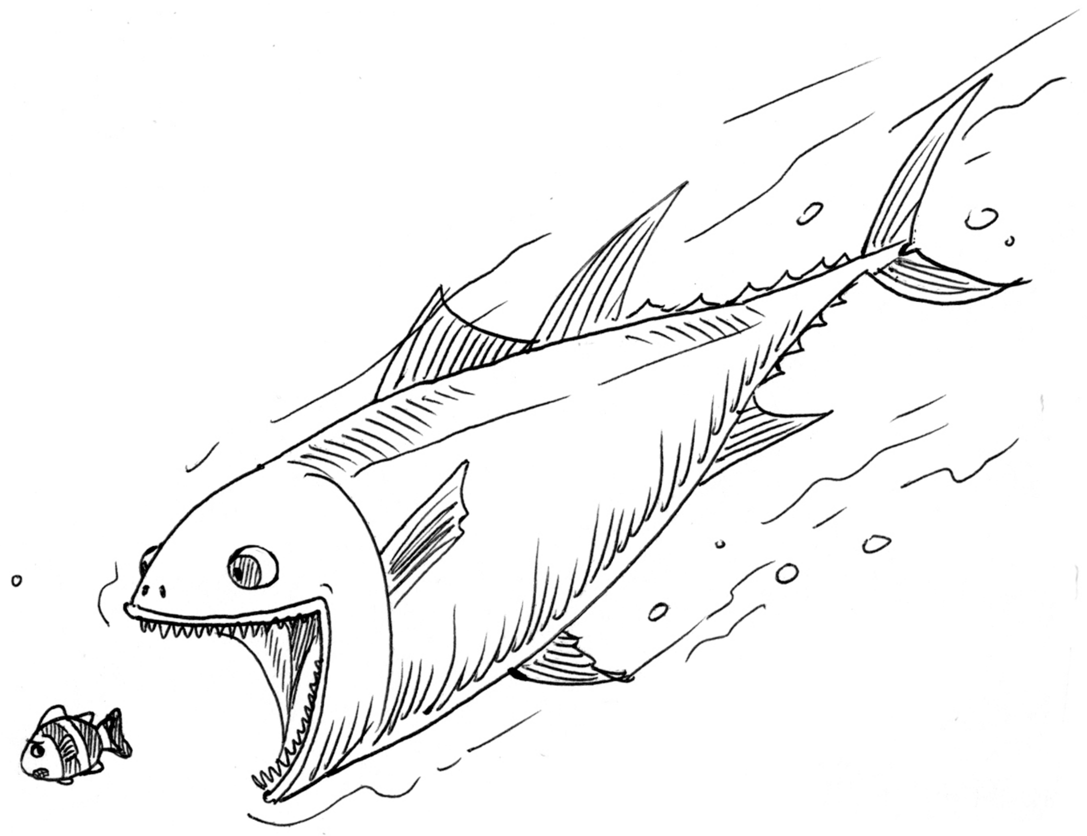
Technical University of Denmark

Fitness Taxis in Reaction-Diffusion Systems

Master Thesis

s113166 Astrid Bro Christensen

February 15, 2017



Fitness Taxis in Reaction Diffusion Systems

Master Thesis

Published at the Technical University of Denmark.

Department of Applied Mathematics and Computer Science

Richard Petersens Plads, building 324, 2800 Kgs. Lyngby, Denmark

Phone: +45 4525 3031. Email: compute@compute.dtu.dk. Web: www.compute.dtu.dk

Front page picture reference [1]

Astrid Bro Christensen (s113166)

Supervisors:

Mads Peter Sørensen

DTU Compute
Department of Applied Mathematics and Computer Science

Uffe Høgsbro Thygesen

DTU Aqua
National Institute of Aquatic Resources

Irene Louise Torpe Heilmann

DTU Compute
Department of Applied Mathematics and Computer Science

Jan Hesthaven

DTU Compute
Department of Applied Mathematics and Computer Science

Abstract

The purpose of this study is to consider a system of partial differential equations describing two spatially distributed populations in a predator-prey interaction with each other. A fitness taxis model is proposed for solving ecological problems in order to analyze the spatial structure of the population densities. This model has the shape of a reaction-advection-diffusion model, where the reaction term expresses the population dynamics. The advection term represents the taxis term, expressed as predator or prey density movement according to the gradient of the net growth rate for the specific species. Undirected movement of the species is represented by diffusion.

It is shown for which conditions Turing-like patterns, known from standard reaction-diffusion systems, are formed by taxis in the model. These conditions are divided i to three different cases for which the equilibrium solution will be unstable to spatial perturbations.

The fitness taxis model only gives rise to results as long as conditions for the third case, `CASE 3`, are met. For this case only little movement due to taxis is present. For the remaining two cases, i.e. `CASE 2` and `CASE 1`, instabilities in the solution will grow unbounded and no results can be computed from the model. A reformulation of the population dynamics is thus proposed for which the predator is able to eat prey within an area around it. This behavior is implemented with a Gaussian integration kernel.

With the kernel formulation, solutions can be obtained for values where the model meets the conditions for the remaining two cases. The spatial patterns that occur from results computed with the integration kernel turned out to change form from hexagonal arrangement of spots, as seen for simulations with very little movement due to taxis, and into stripes spanning the entire domain. At some point, movement due to taxis increases in a way that a very broad integration kernel is needed in order to compute solutions to the model. As a consequence all instabilities in the solutions gets damped, driving the system towards the stable homogeneous equilibrium solution.

Preface

This master thesis of 30 ETCS points is written under the department of Applied Mathematics and Computer Science, DTU Compute, at the Technical University of Denmark, in the period from September 2016 to February 2017. It is written in fulfillment of the requirements for acquiring an M.Sc. in Engineering.

The author of this study wishes to give a great thank to the main supervisor Mads Peter Sørensen at DTU Compute for his guidance during the project.

A special thank also goes to Irene Louise Torpe Heilmann for helping and clarifying with material from the not yet published article [2] and for always taking her time. Furthermore a big thanks to Uffe Høgsbro Thygesen for great dialogue and comments on the subject.

Thanks to Manuel Schmid for providing Julia code with detailed description used in this project. Finally a thank to Jan Hesthaven, the superviser of the project work of Manuel Schmid, for guidance in the further development of the code.

Lyngby, February 2017

Contents

1 Introduction **1**

2 Model Setup **4**

2.1 Population Dynamics 4

2.1.1 Stability of the Population Dynamics 9

2.2 Reaction-Diffusion Model 11

2.3 Fitness Taxis Model 13

2.3.1 Stability of the Fitness Taxis Model 14

2.3.2 Conditions for Pattern Formations 16

3 Algorithm **21**

3.1 Definitions and Notations 22

3.2 Spatial Discretization 23

3.3 Integrating in Time 23

3.4 Julia 26

4 Simulations **27**

4.1 Initial Conditions 27

4.2 Reaction-Diffusion Model 29

4.3 Fitness Taxis in `CASE 3` 31

4.3.1 Scenario $\gamma_u = 0.5, \gamma_v = 0.5$ 32

4.3.2 Scenario with $\gamma_u = 0.5, \gamma_v = 0.8$ 34

4.3.3 Scenario with $\gamma_u = 0.1, \gamma_v = 0.8$ 35

4.4 Effect of Initial Conditions 36

4.5 Spatio-Temporal Development 41

4.6 Summary 45

5 Adjustment to the Model **48**

5.1 Integration Kernel 48

5.2 Modified Non-Local Population Dynamics 50

5.3 Stability of the New Model 53

6 Simulations with Integration Kernel **55**

6.1 Integration Kernel 55

6.2 Comparison *with* and *without* Kernel 56

6.3 Fitness Taxis in `CASE 3` 59

6.3.1 Scenario $\gamma_u = 1.0, \gamma_v = 1.0$ 59

6.4 Fitness Taxis in `CASE 2` 60

6.4.1 Scenario $\gamma_u = 5.0, \gamma_v = 5.0$ 60

6.4.2 Scenario $\gamma_u = 10.0, \gamma_v = 5.0$ 62

6.5 Fitness Taxis in `CASE 1` 62

6.6 Fitness Taxis in `CASE 1`+`CASE 2` 67

6.6.1 Scenario $\gamma_u = 29.0, \gamma_v = 7.0$ 67

6.7 Summary 70

7 Conclusion **72**

8 Further Work **74**

A Appendix **78**

B Finite Difference Approximations **78**

B.1 4th order Centered ∇-operator Scheme 78

B.2 4th order ∇^2-operator Centered Scheme 79

C The Butcher Tableau **80**

C.1 ARK(3)4L[2]SA-ERK 80

C.2 ARK(3)4L[2]SA-ESDIRK 80

D Multigrid Solver **81**

E Simulations **82**

E.1 Spatio-temporal development **without** Integration Kernel 82

E.2 Spatio-temporal development **whit** Integration Kernel 89

F Julia Code **96**

F.1 Cases 96

F.2 Draw 98

F.3 Finite Difference matrices 99
F.4 Grid 101
F.5 Integration Kernel 102
F.6 Time Integration 102
F.7 Multigrid Functions 107
F.8 Helper Functions 111
F.9 Main Script 112

1 Introduction

In 1952 the English mathematician Alan M. Turing published an article, “The Chemical Basis of Morphogenesis”, concerning *stationary* inhomogeneous spatial pattern formation in biological reaction-diffusion systems [3]. Such systems consist of a set of nonlinear coupled equations describing the reaction kinetics for the substances. Adding diffusion for each component thereby results in a reaction-diffusion system.

The patterns or structures arise from an instability of the homogeneous equilibrium of the given system, triggered by some disturbance. The mechanism that drives the spatial inhomogeneity in these reaction-diffusion systems is the diffusion term. This means that even though the homogeneous steady state might be stable to small spatial perturbations without diffusion adding diffusion, will result in an unstable equilibrium point [4].

The components in the reaction-diffusion system turns out to be on the form of an activator-inhibitor system. This implies that the nonlinear reaction kinetics describing the substances act on each other where the activator is stimulating its own production and the production of the inhibitor. The inhibitor on the other hand inhibits the production of the activator. For patterns to arise the inhibitor must diffuse faster than the activator, i.e. the inhibitor moves quickly well ahead [4].

The patterns, today known as Turing patterns, can take on a variety of forms where stripes or hexagonal arrangements of spots are some of the best known [5]. The development of patterns and forms in nature has been well studied since the publication from Alan M. Turing [4]. Example of pattern formation in nature can be seen in the figure below, Figure 1. The left column in this figure shows the pattern on two different puffer fish and the right column shows Turing pattern simulations [6].

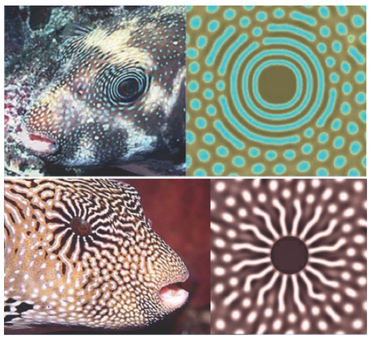


Figure 1: Pattern formation on two different puffer fish (left) and Turing pattern simulations (right)

Just as well as the patterns can take on many different forms they can similarly be seen within many different biological phenomena. An example is the observed patchiness in oceanic planktonic populations. Environmental heterogeneity can contribute to these patterns but these can also emerge due to biological interactions between phytoplankton and herbivorous copepods [7]. This exact phenomenon where spontaneous patterns are generated through interplay of reaction and movement is the very general property of the described reaction-diffusion equations used to model such systems [7].

In this study, predator and prey interaction in an oceanic environment will thus be the field of interest. In the reaction-diffusion model the two species will be equally likable to move around in every spatial direction due to the diffusive term, hence the diffusive flux is undirected. In contrast, one of the main characteristics of living systems is its ability to react to changes in the environment, and so to move towards, or away from, an environmental impact; a behavior known as *taxis* [8]. Many models of spatial dynamics of populations take taxis into account, and its importance has been recognized in modeling various biological and ecological processes. Some examples of taxis are chemotaxis, phototaxis, thermotaxis and gyrotaxis [9].

In this study, a *fitness* taxis term is proposed. The idea behind the fitness taxis is that the movement of each individual species is directed with respect to the gradient of its net growth rate. This directs the motion towards better growth conditions for each specific species. Taxis problems take the form of reaction-advection-diffusion systems not only nonlinearly coupled in the reaction part, but also in the advection part, i.e. the taxis term.

The standard time-dependent reaction-advection-diffusion model described by a set of partial differential equations is a well studied topic [9]. The model deals with the time evolution of chemical or biological species in a flowing medium such as water or air. Applications of this model-type is used in numerous fields to describe behavior such as pollutant transport in air or water. Such models can be very cumbersome to solve since the number of unknown variables in a numerical simulation can become very large. This makes it necessary to design efficient algorithms as a prerequisite to reduce the computational time to a feasible level.

The aim of this study is thus to use the well known theory behind the reaction-diffusion model for pattern generation. Modifying the reaction-diffusion model by implementing movement of the species due to fitness taxis results in a reaction-advection-diffusion model and new conditions for pattern formation must be derived. Finally, different solutions will be computed and investigated in order to look for pattern formation in this new system.

2 Model Setup

This chapter is divided into three main sections, all of which have the task of reviewing the time dependent fitness taxis model step by step. The first section will give a detailed introduction of the population dynamics of the two interacting species followed by a linear stability analysis of the system. The second section goes through the standard reaction-diffusion system and finally the third section regards the full fitness taxis model. The last part of this chapter is a linear stability analysis of the fitness taxis model and a derivation of the conditions for pattern formation.

2.1 Population Dynamics

When real-life scenarios are modeled mathematically, in particular, within the field of biomedical science, idealizations of the real-life system are always used [10]. Idealizations introduce simplifications to the model, but thereby also a deviation from the real-life scenario.

One idealization concerning this study is that the interacting species will be considered isolated, and all external conditions will be assumed to be constant. This assumption is made in order for the, described by differential equations, to have constant coefficients.

It is also important to use sufficiently large population size, in order to neglect fluctuations in the population size and in order to be able to assume that the population is a real variable. When these assumptions are satisfied only the dynamics of the average population sizes must be studied, leading to a model of deterministic differential equations. Variation among individuals are also ignored. This makes the model an *unstructured* population model, i.e. age, genotypes and so on is not taken into account.

