Study of patterns in reaction-diffusion systems

Demetris Avraam

December 2009

Contents

Introduction

Formation of patterns in many natural systems can be modelled by systems of non-linear reaction-diffusion equations. That equations are often difficult to solve explicitly in closed analytic form and some special methods must be applied in analyzing the nature of the solutions. These reaction-diffusion systems have a variety of solutions such as propagating waves, stationary spots, pulsating waves and self-replicating spots. In two-variable systems, depending on the nature and the number of stationary spatially-homogeneous solutions, we can classify the reaction-diffusion systems into different classes such as excitable, oscillatory and multistable. Some solutions can be observed only in specific systems and some others can be observed in all of these systems with an appropriate set of parameters. Homogeneous oscillations for example, can be observed only in oscillatory systems, transitions from one steady state to another are only observed in multistable systems and other solutions such as travelling waves and stationary spots can be observed in all of these systems. In this project we focus on patterns arising in two-variable excitable system described by Fitz-Hugh Nagumo equations. Before we get to study of patterns, we describe analytical techniques for solving systems of ordinary differential equations (chapter 1), numerical methods for partial differential equations (chapter 2), and we examine their solutions. Also we introduce the reaction-diffusion systems and espsecially the modified Fitz-Hugh Nagumo model (chapters 2 and 3). Finally, we describe one-dimensional patterns arising in response to a single stimulation and classify them. All the results presented in the project, can be useful to the reader who is interested about pattern-formation to understand phenomena in hydrodynamic systems, nonlinear optics, autocatalytic chemical reactions and variety of biological systems including excitable tissues (muscles and nerves), developing biological systems (morphogenesis, gastrulation) and many more. For further details about this phenomena, the interested reader should consult [1], [2], [3] and [4].

Chapter 1

Analytical techniques for solving systems of ordinary differential equations

Differential equations arise in many areas of sciences and play an important role to the examination of many phenomena. Differential equations are the mathematical equations for an unknown function of one or several variables that relates the values of the function itself and its derivatives of various orders. Ordinary differential equations, are the differential equations, in which the unknown function is a function of a single independent variable and partial differential equations, are the differential equations in which the unknown fuction is a function of multiple independent variables, and the equation involves its partial derivatives. Both ordinary and partial differential equations are classified as linear and nonlinear. In this first section we present the analytical methods for solving systems of ordinary differential equations and we examine some of their solutions. The solutions give the phase plane which is a visual display of certain characteristics of certain kinds of differential equations and is useful in visualizing the behavior of dynamical systems (if are stable or unstable).

1.1 Linear Systems

1. For one linear differential equation of the form

   $$\frac{dx}{dt} = ax + b \tag{1.1}$$

   where a, b are constants, the solution is given by

   $$x(t) = c_1 e^{at} - \frac{b}{a} \tag{1.2}$$

   where c_1 is arbitrary constant.

   The following phase-portraits shows the different solutions of linear equation (1.1) for $a < 0$ and $a > 0$ respectively:

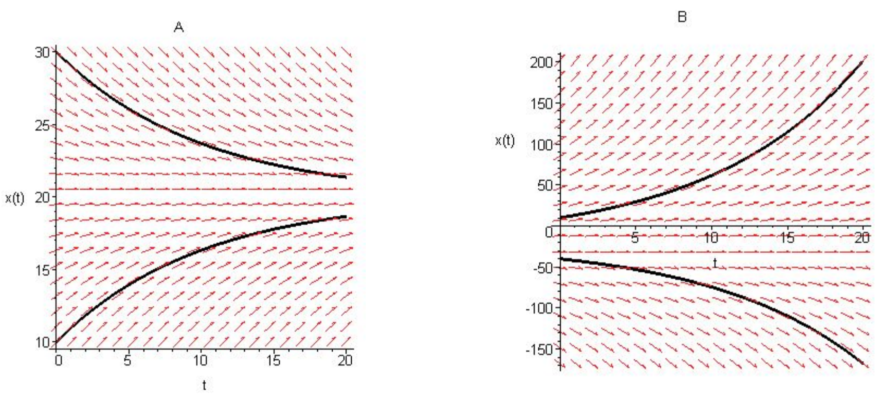

Figure 1.1: Phase-portraits of (1.1). **A**: for $a < 0$ and **B**: for $a > 0$.

2. The linear form of two equations is equivalent to the system:

$$\frac{dx}{dt} = a_{11}x + a_{12}y \tag{1.3}$$

$$\frac{dy}{dt} = a_{21}x + a_{22}y. \tag{1.4}$$

We can define the Jacobian matrix

$$J = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \tag{1.5}$$

where $a_{11}, a_{12}, a_{21}, a_{22}$ are real constants.

Let λ_1 and λ_2 be the eigenvalues of the matrix J, that is

$$\begin{vmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{vmatrix} = 0 \Rightarrow (a_{11} - \lambda)(a_{22} - \lambda) - a_{12}a_{21} = 0 \tag{1.6}$$

$$\Rightarrow \lambda^2 - (a_{11} + a_{22})\lambda + (a_{11}a_{22} - a_{12}a_{21}) = 0 \tag{1.7}$$

$$\Rightarrow \lambda_1, \lambda_2 = \frac{(a_{11} + a_{22}) \pm [(a_{11} + a_{22})^2 - 4(a_{11}a_{22} - a_{12}a_{21})]^{1/2}}{2} \tag{1.8}$$

$$\Rightarrow \lambda_1, \lambda_2 = \frac{trJ \pm [(trJ)^2 - 4detJ]^{1/2}}{2}. \tag{1.9}$$

Let $D = X^{-1}JX \Rightarrow J = XDX^{-1}$. We can write the system (1.3), (1.4) in the form

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = XDX^{-1} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{1.10}$$

The solutions of this system are then

$$\begin{bmatrix} x \\ y \end{bmatrix} = c_1\mathbf{u_1}e^{\lambda_1 t} + c_2\mathbf{u_2}e^{\lambda_2 t}, \tag{1.11}$$

where c_1 and c_2 are arbitrary constants and $\mathbf{u_1}, \mathbf{u_2}$ are the eigenvectors of J corresponding to λ_1 and λ_2 respectively. This form of solutions is for distinct eigenvalues. If the eigenvalues are equal the solutions are proportional to

$$\begin{bmatrix} x \\ y \end{bmatrix} = (c_1 + c_2 t)e^{\lambda t}. \tag{1.12}$$

If λ_1 and λ_2 are real and distinct, then:

(i) If λ_1 and λ_2 have the same sign we take the *node* case where we have for $\lambda_1 \leq 0,\ \lambda_2 \leq 0$, a *stable node*, and for $\lambda_1 > \lambda_2 > 0$, we have an *unstable node.*

(ii) If λ_1 and λ_2 have different signs we have the *saddle point* which is always unstable.

If λ_1 and λ_2 are complex: $\lambda_1, \lambda_2 = a \pm ib,\ b \neq 0$, then

(i) If $a \neq 0$ we have a *spiral* which is stable if $a < 0$ and unstable if $a > 0$.

(ii) If $a = 0$ we have a *centre* which is not stable in the usual sense.

If λ_1 and λ_2 are not distinct, $\lambda_1 = \lambda_2 = \lambda$, we take a *node.*

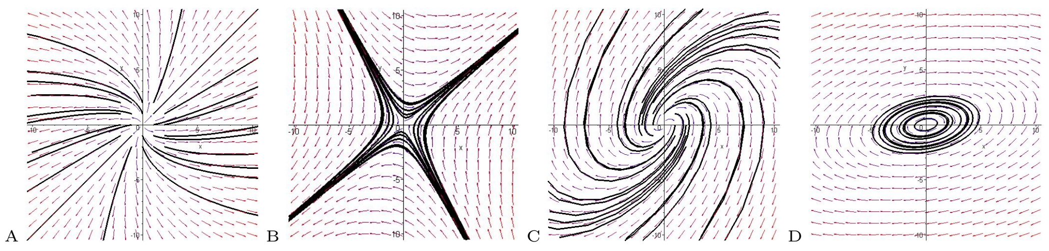

Figure 1.2: Phase portraits for linear differential system (1.3), (1.4). **A**: Unstable node for $\lambda_1, \lambda_2 > 0$, **B**: Saddle point for $\lambda_1 > 0, \lambda_2 < 0$, **C**: Unstable spiral for complex λ_1, λ_2 with positive real parts and **D**: Centre for complex λ_1, λ_2 with real parts zero.

The following diagram showing how $trJ = a_{11}+a_{22}$ and $detJ = a_{11}a_{22}-a_{21}a_{12}$ determine the type of phase plane singularity for the system (1.3), (1.4).

