Universiteit Leiden
The Netherlands

Opleiding Wiskunde & Informatica

Numerics and continuation

for Reaction-Diffusion equations

Renzo Baasdam

s1524054

Supervisors:

Martina Chirilus-Bruckner (MI) & Michael Emmerich (LIACS)

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)

`www.liacs.leidenuniv.nl`

15/08/2017

Abstract

Reaction-Diffusion equations are often used as simplified models to study the emergence of patterns in nature. In the first part of this thesis we focus on numerical methods, in particular the forward Euler en fourth-order Runge-Kutta and finite difference methods. After this we study the concepts of continuation of parameters and use the Matlab package **pde2path** to determine patterns of a Reaction-Diffusion equation. Finally, we determine changes to the patterns when adding an inhomogeneous term to the equation.

Contents

Chapter 1

Introduction

In 1952 Alan Turing published an article in which he described model that could generate patterns [Tur52]. Models of the form Turing described are called *Reaction-Diffusion equations* which are *partial differential equations* of the form

$$\partial_t u = D\Delta u + f(u).$$

In this thesis we will look at two different techniques to study these equations, *numerical methods* for initial value problems and a *continuation of parameters* algorithm to determine patterns and determine changes to the found patterns when we add an inhomogeneous term. In Chapter 1 we will introduce two numerical methods for *ordinary differential equations*, the forward Euler method and the fourth order Runge-Kutta method and compare them. In Chapter 2 we will use these two methods to construct numerical methods for Reaction-Diffusion equations with a finite difference method. In Chapter 3, we will introduce the method of *continuation of parameters* to find patterns of a Reaction-Diffusion equation. In Chapter 4, we will use the **Matlab** package **pde2path** [DRUW14] to determine patterns of a RD equation after which we will add an inhomogeneous term. For all simulations in this thesis we will use **Matlab**.

An important theorem that we will use throughout this thesis is Taylor's theorem.

Theorem 1 (Taylor's theorem [RC99])**.** *If a function $u(t)$ has continuous derivatives up to the $(n+1)$th order on a closed interval containing $x, x+h \in \mathbb{R}$ with $h > 0$, then:*

$$u(x+h) = u(x) + hu'(x) + \ldots \frac{h^n}{n!}u^{(n)}(x) + O(h^{n+1}).$$

Proof. See [RC99]. □

1.1 Ordinary differential equations

An *Ordinary differential equation* (ODE) is an equation that depends on some unknown function of one variable and it's derivatives, so it is of the form

$$f(t, u, u', u'', \ldots) = 0$$

where $u = u(t) \in \mathbb{R}^n$ is the unknown function for some $n \in \mathbb{Z}_{>0}$ and $t \in \mathbb{R}$. In this chapter we will only consider first order ODEs of the form

$$u' = f(t, u). \tag{1.1}$$

Often, ODEs have an initial value which is a constraint

$$u(t_0) = u_0 \in \mathbb{R}^n \tag{1.2}$$

with $t_0 \in \mathbb{R}$ the initial time. An ODE together with an initial value is called an *initial value problem* (IVP). For f nonlinear, most solutions of initial value problems do not have an explicit form. But we can approximate these solutions with computers using *numerical methods*. We note that computers can only calculate finitely many values, so we have to choose finitely many points in time (greater than t_0) at which we will approximate u. We call this the *discretization* of time and ensure this by defining a time step $h \in \mathbb{R}$ and a number of steps $N \in \mathbb{N}$. We will now introduce two numerical methods for solving IVPs, the forward Euler method and the Fourth Order Runge-Kutta method.

1.2 (Forward) Euler method for ODEs

The *(forward) Euler method* uses that by Taylor's Theorem (1) we have for u sufficiently smooth

$$u(t + h) = u(t) + hu'(t) + O(h^2) \tag{1.3}$$

for h small enough, thus by ignoring the $O(h^2)$ term we get an estimate

$$u(t + h) \approx u(t) + hu'(t) \tag{1.4}$$

and substitution of (1.1) in (1.4) gives

$$u(t + h) \approx u(t) + hf(u(t), t). \tag{1.5}$$

We note that from (1.3), smaller values of h will generally give a better estimate as the error also decreases. This estimate is useful because it expresses u at time $t + h$ using only terms at time t. Using this estimate iteratively

starting at the initial time t_0, we can determine an approximation for the solution of the IVP.

Algorithm 1: Euler method for first order ODEs

input : $f(u,t), t_0, u_0, h, N$

$u_1, \ldots, u_N \leftarrow 0$;

for $i \leftarrow 0$ **to** $N-1$ **do**

  $u_{i+1} \leftarrow u_i + h \cdot f(u_i,\ t_0 + i \cdot h)$;

As we have left out the $O(h^2)$ term from (1.3) for the Euler approximation, it is clear that the local truncation error (the error of one time step) is of $O(h^2)$. We will now look at a method with a better local truncation error, the fourth order Runge-Kutta method.

1.3 Fourth Order Runge-Kutta for ODEs

The Euler method uses the derivative of u at time t and uses this to determine the value at $t + t_s$. The *fourth order Runge-Kutta method* (RK4) also uses estimates of the derivative of u at time $t + t_s/2$ and $t + t_s$. From [Hol07] a motivation to consider derivatives at other points can be given by Simpson's rule which states that

$$\int_t^{t+h} f(x)dx = \frac{h}{6}\left[f(t) + 4f(t + h/2) + f(t + h)\right] + O(h^5).$$

Thus, if we consider the simplified version $u'(t) = f(t)$ of (1.1), then integrating gives

$$u(t + h) - u(t) = \int_t^{t+h} f(t)dt$$

and then Simpson's rule gives

$$u(t + h) = u(t) + \frac{h}{6}\left[f(t) + 4f(t + h/2) + f(t + h)\right] + O(t^5).$$

The estimate for RK4 of (1.1) is similarly given by

$$u(t + h) \approx u(t) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{1.6}$$

where k_1 is an estimate of $u'(t)$ given by

$$u'(t) = f(u(t),\ t) =: k_1,$$

k_2 is an estimate of $u'(t + h/2)$ given by

$$u'(t + h/2) = f\left(u(t + h/2),\ t + \frac{h}{2}\right)$$

and by using the estimate of the Euler method (1.5) to obtain the estimate $u(t + h/2) \approx u(t) + (h/2)k_1$ which

gives

$$u'(t+h/2) \approx f\left(u(t) + \frac{h}{2}k_1,\ t + \frac{h}{2}\right) =: k_2,$$

k_3 is again an estimate of $u'(t+h/2)$ using the estimate $u(t+h/2) \approx u(t) + (h/2)k_2$ which gives

$$u'(t+h/2) \approx f\left(u(t) + \frac{h}{2}k_2,\ t + \frac{h}{2}\right) =: k_3$$

and k_4 is an estimate of $u'(t+h)$ using $u(t+h) \approx u(t) + hk_3$ which gives

$$u'(t+h) \approx f(u(t) + hk_3,\ t+h) =: k_4.$$

Algorithm 2: Euler method for first order ODEs

input : $f(u,t), t_0, u_0, h, N$

$u_1, \ldots, u_N \leftarrow 0$;

for $i \leftarrow 0$ **to** $N-1$ **do**

$k_1 \leftarrow f(u_i,\ t_0 + i \cdot h)$;

$k_2 \leftarrow f(u_i + (h/2)k_1,\ t_0 + (i+1/2)h)$;

$k_3 \leftarrow f(u_i + (h/2)k_2,\ t_0 + (i+1/2)h)$;

$k_4 \leftarrow f(u_i + hk_3,\ t_0 + (i+1)h)$;

$u_{i+1} \leftarrow u_i + (h/6)(k_1 + 2k_2 + 2k_3 + k_4)$;

1.4 Comparison

Generally, RK4 is generally considered more accurate than the Euler method because it has a local truncation error of $O(h^5)$ against the $O(h^2)$ of the Euler method. We will now consider the ODE

$$u' = -u$$

with initial condition

$$u(0) = 2.$$

Note that this has exact solution $u(t) = 2e^{-t}$. We approximate the solutions using the Euler and RK4 method for the time step $h = 1/2$ and compare them (Fig.1.1) to the exact solution.

