Investigation of Turing Patterns using Finite Element Method and Symmetry

Undergraduate Thesis

Submitted in fulfilment of the requirements for course BITS F421T
by

Pratik Aghor

Under the guidance of

Dr. Sai Jagan Mohan

Department of Mechanical Engineering
BITS Pilani
June 17, 2015

ABSTRACT

The motivation of this study is to investigate an interesting phenomenon of pattern formation in different reaction-diffusion systems. This work aims to exploit the **symmetry of the problem.** The transient equations are simulated using **Fractional Step Method.** Using the answer of the transient code as an initial guess to the **Newton-Raphson method**, steady state spatial patterns are obtained.
All the computations for transient and steady state were done in open source software FreeFem++.

Utilizing the symmetry of the mesh, a suitable change of basis is done. Applying **fundamental theorem of group representation theory,** the transformation on the Jacobian obtained from the Newton-Raphson iterations **block-diagonalizes** the Jacobian.

Different blocks of the block-diagonalized Jacobian correspond to different symmetries of the solution. Looking at the size of the blocks of the block-diagonal matrix, one can predict the symmetry of the solution.

I dedicate this thesis to all the values that should win in this world.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude towards my mentor Dr Sai Jagan Mohan for introducing me to the field of Fluid mechanics, Computational Fluid Dynamics and Group Theory. I would also like to thank Dr Amol Marathe and Dr Tapomoy Guha Sarkar for teaching me amazing things like non-linear vibrations and introducing me to the theory of relativity.

I would also like to thank Ruchir Dwivedi, without whose endless support and more importantly debugging, I couldn't have concluded this thesis. Thanks to Mohammad Atif, Aaditya Anand, Soham Kulkarni and Abhilash Sarawade who have been there all the time to support me. I also thank Pulkit Dubey for helping me with LaTeX. There are many friends of mine who kept my spirits up when nothing was working out. Akhilesh Bhagurkar and Gaurav Prabhudesai stand foremost in that list.

I am so grateful that I had such nice friends as the ones I mentioned in the last paragraph.

I thank my family for bolstering me and believing in me when I decided to follow my dreams.

And finally, I express my gratitude towards Amruta Agnihotri, an amazing girl who made me realize that life is beautiful.

There are many names deserving this token of gratitude.

For example, a pen-friend of mine, Kaveri. But I cannot mention all of them here.

I thank all those who have been helpful, directly or indirectly.

CONTENTS

LIST OF FIGURES

Part I

MOTIVATION AND INTRODUCTION

1

PATTERNS IN NATURE

"Nobody ever figures out what life is all about, and it doesn't matter. Explore the world. Nearly everything is really interesting if you go into it deeply enough." -Richard P. Feynman

1.1 ANIMAL COAT PATTERNS

Patterns can be found in nature everywhere. From the spots on the leopard to the shapes of snowflakes, from the ripples on water to the bands in desert, patterns surround human existence.

A small class of these patterns is animal coat patterns. How the zebra gets its stripes or how the leopard gets spots? These are some of the intriguing questions that we are trying to find answers to.

Following are some of the common occurrences of these patterns in nature.

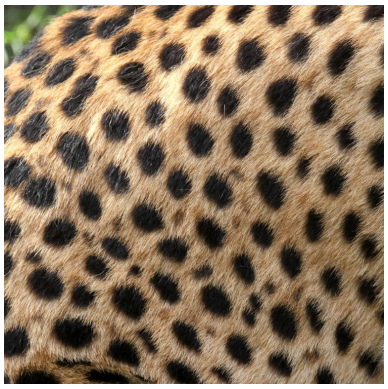


Figure 1: Cheetah

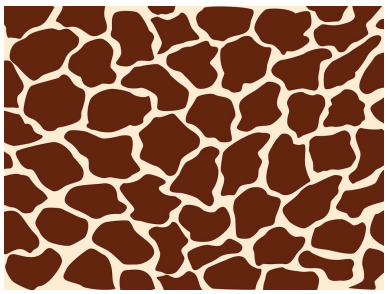


Figure 2: Giraffe

Figure 3: Zebra

Figure 4: Cub

Figure 5: Leopard

1.2 WHY STUDY REACTION-DIFFUSION

How patterns arise in nature is a challenging question that scientists are still trying to answer. Perhaps the most elegant mathematical model to describe pattern formation on animal coats was proposed by an English mathematician Alan Turing in 1952 [1]. Since then, many efforts have been made to describe the phenomenon of morphogenesis in nature.

Turing proposed the activator-inhibitor model which is explained in subsequent chapters. But soon, it became clear that reaction diffusion systems can be used to model a wide range of phenomena.

Propelled by only reaction and diffusion at our disposal, we can extract a great amount of information from many phenomena. With these seemingly innocuous systems, we can draw some powerful conclusions.

With only diffusion equation, one can model events such as spread of epidemics or rumours. Aided with reaction terms, one can model the birth/death rates of the individuals. One can also put forth a simple model for formation of galaxies, migration,population dynamics and perhaps extinction of some species.

Hence it is important to study reaction-diffusion systems.

1.3 EXPLOITATION OF SYMMETRY

It is seen that the patterns have some or other symmetry associated with them. Using this inherent symmetry, we can greatly simplify our calculations to solve the equations. Also, using symmetry, we can a priori predict some of the properties of the solution, without even solving the whole problem!

Part II

MATHEMATICAL BACKGROUND

2

GOVERNING EQUATIONS, BOUNDARY CONDITIONS AND LINEAR STABILITY

"Mathematics is only a means for expressing the laws that govern phenomena."
-Albert Einstein.

2.1 THE REACTION-DIFFUSION SYSTEM:

We start our analysis by the simple Schnakenberg system. The system is bimolecular and the non-dimensional governing equations governing the phenomenon are as follows:

$$\frac{\partial u}{\partial t} = \gamma f(u,v) + \nabla^2 u$$

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

The non-dimensional coefficient d contains information about the relative rates of diffusion of u and v.

We assume **homogeneous Neumann boundary conditions.** That is, there is no flux of u or v coming in or going out of the boundary of the domain. Therefore, the patterns are **self-organized**.

That is

$$\mathbf{n}.\nabla u = \mathbf{n}.\nabla v = 0$$

on the boundary.

2.2 ACTIVATOR-INHIBITOR MODEL:

Many systems in nature have inherent feedback mechanism. To explain pattern formation in bimolecular reaction-diffusion systems, the activator-inhibitor model is used.

Activator-inhibitor model contains two substances which interact with each other. More often than not, this interaction turns out to be non-linear. To understand activator-inhibitor mechanism, one should understand the term *'Autocatalysis'*.

Lets stick to the definition given in J.D. Murray's book.[2] ***'Autocatalysis'*** is the process whereby a chemical is involved in its own production.

To exemplify, lets take a simple reaction $A + X \underset{k_{-1}}{\overset{k}{\rightleftharpoons}} 2X$ Here the chemical X is involved in its own generation.
In activator-inhibitor mechanism, the activator is responsible for its own generation via *autocatalysis*. The activator also stimulates the generation of the inhibitor which in turn tries to supress activator's concentration. This way, a feedback mechanism is set up.

In reaction-diffusion systems, the reaction terms behave as depicted by activator-inhibitor mechanism. On top of that, both the activator and inhibitor also diffuse.

In order to generate patterns from diffusion-driven instability, it is essential to have activator's diffusion rate greater than that of inhibitor's.

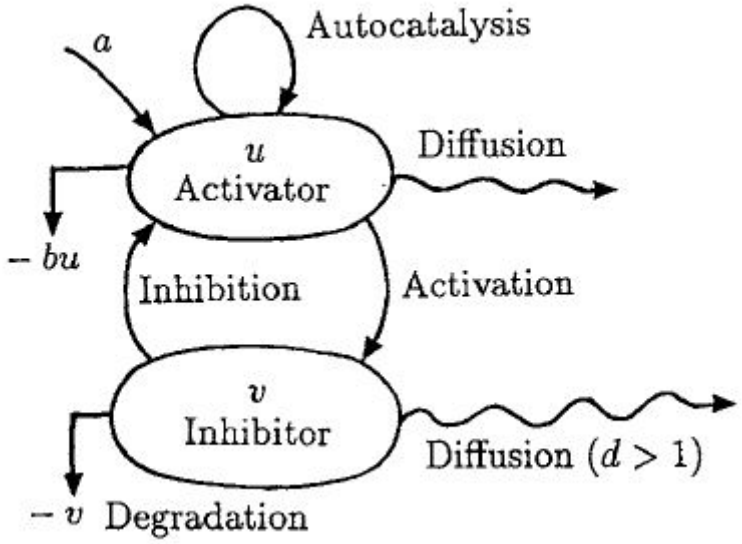


Figure 6: activator inhibitor mechanism

2.2.1 *Linear Stability Analysis*

Linear stability analysis is generally done to identify the bifurcation points and to set up the parameters to get different solutions.

The standard approach to linear stability analysis is as follows:
First find the steady state: For the reaction-diffusion systems, the relevant steady state can be found by simultaneously solving the equations

$f(u,v) = 0$ and $g(u,v) = 0$
The details of the derivation can be found in [2]

Let us denote the steady state solutions by u_0 and v_0 respectively.

As we are interested in ***diffusion driven instability***, we expect the steady state to be *spatially independent*. That is, the steady state should be linearly stable in absence of diffusion terms. With this condition imposed, the u and v satisfy

$$u_t = \gamma f(u,v)$$
$$v_t = \gamma g(u,v)$$

We set w=

$$\begin{pmatrix} u - u_0 \\ v - v_0 \end{pmatrix}$$

Assuming the perturbation to be small, i.e. $|w|$ to be small, we can write above equations as

$$w_t = \gamma A w$$

where A is the Stability matrix evaluated at $u = u_0$ and $v = v_0$.
A=

$$\begin{pmatrix} f_u & f_v \\ g_u & g_v \end{pmatrix}$$

To find the eigenvalues of A,
implies

$$|\gamma A - \lambda I| = \begin{vmatrix} f_u - \lambda & f_v \\ g_u & g_v - \lambda \end{vmatrix}.$$

=o

The characteristic polynomial then becomes,

$$\lambda^2 - \gamma(f_u + g_v)\lambda + \gamma^2(f_u g_v - f_v g_u) = 0$$

Hence,

$$\lambda_1, \lambda_2 = 1/2\gamma[(f_u + g_v) \pm (f_u + g_v - 4(f_u g_v - f_v g_u)^{1/2})]$$

Linear stability is guaranteed if $Re\lambda < 0$, that is,

$$trA = f_u + g_v < 0$$

$$|A| = f_u g_v - f_v g_u > 0$$

Now consider the full equation with diffusion terms included,

$$w_t = \gamma A w + D\nabla^2 w$$

with boundary conditions $(n.\nabla)w = 0$ on boundary ∂B where A is the Stability matrix evaluated at $u = u_0$ and $v = v_0$.
A=

$$\begin{pmatrix} f_u & f_v \\ g_u & g_v \end{pmatrix}$$

and D=

$$\begin{pmatrix} 1 & 0 \\ 0 & d \end{pmatrix}$$

To solve the above system of equations subjected to homogeneous Neumann boundary conditions, we first consider the $W(r)$ to be the **time-independent solution of spatial eigenvalue problem** defined by

$$\nabla^2 W + k^2 W = 0$$

with boundary conditions $(n.\nabla)w = 0$ on boundary ∂B

Using variable separable, we can now write the solution of the general problem to be

$$w(r,t) = \sum_k c_k e^{\lambda t} W(r)$$

c_k is determined by the **Fourier expansion of initial conditions** in terms of $W_k(r)$

Substituting $w(r,t) = \sum_k c_k e^{\lambda t} W(r)$ into $w_t = \gamma A w + D\nabla^2 w$ we get another eigenvalue problem where λ is the eigenvalue that controls **temporal growth.**

The $e^{\lambda}t$ factor cancels out and the temporal eigenvalue problem looks like

$$\lambda W_k = \gamma A W_k + D\nabla^2 W_k$$

using the time-independent spacial eigenvalue problem, the above equation becomes, for each k

$$\lambda W_k = \gamma A W_k - Dk^2 W_k$$

The characteristic polynomial becomes,

$$|\lambda I - \gamma A + Dk^2| = 0$$

that is,

$$\lambda^2 + \lambda[k^2(1+d) - \gamma(f_u + g_v] + h(k^2) = 0$$

where

$$h(k^2) = dk^4 - \gamma(df_u + g_v)k^2 + \gamma^2|A|$$

For linear stability, we want both the solutions of the above characteristic polynomial to have $Re\lambda < 0$. The condition for steady state to be stale in absence of any spacial effects gives us the conditions already derived above. That is, $Re\lambda(k = 0) < 0$ implies

$$trA = f_u + g_v < 0$$

$$|A| = f_u g_v - f_v g_u > 0$$

For steady state to be unstable to spacial perturbation, we need, $Re\lambda(k) > 0$ for $k \neq 0$

To satisfy this condition, we get another constraint. This can happen in two cases, either coefficient of λin characteristic polynomial is < 0 or if $h(k^2) < 0$ for some $k \neq 0$.