When establishing these simplifications, the mathematical model for the dynamics of the biological system can be derived, starting by stating the dynamics for the interacting populations, i.e. the reaction kinetics. This can be described by a set of differential equations, and for this study, a general predator-prey model is stated [10].

$$\frac{\mathrm{d}u}{\mathrm{d}t} = f(u,v)u, \quad \forall t > 0 \tag{1a}$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = g(u,v)v, \quad \forall t > 0 \tag{1b}$$

The model is a system involving two species, the amount or biomass density of prey, $u(x,y,t)$, and predator, $v(x,y,t)$, respectively. Both densities are a function of the two spatial dimensions (x, y) and of time, t, but these will be mostly be omitted for further notation, as done above in Equation (1a)-(1b). Since u and v represent populations, they must be non-negative. The two functions. $f(u, v)$ and $g(u, v)$, describes the specific growth rate for each species, the prey and the predator, respectively.

The equations for the dynamics of the predator-prey interacting populations (1a)-(1b) can be generalized into the following

$$\frac{\mathrm{d}u}{\mathrm{d}t} = A(u) - B(u, v) \tag{2a}$$
$$\frac{\mathrm{d}v}{\mathrm{d}t} = -C(v) + D(u, v) \tag{2b}$$

where the four terms are functions of the corresponding population densities [10]. The function $A(u)$ expresses the positive birth rate of prey and the negative death rate of prey which is not due to the predator. The function $B(u, v)$ expresses the kill rate of prey due to the predator. The function $C(u)$ is the death rate for the predator and finally $D(u, v)$ is the reproduction rate for the predator. Models for predator-prey interacting differs in how these four terms are chosen.

In this study, the chosen model to describe the complex predator-prey dynamics is the *Bazykin model* [11]. The model, named after its inventor, was proposed in 1976 as a modification to the well known Lotka-Volterra set of differential equations for dynamics of predator and prey interacting populations [11]. The Lotka-Volterra model is written below.

$$\frac{\mathrm{d}N}{\mathrm{d}t} = \hat{\alpha}N - \hat{\beta}NP \tag{3a}$$
$$\frac{\mathrm{d}P}{\mathrm{d}t} = -\hat{\gamma}P + \hat{\delta}NP \tag{3b}$$

Here N is the prey and P is the predator and α, β, γ and δ are positive real parameters describing the interaction of the two species.

The analysis of this system in the first quadrant of phase space is well-known with one saddle-equilibrium point at the origin and a center equilibrium point, i.e. a pair of pure imaginary eigenvalues that symbolize the coexistence of the predator and prey populations [10, 12]. The system is conservative, and all of its trajectories form closed orbits, i.e. limit-cycles. This means, that the solutions are periodic, oscillating on a small ellipse around the center equilibrium point [10].

The Bazykin model is thus based on three main assumptions that differs from the Lotka-Volterra model and these are;
(1) Predator saturation,
(2) Predator competition for resources other than prey and
(3) Competition among prey [10].
The Bazykin model is also an extension of the classical and frequently used Rosenzweig-McArthur model for interacting species [13] [10]. These two models differ only in the assumption regarding predator competition for resources other than prey, since this behavior is not included in the later of the two models.

When the classical Lotka-Volterra model (3a)-(3b) is modified by changing the shape of the functions $A(u), B(u,v), C(v), D(u,v)$ in order to achieve a better description of the biological behavior of the interacting species, this will lead to a different phase portrait. The effect of the modification will either be stabilizing or destabilizing depending on each modification of the model. Only a combination of stabilizing and destabilizing effects of the model will influence the equilibrium points. For two or more stabilizing factors considered simultaneously, or for two or more destabilizing factors considered simultaneously, no new results will appear. A combination of only stabilizing factors will lead to stability of the unique equilibrium, and the combination of destabilizing factors will lead to instability.

The first modification to the classical Lotka-Volterra model (3a)-(3b), assumption (1), incorporates saturation of the predator so that neither the predation rate nor the predator reproduction rate is capable of growing infinitely with the growth of the amount of prey. This behavior is a destabilizing factor of the former mentioned equilibrium. The second modification, assumption (2), is unrelated to the amount of prey, and thereby regards only limited external resources of predators, so that the predator population cannot grow infinitely not even with unlimited food supply available. The last modification, assumption (3), tells that the population size of the prey cannot increase infinitely [11]. Both of the last two modifications are factors that stabilize the equilibrium.

The Bazykin model thereby incorporates a destabilizing and two stabilizing effects, which give rise to new fixed point in the system compared to the the classical Lotka-Volterra model (3a)-(3b). Finally, the Bazykin model expressing the populations dynamics can be seen below.

$$\frac{\mathrm{d}N}{\mathrm{d}t} = \hat{r}N\left(1-\frac{N}{\hat{K}}\right) - \frac{\hat{a}NP}{N+\hat{b}} \tag{4a}$$

$$\frac{\mathrm{d}P}{\mathrm{d}t} = -\hat{\mu}P - \hat{c}P^2 + \hat{\epsilon}\frac{\hat{a}NP}{N+\hat{b}} \tag{4b}$$

Note that for $\hat{c} = 0$ this model reduces to the Rosenzweig-McArthur model. These ordinary differential equations (4a)-(4b) are scaled to get dimensionless equations. The variables are scaled as follows [14]:

$$t = \frac{1}{\hat{\mu}}\tau, \quad N = \hat{b}u, \quad P = \hat{\epsilon}\hat{b}v$$

$$r = \hat{r}\frac{1}{\hat{\mu}}, \quad K = \hat{K}\frac{1}{\hat{b}}, \quad a = \hat{a}\frac{\hat{\epsilon}}{\hat{\mu}}, \quad c = \hat{c}\frac{\hat{\epsilon}\hat{b}}{\hat{\mu}}$$

This reduces the dimensional equations (4a)-(4b) to the dimensionless ones seen stated below, where the non-dimensionalized Bazykin model is written in the general form (1a)-(1b) [2].

$$f(u,v) = r\left(1-\frac{u}{K}\right) - \frac{av}{u+1} \tag{5a}$$

$$g(u,v) = -1 - cv + \frac{au}{u+1} \tag{5b}$$

This model has four dimensionless parameters, $a, c, r, K > 0$. For the purpose of biological applications, all parameters are assumed to be real and strictly positive.

Due to the predator-prey interaction, the growth rate of one population will decrease as the growth rate of the other population increases [15]. The nature of the interactions between the two populations is determined by the sign of the partial derivatives $\partial f(u,v)/\partial v$ and $\partial g(u,v)/\partial u$ over the entire space. For this specific model (5a)-(5b), $\partial f(u,v)/\partial v$ is negative over the entire domain and $\partial g(u,v)/\partial u$ is positive. This behavior implies that the predator is an inhibitor and the prey is an activator [4].

From the dimensionless equations (5a)-(5b) the equations for the four general terms $A(u), B(u,v), C(v)$, and $D(u,v)$, which constitutes the description of the specific population dynamics, can be read.

The fertility/reproduction rate for the prey is described by $A(u) = ur(1-u/K)$, and this exact expression is also known as the logistic equation [10]. It expresses the prey-prey interaction, and so describes the growth conditions for the prey. The coefficient r is the birth rate for the prey, and denotes the rate of exponential population growth with smaller populations sizes. The constant K is the stationary population density, determined by available resources, denoted the carrying capacity, and with this the population is restricted in size by some necessary, but limited resource, and can thereby not grow infinitely. With no predator present, i.e. $v = 0$, the growth rate of the prey population size has its maximum at $u = K/2$. At $u = K$ the carrying capacity is reached, and the prey population can no longer grow.

The function $B(u, v)$ describes the predation rate for which prey is consumed. It is a monotonically non-decreasing function of each argument, u and v. The function can be rewritten as $B(u, v) = \eta(u)v$, where $\eta(u) = au/(1 + u)$. In ecology, $\eta(u)$ is known as the *functional response*. This is the functional reaction of the predator to the prey population density. Since the equation can be written on the form $B(u, v) = \eta(u)v$, competition among predators for prey is excluded, and so individual predators do not compete or interact. This means that the rate of prey consumption per unit of predator density is independent of the density of the predator population. The shape of the predation function refers to the dependence of the rate of predation on the density of prey. At small population densities, the individual predators do not hinder each other and will catch the prey independently. At large densities, the predators will remove as many prey as possible from the prey population, and at further growth of the predator population this will not increase the total amount of prey consumed by all the predators. The maximum predation rate when prey is abundant is thereby the size of the constant a. So for this exact form of $B(u, v)$ the predation rate depends on the density of prey.

The fertility function, also called the numerical response, $D(u, v)$, for the predator, is equal to the function for the predation rate, $B(u, v)$, in the non-dimensionalized model (5a)-(5b). Consequently, all prey biomass is converted into predator biomass. This is in contrast to the dimensional model (4a)-(4b), in which the biomass of prey is assumed to be proportional to the biomass of predator, i.e. the assimilation effectiveness denoted $\hat{\epsilon}$. As already stated in the explanation of the function $B(u, v)$, the fertility function for the predator will not increase infinitely with increasing prey population. When the amount of prey is small, the fertility function is proportional to the product of the number of preys and predators, but when the amount of prey is large, the fertility function is solely determined by the quantity of predators. When the number of prey is sufficiently large the rate of growth of the predator population will be determined exclusively by its own magnitude. The two terms $B(u,v)$ and $D(u,v)$ are the terms that represent the dynamics of the interacting predator and prey populations.

In the form of the function $D(u,v)$, only competition for prey is analyzed for the character of the predator. In reality, however, a predator population may also be limited by shortage of other resources such as the size of the habitat suitable for the predator to live and reproduce in [10]. Predator competition for resources other than prey is therefore implemented in the function $C(v) = v + cv^2$ describing the density dependence in the specific growth rate of predators. This results in a decrease in the rate of change in predator density. The constant c represents competition for resources other than prey, and the populations size of predator can thereby not grow infinitely.

2.1.1 Stability of the Population Dynamics

The stability of the population dynamics (5a)-(5b) can be determined from the stability of a linear approximation close to the interacting equilibrium. The model has several equilibria found for which $f(u^*,v^*)u^* = g(u^*,v^*)v^* = 0$. One trivial equilibrium point at the origin, $(u^*,v^*) = (0,0)$, where both species are extinct and one, $(u^*,v^*) = (K,0)$, where the predator goes extinct and the prey reaches its carrying capacity.

When neither the prey nor the predator are extinct, i.e. $u^*, v^* > 0$, the model has *three* additional equilibrium points or *one* additional equilibrium point, depending on how the four parameters, a, c, r and K, in the model (5a)-(5b) are chosen.

The linearized system around $(u^*,v^*) > 0$ is expressed by

$$\mathbf{A} = \begin{bmatrix} f_u^* u^* & f_v^* u^* \\ g_u^* v^* & g_v^* v^* \end{bmatrix} = \begin{bmatrix} \left(-\frac{r}{K} + \frac{av^*}{(u^*+1)^2}\right) u^* & -\frac{au^*}{u^*+1} \\ \frac{av^*}{(u^*+1)^2} & -cv^* \end{bmatrix} \tag{6}$$

using the notation $f_u^* = \dfrac{\partial f(u^*,v^*)}{\partial u^*}$. The stability of the population dynamics can be determined from the trace and determinant of the community matrix, $\mathbf{A}$, and these must obey the following inequalities [4].

$$\mathrm{Tr}[\mathbf{A}] = u^* f_u^* + v^* g_v^* < 0, \quad \det(\mathbf{A}) = u^* f_u^* v^* g_v^* - u^* f_v^* v^* g_u^* > 0$$

The four parameters, a, c, r and K, that arise from the non-dimensionalized Bazykin model (5a)-(5b) can for any positive value be chosen freely and for this study the following values are used

$$a = 5, \quad c = 1.5, \quad r = 2.8, \quad K = 28/3 \tag{7}$$

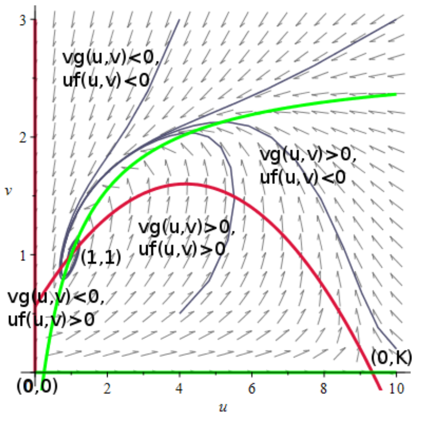
For these exact values there is only *one* equilibrium, $(u^*, v^*) = (1, 1)$, where both species co-exist. Inserting this point in the equations for the population dynamics (5a)-(5b) together with the specified parameter values results in $f(1, 1) = g(1, 1) = 0$. Inserting the specified values in the linearized system from Equation (6) the linear operator $\mathbf{A}$ becomes

$$\mathbf{A} = \begin{bmatrix} 0.95 & -2.5 \\ 1.25 & -1.5 \end{bmatrix}$$

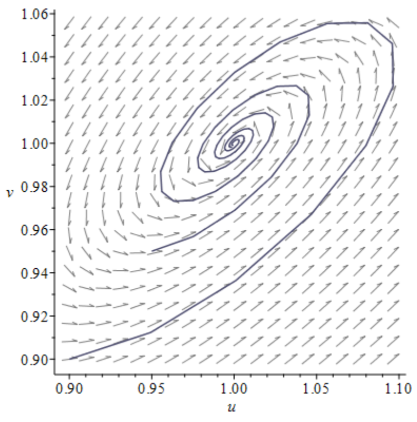
Since the trace is negative, $\text{Tr}[\mathbf{A}] = -0.55$, and the determinant is strictly positive, $\det(\mathbf{A}) = 1.7$, the equilibrium point is *stable*. The real parts of the eigenvalues are negative as well, which also confirms that this equilibrium point is linearly stable.

A phase diagram can be made for the population dynamics with the specified parameter values. This is done in Figure 2a, showing the phase space $M = \{(u, v) : u \geq 0, v \geq 0\}$ with the nullclines and the corresponding vector field. Due to the specified values of the four parameters in the non-dimensionalized model (5a)-(5b), this model only has *three* equilibrium points whereas only *one* exists for the coexistence of both species. These three equilibria are all seen in Figure 2a as well, and these are the points where the green and red nullcline intersect.

The density of the prey is on the horizontal axis and the density of the predator is on the vertical axis. The second figure, Figure 2b, shows the same phase diagram, but now there is a zoom in on the stable equilibrium point. This makes it easier to see, that the motion is directed counterclockwise in skew spirals inwards towards the stable equilibrium. The shape of this motion and how the nullclines interxects are exactly what determines the Turing patterns [4].



(a) Phase diagram of the dimensionless Bazykin predator-prey model (5a)-(5b). The red graphs are the nullclines where $uf(u,v) = 0$ and the green graphs are the nullclines $vg(u,v) = 0$.



(b) Zoom in on the phase diagram from Figure 2a around the stable equilibrium point (u^*, v^*).