We find that RK4 is much more accurate than the Euler method. However, RK4 requires more calculations per time step, therefore we compare the two methods with the Euler method having one fourth the stepsize of RK4 (Fig. 1.2), but we still find that RK4 is more accurate.

Thus, the simulations are in line with the expected result that RK4 is (generally) more accurate than the Euler method.

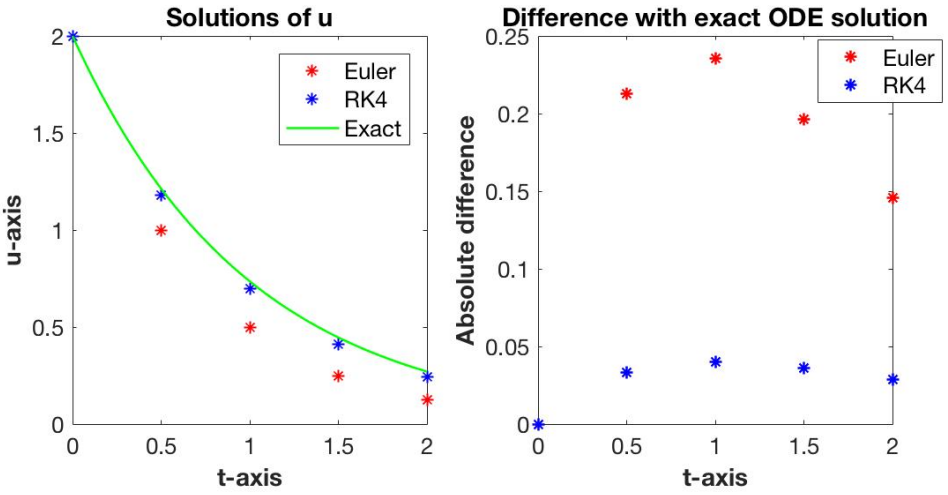

Figure 1.1: *Plots of exact solution and approximations (left) and plots of the absolute difference to the exact solution (right). Both Euler and RK4 have stepsize* $h = 1/2$.

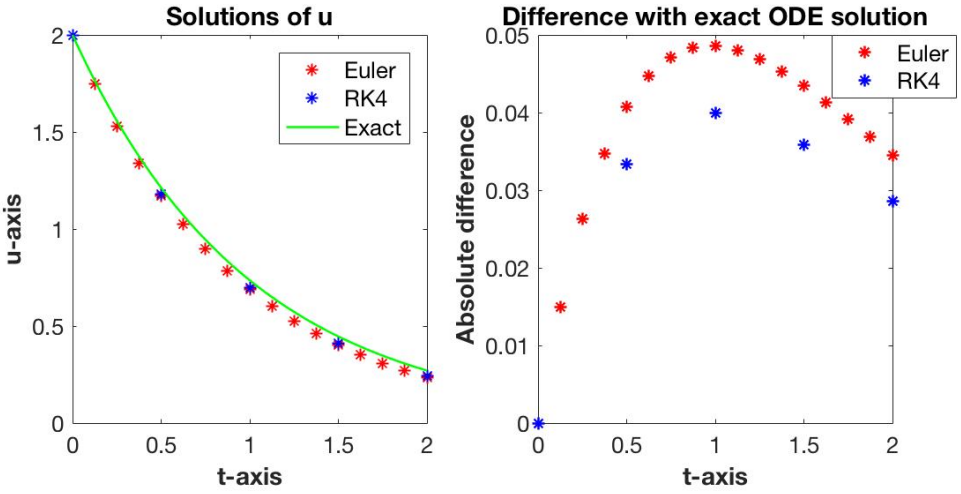

Figure 1.2: *Plots of exact solution and approximations (left) and plots of the absolute difference to the exact solution (right). Euler has stepsize* $h = 1/8$ *and RK4 has stepsize* $h = 1/2$.

Chapter 2

Reaction-Diffusion equations and a finite difference method

2.1 Reaction-Diffusion equations

A *partial differential equation* (PDE) is an equation that depends on some unknown function of more than one variable and its partial derivatives. The difference between ODEs and PDEs is the unknown function, for ODEs $u = u(t) \in \mathbb{R}^n$ is a function of one variable, while for PDEs $u = u(x_1, \ldots, x_m, t) \in \mathbb{R}^n$ a function of multiple variables where $x_1, \ldots x_m \in \mathbb{R}^m$ with $n \in \mathbb{Z}_{>0}$ and $m \in \mathbb{Z}_{>1}$. In particular, a *Reaction-Diffusion equation* (RD equation) is a PDE of the form

$$\partial_t u = \underbrace{D\Delta u}_{\text{Diffusion term}} + \underbrace{f(u)}_{\text{Reaction term}}$$

where D, the diffusion matrix, is a diagonal $n \times n$ matrix, $m \in \{1, 2, 3\}$ and $\Delta u = \sum_{k=1}^{m} \partial_{x_k}^2 u$. In this thesis we will be studying 2-dimensional Reaction-Diffusion equations (n=2) on a 2-dimensional domain (m=2), so of the form

$$\begin{aligned} \partial_t u &= D_1 \Delta u + f_1(u, v), \\ \partial_t v &= D_2 \Delta v + f_2(u, v) \end{aligned} \tag{2.1}$$

with $u = u(x, y, t) \in \mathbb{R}$ and $v = v(x, y, t) \in \mathbb{R}$. We add the constraints that $(x, y) \in U \subset \mathbb{R}^2$ where we call U the *domain* (Fig 2.1)

$$U = (a, b) \times (c, d), \quad \text{with } a, b, c, d \in \mathbb{R},\ a < b,\ c < d,$$

the initial condition

$$u(x, y, t_0) = u_0(x, y) \ \text{ and } \ v(x, y, t_0) = v_0(x, y) \tag{2.2}$$

and the *boundary condition* (homogeneous Neumann)

$$\partial_{\overrightarrow{n}} u(x_b, y_b, t) = 0 \text{ and } \partial_{\overrightarrow{n}} v(x_b, y_b, t) = 0 \tag{2.3}$$

for $(x_b, y_b) \in \partial U, t > t_0$ with $t_0 \in \mathbb{R}$ the initial time and $\overrightarrow{n}$ the normal of the boundary ∂U pointing outwards.

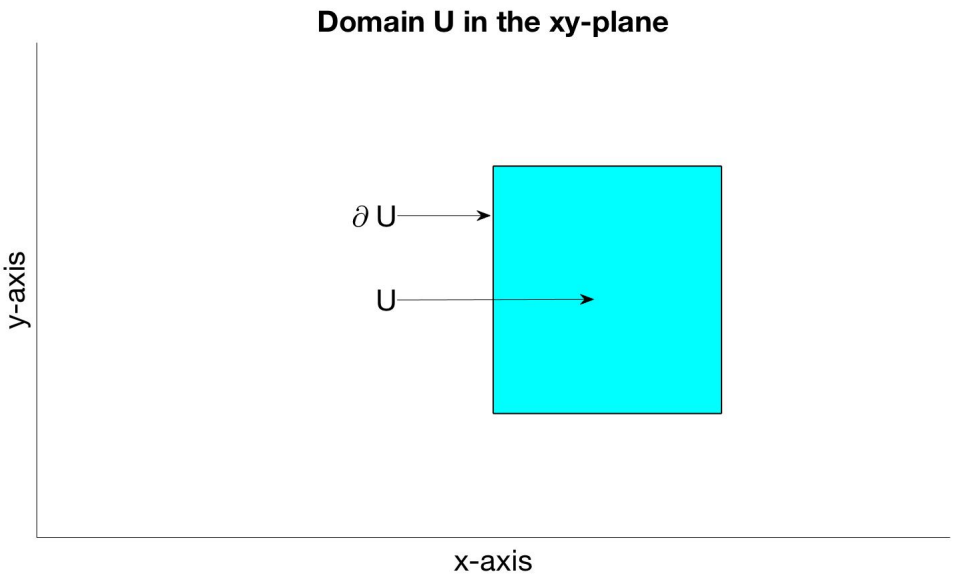

Figure 2.1: *Illustration of domain U.*

Like ODEs, Reaction-Diffusion equations also often don't have explicit solutions (in particular when f is nonlinear). But, we can again approximate RD equations with computers similarly to Chapter 1. To do this we will introduce a *finite-difference method*.