As $f_u + g_v < 0$ and $k^2(1+d) > 0$ this implies
$[k^2(1+d) - \gamma(f_u + g_v)] > 0$ for some $k \neq 0$.

Hence the only way for steady state to be unstable to spacial disturbances is

$$h(k^2) < 0.$$

Characteristic polynomial is quadratic in λ. Solving for λ we get,

$$2\lambda = -[k^2(1+d) - (f_u + g_v)] \pm [(k^2(1+d) - (f_u + g_v))^2 - 4h(k^2)]^{\frac{1}{2}}$$

From the previous two constraints that we already got, we know that $|A| > 0$. From the expression for $h(K^2)$, the only possibility for it to be negative is if $(df_u + g_v) > 0$.

But we know that $f_u + g_v < 0$. Therefore it follows that $d \neq 1$ and f_u and g_v must have opposite signs.

Therefore in addition to two constraints, we have one more condition. That is $(df_u + g_v) > 0$
$\Rightarrow d \neq 1$. and f_u and g_v must have opposite signs.

These conditions are necessary but **not sufficient** to ensure that $Re\lambda(k) > 0$.

$h(k^2) < 0 \Rightarrow h_{min} < 0$
$\Rightarrow \frac{dh}{d(k^2)} = 0$
$\Rightarrow h_{min} = \gamma^2\left(|A| - \frac{(df_u+g_v)^2}{4d}\right)$ and $k^2 = k_m^2 = \gamma\frac{(df_u+g_v)}{2d}$
$h_{min} < 0 \Rightarrow \frac{(df_u+g_v)^2}{4d} > |A|$

When $d = d_c(> 1)$ where d_c is critical diffusion coefficient, $k_m = k_c$. k_c is called critical wavenumber. At the critical wavenumber, $h_{min} = 0$

$k_c^2 = \gamma\frac{(df_u+g_v)}{2d} = \gamma\left(\frac{|A|}{d_c}\right)^{\frac{1}{2}}$

To get the range of spatially unstable wavenumbers, it can be easily shown that
$(df_u + g_v - 4d(f_u g_v - f_v g_u)) > 0$

The conditions that we derived for a reaction-diffusion system to be unstable to spacial perturbations are summarized as follows:

$$f_u + g_v < 0$$

$$f_u g_v - f_v g_u > 0$$

$$(df_u + g_v) > 0$$
$$(\Rightarrow d > 1)$$

and

$$(df_u + g_v - 4d(f_u g_v - f_v g_u)) > 0$$

We apply these conditions to determine the parameter space.

3

TRANSIENT CODE:FRACTIONAL STEP METHODS

"It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts."
-Sir Arthur Conan Doyle

3.1 INTRODUCTION TO FRACTIONAL STEP METHODS

Fractional step methods are basically operator splitting methods. These methods are useful when the physics of the problem is a combination of different physical phenomena, e.g., convection, reaction, diffusion, etc.

By applying fractional steps, we split the given system of PDE's into **well posed PDE's**. These well posed PDE's can be solved sequentially and later amalgamated in appropriate manner to generate the solution of the original system of equations.

Computationally, fractional step algorithm operates on component codes sequentially. After each step, the solution is updated and at the end of the code, we get the final solution.

3.2 AN EXAMPLE:

Consider a simple linear equation.

$$\frac{du}{dt} = Au + Bu$$

$$u(0) = u^0$$

Where A and B are square matrices and u is a vector.

The analytical solution of the above system with the given initial condition will be

$$u(t) = \exp((A+B)t)u^0$$

Now let us proceed sequentially. We use two steps here. First step is:

$$\frac{du}{dt} = Au$$

$$u(0) = u^0$$

The analytical solution of the above system is

$$u(t) = \exp((A)t)u^0$$

The second step would be to solve

$$\frac{d\hat{u}}{dt} = B\hat{u}$$

$$\hat{u}(0) = \hat{u^0} = \exp((A)t)u^0$$

Hence we say that the approximate solution is

$$\hat{u}(0) = \exp(Bt)\exp(At)u^0$$

It should be noted that the solution that we just obtained by sequentially solving two differential equations is, in general different from that we obtained analytically by solving the original equation. This difference is also expected as A and B are not numbers, but matrices. At some instant $t = \triangle t$, using Taylor expansion, we get

$$\begin{aligned} u(\triangle t) &= \exp((A+B)\triangle t)u^0 \\ &= \left(I + \triangle t(A+B) + \frac{(\triangle t)^2}{2}(A+B)^2\right)u^0 + o\big((\triangle t)^2\big) \\ &= \left(I + \triangle t(A+B) + \frac{(\triangle t)^2}{2}(A^2+AB+BA+B^2)\right)u^0 + o\big((\triangle t)^2\big) \end{aligned}$$

While the solution obtained in two steps will be

$$\begin{aligned} u(\triangle t) &= \exp(B\triangle t)\exp(A\triangle t)u^0 \\ &= \left(I + \triangle t(B) + \frac{(\triangle t)^2}{2}(B)^2\right)\left(I + \triangle t(A) + \frac{(\triangle t)^2}{2}(A)^2\right)u^0 + o\big((\triangle t)^2\big) \\ &= \left(I + \triangle t(A+B) + \frac{(\triangle t)^2}{2}(A^2+2BA+B^2)\right)u^0 + o\big((\triangle t)^2\big) \end{aligned}$$

In general, A and B don't commute with each other. That is,
$AB \neq BA$

In some specific cases when A and B commute, then the later solution is approximation of the former and it is second order accurate in time.

3.3 SECOND-ORDER STRANG SYMMETRIC SPLITTING

To solve the reaction-diffusion equations, we performed transient simulations. These transient simulations involve strang-splitting which is second order accurate in time.

3.3.1 *Proof of how Strang splitting gives second order accuracy in time*

Objetive To prove that three step symmetric strang-splitting provides second order accurate solution.

Proof:

$$\frac{du}{dt} = Au$$
$$u(0) = u^0$$

Let us denote the solution of above equation at time $\triangle t$ by $P_A^{\triangle t}$
and

$$\frac{du}{dt} = Bu$$
$$u(0) = u^0$$

Let us denote the solution of above equation at time $\triangle t$ by $P_B^{\triangle t}$
The claim is: Solution of the equation

$$\frac{du}{dt} = Au + Bu$$

$$u(0) = u^0$$

at time $\triangle t$ is approximated to the second order by

$$\hat{u}(\triangle t) = P_A^{\frac{\triangle t}{2}} P_B^{\triangle t} P_A^{\frac{\triangle t}{2}}$$

$$\begin{aligned}
\hat{u}(\triangle t) &= \exp(A\frac{\triangle t}{2}) \exp(B\triangle t) \exp(A\frac{\triangle t}{2}) \\
&= \left(I + (\frac{\triangle t}{2})A + (\frac{(\triangle t)^2}{8})A\right)\left(I + \triangle tB + \frac{(\triangle t)^2}{2}B^2\right)\left(I + (\frac{\triangle t}{2})A + (\frac{(\triangle t)^2}{8})A\right) \\
&= \left(I + \triangle tA + \triangle tB + (\frac{\triangle t}{2})(A^2 + AB + BA + B^2)\right)u^0 + o((\triangle t)^2)
\end{aligned}$$

QED.
More details of fractional step methods can be found in [3]

3.4 ALGORITHM FOR REACTION-DIFFUSION SYSTEMS

The fractional step algorithm that is used in the transient code can be summarized as follows. The reaction diffusion system for which the algorithm is given is as follows.

$$\frac{\partial u}{\partial t} = \gamma f(u, v) + \nabla^2 u$$

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

boundary conditions are homogeneous Neumann boundary conditions.

$$\mathbf{n}.\nabla u = \mathbf{n}.\nabla v = 0$$

on the ∂B

(NOTE: All boundary conditions are homogeneous Neumann boundary conditions on u and v)
Initialize
uold,vold;
Step 1
Solve reaction system (Use fourth order Runge- Kutta for time marching.)

$$\frac{du}{dt} = \gamma f(u, v)$$

$$\frac{dv}{dt} = 0$$

Solve this at time $\frac{\triangle t}{2}$.

$$\frac{du}{dt} = 0$$

$$\frac{dv}{dt} = \gamma g(u, v)$$

Solve this at time $\triangle t$

$$\frac{du}{dt} = \gamma f(u,v)$$
$$\frac{dv}{dt} = 0$$

Solve this at time $\frac{\triangle t}{2}$.

uold=u; vold=v;

Step 2

Solve Diffusion equations at time $\triangle t$

$$\frac{\partial u}{\partial t} = \nabla^2 u$$
$$\frac{\partial v}{\partial t} = d\nabla^2 v$$

(Use FEM for solving).

update uold and vold;

Step 3

Solve reaction system (Use fourth order Runge- Kutta for time marching.)

$$\frac{du}{dt} = \gamma f(u,v)$$
$$\frac{dv}{dt} = 0$$

Solve this at time $\frac{\triangle t}{2}$.

$$\frac{du}{dt} = 0$$
$$\frac{dv}{dt} = \gamma g(u,v)$$

Solve this at time $\triangle t$

$$\frac{du}{dt} = \gamma f(u,v)$$
$$\frac{dv}{dt} = 0$$

Solve this at time $\frac{\triangle t}{2}$.

Update uold,vold;

u=uold;v=vold;

4

LINEARIZATION AND THE NEWTON-RAPHSON METHOD

"What goes up must come down."
-Issac Newton

We do linearization of the steady-state weak form in order to apply Newton-Raphson method.We use this concept of Fréchet derivative to linearize the weak form for Newton-Raphson method.

4.1 THE STEADY STATE GOVERNING EQUATIONS

The steady state governing equations for the bimolecular reaction diffusion equations is given as follows:

$$\gamma f(u,v) + \nabla^2 u = 0$$

$$\gamma g(u,v) + d\nabla^2 v = 0$$

The boundary conditions are still the same, that is, homogeneous Neumann boundary conditions.

$$n.\nabla u = n.\nabla v = 0$$

4.2 THE FRÉCHET DERIVATIVE

We use Hilbert space as our function space. The Fréchet derivative is simply a general form of the derivative of a real valued or vector valued function defined on real space. To introduce the idea of the Freécht derivative, let us consider that the following expansion holds for some function **f** as $h \to 0$.

$$f(x_0 + h) = f(x_0) + Ah + O(h^2)$$

Then the Fréchet derivative of function **f** at $x = x_0$ is given by

$$Df(x_0) = A$$

Note that x_0 belongs to a Hilbert space.

The Fréchet derivative has powerful implications. One of the conclusions that one can derive from the above background is as follows: **Taking a derivative is equivalent to linearizing the operator.**

Although to find the actual linearization, we conveniently go via **Calculus of Variations**, we have collected an important piece of information from the knowledge of Fréchet derivative. The fact that the differentiation is analogous to linearization, or vice verse will be cleverly exploited in the application of **Group Representation Theory.**

4.3 THE NEWTON-RAPHSON METHOD

Newton-Raphson method is employed to solve the system of non-linear equations. We give an initial guess and with the help of the derivative of the linearized weak form, we find the next guess. We do this process iteratively in order to arrive at solution.

