100
```

```c
#define dx 1.0
//#define dy 0.01
#define dt 1.0
#define timeSteps 150000

//void dmatrix_initial (float A[nx][ny]);
void fprintmatrix (float p[nx+2], FILE *f);
void pattern (float p[nx+2], float q[nx+2], FILE *f);
void tridag(float a[], float b[], float c[], float r[], float u[], int n);
void neumannbc(float a[nx+2]);
float r1, r3, R1, R3;// r2, r4;

int main(void)
{
  /*defining required parameters and variables*/

  int i, n, nxx=300;
  float ra, rb, a0, b0, rho, rho1, ba;// bb; gamma, a, b, alpha, u0, v0;
  float a[nx+2], b[nx+2], f[nx], g[nx];
  float a1[nx], b1[nx], c1[nx], d1[nx], t1[nx];
//  float a2[ny], b2[ny], c2[ny], d2[ny], t2[ny];
  float x, t;
  FILE *fp, *fpo, *fpa, *fpb;

  /*initializing the parameters and variables*/

  ba = 0.00001; //rho0 constant term activator
  //bb = 0.0000; //rho' constant term inhibitor
  ra = 0.0100; //mu
  rb = 0.0200; //nu
  rho = 0.0100; //source density activator
  rho1 = 0.0100; //source density inhibitor

  a0 = (rb*rho)/(ra*rho1) + ba/ra;
  b0 = (a0*a0*rho1)/rb;

  r1 = Da*dt/(dx*dx); //r2 = dt/(dy*dy);
  r3 = Db*dt/(dx*dx); //r4 = d*dt/(dy*dy);

  /*Initialize concentration arrays and coupling terms*/

  for(i=1; i<nx+1; i++)
   {
       //s[i-1] = 0.01*(1. + qc*((float) rand()/RAND_MAX));
       a[i] = a0;// + 0.1*((float) rand()/(RAND_MAX/2) - 1.0);
       b[i] = b0;// + 0.1*((float) rand()/(RAND_MAX/2) - 1.0);
    }

  a[1] = 1000*a[1];
 // a[nx] = 1000*a[nx];
  /*initializing boundary values of concentrations using neumann boundary conditions*/

  a[0] = a[2]; a[nxx+1] = a[nxx-1];
  b[0] = b[2]; b[nxx+1] = b[nxx-1];
 //neumannbc(a);
 //neumannbc(b);

  fp=fopen("a_initial.dat","w");

      if(fp==NULL)
        {
          puts("cannot open file\n");
          exit(1);
        }

  fpo=fopen("b_initial.dat","w");

      if(fpo==NULL)
         {
            puts("cannot open file\n");
            exit(1);
        }

 x = 1.0;
 for(i=1; i<nx+1; i++)
   {
     fprintf(fp,"%d\t%f\n",(int) x, a[i]);
     fprintf(fpo,"%d\t%f\n",(int) x, b[i]);
     x += dx;
   }

 fclose(fp);
 fclose(fpo);

for(i=1; i<nxx+1; i++)
  {
      // h[i-1] = rho*u[i][j]*v[i][j]/(1+u[i][j]+K*u[i][j]*u[i][j]);
      f[i-1] = (rho*a[i]*a[i])/b[i] - ra*a[i] + ba;
      g[i-1] = rho1*a[i]*a[i] - rb*b[i];// + bb;
  }

 fp=fopen("evolve_a.dat","w");

       if(fp==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpo=fopen("evolve_b.dat","w");

       if(fpo==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpa=fopen("a_1000.dat","w");

       if(fpa==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpb=fopen("b_1000.dat","w");

       if(fpb==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 /*Begin time loop*/

 for (n=1; n<=timeSteps; n++)
  {
   t = n*dt;
   /*Printing concentrations at intermediate times*/

 if(n%1000==0)
   {
      x = 1.0;
        for(i=1; i<nx+1; i++)
          {
             fprintf(fp,"%d\t%d\t%f\t0\t0\t%f\n",(int) x,(int) t,a[i],a[i]-b[i]);
             x += dx;
          }

     fprintf(fp,"\n\n");

      x = 1.0;
        for(i=1; i<nx+1; i++)
          {
             fprintf(fpo,"%d\t%d\t%f\t0\t0\n",(int) x,(int) t,b[i]);
```