2.2 Discretization of U

The main difference between the unknown function u_O of the ODE (1.1) and u_R of the RDE (2.1) is that for a fixed time $t > t_0$, $u_O(t)$ is a number while $u_R(t)$ is a surface on the domain U. As this surface consists of infinitely many points, we want to discretize the domain so that a computer can calculate with it. Note that there are many ways to discretize, but for the finite difference method we discretize U as a raster. We choose constants $d_x, d_y \in \mathbb{R}$ such that $N_x = \frac{b-a}{d_x} + 1, N_y = \frac{d-c}{d_y} + 1 \in \mathbb{Z}_{>0}$, then we define

$$R = \{(a + d_x k_1, c + d_y k_2) \in \mathbb{R}^2 : k_1, k_2 \in \mathbb{Z}\}$$

and $U_d = U \cap R$ and $\partial U_d = \partial U \cap R$, illustrated in Fig. 2.2. We note that smaller d_x and d_y improve the accuracy of the approximation of u at a fixed time.

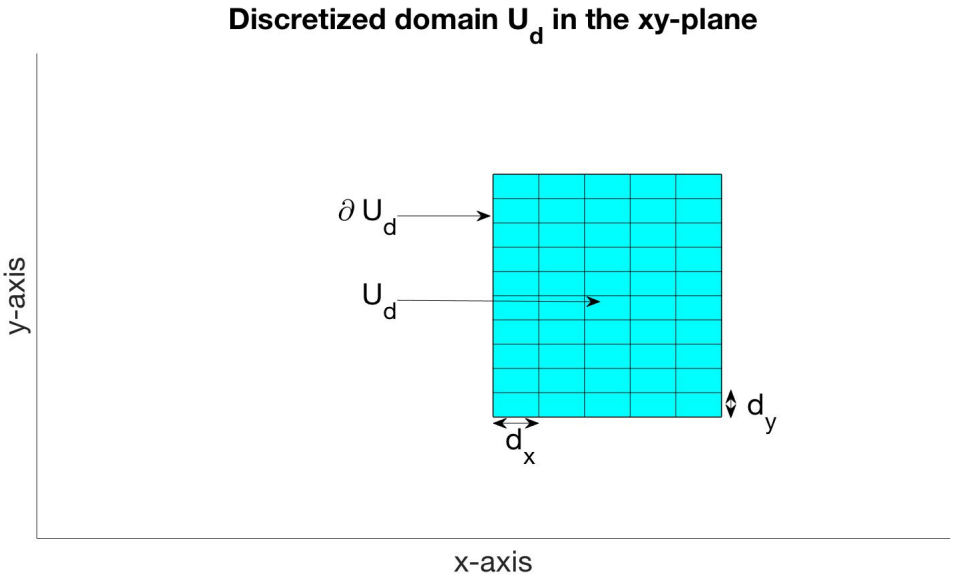

Figure 2.2: *Illustration of domain U_d. Note that U_d only contains the inner vertices of the illustrated area.*

2.3 Approximation of the Laplacian

By Taylor's theorem(1) for sufficiently smooth u we have

$$u(x+d_x,y,t) = u(x,y,t) + d_x\partial_x u(x,y,t) + d_x^2\frac{\partial_x^2 u(x,y,t)}{2} + O\left(d_x^3\right) \tag{2.4}$$

and

$$u(x-d_x,y,t) = u(x,y,t) - d_x\partial_x u(x,y,t) + d_x^2\frac{\partial_x^2 u(x,y,t)}{2} + O\left(d_x^3\right). \tag{2.5}$$

Adding these two equations together we get

$$u(x+d_x,y,t) + u(x-d_x,y,t) = 2u(x,y,t) + d_x^2\partial_x^2 u(x,y,t) + O\left(d_x^4\right),$$

thus by ignoring the $O(d_x^4)$ term and rewriting the equation we get

$$\partial_x^2 u(x,y,t) \approx \frac{u(x+d_x,y,t) - 2u(x,y,t) + u(x-d_x,y,t)}{d_x^2}.$$

Note that we can do exactly the same for y and d_y which together gives the approximation

$$\begin{aligned}\Delta u(x,y,t) \approx &\frac{u(x+d_x,y,t) - 2u(x,y,t) + u(x-d_x,y,t)}{d_x^2} \\ &+ \frac{u(x,y+d_y,t) - 2u(x,y,t) + u(x,y-d_y,t)}{d_y^2}.\end{aligned} \tag{2.6}$$

This approximation for the Laplacian is useful, because it only consists of terms of u at the current time t.

2.4 Approximation for the boundary conditions

For $(x_b, y_b) \in \partial U_d$, we have the boundary condition (2.3). Let $x_b = a$, the left boundary of U_d and assume that (x_b, y_b) is not a corner, i.e. $y_b \neq c, d$. We introduce the imaginary unknown $u(x_b - d_x, y_b, t)$ [LeV07], then by subtracting (2.5) from (2.4) we get

$$\partial_x u(x_b, y_b, t) - u(x_b - d_x, y_b, t) = 2d_x \partial_x u(x_b, y_b, t) + O(d_x^3)$$

which gives

$$\partial_x u(x_b, y_b, t) \approx \frac{u(x_b + d_x, y_b, t) - u(x_b - d_x, y_b, t)}{2d_x} \tag{2.7}$$

and so the boundary condition gives

$$u(x_b + d_x, y_b, t) = u(x_b - d_x, y_b, t). \tag{2.8}$$

Thus, with this unknown we can also approximate the laplacian in (x_b, y_b) by substituting (2.8) in (2.6) we get

$$\begin{aligned} \Delta u(x_b, y_b, t) \approx & \frac{2u(x + d_x, y, t) - 2u(x, y, t)}{d_x^2} \\ & + \frac{u(x, y + d_y, t) - 2u(x, y, t) + u(x, y - d_y, t)}{d_y^2}. \end{aligned} \tag{2.9}$$

which does not contain the imaginary unknown any more. We can find similar approximations for the other boundaries and the corners.

2.5 Discretized PDE as a system of ODEs

We can now write the approximation for the solutions u and v on the discretized domain U_d as the vectors $u(t) = (u_{0,0}, \ldots, u_{N_y-1,N_x-1})$ and $v(t) = (v_{0,0}, \ldots, v_{N_y-1,N_x-1})$, where $u_{i,j}(t) = u(a + d_x j, c + d_y i, t)$ (and idem for v). Thus, we number them from left to right starting at the bottom left corner and working up. Now by (2.6) and (2.9) we can write (2.1) as a (large) system of ODEs

$$\partial_t \begin{pmatrix} u_{0,0} \\ u_{0,1} \\ \vdots \\ u_{N_y-1,N_x-1} \\ v_{0,0} \\ \vdots \\ v_{N_y-1,N_x-1} \end{pmatrix} = A \begin{pmatrix} u_{0,0} \\ u_{0,1} \\ \vdots \\ u_{N_y-1,N_x-1} \\ v_{0,0} \\ \vdots \\ v_{N_y-1,N_x-1} \end{pmatrix} + \begin{pmatrix} f_1(u_{0,0}, v_{0,0}) \\ f_1(u_{0,1}, v_{0,1}) \\ \vdots \\ f_1(u_{N_y-1,N_x-1}, v_{N_y-1,N_x-1}) \\ f_2(u_{0,0}, v_{0,0}) \\ \vdots \\ f_2(u_{N_y-1,N_x-1}, v_{N_y-1,N_x-1}) \end{pmatrix} \tag{2.10}$$

where A is a $2N_yN_x \times 2N_yN_x$ block matrix of the form

$$A = \left(\begin{array}{c|c} D_1A_* & 0 \\ \hline 0 & D_2A_* \end{array}\right)$$

and A_* a $N_yN_x \times N_yN_x$ matrix such that

$$\Delta u \approx A_* u$$

satisfies (2.6) and (2.9) for all respective rows.