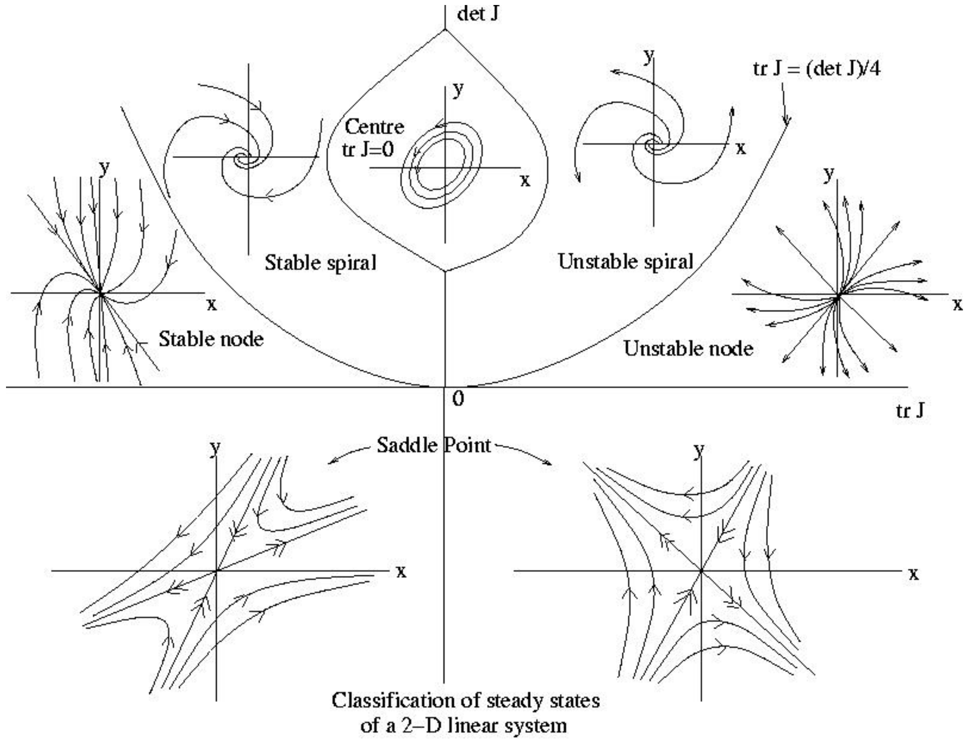

Figure 1.3: Classification of steady-states of the linear system (1.3), (1.4).

1.2 Nonlinear systems

1. For one nonlinear differential equation in the form:

$$\dot{x} = f(x) \tag{1.13}$$

   where f(x) is a function of a variable x the solutions can be found using qualitative analysis.

   Let's consider two simple nonlinear examples:

   (a) Quadratic where the function $f(x)$ in equation (1.13) is written in the form $f(x) = ax^2 + bx + c$ where a, b, c are constants. In particular for the Fisher equation we have $f(x) = x(x-1)$. Here are two stationary points $x = 0$ and $x = 1$. For this example see figure 1.4**A** about the plot of $f(x)$ versus x for Fisher equation. Since $\dot{x} = f(x)$, x is increasing when f is positive, decreasing when f is negative, and stationary when $f = 0$. Thus implies that in this example the first stationary point $x = 0$, is stable and the second $x = 1$, is unstable.

(b) Cubic equation where in (1.13) the function $f(x)$ has the form $f(x) = ax^3 + bx^2 + cx + d$ and in particular form we have the Fitz-Hugh equation where $f(x) = -x(x - x_0)(x - x_1)$. For this example see figure 1.4**B** about the plot of $f(x)$ versus x for Fitz-Hugh equation particular for $x_0 = 1$ and $x_1 = 3$. Here we have 3 stationary points, where $x = 0$ is stable, $x_0 = 1$ is unstable and $x_1 = 3$ is stable. This case is called *multistable.*

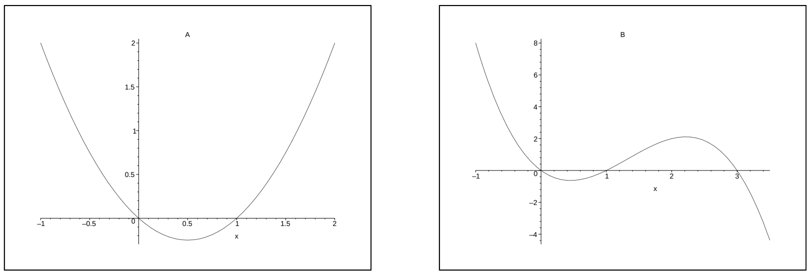

Figure 1.4: Plot of function $f(x)$ for the nonlinear equation (1.13). **A**: for Fisher equation and **B**: for Fitz-Hugh equation.

2. For the nonlinear system with two first order differential equations we consider the general form of:

$$\dot{x} = f(x, y) \tag{1.14}$$

$$\dot{y} = g(x, y) \tag{1.15}$$

where x and y are functions of time t.

Definition
A point (x^*, y^*) is called an equilibrium point of the system (1.14), (1.15) if

$$f(x^*, y^*) = 0, \; g(x^*, y^*) = 0. \tag{1.16}$$

To explore the stability of steady states values of a nonlinear differential system we will apply linearization as described in the next section.

1.3 Linearization

Linearization refers to the linear approximation to a function at a given point. In the study of dynamical systems, linearization is a method for assessing the stability of an equilibrium point of a system of nonlinear differential equations. For the system (1.14), (1.15), we take the steady-state values x_0 and y_0, which satisfy

$$f(x_0, y_0) = 0 \text{ and } g(x_0, y_0) = 0. \tag{1.17}$$

We can examine the stability of these steady states by analyzing them with small deviations. This will result in a linearised system of equation for the small perturbations ε and δ. To achieve this, we must use Taylor series expansions of the functions of two variables, f and g. We assume that

$$x = x_0 + \varepsilon \Rightarrow \dot{x} = \dot{\varepsilon} \tag{1.18}$$

$$y = y_0 + \delta \Rightarrow \dot{y} = \dot{\delta}. \tag{1.19}$$

We approximate $f(x_0 + \varepsilon, y_0 + \delta)$ and $g(x_0 + \varepsilon, y_0 + \delta)$ by the expressions

$$\dot{x} = f(x_0 + \varepsilon, y_0 + \delta) = f(x_0, y_0) + \varepsilon f'_x(x_0, y_0) + \delta f'_y(x_0, y_0) \tag{1.20}$$

$$\Rightarrow \dot{\varepsilon} = \varepsilon f'_x + \delta f'_y \tag{1.21}$$

and similar

$$\dot{\delta} = \varepsilon g'_x + \delta g'_y. \tag{1.22}$$

The Jacobian matrix now has the form

$$J = \begin{pmatrix} f'_x & f'_y \\ g'_x & g'_y \end{pmatrix}. \tag{1.23}$$

Comparing this form of Jacobian with the (1.5), we now take the $tr(J)$ and $det(J)$ by

$$tr(J) = f'_x + g'_y \quad \text{and} \quad det(J) = f'_x g'_y - f'_y g'_x. \tag{1.24}$$

Now the problem has been reduced to a linear system of equations for states that are nearness to the steady state (x_0, y_0). Thus we can determine the stability of (x_0, y_0) by methods for linear system equations as we refer in section 1.1.

Example:

In the following example we have a system of one nonlinear equation (1.25), and one linear equation (1.26)

$$\dot{x} = -x(x - a)(x - 1) - y \tag{1.25}$$

$$\dot{y} = b(x - y) + c. \tag{1.26}$$

The nullclines $\dot{x} = 0$, $\dot{y} = 0$, of this system gives the stationary points for different values of constants a, b, c. If we have one stable steady state point the dynamical system is excitable (see figure 1.5**A**). For one unstable steady state point (see figure 1.5**B**), we have an oscillatory system and for more stationary points we have a multistable system. In particular in the figure 1.5**C** for the multistable system we

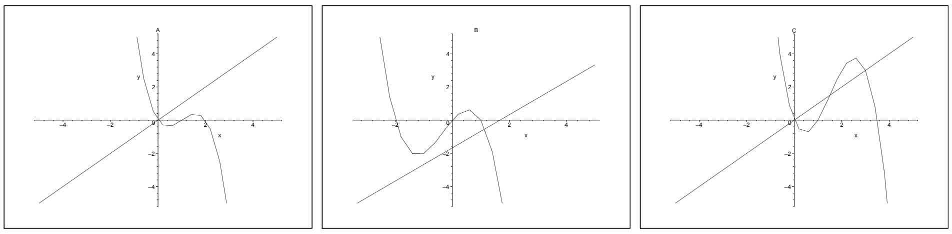

Figure 1.5: Respectively of the number and stability of steady states, we can have: **A**: excitable system, **B**: oscillatory system and **C**: multistable system.

have 3 stationary points, were the first one and the third are stable and the second is unstable.

We have seen that by linearizing a set of nonlinear equations about a given steady state, we can describe the behavior of that system. This behavior falls into the possible cases which were described in figure 1.3. Consider that we have equations with parameters. The bifurcation is a qualitative change of the equations properties as the parameter change continiously. In this project we describe the Hopf bifurcation, where a transition from a stable to an unstable focus, or form an unstable to stable focus exists, as a parameter c is varied.