```
                x += dx;
            }

        fprintf(fpo,"\n\n");
        }

//Printing concentrations at first 1000 timesteps

    if(n <= 1000)
     {
        x = 1.0;
          for(i=1; i<nxx+1; i++)
            {
                fprintf(fpa,"%d\t%d\t%f\t0\t0\t%f\n",(int) x,(int) ↩
                    t,a[i],a[i]-b[i]);
                x += dx;
            }

      fprintf(fpa,"\n\n");

        x = 1.0;
          for(i=1; i<nxx+1; i++)
            {
                fprintf(fpb,"%d\t%d\t%f\t0\t0\n",(int) x,(int) t,b[i]);
                x += dx;
            }

        fprintf(fpb,"\n\n");
        }

        R1 = Da*dt/(2*x*dx);
        R3 = Db*dt/(2*x*dx);

  /*Thomas algorithm to update concentration values*/


      /*Initializing tridiagonal coefficients*/

      for(i=0; i<nxx; i++)
        {
         a1[i] = -0.5*r1+0.5*R1;
         b1[i] = 1 + r1;
         c1[i] = -0.5*r1-0.5*R1;
         d1[i] = (0.5*r1-0.5*R1)*a[i] + (1-r1)*a[i+1] + ↩
             (0.5*r1+0.5*R1)*a[i+2] + dt*f[i];
        }

    /*update boundary coefficients for neumann bc*/

         c1[0] = -r1-R1;
         a1[nxx-1] = -r1+R1;

        /*Solving for tridiagonal matrix equation*/

      tridag(a1,b1,c1,d1,t1,nxx);

      /*Back substitution: updating concentration a*/

      for(i=0; i<nxx; i++)
          a[i+1] = t1[i];

      /*repeating the above procedure for concentration b*/

      for(i=0; i<nxx; i++)
        {
         a1[i] = -0.5*r3+0.5*R3;
         b1[i] = 1 + r3;
         c1[i] = -0.5*r3-0.5*R3;
         d1[i] = (0.5*r3-0.5*R3)*b[i] + (1-r3)*b[i+1] + ↩
             (0.5*r3+0.5*R3)*b[i+2] + dt*g[i];
        }

        c1[0] = -r3-R3;
        a1[nxx-1] = -r3+R3;

      tridag(a1,b1,c1,d1,t1,nxx);

      /*Back substitution: Updating concentrations b*/

      for(i=0; i<nxx; i++)
         b[i+1] = t1[i];

    /*End of first half time step*/

  /*Update coupling terms*/


      for(i=1; i<nxx+1; i++)
       {
         //q = s[i-1]*a[i]*a[i];
         f[i-1] = (rho*a[i]*a[i])/b[i] - ra*a[i] + ba;
         g[i-1] = rho1*a[i]*a[i] - rb*b[i];// + bb;
       }

  /*Update boundary values*/

  a[0] = a[2]; a[nxx+1] = a[nxx-1];
  b[0] = b[2]; b[nxx+1] = b[nxx-1];
  //neumannbc(a);
  //neumannbc(b);

  if(n%7000==0 && nxx < nx)
     nxx += 300;

  }

  fclose(fp);
  fclose(fpo);

  fpo=fopen("finala.dat","w");

     if(fpo==NULL)
          {
            puts("cannot open file");
            exit(1);
          }

  fp=fopen("finalb.dat","w");

     if(fp==NULL)
          {
            puts("cannot open file");
            exit(1);
          }

   fprintmatrix(a,fpo);
   fprintmatrix(b,fp);

   fclose(fpo);
   fclose(fp);

  fp=fopen("pattern_final.dat","w");

     if(fp==NULL)
          {
            puts("cannot open file");
            exit(1);
          }

   pattern(b,a,fp);

   fclose(fp);