2.6 Comparison

As this is a system of ODEs we can approximate the solutions with the Euler and the RK4 method. By Chapter 1.4 we would expect that RK4 generally gives a more accurate approximation than the Euler method. However, for the PDE we have to discretize the domain and approximate the Laplacian to obtain the system of ODEs. Thus, the solution of the ODE itself is an approximation of the PDE. To illustrate this, we consider the heat equation

$$\partial_t u = \Delta u, \tag{2.11}$$

i.e. a PDE with only diffusion and no reaction term.

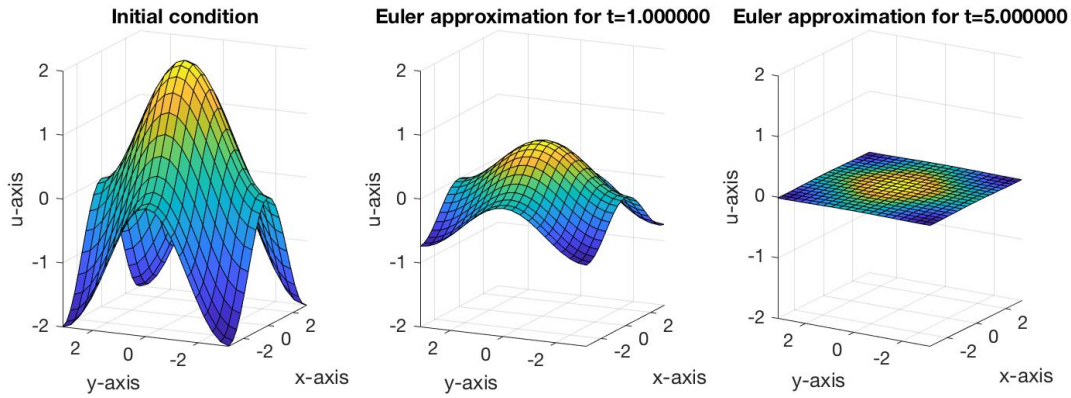

Figure 2.3: *Left plot: Chosen initial condition. Middle plot: Euler approximation of u for $t = 1$. Right plot: Euler approximation of u for $t = 5$. Constants used $N_x = N_y = 21$ and $h = 0.01$. We note that the RK4 plots look very similar to the shown Euler plots.*

We let the initial condition be

$$u(x, y, 0) = \cos(x) + \cos(y)$$

on the domain $(-\pi, \pi) \times (-\pi, \pi)$ (Fig. 2.3). It is straightforward to verify that this initial condition satisfies the boundary condition (2.3). For this equation, we can find an exact solution by assuming that $u(x, y, t) = A(t)B(x, y)$ (separation of variables). The initial conditions give $A(0) = 1$ and $B(x, y) = \cos(x) + \cos(y)$ and

substitution in the heat equation (2.11) gives

$$A'(t)(\cos(x)+\cos(y)) = -A(t)(\cos(x)+\cos(y))$$

and hence

$$A'(t) = -A(t)$$

which gives

$$u(x,y,t) = e^{-t}(\cos(x)+\cos(y)). \tag{2.12}$$

We can also determine the Euler and RK4 approximations (Fig. 2.3) and compare them by taking the difference of these approximations and the exact solutions. To compare, we take the ℓ^2-norm of the differences (Fig. 2.4).

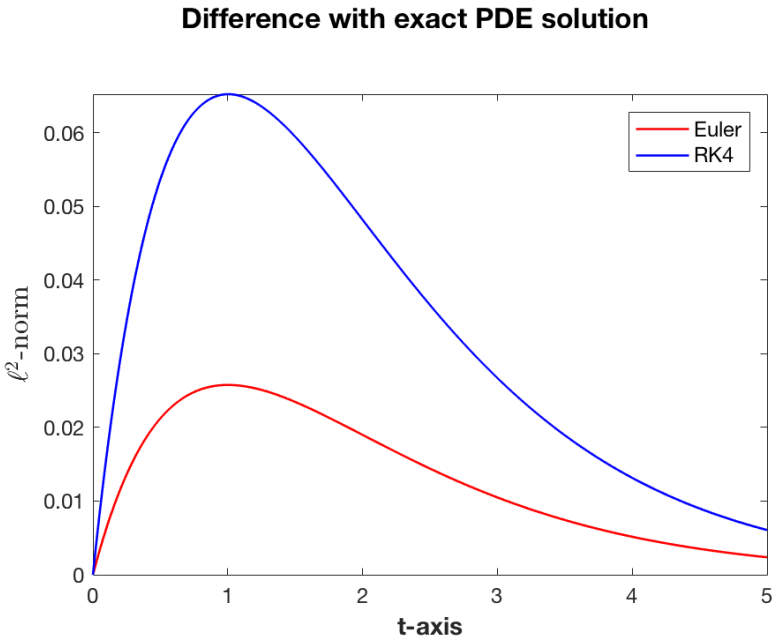

Figure 2.4: *ℓ^2-norm of the difference of between the exact PDE solution and the Euler and RK4 approximations with $h = 1/100$.*

In this example the Euler method is more accurate than the RK4 method in contrast to Chapter (1.4). To explain this, we recall that the Euler and RK4 methods are approximation the system of ODEs obtained after discretizing the domain and Laplacian. In this case, the discretized system is linear, so we can find an exact solution by determining the eigenvalues and eigenvectors of A_*. However, the matrix A_* is very large thus we will determine these numerically. By comparing this solution to the RK4 and Euler approximations (Fig. 2.5), we find that RK4 is much more accurate than the Euler method.

As illustrated in Fig. 2.6, the RK4 approximation is closer to the solution of the discretized ODE system than the Euler approximation. And in this case, the lesser accuracy of the Euler method (to the ODE system) causes it to be a better approximation of the exact PDE solution, however we note that generally this does not have to hold.

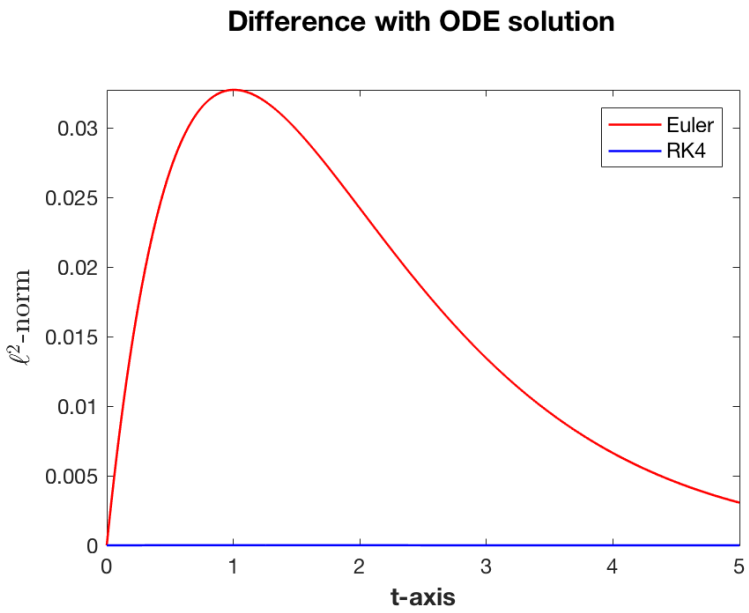

Figure 2.5: *ℓ^2-norm of the difference of between the ODE solution and the Euler and RK4 approximations with $h = 1/100$.*