Let X_n be the vector that stores the values of the solution. The subscript n denotes the n^{th} guess. To find the next guess X_{n+1} we use the Freéchet derivative.

To simplify, consider a one dimensional real valued, non-linear function $f(x)$. The task is to find its zeros.

Let

$$x_n - x_{n+1} = -\delta x$$

Expanding $f(x_{n+1}) = f(x_n + \delta x)$ using Taylor expansion about $x = x_n$ and ignoring higher order terms, we get the following.

$$f(x_{n+1}) = f(x_n + \delta x) = f(x_n) + (\delta x) f'(x_n)$$

But as we are interested in the finding the zeros of f, we put $f(x_{n+1}) = 0$. $\Rightarrow$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Similarly, in case of vector X_n, we can say when

$$X_n - X_{n+1} = -\delta X$$

$$X_{n+1} = X_n - [DF(X_n)]^{-1} F(X_n)$$

Where $DF(X_n)$ is the **Fréchet derivative** of the linearized weak form F, evaluated at $X = X_n$.
$DF(X_n)$ is also called the **Jacobian** of F.
Rearranging the above equation, we get

$$[DF(X_n)]\delta X_n = F(X_n)$$

The quantity $[DF(X_n)]\delta X_n$ is known as **directional derivative** of F in the direction of δX.

It can be proven that for a function of more than one variables, as is the case with nonlinear reaction terms $f(u, v)$ and $g(u, v)$, the **directional derivative**

$$Dfdu = f_u du + f_v dv$$

NOTE:
There is one danger in using Newton-Raphson method. If we provide the wrong initial guess, instead of going towards the solution, it may diverge. Hence the initial condition plays crucial role in determining whether the Newton-Raphson code will converge or diverge.
To take care of this difficulty, we run the transient code with fractional step method for long time and once the transient solution becomes steady, we feed in that solution as the initial guess to Newton-Raphson. We expect the Newton-Raphson to converge to the same solution.

5

DERIVATION OF THE DIRECTIONAL DERIVATIVE FOR SOME REACTION-DIFFUSION SYSTEMS

The details of this chapter can be found in [4]

5.1 UNIMOLECULAR EQUATIONS

5.1.1 *The Allen-Cahn Equation*

Allen-Cahn equation is a unimolecular reaction diffusion equation. The steady state Allen-Cahn cubic quintic euation is given as follows.

$$-\mu\triangle u - \lambda u - u^3 + u^5 = 0$$

$$u = 0$$

on ∂B

Here

$$f(u) = -\lambda u - u^3 + u^5$$

$\Rightarrow$

$$f_u = -\lambda - 3u^2 + 5u^4$$

5.1.2 *The Bratu's Problem*

Bratu's problem is again a scalar nonlinear problem. The steady state governing equation is given as:

$$f(u) = -10(u - \lambda \exp u)$$

Hence,

$$f_u = -10(1 - \lambda \exp u)$$

5.2 GENERAL BIMOLECULAR REACTION-DIFFUSION SYSTEM WITH CONSTANT DIFFUSION COEFFICIENT

The steady state equations are

$$\gamma f(u,v) + \nabla^2 u = 0$$

$$\gamma g(u,v) + d\nabla^2 v = 0$$

Let uh and vh be the test functions corresponding to u and v. The weak form will look as follows: The boundary conditions are homogeneous Neumann boundary conditions.

$$F = \int_\Omega (\gamma f(u,v)uh) + \int_\Omega (\gamma g(u,v)vh) - \int_\Omega \left((\tfrac{\partial u}{\partial x})(\tfrac{\partial u_h}{\partial x}) + (\tfrac{\partial u}{\partial y})(\tfrac{\partial u_h}{\partial y}) \right) - d\int_\Omega \left((\frac{\partial v}{\partial x})(\frac{\partial v_h}{\partial x}) + (\frac{\partial v}{\partial y})(\frac{\partial v_h}{\partial y}) \right) \quad (1)$$

To find the directional derivative of the above weak form, we use a trick similar to that used in calculus of variations. Replace u and v by $(u+\epsilon\delta u)$ and $(v+\epsilon\delta v)$ in the above weak form. Differentiate with respect to *epsilon* and then put $\epsilon = 0$. Hence to find $DFdu$ we do the following.

$$\int_\Omega \left(\gamma f((u+\epsilon\delta u),(v+\epsilon\delta v))uh\right) + \int_\Omega \left(\gamma g((u+\epsilon\delta u),(v+\epsilon\delta v))vh\right) - \int_\Omega \left((\frac{\partial(u+\epsilon\delta u)}{\partial x})(\frac{\partial u_h}{\partial x}) + (\frac{\partial(u+\epsilon\delta u)}{\partial y})(\frac{\partial u_h}{\partial y}) \right) - d\int_\Omega \left((\frac{\partial(v+\epsilon\delta v)}{\partial x})(\frac{\partial v_h}{\partial x}) + (\frac{\partial(v+\epsilon\delta v)}{\partial y})(\frac{\partial v_h}{\partial y}) \right) \quad (2)$$

Now, differentiating the above expression with respect to epsilon and putting $\epsilon = 0$, we get

$$DFdu = \int_\Omega \left(\gamma(f_u(u,v)\delta u + f_v(u,v)\delta v)uh\right) + \int_\Omega \left(\gamma(g_u(u,v)\delta u + g_v(u,v)\delta v)vh\right) - \int_\Omega \left((\frac{\partial\delta u}{\partial x})(\frac{\partial u_h}{\partial x}) + (\frac{\partial\delta u}{\partial y})(\frac{\partial u_h}{\partial y}) \right) - d\int_\Omega \left((\frac{\partial\delta v}{\partial x})(\frac{\partial v_h}{\partial x}) + (\frac{\partial\delta v}{\partial y})(\frac{\partial v_h}{\partial y}) \right) \quad (3)$$

where

$$f_u(u,v) = \frac{\partial f(u,v)}{\partial u}$$

and so on.

5.3 THE SCHNAKENBERG SYSTEM

For the Schnakenberg System,

$$f(u,v) = (-u + u^2 v)$$
$$g(u,v) = (\lambda - u^2 v)$$

$\Rightarrow$

$$f_u = (-1 + 2uv)$$
$$f_v = (u^2)$$
$$g_u = -2uv$$
$$g_v = -u^2$$

5.4 GRAY-SCOTT MODEL

$$\frac{\partial u}{\partial t} = f(u,v) + ru\nabla^2 u$$
$$\frac{\partial v}{\partial t} = g(u,v) + rv\nabla^2 v$$
$$f(u,v) = -u^2 v + f(1-u)$$
$$g(u,v) = u^2 v - (f+k)v$$

$\Rightarrow$

$$f_u = (-2uv - f)$$
$$f_v = (-u^2)$$
$$g_u = 2uv$$
$$g_v = u^2 - (f+k)$$

We use these functions while calculating the directional derivatives of the corresponding systems. The Newton-Raphson method can then give us the zero of the nonlinear system.

6

GROUP THEORETIC APPROACH

"The scientist does not study nature because it is useful to do so. He studies it because he takes pleasure in it, and he takes pleasure in it because it is beautiful. If nature were not beautiful it would not be worth knowing, and life would not be worth living."
-Henri Poincaré

The details of this chapter can be found in [5],[6],[7],[8],[9] and [10]. Pattern formation is inherently related to the symmetry of the problem. While marching towards the solution, when the Newton-Raphson encounters change in stability or a critical point, the Jacobian matrix becomes singular.

To deal with this problem we use **Group Representation Theory** and **Theory of block diagonalization.** This way, we draw on symmetry of the problem. We can also make some apriori comments on the symmetry of the solutions.

We will limit our investigation to finite subgroups of planar orthogonal group ($O(2)$). That are

- The subgroup of rotations (C_n)
- The dihedral subgroups (D_n)

In particular, for the reaction diffusion systems, we are interested in dihedral groups, as the solutions possess D_n symmetries.

6.1 GROUP REPRESENTATIONS:

Loosely speaking, a group representation is a set of **invertible** matrices that mimic the group behaviour.

To exemplify, consider the group D_4, the symmetry of a square. The elements of this group are given as follows:

$$D_4 = \{e, r, r^2, r^3, s, sr, sr^2, sr^3\}$$

Where $r \Rightarrow$ Rotation by $\pi/2$ radian. amd $s \Rightarrow$ Reflection about origin by π radian.

Let us say I have two matrices, call them R and S such that R is a rotation by $\pi/2$ radian and S is a reflection about origine through π radian.

And if matrix multiplications $\{R^2, R^3, SR, SR^2, SR^3\}$ give me the symmetry of as expected by their corresponding counterparts in the elements of D_4, then the set

$$T = \{I, R, R^2, R^3, S, SR, SR^2, SR^3\}$$

is called the group representation corresponding to D_4 Where I is the identity matrix.

6.2 SOME DEFINITIONS FROM GROUP THEORY AND SCHUR'S LEMMA

Let G be a finite group whose order is denoted by $|G|$
V be a finite dimensional vector space.
$GL(V)$ be the group of all nonsingular linear transformations of V onto itself.

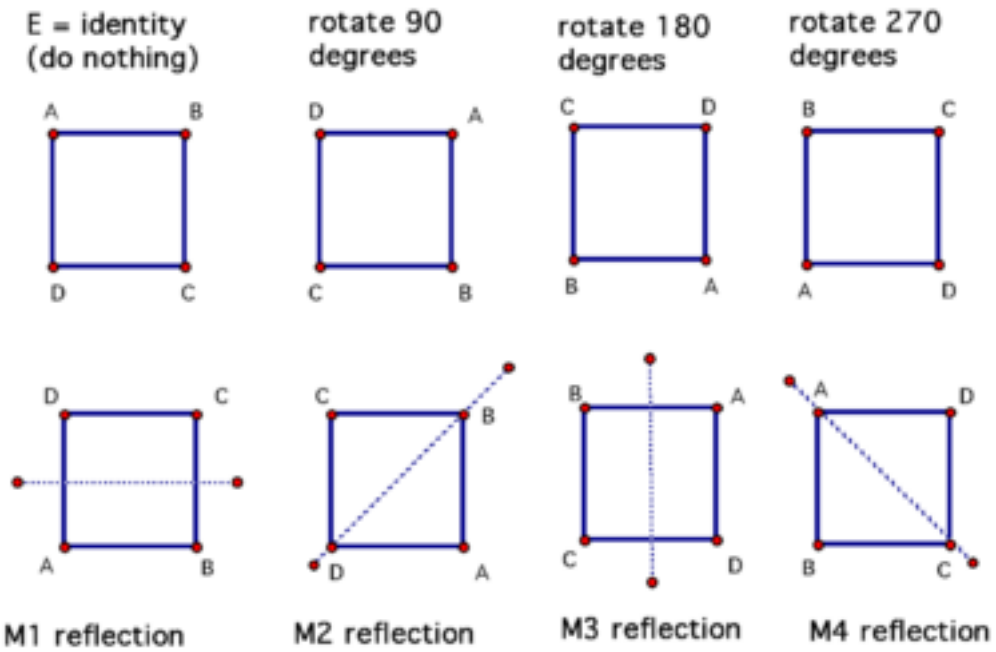


Figure 7: Elements of Dihedral group D_4

6.2.1 *Homomorphism*

A representation G on V is a *homomorphism* $H : G \to GL(V)$, if $H(gh) = H(g)H(h)\ \forall g, h \epsilon G$

6.2.2 *Equivalent Representations*

Two group representations of G, H_1 and H_2 are equivalent if there exists a nonsingular matrix B, such that $H_1(g) = B^{-1}H_2(g)B\ \forall g \epsilon G$.

6.2.3 *Irreducible Representations*

A representation H on G is said to be irreducible if there exists no nontrivial subspace W. In other words, W can only be 0 or V.

6.2.4 *An important result*

Let us state an important result without proof. Every group representation of D_n of dimension greater than or equal to 3 can be decomposed into parts containing one dimensional and two dimensional irreducible representations.