  return 0;
}

void fprintmatrix (float p[nx+2], FILE *f)
{
 int i;
 float x;

```

```c
     x = 1.0;
       for(i=1; i<nx+1; i++)
         {
            fprintf(f,"%d\t%f\t0\t0\n",(int) x,p[i]);
            x += dx;
         }
}

void pattern (float p[nx+2], float q[nx+2], FILE *f)
{
 int i;
 float x, y;

     x = 1.0;
       for(i=1; i<nx+1; i++)
         {
            y = p[i] - q[i];
            if(y > 0)
              fprintf(f,"%d\t%f\t0\t0\n",(int) x, y);
            else
              fprintf(f,"%d\t0.000000\t0\t0\n",(int) x);

            x += dx;
         }
}

void tridag(float a[], float b[], float c[], float r[], float u[], ↩
    int n)
{
  int j;

  float bet, gam[n];

//  gam=dvector(1,n);
  if (b[0] == 0.0) printf("Error 1 in tridag\n");
  u[0]=r[0]/(bet=b[0]);
  for (j=1;j<n;j++) {
    gam[j]=c[j-1]/bet;
    bet=b[j]-a[j]*gam[j];
    if (bet == 0.0) printf("Error 2 in tridag\n");
    u[j]=(r[j]-a[j]*u[j-1])/bet;