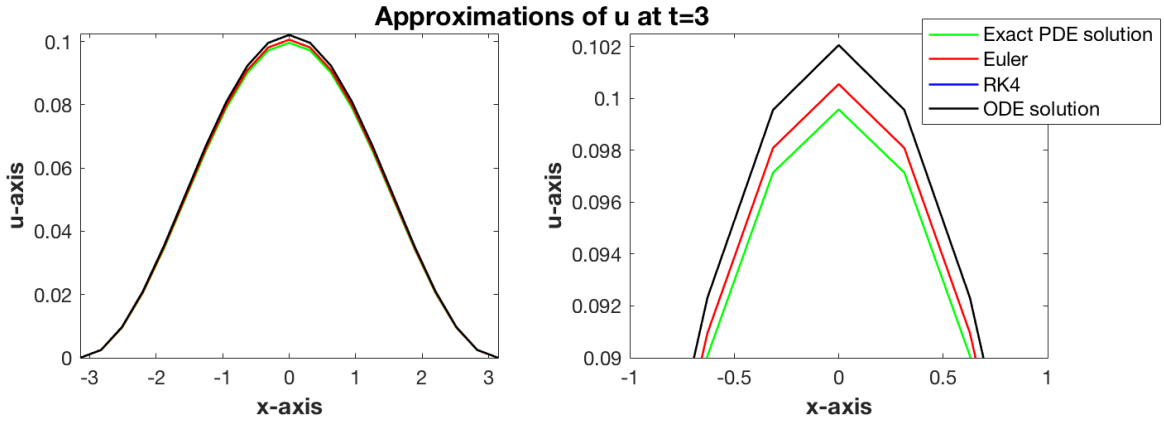

Figure 2.6: *Solutions for $t = 3$ and $y = 0$ and $h = 1/100$. Note that the RK4 approximation is so close to the ODE solution that it isn't visible..*

Thus, we conclude in line with Chapter 1.4 that in this case RK4 is more accurate for approximating the discretized system of ODEs, however as the system itself is an approximation of the PDE, the discretization sizes d_x and d_y are also important. The smaller these values, the smaller the error of the finite difference approximation (2.6), and hence the smaller the error of the ODE solution with respect to the PDE solution. A drawback of decreasing d_x and d_y is that this will increase N_x and N_y which increases the size of the discretized ODE system, which is of size N_xN_y, quickly.

Chapter 3

Bifurcations and continuation

We again consider the Reaction-Diffusion equation

$$\partial_t u = D_1 \Delta u + f_1(u, v)$$
$$\partial_t v = D_2 \Delta v + f_2(u, v)$$

Its stationary solution, that is (u, v) with $\partial_t(u, v) = (0, 0)$ can give rise to patterns. The rest of this thesis is dedicated to determine some of these patterns. The equation to determine stationary solutions is still a PDE, so non-trivial stationary solutions are often hard to find. One method to find approximate stationary solution is by initializing the solver from Chapter 2 with randomized initial values. Because if a solution starting at an initial value converges, then the the surface to which it converges is a stationary solution. However, for the rest of this thesis we will focus on another method called *continuation of parameters* and apply this method in **Matlab** to determine some stationary solutions. An advantage of this method is that there is a certain type of stationary solutions (unstable stationary solutions (Def. 2)) which will not be found using the first method. In this chapter we will discuss the concepts of *continuation* and construct a *continuation* algorithm.

3.1 Bifurcations for ODEs

Consider the autonomous ODE given by

$$u' = f(u, \lambda) \tag{3.1}$$

with $u = u(t) \in \mathbb{R}^n, t \in \mathbb{R}$ and $\lambda \in \mathbb{R}$ a parameter. A stationary solution is a $u_* \in \mathbb{R}^n$ with

$$f(u_*, \lambda) = 0. \tag{3.2}$$

Definition 1 (Definition 1.38 of [Chi06]). *A stationary solution u_* is* stable *if for all neighborhoods $U \subset \mathbb{R}^n$*

containing u_* there exists a neighborhood $V \subset U$ also containing u_* such that for all $v \in V$ we have $u(t) \in U$ the solution of the ODE for all $t > t_0$ with $u(t_0) = v$.

Definition 2. *A stationary solution u_* is* unstable *if it is not stable.*

Definition 3 (Definition 5.1.1 of [Kon14])**.** *A* bifurcation point *of (3.1) is a parameter value λ_b for which phase portrait undergoes a structural change. Where the structural changes include*

1. *the number of stationary solutions changes.*
2. *the stability of a stationary solution changes.*

An example for a bifurcation point and bifurcation diagram is shown in Fig. 3.1.

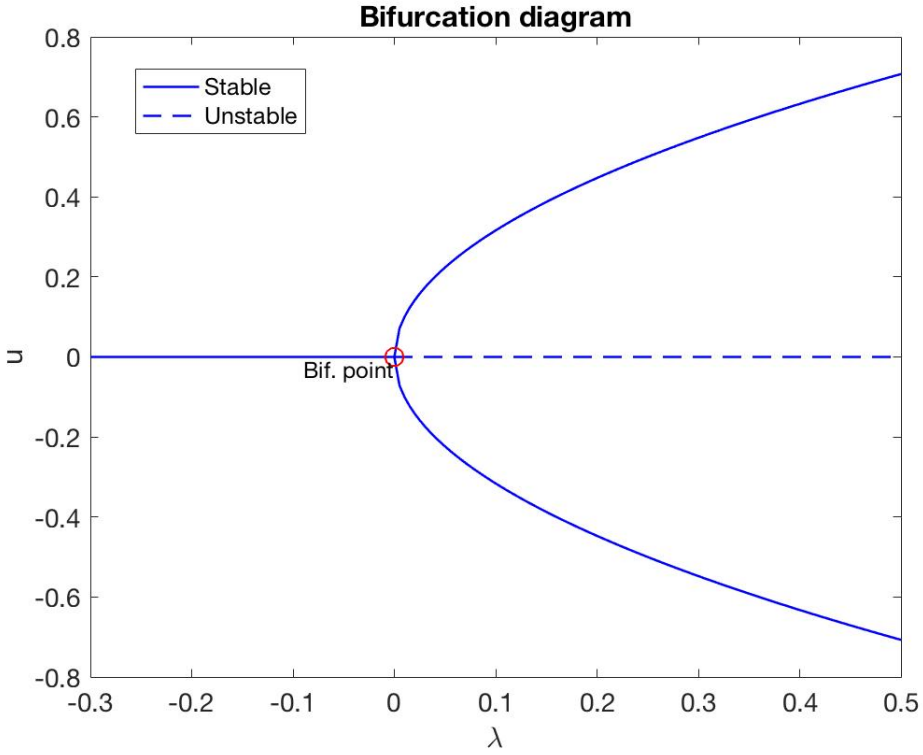

Figure 3.1: *Bifurcation diagram for $u' = \lambda u - u^3$. The branches illustrate stationary solutions and we have that $\lambda = 0$ is a bifurcation point for which both the number of stationary solutions and the stability of a stationary solution changes.*

3.2 Continuation for ODEs

Let u_0 be a stationary solution given arbitrarily for some $\lambda_0 \in \mathbb{R}$. Then the following theorem gives the existence other solutions in a neighborhood of λ_0.

Theorem 2 (Implicit function theorem [KP03])**.** *Let $h(u, \lambda) \in \mathbb{R}^k$ be a C^k function with $u \in \mathbb{R}^k$, $\lambda \in \mathbb{R}^m$ and*

$k, m \in \mathbb{Z}_{>0}$. *Let* $h(u_0, \lambda_0) \in \mathbb{R}^k$ *such that* $h(u_0, \lambda_0) = 0$. *If*

$$Jac_u(h)(u_0, \lambda_0) = \begin{pmatrix} \frac{\partial h_1}{\partial u_1}(u_0, \lambda_0) & \dots & \frac{\partial h_1}{\partial u_n}(u_0, \lambda_0) \\ \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial u_1}(u_0, \lambda_0) & \dots & \frac{\partial h_n}{\partial u_n}(u_0, \lambda_0) \end{pmatrix}$$

is nonsingular (i.e. is invertible), then there exists a neighborhood U *of* λ_0*, such that there exists a unique differentiable function* $g : U \to \mathbb{R}^k$ *with* $u_0 = g(\lambda_0)$ *and* $h(g(\lambda), \lambda) = 0$.

Proof. Theorem 3.3.1 of [KP03]. □

And the following corollary gives a formula for the derivative of the branch g.