6.2.5 *Schur's Lemma*

Armed with these definitions, we state Schur's lemma without proof. R is field of Real numbers and C is field of complex numbers.

Let $V = RorC$ and A be a (possibly rectangular) matrix over V.Assume that T_1 and T_2 are irreducible matrix representations of group G over V, and

$$T_1(g)A = AT_2(g)$$

$\forall g \epsilon G$

1. $A = O$ or else A is square and nonsingular

2. If T_1 and T_2 are inequivalent, then $A = O$.
3. If $T_1(g) = T_2(g)$ for all $g \epsilon G$ and T_1 is irreducible, then $A = \lambda I$ for some $\lambda \epsilon \mathrm{V}$

6.3 EQUIVARIANCE OF THE GOVERNING EQUATIONS

Many governing equations have inherent symmetries associated with them. These association of symmetries with the governing equation is manifested into the equivariance.

Let us say, that I have a matrix R which is a member of some group representation T, where T corresponds to the dihedral group D_n.

Also, let us represent the governing equation as

$$f(\mu, \mathbf{x}) = 0$$
$$f : (R \times V \to V)$$

Where μ is some parameter.

We say that the governing equation $f(\mu, \mathbf{x})$ is **equivariant** under D_n symmetry if

$$f(\mu, R\mathbf{x}) = Rf(\mu, \mathbf{x})$$

6.4 THE FUNDAMENTAL THEOREM OF GROUP REPRESENTATION

The fundamental theorem of group representation states

While solving a linear system $A\mathbf{x} = B$

if the matrix A commutes with **every** member of a group representation T then there exists an orthogonal basis R such that $R^{-1}AR$ is block diagonal.

6.5 PROOF FOR JACOBIAN OF THE NEWTON-RAPHSON METHOD

$$\begin{aligned} \mathbf{f}(\lambda, \mathbf{u}) &= \mathbf{0}, \\ \mathbf{f} : \mathbb{R} \times V &\mapsto V \end{aligned}$$

Often times symmetry in the problem would imply that these equations are equivariant under the action of a Group D_n (say eg, a dihedral group),

$$\mathbf{f}(\lambda, R(D_n)\mathbf{u}) = R(D_n)\mathbf{f}(\lambda, \mathbf{u})$$

Differentiating both sides gives us

$$Df(\lambda, R(D_n)\mathbf{u})R(D_n) = R(D_n)Df(\lambda, \mathbf{u}) \tag{4}$$

The three things to note now are

1. *Looking for symmetric solutions (solutions that possess the symmetry of the dihedral group) means* $R(D_n)\mathbf{u} = \mathbf{u}$ *i.e, the solution is* invariant *to the action of the group. We substitute this into eq(4) to note that*

$$Df(\lambda, \mathbf{u})R(D_n) = R(D_n)Df(\lambda, \mathbf{u})$$

2. *Thus, we conclude from the above equation that along the symmetric solutions path, the Jacobian (a linear operator) commutes with the actions of the group.*
3. **The fundamental theorem of Group Theoretic application** *asserts that linear operators which commute with the group can be block diagonalized in a suitable "symmetry adapted basis". This proof is constructive. We can use this to construct the symmetry adapted basis.*

Part III

RESULTS AND DISCUSSION

7

SOME RESULTS

"It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong." -Richard P. Feynman

7.1 THE SCHNAKENBERG SYSTEM

7.1.1 *Results for $\lambda = 1$ and $\lambda = 2.85$*

- Domain: $[\frac{-L_x}{2}, \frac{L_x}{2}] \times [\frac{-L_y}{2}, \frac{L_y}{2}]$ where $L_x = 39.9$ and $L_y = 19.94$
- Homogeneous Neumann Boundary Conditions

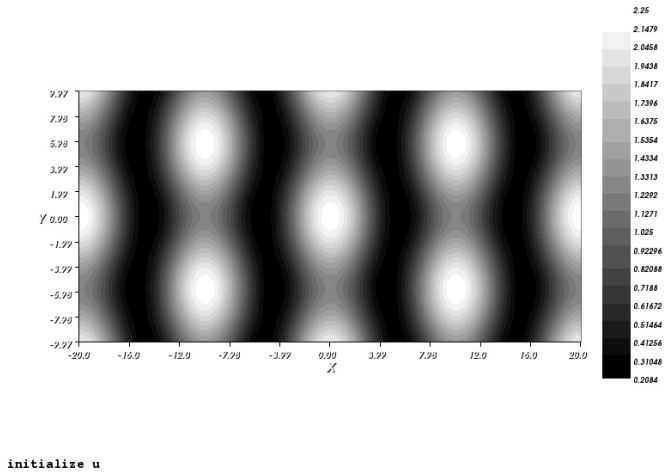


Figure 8: Initialize u $\lambda = 1 and 2.85$

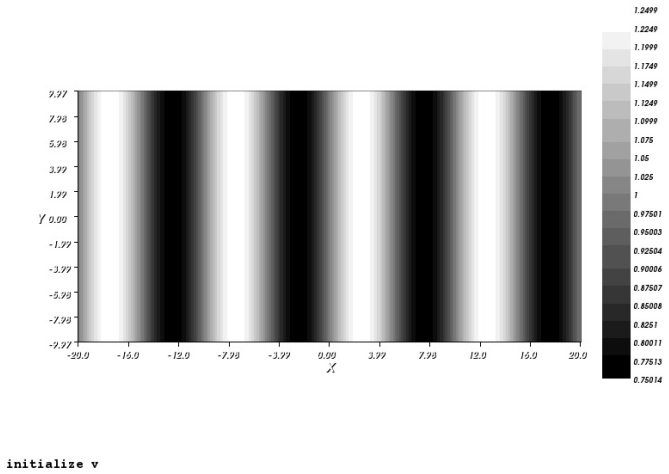
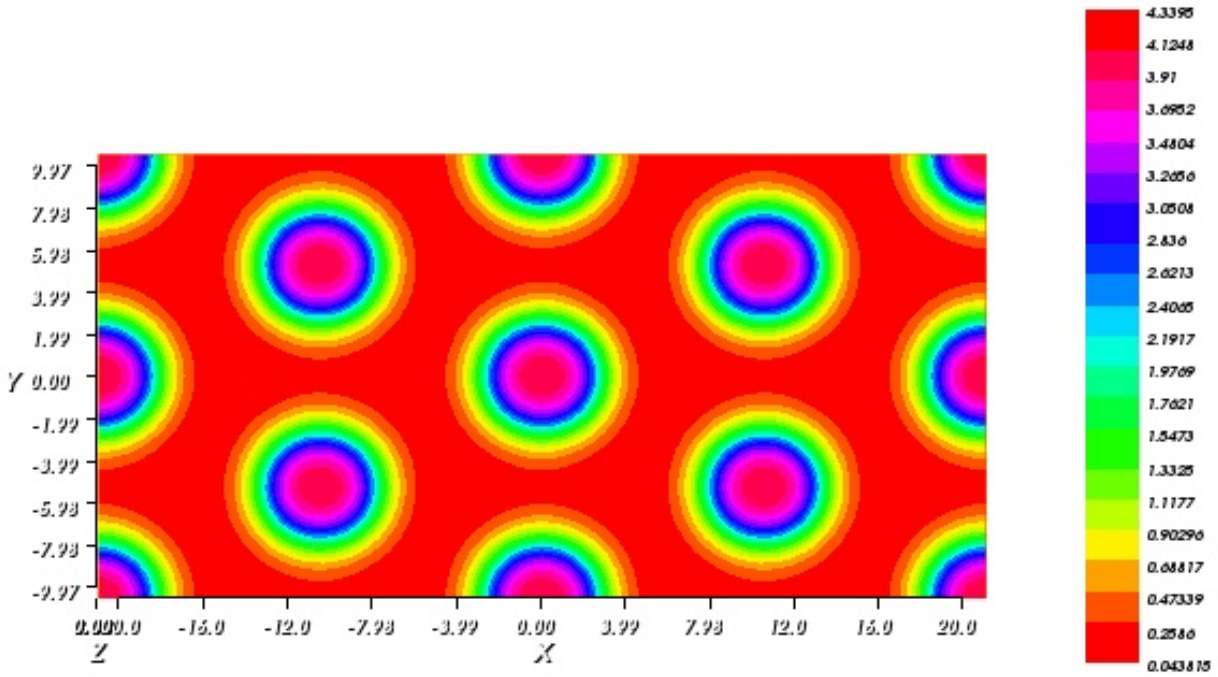