Figure 2: Full and zoomed phase diagram of Bazykin predator-prey model

It is easily seen in Figure 2a that all trajectories starting from a given point $u, v > 0$ in the phase space will thereby spiral inwards to the stable equilibrium point $(u^*, v^*) = (1, 1)$ as expected.

2.2 Reaction-Diffusion Model

Now that the population dynamics for the two interacting species are stated, a diffusion term can be added. This is added with partial derivatives describing the spatio-temporal dynamics, allowing both the predator and prey to disperse by diffusion.

Biomass flux due to diffusion can be expressed by Ficks' first law, which states that the flux, **J**, here of predator and prey biomass, is proportional to the gradient of the concentration of the biomass. In two spatial dimensions the flux for the prey is the following

$$\mathbf{J} = -D_u \nabla u(x, y, t) \tag{8}$$

Diffusion transports matter from regions of high concentration to regions of low concentration which is indicated by a minus sign in the expression. The diffusion flux vector, **J**, measures thus the amount of prey that will flow through a unit area during a unit time interval, and D_u is the constant diffusion coefficient for the prey.

Let s be an arbitrary surface enclosing a volume Ω. The change in density per time unit for prey in Ω is equal to the rate of flow of the prey across s into Ω plus any production of prey inside of Ω. This can be expressed by the general conservation equation written below.

Change of prey per unit time = Total inflow - Total outflow + Produced

The flow is determined by the diffusion and together with the production of prey, the reaction term, $R(x, y, t, u) = f(u, v)u$, this results in the following expression for the change of prey

$$\begin{aligned}\frac{d}{dt}\int_{\Omega} u(x,y,t)\mathrm{d}\Omega &= \int_{\Omega} R(x,y,t,u)\mathrm{d}\Omega - \int_{\partial\Omega} \mathbf{J}\cdot\mathbf{n}\mathrm{d}s \\ &= \int_{\Omega} R(x,y,t,u)\mathrm{d}\Omega + \int_{\partial\Omega} D_u\nabla u(x,y,t)\cdot\mathbf{n}\mathrm{d}s \qquad (9)\end{aligned}$$

with $\mathbf{n}$ being the outward normal vector. The divergence theorem relates the flow, i.e the flux, of a vector field through a surface to the behavior of the vector field inside the surface and this is expressed below.

$$\int_{\Omega} \nabla\cdot\mathbf{F}\mathrm{d}\Omega = \int_{\partial\Omega} \mathbf{F}\cdot\mathbf{n}\mathrm{d}s \qquad (10)$$

Applying this theorem with $\mathbf{F} = D_u\nabla u(x, y, t)$ to the surface integral from Equation (9) this becomes

$$\frac{d}{dt}\int_{\Omega} u(x,y,t)\mathrm{d}\Omega = \int_{\Omega} R(x,y,t,u)\mathrm{d}\Omega + \int_{\Omega} \nabla(D_u\nabla u(x,y,t))\mathrm{d}\Omega \qquad (11)$$

This can be rewritten and expressed as the following

$$\begin{aligned}&\int_{\Omega}\left[\frac{\partial u(x,y,t)}{\partial t} - R(x,y,t,u) - \nabla(D_u\nabla u(x,y,t))\right]\mathrm{d}\Omega = 0 \Leftrightarrow \\ &\frac{\partial u(x,y,t)}{\partial t} = R(x,y,t,u) + \nabla(D_u\nabla u(x,y,t)) \qquad (12)\end{aligned}$$

For this study, the diffusion coefficient D_u is a scalar, and so the amount of diffusion in all coordinate directions is the same, and hence, no cross diffusion takes place. Since the diffusion coefficient D_u is constant, $\nabla(D_u\nabla u(x, y, t)) = D_u\nabla^2 u$.

The same procedure can be used to derive the conservation equation for the predator, v. Doing so and inserting the reaction terms from the population dynamics (5a)-(5b) this gives rise to the well-known system type; a reaction-diffusion system.

$$\frac{\partial u}{\partial t} = f(u,v)u + D_u\nabla^2 u, \quad \forall t > 0 \qquad (13a)$$

$$\frac{\partial v}{\partial t} = g(u,v)v + D_v\nabla^2 v, \quad \forall t > 0 \qquad (13b)$$

Here D_u and D_v are the two diffusion parameters, both constant and non-negative. This system-type is linear in the diffusion terms and non-linear in the reaction terms due to the form in (5a)-(5b). As a consequence of the activator-inhibitor dynamics, the predator must diffuse much faster than the prey in other to form patterns, and so $D_v > D_u$.

2.3 Fitness Taxis Model

To improve the proposed reaction-diffusion model (13a)-(13b) a taxis term is thereby added to the reaction-diffusion model with the shape of an advection-like term. The advection term gives a description of how the fluid or air transports a conserved quantity via bulk motion. Mathematically the motion of the fluid is described by a vector field, $F(x, y, t)$, and the transported material is then described by a scalar field that shows its distribution over space.

For this study, the advection term will be a measure of the *fitness taxis* of the system. Adding the taxis term directs the motion of the predator and prey *up* the fitness gradient, i.e. $\nabla f(u, v)$ and $\nabla g(u, v)$, respectively. This then directs the motions towards better growth conditions for each species, which is an expected behavior from an ecological point of view. Since there is an overall flow, there is an associated flux, an advective flux. This flux is written below for the prey density, u.

$$\mathbf{J}_{\text{fitness taxis}} = \gamma_u u(x, y, t)\nabla f(u, v) \tag{14}$$

Here γ_u is the advection parameter and $\gamma_u \nabla f(u, v)$ describes the speed for which the prey in the flowing media is carried along with. The standard advection form only considers how the species u is carried along with the speed γ_u, which would be expresses as $\gamma_u u$. With the form of the advection-like flux as stated in Equation (14) the advection term is non-linear, and the direction of the movement of each species is improved from an ecological point of view.

The total mass transport will now be $\mathbf{J} = \mathbf{J}_{\text{diffusion}} + \mathbf{J}_{\text{fitness taxis}}$, which expresses the total outflow of prey. Taking again Equation (9), using the divergence theorem from Equation (10) with $\mathbf{J} = \mathbf{F}$, the system now becomes

$$\frac{d}{dt}\int_\Omega u(x, y, t)\mathrm{d}\Omega = \int_\Omega R(x, y, t, u)\mathrm{d}\Omega - \int_{\partial\Omega} \mathbf{J}\cdot\mathbf{n}\mathrm{d}s \Leftrightarrow \tag{15}$$

$$\int_\Omega \left[\frac{\partial u(x, y, t)}{\partial t} - R(x, y, t, u) - \nabla(D_u \nabla u(x, y, t)) + \nabla(\gamma_u u(x, y, t)\nabla f(u, v))\right]\mathrm{d}\Omega = 0$$

Implementing this new behavior, this improvement, for both species, results in a new model, a fitness taxis reaction-diffusion model, for further notation denoted only *fitness taxis model.*

$$\frac{\partial u}{\partial t} = f(u,v)u - \gamma_u \nabla \cdot u \nabla f(u,v) + D_u \nabla^2 u, \quad \forall t > 0 \tag{16a}$$

$$\frac{\partial v}{\partial t} = g(u,v)v - \gamma_v \nabla \cdot v \nabla g(u,v) + D_v \nabla^2 v, \quad \forall t > 0 \tag{16b}$$

In the fitness taxis model two new parameters, $\gamma_u, \gamma_v \geq 0$, are introduced in the advection term. The advection, or taxis, term will inflect the occurrence and the form of the Turing patterns, depending on the parameter values of γ_u and γ_v. For $\gamma_u = \gamma_v = 0$ the model is reduced to the standard reaction-diffusion model (13a)-(13b).

2.3.1 Stability of the Fitness Taxis Model

Now that the fitness taxis model has been established (16a)-(16b), a linear stability analysis will be performed for the full system. It is necessary, as it will provide conditions for the diffusion and advection driven instability due to small *spatial* perturbations of the steady state, and hence the initiation of spatial pattern formation.

The steady state spatially inhomogeneous solution, the Turing patterns, appears because of the linearly unstable eigenfunctions that are growing exponentially with time. Evetually they will be bounded by the nonlinear terms in the reaction-diffusion-advection equations, i.e. homgeneous patterns are formed [4].

The stable equilibrium point, (u^*, v^*), for the population dynamics (5a)-(5b) determined in Section 2.1.1 will also be an equilibrium solution of the fitness taxis model, but might not be a stable solution [2]. By linearizing the full system around this equilibrium point, (u^*, v^*), the long term growth or decay of the solution, i.e. how instabilities will blow up or decay when the equilibrium point is perturbed, can be investigated.

The equilibrium (u^*, v^*) is perturbed by $(\tilde{u}, \tilde{v})$ as expressed in the following

$$u(x,y,t) = u^* + \epsilon \tilde{u}(x,y,t) \tag{17}$$

$$v(x,y,t) = v^* + \epsilon \tilde{v}(x,y,t). \tag{18}$$

These perturbations are inserted in the fitness taxis model, Equation (16a)-(16b), and this results in the system below [2].

$$\frac{\mathrm{d}\tilde{\mathbf{w}}}{\mathrm{d}t} = \mathbf{A}\tilde{\mathbf{w}} - \mathbf{T}\nabla^2\tilde{\mathbf{w}} + \mathbf{D}\nabla^2\tilde{\mathbf{w}} \tag{19}$$

where the solution vector is

$$\tilde{\mathbf{w}}(x,y,t) = \begin{pmatrix} \tilde{u}(x,y,t) \\ \tilde{v}(x,y,t) \end{pmatrix}$$

and the matrices are given by

$$\mathrm{A} = \begin{bmatrix} f_u^* u^* & f_v^* u^* \\ g_u^* v^* & g_v^* v^* \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} \gamma_u f_u^* u^* & \gamma_u f_v^* u^* \\ \gamma_v g_u^* v^* & \gamma_v g_v^* v^* \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} D_u & 0 \\ 0 & D_v \end{bmatrix}.$$

A solution ansatz is employed, which represents a wavelike solution, in order to find a solution to the linear system (19)

$$\tilde{\mathbf{w}}(x,y,t) = \mathbf{w}_0 \mathrm{e}^{\lambda t + i(k_x x + k_y y)} \tag{20}$$

Here λ is the growth rate of the pertubations and k_x and k_y are the wavenumbers for the spatial coordinates and . The growth or decay of the solution will depend on $\mathrm{e}^{\lambda t}$. When solutions of the above type exist they represent traveling waves. A traveling wave is an example of a spatiotemporal pattern, i.e. existing in both time and space. The shape and the amplitude of the wave, i.e. the spatial part, together with its time-varying position and possible shape in space, will be an essential part of the pattern [4].

Substituting the solution ansatz into Equation (19) and canceling out the term $\mathrm{e}^{\lambda t + i(k_x x + k_y y)}$ yields the eigenvalue problem for the full system

$$\lambda \mathbf{w}_0 = (\mathbf{A} + k^2(\mathbf{T} - \mathbf{D}))\mathbf{w}_0 \tag{21}$$

where $k := |\mathbf{k}|$ and $\mathbf{k}$ is the vector of wavenumbers for the spatial coordiantes [2]. With λ as a function of the wavenumber, meaning $\lambda = \lambda(k^2)$, the solution from Equation (20) shows the time behavior of each wave, namely for each k. This is called the *dispersion relation* for λ, i.e. the growth rate of the perturbations, in terms of the wavenumbers k_x and k_y, for which the solution will depend on.

The eigenvalues for the full system are determined by the roots of the characteristic polynomial,

$$\det\left(\lambda \mathbf{I} - (\mathbf{A} + k^2(\mathbf{T} - \mathbf{D}))\right) = 0 \tag{22}$$

In the above expression the terms can be collected w.r.t λ and reformulated into the following

$$\lambda^2 - p(k^2)\lambda + q(k^2) = 0, \tag{23}$$

where

$$p(k^2) = \mathrm{Tr}[\mathbf{A} + k^2(\mathbf{T} - \mathbf{D})] \tag{24}$$

$$q(k^2) = \det(\mathbf{A} + k^2(\mathbf{T} - \mathbf{D})) \tag{25}$$

which are the trace and determinant conditions for the full system [2]. Both are functions of the wavenumber squared, k^2. Solving Equation (23) w.r.t. the eigenvalues, λ, yields the following

$$\lambda = \frac{1}{2}p(k^2) \pm 1/2\sqrt{p(k^2)^2 - 4q(k^2)}. \tag{26}$$

It is already imposed from the previous section, Section 2.1.1, that the steady state is *stable* in the absence of any spatial effects, namely that $Re(\lambda(k^2 = 0)) < 0$. This means, that for the steady state for the full fitness taxis system to be *unstable* to spatial disturbances, the real part of at least one of the eigenvalues must be positive for some $k \neq 0$, i.e. $\mathrm{Re}(\lambda(k^2))>0$. [4].

To summarize; for the equilibrium solution to the full fitness taxis model to be *unstable* it is thus required that one of the two conditions,

$$p(k^2) > 0, \quad q(k^2) < 0 \tag{27}$$

for the trace and determinant must hold, respectively.

Looking into these required conditions from Equation (27), *three* different cases occur for which the two conditions are met [2]. These three cases are summarized in the following section, and whenever the full fitness taxis system falls within one or more of these cases, the stable equilibrium point (u^*, v^*) for the population dynamics (1a)-(1b) will be *unstable* for the full system, and so patterns can occur.

2.3.2 Conditions for Pattern Formations

The first case, `CASE 1`, for which the full system is unstable, arises from the condition of the trace $p(k^2)$, see Equation (24), being positive. Equation (24) can be rewritten to the following.

$$p(k^2) = \mathrm{Tr}[\mathbf{A} + k^2(\mathbf{T} - \mathbf{D})] = \mathrm{Tr}[\mathbf{A}] + k^2\mathrm{Tr}[\mathbf{T} - \mathbf{D}] \tag{28}$$

Since $\mathrm{Tr}[\mathbf{A}] < 0$ from Section 2.1.1, $\mathrm{Tr}[\mathbf{T} - \mathbf{D}]$ must be positive for the trace $p(k^2)$ of the full system to be positive.

This implies that Equation (28) is a linear function with positive slope. For $k^2 = 0$, the trace is just equal to the trace of the linearized system for the population dynamics seen in Equation (6), which, as stated, is negative, hence stable. The graph for the trace condition as a function of the wavenumber squared looks as follows.