Corollary 1. *Consider the implicit function theorem for* $m = 1$ *so*

$$h(g(\lambda), \lambda) = 0.$$

Taking a derivative with respect to λ *implicitly gives*

$$g'(\lambda)[Jac_u(h)(g(\lambda), \lambda)] + h_\lambda(g(\lambda), \lambda) = 0$$

which gives

$$g'(\lambda) = -[Jac_u(h)(g(\lambda), \lambda)]^{-1} h_\lambda(g(\lambda), \lambda)). \tag{3.3}$$

A remaining question is how to approximate the branch of stationary solutions $g = g(\lambda)$. We will use a predictor-corrector method to determine this branch, the predictor part will create an initial guess and the corrector part will improve that guess.

3.2.1 Predictor

The following predictor-corrector algorithm is taken from [GG13]. Let u_0 and λ_0 be given such that $f(u_0, \lambda_0) = 0$. If the implicit function theorem holds in this point, then there exists the unique differentiable branch $g(\lambda)$ with $u_0 = g(\lambda_0)$ and $f(g(\lambda), \lambda) = 0$ for λ in a neighborhood of λ_0. Let $\delta \in \mathbb{R}$ be a stepsize, then as

$$g(\lambda_0 + \delta) \approx g(\lambda_0) + \delta g'(\lambda_0)$$

as derived in (1.4), substitution of the formula of g' by Corollary 1 gives

$$g(\lambda_0 + \delta) \approx g(\lambda_0) - \delta[Jac_u(f)(g(\lambda_0), \lambda)]^{-1} f_\lambda(g(\lambda_0), \lambda).$$

This approximation is called the predictor step.

3.2.2 Corrector

It is possible to improve the accuracy of the predictor step with a corrector, for this algorithm we will use Newton's method [cite]. If u^0 is an approximation for a root of $f(u, \lambda) = 0$, then Newton's method gives that

$$u^1 = u^0 - [Jac_u(f)(u^0, \lambda)]^{-1} f(u^0, \lambda)$$

is a better approximation of the root if u^0 was close enough to the actual root. This step can be used iteratively until the error $f(u^i, \lambda)$ becomes small enough.

Algorithm 3: Predictor-corrector continuation

input : $f(u, \lambda), \lambda_0, u_0, \delta, M, C$

$u_1, \ldots, u_{t_n} \leftarrow 0$;

for $i \leftarrow 0$ **to** $M - 1$ **do**

  $Jac \leftarrow Jac_u(f)(u_i, \lambda_i)$;

  if *isInvertible(Jac)* **then**

    $invJac \leftarrow$ invertMatrix(Jac);

    $u_{i+1} \leftarrow u_i - \delta \cdot invJac \cdot f(u_i, \lambda_i)$;

    for $j \leftarrow 1$ **to** C **do**

      $Jac \leftarrow Jac_u(f)(u_{i+1}, \lambda_{i+1})$;

      $invJac \leftarrow$ invertMatrix(Jac);

      $u_{i+1} \leftarrow u_{i+1} - invJac \cdot f(u_{i+1}, \lambda_{i+1})$

  else

    break;

The continuation algorithm for Fig. 3.1 is shown in Fig. 3.2.

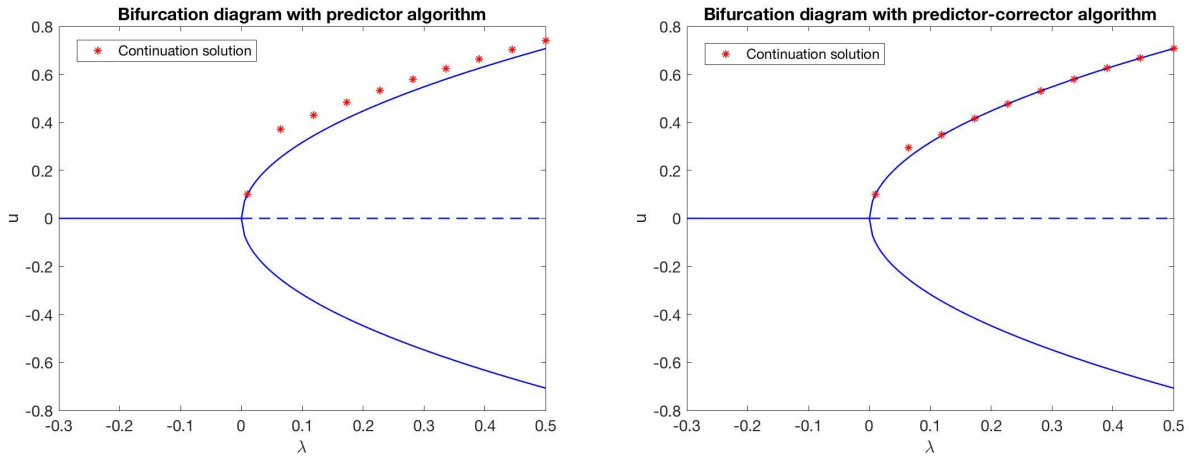

Figure 3.2: *Bifurcation diagram for* $u' = \lambda u - u^3$ *and continuation approximations. The used variables are* $\lambda_0 = 0.01, u_0 = \sqrt{0.01}$ *and* $\delta = 0.0544$. *The left plot is the continuation algorithm with only the predictor part and the right plot is the continuation with both the predictor and corrector part with one corrector iteration. Clearly the predictor-corrector algoritm is more accurate in this case.*

3.3 Continuation for RDEs

Continuation for RDEs can be done by discretization of the domain (Chapter 2.2) which gives a system (2.10) with the left-hand side zero. This system is of the form of (3.2), thus the algorithm described in Chapter 3.2 can now be used to approximate stationary solutions for RDEs. To illustrate the bifurcation diagram similarly to Fig. 3.1 for RDEs, a norm of the solution can be taken as the vertical axis. We refrain from developing code for continuation of RDEs as we will be using a Matlab package **pde2path** in the next chapter.

Chapter 4

Continuation for a Reaction-Diffusion equation with Pde2path

4.1 Patterns for a homogeneous RDE

We will now determine stationary solutions of the RDE

$$\begin{aligned} \partial_t u &= D_1 \Delta u - u + u^2 v \\ \partial_t v &= D_2 \Delta v + \lambda - u^2 v \end{aligned} \tag{4.1}$$

with $k_c = \sqrt{\sqrt{2}-1}$ and domain

$$U = (-8\pi/k_c, 8\pi/k_c) \times (-2\pi/(\sqrt{3}k_c), 2\pi/(\sqrt{3}k_c)),$$

using the **Matlab** package **pde2path** and based on the demo **schnakfold** [UW14]. It uses more advanced algorithms based on the concepts described in Chapter 3. Some major differences are:

- A finite element method is used instead of a finite difference method 2, which uses a different discretization.
- Branch switching at bifurcation points (Def. 3) at which multiple branches bifurcate.

From [DRUW14] we take $D_1 = 1, D_2 = 60$ and y assuming (u_*, v_*) stationary solutions that are constant for all time we get

$$\begin{aligned} u_*^2 v_* - u_* &= 0 \\ \lambda - u_*^2 v_* &= 0 \end{aligned}$$

which gives $u_* = \lambda$ and $v_* = 1/\lambda$. By starting with a constant solution we can find some patterns (Fig. 4.1).

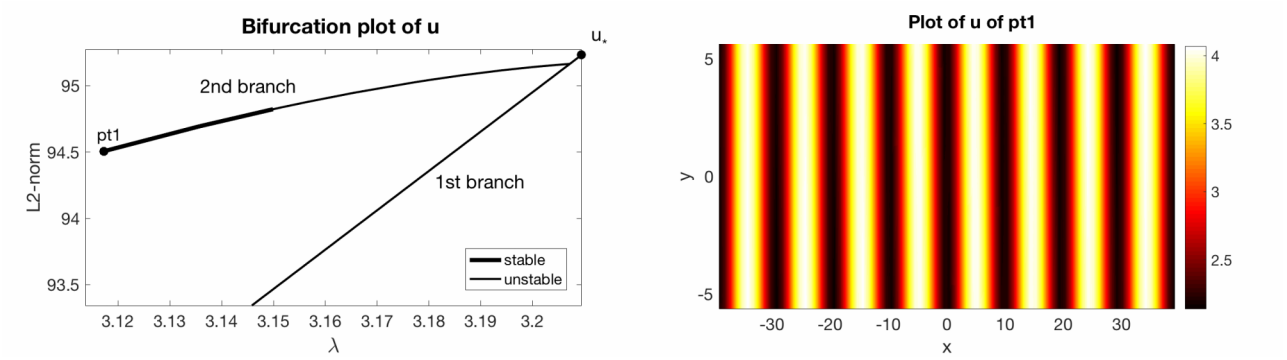

Figure 4.1: Continuation starting at $\lambda_* = \sqrt{60}\sqrt{3-\sqrt{8}}$, $u_* = \lambda_*$ and $v_* = 1/\lambda_*$ [DRUW14]. The intersection of the two branches is a bifurcation point detected while determining the first branch, then we branch switch at this bifurcation point to determine the second branch.