  }

  for (j=(n-2);j>=0;j--)
    u[j] -= gam[j+1]*u[j+1];
//  free_dvector(gam,1,n);
}

void neumannbc(float a[nx+2])
{

       a[0]=a[2];
       a[nx+1]=a[nx-1];
}
```

B.2.2 $m = m' = 1$

m1.c

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define Da 0.02
#define Db 20*Da
#define nx 3000
//#define ny 100
#define dx 1.0
//#define dy 0.01
#define dt 1.0
```

```
#define PI 3.14159265
#define timeSteps 150000

//void dmatrix_initial (float A[nx][ny]);
void fprintmatrix (float p[nx+2], FILE *f);
void pattern (float p[nx+2], float q[nx+2], FILE *f);
void tridag(float a[], float b[], float c[], float r[], float u[], ↩
    int n);
void neumannbc(float a[nx+2]);
float r1, r3, R1, R3;// r2, r4;

int main(void)
{
  /*defining required parameters and variables*/

  int i, n, nxx=300;
  float ra, rb, a0, b0, rho, rho1, ba;// bb; gamma, a, b, alpha, ↩
      u0, v0;
  float a[nx+2], b[nx+2], f[nx], g[nx];
  float a1[nx], b1[nx], c1[nx], d1[nx], t1[nx];
//  float a2[ny], b2[ny], c2[ny], d2[ny], t2[ny];
  float x, t, e, m, c;
  FILE *fp, *fpo, *fpa, *fpb;

  /*initializing the parameters and variables*/

  ba = 0.00001; //rho0 constant term activator
  //bb = 0.0000; //rho' constant term inhibitor
  ra = 0.0100; //mu
  rb = 0.0200; //nu
  rho = 0.0100; //source density activator
  rho1 = 0.0100; //source density inhibitor
  e = cos(PI/6);
  m = sin(PI/6);
  c = 1/sqrt(2*PI);
  a0 = (rb*rho)/(ra*rho1*c*e) + (ba*e)/(ra*c);
  b0 = (a0*a0*rho1*c*e)/rb;

  r1 = Da*dt/(dx*dx); //r2 = dt/(dy*dy);
  r3 = Db*dt/(dx*dx); //r4 = d*dt/(dy*dy);

  /*Initialize concentration arrays and coupling terms*/

  for(i=1; i<nx+1; i++)
   {
       //s[i-1] = 0.01*(1. + qc*((float) rand()/RAND_MAX));
       a[i] = a0;// + 0.1*((float) rand()/(RAND_MAX/2) - 1.0);
       b[i] = b0;// + 0.1*((float) rand()/(RAND_MAX/2) - 1.0);
    }

  a[1] = 1000*a[1];
 // a[nx] = 1000*a[nx];
  /*initializing boundary values of concentrations using neumann ↩
      boundary conditions*/

  a[0] = a[2]; a[nxx+1] = a[nxx-1];
  b[0] = b[2]; b[nxx+1] = b[nxx-1];
 //neumannbc(a);
 //neumannbc(b);

  fp=fopen("a_initial.dat","w");

      if(fp==NULL)
        {
          puts("cannot open file\n");
          exit(1);
        }

  fpo=fopen("b_initial.dat","w");

      if(fpo==NULL)
         {
            puts("cannot open file\n");
            exit(1);
        }

 x = 1.0;
 for(i=1; i<nx+1; i++)
   {
     fprintf(fp,"%d\t%f\n",(int) x, a[i]);
     fprintf(fpo,"%d\t%f\n",(int) x, b[i]);
     x += dx;
   }

 fclose(fp);
 fclose(fpo);

for(i=1; i<nxx+1; i++)
  {
      // h[i-1] = rho*u[i][j]*v[i][j]/(1+u[i][j]+K*u[i][j]*u[i][j]);
      f[i-1] = (rho*a[i]*a[i])/b[i] - ra*a[i] - (Da*c*a[i])/(x*x) + (ba*e)/c - (ba*m)/(c*a[i]);
      g[i-1] = rho1*a[i]*a[i]*c*e - rb*b[i] - (Db*c*b[i])/(x*x) + (rho1*a[i]*a[i]*c*m)/b[i];// + bb;
  }

 fp=fopen("evolve_a.dat","w");

       if(fp==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpo=fopen("evolve_b.dat","w");

       if(fpo==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpa=fopen("a_1000.dat","w");

       if(fpa==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 fpb=fopen("b_1000.dat","w");

       if(fpb==NULL)
         {
           puts("cannot open file");
           exit(1);
         }

 /*Begin time loop*/

 for (n=1; n<=timeSteps; n++)
  {
   t = n*dt;
   /*Printing concentrations at intermediate times*/

 if(n%1000==0)
   {
      x = 1.0;
        for(i=1; i<nx+1; i++)
          {
            fprintf(fp,"%d\t%d\t%f\t0\t0\t%f\n",(int) x,(int) t,a[i],a[i]-b[i]);
            x += dx;
          }

     fprintf(fp,"\n\n");

      x = 1.0;
        for(i=1; i<nx+1; i++)
```

```
            {
               fprintf(fpo,"%d\t%d\t%f\t0\t0\n",(int) x,(int) t,b[i]);
               x += dx;
            }

      fprintf(fpo,"\n\n");
      }

//Printing concentrations at first 1000 timesteps

    if(n <= 1000)
     {
        x = 1.0;
