NTNU – Trondheim
Norwegian University of
Science and Technology

Waves in Excitable Media

Bjørn Bjørge Theisen

Master of Science in Physics and Mathematics
Submission date: June 2012
Supervisor: Anne Kværnø, MATH
Co-supervisor: Harald Hanche-Olsen, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

CONTENTS

List of Figures

Abstract

This thesis is dedicated to the study of Barkley's equation, a stiff diffusion-reaction equation describing waves in excitable media. Several numerical solution methods will be derived and studied, range from the simple explicit Euler method to more complex integrating factor schemes. A C++ application with guided user interface created for performing several of the numerical experiments in this thesis will also be described.

Sammendrag

Denne oppgaven er dedikert til studiet av Barkleys ligning som er en stiv reaksjons-diffusjonsligning som beskriver bølger i eksiterbar media. Oppgaven vil utlede og utforske flere numeriske løsningsmetoder som kan anvendes på nevnte ligning, deriblant metoder basert på eksponensialintegratorer såvel som enklere skjemaer. En applikasjon skrevet i C++ som vil bli brukt under enkelte av de numeriske eksperimentene i oppgaven vil også bli beskrevet.

Preface

This thesis culminates my five years degree in applied mathematics at the Norwegian University of Science and Technology (NTNU). It has been carried out at the Department of Mathematics during the spring and summer of 2012 under supervision of Anne Kværnø (primary supervisor) and Harald Hanche-Olsen (secondary supervisor).

The work presented here is a continuation of the work done for my project work fall 2011. The project was at first intended to be a study in the theory behind numerical solution of stochastic with the application to Barkley's equation as a case study. However, Barkley's equation and the concept of waves in excitable media with their aesthetically and theoretically intriguing qualities proved to be more interesting, and therefore I chose to continue with this study in the master's thesis. This choice has given me the opportunity to first learn several practical programming skills through the work with the C++ application Wavefront, and then to learn the fascinating and highly applicable theory of Exponential Integrators.

I would like to thank my supervisors for giving me the opportunity to study this topic, as well as for their guidance. I would also like to thank my friends and family for support and specifically thank Pao Voung Vo and Christian Magnus Page for proofreading.

Bjørn Thæsen

June 28, 2012, Trondheim

I

Introduction

This thesis is an introduction to waves in excitable media represented through Barkley's equation. Chapter II entitled "Waves in Excitable Media" will discuss some of the conceptual theory behind this kind of waves, where in nature they will appear and motivate some of the interest for studying them. We will also examine some of mathematical models and see how Barkley's equation can be derived from the famous Hodgkin-Huxley equation. In this chapter we will also introduce *Wavefront* - a C++ application featuring real time interaction and a guided user interface which will later be used as a practical tool for examining some of the properties of these waves. We will also discuss some properties of the software.

The next two chapters will be dedicated to numerical aspects of the solution schemes of varying complexity implemented in *Wavefront*. In chapter III we will take a closer look at some already existing schemes including a semi-implicit approximation and a semi-spectral method, and in chapter IV we will study an interesting class of schemes known as exponential integrators.

This thesis will mainly be founded on work by mathematicians like Shardlow and Barkley, which have published several papers on the subject of waves in excitable media. However, their papers has been limited to the simple solution schemes described in chapter III. The contribution to the field by this thesis will primarily be in chapter IV. Barkley's equation is a good example of what is known as a stiff reaction-diffusion equation, so the work presented here can also be relevant for the solution of this class of problems in general.

This thesis is also a natural continuation of earlier project work on this subject done by the author, which was limited to some of the topics discussed in the introductory chapter II and chapter III.

II

Waves in Excitable Media

I. Introduction

Excitable media are spatially distributed system able to sustain waves. The system can be characterized as a group of individual points communicating only with it's intimate neighbors, and that inherit a degree of excitableness. A point can be either at a state of rest (in literature often denoted the *quiescent state*), in a *state of excitement* or somewhere between these two extrema. Small perturbations will cause a point at rest to remain at rest, while impulses over a certain threshold will cause an excitation and after a certain time period the point will gradually return to rest, which is called the *refractory phase*. A point will pass its excitation to neighbor points through diffusion. A state of recovery will follow the refractory phase, and during this phase, the point will not be able to be excited. A forest fire is an intuitive example; a fire will spread like a diffusion and trees on fire will ignite neighbouring trees. After a tree has burned down, it will have to regrow until the fire can pass this point anew.

There are several idiosyncrasies which separate this sort of waves from propagation of waves in passive media, e.g. sound waves, heat waves etc. As mentioned, the excitable media will have to go through a state of rest before it can carry new waves, and the waves will be without gradual damping (e.g. from friction). Also, there can be no sort of interference - two sound waves will pass each other more or less undisturbed, but two fire fronts will simply extinguish each other.

As mentioned, impulses travel by diffusion and the propagation speed of the impulses are determined both by the diffusion rate and the curvature of the wave. The speed is approximately given by the relation (see [1])

$$c = c_0 + D\rho,$$

where c is the wave propagation speed, ρ is the curvature of the wave defined to be the negative reciprocal of the local radius of curvature, D is the diffusion coefficient and c_0 is the planar wave speed. The equation is valid for ρ close to zero, and for planar waves, the speed is only determined by the state of the points in front of the wave and the diffusion coefficient. This shows that the speed of the wave front is not uniform as it would be for homogeneous inactive media.

Many fascinating phenomena can occur in such systems. The main motivation for studying waves in excitable media has been their ability to form and sustain spiral waves and at which conditions these waves are stable or unstable (see fig II.8 and II.15 for illustrations of broken wave patterns). Another topic of interest has been meandering of spiral tips (see examples in [2]), and they are also able to generate labyrinthine patterns (see fig II.14). Mathematicians have not studied such systems for their purely aesthetic qualities only - many physical systems can be described as waves in excitable media, examples include several chemical reactions that inherits these characterizations (most well-known being the Belousov-Zhabotinsky reaction) and such reactions have, in fact, been invoked in order to explain how different patterns on animal skin emerges [13]. Other examples includes auto-catalysis on metal surfaces and communication within and across nerve cells.

Figure II.1: Belousov-Zhabotinsky reaction is a well known example of a chemical non-equilibrium reaction and a physical visualization of waves in an excitable medium. The patterns in the photo above are made as Cerium(IV)(yellow) is reduced to Cerium(III)(colorless) by a propanedioic acid, and then oxidized back again by bromate(V) ions [12]. Notice the resemblance to the pattern in fig II.15. © Arthur Winfree/Science Photo Library.

Another particularly interesting example is propagation of electric pulses causing contraction of cardiac muscles. The heart beat is coordinated by a set of cells acting as independent oscillators (called the sinoatrial node). The electrical current is distributed to the ventricular tissue (the ventricles are the large chambers pushing the blood out of the heart) through a system of specialized fibres called the His-Purkinje system. The electric wave propagates from inside to outside, terminating at the Epicardium (the outer layer of the heart). This process might spawn both one, two and three dimensional self sustaining waves, depending on the local anatomy of the heart, and might in some cases cause potentially fatal cardiac arrhythmias (abnormal heart rhythms). Two dimensional arrhythmias are associated with self sustaining two dimensional waves in the atria (the upper chambers of the heart). The atrial wall is quite thin, and therefore, waves in this part of the heart can be considered two dimensional. Often, these waves have the form of spirals. The disorganized electric cardiac activity might be of permanent nature - they are not immediately life threatening, but while sustaining over a longer time span, they might lead to strokes.

The standard clinical method to eliminate the arrhythmias is application of electric shock through the heart, however, this is both painful for the patient and not always effective. The study of self sustaining spirals in excitable media is therefore being applied in order to find alternative ways to dissolve the atrial arrhythmias. [3]

II. Mathematical Models of Waves in Excitable Media

One way to describe this sort of interaction is to use a system of partial differential equations, where each equation describes one individual cell. The DiFrancesco-Noble model of Purkinje fibers is an example of this sort of models. Such systems might require an extensive amount of parameters and variables, and is therefore hard to simulate and analyse [2]. Another way is to use cellular automata. This is intuitive and easy to implement, but most models fail to model curvature effects [4]. On the other hand, there are systems of differential equations of simpler nature which lacks the drawbacks of the cellular automata.

The equation we will study in this paper is Barkley's simplification of the FritzHugh-Nagumo model for transmembrane potentials. FritzHugh-Nagumo model is again a simplification of the famous Hodgkin-Huxley model for whom Alan Lloyd Hodgkin (1914-1998) and Andrew Huxley (1917-2012) won the 1963 Nobel Prize in Physiology or Medicine, and which is considered one of the major results in biophysics of the 20th century. The system of equations was a result of experiments conducted of the axons of a giant squid. The axon of a giant squid, controlling the water jet propulsion system, is particularly large (up to 1 mm in diameter) and exhibits excitable nature and is therefore well suited for this sort of experiments. They knew initially that the cell membrane carries a potential across the inner and outer surfaces, not unlike the basic principles of how a capacitor and a resistance work in parallel, which can be described as

$$C_m \frac{\mathrm{d}V}{\mathrm{d}t} = -\frac{V - V_{eq}}{R} + I_{appl}.$$

Here, C_m is the membrane capacitance, R the resistance, V is the potential between the inner and outer surface, V_{eq} is the rest potential and I_{appl} is the applied current. Hodgkin and Huxley assumed the potensial in the squid axon was determined by a flow of potassium and sodium ions, and this assumption leads to a new version of the above equation,

$$C_m \frac{\mathrm{d}V}{\mathrm{d}t} = -g_K n^4 (V - V_K) - G_{Na} m^3 h (V - V_{Na}) - G_L (V - V_L) + I_{appl}$$

where the subscripts K, Na and L corresponds to potassium, sodium and leakage channels. The terms $g_K n^4$, $g_{Na} m^3 h$ and g_L are the conductances. The variables n, m and h are hypothesized variables controlling the flow of ions. These were governed by the following relation,

$$\rho_w(V) \frac{\mathrm{d}w}{\mathrm{d}t} = w_\infty(V) - w, \qquad w = n,\ m,\ h.$$

Toghether, the two equations above constitutes the Hodgkin-Huxley model.[6]

The Hodgkin-Huxley model is a four dimensional system which Richard FitzHugh (1922–2007) was able to reduce to a two dimensional system. He applied the observation

that the two gating variable n and h have slow kinetics compared to m, approximated $n + h$ by 0.8. The latter was justified by the parameters chosen by Hodgkin and Huxley. He also noticed that the n-nullcline could be approximated by a straight line, and that the V-nullcline had the shape of a cubic function. He was thus able to reduce the model to the following two dimensional system,

$$\begin{aligned}\frac{\mathrm{d}v}{\mathrm{d}t} &= v(v-\alpha)(I-v) - w + I \\ \frac{\mathrm{d}w}{\mathrm{d}t} &= \epsilon(v - \gamma w)\end{aligned}$$

v represents the potential, w is the sodium gating variable, and α, γ and ϵ are parameters. Later, Jin-ichi Nagumo was able to construct a circuit replicating the model, and therefore, the system is now known as the FitzHugh-Nagumo model.

In this paper we will study a further simplification of the FitzHugh-Nagumo model that was proposed by Dwight Barkley in [7], describing the interaction between an excitation field u and an inhibitor field v. In its most general form, *the Barkley equation* is thus

$$\begin{aligned}\frac{\mathrm{d}u}{\mathrm{d}t} &= D\nabla^2 u + \tfrac{1}{\epsilon} f(u,v) \\ \frac{\mathrm{d}v}{\mathrm{d}t} &= D_v \nabla^2 v + g(u,v)\end{aligned} \tag{II.1}$$

D is the diffusion coefficient of the excitation field, D_v is likewise the diffusion coefficient of the inhibitor field (usually set to $D_v = 0$) and ∇ is the gradient operator. The functions f is always given by

$$f(u,v) = u(1-u)(u-\gamma(v))$$

with

$$\gamma(v) = \frac{v+b}{a}.$$

the constants ϵ, a and b are parameters whose influence will be described later. We see that $f(u,v)$ consist of three terms u, $1-u$ and $u-\gamma(v)$, the first term u will cause the solution to grow slowly near $u = 0$, the second term $1-u$ will likewise cause the solution to grow slowly and not pass $u = 1$ and the last term $u - \gamma(v)$ will adjust the solution according to the inhibitor field. The function $g(u,v)$ controlling the growth of the inhibitor field is found to vary in the literature, but the three most common versions are

$$g(u,v) = u - v \qquad \text{(standard inhibitor kinetics)}$$

$$g(u,v) = u^3 - v \qquad \text{(delayed inhibitor kinetics)}$$

$$g(u,v) = \begin{cases} -v, & u \le c_{th} \\ 1 - 6.75u(u-1)^2 - v, & c_{th} \le u \le 1 \\ 1 - v, & u > 1 \end{cases} \qquad \text{(M. Bär and M. Eiswirth kinetics)}$$

All three have different abilities - spiral break ups and turbulence are particularly easy to provoke for the delayed kinetics, and the last version was introduced by M. Bär and M.

Eiswirth (see [18]) in order to recreate chemical surface reactions such as the CO oxidation of Pt(110), taking into consideration that the nature of this reaction will be altered as the concentration of one of the terms exceeds a certain threshold c_{th}.

III. System Dynamics of Barkley's Equation

In the case of standard inhibitor kinetics

In order to properly being able to implement a stable numerical solution of Barkley's equation, we need to have knowledge about some aspects of its behaviour. Let us first assume we are only studying the case of standard inhibitor kinetics, $\frac{dv}{dt} = g(u, v) = u - v$, although most properties of the system is regardless of what inhibitor kinetics we choose, especially those associated with the excitation term $f(u, v)$.

The relation between the excitation variable u and the inhibitor variable v for a single point is illustrated in fig II.4. The dynamics of the system $\frac{\mathrm{d}u}{\mathrm{d}t} = f(u, v)$ and $\frac{\mathrm{d}v}{\mathrm{d}t} = g(u, v)$ is shown in fig II.2. The system has two unstable equilibrium points $(u, v) = (0, 0)$ and $(u, v) = (1, 1)$. It is also worth noticing that solutions will not be excited for values above the diagonal blue line denoting the nullcline (i.e. the lines such that $\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{\mathrm{d}v}{\mathrm{d}t} = 0$), $u = \frac{v+b}{a}$. The different states and their respective domain in the phase plane is illustrated in fig II.3.

Notice from figure II.2 that a deterministic solution starting within the marked quadrant $0 \leq u \leq 1$ and $0 \leq v \leq 1$ will never leave the region, but for $u > 1$ solutions will diverge toward infinity. This is very important to notice since models with additive noise will easily be pushed into this region. The same is the case for the region $u < 0$, although here the solution will diverge to minus infinity. A numerical solution needs to take this into consideration, or else the solution will not be stable.

The parameters a and b are determining how susceptible solutions are to excitation, and keep in mind that they might as well be negative (although only b can be zero). As we see from the second term in the expression for $f(u, v)$ (II.II), for solution trajectories undisturbed by diffusion, if $u - \frac{v+b}{u} < 0$ then, given that $u \leq 1$ which it usually is, then $\frac{\mathrm{d}u}{\mathrm{d}t}$ will be declining. We can from this observation rationalize that for values of u and v such that $u < \frac{v+b}{a}$, then u will enter the refractory phase. Any perturbations of u will have to push u above this line in order to cause excitations. This is verified by fig II.2, where we can observe a shift in the orientation of the arrows representing the vector field above the blue diagonal line representing $u = \frac{v+b}{a}$. It is then obvious that by increasing b (this will in context of fig II.2 shift the blue line downwards) the ability of the medium to get excited will decrease and vice versa. By increasing a, the medium's susceptibility for excitations by rotating the blue line of II.2 counter-clockwise.

From these observations, we can predict, that for values for b such that the line $u = \frac{v+b}{a}$ passes above origo in fig II.2 (i.e. $b/a > 0$), the system will oscillate. Oscillating waves is common in e.g. the Belousov-Zhabotinsky reaction. It would likewise be trivial to find combinations of a and b that would render the system inexcitable, or only allow low excitations.

The parameter ϵ controls the separation between the excitation and inhibitor time scales, as it will control the magnitude of $f(u, v)$. High values of ϵ will cause $\frac{\mathrm{d}u}{\mathrm{d}t}$ to grow slowly, thus increase the length of an excitation, and likewise small values will cause the

time span of the excitations to shorten. Keep in mind that ϵ in itself does not alter the ability to get excited. However, ϵ does actually control the time before the system can be excited anew (this time period was in the introduction mentioned as the rest state). ϵ is thus crucial for determining the periodicity of wave trains and spiral waves.

By closely observing the phase plane in fig II.2 we can see that solutions will diverge for initial positions in the region $u > 1$ for any v and the region $v < 0$ for any u. Initially, deterministic solution trajectories will never enter this area but in the presence of stochastic noise and probably also in the case of improper solution schemes, this is a possibility that cannot be neglected. We could solve this issue by implementing our solution scheme with a simple threshold technique, denying solutions access to the divergent domain, but in order to obtain a faster and more analyzable implementation, I will rather use the same solution as Shardlow in [4]. Let f and g in II.1 be given by

$$\tilde{f}(u,v) = \begin{cases} f(u,v), & \text{if } u \leq 1 \\ -|f(u,v)|, & \text{if } u \geq 1 \end{cases}$$

and

$$\tilde{g}(u,v) = \begin{cases} g(u,v) & \text{if } v \geq 1 \\ |g(u,v)| & \text{if } v < 0 \end{cases}$$

This nonlinear modification will ensure the solution to be bounded for all initial data, and will be used in the numerical experiments of this paper (if not otherwise noted). An illustration of the dynamics of the modified systems substituting $\tilde{f}(u,v)$ and $\tilde{g}(u,v)$ for $f(u,v)$ and $g(u,v)$ is given in fig II.5. When modifying the reaction terms we run the risk of slowing down the simulations, however, simple tests (in MATLAB) show that the implementation of the modified test terms are 1.23 ($\tilde{f}(u,v)$) and 10.38 ($\tilde{g}(u,v)$, this is high, but this functions is also very little time consuming) times slower.