4.2 Patterns for a inhomogeneous RDE

We will now study what happens when we add an inhomogeneous term $q = (x, y)$ to the equation (4.1),

$$\partial_t u = D_1 \Delta u - u + u^2$$
$$\partial_t v = D_2 \Delta v + \lambda + q - u^2 v.$$

4.2.1 Inhomogeneous spot

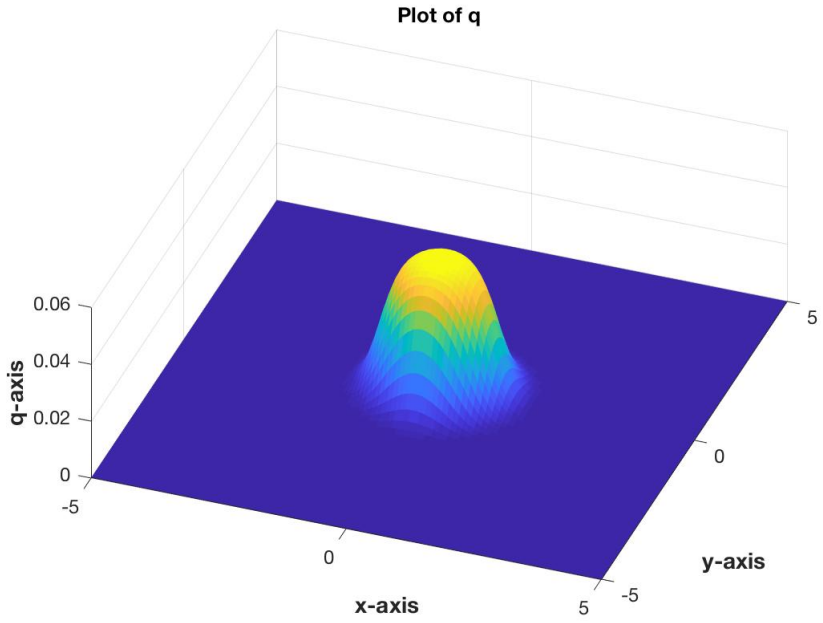

Figure 4.2: *Illustration of* $q = \mu\left[\tanh(x^2+y^2+\theta^2)+\tanh(x^2+y^2-\theta^2)\right]$ *with* $\mu = 0.03$ *and* $\theta = 1$. *Note that we have taken a different domain for this illustration.*

Let

$$q = \mu \left[\tanh(x^2 + y^2 + \theta^2) + \tanh(x^2 + y^2 - \theta^2)\right],$$

with $\mu, \theta \in \mathbb{R}$ this illustrates a single spot with a radius of θ (Fig. 4.2). Continuation starting with the solution found in Fig. 4.1 as stationary solution and again μ as bifurcation parameter we get Fig. 4.3. We observe that

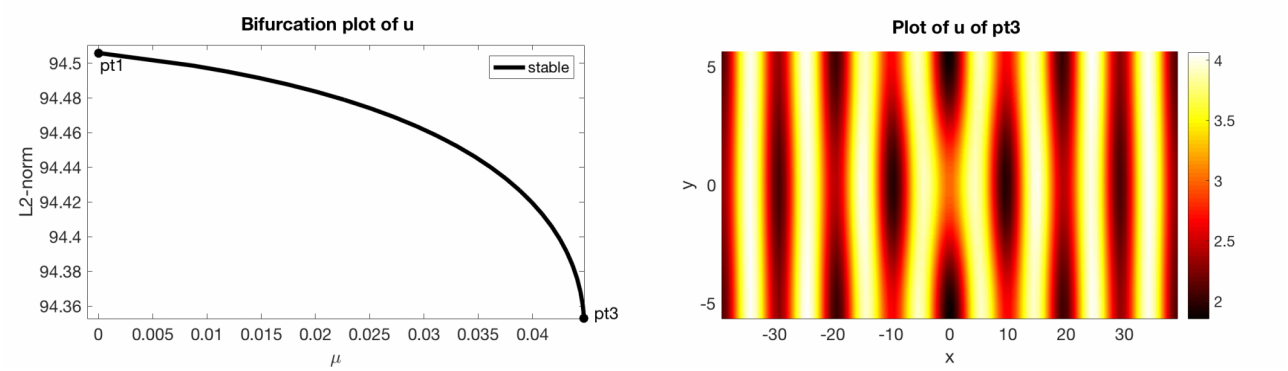

Figure 4.3: *Continuation starting at $\mu = 0$ and pt1 represents the solution found in Fig. 4.1 with $\theta = 2$. Note that we have taken a different domain for this illustration.*

stripes deform a bit in the middle where the spot is located which in turn effects the stripes further away although to a lesser extend.

4.2.2 Inhomogeneous stripe

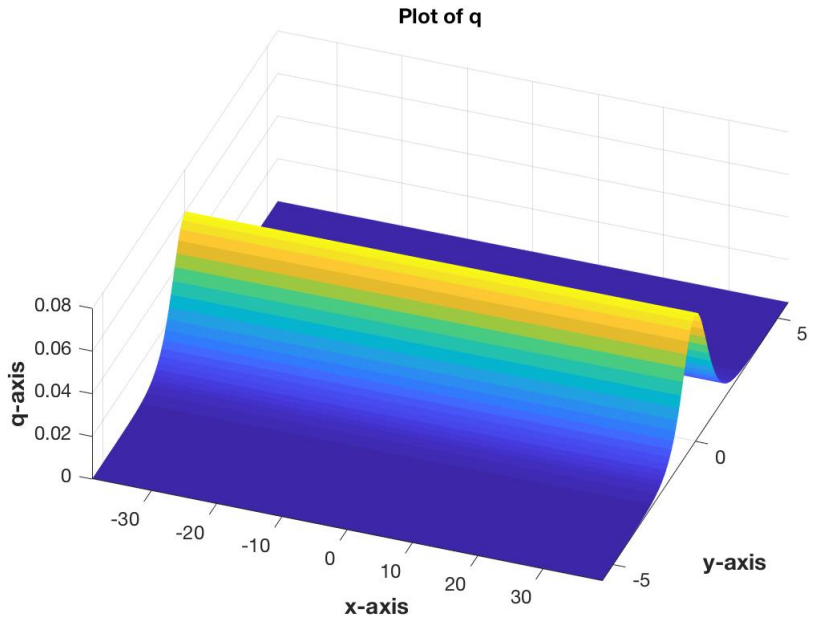

Figure 4.4: *Illustration of $q = \mu \left[\tanh(y + \theta) + \tanh(y - \theta)\right]$ with $\mu = 0.3$ and $\theta = 0.1$.*

We will now consider a different inhomogeneous term q. Let

$$q = \mu \left[\tanh(y + \theta) + \tanh(y - \theta)\right],$$

with $\mu, \theta \in \mathbb{R}$ this illustrates a single horizontal stripe with width from $y = -\theta$ to $y = \theta$ (Fig. 4.4). We can then continue from the pattern found in Fig. 4.1 with μ as the bifurcation parameter (Fig. 4.5). We observe that the