Figure 9: Initialize u $\lambda = 1 and 2.85$

7.1.2 *Some results for block-diagonalization of Jacobian matrix*



u at t=0

Figure 10: Transient end Schnakenberg u at $\lambda = 1$

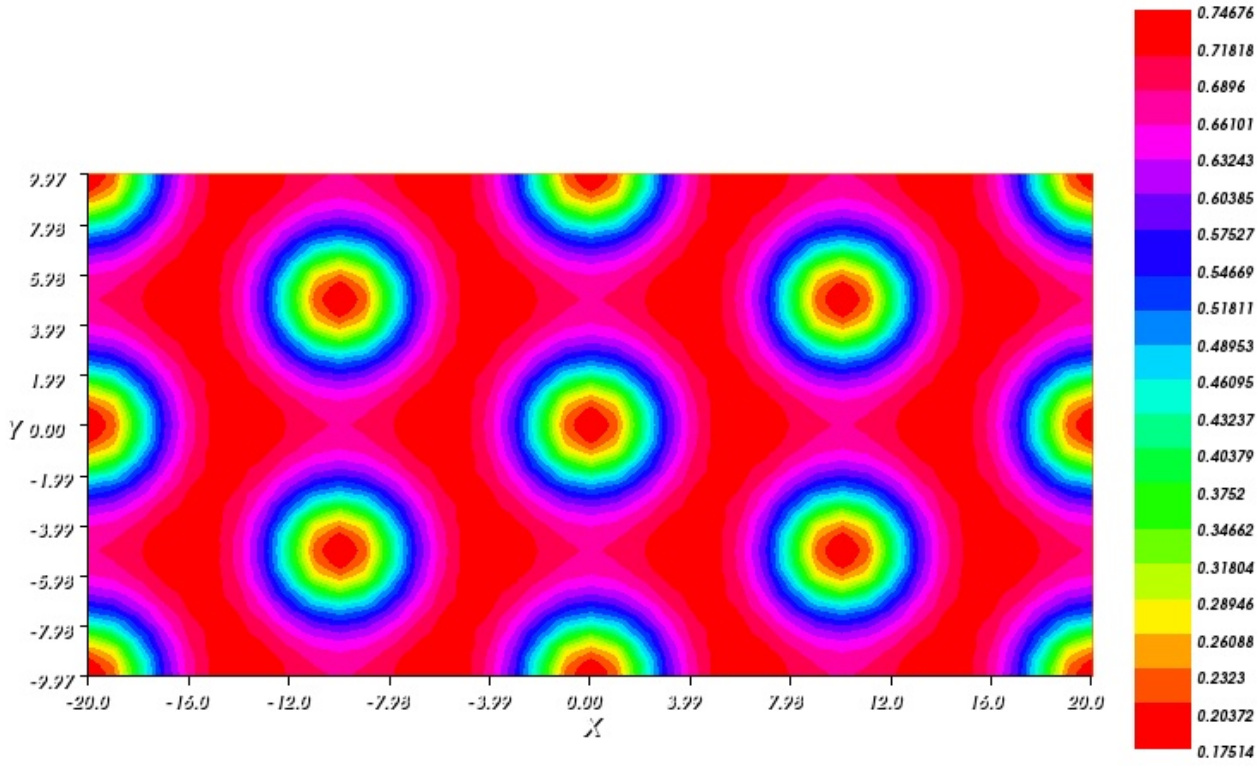


Figure 11: Transient end Schnakenberg v at $\lambda = 1$

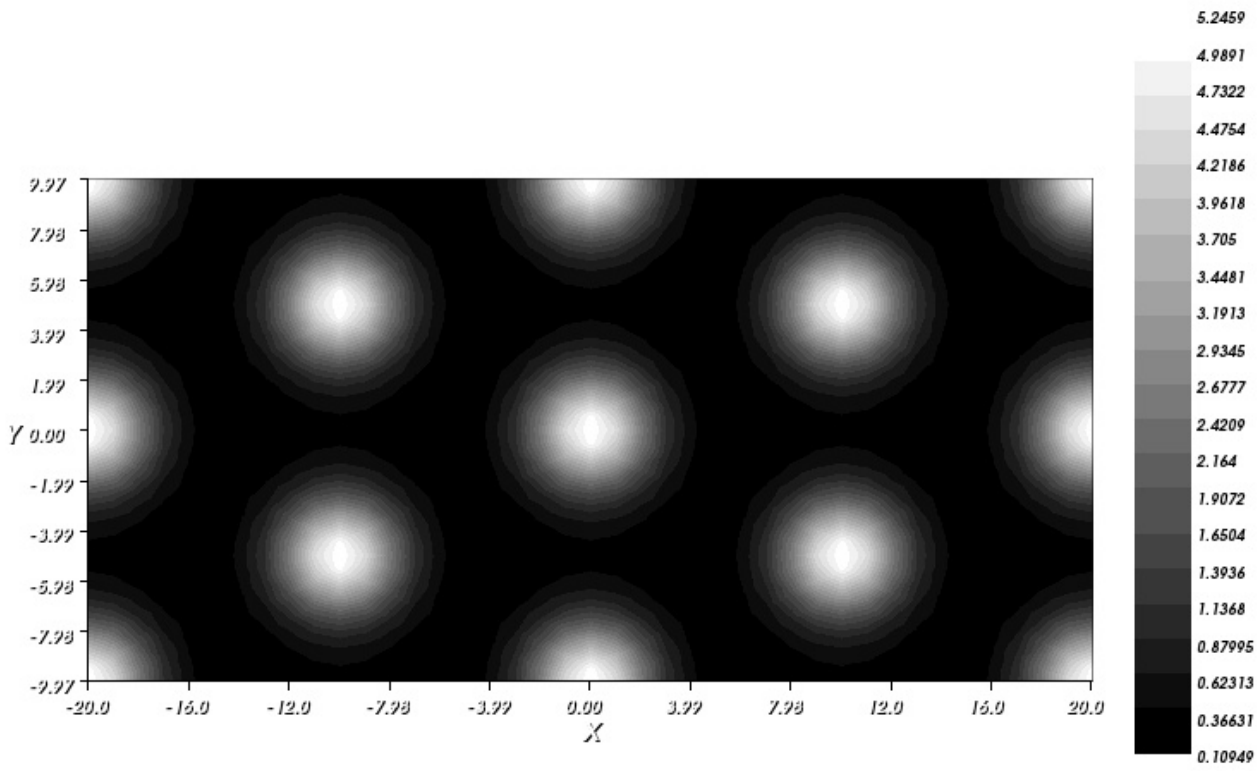


Figure 12: Converged solution u by Newton-Raphson for Schnakenberg at $\lambda = 1$

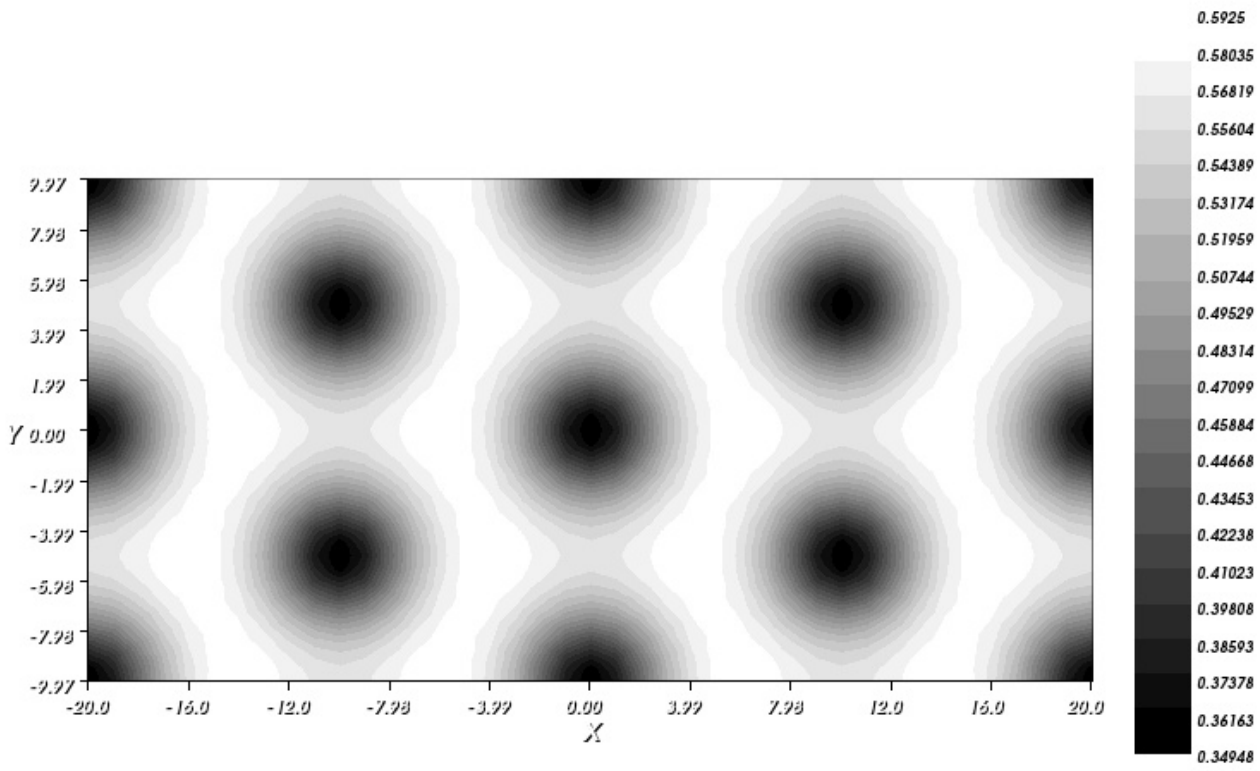


Figure 13: Converged solution v by Newton-Raphson for Schnakenberg at $\lambda = 1$

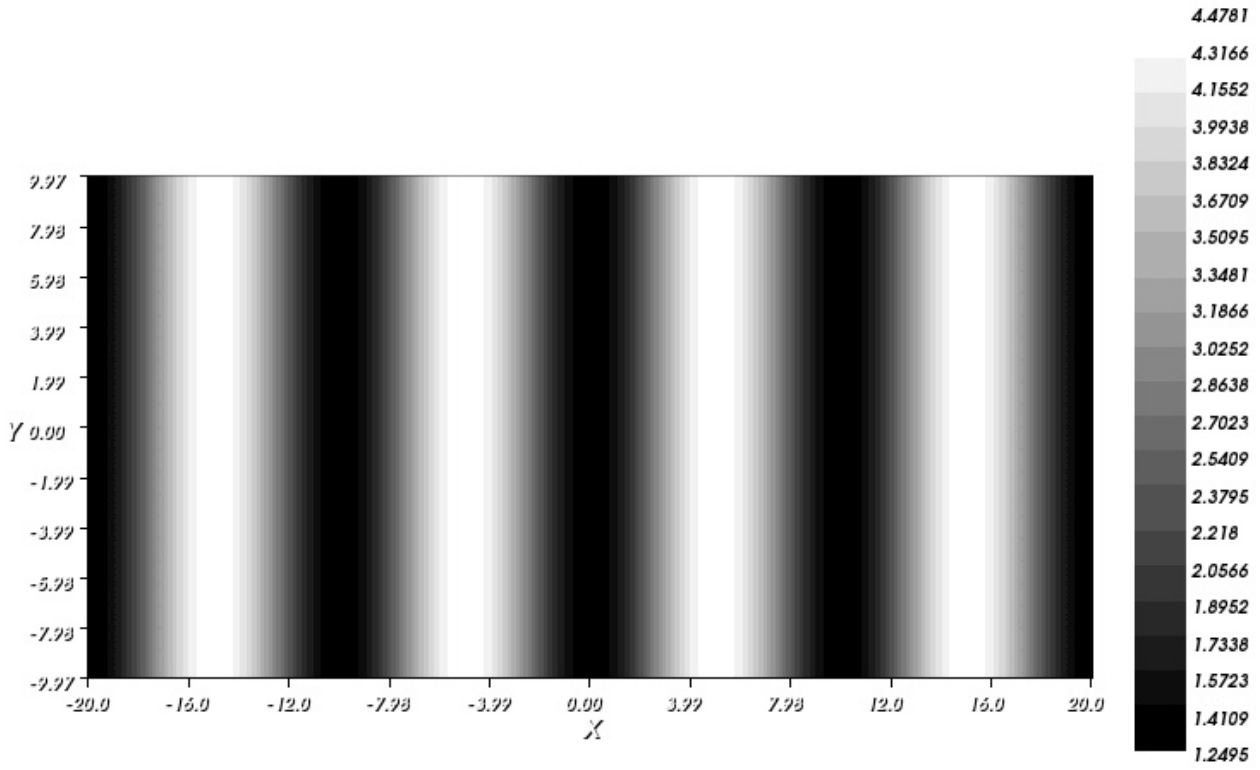


Figure 14: Converged solution u by Newton-Raphson for Schnakenberg at $\lambda = 2.85$

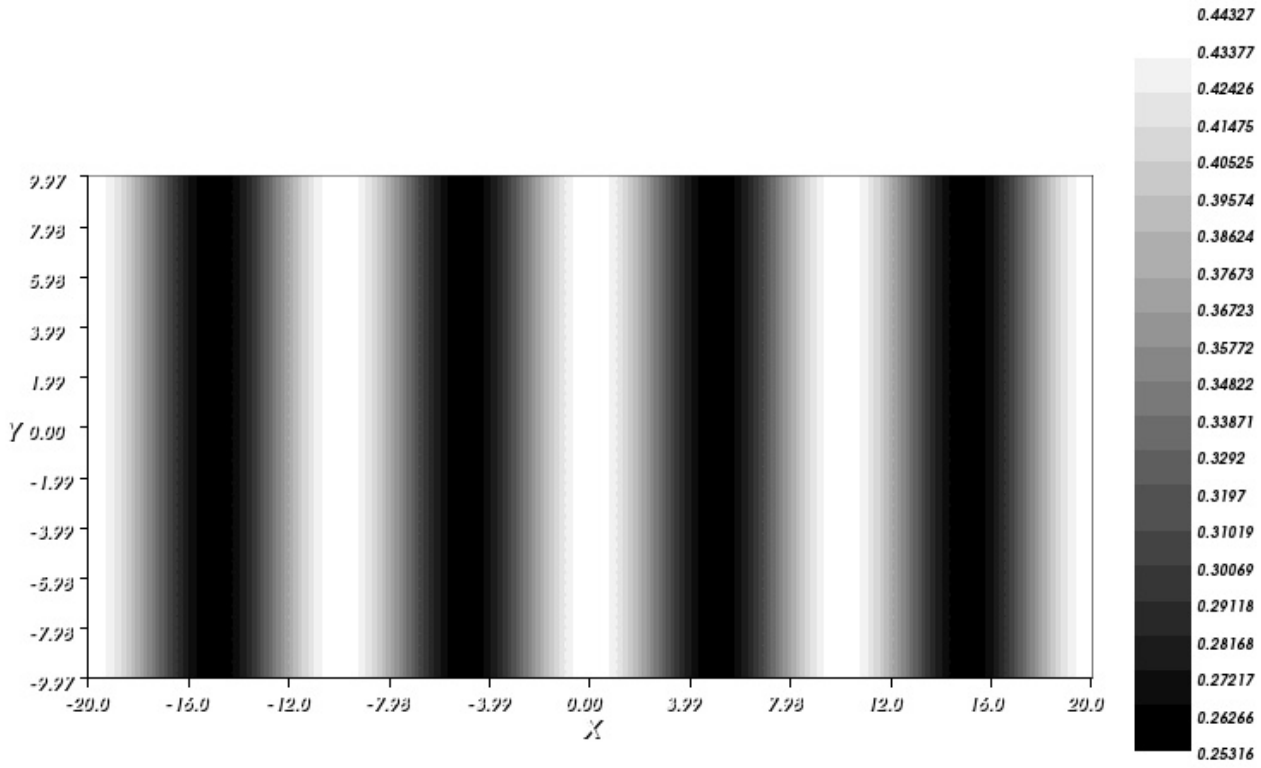


Figure 15: Converged solution v by Newton-Raphson for Schnakenberg at $\lambda = 2.85$