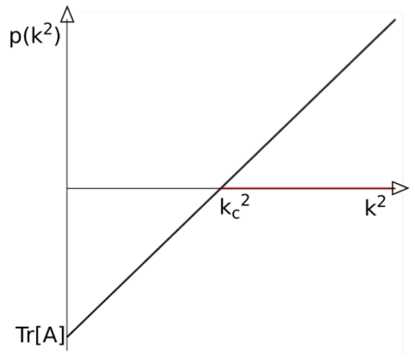


Figure 3: The trace condition as a function of the wavenumber squared

At some point, the graph for $p(k^2)$ will intersect the k^2-axis, and thereby change sign. This intersection will be at the point called the *critical wavenumber* and is found where $p(k_c^2) = 0$. This is expressed by the following

$$k_c^2 = -\frac{\mathrm{Tr}[\mathbf{A}]}{\mathrm{Tr}[\mathbf{T}-\mathbf{D}]}. \tag{29}$$

For wavenumbers $k^2 < k_c^2$ the trace of the full system will be negative, and so the system will be stable. For larger wavenumbers, i.e. $k_c^2 < k^2 \to \infty$, which is indicated by a red line in Figure 3, the trace of the full system will be positive and the real part of the largest eigenvalue will grow unboundedly, hence the full system will be unstable. The above discoveries are summed in the following.

`CASE 1: Taxis driven`

For wavenumbers $k^2 > k_c^2$ the full system is unstable if the condition $\mathrm{Tr}[\mathbf{T}-\mathbf{D}] > 0$ is met, corresponding to

$$\gamma_u u^* f_u^* + \gamma_v v^* g_v^* > D_u + D_v \tag{30}$$

This condition requires large prey taxis, γ_u, and so the predator taxis, γ_v and the diffusion parameters, D_u, D_v, must all be small. This condition can in fact be met without any diffusion at all, and is therefore driven by the taxis of the prey [2]. When the inequality in Equation (30) is fulfilled, this also ensures that $\mathrm{Tr}[\mathbf{T}-\mathbf{D}] \neq 0$, which is required from

Equation (29).

The determinant condition, see Equation (27), gives rise to the remaining two cases, `CASE 2` and `CASE 3`, respectively. The determinant is expressed in Equation (25) and can be rewritten in the following way

$$q(k^2) = \det(\mathbf{A} + k^2(\mathbf{T} - \mathbf{D})) = k^4 \det(\mathbf{T} - \mathbf{D}) + k^2 S + \det(\mathbf{A}) \tag{31}$$

where $S = \gamma_u \det(\mathbf{A}) + \gamma_v \det(\mathbf{A}) - u^* f_u^* D_v - v^* g_v^* D_u$ [2]. Note that $q(0)$ equals the positive determinant of the linear operator $\mathbf{A}$ from Equation (19).

Equation (31) expresses a parabola, with the roots given by

$$k_{1,2}^2 = -\frac{S \pm \sqrt{R}}{2\det(\mathbf{T} - \mathbf{D})}, \tag{32}$$

$$R = S^2 - 4\det(\mathbf{A})\det(\mathbf{T} - \mathbf{D}). \tag{33}$$

There are two different scenarios for this parabola, i.e. $q(k^2)$, based on the expression $\det(\mathbf{T} - \mathbf{D})$ being either negative, referring to `CASE 2`, or positive, referring to `CASE 3`. The first scenario is visualized in the following figure.

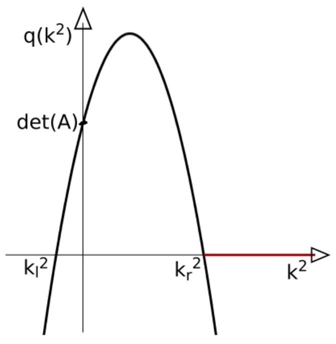


Figure 4: The determinant condition as a function of the wavenumber squared for $\det(\mathbf{T} - \mathbf{D}) < 0$

Conditions for this situation is summarized in the following.

`CASE 2:  Taxis and diffusion driven`

For wavenumbers $k^2 > k_l^2$ the full system is unstable if the condition $\det(\mathbf{T} - \mathbf{D}) < 0$ is met, corresponding to

$$D_v \gamma_u u^* f_u^* + D_u \gamma_v v^* g_v^* > \gamma_u \gamma_v \det(\mathbf{A}) + D_u D_v \tag{34}$$

This condition cannot be met, if neither the prey taxis, γ_u, nor the predator diffusion, D_v, vanishes. Due to this, the conditions for `CASE 2` is a combination of large predator

diffusion, D_v, and large prey taxis γ_u. For the opposite situation, large prey diffusion D_u and large predator taxis γ_v, this condition will be very hard to meet [2].
Finally the last situation, referring the U-shaped solution of the parabola $q(k^2)$ expressed in Equation (31) with the term $\det(\mathbf{T}-\mathbf{D})$ being positive is visualized in the following.

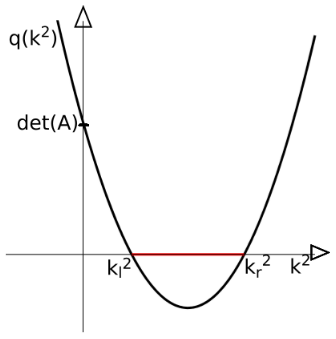


Figure 5: The determinant condition as a function of the wavenumber squared for $\det(\mathbf{T}-\mathbf{D}) > 0$

Conditions for the full system to be unstable for this situation are summarized in the following. `CASE 3: Diffusion driven`
For wavenumbers $k_l^2 < k^2 < k_r^2$ the full system is unstable if the following *three* conditions are all met, corresponding to

$$\gamma_u u^* f_u^* + \gamma_v v^* g_v^* > (\gamma_u + \gamma_v)\det(\mathbf{A}) \tag{35}$$

$$\gamma_v - \gamma_u < \frac{\gamma_u u^* f_u^* - \gamma_v v^* g_v^* - 2\sqrt{-u^* f_v^* v^* g_u^* D_u D_v}}{\det(\mathbf{A})} \tag{36}$$

$$D_v \gamma_u u^* f_u^* + D_u \gamma_v v^* g_v^* < \gamma_u \gamma_v \det(\mathbf{A}) + D_u D_v \tag{37}$$

These conditions corresponds to (35) $S < 0$, (36) $R > 0$ and (37) $\det(\mathbf{T}-\mathbf{D}) > 0$ [2]. This is the last case, and all three subconditions must be fulfilled for the full system to be unstable. These conditions can all be met, if there is no taxis present, i.e., if $\gamma_u = \gamma_v = 0$. In that case the system is back to the classical Turing model, and so classical Turing patterns will occur. Without prey diffusion, $D_u = 0$, the conditions for `CASE 3` can also be met, but not without predator diffusion, $D_v > 0$. This case is thereby diffusion driven. This means that only with small values of the fitness taxis parameters, the above conditions can be fulfilled. Due to this Touring patterns will still occur, even though the advection is added.

For both cases, `CASE 2` and `CASE 3`, the trace condition does not play a role since the slope of the graph is negative, i.e $p(k^2) < 0$, for all k^2 when one of the two scenarios in Figure 4 and Figure 5 are fulfilled. This means that the trace will be negative for all value of k^2 and thereby that instability in the full system is solely determined by the determinant condition in these two cases. The only exception to this regards an overlap between `CASE 1` and `CASE 2` where the system meets the requirements for both cases and so will be a combination of these, i.e. will be unstable for all $k^2 > 0$. Finally the above conditions that must be met for instability in the full system are summarized below.

As a recap of the above conditions, the three cases can all be expressed as functions of the two taxis parameters γ_u and γ_v [2]. When doing so, it is possible to draw a bifurcation diagram for the full system for specific values of the diffusion coefficients, D_u and D_v. This allows for the system to be investigated, in order to see for which parameter values of γ_u and γ_v, the full system will fall within one or several of the three cases and thereby be unstable to spatial perturbations and so able to form patterns.

The ratio between the diffusion coefficients, D_v/D_u, must be above 10.5 when the four parameters are specified as in Equation (7) in order for the Turing patterns to show [2]. The values for the diffusion coefficients will for this study be set to $D_u = 1$ and $D_v = 15$, corresponding to the predator diffusing much faster than the prey. For these specific values the following diagram can be made, see Figure 6.

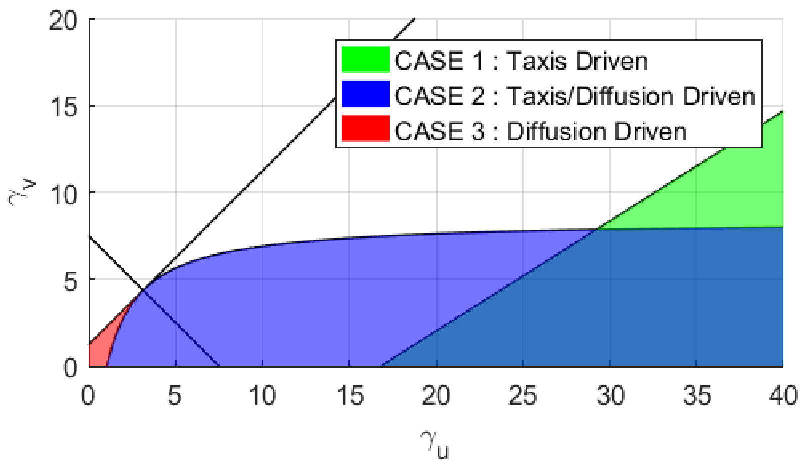


Figure 6: Sketch of the bifurcation diagram for the advection parameters γ_u, γ_v with $D_u = 1$ and $D_v = 15$.

For parameter values that lie outside of the three areas, green, blue and red, no stable patterns will form. If no diffusion is present, then the system is a reaction-advection model and the bifurcation diagram will look as follows.

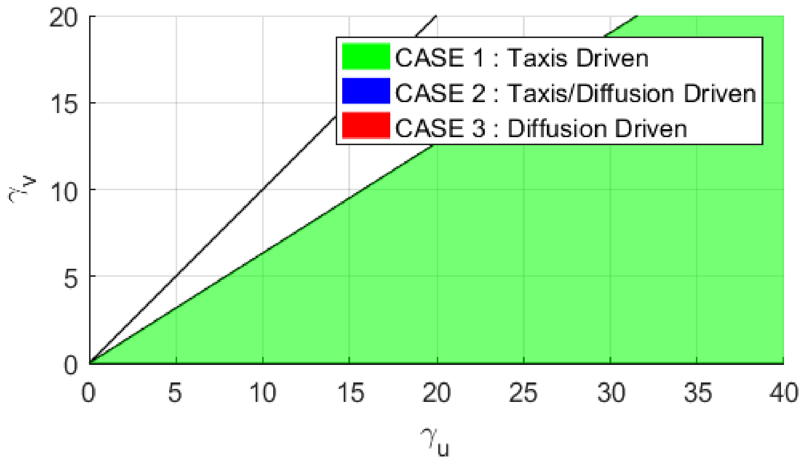


Figure 7: Sketch of the bifurcation diagram for the advection parameters γ_u, γ_v with $D_u = 0$ and $D_v = 0$.

This is in good accordance with the description of `CASE 1`, the taxis driven instabilities, since Figure 7 shows that the full system can only fall within the first case.

3 Algorithm

Now that conditions for patterns formation in the full system (16a)-(16b) has been investigated, the aim is to solve the model numerically in these regions. To do so, a good and efficient solver must be implemented in order to compute for long time periods.

For this study, the solver is written in Julia, a new homoiconic functional language focused on technical computing. The solver will take advantage of the different parts of the model equations, and is developed in particular to be effective for this model-type problem.

The fitness taxis model consists, as previously mentioned, of three different terms; a reaction, an advection and a diffusion part. These three terms have a different behavior when solved numerically, and so, optimally, must be treated differently.

The diffusion term is linear but very stiff. This means that diffusion takes place on very small time scale compared to the overall time scale. The stiffness implies that such a term has to be solved implicitly, which makes it difficult and very time consuming. The non-linear advection and reaction terms on the other hand are less stiff than the diffusion part, and so will be integrated explicitly in time.

If the entire equation was to be integrated implicitly it would be possible to take much larger time steps, but each step would involve the computationally expensive solution of a system of nonlinear equations, namely the reaction and advection terms.

Fortunately, a number of time-stepping techniques make use of exactly the fact that the stiff term is linear, and are thereby able to split the equation in order to integrate implicitly and explicitly for the different terms.

For this study such a time-stepping method, an implicit-explicit (IMEX) partitioned Runge-Kutta method, is implemented. With regard to time-dependency a semi-discretization is used, i.e., the method of lines. For this, each partial differential equation is transformed into an ordinary differential equation by finite difference spatial discretization.

This chapter is divided into four parts with a step by step description of the algorithm. Initially notations regarding the remainder of this study is presented. In the second section discritization in space is expressed. Section three goes trought the integration in time and how the system is solved in each time step. Finally a short introduction to Julia is given in the last part of this chapter.

3.1 Definitions and Notations

Before studying the fitness taxis problem both the domain and the notation that will be employed will be defined. The domain $\Omega = [0, L_x] \times [0, L_y]$, where L_x and L_y are the lengths of the computational domain in each of the two spatial directions, x and y respectively. The domain is divided into $N = N_x \cdot N_y$ equally distributed number of grid points, where N_x is the number of grid points in the x-direction and N_y for the y-direction. This results in a spacing in each spatial direction, $\Delta x = L_x/N_x$ and $\Delta y = L_y/N_y$. The boundary conditions will be periodic, i.e.

$$\begin{aligned}
u(0,y,t) &= u(L_x,y,t), \quad u(x,0,t) = u(x,L_y,t) \\
v(0,y,t) &= u(L_x,y,t), \quad v(x,0,t) = v(x,L_y,t).
\end{aligned}$$

3.2 Spatial Discretization

The first thing to do is to discretize the system (16a)-(16b) in space in order to transform the system of partial differential equations into a system of $n \cdot N$ ordinary differential equations. Here n is the number of species, for this study, $n = 2$. A finite difference scheme is used to approximate the spatial derivatives in both the diffusion and advection terms. To be specific this regards the discretization of the two operators ∇^2 and ∇. With this transformation the system now becomes

$$\frac{\partial u_h}{\partial t} = f(u_h, v_h)u_h - \gamma_u \mathrm{FD}_A \cdot u_h \mathrm{FD}_A \cdot f(u_h, v_h) + D_u \mathrm{FD}_D \cdot u_h \tag{38a}$$

$$\frac{\partial v_h}{\partial t} = g(u_h, v_h)v_h - \gamma_v \mathrm{FD}_A \cdot v_h \mathrm{FD}_A \cdot g(u_h, v_h) + D_v \mathrm{FD}_D \cdot v_h \tag{38b}$$

where $u_h = u_h(t) \in \mathrm{R}^N$ and $v_h = v_h(t) \in \mathrm{R}^N$ are the vectors of the two species at all N grid points, respectively. The finite difference matrix $\mathrm{FD}_D = \mathrm{FD}_{Dx} + \mathrm{FD}_{Dy}$ is the two-dimensional fourth-order centered discretization of the operator ∇^2. The finite difference matrix $\mathrm{FD}_A = \mathrm{FD}_{Ax} + \mathrm{FD}_{Ay}$ is the two-dimensional fourth-order centered discretization of the operator ∇. The matrices, FD_D and FD_A, can be seen in Appendix B.1 and Appendix B.2.