1.4 Hopf bifurcation

Let us consider the following system of differential equations depending on parameter c

$$\dot{x} = f(x, y, c) \tag{1.27}$$

$$\dot{y} = g(x, y, c). \tag{1.28}$$

We assume that the system has an equilibrium point at $(x_0(c), y_0(c))$, such that eigenvalues of the Jacobian matrix

$$J(c) = \left(\begin{array}{cc} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{array} \right)_{(x_0, y_0)} \tag{1.29}$$

are complex in the form $\lambda_{1,2} = a(c) \pm ib(c)$, where $a(c)$ is the real part and $ib(c)$ is the imaginary part of λ. The solutions have the form:

$$\left[\begin{array}{c} x \\ y \end{array} \right] = c_1 \mathbf{u_1} e^{(a(c)+ib(c))t} + c_2 \mathbf{u_2} e^{(a(c)-ib(c))t}. \tag{1.30}$$

At some point, however, the fixed point looses its stability, which implies that the real part of the eigenvalues changes from negative to positive. In other words, the real part passes through zero. We suppose that there is a value c_1 such that $a(c_1) = 0$ and $b(c_1) \neq 0$. At this point the behaviour of the system changes form stable spiral to unstable or vice versa. The point where the transition occurs is called a bifurcation point, and c is the bifurcation parameter. If the solution lose its stability, the case is called a supercritical Hopf-bifurcation and if the solution which appears the Hopf bifurcation, is itself unstable, this case is called subcritical Hopf bifurcation.

1.5 Reduction of fast-slow systems

We consider the system of differential equations

$$\frac{du}{dt} = f(u, v) \tag{1.31}$$

$$\frac{dv}{dt} = \varepsilon g(u, v) \tag{1.32}$$

of which u is 'fast' variable and v is 'slow', and $0 < \varepsilon << 1$ is a very small parameter. The transformation of time $T = \varepsilon t$ brings the system to the form

$$\varepsilon \frac{du}{dT} = f(u, v) \tag{1.33}$$

$$\frac{dv}{dT} = g(u, v). \tag{1.34}$$

Systems (1.31), (1.32) (the fast-time system) and (1.33), (1.34) (the slow-time system) are equivalent to each other for every fixed $\varepsilon > 0$, but have different properties in the limit $\varepsilon \to 0$.

Since $\varepsilon = 0$ from the fast-time system,

$$\frac{dv}{dt} \simeq 0 \;\Rightarrow\; v = \text{constant} = v_1$$

and the system (1.31), (1.32) reduces to the equation

$$\frac{du}{dt} = f(u, v_1) \tag{1.35}$$

which means that the slow variable v remain constant and only the fast variable u vary.

From the slow-time system for $\varepsilon = 0$

$$f(u, v) = 0 \;\Rightarrow\; u = \Phi(v) \text{ where } \Phi \text{ is a function of variable } v$$

and the system (1.33), (1.34) reduces to the equation

$$\frac{dv}{dt} = g(\Phi(v), v). \tag{1.36}$$

Reduction is the technique used to reduce the differential system of two variables u and v to a system of one variable. We consider the two dynamic variables u and v which are one 'fast' and the other 'slow' and we can called the differential system fast-slow time system.

Chapter 2

Spatio-temporal patterns in systems of partial differential equations

2.1 Spatially distributed systems

Assume that we have a function of two variables $f(x,t)$, i.e. x is the space and t is the time. We can find derivatives of this function in the x-direction $\partial f/\partial x$ and in the t-direction $\partial f/\partial t$.

If we have a process of diffusion of a substance, then it is possible to show that the flux is directly proportional to the concentration gradient and directed to the side which is opposite to the gradient. Therefore we can write:

$$\overrightarrow{J} = -D\nabla u, \tag{2.1}$$

where $J(x)$ is a flux of particles at the section x which means number of particles crossing a unit area at x in the x direction per unit of time, and in a multi-dimensional space we take a vector $\overrightarrow{J}$ as the flux depends on orientation of a surface. D is the proportionality coefficient which is called the diffusion coefficient, and ∇u is the gradient of the function u, i.e. is a vector which has the x-coordinate $\partial u/\partial x$ and t-coordinate $\partial u/\partial t$. The equation which describes the diffusion in one dimension (1D) is given by:

$$\frac{\partial u}{\partial t} = -\frac{\partial J}{\partial x} = D\frac{\partial^2 u}{\partial x^2}. \tag{2.2}$$

2.2 Reaction diffusion systems

A theoretical consideration of many natural systems shows that they can described by systems of nonlinear reaction-diffusion equations. Depending on the nature and the number of stationary spatially homogeneous solutions, reaction diffusion systems can fall into different classes such as excitable, oscillatory and multistable. The widely known solutions of that systems are travelling waves and stationary spots. In chapter 3, we focus on patterns arising in two-variable excitable system described by modified Fitz-Hugh Nagumo equations and we describe all one-dimensional patterns of propagating waves and stationary spots.

Mathematically, reaction-diffusion systems can be described by nonlinear partial differential equations in the general form as follows:

$$\frac{\partial u}{\partial t} = D_u \Delta u + \epsilon_u f(u, v) \tag{2.3}$$

$$\frac{\partial v}{\partial t} = D_v \Delta v + \epsilon \phi(u, v), \tag{2.4}$$

where the functions $f(u, v)$ and $\phi(u, v)$ define the kinetics of the dependent variables u and v, ϵ_u and ϵ specify the rate of kinetics terms, $\Delta u = \partial^2 u/\partial x^2$, $\Delta v = \partial^2 v/\partial x^2$ and D_u, D_v are the diffusion coefficients.

Modified Fitz-Hugh Nagumo model

Consider we have the reaction diffusion system (2.3), (2.4). For the functions $f(u, v)$ and $\phi(u, v)$, we can use the kinetics terms

$$f(u, v) = -k_u u(u - u_0)(u - u_1) - k_v v \text{ and } \phi(u, v) = u - v.$$

These terms are from the widely known Fitz-Hugh Nagumo (FHN) model describing exitable media (for further details the reader should consult [5]). Also rescaling of time and space allows elimination of one of the diffusion coefficients and one of the ϵ -parameters, so that we can put into (2.3), (2.4), $D_u = 1$ and $\epsilon_u = 1$. Thus the system take the form

$$\frac{\partial u}{\partial t} = \Delta u - (k_u u(u - u_0)(u - u_1) + k_v v) \tag{2.5}$$

$$\frac{\partial v}{\partial t} = D_v \Delta v + \epsilon(u - v). \tag{2.6}$$

At this system we can define the variable u as an activator which involved in the increase of its own production and variable v as an inhibitor which reduces the production rate of activator. The diffusive spread of the inhibitor around the excited area is often referred to as lateral inhibition (more information in [6]), where k_u, k_v are constants adjusting the kinetics (i.e. excitable, bistable) of the system.

For numerical integration of the system (2.5), (2.6), we have used the Euler explicit method with central differencing for the diffusion terms and with no-flux conditions on the boundaries of the medium (described in section 2.5).

2.3 Propagating wave solution

Consider we have the excitable system described by the following Fitz-Hugh Nagumo system:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + f(u,v) \tag{2.7}$$

$$\frac{\partial v}{\partial t} = \epsilon(u-v) \tag{2.8}$$

where $f(u,v) = -u(u-a)(u-1) - v$.

Travelling waves solutions of the system (2.7), (2.8), in which u and v are functions only of the travelling coordinate variable $z = x - ct$, satisfy the travelling coordinate form of (2.7), (2.8), namely

$$-c\frac{\partial u}{\partial z} = D\frac{\partial^2 u}{\partial z^2} + f(u,v) \tag{2.9}$$

$$-c\frac{\partial v}{\partial t} = \epsilon(u-v) \tag{2.10}$$

where c denotes the velocity of the wave.

While the wave is moving right we examine the two parts. The wave front ($v = v^+$) where u increases, and the wave back ($v = v^-$) where u decreases. From the equation (2.9), for the wave front part we have

$$D\frac{\partial^2 u}{\partial z^2} + c\frac{\partial u}{\partial z} + f(u,v^+) = 0. \tag{2.11}$$

We multiply this equation by $\partial u/\partial z$

$$\Rightarrow \frac{1}{2}D[(\frac{\partial u}{\partial z})^2]'_z + c(\frac{\partial u}{\partial z})^2 + f(u,v^+)\frac{\partial u}{\partial z} \tag{2.12}$$

and integrate respect to z

$$\Rightarrow \int_0^\infty \frac{1}{2}D[(\frac{\partial u}{\partial z})^2]'_z\partial z + \int_0^\infty c(\frac{\partial u}{\partial z})^2\partial z + \int_0^\infty f(u,v^+)\frac{\partial u}{\partial z}\partial z. \tag{2.13}$$

So we get the velocity c of the wave in the form

$$c = \frac{-\int_0^1 f(u,v^+)\partial u}{\int_0^\infty(\frac{\partial u}{\partial z})^2\partial z}. \tag{2.14}$$

Similarly about the back wave part we have

$$c = \frac{-\int_0^1 f(u,v^-)\partial u}{\int_0^\infty(\frac{\partial u}{\partial z})^2\partial z}. \tag{2.15}$$