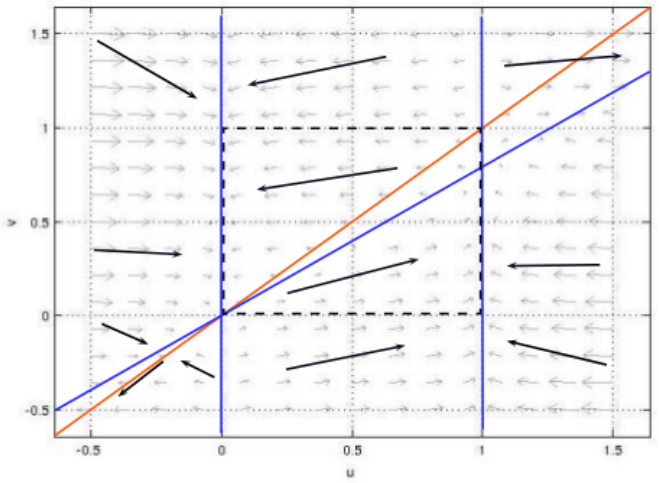


Figure II.2: Illustration of the dynamics for $\frac{du}{dt} = f(u, v)$ and $\frac{dv}{dt} = g(u, v)$, using the standard parameter values $a = 0.75$, $b = 0.01$, $\epsilon = 0.03$ and $D = 1$. The blue lines are nullclines for $\frac{du}{dt}$ and the green line is the nullcline for $\frac{dv}{dt}$. The area in the middle marked by striped borders is the stable area for $0 \leq u \leq 1$ and $0 \leq v \leq 1$. The arrows marks the vector fields.

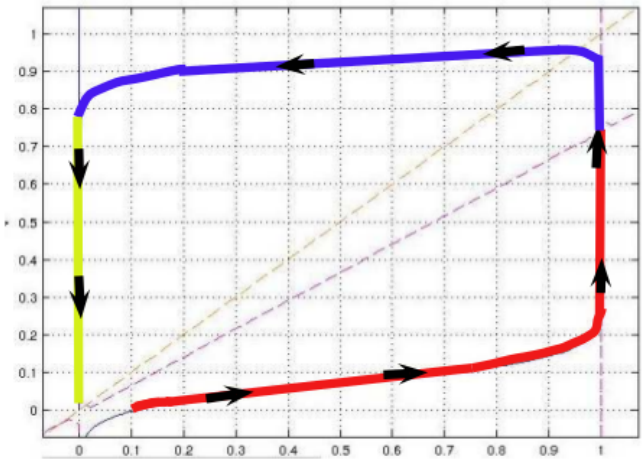


Figure II.3: The illustration shows a cut of the quadratic center region of fig II.2. The coloured path show a solution with initial conditions $(u, v) = (0.1, 0)$. Starting at the given initial position, the point is first excited (red), then at the refractory state (blue) and at last at the rest state (marked as yellow, at which it cannot be excited) before it returns to the equilibrium solution $(u, v) = (0, 0)$.

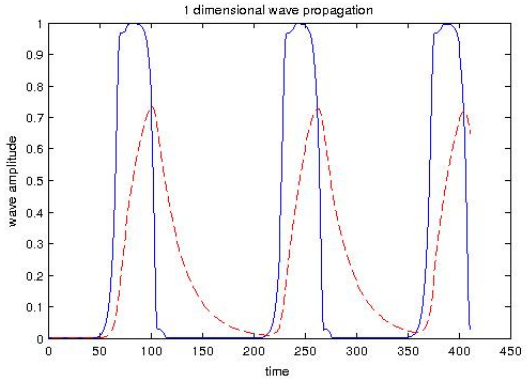


Figure II.4: One dimensional deterministic wave propagation. The image illustrates the relation between the excitation variable u (continuous line) and the inhibitor variable v (striped line) for a single point passed by three planar waves. The image was generated using the standard parameter values $a = 0.75$, $b = 0.01$, $\epsilon = 0.03$ and $D = 1$, for a time span $t_0 = 0$ to $t_{end} = 24.44$. $\Delta t = 0.04$.

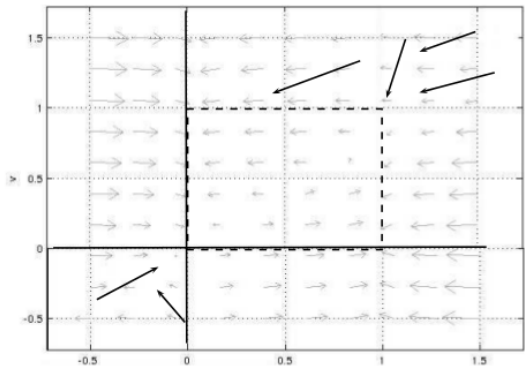


Figure II.5: System dynamics for modified phase plot. Compare with fig II.2.

Dynamics in the case of other forms of inhibitor kinetics

The prime difference observed between the *standard inhibitor kinetics* and the *delayed inhibitor kinetics* is the heavy susceptibility of the latter to cause spiral breakups. This version is, of course, also able to sustain spiral waves, but seemingly within a much lower regime of parameters.

Figure II.6 shows the phase plot of the system with delayed inhibitor kinetics including nullclines and a sample solution. Notice that in this case we have a new point where the nullclines coincide, and as observed from the figure, solutions close to this point will circulate with a growing radius and thus spend more time within this region until returning to the refractory phase. When ϵ grows, the orbiting time will increase, and this effect might disrupt stable spiral patterns.

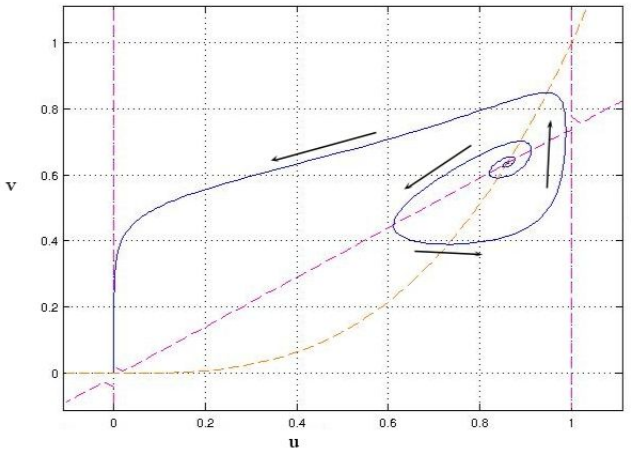


Figure II.6: Phase plot for system dynamics with delayed inhibitor kinetics, using $\epsilon = 0.05$, $a = 0.75$ and $b = 0.01$. The green and the blue striped lines shows the nullclines for $\frac{dv}{dt}$ and $\frac{du}{dt}$ respectively. Notice that in this case we have a second point where the two nullclines coincide; the black curve is a solution with initial conditions in this point.

IV. The Wavefront Application

The primary workload of this thesis was the interactive excitable wave simulation and mathematical dalliantical software named *Wavefront*. The program is written in C++ with openMP paralellization, it contains about 1500+ lines of code and has a user interface made with the cross-platform application framework Qt. The program features simulation with real time change of parameters, solution schemes, reaction kinetics and a certain range of other forms of interactivity.

The program solves Barkley's equation on a mesh of size 512×512 with adjustable real length L. Both additive noise (as described in chapter III.1) and randomly distributed inexcitable obstacles (as shown in fig II.10) are also supported. It also has a set up for simple change of reaction terms so that it is not only limited to the Barkley's equation, but can be used on several reaction diffusion equation.
Current solution schemes supported are

1. Explicit and semi-implicit Euler method.
2. Solution with diffusion added in Fourier space and reaction terms updated with the semi-implicit Euler approximation (later in this paper denoted as the Shardlow method).
3. Lawson-Euler.
4. Fourth order integrating factor method (IFRK4).

We will have a closer look at these schemes in chapter III and IV.

Overview of functionality

An illustration of the interface as well as some of the functionality is given in fig II.7. The main interface consist of three screens giving information about the state of u and v. The solution u_n (in blue) is displayed at the main screen (F), but you can switch to view v_n (in green) by hitting a button (undermost button at (C)). It has also a possibility to view a combination of both the two variables. The latter option will result in the display showing green somewhere, blue other places, and at the points where both the u and v values are high we will get different shades of the combination of blue and green known as cyan. The display (G) shows $(u_n)_{ij}$ plotted against $(v_n)_{ij}$ for a selected point (i, j) at (A). The default position of (i, j) is in the center. The graph (H) shows the same information plotted separately. The point of focus (i, j) can be changed by clicking anywhere on (F). You can also set the value of $(u_n)_{ij}$ and $(v_n)_{ij}$ directly by clicking inside (G).

Panel (D) in fig II.7 allows adjustment on the inhibitor kinetics (standard, delayed and M. Bär and M. Eiswirth kinetics are supported). Here we can also change solution scheme.

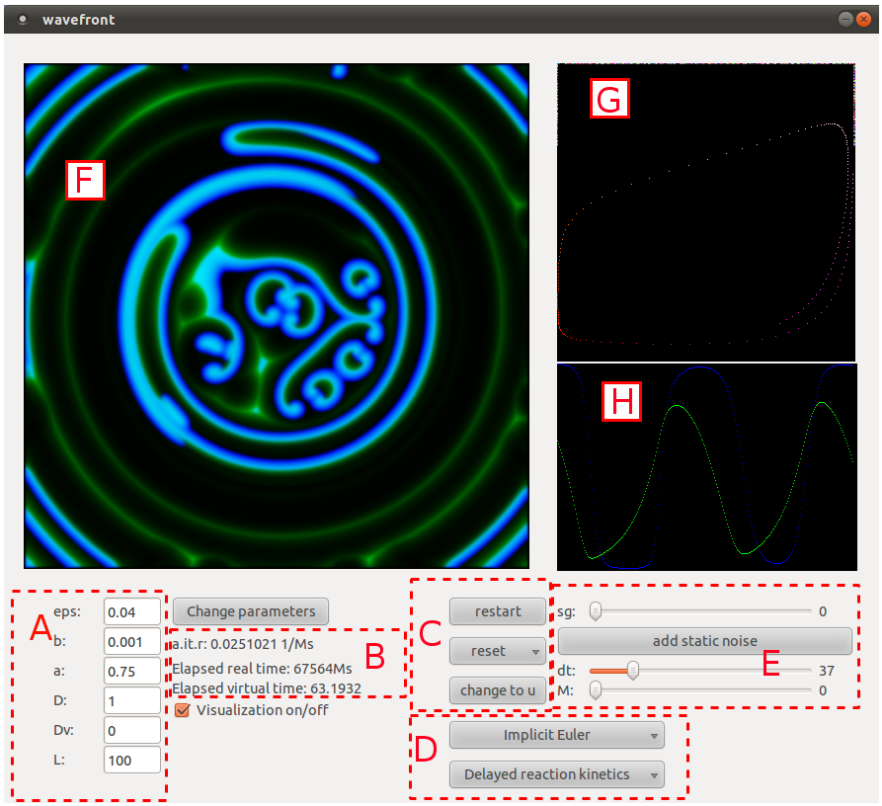


Figure II.7: *The interface of the Wavefront application. A: Parameters. B: Overview of average number of iterations per millisecond, elapsed real time and elapsed virtual time. C: Buttons for pause on/off, change of initial conditions, and button for switching between visible variables (currently it displays a combination of u and v). D: Change solution scheme and inhibitor kinetics. E: Adjustment of noise intensity, time stepping and M. F: Main display, shows either u, v or a combination. G and H: Plots of the relation between u and v for a selected position in the main display. The displays F, G and H can be turned off to save computational time.*

Some information like virtual and real time elapsed since last restart is given in panel (B). Panel (E) let you adjust variables associated with noise like M and σ (here denoted sg), add randomly distributed inexcitable points in (F) by hitting the button ”add static noise”, and change temporal step length Δt. To remove the inexcitable points, hit the same button again. The variable Δt is here denoted ”dt”, and the numerical indicator on the right side of the bar is to be divided by 100 to give the correct step length. Notice that not all of the functionality associated with the noise nor diffusion in the v-term is available for all solution schemes.

You can also capture a screen shot of the (F) field by hitting the button ”s” on the keyboard. The following table gives a quick overview of the supported abilities of the

different implemented methods;

	Diffusion in v term	Static noise	$\tilde{g},\ \tilde{f}$	adjustable $g(u,v)$
Explicit Euler	yes	yes	yes	yes
Implicit Euler	yes	yes	yes	yes
Implicit Shardlow	no	no	yes	yes
Lawson-Euler	no	no	no	yes
IFRK4	yes	no	yes	no

The row $\tilde{g},\ \tilde{f}$ tells whether or not the modified reaction terms works for the given method, which is necessary to know if we are to use additive noise. Static noise is not implemented for the methods using spectral diffusion, simply because this is a rather non-trivial task.

Some notes on implementation

The implementations of the numerical methods will utilize openMP paralellization wherever possible and thus have an advantage over the most implementations written in MATLAB. On the other hand, Wavefront will perform matrix and vector operations by means of for-loops, which will not be as efficient as the corresponding operations in MATLAB. Some tests shows that Wavefront typically will perform better in terms of iterations per second when it comes to the simpler methods like the Euler method and Lawson-Euler, but for more complex schemes like IFRK4, MATLAB is the fastest. The latter should be explainable by the heavy use of matrix operations this method requires. Comparison of some methods implemehted on both platforms is given in fig III.5.

The Fourier transformation is also quite important for the speed we are able to obtain. There is a plethora of available implementations of the Fourier transform for C++ and the one that was chosen for Wavefront is the third party subroutine library FFTW3.0, which is a highly optimized library featuring a well of sophisticated abilities such as detailed hardware adaptations, automatic parallelization and subroutines in Fortran (and even MPI support, but this will not be relevant for us).

The implementation of the Lawson-Euler method can be found in appendix E. Some notes on how the source files are structured is given in appendix G, and a quick guide on how to install the application is described in appendix F. Notice that some third party software is necessary to run the program, this includes FFTW3.0 for Fourier transforms, the numerical library GSL (GNU Scientific Library) for generation of random numbers and Qt for guided user interface.

The way the inexcitable obstacle functionality is implemented is simply by ignoring any reaction and diffusion in this single point - when the algorithms reaches one of these points it will simply skip it. However, this is not an entirely mathematical correct approach. The correct way would be to modify the central difference scheme used to approximate the Laplacian, but taking the nature of the excitable medium in account, the error caused

by this would probably be small.

*

We will in the next sections see some illustrations of the functionality the application provides.

V. Spiral waves

Spiral waves are an often occurring pattern in excitable media (in three dimensions, these will take the form of *scroll waves* [17]), among other because they are easily obtainable both in nature and artificially - as mentioned, cardiac arrhythmias can have the form of spiral waves and they are easily observable on fig II.1. In simulations, we can construct a spiral wave by initiating a planar wave and then breaking the wave in the middle, returning certain parts of u and v back to the unstable equilibrium point (0,0), and thus the waves will coil around the broken sector, causing a spiral wave.

Much work is done on analyzing the spirals, and in [17] (see also [14]), it is shown that the period of a (rigidly) rotating spiral is predicted by the following proposition,

Proposition 1.1. *The period ω of a (rigidly) rotating spiral is approximated by*

$$\omega = \frac{0692\mu}{\epsilon^{\frac{1}{3}}} - \frac{0.926}{a}$$

for

$$\mu \approx 2.70\frac{v_s(1-v_s)^{\frac{2}{3}}}{a} \qquad v_s = \frac{a}{2} - b$$

A spiral can either be rotating rigidly with its core at a certain point at all time, or it might be drifting, probably leaving the domain or moving in a circle or a complex meander-like pattern (examples of patterns are given in [2]). In his selected YouTube Videos [15], Barkley illustrates that the spiral core can even behave like a particle. The most interesting aspect of spiral waves are probably their breakups. Barkley show examples, both of collapse in spiral pattern close to its center (near field breakup) and in its periphery (far field breakup) at his home page [16], induced by a sudden shift in the parameter ϵ.

We will use the Wavefront application to see what we can do with spiral waves. Core breakups are easily generated by with delayed or M. Bär and M. Eiswirth kinetics as the spiral tips of these systems can collide and collapse into themselves without much outside stimuli. This is illustrated in fig II.8, where we have started a simulation using delayed inhibitor kinetics and let it evolve for a while.

Sometimes we can make spiral arms outside the core collide into themselves and cause turbulence, and now we will see how we can use Wavefront to investigate this. Fig II.9 shows a recreation of this phenomena using M. Bär and M. Eiswirth kinetics and some shifting of parameters. This was done by first using the parameters $N = 60$, $b = 0.045$, $a = 0.84$, $D = 1$ and $\epsilon = 0.04$. We then let the spiral pattern evolve for a while, and

at $T = 70$ we changes b to -0.045 and let the system stabilize for a while until $T = 200$ where we changes ϵ to 0.059. This causes modulation in the spiral arms that develops and at $T = 238$ we observe that the arm is totally broken and two new spiral cores emerges. At $T = 280$ we see that the turbulence is propagating outwards. If we continued observing the system we would see that the chaos slowly would move out of the domain. The interesting lesson to learn from this case was that the modulation causing the turbulence in the first case only was motivated by the change of ϵ early on; this will not happen again. The reason we had b negative to begin with was to keep the system from oscillating before the spiral was formed.