3.3 Integrating in Time

The approach to integrate in time is to use *splitting*. This means, that the problem is broken into smaller pieces, in order to apply the most efficient time integration method for each part. This adds both splitting error, but also integration error. The IMEX methods, are methods that consist of suitable mixtures of implicit and explicit methods. The IMEX methods exist either as linear multistep type or Runge-Kutta type. Partitioned Runge-Kutta methods allow for partitioning the equations by terms and applying a suitable method to each term [9].

The exact solver for integration in time used for this study applies a 2-additive Runge-Kutta (ARK2) method that combines an explicit Runge-Kutta (ERK) scheme with an explicit, singly diagonal implicit Runge Kutta (ESDIRK) scheme [16].

When the scheme is applied to the system (38a)-(38b), which, to recap, is discretized in space with a fourth-order centered finite difference discretization of both the advection and the diffusion term, it results in the following scheme.

$$\xi_i = \mathbf{X}_h^n + \Delta t \left(\sum_{j=1}^{i} a_{ij}^I D\mathrm{FD}_D \xi_j - \sum_{j=1}^{i-1} a_{ij}^E \gamma \mathrm{FD}_A(\xi_j \mathrm{FD}_A \mathbf{f}_R(\xi_j)) + \sum_{j=1}^{i-1} a_{ij}^E \mathbf{f}_R(\xi_j)\xi_j \right) \tag{39}$$

$$\mathbf{X}_h^{n+1} = \mathbf{X}_h^n + \Delta t \sum_{i=1}^{s} b_i (\mathrm{FD}_D \xi_i - \gamma \mathrm{FD}_A(\xi_i \mathrm{FD}_A \mathbf{f}_R(\xi_i) + \mathbf{f}_R(\xi_i)\xi_i)) \tag{40}$$

where $\mathbf{X}_h = [u_h, v_h]$, D is the diffusion matrix, with D_u and D_v in the diagonal, γ is the advection matrix, with again γ_u and γ_v in the two diagonal elements, respectively. $\mathbf{f}_R$ represents the reactive terms, $f(u, v)$ and $g(u, v)$.

The term a_{ij}^E represents the coefficients of the explicit Runge-Kutta (ERK) scheme and the term a_{ij}^I represents the coefficients of the explicit first stage, singly diagonally implicit Runge-Kutta (ESDIRK) scheme. The scheme is derived in [17]. It is third-order accurate and the coefficients are listed in the Butcher-array in Appendix C.2.

To calculate the stages of the method, Equation (39) must be solved for ξ_i. Since the first stage of the ESDIRK scheme is explicit and all other stages $a_{ii} = \sigma$ has the same value, Equation (39) can be rewritten into in the following way for $1 < i \leq s$. Here, s is the number of stages and for this study $s = 4$.

$$\begin{aligned}&(I - \Delta t \cdot \sigma \cdot D\mathrm{FD}_D)\xi_i = \\ &\mathbf{X}_h^n + \Delta t \sum_{j=1}^{i-1} \left(a_{ij}^I D\mathrm{FD}_D \xi_j - a_{ij}^E \gamma \mathrm{FD}_A(\xi_j \mathrm{FD}_A \mathbf{f}_R(\xi_j)) + a_{ij}^E \mathbf{f}_R(\xi_j)\xi_j \right)\end{aligned} \tag{41}$$

Written like this Equation (41) has the form $\mathbf{Ax} = \mathbf{b}$, where all terms on the right hand side are known. From this, it is possible to solve the system with a direct solver. Unfortunately, the size of $\mathbf{A}$ is proportional to $(N_x \cdot N_y)^2$, so as the grid becomes finer, the system grows fast. It is thus possible to take advantage of the fact that the matrix $\mathbf{A}$ is sparse and on as results an iterative solver is implemented [16]. The iterative solver will be the weighted Jacobi method that decomposes the sparse matrix $\mathbf{A}$ and so the solution is obtained iteratively.

$$\mathbf{x}^{k+1} = (\mathbf{I} - \omega \mathbf{D}^{-1}\mathbf{A})\mathbf{x} + \omega \mathbf{D}^{-1}\mathbf{b} \tag{42}$$

A method like the weighted Jacobi method acts as a smoother for the system, i.e. fast frequencies of the error will disappear quickly and the slower frequencies will stay longer.

In order to deal with the slower frequencies and thereby get a faster convergence, it is possible to project the error on to a coarser grid, where the slow frequencies become faster. This means that applying the smoother on the coarse grid, the slower frequencies of the error can be reduced. The system can thereafter be projected back onto the finer grid. This procedure is done by a multigrid solver. The solver can be seen in Appendix D, where h refers to the finer grid and H to the coarser grid [16].

As the spatial approximation with finite difference has a simple, regular form, the finite difference matrices for the ∇- and ∇^2-operator can easily be created for other, coarser grids as well. These matrices will have the exact same form on a finer grid as on a coarser grid, except that they on a coarser grid have fewer entries. As a consequence, it is imposed in the code that the number of intervals for the grid is a power of two to ensure that the matrices can easily be mapped to a coarser grid whitout building complete new ones [16].

In order for the multigrid solver to work, it needs an initial guess for x from Equation (42). This can be done by remembering the k'th last time steps and calculating a solution as a linear combination of those k-vectors first [16]. Doing this, the problem is projected into the k-dimensional subspace which are spanned by these vectors. Finally QR-decomposition is used in order to avoid numerical problems [16].

To control the step size the algorithm uses an adaptive step size controller that takes advantage of the local error estimate. Since some of the matrices in the problem can be reused, an interval $0.5 \leq q \leq 2$ is introduced, where

$$q = \left(\frac{tol}{\epsilon} \right)^{1/3}.$$

For this, the step size is not nescessarily changed in each iteration, but only if q is outside of the interval [16].

For values $\gamma_u = \gamma_v = 0$ the system reduces to a reaction-diffusion system, and in conjunction with the algorithm the solver is well tested [16]. When the advection term is added, no manufactured solution exists, and the validation of the program must be tested differently.

3.4 Julia

The simulations will be implemented and executed in the dynamic language, *Julia*, which was released in February 2012 as an open source project [18]. Julia is designed for performance with the purpose of being both easy and fast, still using a high-level dynamic language. Scientific computing programs such as `MATLAB`, Octave, R, SciPy and SciLab are examples of programs that uses a high-level code language. These fall within the category named *dynamic languages* or *dynamically typed languages* and are today often preferred, despite the fact that they often lack sufficient performances [19].

In these languages the programmers write a simple high-level code omitting any specification of types like `inf`, `float` or `double` [18]. In contrast *statistically typed languages* as C and Fortran are dominated by the need for specification of types and both programs perform much more efficient for computationally intensive problems [18].

This issue between using a favorable high-level language that performs badly for intense problems or using a better performing but less favorable language is today often handled by parallel programming, and this is where Julia enters. Julia is developed in order to fill this exact gap, so that one does not have to program in two different languages. It is designed from the ground to take advantages of modern techniques for executing dynamic languages efficiently resulting in a combination of productivity and performance.

The browser-based version of Julia, named JuliaBox, is available for free and unlimited usage. This version requires no set-up and is the mainly used version for students worldwide. All code for this study are written in the Julia version 0.5.0, since this is one of the newest versions and installed packages turned out to have the best performance.

4 Simulations

This chapter will concern simulations of the fitness taxis model mainly performed in Julia. The chapter is divided in to six main parts where the first part give information on the initial conditions for all simulations. Next the standard reaction-diffusion system (13a)-(13b) will be solved, in order to see, if the system gives rise to the well known Turing patterns.

From there on advection will be added in three different scenarios in order to compute solutions and to see the spatio-temporal pattern formation of the full system, the fitness taxis model (16a)-(16b). The results computed in Julia will along the way be compared with results from simulations from `COMSOL` Multiphysics, a modeling program based on finite element computations. In the fourth section the effect of the initial conditions will be investigated.

Finally in the fifth part, the results from three different scenarios with varying amount of taxis for each species will be compared to one another and to the result from the taxis free system, i.e. the reaction-diffusion system (13a)-(13b). The results from this chapter will be summed up in the very last part.

4.1 Initial Conditions

As a very first step, values from the experimental setup must be specified. These regards in particular the size of the domain and the grid spacing. All values from the experimental setup are thereby listed below in Table 1. This table also summarizes some of the already specified parameter values, i.e. the four parameters from the model for the population dynamics (5a)-(5b) and the values for the two diffusion coefficients, D_u and D_v respectively.

Table 1: Specified parameters for simulations in Julia

L_x	L_y	N_x	N_y	Δx	Δy	k_x	k_y	D_u	D_v	a	c	r	K
20π	20π	64	64	0.98	0.98	0.1	0.1	1	15	5	1.5	2.8	28/3

These are all variables that will be kept unchanged throughout all the simulations, which is done in order to compare results regarding, in particular, the influence on the advective term by varying the two advection parameters γ_u and γ_v. The number of time steps, N_t, and the time step, Δt, will thus change as a result of the error control that leads to an adaptive step size, see Section 3.3.

For computations in COMSOL the domain will be of the same size, but the with a finer grid. This gives rise to smoother results than seen for those computed in Julia. Decreasing the grid spacing in Julia increases the computational time dramatically which is not the case for computations in COMSOL. For that reason the values listed in Table 1 will be kept unchanged for computations in Julia unless otherwise stated.

The initial condition can also be varied in numerous ways, but will for all following simulations, unless otherwise stated, be specified as in the following.

$$u_0(x, y, t = 0) = \frac{1 + tanh(\frac{(x-r_1)-(y-r_2)}{4})}{2} \tag{43}$$

$$v_0(x, y, t = 0) = \frac{1 + tanh(\frac{-(x-r_2)+(y-r_1)}{4})}{2} \tag{44}$$

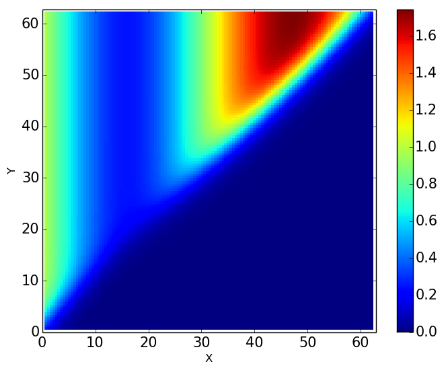
where $r1 = 26.0$ and $r2 = 24.0$. On this form each species will be distributed in only one half of the domain, the prey in the upper left half of the domain, and the predator in the lower right part.

The initial conditions is the perturbed in the following way

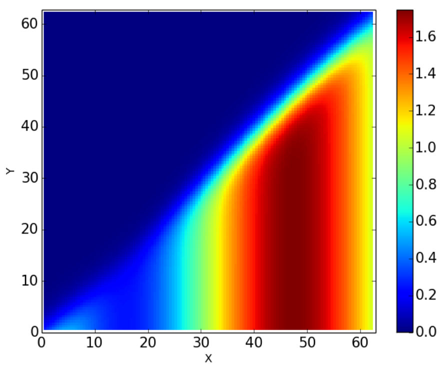
$$\tilde{u} = u_0(x, y, 0)(1 + Bsin(k_x)) \tag{45}$$

$$\tilde{v} = v_0(x, y, 0)(1 + Bsin(k_x)) \tag{46}$$

where $B = -0.75$.
The distribution of the initial conditions for the predator and prey, i.e. the perturbed initial conditions, are illustrated below.



(a) Initial distribution of prey



(b) Initial distribution of predator

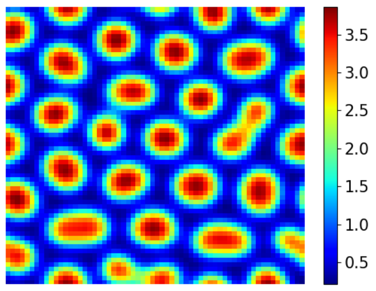
Figure 8: Initial distribution of prey and predator respectively

As seen, the prey and predator are thereby both distributed around the equilibrium point $(u^*, v^*) = (1, 1)$. When distributed as above with no prey in the lower right triangular part and no predator present in the upper left triangular part, the movement of each species can closely be followed. This is an advantage in the investigation of whether the system is most taxis or diffusion driven since different mechanism drive these two motions.

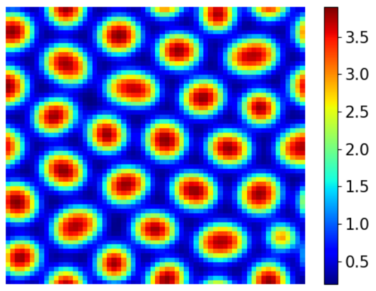
As the the domain size for all simulations is left unchanged, with the specified values listed in Table 1, the notation will be omitted in all following figures.

4.2 Reaction-Diffusion Model

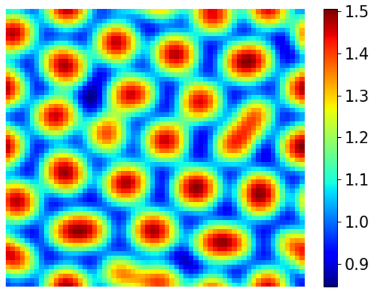
The fitness taxis model (16a)-(16b) reduces to the well known reaction-diffusion model (13a)-(13b) when $\gamma_u = \gamma_v = 0$. This system can be solved with the well-tested unmodified Julia code [16], in order to investigate the emergence of the predicted Turing patterns. The following four figures are the results simulated for a long time period, here at time $t = 100$ in Figure 9a and Figure 9c and at time $t = 200$ in Figure 9b and Figure 9d at both time intervals for the prey and predator respectively.



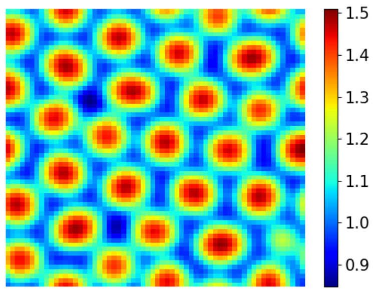
(a) Distribution of prey at time $t = 100$ for the model (13a)-(13b).



(b) Distribution of prey at time $t = 200$ for the model (13a)-(13b).



(c) Distribution of predator at time $t = 100$ for the model (13a)-(13b).



(d) Distribution of predator at time $t = 200$ for the model (13a)-(13b).