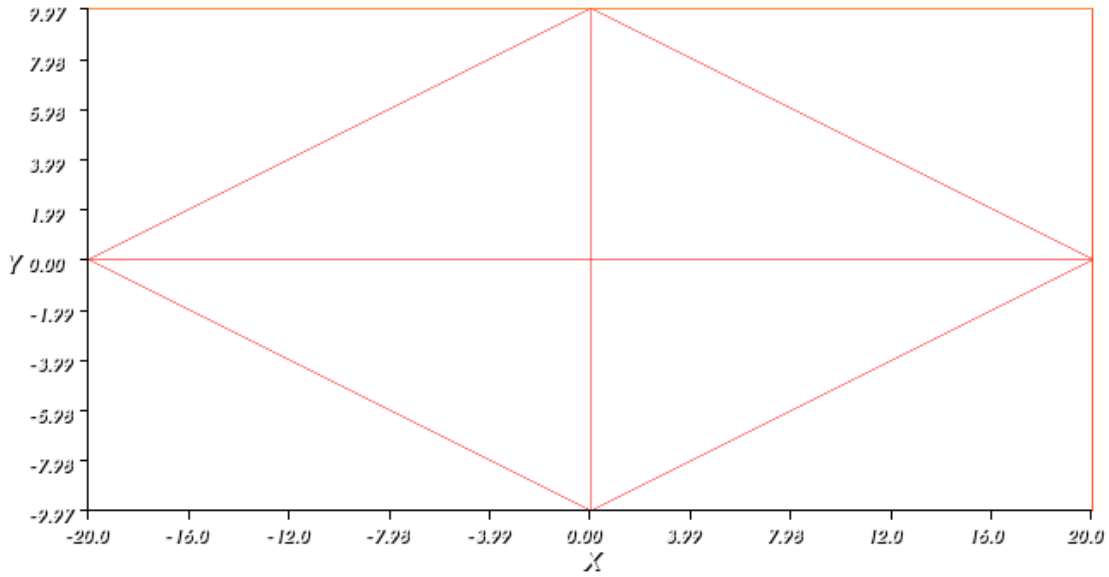


Figure 16: 2×2 symmetric mesh

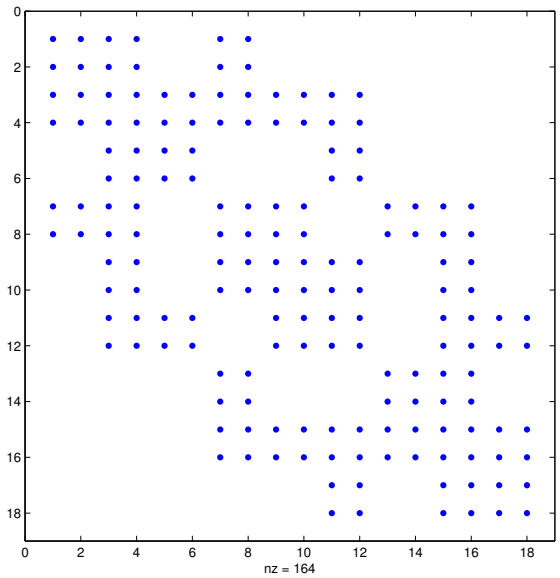


Figure 17: Jacobian for 2×2 mesh at $\lambda = 1$

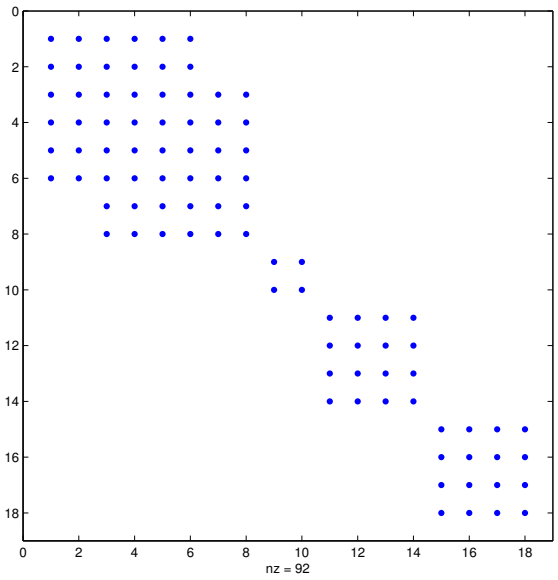


Figure 18: Block diagonalized J for 2×2 mesh at $\lambda = 1$

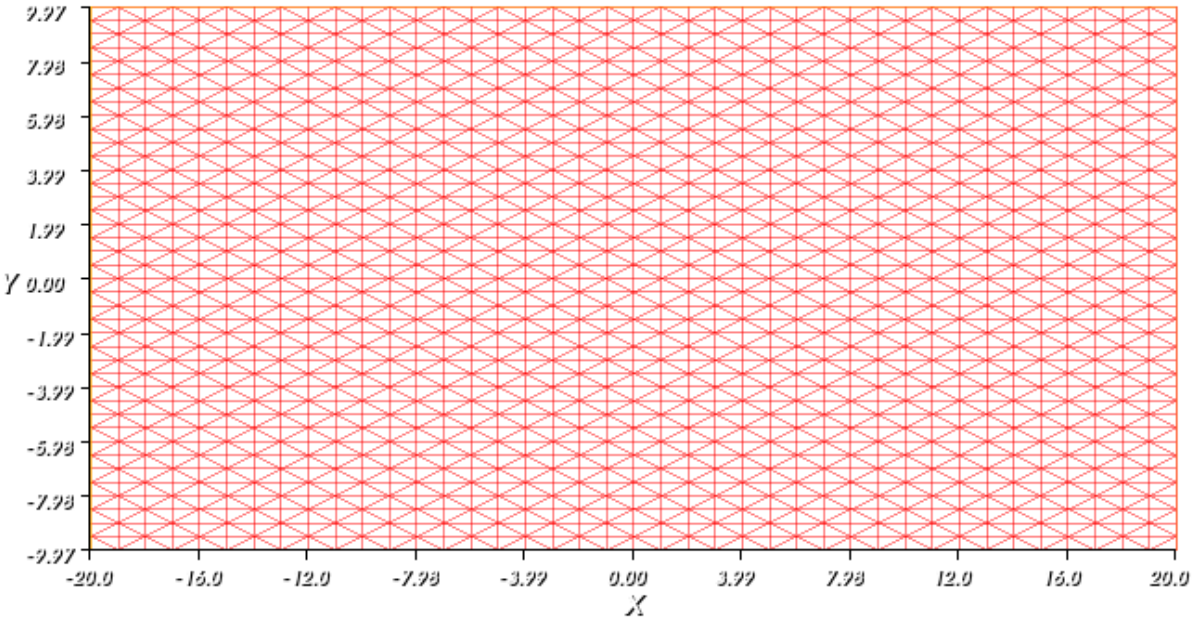


Figure 19: 40×40 symmetric mesh

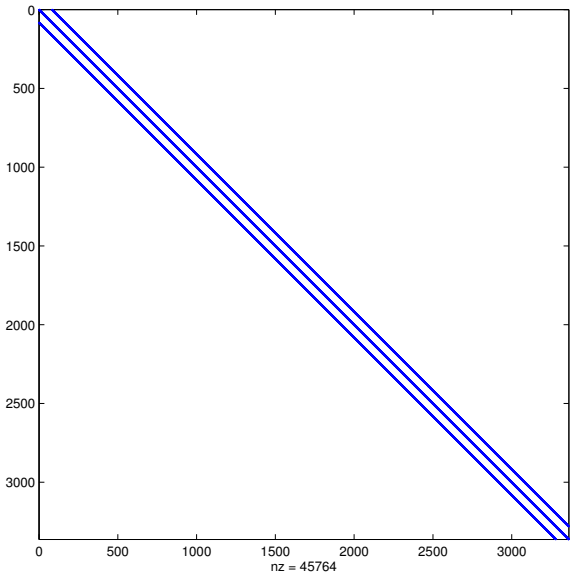


Figure 20: Jacobian for 2×2 mesh at $\lambda = 3.3$

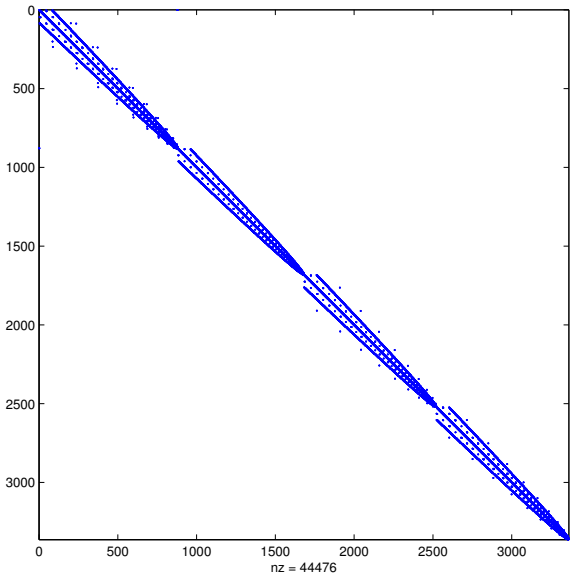


Figure 21: Block diagonalized J for 40×40 mesh at $\lambda = 3.3$

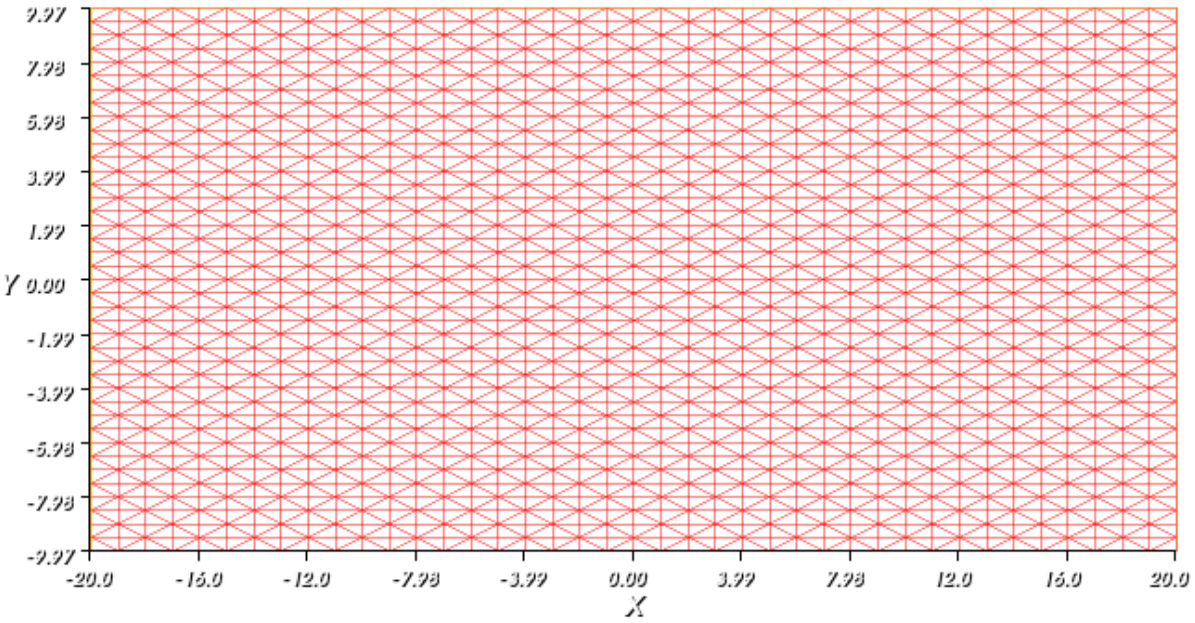


Figure 22: 40×40 symmetric mesh

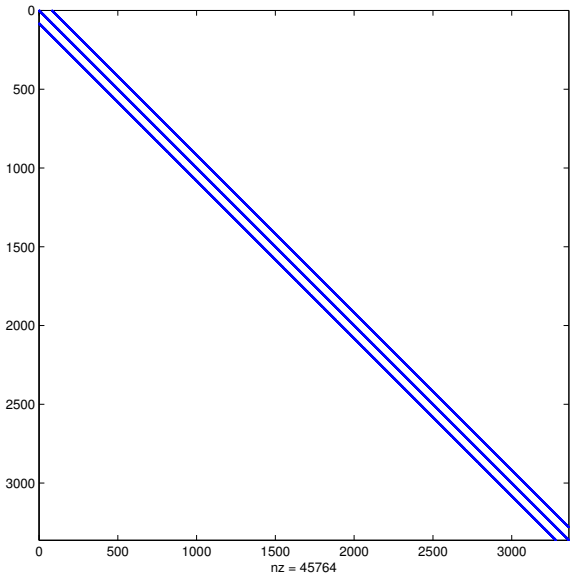


Figure 23: Jacobian for 2×2 mesh at $\lambda = 1$

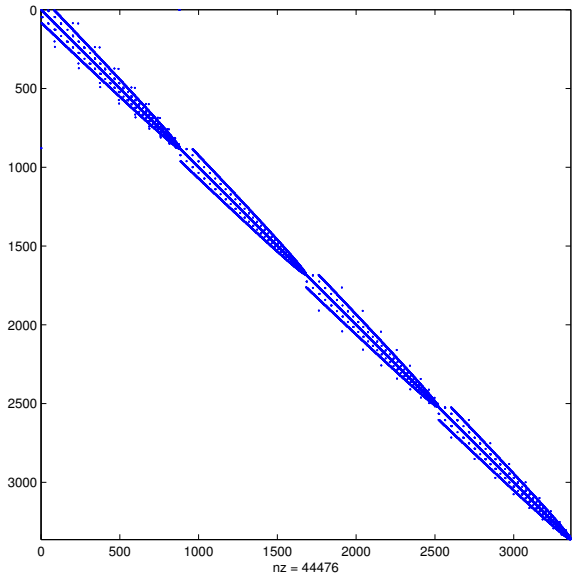


Figure 24: Block diagonalized J for 40×40 mesh at $\lambda = 1$

8

FREEFEM++ AND MATLAB CODES

8.1 FREEFEM++ CODE: SCHNAKENBERG.EDP

For more details of FreeFem++ coding, please refer to [4].

```
//**********Schnakenberg model transient+steady state code**************************//
//**********Authors: Pratik Aghor, Ruchir Dwivedi**********************************//

real a,b,d;
real lambda=1.00;
//parameter definition// [a,b,d]=[0.1,1.1,12] or [0.9,2,24]
a=0;
b=lambda;
d=60;
real t=0.00;
real dt = 0.05;
real tmax=180;
real Lx = 39.9;
real Ly = 19.94;
int it=0;
real gamma=1;
real theta=0.49;
//real alpha= 0.5;
int icase=1;
//
mesh Th=square(90, 90, [-Lx/2+Lx*x, -Ly/2+Ly*y],flags=icase);
plot(Th, wait=0);
fespace Vh(Th, P1);
Vh u,uold,uh,uf;
Vh v,vold,vh,vf;
fespace Wh(Th, P1);
Wh dxu, dyu, dxv, dyv;



//weak form for u
problem heatu (u, uh) =
int2d(Th)(u*uh/dt)
-int2d(Th)(uold*uh/dt)
+int2d(Th)((1-theta)*dx(u)*dx(uh)
+(1-theta)*dy(u)*dy(uh))
+int2d(Th)((theta)*dx(uold)*dx(uh)
+(theta)*dy(uold)*dy(uh));
//
```

```
//weak form for v
problem heatv (v, vh) =
int2d(Th)(v*vh/dt)
-int2d(Th)(vold*vh/dt)
+int2d(Th)(d*(1-theta)*dx(v)*dx(vh)
+d*(1-theta)*dy(v)*dy(vh))
+int2d(Th)(d*(theta)*dx(vold)*dx(vh)
+d*(theta)*dy(vold)*dy(vh));

//Initializing
real m=1;
real kc=0.63;
real epsilon=1e-10;
real A=6;
real B=4;
u =1+0.5*cos(kc*sqrt(3)*(y)/2)*cos(kc*(x)/2)+0.75*cos(kc*x);
uold = u;
plot(u,value=1,fill=1,grey=1,cmm="initialize u",ps="initial_u.jpg");
v  = 1+0.5*sin(kc*(x)/2)*cos(kc*(x)/2);
vold = v;
plot(v,value=1,fill=1,grey=1,cmm="initialize v",ps="initial_v.jpg");
//**********************Fractional Step Method*********************************//
//**********************Step 1*************************************************//
for(real t=0;t<=tmax;t=t+dt)
{
/////////4rth order Ruge-Kutta to solve reaction system:
//eqn is u'=f(t,u); time is from t1 to t2; time step h=(t2-t1)/N; u(t1)=u0;
//step1:
real t2=dt/2,t1=0;
real N=5;
real h= (t2-t1)/N;
//step2:
for(real t=t1;t<t2;t=t+h)
{
Vh K1,K2,K3,K4;
K1=h*(-uold+uold^2*vold); //K1=h*f(t,u(t1));
K2=h*(-(uold+K1/2)+(uold+K1/2)^2*vold);//K2=h*f(t+h/2,u(t1)+K1/2);
K3=h*(-(uold+K2/2)+(uold+K2/2)^2*vold);//K3=h*f(t+h/2,u(t1)+K2/2);
K4=h*(-(uold+K3)+(uold+K3)^2*vold);//K4=h*f(t+h,u(t1)+K3);
uold=uold+(K1+2*K2+2*K3+K4)/6;
//plot(uold,value=1,fill=0);
//t=t1+i*h;
}
for(real t=t1;t<2*t2;t=t+h)
{
Vh A1,A2,A3,A4;
A1=h*(lambda-uold^2*vold);//A1=h*g(t,v(t1));
A2=h*(lambda-uold^2*(vold+A1/2));//A2=h*g(t+h/2,v(t1)+K1/2);
A3=h*(lambda-uold^2*(vold+A2/2));//A3=h*g(t+h/2,v(t1)+K2/2);
A4=h*(lambda-uold^2*(vold+A3));//A4=h*g(t+h,v(t1)+K3);
vold=vold+(A1+2*A2+2*A3+A4)/6;
//t=t1+j*h;
//plot(v,value=1,fill=0);
```

```
}
for(real t=t1;t<t2;t=t+h)
{
Vh K1,K2,K3,K4;
K1=h*(-uold+uold^2*vold); //K1=h*f(t,u(t1));
K2=h*(-(uold+K1/2)+(uold+K1/2)^2*vold);//K2=h*f(t+h/2,u(t1)+K1/2);