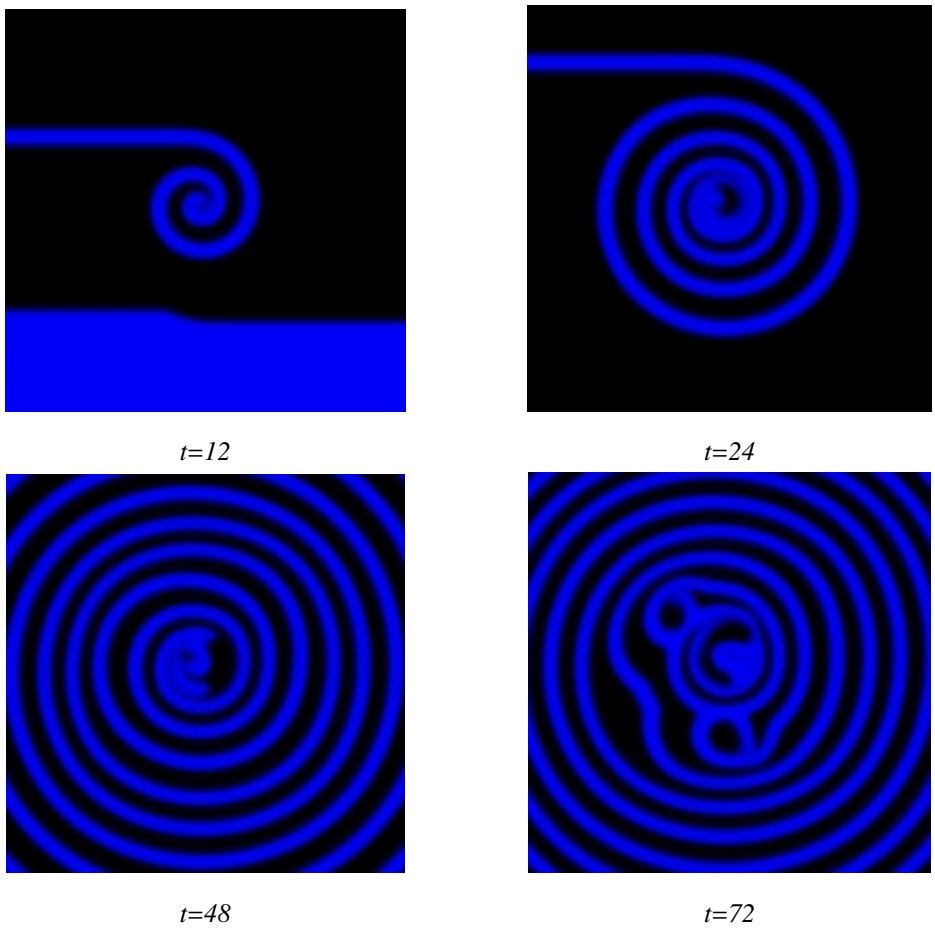


Figure II.8: *Spiral core breakup with delayed inhibitor kinetics reproduced with Wavefront. The simulations are done using a set of parameters which is not able to sustain stable spiral waves. Notice how the spiral tip is not clearly defined at* $t = 12$*, and as time goes the core will collide with itself and cause turbulence. Parameters used are* $\epsilon = 0.06$*,* $a = 0.75$*,* $b = 0.001$*,* $D/(\Delta x^2) = 1$*,* $L_x = L_y = 200$ *and* $\Delta t = 0.08$*.*

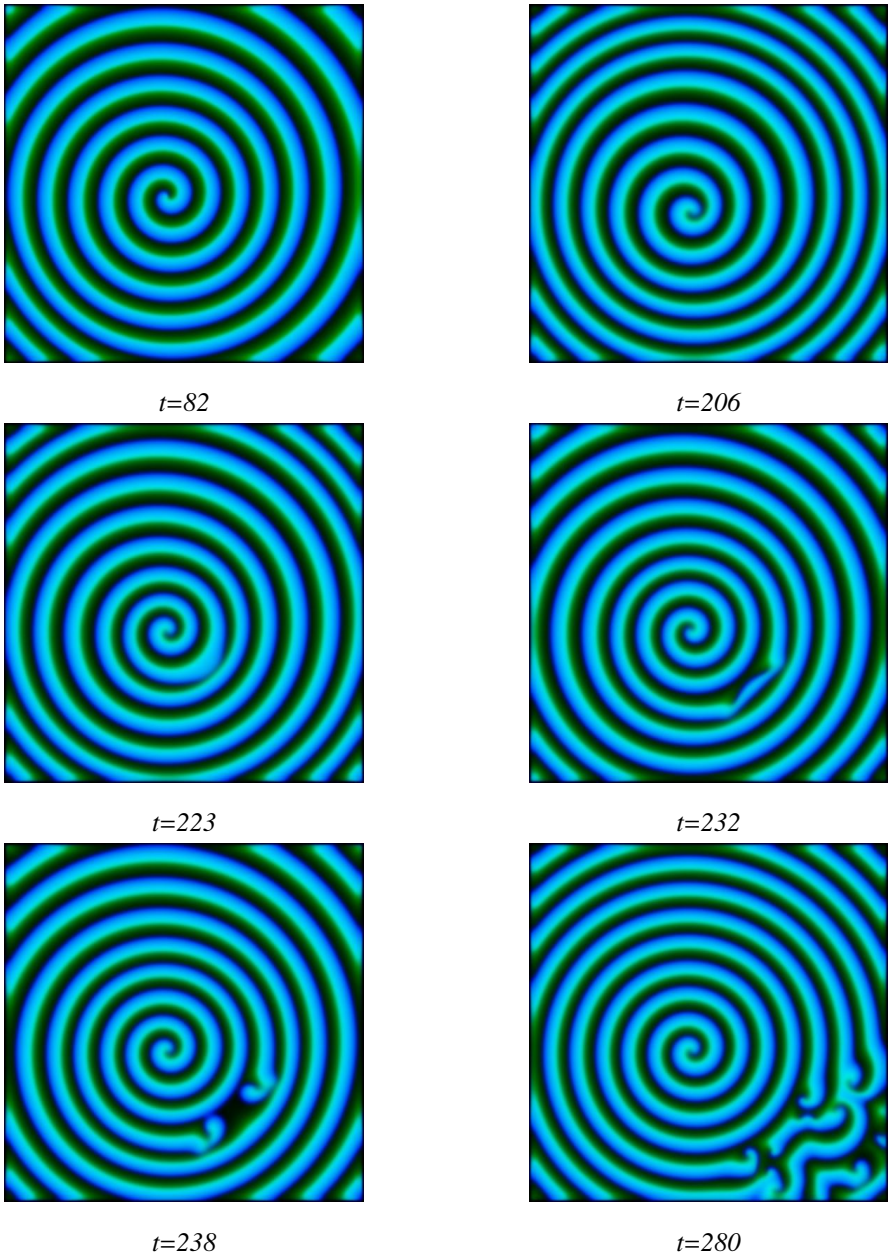


Figure II.9: *Far-field break up.*

VI. Introducing randomness

In this chapter we will describe what previously has been done do introduce stochasticity in models of waves in excitable media in general, and then move on to do some exper-

iments. The first attempt was probably to distribute random recovery rates in cellular automata. Another technique was applied by K. H. W. J. Ten Tusscher et al in [8], where non-excitable obstacles was randomly distributed on (sub-) domains in order to study its effect on spiral wave dynamics. This is illustrated in fig II.10. The distribution of inexcitable obstacles was inspired by the fact that cardiac arrhythmias is strongly correlated by the presence of inexcitable fibrotic tissue. Their conclusion was that in the influence of inactive obstacles, the spirals actually increased the stability of the spirals - in cases where spirals otherwise would break up, they were preserved in the presence of obstacles. Their explanation was that the obstacles reduced the spiral propagation period, which rendered the spiral more stable.

In [4], Shardlow mentions several stochastic extensions to Barkley's equation. The simplest is

$$\begin{aligned} \mathrm{d}u &= \left[D\nabla^2 u + \tfrac{1}{\epsilon} f(u,v)\right]\mathrm{d}t + \sigma \mathrm{d}W(t,x) \\ \mathrm{d}v &= g(u,v)\mathrm{d}t \end{aligned} \tag{II.2}$$

which only involves additive noise (here on, this model will be referred to as the *additive noise model*). Here, σ is a scalar controlling the noise intensity and $W(t,x)$ is a Wiener process of any spatio-temporal structure seen fit for the specific context (although white noise is usually chosen due to its simplicity). Spontaneous excitation has been observed in this system. Another two stochastic extensions are

$$\begin{aligned} \mathrm{d}u &= \left[D\nabla^2 u + \tfrac{1}{\epsilon} f(u,v)\right]\mathrm{d}t \\ \mathrm{d}v &= g(u,v)\mathrm{d}t + \sigma \mathrm{d}W(t,x) \end{aligned} \tag{II.3}$$

and

$$\begin{aligned} \mathrm{d}u &= \left[D\nabla^2 u + \tfrac{1}{\epsilon} f(u,v)\right]\mathrm{d}t + h(u)\mathrm{d}W(t,x) \\ \mathrm{d}v &= g(u,v)\mathrm{d}t \end{aligned} \tag{II.4}$$

for a function $h(u)$. The first proposed by Busch et.al in [9], the second considered by Alonso et al. in [10]. Here, $h(u)$ is chosen such that the random fluctuations only will affect the b, the variable which they assume controls the excitability of the system. This approach was inspired by how the photosensitive Belousov-Zhabotinsky reaction would be influenced by changing light conditions.

The last variation mentioned, is studied by Garcia-Ojalvo et al. in [11] is

$$\begin{aligned} \mathrm{d}u &= \left[D\nabla^2 u + \tfrac{1}{\epsilon} f(u,v)\right]\mathrm{d}t \\ \mathrm{d}v &= g(u,v)\mathrm{d}t + u\sigma \mathrm{d}W(t,x) \end{aligned} \tag{II.5}$$

This version will exhibit variation in the inhibitor field, sometimes causing breakup of spirals due to backfiring events for parameters that would sustain stable spirals for deterministic models (see fig II.13). After the stable pattern had broken up, the chaotic pattern

(Garcia-Ojalvo et al. names this *complex spiral dynamics*) will remain for a certain time period until the meandering cause the spiral cores to move away from the observed domain. In the rest of the paper, this model will be referred to as the *fluctiating inhibitor field model*.

First thing we will do with Wavefront is to check the influence of inexcitable obstacles on point waves. The result for delayed inhibitor kinetics is presented in fig II.10, and as we can see, the point wave is able to spawn secondary waves when passing dense clusters of inexcitable grid points. This is interesting because it might show how electrical pulses in cardiac tissue can form self sustaining waves turning into arrhythmia. The same effect could be reproduced with Bär and M. Eiswirth kinetics, but not standard inhibitor kinetics.

The next thing we can take a closer look at is formation of labyrinthine patterns in presence of additive noise. These patterns form for most reaction-diffusion equations and are one of their most interesting characteristics. In fig II.11 we see two patterns developing from the same parameters and initial conditions, except the images to the right is formed with noise. As we see, the left labyrinth develops symmetrically, while the right labyrinth fast loses it's ordered structure and turns into chaos.

Another thing that would be interesting to know is what discrepancy the noise causes depending on *when* the noise is introduced. The result of this experiment is displayed in fig II.12, where we have started the deterministic simulation with the same parameters as in fig II.11, but paused the simulation and introduced noise (corresponding to II.11 again) at time $T = 10$, $T = 20$ and $T = 40$. The result shows that adding noise after $T = 40$ hardly has any influence on the final pattern at time $T = 112$. There are differences between deterministic $T = 112$ in fig II.11 and the corresponding pattern for noise introduced $T = 40$ in fig II.12, but they are difficult to see.

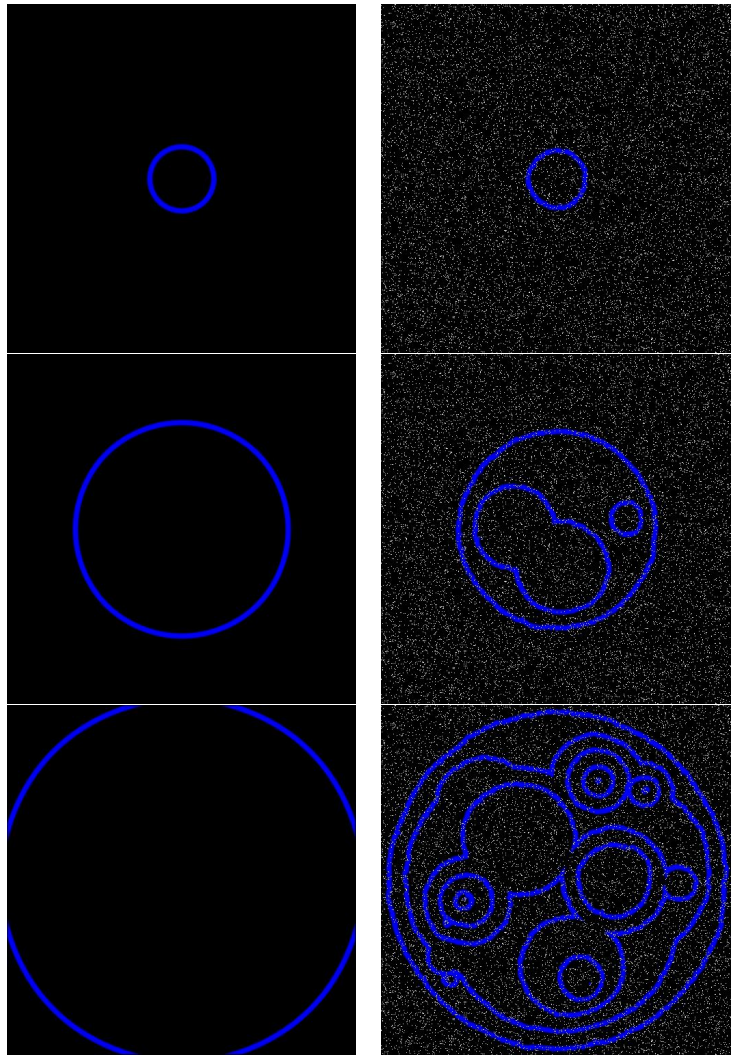


Figure II.10: *Comparison between deterministic wave propagation in medium with homogeneous excitement potential (left), and wave propagation in waves with randomly distributed inexcitable obstacles (right). The images shows that in the presence of obstacles the wave will propagate slower and spawn new waves. The phenomena that simple waves can spawn self sustaining new waves illustrates how cardiac arrhythmia can appear in sick or aged cardiac tissue.*

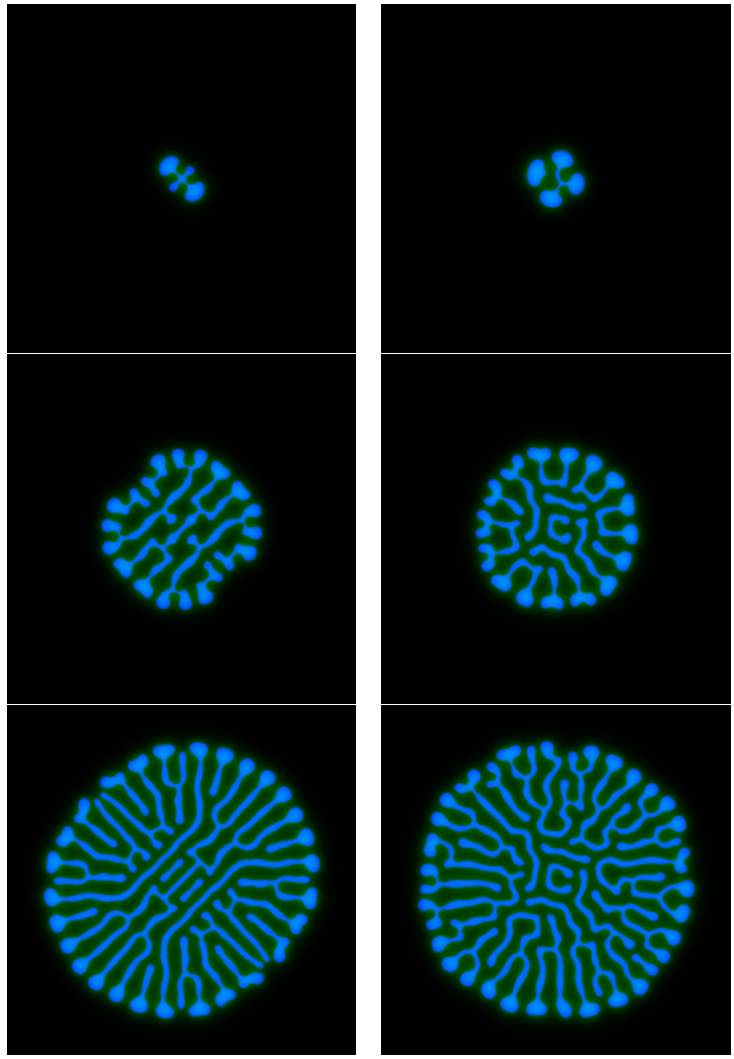


Figure II.11: *The figure shows the development of two labyrinthine patterns initiating from the same set of parameters and initial condition, except the right is formed under influence of noise. The parameters are* $L = 60$*,* $D = 0.3$ *and* $Dv = 5$*. For the left plot we have* $M = 512 = N$ *and* $\sigma = 90$*. None spontaneous excitation were appearing during the simulation. The images are made at* $T = 29$*,* $T = 72$ *and* $T = 112$*.*