While the wave is moving right, if its shape does not change, then the velocity must be equal in the two parts of the wave

$$\Rightarrow \int_0^1 f(u,v^+)\partial u = \int_0^1 f(u,v^-)\partial u. \tag{2.16}$$

2.4 Turing instability

Consider we have the system

$$\frac{\partial u}{\partial t} = \Delta_u + \gamma f(u,v) \tag{2.17}$$

$$\frac{\partial v}{\partial t} = D_v \Delta_v + \gamma \phi(u,v). \tag{2.18}$$

We suppose that (u_0, v_0) is a constant equilibrium solution,

$$f(u_0, v_0) \text{ and } \phi(u_0, v_0) = 0. \tag{2.19}$$

In fact, the constant solution (u_0, v_0) is also an equilibrium solution of a system of ordinary differential equations:

$$\frac{du}{dt} = \gamma f(u,v) \tag{2.20}$$

$$\frac{dv}{dt} = \gamma \phi(u,v). \tag{2.21}$$

Turing suggested that, under certain conditions, chemical can react and diffuse in such way as to produse non-constant equilibrium solutions, which represent spatial patterns of chemical or morphogen concentration. Turing's theory said that if in the absense of the diffusion (system (2.20), (2.21)), we have a stable steady state, then, under certain conditions, the uniform steady state can become unstable and spatial patterns can evolve through bifurcations. So a constant equilibrium can be stable with respect to the system (2.20), (2.21), but it is unstable with respect to (2.17), (2.18). Therefore this constant equilibrium solution becomes unstable because of the diffusion. More information of Turing instability can be found in [1].

In the absense of diffusion the solution of the system is determined by analytical techniques which are described in chapter 1. With linearization we take the Jacobian matrix (1.23), with the partial derivatives of f and ϕ. The eigenvalues have the form

$$\lambda_1, \lambda_2 = \frac{(f'_u + \phi'_v) \pm [(f'_u + \phi'_v)^2 - 4(f'_u\phi'_v - f'_v\phi'_u]^{1/2}}{2}. \tag{2.22}$$

Linear stability is defined if the eigenvalues have negative real part, $Re\lambda < 0$, which is equivalent to

$$tr(J) = f'_u + \phi'_v < 0 \text{ and } det(J) = f'_u\phi'_v - f'_v\phi'_u > 0. \tag{2.23}$$

Now suppose that (u_0, v_0) is unstable with respect to the reaction-diffusion system (2.17), (2.18). Then (u_0, v_0) must be unstable with respect to the following linearized system, with small perturbations ϵ and δ

$$\frac{\partial \epsilon}{\partial t} = \frac{\partial^2 \epsilon}{\partial x^2} + \gamma f'_u(u_0, v_0)\epsilon + \gamma f'_v(u_0, v_0)\delta, \tag{2.24}$$

$$\frac{\partial \delta}{\partial t} = D_v \frac{\partial^2 \delta}{\partial x^2} + \gamma \phi'_u(u_0, v_0)\epsilon + \gamma \phi'_v(u_0, v_0)\delta. \tag{2.25}$$

We consider the matrix notation:

$$\Psi_t = D\Psi_{xx} + \gamma J \Psi, \tag{2.26}$$

where

$$\Psi(t,x) = \begin{pmatrix} \epsilon(t,x) \\ \delta(t,x) \end{pmatrix}, \text{ and } D = \begin{pmatrix} 1 & 0 \\ 0 & D_v \end{pmatrix}. \tag{2.27}$$

For the separation of variables for single equation, we except that ϵ and δ are both separable, and we try the following form

$$\Psi(t,x) = \begin{pmatrix} \epsilon(t,x) \\ \delta(t,x) \end{pmatrix} = \begin{pmatrix} u_k(t) \\ v_k(t) \end{pmatrix} cos(k\pi x) = W_k(t) cos(k\pi x). \tag{2.28}$$

We shall try this form of solution, and the general solution of Ψ would be

$$\Psi(t,x) = \sum_{k=0}^{\infty} W_k(t) cos(k\pi x). \tag{2.29}$$

By substitution of this form into (2.26), we obtain that $W_k(t)$ satisfies a system of ordinary differential equations:

$$\frac{dW_k}{dt} = (\gamma J - k^2\pi^2 D) W_k. \tag{2.30}$$

If (2.24), (2.25) is an unstable system, then at least one of $W_k(t)$ would go to infinity as $t \to \infty$, one of the eigenvalues of matrix $M_k = \gamma J - k^2\pi^2 D$ has positive real part. The eigenvalues λ_1 and λ_2 of M_k satisfy

$$\lambda^2 - [\gamma(f'_u + \phi'_v) - (D_v + 1)k^2\pi^2]\lambda + (f'_u\phi'_v - f'_v\phi'_u)\gamma^2 - k^2\pi^2(D_v f'_u + \phi'_v)\gamma + D_v k^4\pi^4 = 0, \tag{2.31}$$

and

$$tr(M_k) = \gamma(f'_u + \phi'_v) - (D_v + 1)k^2\pi^2, \tag{2.32}$$

$$det(M_k) = (f'_u\phi'_v - f'_v\phi'_u)\gamma^2 - k^2\pi^2(D_v f'_u + \phi'_v)\gamma + D_v k^4\pi^4. \tag{2.33}$$

Since $f'_u + \phi'_v < 0$, then $tr(M_k) < 0$. So M_k cannot be an unstable spiral or unstable node (see figure 1.3). If M_k is unstable, then it can only be a saddle. Therefore we must have $det(M_k) < 0$.

We only need one of M_k is unstable. So we look the values of $h(k^2\pi^2) = det(M_k)$ for different k. Here we define the function

$$h(\omega) = (f'_u\phi'_v - f'_v g'_u)\gamma^2 - \omega(D_v f'_u + g'_v)\gamma + D_v\omega^2. \tag{2.34}$$

When $\omega = 0,\ h(0) = (f'_u\phi'_v - f'_v\phi'_u)\gamma^2 > 0$ because we require $f'_u\phi'_v - f'_v\phi'_u > 0$, and when $k^2\pi^2 \to \infty$, we also have

$$\lim_{\omega \to \infty} h(\omega) = \infty. \tag{2.35}$$

So we must have for some $0 < k^2\pi^2 < \infty$, $h(k^2\pi^2) < 0$. Since h is a quatratic equation, we must require

$$\min_{\omega \geq 0} h(\omega) < 0. \tag{2.36}$$

It is easy to caldulate that the minimum value at ω_0 is achieved at

$$\omega_0 = \frac{D_v f'_u + \phi'_v}{2D_v}. \tag{2.37}$$

This implisitly implies that

$$D_v f'_u + \phi'_v > 0, \tag{2.38}$$

otherwise $h(\omega) > 0$ for all $\omega \geq 0$. The minimum value at ω_0 is

$$h(\omega_0) = \gamma^2 (f'_u \phi'_v - f'_v \phi'_u - \frac{(D_v f'_u + \phi'_v)^2}{4D_v}), \tag{2.39}$$

which must be negative, so

$$(D_v f'_u + \phi'_v)^2 - 4D_v(f'_u \phi'_v - f'_v \phi'_u) > 0. \tag{2.40}$$

Summarizing all conditions we got, if (u_0, v_0) is stable with respect to (2.17), (2.18) but unstable with respect to (2.20), (2.21), then D_v and J must satisfy

$$f'_u + \phi'_v < 0, \quad f'_u \phi'_v - f'_v \phi'_u > 0, \tag{2.41}$$

$$D_v f'_u + \phi'_v > 0, \quad (D_v f'_u + \phi'_v)^2 - 4D_v(f'_u \phi'_v - f'_v \phi'_u) > 0. \tag{2.42}$$

2.5 Numerical methods of integration of partial differential equations

In order to study partial differential equation (PDE) numerically we need to find numerical approximations for the main spatial and time derivatives of the function $u(x)$ and other functions which are used in our equations. In this section we describe how we can solve numerically the partial differential equations. Let us formulate a procedure for numerical solution of PDE similar to the diffusion equation (2.2) in 1D space. We need to specify in which region R of space we want to solve our equation, and we need to specify the initial distribution of the substance $u = u_0$. After that we can discretize our equation in the region R.

Assume that our region R is the interval $[x_0, x_k]$ and that the point x_i belong in this interval for $i = 0, 1, \ldots, k$. Let us correspond all the points x_i from region R into the function u. For example for the point x_i we have $u(x_i)$. Also we define $\Delta x = x_{i+1} - x_i$.