K3=h*(-(uold+K2/2)+(uold+K2/2)^2*vold);//K3=h*f(t+h/2,u(t1)+K2/2);
K4=h*(-(uold+K3)+(uold+K3)^2*vold);//K4=h*f(t+h,u(t1)+K3);
uold=uold+(K1+2*K2+2*K3+K4)/6;
//plot(uold,value=1,fill=0);
//t=t1+i*h;
}

//**********************************end of step 1**************************************//

//**********************************Step 2**************************************//
// over time  ( dt )
heatv; vold = v;
//plot(v,fill=1,value=1);
heatu;
uold = u;
//plot(u,fill=1,value=1);
//**********************************end of step 2**************************************//
//**********************************Step 3**************************************//
for(real t=t1;t<t2;t=t+h)
{
Vh K1,K2,K3,K4;
K1=h*(-uold+uold^2*vold); //K1=h*f(t,u(t1));
K2=h*(-(uold+K1/2)+(uold+K1/2)^2*vold);//K2=h*f(t+h/2,u(t1)+K1/2);
K3=h*(-(uold+K2/2)+(uold+K2/2)^2*vold);//K3=h*f(t+h/2,u(t1)+K2/2);
K4=h*(-(uold+K3)+(uold+K3)^2*vold);//K4=h*f(t+h,u(t1)+K3);
uold=uold+(K1+2*K2+2*K3+K4)/6;
//plot(uold,value=1,fill=0);
//t=t1+i*h;
}

for(real t=t1;t<2*t2;t=t+h)
{
Vh A1,A2,A3,A4;
A1=h*(lambda-uold^2*vold);//A1=h*g(t,v(t1));
A2=h*(lambda-uold^2*(vold+A1/2));//A2=h*g(t+h/2,v(t1)+K1/2);
A3=h*(lambda-uold^2*(vold+A2/2));//A3=h*g(t+h/2,v(t1)+K2/2);
A4=h*(lambda-uold^2*(vold+A3));//A4=h*g(t+h,v(t1)+K3);
vold=vold+(A1+2*A2+2*A3+A4)/6;
//t=t1+j*h;
//plot(v,value=1,fill=0);
}
for(real t=t1;t<t2;t=t+h)
{
Vh K1,K2,K3,K4;
K1=h*(-uold+uold^2*vold); //K1=h*f(t,u(t1));
K2=h*(-(uold+K1/2)+(uold+K1/2)^2*vold);//K2=h*f(t+h/2,u(t1)+K1/2);
K3=h*(-(uold+K2/2)+(uold+K2/2)^2*vold);//K3=h*f(t+h/2,u(t1)+K2/2);
```

```
K4=h*(-(uold+K3)+(uold+K3)^2*vold);//K4=h*f(t+h,u(t1)+K3);
uold=uold+(K1+2*K2+2*K3+K4)/6;
//plot(uold,value=1,fill=0);
//t=t1+i*h;
}
//**********************************end of step 3*******************************//


//plot(v,fill=1,value=1);
u=uold;v=vold;
cout<<"t="<<t<<endl;
}
plot(u,fill=1,grey=1,value=1,cmm="u at t="+t,
wait=1,ps="u(t)("+t+").jpg");
plot(v,fill=1,grey=1,value=1,cmm="v at t="+t,
wait=1,ps="v(t)("+t+").jpg");

//**********************************end of transient code************************//

fespace Ph(Th,[P1,P1]);
Ph [uh1,vh1],[u1,v1],[du,dv],[up,vp];
   u1,v1]=[uold,vold];
//*****************************Weak Form(Steady State)*************************//
varf F([du,dv],[uh1,vh1])= int2d(Th)
(gamma*(((a-u1+u1^2*v1)*uh1)
+((b-u1^2*v1)*vh1))
-(dx(u1)*dx(uh1)+dy(u1)*dy(uh1))
-d*(dx(v1)*dx(vh1)+dy(v1)*dy(vh1)));

//***************Linearization*****************************************************//
varf DF([du,dv],[uh1,vh1])=int2d(Th)(
gamma*(-uh1*du+2*u1*v1*uh1*du-2*u1*v1*vh1*du
+u1^2*uh1*dv-u1^2*vh1*dv)-(dx(du)*dx(uh1)+dy(du)*dy(uh1))
-d*(dx(dv)*dx(vh1)+dy(dv)*dy(vh1))
)
;
//******************End of linearization*************************************//
//******************Newton-Raphson*****************************************//
Ph [up1,up2];
for (int i=0;i<400;i++)
{
up[]=u1[];
real[int] B= F(0,Ph);
//real res= B[]'*B[];
//cout << i <<  " residu^2 = " <<  res  << endl;
matrix J;
J= DF(Ph,Ph,solver=UMFPACK);
real[int] w=J^-1*B;
u1[] -= w;
cout << " iter = "<< i <<" "<< " w.linfty= " << w.linfty  << endl;
if( w.linfty< 1e-6) break;
plot (u1,fill=1,value=1,cmm="solution with Newton Raphson u"
,ps="nrSolu("+i+").jpg",grey=1);
```

```
plot (v1,fill=1,value=1,cmm="solution with Newton Raphson v"
,ps="nrSolv("+i+").jpg",grey=1);
ofstream f("J.dat");
f<<J<<endl;
}
//*****************End of Newton-Raphson*****************************************//
```

8.2 MATLAB CODES FOR BLOCK DIAGONALIZATION

The change of basis matrix is written for $N \times N$ symmetric mesh where N is an even number. D_2 symmetry is assumed as the domain was rectangle.