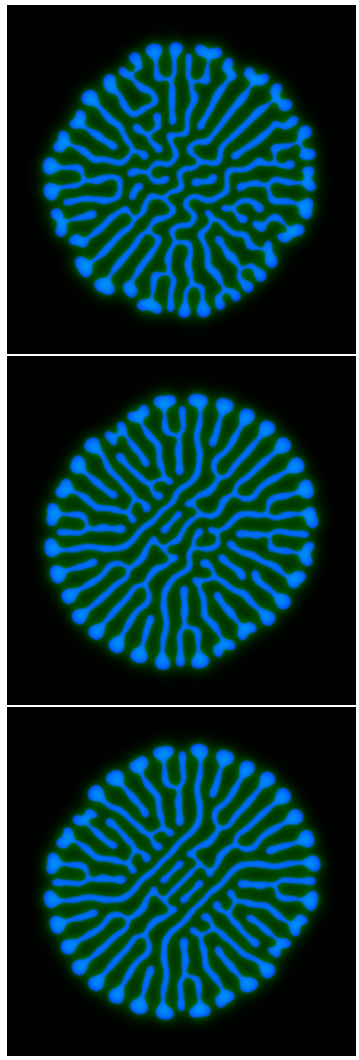


Figure II.12: *Same as fig II.11, except that the noise has been introduced at different points. The figure on top shows the system at time* $T = 112$ *with noise introduced at* $T = 10$*, second with noise introduced at* $T = 20$ *and the last figure at the bottom* $T = 40$*. Compare with the lower left image of II.11.*

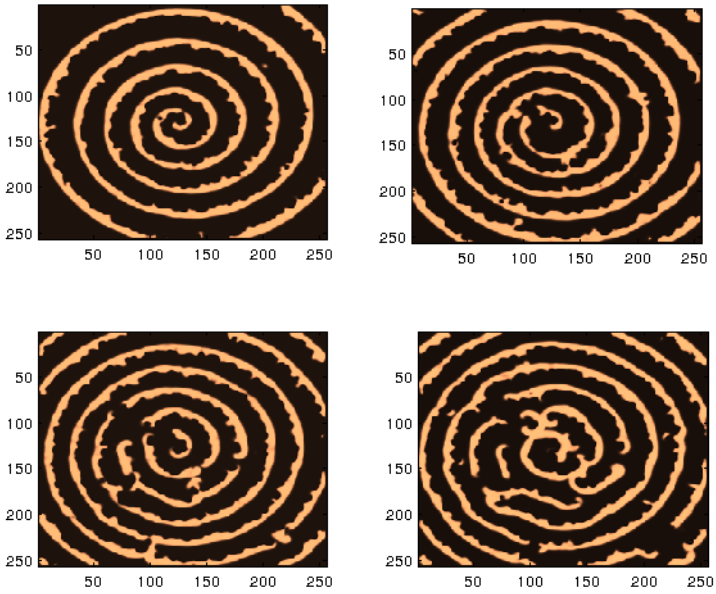


Figure II.13: *Spiral break up due to backfiring may appear in the fluctuating inhibitor field model II.5. This illustration is made with the parameters* $\sigma = 250$, $\xi = 2$, $\Delta t = 0.05$ *and* $L = 80$ *on a grid with resolution* 256^2.

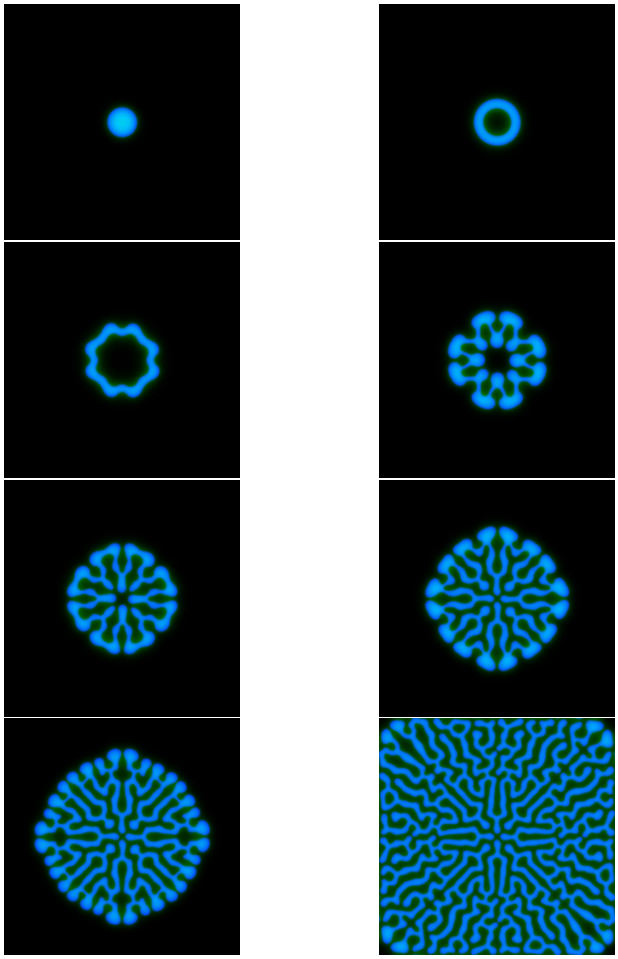


Figure II.14: *Development of a labyrinthine pattern in Wavefront. This sort of patterns will form for certain values of D_v, and every set of parameters is associated with a unique solution. The initial condition was in this case a point stimuli in the center. $D = 0.1$, $D_v = 3$ and $L = 16\pi$.*

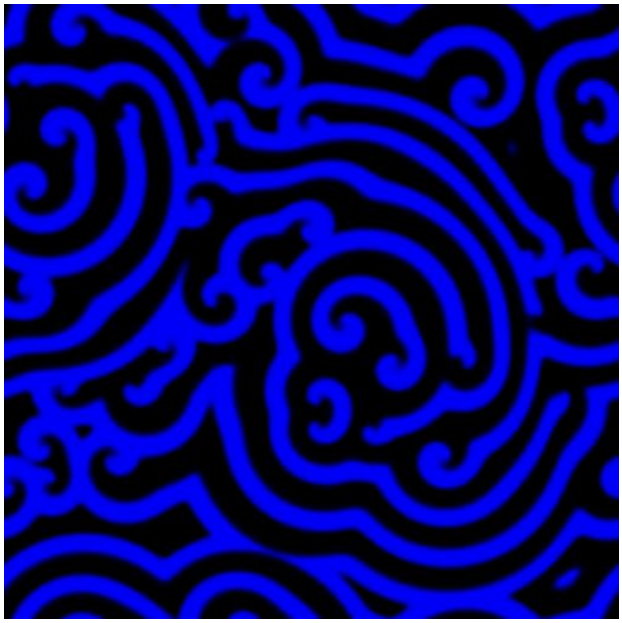


Figure II.15: *Broken spiral waves.*

III

Numerical Methods

Introduction

This chapter will be dedicated to the discussion of several solution schemes of varying complexity that will be used to solve Barkley's equation numerically. We will discuss implementation and subsequently rate them according to their stability, speed and other relevant aspects.

Relevant questions to arise prior to a simulation is how the diffusion should be implemented, how structured noise should be constructed and subsequently added, and whether an implicit or explicit scheme is the most efficient. The first solution scheme to be discussed is a simple explicit Euler method, which due to the stiffness of the $\frac{1}{\epsilon}f(u,v)$ term as well as the diffusion operator, is inefficient and slow and therefore not generally applicable to this form of equations, but will be used to in order to be compared with the more complex schemes. We will later move on to study semi-implicit schemes utilizing Fourier transforms for dealing with diffusion, and in the next chapter we will take a closer look at schemes based on exponential integrators.

Neither the explicit Euler method with spectral or difference approximation to the Laplacian is original to this paper (see [4]), nor is the semi implicit scheme which was invented by Barkley in ([7]), but this paper will make an attempt of obtaining a more rigorous analysis of their performance. On the other hand, to the author's knowledge none earlier attempts of using exponential integrators to this specific problem has been reported, and their applicability and efficiency might held relevance to diffusion-reaction problems in general.

In this chapter we will assume that we are studying the system II.1 with standard inhibitor kinetics, unless otherwise noted. However, the numerical properties of the different inhibitor kinetics is hardly different and thus not a major source of concern compared to the $f(u,v)$ term as well as the diffusion. We will also assume uniform grid size $\Delta x = \frac{L}{N}$ and constant temporal step size Δt, and all computations will be done on a square grid of 256^2 nodes on the domain $[0,L]^2$ where $L = N$ if not otherwise is noted. We will also be using a set of standard parameter values, which is $\Delta x = L/256$, $a = 0.75$, $b = 0.001$ and $\epsilon = 0.03$. T will denote a given approximation to the Laplacian.

For simplicity, when we examine methods including Fourier transforms we will be assuming periodic boundary conditions, in case of differential methods, we will use ho-

mogeneous Neumann conditions.

Constructing Noise

We will in this paper use the same sort of structured noise as Shardlow in [4], which is not a purely white noise but has a certain structure both in space and time. The derivation of this kind of noise is rather complex and beyond the scope of this paper (it is given in [4]), but we will state it's basic properties and show some examples on what it looks like and how it can be implemented.

Let our Gaussian variable $W(t, \mathbf{x})$ be defined by

$$W(t, \mathbf{x}) = \sum_{i,j \geq 0} \alpha_{ij} \mathbf{e}_{ij}(\mathbf{x}) \beta_{ij}(t) \tag{III.1}$$

where β_{ij} is an independent standard Brownian variable with mean 0 and variance $\frac{\sigma^2}{L^2}\Delta t$. The vectors $\mathbf{e}_{ij}(\mathbf{x}) = \mathbf{e}_i(x)\mathbf{e}_j(y)$ if $\mathbf{x} = (x, y)$, are the orthonormal eigenvalues of the Laplacian on $[0, L]$, given by

$$\mathbf{e}_0(x) = \sqrt{1/L}, \qquad \mathbf{e}_j(x) = \sqrt{2/L}\cos(\pi j x/L), \qquad j = 1, 2, 3, ...$$

The coefficient α_{ij} are to be derived such that the following condition concerning the spatial correlation of $W(t, \mathbf{x})$ is satisfied,

$$\mathbf{E}W(t_1, \mathbf{x}_1)W(t_2, \mathbf{x}_2) \approx C(\mathbf{x}_1 - \mathbf{x}_2)\min\{t_1, t_2\}, \qquad C(\mathbf{x}) = \frac{1}{4\xi^2}\exp\left(-\frac{\pi}{4}\frac{\|\mathbf{x}\|^2}{\xi^2}\right)$$

Here, ξ is the spatial correlation length, $\|\cdot\|$ is the standard Euclidean norm and $C(\mathbf{x})$ describes the spatial correlation. We see that if the distance $\|\mathbf{x}_1 - \mathbf{x}_2\|$ grows larger than ξ, then the correlation between the two points $\mathbf{x}_1$ and $\mathbf{x}_2$ vanishes. If $\xi \to 0$, the noise $W(t, \mathbf{x})$ approaches a standard space-time white noise process without any correlation. Notice that while the correlation decays exponentially in space, due to the $\min\{t_1, t_2\}$ term it will grow in time.

It can then be shown, assuming $\xi \ll L$, that the α_{ij} satisfying these conditions will have the form

$$\alpha_{kl}^2 = \exp\left(\frac{-\lambda_{kl}\xi^2}{\pi}\right)$$

where

$$\lambda_{kl} = (\pi/L)^2\left(k^2 + l^2\right) \tag{III.2}$$

are the eigenvalues of the Laplacian applied to $\mathbf{e}_{kl}$, i.e. $\nabla^2\mathbf{e}_{kl} = \lambda_{kl}\mathbf{e}_{kl}$. The noise $W(t, \mathbf{x})$ defined in III.1 will now satisfy

$$\mathbf{E}W(t_1, \mathbf{x}_1)W(t_2, \mathbf{x}_2) \approx C(\mathbf{x}_1 - \mathbf{x}_2)\min\{t_1, t_2\} + \text{corrections on the boundary}$$

Now that $W(t, \mathbf{x})$ is correctly defined we can move on to describe how it might be simulated. We will also here omit some of the necessary calculations, but they can still be found in [4].

In order to calculate the noise term W_{ij} as efficiently as possible, we will use the rapid decay of the Fourier coefficients in the expansion in III.1, and use the following lemma which describes the decay in expected difference between two points at a given time,

Lemma 1.2. *For an integer $M > 0$, consider the solutions $u(t)$ and $u^M(t)$*

$$du(t) = \nabla^2 u(t)dt + dW(t, \mathbf{x}), \qquad u(0) = 0$$

$$du^M(t) = \nabla^2 u^M(t)dt + dW^M(t, \mathbf{x}), \quad u^M(0) = 0$$

subject to homogeneous Neumann boundary conditions on $[0, L]^2$ where $W^M(t, \mathbf{x})$ is the Wiener process

$$W^M(t, \mathbf{x}) = \sum_{ij=0}^{M-1} \alpha_{ij} \mathbf{e}_{ij}(\mathbf{x}) \beta_{ij}(t) \tag{III.3}$$

Then,

$$\mathbf{E}\|u(t) - u^M(t)\|^2_{L_2([0,L]^2)} \leq \frac{1}{8\lambda_{M0}} \frac{L^2}{\xi^2} \exp\left(-\pi(M-1)^2 \frac{\xi^2}{L^2}\right) \tag{III.4}$$

which in [4] is listed as Lemma 1, where also the proof is to be found. Basically, it states that the difference between applying a truncated Wiener process (as in III.3) and a complete Wiener process (as in III.1) is limited, so in effect we can use the lemma to reduce the calculations of the noise. In order to decide the lowest useful M, we can compare the error produced by III.4 by the error we expect to achieve from our numerical scheme. In the case of a standard deterministic heat equation, using a five point approximation to the Laplacian, the expected error in the L_2 norm would be $O(\Delta x^2 + \Delta t)$. Again, according to Shardlow, when we use our specific type of noise this error would typically obtain a form of $O(\Delta x^2 + \Delta t^{1/2})$, and by keeping the ratio $\Delta t / \Delta x$ fixed our error estimate would be $O(\Delta x)$, and by using the lemma this says that the right hand side error of lemma 1.2, thus gives us the following bounds for M,

$$\frac{1}{8\lambda_{M0}} \frac{L^2}{\xi^2} \exp\left(-\pi(M-1)^2 \frac{\xi^2}{L^2}\right) \leq \Delta x^2 \tag{III.5}$$

This can be achieved for many reasonable values of M, e.g. if $N = L = 512$, $M = N/2 = 256$, $\epsilon = 5$, then the LHS of III.5 would yield 1.84e-6, while RHS would yield 1. Further decreasing L to $L = N/2$ (and then $\Delta x = 0.5$) would decrease LHS to 4.76e-33, which would allow a further decrease in M. Also, in its applications we would multiply W with the intensity constant σ, which will also affect the highest reasonable value of M.

The noise can be added to the Fourier transform $\hat{u}$ of a given spatial distributed system u by the following update procedure,

$$\hat{u}^n_{(i,j)} = \begin{cases} \hat{u}^n_{(i,j)} + \sigma w^n_{ij}, & i, j \leq M \\ \hat{u}^n_{(i,j)} & \text{otherwise} \end{cases}$$

An simple implementation illustrating how this form of noise can be constructed is given in appendix B. Several examples of what this kind of noise will look like is given in fig III.1.

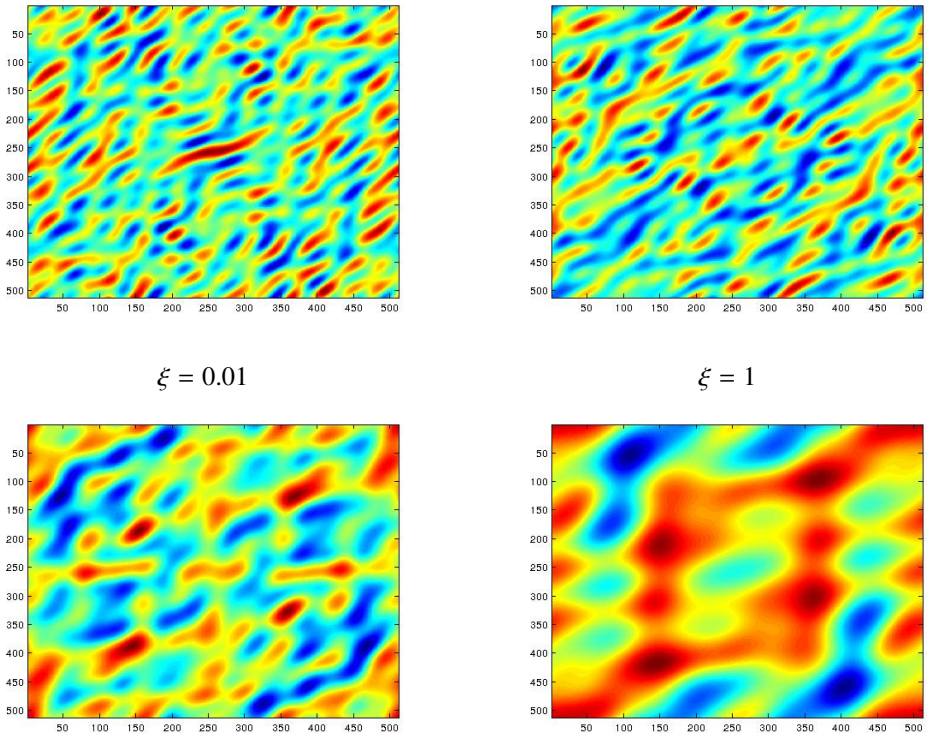


Figure III.1: *Examples of generated noise under different spatial correlation length. All images are generated in MATLAB using the code given in III.1, with parameters* $M = N/32 = 16$ *and* $\sigma = 1$. *Yellow indicates average values, blue indicates high values and red indicates low values. No diffusion (or reaction) has been present. All images are gathered from random time instances.*

Explicit Euler method

The explicit Euler method is probably the simplest and most intuitive solution scheme applicable to our problem and should not need any further introduction. In this section we will study its performance using an five-point approximation to the Laplacian.