Using Taylor expansion in a general point x we can write:

$$u(x + \Delta x) = u(x) + u'(x)\Delta x + \frac{u''(x)(\Delta x)^2}{2} \tag{2.43}$$

$$u(x-\Delta x) = u(x) - u'(x)\Delta x + \frac{u''(x)(\Delta x)^2}{2}. \tag{2.44}$$

From the sum of two equations we take

$$u(x+\Delta x) + u(x-\Delta x) = 2u(x) + u''(x)(\Delta x)^2 \tag{2.45}$$

$$\Rightarrow u''(x) = \frac{u(x+\Delta x) - 2u(x) + u(x-\Delta x)}{(\Delta x)^2}. \tag{2.46}$$

If we substitute the expression (2.46) for the second derivative, into (2.3), and

$$\frac{\partial u}{\partial t} = \frac{u_{t+1} - u_t}{\Delta t}$$

for the first derivative, with $\epsilon_u = 1$, then we have

$$\frac{u_{t+1}(x) - u_t(x)}{\Delta t} = D_u \frac{u_{x+1} - 2u_x + u_{x-1}}{\Delta x^2} + f(u,v) \tag{2.47}$$

$$\Rightarrow u_{t+1}(x) = u_t(x) + \Delta t[D_u \frac{u_{x+1} - 2u_x + u_{x-1}}{\Delta x^2} + f(u,v)]. \tag{2.48}$$

The above method which have used, is the widely known Euler explicit method and it used for the numerical integration of partial differential equations.

Chapter 3

Patterns in excitable systems with lateral inhibition

Stimulation of the medium described by reaction diffusion system (2.3), (2.4) results in the formation of different spatio-temporal patterns. If the inhibitor's diffusion is small compared with that of the activator, the stimulation causes the formation of propagating waves. If the inhibitor's diffusion is large stationary spots are formed. Generally we can divide the parameter space of system (2.3), (2.4), into 4 domains ($R1 - R4$) corresponding to four different types of solutions. $R1$ is a domain corresponding to propagating wave solutions (figure 3.3). $R2$ is a domain were the patterns are unstable and the medium returns to the homogeneous state. $R3$ is a domain where stable pulsating spots are observed (figure 3.2**A**). $R4$ is a domain where stationary spots arise (figure 3.2**B**).

3.1 Stability of stationary solutions

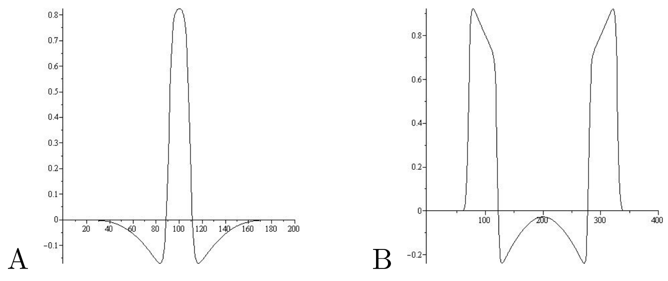

Figure 3.1: The profiles of u. **A**: for a stationary spot, **B**: for a propagating wave (moving right).

To simplify our analysis we replace both u-profiles (figure 3.1), by rectangular functions: $u = 1$ in the excited region and $u = 0$ in the rest of the medium. This approximation can be justified by the fact that $R1 - R4$ solutions are also observed in the system (2.3), (2.4) where $f(u, v)$ is a piecewise linear function (consult [7]) such that u-profiles in corresponding solutions are almost rectangular.

Now we can derive the analytic expression for the v-profile in a stationary spot by solving the linear equation

$$D_v v'' + \epsilon(u - v) = 0 \tag{3.1}$$

when $u = 1$ for $|x| \leq a$ and $u = 0$ for $|x| > a$ with boundary contitions

$$v(-\infty) = v(\infty) = 0 \tag{3.2}$$

assuming that $v \in C^1(-\infty, \infty)$ and $v(-a) = v(a) = v_0$. The solution is a symmetric function

$$v(x) = 1 + (v_0 - 1)\frac{cosh(\lambda x)}{cosh(\lambda a)}, \text{ for } |x| \leq a \tag{3.3}$$

and

$$v(x) = v_0 e^{[\lambda(a-|x|)]}, \text{ for } |x| > a, \tag{3.4}$$

where $\lambda = \sqrt{\epsilon/D_v}$. The requirement that $v'(x)$ is continuous at $|x| = a$, gives the relationship

$$a\lambda = 0.5ln(1 - 2v_0)^{-1}, \tag{3.5}$$

i.e. the product $a\lambda$ is constant. Note that the maximum value of inhibitor, $v(0) = 1 - \sqrt{1 - 2v_0}$, depends only on v_0.

Let us assume that the symmetric variations in the size of the spot and in the value of the inhibitor on the spot's 'wall' affect each other according to the system

$$\frac{da}{dt} = b_{11}\delta a + b_{12}\delta v \tag{3.6}$$

$$\frac{dv}{dt} = b_{21}\delta a + b_{22}\delta v. \tag{3.7}$$

The stationary spot is stable if for the Jacobian matrix B, associated with this system, $det(B) > 0$ and $tr(B) < 0$. Since $D_v = D_3$ corresponds to the stationary-pulsating spot transition, this bifurcation point is associated with

$$tr(B) = b_{11} + b_{22} = 0. \tag{3.8}$$

For b_{11}, we can write

$$b_{11} = c'(v_0)v'(a) \tag{3.9}$$

where $c(v)$ is a velocity of the spot's 'wall' as a function of the inhibitor concentration. The derivative $c'(v)$ is negative (c increases with a decrease of v, more informations can be found in [5]) and its exact value can be found from (2.3). Further,

$$v'(a) = -\lambda v_0 \text{ and thus } b_{11} = \lambda v_0 |c'(v_0)|. \tag{3.10}$$

It is easy to see that

$$b_{22} = D_v \Delta - \epsilon, \tag{3.11}$$

where Δ is the Laplasian operator.

The spatial distribution of the perturbation of v can be represented as a series

$$\sum_{i=0}^{\infty} c_k cos(kx), \tag{3.12}$$

which gives for mode k, $\Delta = -k^2$. Therefore the condition for each mode to become unstable is

$$tr(B) = \lambda v_0 |c_v| - D_v k^2 - \epsilon > 0. \tag{3.13}$$

This indicates that the most sensitive mode corresponds to $k = 0$ or when $\delta v(x) = \delta v$ is a constant. Finally for the bifurcation point D_3 we have $tr(B) = \lambda v_0 |c_v| - \epsilon = 0$ or taking into account that $\lambda = \sqrt{\epsilon / D_v}$:

$$D_3 = (c_v v_0)^2 \epsilon^{-1}, \tag{3.14}$$

i.e., D_3 is proportional to ϵ^{-1}. This confirms results obtained in [8] and [9] where more complicated methods were used to analyse more general systems. To analyse the condition $det(B) > 0$ we shall find expressions for b_{12} and b_{21}. It is easy to see that $b_{12} = c'(v_0)$ and the expression for $b_{21} = \epsilon(v_0(a + \delta a) - v(a + \delta a))$, i.e., b_{21} depends on the difference between the value of v at the front of stationary spot with size $a + \delta a$, which is

$$v_0(a + \delta a) = v_0 + (1 - 2v_0)\lambda \delta a, \tag{3.15}$$

and the value of v at distance δa from the front of stationary spot of size a,

$$v(a + \delta a) = v_0(1 - \lambda \delta a). \tag{3.16}$$

Thus $b_{21} = \epsilon(1 - v_0)\lambda$ and the condition $det(B) = b_{11}b_{22} - b_{12}b_{21} > 0$ trasforms into $2v_0 < 1$. The sign of $det(B)$, which is positive if $v_0 < 0.5$, does not depend on D_v and ϵ.

Example of stationary and pulsating spots

From the Fitz-Hugh Nagumo system (2.5), (2.6), for different values of the kinetics terms and of the diffusion coefficients, we have different kinds of one dimensional solutions (different patterns forms). In this example we have used the following values (see Init procedure in chapter 4), to produce pulsating and stationary spots. To create a pulsating spot (figure 3.2**A**) we have used space-step = 0.16, time-step = 0.02, $D_u = 1$, $D_v = 3.2$, $k_u = 4.5$, $k_v = 1$, $u_1 = 1$, $u_0 = 0.05$, $\epsilon = 1$, and size of the plot 200×201. For the time-space plot of the stationary spot (figure 3.2**B**), we have used space-step = 0.16, time-step = 0.01, $D_u = 1$, $D_v = 5$, $k_u = 4.5$, $k_v = 1$, $u_1 = 1$, $u_0 = 0.05$, $\epsilon = 1$, and size of the plot 400×500.

A B

Figure 3.2: Time-space plot of one dimensinal solutions of the reaction-diffusion system (2.3), (2.4). **A**: pulsating spot, **B**: stationary spot.