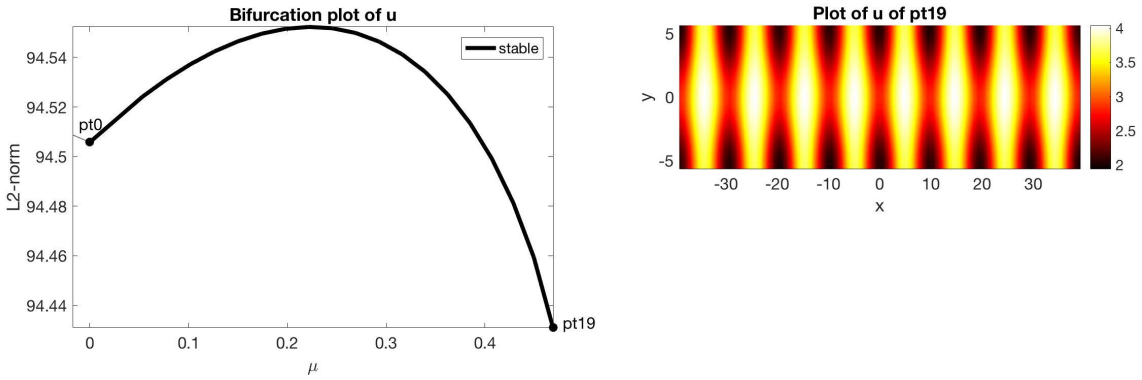

Figure 4.5: *Left side shows continuation of bifurcation plot starting at $\mu = 0$ and pt0 represents the solution pt1 found in Fig. 4.1 with $\theta = 1/10$. Right side shows a plot of u at pt19 corresponding to the bifurcation plot.*

width of the stripe pattern increases around the area where the inhomogeneous stripe is located as μ increases. As we continue the branch (Fig. 4.6) we note that μ starts to decrease and that the pattern loses symmetry (Fig. 4.7).

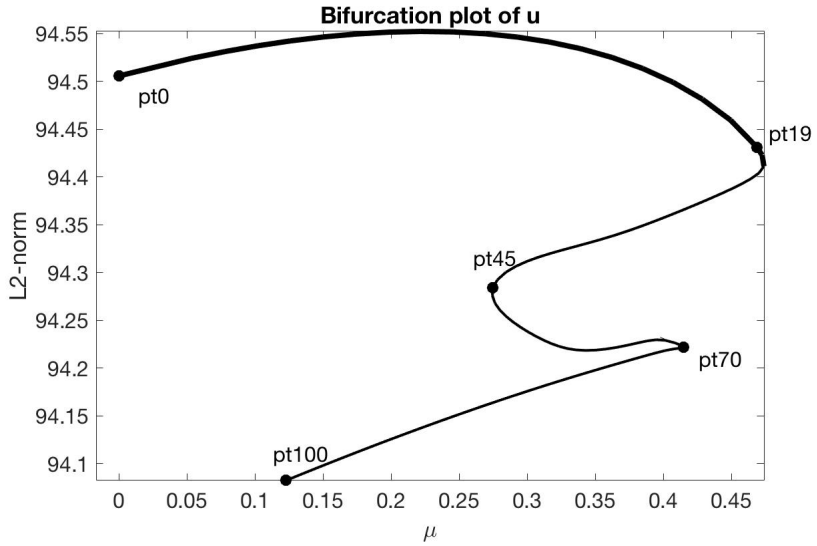

Figure 4.6: *Extended bifurcation plot of Fig. 4.5.*

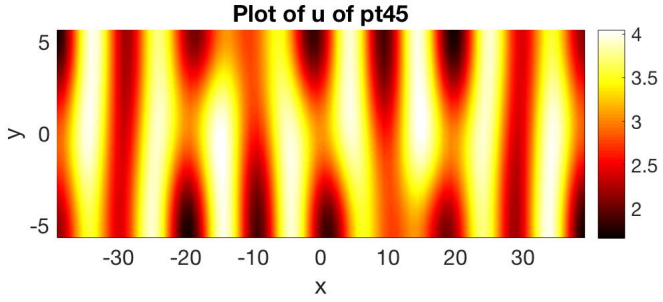

Figure 4.7: *Plot of u at pt45 corresponding to the bifurcation plot (Fig. 4.6).*

After this μ again increases and the pattern appears to become symmetric again and starts to illustrate spots

(Fig. 4.8).

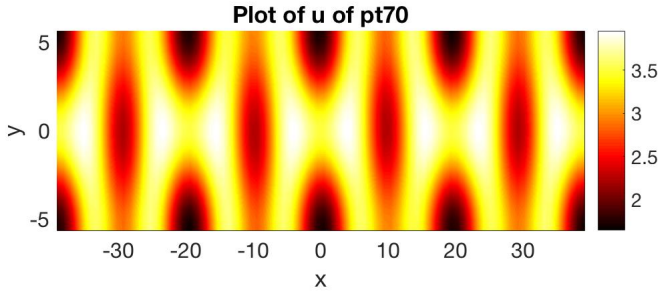

Figure 4.8: *Plot of u at pt70 corresponding to the bifurcation plot (Fig. 4.6).*

Further continuation of the branch however causes μ to decrease and we again lose symmetry (Fig. 4.9).

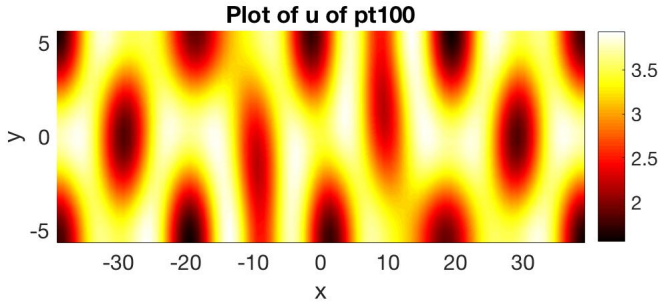

Figure 4.9: *Plot of u at pt100 corresponding to the bifurcation plot (Fig. 4.6).*

Chapter 5

Conclusion

We have discussed some basic numerical methods for ODEs, the Euler and Runge-Kutta fourth order methods. Then we extended these methods to Reaction-Diffusion equations with a finite difference method and showed that for these equations there is an additional approximation layer, the discretization of the domain. To find some stationary solutions of RDEs we introduced some basic concepts of continuation of parameters and used the **pde2path** to determine some patterns. Then after adding an inhomogeneous term to the RDE we determined how this changed a certain pattern.

To combine the patterns found in Chapter 4 with the initial condition solver from Chapter 2, a form of interpolation is required as the discretized domains differ. However, we note that both methods have different discretization errors, which can also lead to different results.

Bibliography

[Chi06] Carmen Chicone. *Ordinary Differential Equations with Application*, volume 34. Springer-Verlag New York, 2nd edition, 2006.

[DRUW14] T. Dohnal, J. D. M. Rademacher, H. Uecker, and D. Wetzel. pde2path - version 2.0: faster FEM, multi-parameter continuation, nonlinear boundary conditions, and periodic domains - a short manual. *ArXiv e-prints*, September 2014.

[GG13] Michael Ghil and Andreas Groth. Continuation of fixed-point solutions: A tutorial. *Retrieved from: http://www.environnement.ens.fr/IMG/file/DavidPDF/modelisation/MG+AG-Numerical-continuation_v2_appendix.pdf*, 2013.

[Hol07] M. H. Holmes. *Introduction to Numerical Methods in Differential Equations*, volume 52. Springer-Verlag New York, 2007.

[Kon14] Qingkai Kong. *A Short Course in Ordinary Differential Equations*. Springer International Publishing, 1 edition, 2014.

[KP03] Steven G. Krantz and Harold R. Parks. *The Implicit Function Theorem: History, Theory, and Applications*. Birkhäuser Basel, 1st edition, 2003.

[LeV07] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM, July 2007.

[RC99] F. John R. Courant. *Introduction to Calculus and Analysis I*. Springer-Verlag Berlin Heidelberg, 1st edition, 1999.

[Tur52] A. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 237(641):37–72, August 1952.

[UW14] Hannes Uecker and Daniel Wetzel. Numerical results for snaking of patterns over patterns in some 2d Selkov–Schnakenberg reaction-diffusion systems. *SIAM Journal on Applied Dynamical Systems*, 13(1):94–128, 2014.