The standard Euler scheme for Barkley's equation is given by

$$\begin{aligned} u_{ij}^{n+1} &= u_{ij}^n + DTu_{ij}^n\Delta t + \tfrac{\Delta t}{\epsilon} f\left(u_{ij}^n, v_{ij}^n\right) \\ v_{ij}^{n+1} &= v_{ij}^n + \Delta t g\left(u_{ij}^n, v_{ij}^n\right) \end{aligned} \tag{III.6}$$

Due to the slow nature of the reaction term $g(u, v)$, we will only have to consider the first equation in order to establish the stability criteria of the scheme.

Proposition 1.3. *Given a differential equation $\frac{df}{dt} = Tu$, for a differential operator T, the stability of the Euler scheme for this equation will be given by*

$$|1 + \lambda\Delta t| < 1$$

where $\lambda \in \mathbb{C}$ is the biggest eigenvalue of T.

The proof of 1.3 is considered common knowledge and is therefore omitted. It is theoretically easy to establish a stability criteria since we have the analytical expression for λ, which there is in the one dimensional case where it is simply given by $\lambda_{\max} \approx -D\frac{4}{(\Delta x)^2}$, and in two dimensions it is approximately $\lambda_{\max} \approx -D\frac{8}{(\Delta x)^2}$([23]), which will lead us to the following stability criteria,

Proposition 1.4. *Given a differential equation $\frac{df}{dt} = Tu$, for a two dimensional differential operator T, the time steps of corresponding Euler scheme for this problem will be limited by the relation*

$$\Delta t < \frac{(\Delta x)^2}{4D}$$

so if we set $\frac{(\Delta x)^2}{D} = 1$ will force the time steps to be less than 0.25, and a further increase in the diffusion coefficient D will decrease the possible length of the time steps. This has been verified empirically.

Now we can move on to analyze the stiff reaction term $\frac{1}{\epsilon} f(u, v)$, where the obvious problem is it's non-linearity. We know that u will vary between 0 and 1, and therefore we can linearize the problem into $\frac{\mathrm{d}u}{\mathrm{d}t} = f'(0)(u)$ and $\frac{\mathrm{d}u}{\mathrm{d}t} = f'(1)(u-1) - f(1)$, and examine their respective eigenvalues instead. If we do this we will in the first case obtain the following linear equation

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{v + b}{a}u = \lambda u$$

which using III.6 translates into the following criteria for stability near $u = 0$,

$$\Delta t < 2\frac{a\epsilon}{1 + b}$$

Using that, which in the introduction to this chapter is considered standard parameter values, this gives a maximum step size near $u = 0$ approximately 0.045.

Following the same procedure, near $u = 1$, we will in the case v close to 0 get the criteria

$$\Delta t < 2\frac{1}{\frac{a\epsilon}{1+b} - 1}$$

and close to $v = 1$ we will get

$$1 + \left(\frac{1+b}{a\epsilon} - 1 \right) < 1$$

We see that both of these criteria is extremely hard to achieve, and thus unconditionally stable solutions is probably not worth pursuing for this scheme.

However, simple testing shows that for the set of standard parameter values, there is very few or none symptoms of instability to observe as long as $\Delta t \leq 0.035$. The reason for this is probably that the system has some sort of self correcting behavior. As observed in fig III.2, the solution will for certain step lengths move in and out of the convergent area, but for other values diverge. Thus, the numerical instabilities can be visible to a heavy extent without causing the system to diverge, in addition, instabilities are, for reasons not yet understood, more prone to appear at certain parts of a wave than other places. E.g. if we use homogeneous boundary condition $u, v = 0$ along the edge, we will often see instabilities where the waves breaks off at the boundaries.

Taking the situation in account, no extensive or complete analysis of the performance under the explicit Euler method will be worth a large investment of time. Anyway, a few run times for *reasonable regimes of parameters*, i.e. parameters almost without visible instabilities, is given in table III.1.

	explicit		implicit	
standard parameter exceptions	Δt	time to reach $t = 100$		Δt
none	0.075	34.30s	24.63s	0.100
$D/(\Delta x)^2 = 2$	0.060	45.37s	44.90s	0.055
$\epsilon = 0.04$	0.105	26.58s	24.54s	0.110
$\epsilon = 0.04$, $D = 2$	0.070	41.20s	47.80s	0.055
$D/(\Delta x)^2 = 0.5$	0.080	35.82s	12.56s	0.220
$D/(\Delta x)^2 = 0.1$	0.080	36.27s	41.95s	0.650

Table III.1: *Some examples demonstrating the performance of the explicit and implicit Euler method. The table shows elapsed real time to reach virtual time 100. Tests were done without any form of noise.*

Semi-Implicit Euler Method

We will now turn our attention to a semi-explicit Euler method, which is a simple semi-implicit modification to the explicit version and holds the advantages of being easy to

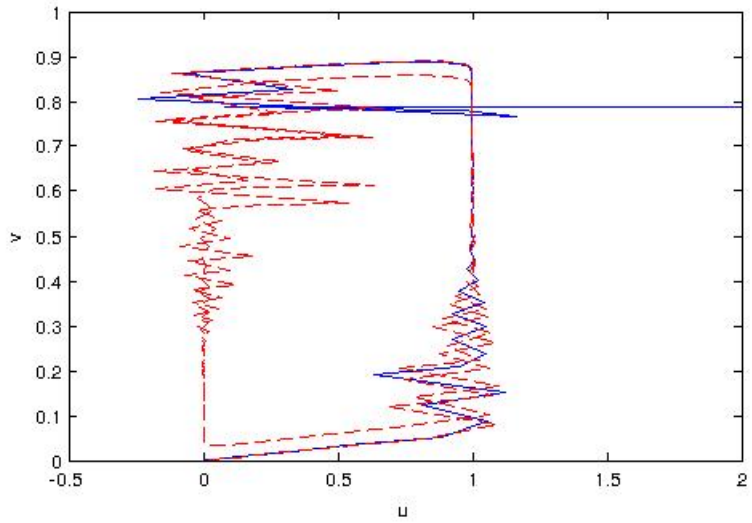


Figure III.2: *Different solution trajectories by the explicit Euler method, in the above case without influence of diffusion and using unmodified reaction terms. Instabilities caused by suboptimal parameter choice (compare with theoretical trajectory in fig II.3). The red trajectory ($a = 0.75$, $b = 0.001$ and $\epsilon = 0.013$) shows signs of unstable behavior but does not diverge. The blue trajectory (same parameters but $\epsilon = 0.01224$) breaks out of the stable domain and diverges. Compare with fig III.4.*

implement and feature good numerical stability. The scheme was invented by Barkley, see e.g. [7].

Let us momentarily ignore the diffusion and consider one time step of the explicit reaction kinetics,

$$\begin{aligned} u^{n+1} &= u^n + \tfrac{\Delta t}{\epsilon}(u^n)(1-u^n)(u^n-\phi) \\ v^{n+1} &= v^n + \Delta t(u^n - v^n) \end{aligned} \tag{III.7}$$

Where $\phi = (v^n + b)/a$. The simplest semi-implicit approximation we can use to improve III.7 is the following,

$$u^{n+1} = \begin{cases} u^n + \tfrac{\Delta t}{\epsilon}u^{n+1}(1-u^n)(u^n-\phi), & u^n \le \phi \\ u^n + \tfrac{\Delta t}{\epsilon}u^n(1-u^{n+1})(u^n-\phi), & u^n > \phi \end{cases}$$

which by simple algebra can be translated into

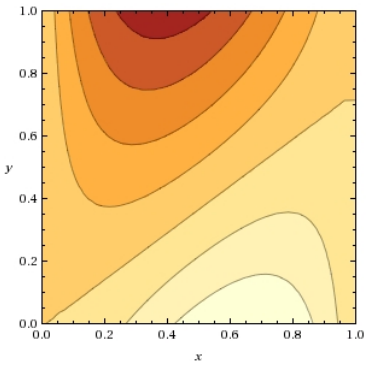
$$u^{n+1} = F(u^n, v^n) = \begin{cases} u^n / \left(1 - \tfrac{\Delta t}{\epsilon}(1-u^n)(u-\phi)\right), & u^n \le \phi \\ \left(u^n + \tfrac{\Delta t}{\epsilon}u^n(u^n-\phi)/(1+\tfrac{\Delta t}{\epsilon}u^n(u^n-\phi)\right), & u^n > \phi \end{cases}$$

and then we revive the initial scheme by adding diffusion in the following manner,

$$\begin{aligned} u_{ij}^{n+1} &= DTu^n\Delta t + F(u_n, v_n) \\ v_{ij}^{n+1} &= v_{ij}^n + \Delta t g(u_{ij}^n, v_{ij}^n) \end{aligned} \tag{III.8}$$

We could easily rewrite the update rule for v^n to be implicit, but experience tells us it would be little to gain from it.

There is a simple logic behind the construction of the semi-implicit approximation, which can be explained from figure III.3. The main idea is that we want the scheme to reflect which term of $f(u, v)$ that has the larges relative change at a given point in space. At the dark (negative) upper half of the contour plot in fig III.3, the term u^n, will have the biggest relative change, and therefore we switch it with the corresponding future time step u^{n+1}. The principle is the same for the part below the nullcline.



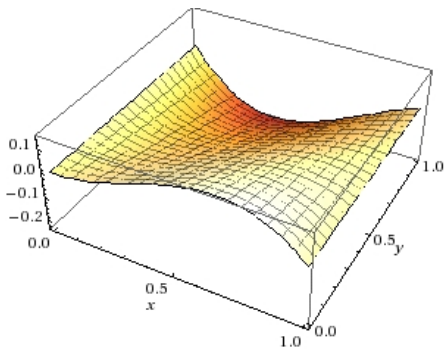


Figure III.3: The figure to the left is the contour plot of $f(u, v) = u(1 - u)(u - (v + 0.01)/0.75)$ and the figure to the right is the corresponding plot. In the contourplot, the nullcline $u = (v + 0.01)/0.75$ can clearly be seen as the diagonal line in the middle. Notice which part of the domain where the absolute value of $f(u, v)$ is at largest; this is where $\frac{du}{dt}$ will change the most, and the idea behind the semi-implicit approximation III.IV is to substitute the term of $f(u, v)$ that at this point has the largest relative change with its corresponding future time step.

We will not conduct a similar stability analysis as in the precedent section, simply due to experimental data indicating that the scheme is unconditionally stable (that is, the stability is only bounded by the stability of the approximation to the Laplacian operator) as long as the step size does not exceed the physical domain. However, fig III.4 gives a hint about the lack of exactness that will appear as Δt grows too large. We see that the trajectories will shrink and at a certain time weaken the ability to get excited, so as a heuristically founded rule, we should probably keep Δt less that 0.1.

Table III.1 shows examples of run times for both the explicit and implicit version. For the standard parameters the implicit scheme performs far better, for $D/(\Delta x)^2 > 1$ it is hardly any improvement indicating that above this point the diffusion will limit the step size, but for $D/(\Delta x)^2 < 1$ we observe that the implicit scheme far outperforms the explicit.

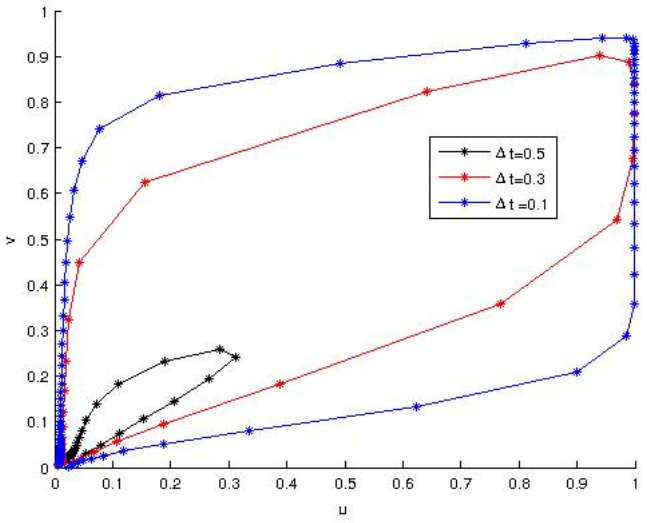


Figure III.4: *Sample trajectories with the semi-implicit Euler scheme described in III.8(standard parameter values except* $\epsilon = 0.01$ *and initial condition* $u, v = 0.1, 0$*). While* Δt *increases, the solution will become less accurate but not diverge. Compare with fig III.2. We also see that the oscilations disappear when using the semi-implicit approximation.*

Semi-Spectral Methods

The numerical experiments in the last two sections where done without any form of noise, and if we were to include the noise it would greatly delay the computational cost of the implementation almost independently of M, because we have to perform at least one additional Fourier transforms per iteration. It would probably be of great advantage if we could combine either the diffusion and/or the addition of the reaction terms within Fourier space not only in order to save time but also to obtain a more exactly solution compared to the semi-implicit scheme. It turns out there are several ways of solving Barkley's equation with spectral methods, and in this section we will describe one of the methods Shardlow described in [4].

In [4], Shardlow suggest the following solution method for Barkley's equation with additive noise, which is a combination of spectral differentiation for the the diffusion term and the explicit Euler update rule for the reaction terms. This method will be denoted as *Shardlows method*.

1. Compute the coefficients $\hat{u}_{ij}^n$ of u_{ij}^n by the FFT. Update

$$\hat{u}_{ij}^{n+1/2} = \begin{cases} \exp(-D\lambda_{ij}\Delta t)(\hat{u}_{ij}^n + \alpha_{ij} w_{ij}^n), & i, j < M, \\ \exp(-D\lambda_{ij}\Delta t)\hat{u}_{ij} \end{cases}$$

   Calculate $u^{n+1/2}$ as the inverse transform of $\hat{u}^{n+1/2}$.

2. Apply reaction terms

$$\begin{aligned} u_{ij}^{n+1} &= u_{ij}^{n+1/2} + (\Delta t/\epsilon) f(u_{ij}^{n+1/2}, v_{ij}^n) \\ v_{ij}^{n+1} &= v_{ij}^n + \Delta t g(u_{ij}^{n+1/2}, v_{ij}^n) \end{aligned} \tag{III.9}$$

Part 1 can be recognized from sample code III.1 given earlier in the appendix, where also α_{ij} and w_{ij} are specified. Why we can multiply with $\exp(-D\lambda_{ij}\Delta t)$ for adding diffusion in Fourier space might not be immediately intuitive, but it will become clearer in the next chapter when we are examining Integrating Factor methods. And of course, we could also use the semi-implicit approximation in step two. Shardlow also discusses related schemes solving Barkley's equation for the *fluctuating inhibitor field model* (the scheme above is easily adapted to other kinds of noise, so we will not describe how in details), and it is also found for which value of M which of the semi-spectral method or the pure explicit Euler method is the most effective.

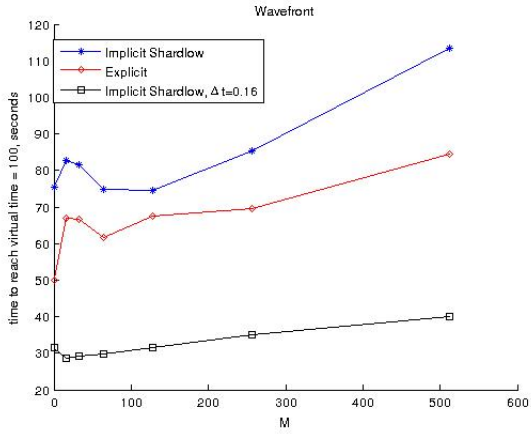
*

Obviously, this solution paradigm involves an extensive amount of Fourier transforms, so one of the prime characteristics of how useful the methods originating from this sort of thinking are, is how effective the particular implementation of Fourier transform we chose to use. For the MATLAB implementations we will use the in-build functions FFT2() and

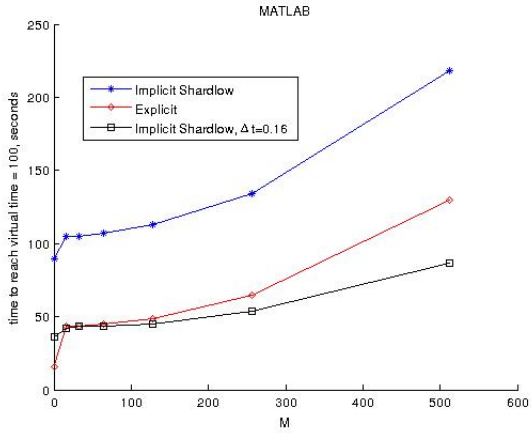
IFFT2() which are effective and easy to use. For the C++ implementations we will use the slightly more complicated third party sub routine library FFTW3.0.

A few things which is of interest is to which degree Shardlows method outperforms the simpler Euler schemes depending on M and Δt. At $M = 0$ each iterations with the Euler schemes would probably be faster due to the lack of Fourier transforms, but as M increases the difference would probably shrink.

Fig III.5 shows the performance of some of the solution schemes discussed so far as a function of M and Δt implemented both in C++ and MATLAB. The first thing to notice is how the implementation in C++ outperforms MATLAB in all cases except for the explicit Euler scheme with $M = 0$, but it is important to keep in mind that the simulations done in MATLAB does not utilize parallelization as the first implementation does. We also see that the step from no Fourier transforms to one Fourier transform causes a lot of extra work for the Explicit Euler method.