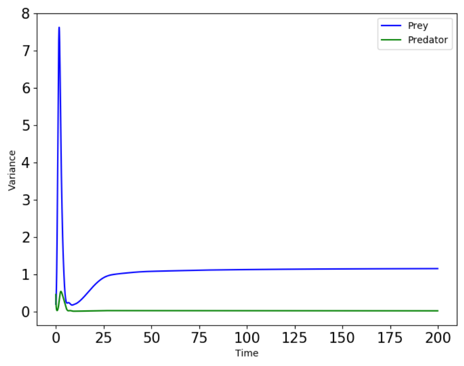
Figure 9: Pattern formation for the reaction-diffusion system (13a)-(13b) after time $t = 100$ (left) and $t = 200$ (right) for both species respectively

The last two figures shows, in particular, the stable pattern formation for the reaction-diffusion system. From time $t = 100$ and until time $t = 200$ the patterns do not change much, and can therefore after $t = 200$ be said to be stable.

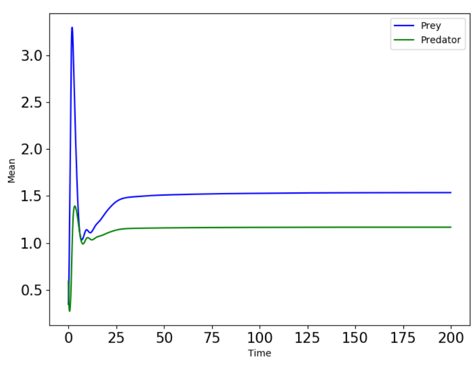
The stable pattern for both species is shaped as small separated round clusters consisting of the prey and predator respectively. These clusters, or spots, are distributed in hexagonal arrangements over the entire domain. This pattern structure is one of the best known Turing patterns [5]. Each cluster peaks in the middle with the biggest amount of prey and predator respectively. From Figure 9 it is furthermore seen, that the clusters for both prey and predator are located at the same spatial locations in the entire domain.

The clusters for the prey density varies more compared to the cluster consisting of the predator density. Consequently the prey density outside of the clusters, seen for instance in Figure 9b, are less than the predator density outside of the clusters correspondingly seen in Figure 9d.

Clarifying that the patterns above are stable can be seen when looking at the variance and the mean of each the two populations as a function of time. This is shown below.



(a) The variance of the predator poulation (green) and prey population (blue) as a function of time



(b) The mean of the predator poulation (green) and prey population (blue) as a function of time

Figure 10: The variance (left) and the mean (right) as a function of time of the two populations in the reaction-diffusion system (13a)-(13b)

Already around time $t = 25$ both figures show that the variance and the mean of each population density are almost stable, and this indicates stationarity of the patterns.

On behalf of the above experiences, all following simulations will only be computed until time $t = 100$. This in in order to see how the pattern is changed for both species when advection is added after an equal amount of time. Simulating for a longer amount of time is very time consuming computationally wise, and therefore in order to compare only until time $t = 100$ is shown.

4.3 Fitness Taxis in `CASE 3`

Now it is time time to investigate the full fitness taxis model (16a)-(16b). The two parameters γ_u and γ_v can be varied as desired to see the dependence on the taxis term, i.e. how the species move around due to the taxis term and how this inflects pattern formation. Either taxis can be added equally for the two species, i.e. $\gamma_u = \gamma_v$, or more can be added for one species compared to the other. Finally, since the predator diffuses much faster than the prey, namely $D_v/D_u = 15$, it is notable that the addition of prey taxis in much greater amount than the predator taxis, can change the behavior of which species that moves around the fastest.

The following Figure 11, is zoomed in on the area regarding the conditions for `CASE 3`, where the model is diffusion driven, and thereby behaves a lot like a standard reaction-diffusion system. From this it is easily seen which exact values the taxis parameters regarding `CASE 3` concerns, by reading them of the red area in the graph.

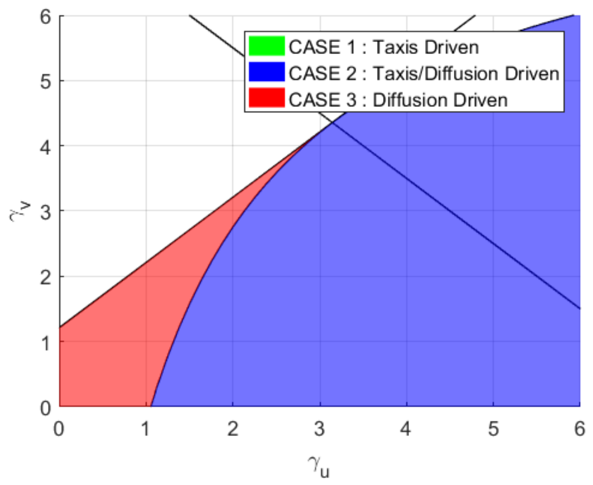


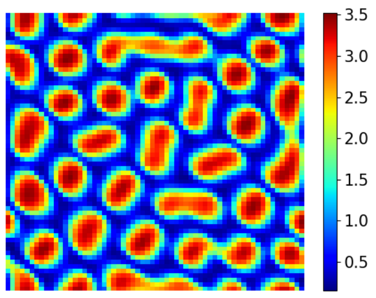
Figure 11: Zoom of the bifurcation diagram from Figure 6 showing only `CASE 3`

In the following three scenarios with a different amount of prey and predator taxis computed in both Julia and `COMSOL`, will be shown. As mentioned in Section 3 the full algorithm for the reaction-diffusion system is well tested, i.e. for $\gamma_u = \gamma_v = 0$ [16]. Due to the modification in the implementation of the advection term, `COMSOL` is used to compare the results in order to see if the two programs give rise to similar results. Since a model problem with known solution with the form of the fitness taxis model (16a)-(16b) is difficult to find, this comparison can be used verify that the program implemented in Julia give rise to usable results.

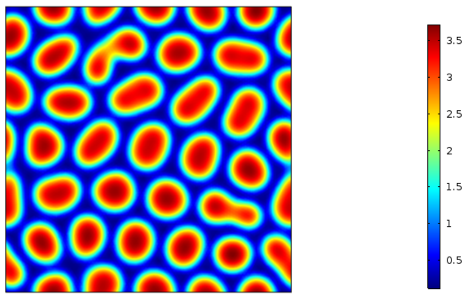
4.3.1 Scenario $\gamma_u = 0.5, \gamma_v = 0.5$

As a first simulation of the full fitness taxis model (16a)-(16b) a small amount of taxis is added equally for both species, i.e. $\gamma_u = \gamma_v = 0.5$. For these values of the advection parameters, the model is also simulated in `COMSOL`.

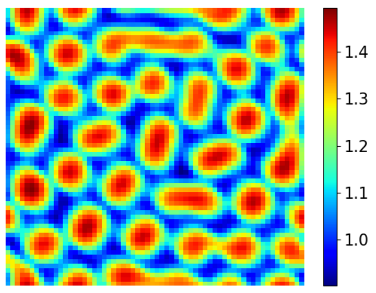
The simulations from the two programs results in the following patterns seen below. Figure 12a and Figure 12c are the results computed in Julia for the prey and predator respectively and Figure 12b and Figure 12d are the results computed in `COMSOL` for the prey and predator respectively. All four pictures are the result after the time $t = 100$.



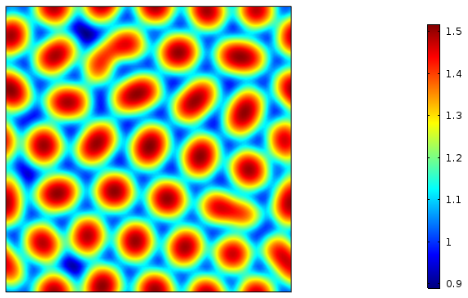
(a) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.5$ simulated in Julia.



(b) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.5$ simulated in `COMSOL`.



(c) Distribution of predator at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.5$ simulated in Julia.



(d) Distribution of predator at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.5$ simulated in `COMSOL`.

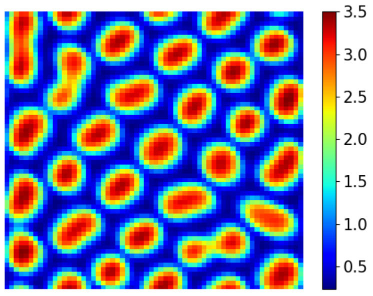
Figure 12: Distribution of prey (top) and predator (bottom) at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.5$ simulated in Julia and `COMSOL` repsectively.

First of all it can be seen that the results from the two different programs look a lot alike, only slightly deviating from one another. It is easily seen, than the results from both programs shows a pattern formation. The patterns all show hexagonal arrangement of spots similarly to those seen in Section 4.2. The clusters in all four pictures are distributed in the direction of the anti-diagonal and the results from both program show that not all clusters are completely separated round peaks, as described for the reaction-diffusion case in Section 4.2.

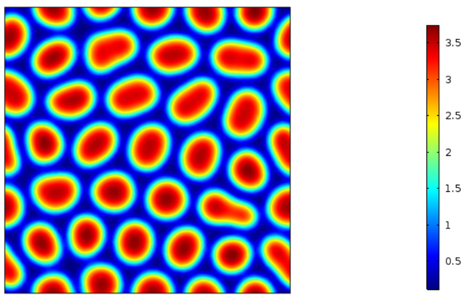
Compared to earlier results some cluster are now stretched, as if consisting of two clusters not yet separated. This indicates, that the species are less likely to spread out and separate into round clusters, when taxis is added. The results from the two programs are in good accordance with each other, since the same trend is evident in the results computed from the two different programs.

4.3.2 Scenario with $\gamma_u = 0.5, \gamma_v = 0.8$

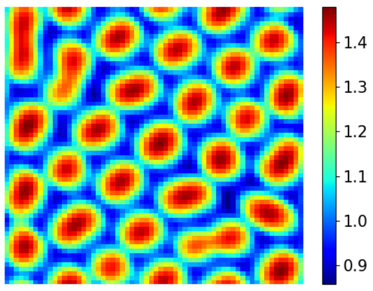
Adding a bit more predator taxis, the fitness taxis model (16a)-(16b) with the parameter values $\gamma_u = 0.5$ and $\gamma_v = 0.8$ are computed in Julia and in `COMSOL` as well. Figure 13a and Figure 13c are the results at time $t = 100$ computed in Julia and Figure 13b and Figure 13d computed in `COMSOL` for the prey and predator respectively.



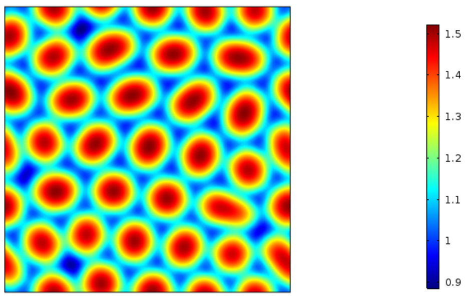
(a) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.8$ simulated in Julia.



(b) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.8$ simulated in `COMSOL`.



(c) Distribution of predator at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.8$ simulated in Julia.



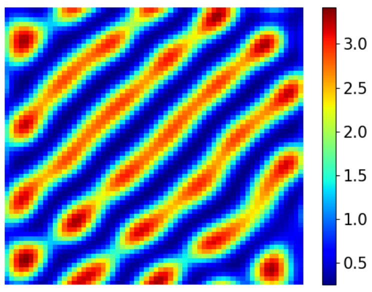
(d) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.8$ simulated in `COMSOL`.

Figure 13: Distribution of prey (top) and predator (bottom) at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.8$ simulated in Julia and `COMSOL` repsectively.

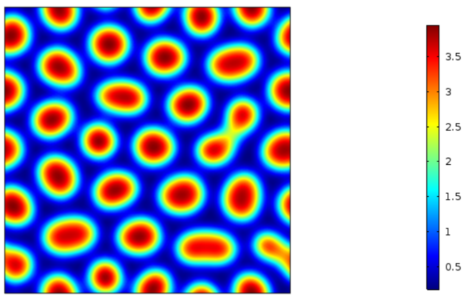
The previous trend is also seen from these results, where some clusters are not yet separated at time $t = 100$. In the results from Julia this is a bit more evident than in the results from `COMSOL`. The formed patterns do not vary much from what is previous seen in Section 4.2 and Section . They deviated most in the obsavation, that stationarity of the system is reached a bit later, since all patterns will separate completely for a longer amount of time.

4.3.3 Scenario with $\gamma_u = 0.1, \gamma_v = 0.8$

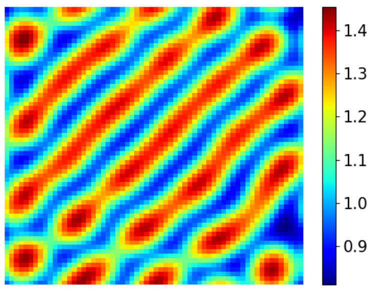
As a third example simulations with the parameter values $\gamma_u = 0.1$ and $\gamma_v = 0.8$ are computed. Results are again computed in both Julia and COMSOL. These are seen below, where Figure 14a and Figure 14c are the results computed in Julia for the prey and predator respectively and Figure 14b and Figure 14d are the results computed in COMSOL for the prey and predator respectively. All four pictures are the result after the time $t = 100$.



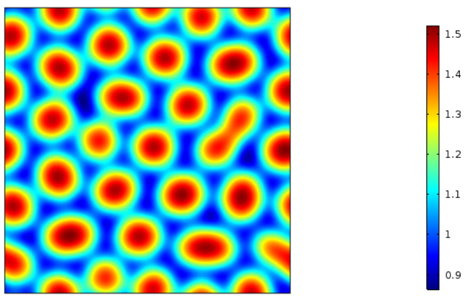
(a) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$,$\gamma_v = 0.8$ simulated in Julia.



(b) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$,$\gamma_v = 0.8$ simulated in COMSOL.



(c) Distribution of predator at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$,$\gamma_v = 0.8$ simulated in Julia.



(d) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$,$\gamma_v = 0.8$ simulated in COMSOL.

Figure 14: Distribution of prey (top) and predator (bottom) at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$,$\gamma_v = 0.8$ simulated in Julia and COMSOL repsectively.

For these simulations, the results between COMSOL and Julia deviate the most from one another compared to the previous sets of simulations. The former trend, seen especially from the results computed in Julia, with the clusters more likely to be linked together, is also seen for these simulations. It is particularly visible in Figure 14a and Figure 14c which are the results from Julia. Here the formed pattern consists of one long stripe in the direction of the anti-diagonal. In contrast to the former results, all clusters in the anti-diagonal are now linked together. This behavior is only slightly seen in the results from `COMSOL`. Here the patterns are still mainly separated clusters as seen in all previous simulations, but some clusters are thus linked together in the same direction as seen in the results from Julia.