3.2 Existence of wave solution

We assume now that $D_v = D_1$, the v-diffusion coefficient for propagating waves. To estimate D_1 we shall find conditions when the system (2.3), (2.4) has a travelling wave solution. In a similar way to the case of stationary spot we replace the u-profile (see figure 3.1**B**), by the rectangular wave and look for a solution $v(x-ct) = v(z)$ to the equation $D_v v'' + cv' + \epsilon(u-v) = 0$ under the same conditions as for the stationary spot, except for $v(a) = v^+$ and $v(-a) = v^-$. The solution is

$$v(z) = v_0 e^{[\lambda_1 (z-a)]} \text{ for } z > a, \tag{3.17}$$

$$v(z) = v_1 e^{[\lambda_2 (z+a)]} \text{ for } z < -a \tag{3.18}$$

and

$$v(z) = 1 - \lambda_2 \frac{e^{[\lambda_1 (z+a)]}}{(\lambda_2 - \lambda_1)} + \lambda_1 \frac{e^{[\lambda_1 (z-a)]}}{(\lambda_2 - \lambda_1)} \text{ for } |z| \leq a \tag{3.19}$$

where

$$\lambda_{1,2} = \frac{(-c \mp \sqrt{c^2 + 4\epsilon D_v})}{(2D_v)}. \tag{3.20}$$

Parameters c and a are defined by v^+ and v^- according to the following equations:

$$v^+ = \lambda_2 [1 - \frac{e^{(2\lambda_1 a)]}}{(\lambda_2 - \lambda_1)} \tag{3.21}$$

and

$$v^- = \lambda_1 [\frac{e^{(-2\lambda_2 a)]}}{(\lambda_2 - \lambda_1)}. \tag{3.22}$$

Note that the size a, of the wave, must be big enough for $v^- > v_0$. To find conditions for the existence of a propagating wave solution we rewrite the expression for v^- as

$$v^- \frac{(\lambda_2 - \lambda_1)}{\lambda_1} + 1 = e^{(-2\lambda_2 a)}. \tag{3.23}$$

Since $0 < e^{(-2\lambda_2 a)} < 1$, we conclude $-1 < v^-(\lambda_2 - \lambda_1)/\lambda_1 < 0$. From the last inequality, after putting the expressions for λ_1 and λ_2 we take

$$D_v \epsilon < c^2 v^- \frac{(1 - v^-)}{(2v^- - 1)^2}. \tag{3.24}$$

Since the velocity c, of the wave, can vary in a limited range and $c = O(1)$ (consult [5]) and $v^- = O(1)$, we can expect that:

$$D_1 \propto \epsilon^{-1}. \tag{3.25}$$

This result was also obtained in [8] where a propagating wave solution of (2.3), (2.4) was studied using analogies with mathematical systems. Our numerical results confirm both contitions (3.14), (3.25).

Example of propagating wave

For this example we have used the Fitz-Hugh Nagumo system (2.5), (2.6). The one-dimensional solution of this system represented by the follow time-space plot, where vertical axis is the time and horizontal axis is the space.

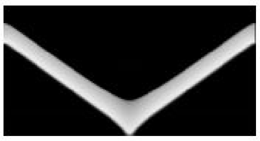

Figure 3.3: Time-space plot of propagating waves.

A set of numerical simulations carried out with various grid sizes and time steps showed that to achieve continuity and accuracy of the solution (while $D_u = 1$) we can set the value of grid size to 0.4 and the time step to 0.01 (or less when $D_v < 4$). For this time-space plot (figure 3.3), we have used space-step = 0.16, time-step = 0.01, $D_u = 1$, $D_v = 1$, $k_u = 4.5$, $k_v = 1$, $u_1 = 1$, $u_0 = 0.05$, $\epsilon = 1$, and size of the plot 400×210 (see Init procedure in chapter 4).

Chapter 4

Maple work

4.1 Useful procedures

LinearDifferentialSystem procedure:

With this procedure we can input the real coefficients a,b,c,d of the system

$$\frac{dx}{dt} = ax(t) + by(t), \quad \frac{dy}{dt} = cx(t) + dy(t).$$

The programme output a plot of the solution of the differential system with initial contitions $x(0) = i$, $y(0) = j$, where $-2 \leq i, j \leq 2$. We used this procedure in section 1.1 to plot the figure 1.2 showing the phase-portraits for a variety of the solutions of the linear system (1.3), (1.4).

```
> with(DEtools):
> #-----------------------------------------------------------------
# Name: LinearDifferentialSystem:=proc(A::realcons,B::realcons,
C::realcons,D::realcons)
# Arguments:
#    A - Coefficient of function x in the first ode.
#    B - Coefficient of function y in the first ode.
#    C - Coefficient of function x in the second ode.
#    D - Coefficient of function y in the second ode.
# Output:
#    A plot of the ode system for a selection of initial
#    conditions and parameters described above.
#-----------------------------------------------------------------

LinearDifferentialSystem:=proc(A::realcons, B::realcons ,
 C::realcons ,D::realcons)

    #Declare the variables
```

```
   local i, j, ode_system, initial_conditions;

    #Setup the ode system given the parameters passed

  ode_system:=[diff(x(t),t)=A*x(t)+B*y(t), diff(y(t),t)=C*x(t)+D*y(t)];
    initial_conditions :=[];

    #Rather than repeatedly call DEplot, we can give it a list of
    # initial conditions, which it will plot for us

   for i from -2 to 2 do
        for j from -2 to 2 do
            initial_conditions := [op(initial_conditions),[x(0)=i,y(0)=j]];
        end do;
    end do;

    #Now call DEplot with the selection of initial conditions and
    # display the results

 DEplot(ode_system,[x(t),y(t)],t=-1..1,x=-10..10,y=-10..10,
color=magnitude,arrows=curve, initial_conditions,  linecolor=black);

 end proc;
```

DetermineTheSolution procedure:

In the procedure DetermineTheSolution we input the coefficients of the system

$$\frac{dx}{dt} = ax(t) + by(t), \quad \frac{dy}{dt} = cx(t) + dy(t)$$

and the programme gives the two eigenvalues of the Jacobian matrix of the coefficients and the kind of the solution of the system, which depends from the kind of the eigenvalues (all the cases described in section 1.1).

```
> with(LinearAlgebra):
> #----------------------------------------------------------------
#Name:DetermineTheSolution:=proc(A::realcons,B::realcons,
    C::realcons,D::realcons)
#Arguments:
#    A - Coefficient of function x in the first ode.
#    B - Coefficient of function y in the first ode.
#    C - Coefficient of function x in the second ode.
#    D - Coefficient of function y in the second ode.
# Output:
# A list with the two eigenvalues and the kind of solution of the
 differential system
#----------------------------------------------------------------------
DetermineTheSolution:=proc(A::realcons,B::realcons,C::realcons,D::realcons)

  #Declare the variables
  local R1,R2,I1,I2,J,l,ode_system;
  l:=list[];

  #The form of the system with the input parameters
  ode_system:=[diff(x(t),t)=A*x(t)+B*y(t), diff(y(t),t)=C*x(t)+D*y(t)];

  #The Jacobian matrix to find the eigenvalues
  J:=Matrix([[A,B],[C,D]]);
  l:=Eigenvalues(J,output='list');

  #Get the real and imaginary parts of the eigenvalues
  R1 := evalf(Re(l[1]));
  R2 := evalf(Re(l[2]));
  I1 := evalf(Im(l[1]));
  I2 := evalf(Im(l[2]));
  print(l);

  #the kind of eigenvalues determine the kind of solutions
  if I1=0 and I2=0 then #Non-Complex
    if (R1>0) and (R2>0) then
      print("2 Real And Distinct positive Eigenvalues :
    Solution = UNSTABLE NODE");
    elif (R1<=0) and (R2<=0) then
      print("2 Real And Distinct negative Eigenvalues:
     Solution = STABLE NODE");
    elif (R1>0 and R2<0) or (R1<0 and R2>0) then
   print("2 Real Distinct Eigenvalues with different signs:
     Solution = SADDLE POINT");
    end if;
  elif I1<>0 or I2<>0 then #Complex
    if R1=0 and R2=0 then
      print("2 Complex Eigenvalues with Real Parts=0:
     Solution = CENTRE");
    elif R1>0 and R2>0 then
      print("2 Complex Eigenvalues with Real Parts >0:
      Solution = UNSTABLE SPIRAL");
    elif R1<0 and Re2<0 then
```

```
      print("2 Complex Eigenvalues with Real Parts < 0:
     Solution = STABLE SPIRAL");
    end if;
  else
    print("2 Real Non-Distinct Eigenvalues: Solution=NODE");
  end if;
end proc;
```

GeneralSolutionDifferentialSystem procedure:

With the procedure GeneralSolutionDifferentialSystem, we can input the coefficients of the linear differential system (1.3), (1.4), from the section 1.1, and the values of the initial contitions at $t = 0$ and the programme output the general solution of this system.

```
> with(DEtools):
> #---------------------------------------------------
#Name:GeneralSolutionDifferentialSystem:=proc(A::ralcons,B::realcons,
C::realcons,D::realcons,i::realcons,j::realcons)
# Arguments:
#    A - Coefficient of function x in the first ode.
#    B - Coefficient of function y in the first ode.
#    C - Coefficient of function x in the second ode.
#    D - Coefficient of function y in the second ode.
#    i - Value of function x if the time t=0.
#    j - Value of function y if the time t=0.
# Output:
#    The general solution of the ode system for the input
#    coefficients with the input values of initial contitions.
#---------------------------------------------------
GeneralSolusionDifferentialSystem:=proc(A::realcons,B::realcons,
C::realcons,D::realcons,i::realcons,j::realcons)
# Declare the variables

local ode_system,initial_contitions,general_solution;

# Setup the ode system and the initial contitions given the parameters
 passed

ode_system:={diff(x(t),t)=A*x(t)+B*y(t), diff(y(t),t)=C*x(t)+D*y(t)};
initial_contitions:={x(0)=i,y(0)=j};

# With Detools the programme output the general solution
of the ode system using the command dsove
```

```
general_solution:=evalf(dsolve(ode_system union
            initial_contitions,{x(t),y(t)}));

 end proc;
```