Wavefront



MATLAB

Figure III.5: *Run times as a function of M for the C++ wavefront application (without visualization) and MATLAB. The plots measures the number of seconds to reach virtual time equal 100, that is, the time it takes to reach* $i\Delta t = 100$. *Here,* $\Delta t = 0.06$ *unless otherwise noted.*

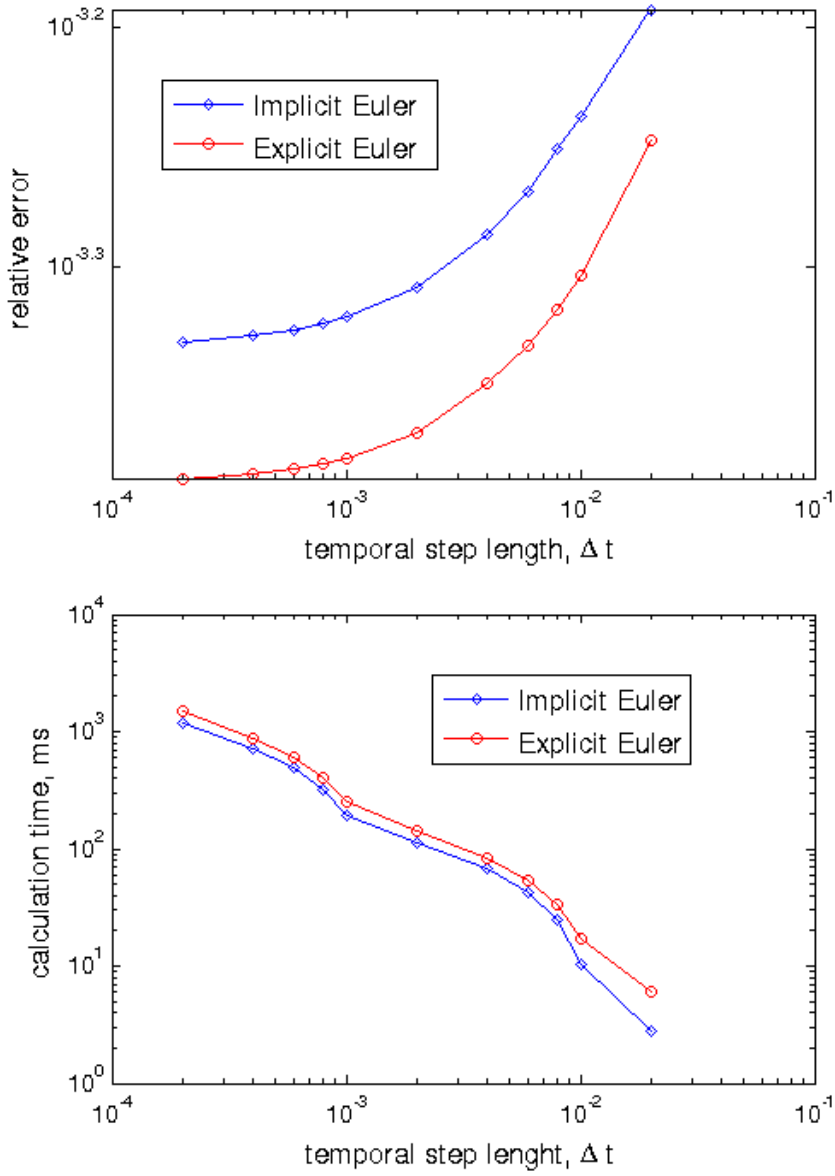


Figure III.6: *Calculation time and relative error of the explicit and implicit Euler method with five-point approximation to the Laplacian. The error is measured relative to the solution obtained with the ETDRK4B method with* $\Delta t = 0.0001$ *at* $T = 8$.

IV

Exponential Integrator Methods

In this chapter we will study the application of exponential integrator methods on Barkley's equation. Anyone trying to get a hold on exponential integrators will soon realize that the topic is a bit perplexing; the methods has usually been invented and reinvented several times causing the same method to be known under several different names. This chapter will therefore start with a short introduction to the main subclass of methods, and then move on to examine how the schemes can be applied to Barkley's equation. For a throughout and easy accessible introduction to the topic, *A review of exponential integrators for first order semi-linear problems* by B. V. Minchev and W. M. Wright ([25]) can be recommended.

A Short Introduction to Exponential Integrator Methods

Exponential Integrator Methods are a set of methods developed to obtain a numerical solution of differential equations on the form

$$u_t = f(u,t) = Lu + N(u,t) \tag{IV.1}$$

if L is a linear operator (e.g. a diffusion operator) and $N(u,t)$ is a nonlinear function of u and t. That is, Exponential Integrator methods are methods developed for problems that can be splitted into a linear and nonlinear part. The aim of the method is to solve the linear part exactly and then iterate to find a approximation to the nonlinear part.

There exists different classes of Exponential Integrator methods with widely different properties and we will later examine some of then applied to a general context, but according to [24] they all have two main features in common:

1. If $L \equiv 0$ then the scheme will reduce itself to a standard general linear scheme, in literature denoted as the "underlying standard general linear scheme".

2. If $N(u,t) \equiv 0$ then then the scheme will reproduce the exact solution of $u_t = Lu$.

These features can alone be understood as a bit diffuse, but the exact meaning will later be made clear. We can also define the Exponential Integrator methods in the same manner. Following [25],

Definition 1.1. An exponential integrator is a numerical method which involves an exponential function (or a related function) of the Jacobian or an approximation to it.

Exponential Time Differenecing methods

The philosophy behind the exponential integrators can be illuminated by studying modifications of the Euler method. As the reader should know and which is also stated earlier in this paper, Euler's explicit method for ordinary differential equations on the form IV.1 is given by

$$u_{n+1} = u_n + \Delta t f(u_n, t_n)$$

The above equation can be linearized to obtain

$$u' = f(u_n) + f'(u_n)(u - u_{n-1})$$

The linearized version has the analytical solution

$$u_{n+1} = u_n + \Delta t \phi_1(hf'(u_n, t_n)) f(u_n, t_n), \qquad \phi_1(z) = \frac{e^z - 1}{z}$$

What we have just derived is an example of a simple Exponential Integrator method called the Exponential Euler Method. For general problems on the form IV.1 this scheme will have accuracy of order two. This method has been further generalized in the direction of generalized Runge-Kutta methods and linear multi-step methods. Notice that solution schemes based on this thinking will usually use an approximation to the Jacobian (in two or more dimensions) and not the Jacobian itself. e.g we a natural approximation to the Jacobian would be L itself, yielding the following method

$$\begin{aligned} u_{n+1} &= u_n + \Delta t \phi(\Delta t L)(Lu_n + N(u_n, t_n)) \\ \Rightarrow u_{n+1} &= e^{\Delta t L} u_n + \Delta t \phi_1(\Delta t L) N(u_n) \end{aligned} \tag{IV.2}$$

This method is now know as the ETD-Euler or the ETD-Nørsett method, where ETD is an acronym for *Exponential Time Differenecing*.

There is an alternative way of obtaining the ETD-Euler method. If we multiplied IV.1 with $e^{(t_n - t)L}$ we could use the chain rule to change the initial differential equation into the following:

$$\begin{aligned} e^{(t_n - t)L} u'(t) &= e^{(t_n - t)L} Lu(t) + e^{(t_n - t)L} N(u(t)) \\ \Rightarrow \left(e^{(t_n - t)L} u(t) \right)' &= e^{(t_n - t)L} N(u(t)) \end{aligned} \tag{IV.3}$$

If we integrate IV.3 we would obtain the following representation of u_{n+1}:

$$u_{n+1} = e^{\Delta t L} u(t_n) + e^{(t_n + \Delta t)L} \int_{t_n}^{t_n + \Delta t} e^{-\tau L} N(u(\tau)) \mathrm{d}\tau$$

and now if we substitute $\tau = t_n + \theta \Delta t$ we will have

$$u_{n+1} = e^{\Delta t L} u(t_n) + \Delta t \int_0^1 e^{(1-\theta)\Delta t L} N(u(t_n + \theta \Delta t), t_n + \theta \Delta t) \mathrm{d}\theta$$

The subclass of Exponential Integrators know as ETD-methods, including the ETD-Euler method, will arise from approximating $N(u(\tau), \tau)$ by appropriate polynomials $p(\theta)$ and then integrate exactly.

Integrating factor methods

From IV.3 we can also derive another class of Exponential Integrators. Let us apply the exponential Euler method on it and then transform back we would get

$$u_{n+1} = e^{\Delta tL} u_n + e^{\Delta tL} \Delta t N(u_n) \tag{IV.4}$$

which is known as the Lawson-Euler, and the class of explicit integrators based on this philosophy is known as Lawson methods or Integrating Factor (IF) methods. The transformation we used, $v(t) = e^{(t_n - t)L} u(t)$ is known as the Lawson transformation. Of course, the solution schemes applicable on IV.3 is not limited to the explicit Euler rule, and we could instead use e.g an M stage Runge-Kutta scheme (often denoted Lawson-Runge-Kutta schemes) and the methods would then take the form

$$\begin{aligned} Y_i &= \sum_{j=1}^{i-1} a_{ij} e^{(c_i - c_j)\Delta tL} \Delta t N(Y_j) + e^{c_i \Delta tL} u_n, \qquad i = 1, 2, ..., M, \\ u_{n+1} &= \sum_{j=1}^{M} b_i e^{(c_i - c_j)\Delta tL} \Delta t N(Y_j) + e^{\Delta tL} u_n \end{aligned} \tag{IV.5}$$

and we will later see how this works out in practice.

Some of the drawbacks of the Integrating factor methods necessary to be aware of are that their stiff order is limited to one, which is due to the fact that they only use the exponential function.

*

The Lawson methods and the ETD-methods are just two classes of methods of the many constituting the currently developed Exponential Integrator methods. Among methods not mentioned here is the Lie Group methods which solves the differential equation by transforming it to an equation evolving on a Lie group, and the Generalized Lawson schemes. All Exponential Integrator methods can be generalized by the following framework, known as the Exponential General Linear methods, defined by the following internal stages and the given output approximation,

$$\begin{aligned} Y_i &= \sum_{j=1}^{M} a_{ij}(\Delta tL) \Delta t N(Y_j, t_n + c_j \Delta t) + \sum_{j=1}^{r} y_{ij}(\Delta tL) u_j^{[n-1]} \\ u_i^{[n]} &= \sum_{j=1}^{M} b_{ij}(\Delta tL) \Delta t N(Y_j, t_n + c_j \Delta t) + \sum_{j=1}^{r} v_{ij}(\Delta tL) u_j^{[n-1]} \end{aligned} \tag{IV.6}$$

The r quantities $u_1^{[n-1]}, u_2^{[n-1]} ... u_r^{[n-1]}$ are known. The coeffcient functions a, b, c,y and v are

to be represented in tableau form,

$$\begin{array}{c|ccc|ccc} c_1 & a_{11}(z) & \dots & a_{1M}(z) & y_{11}(z) & \dots & y_{1M}(z) \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ c_M & a_{M1}(z) & \dots & a_{MM}(z) & y_{M1}(z) & \dots & y_{MM}(z) \\ \hline & b_{11}(z) & \dots & b_{1M}(z) & v_{11}(z) & \dots & v_{1M}(z) \\ & \vdots & & \vdots & \vdots & & \vdots \\ & b_{r1}(z) & \dots & b_{rM}(z) & v_{r1}(z) & \dots & v_{rM}(z) \end{array} \tag{IV.7}$$

The two simplest schemes would be first the Lawson-Euler scheme,

$$\begin{array}{c|c|c} 0 & 0 & 1 \\ \hline & e^z & e^z \end{array}$$

which give rise to IV.4, and the ETD-Euler tableau

$$\begin{array}{c|c|c} 0 & 0 & 1 \\ \hline & \phi_1(z) & e^z \end{array} \tag{IV.8}$$

which gives IV.2

Solving Barkley's equation with Integrating Factor methods

As an example, it will be shown how to create an explicit Runge Kutta method without stiffness by first transforming the equation. A further discussion of such methods can be found in [20] and [21]. A particularly accessible introduction to some of the concepts used can be found in [19]. If we apply the well know Fourier transformation to Barkley's equation we would obtain

$$U_t(k,l) = -D(\pi/L)^2(k^2+l^2)U(k,l) + \tfrac{1}{\epsilon}\mathcal{F}[f(u,v)]$$

$$V_t(k,l) = -D_v(\pi/L)^2(k^2+l^2)V(k,l) + \mathcal{F}[g(u,v)]$$

where U and V are the Fourier transforms of u and v, i.e.

$$\mathcal{F}[u(x,y)] = U(k,l) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(x,y)e^{-2\pi i(kx+ly)}\mathrm{d}x\mathrm{d}y$$

and k and l are the wave modes corresponding to each spatial dimension. We will now discretize the domain into a unit square with homogeneous mesh, and use the substitution $\lambda_{ij} = (\pi/L)^2(i^2+j^2)$ so that i and j corresponds to given wave modes. If we set

$$U_{ij} = e^{-D\lambda_{ij}t}\tilde{U}_{ij}, \qquad V_{ij} = e^{-D_v\lambda_{ij}t}\tilde{V}_{ij}$$

and use the chain rule and remove the linear term of the initial transformed problem, leaving us with

$$(\tilde{U}_t)_{ij} = e^{D\lambda_{ij}t}\mathcal{F}[f(u_{ij}, v_{ij})], \qquad (\tilde{V}_t)_{ij} = e^{D_v\lambda_{ij}t}\mathcal{F}[g(u_{ij}, v_{ij})] \tag{IV.9}$$

Needless to say, the above equation is far more appealing than our starting point; the stiff diffusion term is now gone. What is left is to find a suitable solution scheme. In [20] it is shown what the solution would look like if a general M-stage Runge Kutta scheme were to be used, and in order to give a more complete intuition for what those possible solution schemes would look like, we will continue and examine this method.

Solving IV.9 with the classical explicit 4-stage Runge-Kutta method (IFRK4)

An overview of Runge Kutta methods can be found e.g. in [22] section 8.3. As in [20], denote μ_i and ν_j denote the k's corresponding with the different variables. In general, an explicit M-stage Runge Kutta method is given by

$$U_{n+1} = e^{-D\lambda\Delta t}\left[U_n + \sum_{i=1}^{M} b_i\tilde{\mu}_i\right], \qquad V_{n+1} = e^{-D_v\lambda\Delta t}\left[V_n + \sum_{i=1}^{M} b_i\tilde{\mu}_i\right],$$

where the coefficient functions are given by

$$\begin{aligned}\tilde{\mu}_i &= e^{D\lambda a_i\Delta t}\Delta t\mathcal{F}\{\tfrac{1}{\epsilon}f[\mathcal{F}^{-1}(U_{(n+c_i)}), \mathcal{F}^{-1}(V_{(n+c_i)})]\},\\ \tilde{\nu}_i &= e^{D_v\lambda c_i\Delta t}\Delta t\mathcal{F}\{g[\mathcal{F}^{-1}(U_{(n+a_i)}), \mathcal{F}^{-1}(V_{(n+c_i)})]\},\end{aligned} \tag{IV.10}$$

and

$$U_{n+c_i} = e^{-D\lambda\Delta t}\left[U_n + \sum_{j=1}^{i-1} a_{ij}\tilde{\mu}_j\right], \qquad V_{n+c_i} = e^{-D_v\lambda\Delta t}\left[V_n + \sum_{j=1}^{i-1} a_{ij}\tilde{\nu}_j\right] \tag{IV.11}$$

Notice that we have used the following substitution

$$\tilde{\mu}_i = \mu_i e^{-D\lambda t_n}, \qquad \tilde{\nu}_i = \nu_i e^{-D_v\lambda t_n}.$$

The coefficients a_i, b_{ij} and c_i comes from a Butcher tableau. In this example we will show how the solution turns out with the classical 4-stage Runge-Kutta method, defined by the tableau IV.1. We will denote this scheme as IFRK4. Combining the information in the aforementioned tableau with the the above computations, we will arrive an update rule with the following stages

$$
\begin{array}{c|cccc}
0 & & & & \\
1/2 & 1/2 & & & \\
1/2 & 0 & 1/2 & & \\
1 & 0 & 0 & 1 & \\
\hline
 & 1/6 & 1/3 & 1/3 & 1/6
\end{array}
\tag{IV.13}
$$

Table IV.1: *The classical 4-stage Runge-Kutta scheme.*

$$
\begin{aligned}
&U_n = \mathcal{F}(u), && V_n = \mathcal{F}(v) \\
&U_1 = U_n, && V_1 = V_n \\
&F_1 = \Delta t \mathcal{F}[\tfrac{1}{\epsilon} f(u_n, v_n)] && Fv_1 = \Delta t \mathcal{F}[g(u_n, v_n)] \\
&U_2 = e^{-D/2\lambda_{ij}\Delta t}(U_1 + 0.5F_1) && V_2 = e^{-D_v/2\lambda_{ij}\Delta t}(V_1 + 0.5Fv_1) \\
&F_2 = \Delta t e^{D/2\lambda_{ij}\Delta t}\mathcal{F}\left[\tfrac{1}{\epsilon} f(\mathcal{F}^{-1}(U_2), \mathcal{F}^{-1}(V_2))\right] && Fv_2 = \Delta t e^{D_v/2\lambda_{ij}\Delta t}\mathcal{F}\left[\tfrac{1}{\epsilon} f(\mathcal{F}^{-1}(U_2), \mathcal{F}^{-1}(V_2))\right] \\
&U_3 = e^{-D/2\lambda_{ij}\Delta t}(U_1 + 0.5F_2) && V_3 = e^{-D_v/2\lambda_{ij}\Delta t}(V_1 + 0.5F_2) \\
&F_3 = \Delta t e^{D/2\lambda_{ij}\Delta t}\mathcal{F}\left[\tfrac{1}{\epsilon} f(\mathcal{F}^{-1}(U_3), \mathcal{F}^{-1}(V_3))\right] && Fv_3 = \Delta t e^{D_v/2\lambda_{ij}\Delta t}\mathcal{F}\left[\tfrac{1}{\epsilon} f(\mathcal{F}^{-1}(U_3), \mathcal{F}^{-1}(V_3))\right] \\
&U_4 = e^{-D\lambda_{ij}\Delta t}(U_1 + F_3) && V_4 = e^{-D_v\lambda_{ij}\Delta t}(V_1 + Fv_3) \\
&F_4 = \Delta t e^{D\lambda_{ij}\Delta t}\mathcal{F}\left[\tfrac{1}{\epsilon} f(\mathcal{F}^{-1}(U_3), \mathcal{F}^{-1}(V_3))\right] && Fv_4 = \Delta t e^{D_v\lambda_{ij}\Delta t}\mathcal{F}\left[\tfrac{1}{\epsilon} f(\mathcal{F}^{-1}(U_3), \mathcal{F}^{-1}(V_3))\right] \\
&U_{n+1} = e^{-D\lambda_{ij}\Delta t}(U_n + \tfrac{1}{6}(F_1 + F_2 + F_3 + F_4)) && V_{n+1} = e^{-D_v\lambda_{ij}\Delta t}(U_n + \tfrac{1}{6}(Fv_1 + Fv_2 + Fv_3 + Fv_4)) \\
&u_{n+1} = \mathcal{F}^{-1}(U_{n+1}) && v_{n+1} = \mathcal{F}^{-1}(V_{n+1})
\end{aligned}
\tag{IV.12}
$$

for updating u_n. An interesting question is whether we should use the given update scheme for v_n also, or simply use the explicit Euler scheme. The last option could seem feasible due to the slow nature of the $g(u, v)$ term and that it's time consumption is very low, and we can test the different options in practice to see what errors they give. This is illustrated in fig IV.1, where we can see that for large Δt, the error is rather large but when Δt decreases the difference will diminish. This is good, because using IFRK4 for both u and v will double the time consumption compared to IFRK4 only for u and explicit Euler for v, but keep in mind that the plot would probably look very different if we included diffusion in the v term.