          for(i=1; i<nxx+1; i++)
            {
               fprintf(fpa,"%d\t%d\t%f\t0\t0\t%f\n",(int) x,(int) ↩
                  t,a[i],a[i]-b[i]);
               x += dx;
            }

       fprintf(fpa,"\n\n");

        x = 1.0;
          for(i=1; i<nxx+1; i++)
            {
               fprintf(fpb,"%d\t%d\t%f\t0\t0\n",(int) x,(int) t,b[i]);
               x += dx;
            }

      fprintf(fpb,"\n\n");
      }

        R1 = Da*dt/(2*x*dx);
        R3 = Db*dt/(2*x*dx);

  /*Thomas algorithm to update concentration values*/


      /*Initializing tridiagonal coefficients*/

      for(i=0; i<nxx; i++)
        {
         a1[i] = -0.5*r1+0.5*R1;
         b1[i] = 1 + r1;
         c1[i] = -0.5*r1-0.5*R1;
         d1[i] = (0.5*r1-0.5*R1)*a[i] + (1-r1)*a[i+1] + ↩
            (0.5*r1+0.5*R1)*a[i+2] + dt*f[i];
        }

    /*update boundary coefficients for neumann bc*/

         c1[0] = -r1-R1;
         a1[nxx-1] = -r1+R1;

        /*Solving for tridiagonal matrix equation*/

      tridag(a1,b1,c1,d1,t1,nxx);

      /*Back substitution: updating concentration a*/

      for(i=0; i<nxx; i++)
          a[i+1] = t1[i];

      /*repeating the above procedure for concentration b*/

      for(i=0; i<nxx; i++)
        {
         a1[i] = -0.5*r3+0.5*R3;
         b1[i] = 1 + r3;
         c1[i] = -0.5*r3-0.5*R3;
         d1[i] = (0.5*r3-0.5*R3)*b[i] + (1-r3)*b[i+1] + ↩
            (0.5*r3+0.5*R3)*b[i+2] + dt*g[i];
        }

        c1[0] = -r3-R3;
        a1[nxx-1] = -r3+R3;
      tridag(a1,b1,c1,d1,t1,nxx);

      /*Back substitution: Updating concentrations b*/

      for(i=0; i<nxx; i++)
         b[i+1] = t1[i];

    /*End of first half time step*/

  /*Update coupling terms*/


      for(i=1; i<nxx+1; i++)
       {
         //q = s[i-1]*a[i]*a[i];
          f[i-1] = (rho*a[i]*a[i])/b[i] - ra*a[i] - ↩
               (Da*c*a[i])/(x*x) + (ba*e)/c - (ba*m)/(c*a[i]);
         g[i-1] = rho1*a[i]*a[i]*c*e - rb*b[i] - (Db*c*b[i])/(x*x) ↩
             + (rho1*a[i]*a[i]*c*m)/b[i];// + bb;
       }

  /*Update boundary values*/

  a[0] = a[2]; a[nxx+1] = a[nxx-1];
  b[0] = b[2]; b[nxx+1] = b[nxx-1];
  //neumannbc(a);
  //neumannbc(b);

  if(n%7000==0 && nxx < nx)
     nxx += 300;

  }

  fclose(fp);
  fclose(fpo);

  fpo=fopen("finala.dat","w");

     if(fpo==NULL)
          {
            puts("cannot open file");
            exit(1);
          }

  fp=fopen("finalb.dat","w");

     if(fp==NULL)
          {
            puts("cannot open file");
            exit(1);
          }

   fprintmatrix(a,fpo);
   fprintmatrix(b,fp);

   fclose(fpo);
   fclose(fp);

  fp=fopen("pattern_final.dat","w");

     if(fp==NULL)
          {
            puts("cannot open file");
            exit(1);
          }

   pattern(b,a,fp);

   fclose(fp);

  return 0;
}

void fprintmatrix (float p[nx+2], FILE *f)
```

```
{
 int i;
 float x;

     x = 1.0;
       for(i=1; i<nx+1; i++)
         {
            fprintf(f,"%d\t%f\t0\t0\n",(int) x,p[i]);
            x += dx;
         }
}

void pattern (float p[nx+2], float q[nx+2], FILE *f)
{
 int i;
 float x, y;

     x = 1.0;
       for(i=1; i<nx+1; i++)
         {
            y = p[i] - q[i];
            if(y > 0)
              fprintf(f,"%d\t%f\t0\t0\n",(int) x, y);
            else
              fprintf(f,"%d\t0.000000\t0\t0\n",(int) x);

            x += dx;
         }
}

void tridag(float a[], float b[], float c[], float r[], float u[], ↩
    int n)
{
  int j;

  float bet, gam[n];

//  gam=dvector(1,n);
  if (b[0] == 0.0) printf("Error 1 in tridag\n");
  u[0]=r[0]/(bet=b[0]);
  for (j=1;j<n;j++) {
    gam[j]=c[j-1]/bet;
    bet=b[j]-a[j]*gam[j];
    if (bet == 0.0) printf("Error 2 in tridag\n");
    u[j]=(r[j]-a[j]*u[j-1])/bet;

  }

  for (j=(n-2);j>=0;j--)
    u[j] -= gam[j+1]*u[j+1];
//  free_dvector(gam,1,n);
}

void neumannbc(float a[nx+2])
{

       a[0]=a[2];
       a[nx+1]=a[nx-1];
}
```