Init-Step-Draw and Run procedures

1. This is a more complicated programme which used to draw the time-space plots of the patterns (see figures 3.2 and 3.3). With the Init procedure we introduce the parameters and initial conditions for the different figures. Moreover this procedure set the image contents to black and initiate the pattern by stimulating a small area in the middle of the medium. The Step procedure calculate the diffusive terms for the PDEs and calculate u and v at point x. These calculations used for the numerical integration of partial differential equations as we have seen in section 2.5. The Draw procedure output the state of the system at time t, which is a column. Finally Run procedure calls repeatedly the Step and Run procedures, and effectively the programme solving and drawing the solution and output the time-space plots of the patterns.

```
> #------------------------------------------------------
# Reference all essential libraries here, and declare any global
# variables required for the simulation.
#------------------------------------------------------
# MediumWidth:            #The width of the stimulation medium
# MediumHeigth:           #The heigth of the stimulation medium
# OutputImage:            #The image that will finaly hold the simulation
# U:                      #The current concentration for the U variable
                                                         (1D array)
# V:                      #The current concentration for the V variable
                                                        (1D array)
# hU:                     #THe current calculation state of U chemical
# hV:                     #The current calculation state of V chemical
# TimeStep::float:=0.0:    #The temporal increment between simulation steps
# kU::float:=0.0:
# u0::float:=0.0:
# kV::float:=0.0:
# u1::float:=0.0:
# URestState::float:=0.0:#The chemical concentration before the simulatin
                                              begins for U
# VRestState::float:=0.0:#The chemical concentration before the simulatin
                                              begins for V
# UStimulus::float:=0.0: #A small increase in chemical concentration to
                                            initiate the simulation
```

```
# SpaceStep::float:=0.0: #The spatial increment for calculating chemicals
# RelaxTime::float:=0.0:
# Du::float:=0.0:        #The diffusion coefficient for u chemical
# Dv::float:=0.0:        #The diffusion coefficient for v chemical
#--------------------------------------------------------------------

restart;
with(ImageTools):
With(LinearAlgebra):
> #--------------------------------------------------------------
#Name:Init:=proc(figure)
#Arguments:
#figure:The figure we wish to reproduce. For different
#values of parameters we take different kinds of patterns.
#Effect:
#This procedure must be called before run with an appropriate figure
 choice, to ensure the parameters, initial contitions and data
structures are create\define correctly
#--------------------------------------------------------
Init:=proc(figure)

#First we make references to global variables so we can initialize them

global
kU,kV,u0,u1,TimeStep,SpaceStep,RelaxTime, URestState, VRestState,
 UStimulus,UDiffusion, VDiffusion, Du, Dv,
OutputImage, MediumWidth, MediumHeight, U, hU, V, hV;

#Now we define some variables we will need for this procedure

local x::nonnegint,t::nonnegint,i::nonnegint;

#We except the user to call this procedure for a simulation type,
so we can initialize the parameters and initial contitions correctly

if figure = fig1a then
print("figureA: propagating wave");
 TimeStep := 0.01;
 u0 := 0.05;
 u1 := 1.0;
 kU := 4.5;
 kV := 1.0;
 URestState := 0.1;
 VRestState := 0.1;
 UStimulus := 0.29;
 MediumWidth := 400;
 MediumHeight := 210;
 SpaceStep := 0.16;
 RelaxTime := 10.0;
 UDiffusion := 1.0;
 VDiffusion := 1.0;
 Du := 1.0;
 Dv := 1.0;
elif
figure = fig1b then
print("figureB: pulsating spot");
 TimeStep := 0.02;
 u0 := 0.05;
 u1 := 1.0;
 kU := 4.5;
 kV := 1.0;
 URestState := 0.1;
 VRestState := 0.1;
 UStimulus := 0.5;
 MediumWidth := 200;
 MediumHeight := 201;
 SpaceStep := 0.16;
 RelaxTime := 10.0;
 UDiffusion := 1.0;
 VDiffusion := 1.0;
 Du := 1.0;
 Dv := 3.2;
elif
figure = fig1c then
print("figureC: stationary spot");
 TimeStep := 0.01;
 u0 := 0.05;
 u1 := 1.0;
 kU := 4.5;
 kV := 1.0;
 URestState := 0.1;
 VRestState := 0.1;
 UStimulus := 0.5;
 MediumWidth := 400;
 MediumHeight := 600;
 SpaceStep := 0.16;
 RelaxTime := 10.0;
 UDiffusion := 1.0;
 VDiffusion := 1.0;
 Du := 1.0;
 Dv := 5;
else
print("Unkown Initialization For Init(figure)!");
end if;

#Given that above choice, we now create the data structures
for calculating the Fitz-Hugh-Nagumo equations

U:=Vector(MediumWidth,URestState,datatype=float);
V:=Vector(MediumWidth,VRestState,datatype=float);
hU:=Vector(MediumWidth,0.0,datatype=float);
hV:=Vector(MediumWidth,0.0,datatype=float);
OutputImage:= Create(MediumWidth,MediumHeight):

#Set the image contents to black

for x from 1 to MediumWidth do
       for t from 1 to MediumHeight do
           OutputImage[x,t]:=0;
      end do;
   end do;

#Initiate the pattern by simulating a small area in the middle of the medium

 for i from MediumWidth/(2)-5 to MediumWidth/(2)+5 do
       U[i]:=UStimulus;
      hU[i]:=UStimulus;
  end do;
   end proc;
> #----------------------------------------------------------------------
#Name: Step:=proc()
#Arguments:
# None
#Effect:
#This scheme employs centered space differencing scheme
that systematically calculates the state of the system forward
in time. Thus this procedure is called many times to converge
on the solution, given initial parameters and contitions, by
solving the current state of the system at time t, to produce
the new state at time t+1
#--------------------------------------------------------------------
```

```
 Step:=proc()

#First we make references to global variables so we can initialize them

global
U,V,hU,hV,MediumWidth,TimeStep,SpaceStep,
RelaxTime,UDiffusion,VDiffusion,Du,Dv;

#Now define some variables we will need for this procedure

 local
 x::nonnegint:=0,t::nonnegint:=0,
curu::float:=0.0,curv::float:=0.0;

#Now solve the equations for the current time step by iterating
 over the medium left to right

 for x from 2 to MediumWidth-1 do
             curu:=U[x];
    curv:=V[x];

#Calculate the diffusive terms for the PDEs

 UDiffusion := evalf((U[x+1]+U[x-1]-2.0*curu)/SpaceStep);
 VDiffusion := evalf((V[x+1]+V[x-1]-2.0*curv)/SpaceStep);

#Now calculate the PDE with the above diffusive term at point x
    hU[x]:=curu+TimeStep*(Du*UDiffusion-(kU*curu*(curu-u0)*(curu-u1)+kV*curv));
           hV[x]:=curv+TimeStep*(Dv*VDiffusion+(curu-curv)/RelaxTime);

end do;

#Now set Boundary contitions for the medium

 hU[1]:=hU[2];
 hU[MediumWidth-1]:=hU[MediumWidth-2];
 hV[1]:=hV[2];
 hV[MediumWidth-1]:=hV[MediumWidth-2];

#Now uptade the medium for the next step

   for x from 1 to MediumWidth do
       U[x]:=hU[x];
       V[x]:=hV[x];
```

```
        end do;
end proc;
> #-------------------------------------------------------------
#Name: Draw:=proc(t::nonnegint)
#Arguments:
# t-The current column of the t axis we drawing
#Effect:
# To draw the state of the system at time t, which is effectively
a column in the output image
#-------------------------------------------------------------

Draw:=proc(t::nonnegint)

#First we make references to global variables so we can initialize them

 global   U,V,OutputImage,MediumWidth;


#Now we define some variables we will need for this procedure

 local    x;

#Copy the current solution to the output image

for x from 1 to MediumWidth do
      OutputImage[x,t]:=U[x];
  end do;
end proc;
> #------------------------------------------------------------------
#Name: Run:=proc(endtime::float)
#Arguments:
# endtime: The time at which the simulation stop
#Effects:
# The main iteration for the simulation that repeatedly calls the
Step and Draw procedure, effectively solving and drawing the solution
#----------------------------------------------------------------

Run:=proc(endtime::float)

#First we make references to global variables so we can
initialize them

global TimeStep;
```

```
#Now define some variables we will need for this procedure

local t::float:=0.0, step::integer:=0, vb::integer:=1;

#Iterate the Step function until the endtime has been reached

while t<=endtime do
  Step();
 if  step mod 50 =0 then
 Draw(vb);
 vb:=vb+1;
end if;

#Increace time by prescrived step

t:=t+TimeStep;
step:=step+1;
end do;
end proc;

> Init(fig1a);
Run(200.0);
Preview(OutputImage);
```

2. The next procedure is very similar with the previous Init-Step-Draw-Run procedure with some changes. We use the next procedure to draw the u-profile (figure 3.1), for the patterns at the time t in the middle of stimulation. The difference from the previous procedure is that the Run procedure stop at time $t = EndTimeStep/2$, where is the middle of stimulation and using the command

   ```
   > with(plots):
   ```

   the programme output the plot of u concentration profile against x.