We will study the performance of this method compared to other methods later in this

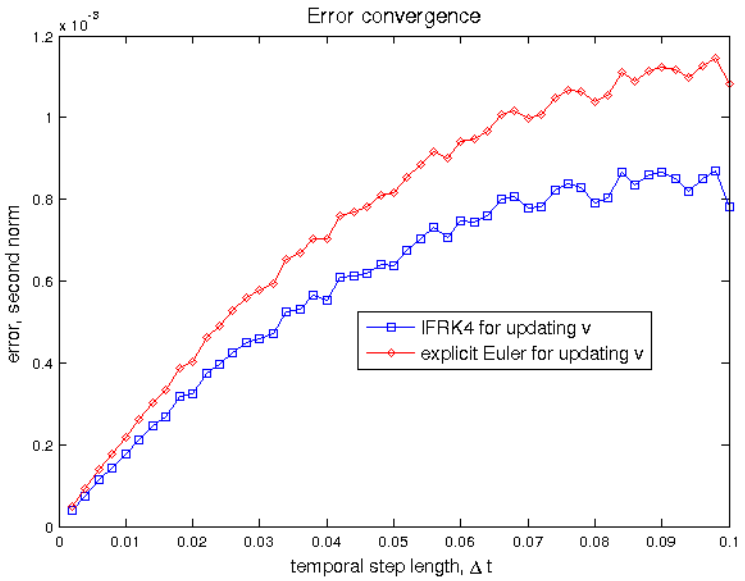


Figure IV.1: *The plot shows the relative error of the IFRK4 method with different update rules for v_n. The error is measured against the solution of the ETDRK4B method with Δt = 1e-5. The error has been measured on a grid with $N = 256$.*

chapter. A sample implementation is given in fig A.1 in the appendix.

Solving Barkley's equation with Exponential Time Differencing methods

For our purpose we will use the following scheme invented by S. Krogstad in [26] applied to the Fourier transformed verson of Barkley's equation IV.9. It is given the name ETDRK4-B as it is an improvement of the standard fourth order ETD Runge-Kutta scheme ETDRK4.

$$\begin{aligned} U_{n+1} = \quad & \phi_0(L\Delta t)U_n + \Delta t[4\phi_3(L\Delta t) - 3\phi_2(L\Delta t) + \phi_1(L\Delta t)]N(U_n, V_n)+ \\ & 2\Delta t[\phi_2(L\Delta t) - 2\phi_3(L\Delta t)]N(\mu_2, \nu_2)+ \\ & 2\Delta t[\phi_2(L\Delta t) - 2\phi_3(L\Delta t)]N(\mu_3, \nu_3)+ \\ & \Delta t[4\phi_3(L\Delta t) - \phi_2(L\Delta t)]N(\mu_4, \nu_4) \end{aligned} \tag{IV.14}$$

where (in the case of Barkley's equation) $N(U, v) = \frac{1}{\epsilon}\mathcal{F}[f(\mathcal{F}^{-1}(U), \mathcal{F}^{-1}(V))]$, and with

the following intermediate stages

$$\begin{aligned}
\mu_2 &= \phi_0(L\Delta t/2)U_n + (\Delta t/2)\phi_1(L\Delta t/2)N(U_n, V_n) \\
\mu_3 &= \phi_0(L\Delta t/2)U_n + (\Delta t/2)[\phi_1(L\Delta t/2) - 2\phi_2(L\Delta t/2)]N(U_n, v_n)+ \\
&\quad \Delta t\phi_2(L\Delta t/2)N(\mu_2, \nu_2) \\
\mu_4 &= \phi_0(L\Delta t)U_n + (\Delta t)[\phi_1(L\Delta t) - 2\phi_2(L\Delta t)]N(U_n, V_n)+ \\
&\quad 2\Delta t\phi_2(L\Delta t)N(\mu_3, \nu_3)
\end{aligned} \tag{IV.15}$$

where the exponential functions $\phi_i(z)$ are given by

$$\phi_0(z) = e^z, \quad \phi_1(z) = \frac{e^z - 1}{z}, \quad \phi_2(z) = \frac{e^z - 1 - z}{z^2}, \quad \phi_3(z) = \frac{e^z - 1 - z - z^2/2}{z^3}.$$

The update rule for V_n and the intermediates steps ν_2, ν_3 and ν_4 are equivalent.

However, implementing this is not as straight forward task as it might seems because the exponential functions $\phi_i(z)$ inherits serious numerical instabilities. The simplest illustration of this will be to examine $\phi_1(z)$ for $z \to 0$; L'Hopital's rule tells us that the limiting value in this case will approach 1, but for any digital implementation this will lead to cancellation errors. The problem is further discussed by Kassam and Trefethen in [27], where the problem is described to be related to the L matrix, which for Barkley's equation and other reaction-diffusion equations is close to singular.

Kassam and Trefethen also discuss some solutions to the problem. The simplest consist of using the Taylor expansion of the exponential function in the domain where the cancellation errors grows too prominent. However, a more interesting approach they also discuss is to use results from complex analysis. Their suggestion is to integrate the following integral around a contour Γ containing the point $z = 0$,

$$f(z) = \frac{1}{2\pi i}\int_\Gamma \frac{f(t)}{t - z}\mathrm{d}t \tag{IV.16}$$

where the function $f(t)$ is the exponential function $\phi_i(z)$ we want to evaluate. Kassam and Trefethen suggest we approximate the integral by means of the trapezoidal rule, and how this can be done is shown in the appendix section A.4.

Results and summary of observations

We will now test the performance of some of the methods discussed so far. We will measure the error at time $T = 8$ relative to the corresponding solution of the ETDRK4B scheme solved with $\Delta t = 1e{-}5$ on a grid with $N = 256$. The resulting plot showing relative error is given in fig IV.2. The required calculation time for different step sizes is given in fig IV.3, and the required calculation time corresponding to different relative errors are given in fig IV.4. All implementations are done in MATLAB. The initial condition has

been a spiral wave, and $T = 8$ for a spiral with the given parameters gives us about one rotation.

Fig IV.2 is supplemented with a reference slope illustrating $1/\Delta t$, proving that the other methods has a convergence on the form $C(\Delta t)^{-1}$ for a constant C. The plot showing the error of the ETDRK4B method appears to have some anomalies who's origin must be the fact that we are comparing it with the relative error of it self, but can we see some indications that it also fit this characteristic.

By comparing fig IV.2 and fig IV.3 we observe that ETD-Euler is the fastest but also most inaccurate scheme. If we ignore ETDRK4B, we can use fig IV.4 to see that it is actually IFRK2 that gives the best accuracy fastest.

We see from fig III.6 and fig IV.2 that although the Euler schemes are simple and easy to implement, exponential integrator methods of equal simplicity e.g. the ETD-Euler method performs much better and has by far stronger stability. Also, even though the semi-implicit Euler scheme works well for some temporal step sizes, the error it produces will grow to large and cause of solution to become erroneous. The implicit Shardlow method works quite well, but cannot compete with the higher order schemes like IFRK4 and ETDRK4B. We can also clearly see that IFRK4 converges quite quickly, this behavior is expected for exponential integrator methods of higher order.

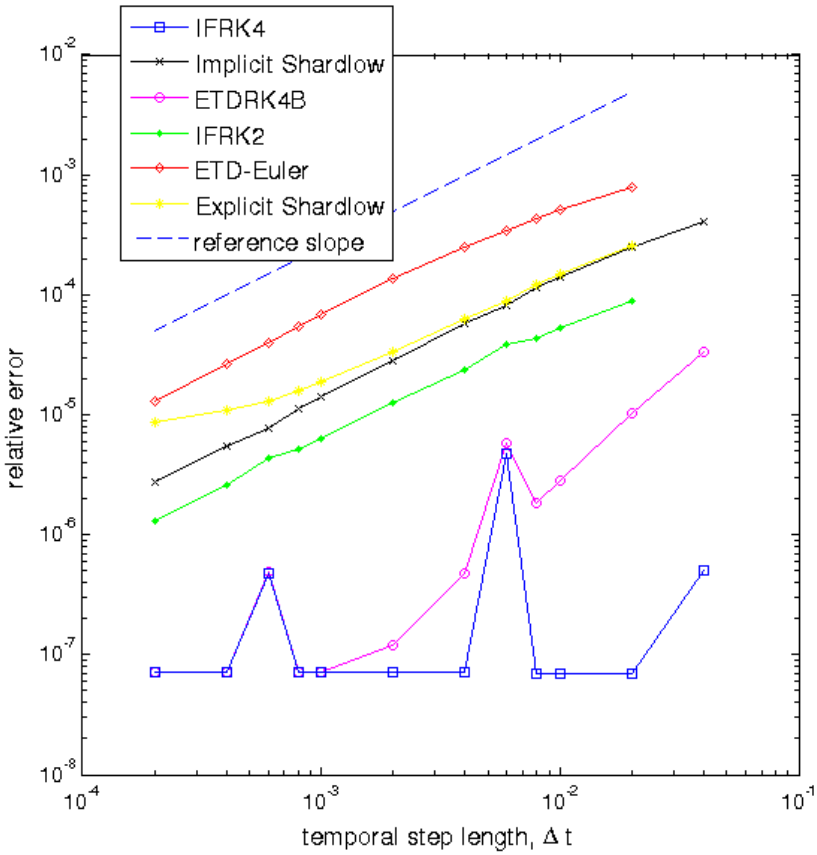


Figure IV.2: *Relative error.*

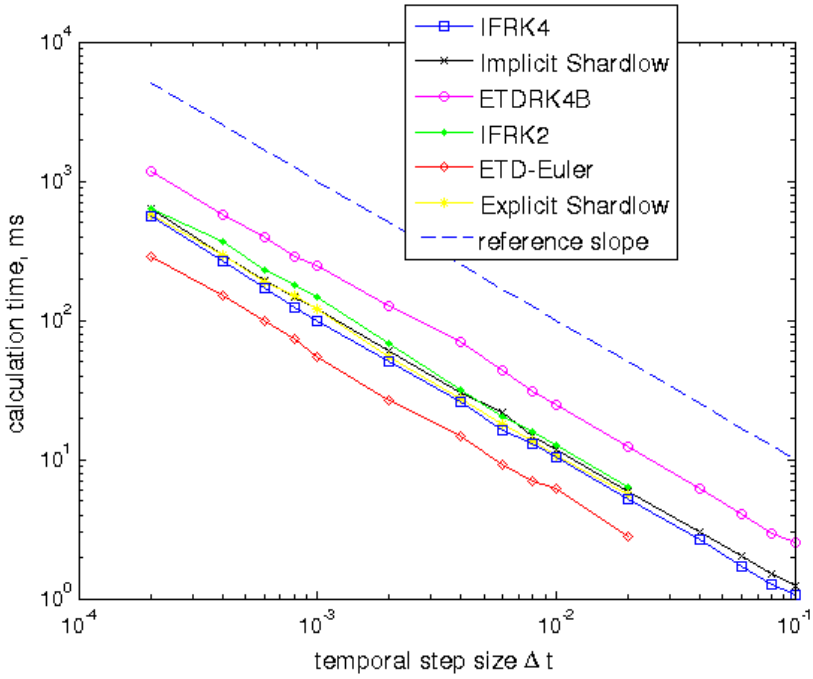


Figure IV.3: *Required calculation time.*

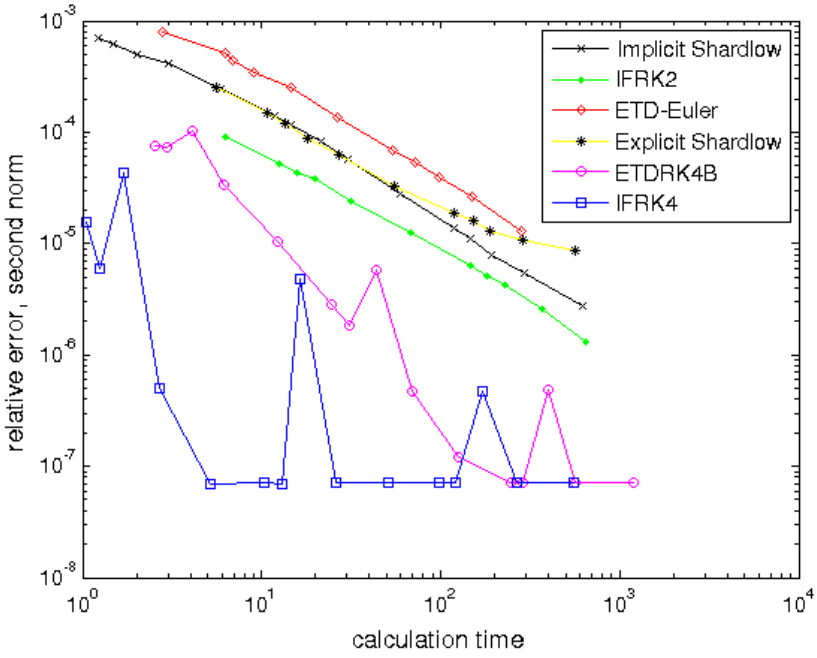


Figure IV.4: *Required calculation time plotted against relative error.*

V

Concluding Remarks

Some of the features that were planned for the Wavefront application was first of all support for the ETD methods and use of existing scientific libraries (e.g. BLAS) for matrix and vector operations. However, due to lack of time these features where abandoned, mainly because of the unexpected problem with numerical instabilities of the $\phi(z)$ functions associated with the ETD methods. Other features that could be implemented as a continuation of the work could be spiral tip tracers in order to draw meander patterns and cluster detections algorithms to collect data of formation of stable spirals in chaotic patterns. But of course, there will never be any practical endings for the multitude of functionality that can be added to such applications.

Using C++ as a practical tool for scientific purposes has positive and negative aspects. As we have seen, all implementations made in C++ are quite faster than the implementations made in MATLAB due to the fact that it utilizes paralellization and of course the hardware adaptable implementation of the Fourier transform that were used. If we had implemented a scientific numerical libraries like BLAS instead of for-loops for matrix operations, we would probably have seen an additional significant speedup. The negative aspects include difficulties in implementation; several hours was necessary to learn and understand the FFTW3.0 software for Fourier transforms, and in C++, the code for the IFRK4 method fills pages (compare with A.1) and required almost a day of work to function properly, while a fully functional implementation in MATLAB is done in half an hour. The last drawback of using C++ is difficulties in gathering of data; e.g. all the statistics about the error of the different methods represented in IV had to be done in MATLAB simply because it was the most practical.

BIBLIOGRAPHY

[1] Gill Bub, OPTICAL MAPPING OF PACEMAKER INTERACTIONS (1999).

[2] Arthur T. Winfree, VARIETIES OF SPIRAL WAVE BEHAVIOUR: AN EXPERIMENTALIST'S APPROACH TO THE THEORY OF EXCITABLE MEDIA, Chaos (1991).

[3] James P. Keener, ARRHYTHMIAS BY DIMENSION (1991).

[4] Tony Shardlow, NUMERICAL SIMULATION OF STOCHASTIC PDES FOR EXCITABLE MEDIA, 2004.

[5] Tony Shardlow, NUCLEATION OF WAVES IN EXCITABLE MEDIA BY NOISE, 2003.

[6] William O. Bray, lecture notes in Nonlinear Dynamics & Biological Systems, Lecture 6.

[7] Dwight Barkley, A MODEL FOR FAST COMPUTER SIMULATION OF WAVES IN EXCITABLE MEDIA, (1991).