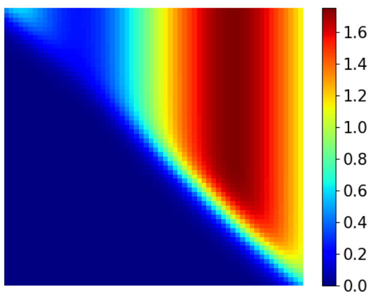
4.4 Effect of Initial Conditions

It is also of great interest to investigate the effect of the initial conditions. This is done in order to see if changing the initial state of the system, will change the generated patterns. The direction of the patterns that appeared in Section 4.3.3 from computation in Julia, i.e. stripes, located in the anti-diagonal is tested. This is done by changing the initial conditions so that the prey is now distributed in the upper right corner and the predator in the lower left corner. For the parameter values $\gamma_u = 0.1$ and $\gamma_v = 0.8$ the new initial conditions are used in order to compare the result with the stripes, and to see if the stripes change their direction or disappear. The equations for the new initial condition are now

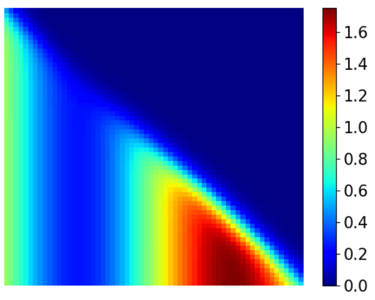
$$u_0(x, y, t = 0) = \frac{1 + tanh(\frac{(x-r_1)+(y-r_2)}{4})}{2} \tag{47}$$

$$v_0(x, y, t = 0) = \frac{1 + tanh(\frac{-(x-r_2)-(y-r_1)}{4})}{2} \tag{48}$$

where $r1 = 30.0$ and $r2 = 32.0$. These are perturbed as done previously in Equation (46). The distribution of the new initial conditions for the predator and prey, i.e. the perturbed initial conditions looks as the following.



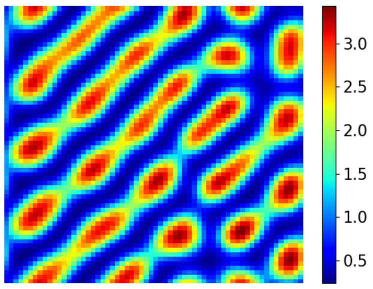
(a) New initial distribution of prey.



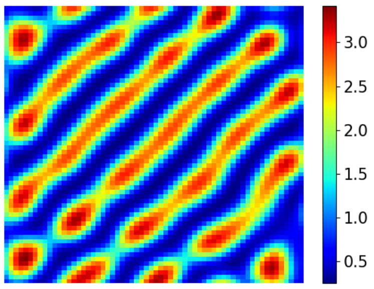
(b) New initial distribution of predator.

Figure 15: Initial distribution of prey and predator respectively

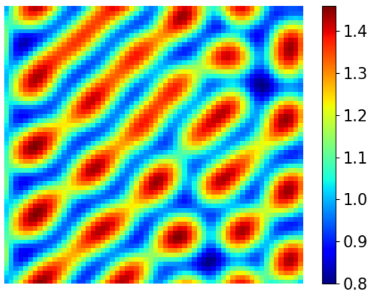
The results after the time $t = 100$ are seen below in Figure 16a and Figure 16c for the prey and predator respectively. The old results from Figure 14a and Figure 14c are show here again in order to clearly see the difference.



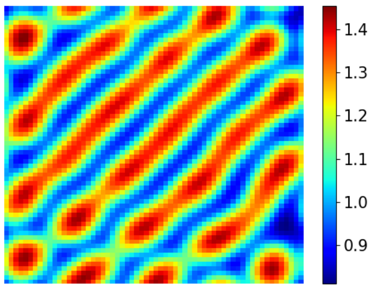
(a) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$ and $\gamma_v = 0.8$ and initial conditions from Equation (47).



(b) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$ and $\gamma_v = 0.8$ and initial conditions from Equation (48).



(c) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$ and $\gamma_v = 0.8$ and initial conditions from Equation (43).

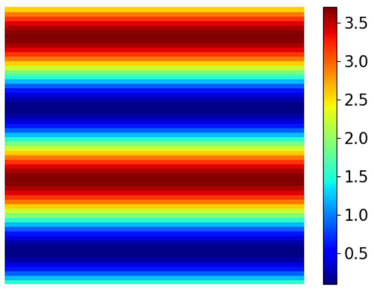


(d) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$ and $\gamma_v = 0.8$ and initial conditions from Equation (44).

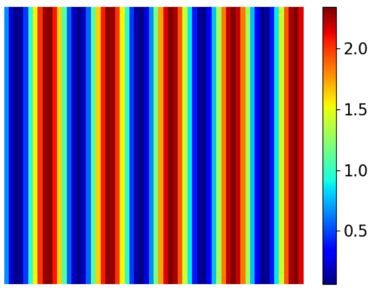
Figure 16: Distribution of prey (top) and predator (bottom) at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$ and $\gamma_v = 0.8$ and initial conditions from Equation (43)-(44) and (47)-(48) , respectively

What is seen is that the stripes still occur and that they are directed in the same way as before. The stripes from the new simulations are not as evolved as in the situation with the former initial conditions. Simulated for a longer period of time these will nevertheless end up in the same stationary pattern as for the simulations in Section 4.3.3.

As a last approach for this same situation regarding the parameter values $\gamma_u = 0.1$ and $\gamma_v = 0.8$, the prey and predator are both distributed over the entire domain as a third choice of initial conditions. The distributions of each population density is perturbed in the same way as earlier, i.e. by Equation (46). The distribution of these initial conditions for the predator and prey, i.e. the perturbed initial conditions looks as the following.



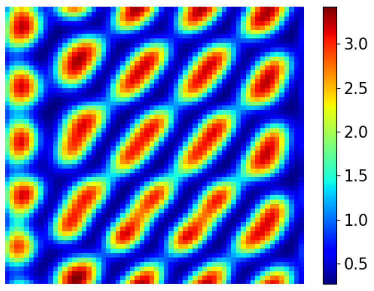
(a) New initial distribution of prey.



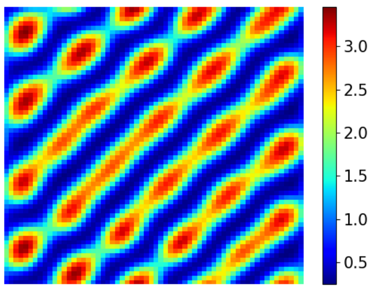
(b) New initial distribution of predator.

Figure 17: Initial distribution of prey and predator respectively

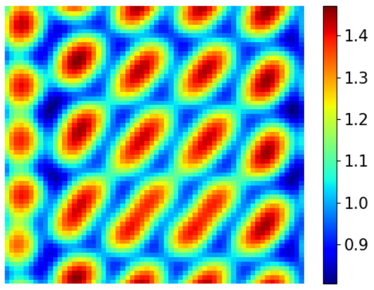
The wavenumbers for these initial conditions are $k_x = 0.5$ and $k_y = 0.2$. The results are shown in the following figure, where simulations at time $t = 100$ and $t = 200$ are both shown. This is due to the fact that with these initial conditions, the amount of time for the stationary pattern to form is longer than for the previous scenarios.



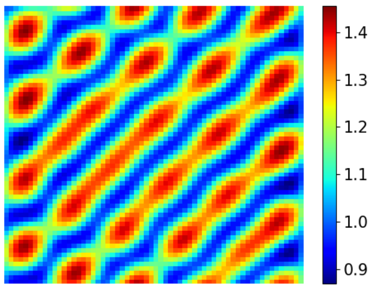
(a) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$, $\gamma_v = 0.8$ and initial conditions.



(b) Distribution of prey at time $t = 200$ for the model (16a)-(16b) with $\gamma_u = 0.1$, $\gamma_v = 0.8$ and .



(c) Distribution of prey at time $t = 100$ for the model (16a)-(16b) with $\gamma_u = 0.1$, $\gamma_v = 0.8$ and initial conditions from Equation (43).



(d) Distribution of prey at time $t = 200$ for the model (16a)-(16b) with $\gamma_u = 0.1$, $\gamma_v = 0.8$ and initial conditions from Equation (44).

Figure 18: Distribution of prey (top) and predator (bottom) at time $t = 100$ and time $t = 200$ for the model (16a)-(16b) with $\gamma_u = 0.1$, $\gamma_v = 0.8$ and initial conditions , respectively

Again the formed patterns are similar to the results from Figure 16. Even thought it takes a longer amount of time for the inhomogeneous patterns to stabilize, the final pictures are the same as previous seen.

Different scenarios have also been tested, one where the prey and predator do not reach each other at time $t = 0$. This causes the predator to die out quickly, even if it is capable of moving faster than the prey.Due to the continuous model, the predator will grow as soon as some amount of prey reaches the predator. This means that even though the amount of predator is very small, i.e. $v(x, y, t) << 1$, the population density will still be able to grow. As a consequence the predator will never go extinct due to numerical fluctuations.

4.5 Spatio-Temporal Development

A qualitative analysis of the spatio-temporal development of the three taxis scenarios from Section 4.3.2, 4.3.2 and 4.3.3, respectively, together with the taxis free scenario from Section 4.2 can be made. The developments of all four scenarios are shown at time the $t = [2, 5, 7, 10, 50]$ in order to show how the species distributes in space over time. This is first shown for the prey in Figure 19. The first column is the time development for the system with $\gamma_u = \gamma_v = 0.0$, the second column is for the parameter values $\gamma_u = \gamma_v = 0.5$, the third is for the values $\gamma_u = 0.5, \gamma_v = 0.8$ and the last column is the development for the system with parameter values $\gamma_u = 0.1, \gamma_v = 0.8$. In Figure 20 this development is shown for the predator as well. Full spatio-temporal development for both prey and predator from time $t = [1 : 10]$ for every $\Delta t = 1$ and time $t = [10 : 50]$ for $\Delta t = 10$ can be seen in Appendix E.1.

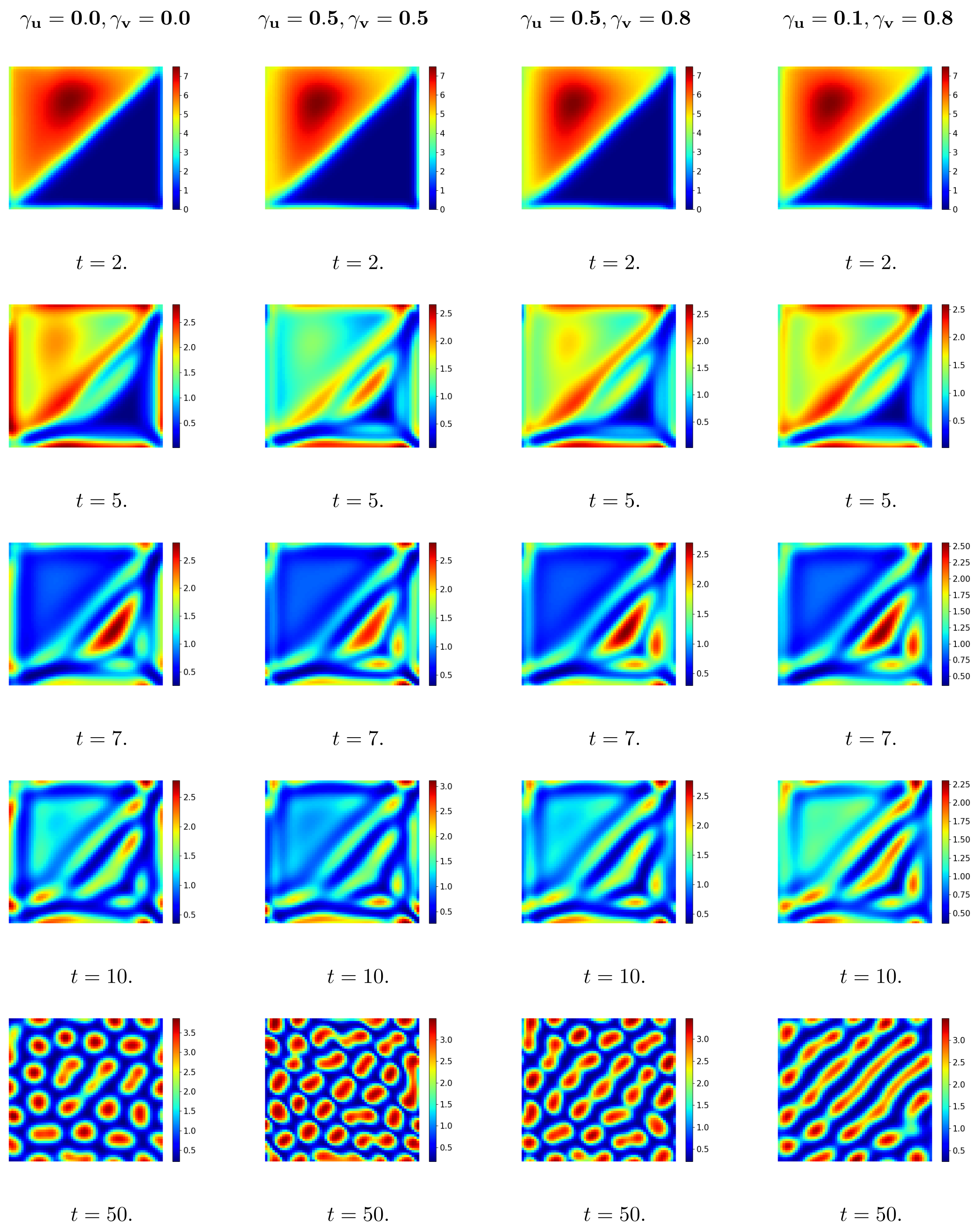


Figure 19: Spatio-temporal development of prey for the four previous scenarios at time $t = [2, 5, 7, 10, 50]$.