8.3 P1D2.M

```
%N=10;
%c=[];
%for j=1:N/2+1
%for i=1:N/2+1
 %   nodenumber=(j-1)*(N+1)+i
 %   [orbit]=p2(nodenumber)
 %   [c1,c2]=p3(orbit)
 %   c=[c c1 c2];
%end
%end

function Rmat=p1_d2(N)
%N=10;
Rmat=[];
for i=1:N/2
    numberofMorbits=1;
    xco1=N/2+1; yco1=i;
    xco2=N+2-i; yco2=N/2+1;
    xco3=xco1;  yco3=N+2-i;
    xco4=i; yco4=N/2+1;
    xco=[xco3 xco1 ];
    yco=[yco3 yco1 ];

   nodenumber=[];
     for i1=1:2
          nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
     end
     nodenumber
     c=columnM_d2(nodenumber,N);
```

```
    Rmat=[Rmat c];

   xco=[xco2 xco4 ];
   yco=[yco2 yco4 ];

   nodenumber=[];
    for i1=1:2
         nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
    end
    nodenumber
    c=columnV_d2(nodenumber,N);
 Rmat=[Rmat c];


  for j=i:N/2

     xco1=j; yco1=i;
     xco2=N+2-j; yco2=yco1;
     xco3=xco2; yco3=N+2-i;
     xco4=xco1; yco4=yco3;
     xco=[xco3 xco4 xco1 xco2];
     yco=[yco3 yco4 yco1 yco2];


          nodenumber=[];
for i1=1:4
         nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
end
nodenumber
c=column2_d2(nodenumber,N);
Rmat=[Rmat c];


  end

   if (i+1)==N/2+1 break
    else
   end

  for j=i+1:N/2

     xco1=N+2-i; yco1=j;
     xco2=xco1; yco2=N+2-j;
     xco3=i; yco3=yco2;
     xco4=xco3; yco4=yco1;


     xco=[xco2 xco3 xco4 xco1];
     yco=[yco2 yco3 yco4 yco1];

                 nodenumber=[];
for i1=1:4
         nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
```

```
 end
 nodenumber
 c=column2_d2(nodenumber,N);
 Rmat=[Rmat c];

   end


   end
```

8.4 P2D2.M

```
%N=10;
%c=[];
%for j=1:N/2+1
%for i=1:N/2+1
 %   nodenumber=(j-1)*(N+1)+i
 %   [orbit]=p2(nodenumber)
 %   [c1,c2]=p3(orbit)
 %   c=[c c1 c2];
%end
%end

function Rmat=p2_d2(N)
%N=10;
Rmat=[];
for i=1:N/2

   for j=i:N/2

      xco1=j; yco1=i;
      xco2=N+2-j; yco2=yco1;
      xco3=xco2; yco3=N+2-i;
      xco4=xco1; yco4=yco3;
      xco=[xco3 xco4 xco1 xco2];
      yco=[yco3 yco4 yco1 yco2];


           nodenumber=[];
 for i1=1:4
         nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
 end
 nodenumber
 c=column2_d2_12(nodenumber,N);
 Rmat=[Rmat c];


   end

    if (i+1)==N/2+1 break
     else
    end
```

```
   for j=i+1:N/2

      xco1=N+2-i; yco1=j;
      xco2=xco1; yco2=N+2-j;
      xco3=i; yco3=yco2;
      xco4=xco3; yco4=yco1;


      xco=[xco2 xco3 xco4 xco1];
      yco=[yco2 yco3 yco4 yco1];

                  nodenumber=[];
 for i1=1:4
          nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
 end
 nodenumber
 c=column2_d2_12(nodenumber,N);
 Rmat=[Rmat c];

   end


   end
```

8.5 p3d2.m

```
%N=10;
%c=[];
%for j=1:N/2+1
%for i=1:N/2+1
 %   nodenumber=(j-1)*(N+1)+i
 %   [orbit]=p2(nodenumber)
 %   [c1,c2]=p3(orbit)
 %   c=[c c1 c2];
%end
%end

function Rmat=p3_d2(N)
%N=10;
Rmat=[];
for i=1:N/2
    numberofMorbits=1;
    xco1=N/2+1; yco1=i;
    xco2=N+2-i; yco2=N/2+1;
    xco3=xco1;  yco3=N+2-i;
    xco4=i; yco4=N/2+1;
    %xco=[xco3 xco1 ];
    %yco=[yco3 yco1 ];
```

```
  %nodenumber=[];
   % for i1=1:2
    %     nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
    %end
    %nodenumber
    %c=columnM_d2(nodenumber,N);
    %Rmat=[Rmat c];

   xco=[xco2 xco4 ];
   yco=[yco2 yco4 ];

   nodenumber=[];
    for i1=1:2
         nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
    end
    nodenumber
    c=columnV_d2_13(nodenumber,N);
 Rmat=[Rmat c];


  for j=i:N/2

     xco1=j; yco1=i;
     xco2=N+2-j; yco2=yco1;
     xco3=xco2; yco3=N+2-i;
     xco4=xco1; yco4=yco3;
     xco=[xco3 xco4 xco1 xco2];
     yco=[yco3 yco4 yco1 yco2];


          nodenumber=[];
for i1=1:4
         nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
end
nodenumber
c=column2_d2_13(nodenumber,N);
Rmat=[Rmat c];


  end

   if (i+1)==N/2+1 break
    else
   end

  for j=i+1:N/2

     xco1=N+2-i; yco1=j;
     xco2=xco1; yco2=N+2-j;
     xco3=i; yco3=yco2;
     xco4=xco3; yco4=yco1;
```

```
      xco=[xco2 xco3 xco4 xco1];
      yco=[yco2 yco3 yco4 yco1];

                    nodenumber=[];
 for i1=1:4
          nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
 end
 nodenumber
 c=column2_d2_13(nodenumber,N);
 Rmat=[Rmat c];

   end


   end
```

8.6 p4d2.m

```
%N=10;
%c=[];
%for j=1:N/2+1
%for i=1:N/2+1
 %   nodenumber=(j-1)*(N+1)+i
 %   [orbit]=p2(nodenumber)
 %   [c1,c2]=p3(orbit)
 %   c=[c c1 c2];
%end
%end

function Rmat=p4_d2(N)
%N=10;
Rmat=[];
for i=1:N/2
    numberofMorbits=1;
    xco1=N/2+1; yco1=i;
    xco2=N+2-i; yco2=N/2+1;
    xco3=xco1;  yco3=N+2-i;
    xco4=i; yco4=N/2+1;
    xco=[xco3 xco1 ];
    yco=[yco3 yco1 ];

   nodenumber=[];
     for i1=1:2
          nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
     end
     nodenumber
     c=columnM_d2_14(nodenumber,N);
     Rmat=[Rmat c];
```

```
  for j=i:N/2

     xco1=j; yco1=i;
     xco2=N+2-j; yco2=yco1;
     xco3=xco2; yco3=N+2-i;
     xco4=xco1; yco4=yco3;
     xco=[xco3 xco4 xco1 xco2];
     yco=[yco3 yco4 yco1 yco2];


          nodenumber=[];
for i1=1:4
         nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
end
nodenumber
c=column2_d2_14(nodenumber,N);
Rmat=[Rmat c];


  end

   if (i+1)==N/2+1 break
    else
   end

  for j=i+1:N/2

     xco1=N+2-i; yco1=j;
     xco2=xco1; yco2=N+2-j;
     xco3=i; yco3=yco2;
     xco4=xco3; yco4=yco1;


     xco=[xco2 xco3 xco4 xco1];
     yco=[yco2 yco3 yco4 yco1];

                 nodenumber=[];
for i1=1:4
         nodenumber=[nodenumber (yco(i1)-1)*(N+1)+xco(i1)];
end
nodenumber
c=column2_d2_14(nodenumber,N);
Rmat=[Rmat c];

  end


  end
```

8.7 BLOCKD2END.M

This code reads the Jacobian given by FreeFem++ and using Rmatrixd2.m, converts the J into the block diagonal form $JStar$. This code also extracts different blocks from the $JStar$ matrix.

```
fid = fopen('J.dat');
n1    = textscan(fid,'%f',1,'HeaderLines',3);   n1       = n1{1};
m1    = textscan(fid,'%f',1);                   m1       = m1{1};
issym = textscan(fid,'%u',1);                   issymNSL = issym{1};
ncoef = textscan(fid,'%f',1);                   ncoef    = ncoef{1};
%--- read data line by line
        ii  = zeros(ncoef,1);
        jj  = zeros(ncoef,1);
        aij = zeros(ncoef,1);
        for kk=1:ncoef
            iitmp  = textscan(fid,'%u',1);              ii(kk)  = iitmp{1};
            jjtmp  = textscan(fid,'%u',1);              jj(kk)  = jjtmp{1};
            aijtmp = textscan(fid,' %f %c %f %c',1);    aij(kk) = aijtmp{1};
        end
fclose(fid);
Jac = sparse(ii,jj,aij,n1,m1);
        clear('aij','ii','jj');

%Jac+0
spy(Jac)
%************************************************************************%
N=40;
[Rmat,n1,m1,m2,m3,m4]=Rmatrix_d2(N);

JStar=Rmat'*Jac*Rmat;
n1=size(Jac);
for k=1:n1
    for l=1:n1
       if (abs(JStar(k,l))<0.00001 )
           JStar(k,l)=0;
       end
    end
end
block1=JStar(1:2*m1,1:2*m1);
block2=JStar(2*m1+1:2*(m1+m2),2*m1+1:2*(m1+m2));
block3=JStar(2*(m1+m2)+1:2*(m1+m2+m3),2*m2+2*m1+1:2*(m1+m2+m3));
block4=JStar(2*(m1+m2+m3)+1:2*(m1+m2+m3+m4),2*(m1+m2+m3)+1:2*(m1+m2+m3+m4));
```

BIBLIOGRAPHY

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

[2] James D Murray. Animal coat patterns and other practical applications of reaction diffusion mechanisms. In *Mathematical Biology*, pages 435–480. Springer, 1993.

[3] Florian De Vuyst. Numerical modeling of transport problems using freefem++ software with examples in biology, cfd, traffic flow and energy transfer, ens cachan, 2013, ¡cel-00842234¿" pages=162 year =.

[4] Frédéric Hecht, Olivier Pironneau, A Le Hyaric, and K Ohtsuka. Freefem++ manual, 2005.

[5] Kazuo Murota and Kiyohiro Ikeda. Computational use of group theory in bifurcation analysis of symmetric structures. *SIAM journal on scientific and statistical computing*, 12(2):273–297, 1991.

[6] Kiyohiro Ikeda and Kazuo Murota. Bifurcation analysis of symmetric structures using block-diagonalization. *Computer methods in applied mechanics and engineering*, 86(2):215–243, 1991.

[7] Timothy J Healey. A group-theoretic approach to computational bifurcation problems with symmetry. *Computer Methods in Applied Mechanics and Engineering*, 67(3):257–295, 1988.

[8] Martin Golubitsky and Ian Stewart. *The symmetry perspective: from equilibrium to chaos in phase space and physical space*, volume 200. Springer, 2003.

[9] SJ Mohan and R Pratap. A group theoretic approach to the linear free vibration analysis of shells with dihedral symmetry. *Journal of sound and vibration*, 252(2):317–341, 2002.

[10] Sai Jagan Mohan and Rudra Pratap. A natural classification of vibration modes of polygonal ducts based on group theoretic analysis. *Journal of sound and vibration*, 269(3):745–764, 2004.