[8] K. H. W. J. Ten Tusscher and A. V. Panfilov, WAVE PROPAGATION IN EXCITABLE MEDIA WITH RANDOMLY DISTRIBUTED OBSTACLES, (2005).

[9] H. Busch and F. Kaiser, INFLUENCE OF SPATIOTEMPORALLY CORRELATED NOISE ON STRUCTURE FORMATION IN EXCITABLE MEDIA, Physical Review (2003).

[10] S. Alonso, F. Sagués and J. M. Sancho, EXCITABILITY TRANSITIONS AND WAVE DYNAMICS UNDER SPATIOTEMPORAL STRUCTURED NOISE, Physical review (2002).

[11] J. Garcia-Ojalvo and L. Schimansky-Greier, NOISE-INDUCED SPIRAL DYNAMICS IN EXCITABLE MEDIA, Europhysics letters (1999).

[12] David Bradley, Zombie reaction returns from the dead, Royal Society of Chemistry (2011), http://www.rsc.org/chemistryworld/News/2011/November/21111102.asp

[13] Noboru Suzuki, Masashi Hirata and Shigeru Kondo, Traveling stripes on the skin of a mutant mouse (2003).

[14] Dwight Barkley and Daniel Margerit, ROTATION FREQUENCY OF SPIRAL AND SCROLL WAVES IN EXCITABLE MEDIA. http://www.warwick.ac.uk/staff/D.Barkley/Research/asymptotics/asymptotics.html

[15] Dwight Barkley, Sprial Pinball, http://www.youtube.com/watch?v=YGVvZVD_ddo.

[16] Dwight Barkley and Paul Wheeler, SPIRAL WAVE SPECTRA AND SPIRAL BREAKUP. http://www.warwick.ac.uk/staff/D.Barkley/Research/spiral_spectra/spiral_spectra.html

[17] Daniel Margerit and Dwight Barkley, Cookbook asymptoyics for spiral and scroll waves in excitable media, CHAOS (2002).

[18] M. Bä r and M. Eiswirth, Turbulence due to spiral breakup in a continuous excitable medium, Physical Review (1993).

[19] Lloyd N. Trefethen, Spectral Methods in Matlab, Siam (2000).

[20] R.V. Craster ans R. Sassi, Spectral algorithms for reaction-diffusion equations, (2006).

[21] Wesley B. Jones and James J. O'Brien, Pseudospectral methods and linear instabilities in reaction-diffusion fronts, CHAOS (1996).

[22] David Kincaid and Ward Cheney, Numerical Analysis third edition, AMS (2002).

[23] Arieh Iserles, A First Course in the Numerical Analysis of Differential Equations, Cambride Texts In Applied Mathematics (1996).

[24] Håvard Berland, Exponential integrators (slides), (2005).

[25] Borislav V. Minchev and Will M. Wright, A review of exponential integrators for first order and semi-linear problems, (2005).

[26] S. Krogstad, Generalized integrating factor methods for stiff PDEs, Journal of Computational Physics (2004).

[27] A.-K. Kassam and L. N. Trefethen, Fourth order time stepping for stiff PDES, Siam (2005).

A

Appendix

A. 2D IFRK4 IMPLEMENTED IN MATLAB

```
%% SOLVING BARKLEY'S EQUATION WITH AN INTEGRATING FACTOR METHOD
% u_t = D (u_xx +u_yy) +1/eps u*(1-u)*(u-(v+b)/a))
% v_t = Dv (v_xx +v_yy) + (u-v)
% This code solves the problem in two dimensions, with a spiral as the
% initial condition
%% =========  GENERIC SET UP AND INITIAL CONDITIONS  ===============
    N =512; Lt=pi*N/32; dt=0.01;
    a=0.75;b=0.001;eps=1/0.03; D=0.2; Dv=3;
    u = zeros(N); v=zeros(N);
    u(N/2:N,:)=0.9; v(:,N/2:N)=0.9; nmax=10000;
    L=eigenvalues(N,Lt); U=fft2(u); V=fft2(v);

    EuM5 = exp(-D*eig*.5*dt); Eu5  = exp(D*eig*.5*dt);
    EuM1 = exp(-D*eig*1.0*dt); Eu1  = exp(D*eig*1.0*dt);
    EvM5 = exp(-Dv*eig*.5*dt); Ev5  = exp(Dv*eig*.5*dt);
    EvM1 = exp(-Dv*eig*1.0*dt); Ev1  = exp(Dv*eig*1.0*dt);
%% ======================  TIME STEPPING LOOP ====================
    for i=1:nmax
        F1=dt*fft2(eps*tildef(u,v,a,b,I));
        F1v=dt*fft2(tildeg(u,v,I));
        U2 = EuM5.*(U+0.5*F1); V2 = EvM5.*(V+0.5*F1v);
        F2=dt*Eu5.*fft2(eps*tildef(ifft2(U2),ifft2(V2),a,b,I));
        F2v=dt*Ev5.*fft2(tildeg(ifft2(U2),ifft2(V2),I));
        U3 = EuM5.*(U+0.5*F2); V3 = EvM5.*(V+0.5*F2v);
        F3=dt*Eu5.*fft2(eps*tildef(ifft2(U3),ifft2(V3),a,b,I));
        F3v=dt*Ev5.*fft2(tildeg(ifft2(U3),ifft2(V3),I));
        U4 = EuM1.*(U+F3); V4 = EvM1.*(V+F3v);
        F4=dt*Eu1.*fft2(eps*tildef(ifft2(U4),ifft2(V4),a,b,I));
        F4v=dt*Ev1.*fft2(tildeg(ifft2(U4),ifft2(V4),I));
        U=EuM1.*(U+1/6*(F1+2*F2+2*F3+F4));
        V=EvM1.*(V+1/6*(F1v+2*F2v+2*F3v+F4v));

        v=ifft2(V); u=ifft2(U);
    end
    imagesc(u);
```

Figure A.1: IFRK4 implemented in MATLAB. This particular setup will give rise to a stagnating labyrinthine pattern.

B. Constructing noise in MATLAB

```
1
2  initiate quadratic matrix u.
3  initiate matrix eigv of eigenvalues according to III.2
4
5  while (true)
6     uhat=fft2(u);
7     uold=u;
8     uhat=fftshift(uhat);
9
10    for xx=1:N
11        for yy=1:N
12           %by uncommenting the following line we will add diffusion.
13           %uhat(xx,yy) = exp(-D*eigv(yy,xx)*dt)*uhat(xx,yy);
14
15              % add noise to frequencies less than M
16              if (xx<M && yy<M)
17                   alpha=sqrt(exp(-eigv(xx,yy)*spatl^2/pi));
18                   uhat(xx,yy)=uhat(xx,yy)+sigma*(sigma^2*dt/(L^2))*randn()*
                         alpha;
19              end
20        end
21    end
22
23      uhat=ifftshift(uhat);
24      u=real(ifft2(uhat));
25
26     pause(0.01)
27     imagesc(real(u))
28 end
```

Figure A.2: UHAT is the Fourier transform of u, EIGVAL(xx, yy) is a function returning the corresponding eigenvalue of the coordinate (xx, yy) and sigma is a constant regulating the noise intensity (the same σ used e.g. in II.2). The function RANDN() returns a number drawn from the standard normal distribution. The rest should be self-explanatory. Notice that whether or not use of the function FFTSHIFT(), which is an in-build function that "shift zero-frequency component to center of spectrum", is necessary or not is depending on how the eigenvalues is formatted. If we are to use the simple form given in III.2, the result will not be correct unless FFTSHIFT() is used.

C. 2D ETDRK4-B implemented in MATLAB

```
1  %% =========  GENERIC SET UP AND INITIAL CONDITIONS  ===============
2      N =512; L=pi*N/32; dt=0.01;
3      a=0.75;b=0.001;eps=1/0.03; D=0.2; Dv=3;
4      u = zeros(N); v=zeros(N);
5      u(N/2:N,:)=0.9; v(:,N/2:N)=0.9; nmax=10000;
6      L=eigenvalues(N,L); U=fft2(u); V=fft2(v);
7
8      phi0 = exp(-D*L*dt); phi0v = exp(-Dv*L*dt);
9      phi02=expfunc(0,1,N,-D*L,dt); phi02v=expfunc(0,1,N,-Dv*L,dt);
10     phi1=expfunc(1,0,N,-D*L,dt); phi1v=expfunc(1,0,N,-Dv*L,dt);
```

```
    phi12=expfunc(1,1,N,-D*L,dt); phi12v=expfunc(1,1,N,-Dv*L,dt);
    phi2=expfunc(2,0,N,-D*L,dt); phi2v=expfunc(2,0,N,-Dv*L,dt);
    phi22=expfunc(1,1,N,-D*L,dt); phi22v=expfunc(1,1,N,-Dv*L,dt);
    phi3=expfunc(3,0,N,-D*L,dt); phi3v=expfunc(3,0,N,-D*L,dt);

    U=fft2(u); V=fft2(v);
%% ======================  TIME STEPPING LOOP ====================
    for i=1:nmax
       NUV=fft2(eps*tildef(u,v,a,b,I)); NUVv=fft2(tildeg(u,v,I));

        nu2 = phi02v.*V+(dt/2)*phi12v.*NUVv;
        mu2 = phi02.*U+(dt/2)*phi12.*NUV;
        ifft2mu2=ifft2(mu2); ifft2nu2=ifft2(nu2);
        fftfmu2nu2 = fft2(eps*tildef(ifft2mu2,ifft2nu2,a,b,I));
        fftfmu2nu2v = fft2(tildeg(ifft2mu2,ifft2nu2,I));

        nu3 = phi02v.*V+(dt/2)*(phi12v-2*phi22v).*NUVv+...
            dt*phi22v.*fftfmu2nu2v;
        mu3 = phi02.*U+(dt/2)*(phi12-2*phi22).*NUV+...
            dt*phi22.*fftfmu2nu2;
        ifft2mu3=ifft2(mu3); ifft2nu3=ifft2(nu3);
        fftfmu3nu3v = fft2(tildeg(ifft2mu3,ifft2nu3,I));
        fftfmu3nu3 = fft2(eps*tildef(ifft2mu3,ifft2nu3,a,b,I));

        nu4= phi0v.*V+dt*(phi1v-2*phi2v).*NUVv + ...
            2*dt*phi2v.*fftfmu3nu3v;
        mu4 = phi0.*U+dt*(phi1-2*phi2).*NUV + ...
            2*dt*phi2.*fftfmu3nu3;
        ifft2mu4=ifft2(mu4); ifft2nu4=ifft2(nu4);

        U = phi0.*U + ...
            dt*(4*phi3-3*phi2+phi1).*NUV+...
            2*dt*(phi2-2*phi3).*fftfmu2nu2 + ...
            2*dt*(phi2-2*phi3).*fftfmu3nu3 + ...
            dt.*(4*phi3-phi2).*fft2(eps*tildef(ifft2mu4,ifft2nu4,a,b,I));

        V = phi0v.*V + ...
            dt*(4*phi3v-3*phi2v+phi1v).*NUVv+...
            2*dt*(phi2v-2*phi3v).*fftfmu2nu2v + ...
            2*dt*(phi2v-2*phi3v).*fftfmu3nu3v + ...
            dt.*(4*phi3v-phi2v).*fft2(tildeg(ifft2mu4,ifft2nu4,I));

        v = ifft2(V); u = ifft2(U);
    end
    imagesc(u);
```

Figure A.3: ETDRK4-B implemented in MATLAB. This particular set up will give rise to a stable spiral pattern.

D. Calculation of the exponential functions $\phi_i(z)$ by means of approximating complex integrals in MATLAB

```
function phi=expfunc(i,secondary,N,L,dt)
```

```
    M=64; %number of points in the complex half plane
    r = exp(1i*pi*((1:M)-0.5)/M);
    L = L(:); z = dt*L(:,ones(M,1))+r(ones(N^2,1),:)/2^secondary;

    if (primary==0)
        phi = exp(dt*L/2^secondary);
    elseif (i==1)
        phi = real(mean((exp(z)-1)./(z),2));
    elseif (i==2)
        phi = real(mean((exp(z)-1-z)./(z.*z),2));
    elseif (i==3)
        phi = real(mean((exp(z)-1-z-z.*z/2)./(z.*z.*z),2));
    end

    phi=reshape(phi,N,N);
return
```

Figure A.4: This is an example showing how the exponential functions ϕ_i for $i = 0, 1, 2, 3$ can be approximated using MATLAB. The function utilizes the complex integral method invented by Kassam et.al in [27].

E. Implementation of Euler-Lawson in C++

```
void wavespace::eulerlawson_update_space(double intencity, int seed,
    QString reaction_kinetics)
{
/* -------- variables for adding noise --------*/
  gsl_rng * r = gsl_rng_alloc(gsl_rng_mt19937);
  gsl_rng_set(r,time(0)*seed);
  double r1;

  int k=0;

  /* Populate the input data in row-major order */
  for (int i = 0; i < N; i++)
  {
      for (int j = 0; j < N; j++, k++)
    {
        fourier_space[k][0] = u_space[i][j];
      fourier_space[k][1] = 0;
    }
    }

  /* Transform u_space fourier_space=fft(u_space) */
  fftw_execute(transform);

  /* Determine the product u.*(I-u).*(u-gamma) */
  k=0;
  for (int i = 0; i < N; i++)
  {
      for (int j = 0; j < N; j++, k++)
    {
        fourier_space_frec[k][0] = u_space[i][j]*(1-u_space[i][j])*(
            u_space[i][j]-(v_space[i][j]+b)/a);
```

```
30        fourier_space_frec[k][1] = 0;
31      }
32      }
33
34    /* Transform the frec product to fourier space, */
35    fftw_execute(frec_plan);
36
37    /*oppdater euler_lawson step */
38    #pragma omp parallel for schedule (static)
39    for (int ww=0; ww<(N*N); ww++)
40    {
41      double exphL=exp(-D*eig[ww]*dt);
42      double phi=((exphL)-1)/exphL;
43      phi = exphL;
44      fourier_space[ww][0] = phi*fourier_space[ww][0]+dt/eps*phi*
            fourier_space_frec[ww][0];
45      fourier_space[ww][1] = phi*fourier_space[ww][1]+dt/eps*phi*
            fourier_space_frec[ww][1];
46
47      /*  applying noise
48      double intencity3=intencity*intencity*intencity
49      x = ww%spat_dist, y = floor(i/spat_dist)   */
50      if ((ww%spat_dist)<M && (int)(floor(ww/spat_dist))<M)
51      {
52        r1 = gsl_ran_gaussian(r,1.0);
53        double alpha=sqrt(exp(-eig[ww])*spatialCorr/pi);
54        fourier_space[ww][0]+= intencity3*(dt/(L*L))*r1*alpha;
55        fourier_space[ww][1]+=intencity3*(dt/(L*L))*r1*alpha;
56      }
57    }
58
59    /* transform back to spatial space*/
60    fftw_execute(transform_inverse);
61
62    /* Re-populate the u_space-matrix and scale it*/
63    k=0;
64
65    for (int i = 0; i < N; i++)
66    {
67        for (int j = 0; j < N; j++,k++)
68      {
69          u_space[i][j] = fourier_space[k][0]/(spat_dist*spat_dist);
70        v_space[i][j] = v_space[i][j]+ dt*g_reaction(u_space[i][j],v_space[i
              ][j],reaction_kinetics);
71        //v_space[i][j] =v_space[i][j]+dt*(u_space[i][j]-v_space[i][j]);
72      }
73      }
74
75    delete r;
76  }
```

Figure A.5: The code illustrates how the Euler-Lawson method is implemented in the C++ Wavefront application. The function has three input variables; *intencity* which controls the intensity of the noise, *seed* which seeds the random number generator and *reaction_kinetics* which decides which inhibitor kinetic to use when updating *v*. FFTW_EXECUTE() is an FFTW function that executes a Fourier transform and the #PRAGMA command tells the compilator to paralellize the following for-loop.

F. Wavefront: Instructions for Installation

The Wavefront application is utilizing some third party software like FFTW3 for Fourier transforms, the numerical library GSL (GNU Scientific Library) for generation of random numbers and Qt for guided user interface. All these three libraries are free and cross platform, but unless the user already have them installed, they will have to be downloaded and installed before starting Wavefront. The application is developed for Ubuntu, but has also been run successfully on OS X, and although it has not been tested, it should in theory work well on Windows.

- Qt can be found at `http://qt.nokia.com/products/`
- GSL (GNU Scientific Library) can be downloaded from `http://www.gnu.org/software/gsl/`
- The Fourier transform software FFTW3 can be obtained from `http://www.fftw.org/download.html`

Once all the dependencies are dealt with, the application can be compiled the following way,

1. Download and unzip all necessary files.
2. Open ”wavefront.pro” in Qt.
3. An import wizard will appear, hit ”done”.
4. Compile the program by hitting the ”run” button in the Qt Creator application.

Any questions or inquiries can be directed to bjorn.theisen@gmail.com.

G. Wavefront: Some notes on editing

The Wavefront source code primarily consist of three .cpp files including their respective header files. These three files are *wavefront.cpp*, *wavespace.cpp* and *phasedisp.cpp*. In *wavefront.cpp* you will find the main components determining the guided user interface of the application, and you will need to understand Qt in order to edit it. *wavespace.cpp* contains most of the numerical calculations and you will find implementations of all the methods here. This file mostly contains plain C++ and could be easily edited by people knowing the language, although somewhere it is necessary to know how to use FFTW3.0 which is not trivial but tutorials and documentation can be found on the software's home page. The variables you will need knowledge about is U_SPACE and V_SPACE which are two dimensional floating point arrays denoting the discretization of the u and v variables. All methods implemented in *wavespace.cpp* modifies these variables directly. The last file *phasedisp.cpp* contains everything related to the phase plots ((G) and (H) in fig II.7).