   ```
   > restart;
   with(plots):
   with(ImageTools):
   with(LinearAlgebra):
   > Init:=proc(figure)

   local x::nonnegint,t::nonnegint,i::nonnegint,
   HalfMediumWidth::nonnegint;
   ```

```
global kV,kU,u0,u1,gDrawStep,gEndTimeStep,gTimeStep,
gSpaceStep,gRelaxTime,gURestState,gVRestState,
gUStimulus,gUDiffusion,gVDiffusion,gUDiffusionCoeff,
gVDiffusionCoeff,gOutputImage,gMediumWidth,
gMediumHeight, gU, ghU, gV, ghV;

if figure = fig1a then
print("Initializing For Fig1A");
gDrawStep := 50;
gSpaceStep := 0.16;
gEndTimeStep := 100;
gTimeStep := 0.1;
kV := 1.0;
u0 := 0.5;
kU := 4.5;
u1 := 1.0;
gURestState := 0.1;
gVRestState := 0.1;
gUStimulus := 0.29;
gMediumWidth := 400;
gMediumHeight := 210;
gRelaxTime := 10.0;
gUDiffusionCoeff := 1.0;
gVDiffusionCoeff := 1.0
elif
figure = fig1b then
print("Initializing For Fig1B");
gDrawStep := 100;
gSpaceStep := 0.16;
gEndTimeStep := 400;
gTimeStep := 0.2;
kV := 1.0;
u0 := 0.5;
kU := 4.5;
u1 := 1.0;
gURestState := 0.1;
gVRestState := 0.1;
gUStimulus := 0.5;
gMediumWidth := 200;
gMediumHeight := 201;
gRelaxTime := 10.0;
gUDiffusionCoeff := 1.0;
gVDiffusionCoeff := 3.2
else
```

```
print("Unkown Initialization For Init(figure)!")
end if;

gU:=Vector(gMediumWidth,gURestState,datatype=float);
gV:=Vector(gMediumWidth,gVRestState,datatype=float);
ghU:=Vector(gMediumWidth,0.0,datatype=float);
ghV:=Vector(gMediumWidth,0.0,datatype=float);
gOutputImage:=Create(gMediumWidth, gMediumHeight);

for x to gMediumWidth do
for t to gMediumHeight do
gOutputImage[x,t]:= 0;
end do;
end do;
HalfMediumWidth:=(1/2)*gMediumWidth;

for i from HalfMediumWidth-5 to HalfMediumWidth+5 do
gU[i]:=gUStimulus;
ghU[i]:=gUStimulus
end do;

end proc;
> Step := proc ()

local x::nonnegint,t::nonnegint,UDiffusion::float,
VDiffusion::float,curu::float,curv::float;

global gU,gV,ghU,ghV,gMediumWidth,gTimeStep,
gSpaceStep,gRelaxTime,gUDiffusionCoeff,
gVDiffusionCoeff;x::nonnegint:=0;t::nonnegint:=0;
UDiffusion::float:=0.0;VDiffusion::float:=0.0;
curu::float := 0.0; curv::float:= 0.0;

for x from 2 to gMediumWidth-1 do
curu := gU[x]; curv := gV[x];
UDiffusion:=evalf((gU[x+1]+gU[x-1]+(-1)*2.0*curu)/gSpaceStep);
VDiffusion:=evalf((gV[x+1]+gV[x-1]+(-1)*2.0*curv)/gSpaceStep);
ghU[x]:=curu+gTimeStep*(gUDiffusionCoeff*UDiffusion-
kU*curu*(curu-u0)*(curu-u1)-kV*curv);
ghV[x] := curv+gTimeStep*(gVDiffusionCoeff*VDiffusion+
(curu-curv)/gRelaxTime);
end do;

ghU[1] := ghU[2];
```

```
ghU[gMediumWidth-1]:=ghU[gMediumWidth-2];
ghV[1] := ghV[2];
ghV[gMediumWidth-1]:=ghV[gMediumWidth-2];

for x to gMediumWidth do
gU[x] := ghU[x];
gV[x] := ghV[x];
end do;
end proc;
> Run := proc (figuretoplot)

local t::float,step::integer, vb::integer;

global gTimeStep,gEndTimeStep,gDrawStep;t::float:=0.0;
step::integer:=0;vb::integer:=1; Init(figuretoplot);

while t <= (1/2)*gEndTimeStep do
Step();
if step mod gDrawStep=0 then
Draw(vb);
vb := vb+1;
end if;

t := t+gTimeStep;
step := step+1;
end do;
end proc;

> Run(fig1a);
Preview(gOutputImage);
gPlot := [];
for x to gMediumWidth do
 gPlot := [op(gPlot), [x, gU[x]]];
end do;
pointplot(gPlot, connect = true);
```

4.2 Simple work on Maple

Phase-portraits for one linear ordinary differential equation:

Using the package DEtools on Maple we can plot the phase-portraits of linear differential equation (1.1) (see figure 1.1 in section 1.1. For example, for the figure 1.1**A** we have used the values $a = -0.1$, $b = 2$ and the initial conditions $x(0) = 30$, $x(0) = 10$. For figure 1.1**B** the values $a = 0.1$, $b = 2$ and the initial conditions $x(0) = -40,\ x(0) = 10$.

```
> with(DEtools):
> OneLinearEquation := diff(x(t),t) = a*x(t)+b;
  a:=-0.1; b:=2;
> ivs := [x(0)=30,x(0)=10];
> DEplot( OneLinearEquation, x(t), t=0..20,
 ivs,linecolour=[black,black],title=A);

> with(DEtools):
> OneLinearEquation := diff(x(t),t) = a*x(t)+b;
   a:=0.1;  b:=2;
> ivs := [x(0) = -40, x(0) = 10];
> DEplot(OneLinearEquation, x(t), t=0..20,
ivs,linecolour=[black,black],title=B );
```

Plots of $f(x)$ for nonlinear differential equations

Using the command with(plots) Maple can output the plots of functions $f(x)$ for a nonlinear equation $\dot{x} = f(x)$. In particular for the figure 1.4 I used the function $f(x) = x(x-1)$ for the Fisher equation and the function $f(x) = -x(x-x_0)(x-x_1)$ with $x_0 = 1, x_1 = 3$ for the Fitz-Hugh equation.

```
> with(plots):
> f:=x*(x-1);
> plot(f,x=-1..2);

> with(plots):
> f:=-x*(x-1)*(x-3);
> plot(f,x=-1..3.5);
```

FitzHughNullclines procedure

With the FitzHugNullcines procedure we input the constants a, b, c and the programme output the plot of the nullclines for one nonlinear equation (1.25) and one linear equation (1.26). We use the output plots of this procedure, in chapter 1 in the example of section 1.3 to find the steady state point of the nullclines and to determine the kind of the system.

```
> with(plots):
> #----------------------------------------------------
#Name:FitzHughNullcines:proc(A::realcons,B::realcons,
           C::realcons)
#Arguments:
# A-Constant in the nonlinear function
# B,C-Constants in the linear function
#Output:
# A plot of the nullclines of a system with one
    nonlinear
 (cubic form) and one linear equation
#(Fitz-Hugh-Nagumo nullclines)
#-------------------------------------------------------
FitzHughNullclines:=proc(A::realcons,B::realcons,
   C::realcons)

#Declare the variables
local nullclines;

#The form of the nullclines
nullclines:=[-x*(x-A)*(x-1)-y=0,B*(x-y)+C=0];

implicitplot(nullclines,x=-5..5,y=-5..5);
end proc;
```

Bibliography

[1] J.D.Murray, Mathematical Biology **19** 767.

[2] L.Edelstein-Keshet, Mathematical Models in Biology (2005) 586.

[3] F.H.Busse, S.C.Muller, Evolution of spontaneous Structures in Dissipative Continuous systems, Berlin, 1998.

[4] M.A.J.Chapain, G.D.Singh, J.C.McLachlan, On Growth and Form: Spatio-Temporal Pattern Formation in Biology, Chichester, 1999

[5] R. Fitzhugh, in: H.P.Schwan, Biological Engineering, New York, 1969.

[6] H.Meinhardt, A.Giere, Bioessays **22** (2000) 753.

[7] B.Vasiev, A.Panfilov, R.Khramov, Physics Letters A **192**, (1994) 227.

[8] B.S.Kerner, V.V.Osipov, Sov. Microelectronics **12** (1983) 512.

[9] Y.Nishiura, M.Mimura, SIAM J. Appl. Math. **49** (1989) 481.