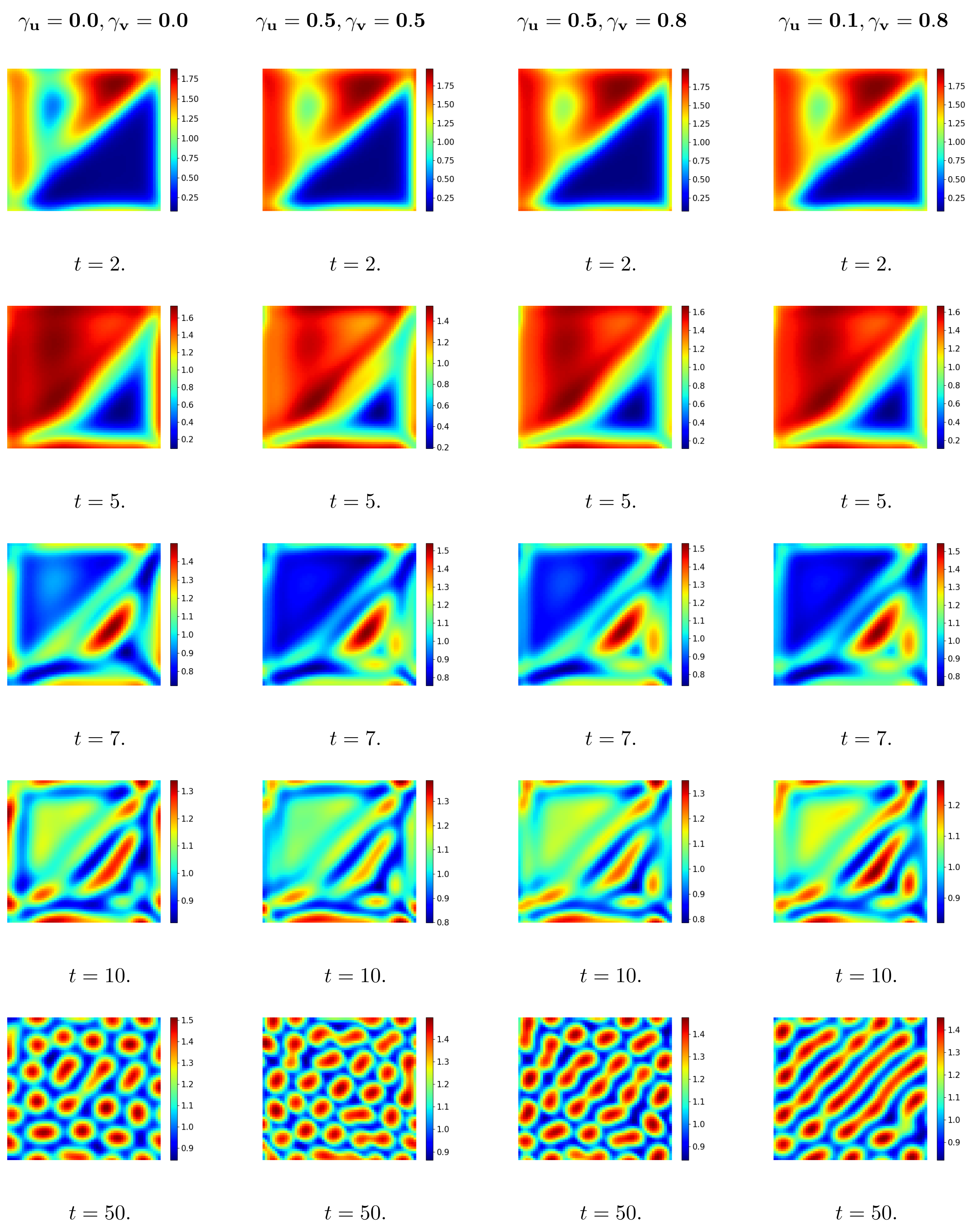


Figure 20: Spatio-temporal development of predator for the the four previous scenarios at time $t = [2, 5, 7, 10, 50]$.

The four scenarios do not deviate much from one another when looking at the early spatial development in time. The prey population, seen in Figure 19, with the initial condition from Figure 8a, will blossom up very quickly in the upper left half of the domain. Here the population density increases towards the value of the carrying capacity, i.e. $K \approx 9.33$. No predator is initially present in this area so the predator has to move before being able to reach the prey and eat it. As a consequence the prey has great growth conditions in the initial time steps.

The predator is by far the fastest diffusing species of the two, and it is seen that already at time $t = 2$ almost all the predator population located at the initial position has moved or died. The predators are now located in the same upper left half as where the prey is located. This gives great growth conditions for the predator, and at time $t = 5$ the population is very large. No predator is left at the center of the lower right half of the domain at this time, since the prey has not yet moved to here, and the predator can thereby not survive in this area.

At this same time step, $t = 5$, the prey population is sharply reduced as a consequence of the presence of the predator and the prey starts to move away. This results in an amount of prey located in the middle of the lower right part of the domain. Since the prey is already highly reduced at time $t = 5$, the predator population quickly decreases, due to the lacking amount of prey.

At time $t = 7$ the two population densities are thereby mainly located at the same place, the place where the prey started escaping towards, i.e. in the middle of the lower right part of the domain. At this point, only slight variations in the population sizes are seen. The population of both species have grown rapidly, decreased rapidly and will now start to slowly to increase towards a new inhomgeneous stable solution. This part is thus the beginning of the stable pattern formation.

At time $t = 50$ the patterns for all scenarios has already almost formed, and in regard to the last scenario, the stripes will only develop more, which is seen by comparing the pattern at time $t = 50$ to that of time $t = 100$ from Figure 13a and Figure 13c for the prey and predator respectively. In the remaining three situations, the clusters will only separate more over time.

Another observation for the above scenarios is that the patterns will form faster when less taxis is added. This is most easily understood by looking at the more detail spatio-temporal development of the patterns in Appendix E.1, for which several time step between $t = 10$ and $t = 50$ is shown.

4.6 Summary

The different scenarios computed in this chapter all show a very similar behavior. Mostly hexagonal arrangement of spots are seen in the results. When taxis is added the number of spots slightly increases and these spots are thereby located closer to one another. The patterns' formation are thereby similar, they are both shaped equally and formed and stabilized around the same time. This is what is expected, since for all of the situations, only a small amount of taxis is added.

The main observation for these scenarios regards the last of the three simulations with taxis. Here some effect on the amount of taxis in the system comes into account for the computations in Julia compared to the results computed in COMSOL. In relation to the results from the former simulations this can either be an effect of the ratio between the two taxis parameters, γ_u/γ_v or γ_v/γ_u, or it can be as a consequence of the higher amount of taxis, or a combination. One should also note that no exact results exist for this model type which means that whether results from COMSOL or Julia are correct is in fact not known.

In Julia no stripes are seen for computations with the parameter values $\gamma_u = 0.5$ and $\gamma_v = 0.8$. This indicates that a high ratio between the parameters γ_v/γ_u might give rise to the stripes seen in Figure 14a and Figure 14c. Here it is thus important to notice that one common Turing type pattern is in fact the formation of stripes. Since the simulations are all within the behavior of a standard reaction-diffusion like model, the program written in Julia might just be more sensitive to the mechanism that drives the formation of stripes compared to COMSOL.

In order for these computations from the two programs to look more alike, the spatial discretization of the advection term used to compute results in Julia has been investigated. An upwind and a downwind discretization has been implemented, but both with no noticeable change in the solution. As a different approach the time integration has also been changed to seek for a better match in the results. Here the part solved implicitly and the part solved explicit is changed, so that the advection term has been tried solving implicitly as well. Again this gave rise to the same results.

The effect of the initial conditions is also shown to be of very little importance. What in general can be concluded when changing the initial conditions is that these do not effect the formed pattern much. Separating the two population densities at the initial state showed that the predator population is able to grow as soon as any amount of predator reaches the prey. Even though, from a biological point of view, no predator is present when the population density goes below some lower threshold, this is not predicted by the model. This indicates that implementing a cut-off below some specified threshold for the population densities would make the model more consistent with real life scenarios.

Results that shows what happens when the amount of prey taxis is high, maybe even higher than predator taxis, is for this model not possible to compute. This is due to the fact that the model only gives results for a small influence of the taxis term, especially small influence from the prey taxis term. This is the case for both computations in `COMSOL` and in Julia. Even for values within the area of `CASE 3`, i.e. the diffusion-driven part, spatio-temporal solutions can not be computed. When the amount of prey taxis becomes too large, i.e. for $\gamma_u = \gamma_v = 0.8$ or $\gamma_u = 0.8, \gamma_v = 0.1$, no solutions can be found. In `COMSOL` some results with more prey taxis present can thus be computed, for instance $\gamma_u = 1.0, \gamma_v = 1.0$, but these give rise to large oscillations in the results.

The problem that arises when the value of the taxis coefficients is increased stems from the wavelengths becoming infinitely short, i.e the frequency grows unbounded. This requires a finer discretization in space which then require a finer discretization in time and as a consequence instabilities in the model will grow too heavily. This phenomenon is known as the *the ultra violet catastrophe* and due to this the model will no longer be well-posed.

To simulate for larger values of the taxis parameters, and thereby to seek pattern formation in the remaining two areas as well, i.e. the areas regarding `CASE 1` and `CASE 2`, long range effects must be included into the model in order to dampen the high frequencies. One approach is to implement a higher order term to the model. This can be done in many ways, but one is just to add a fourth order term to the model. This would thereby damp the instabilities, i.e. the exploding waves and such a model looks like seen in the following.

$$\frac{\partial u}{\partial t} = f(u,v)u - \gamma_u \nabla \cdot u \nabla f(u,v) + D_u \nabla^2 u + a_u \nabla^4 u \tag{49a}$$

$$\frac{\partial v}{\partial t} = g(u,v)v - \gamma_v \nabla \cdot v \nabla g(u,v) + D_v \nabla^2 v + a_v \nabla^4 v \tag{49b}$$

With both parameters a_u and a_v positive. This model has been investigated in both

Julia and `COMSOL`, with different values of the two parameters a_u and a_v, but without giving any usable results.

As a different approach, an integral equation formulation has been implemented in the model. This require a reformulation of the population dynamics and as a results a new model is proposed. This formulation can be very computationally costly whereas adding a higher order term would be less costly. From a biological point of view it is thus more satisfactory to incorporate the long term effects with the integration formulation than with adding a higher order term. To determine the spatio-temporal properties of the new model the integration kernel has to be specified and the modified model will be presented in the next chapter.

5 Adjustment to the Model

With the integral formulation the rate of change of a species, u or v, at position (x, y) at time t will now depend on the influence of all neighboring species or populations at all other neighboring points $(\bar{x}, \bar{y})$ [4].

This is implemented with a so-called *kernel function* in the equations for the population dynamics (1a)-(1b), to be specific, in the expression for the feeding interaction [2]. The effect of the neighboring species, for instance the effect of $u(\bar{x}, \bar{y}, t)$ on $u(x, y, t)$, is quantified by the kernel function, where the form of this kernel assumes that the influence depends only on the distance between (x, y) and $(\bar{x}, \bar{y})$. It is also called a smoothing, or an integration kernel, and as a result of implementing the kernel, the expression for the local dynamics will now depend on x and y as well.

This chapter is thereby divided in to three parts, where the first section introduces the Gaussian function. For this study this function will be used as the shape of the integration kernel. The reformulation of the population dynamics due to the implementation of the integral formulation will be explained in the second part of this chapter. Finally, some small comments on the stability analysis of the new model will be given in the last section.

5.1 Integration Kernel

The Gaussian function used as the integration integration kernel is expressed below.

$$\Phi(\bar{x}, x, \bar{y}, y) = \frac{1}{2\pi\sigma_x\sigma_y} \mathrm{e}^{-\left(\dfrac{(\bar{x}-x)^2}{2\sigma_x^2} + \dfrac{(\bar{y}-y)^2}{2\sigma_y^2}\right)} \tag{50}$$

The kernel is considered in two dimensions and for an infinite space domain, $(\bar{x}, \bar{y}) \in \Omega = \mathbb{R}^2$. The implementation of the kernel results in a system of one or two additional parameters. In two dimensions this is due to the fact that the width of the integration kernel, the standard deviation, is required, which corresponds to σ_x and σ_y for each of the two spatial dimensions. Setting $\sigma_x = \sigma_y$, corresponding to a symmetric kernel, will only add one more parameter, leading to a model of five parameters in total. For $\sigma_x \neq \sigma_y$ the function is elliptic and the model will be a system of six parameters. For the rest of the study, $\sigma_x = \sigma_y = \sigma$ and σ^2 equals the variance.

The standard deviation concerns the spread of the Gaussian bell and the larger the standard deviation, the broader the peak, which also corresponds to a greater influence on the neighboring species. The degree of smoothing is thereby determined by the standard deviation, but a large standard deviation of the Gaussian will, of course, require a larger convolution kernel in order to be accurately represented. The amplitude $\frac{1}{2\pi\sigma^2}$ is the height of the peak of the Gaussian distribution. This is also the normalization constant that ensures that the distribution integrates to one. The Gaussian function is illustrated in the following figure, Figure 21.

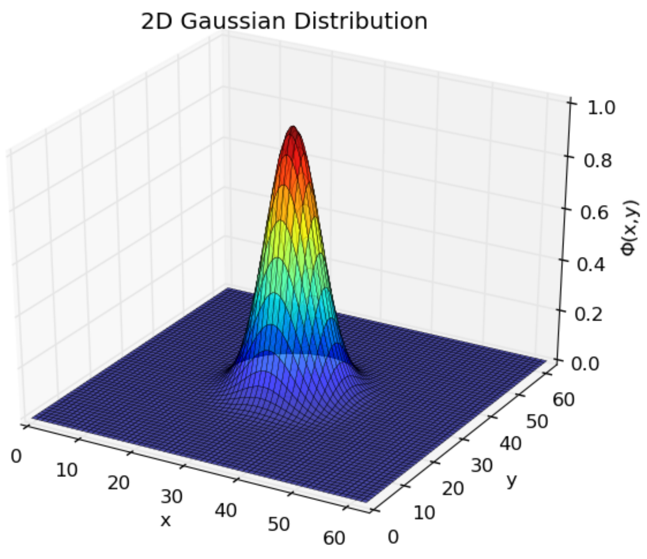


Figure 21: Gaussian function in 2D

The idea of Gaussian smoothing is to use its two-dimensional distribution as a *point-spread* function, which is achieved by convolution. The Gaussian kernel is continuous, but since the amount of prey and predator is stored as a collection of discrete points, a discrete approximation to the Gaussian function must be produced before convolution can be performed. For the purpose of this study, the discrete equivalent is obtained by sampling the continuous Gaussian in discrete points over the entire domain which can be done, as long as the width of the Gaussian function does not become too small compared to the grid size. Since the boundary conditions in all simulations are periodic, see Section 3.1, the kernel also needs to be periodic in the entire space domain. Normalization of the kernel function is also required, in order for the equilibrium solution from the original system (16a)-(16b) to also be an equilibrium for the new modified system.

When the normalization is done along with the periodicity of the Gaussian function, smoothing with the kernel ensures the amount of prey and predator will remain the same.

5.2 Modified Non-Local Population Dynamics

The idea behind the implementation of the Gaussian kernel, is to allow a predator to eat prey in an *area* around it. This is in contrast to the point-wise local formulation, where only prey located at the exact same point as the predator can be eaten by that predator.

In the equations for the local dynamics (1a)-(1b), only the interacting term expresses a 'meeting' between the two species, i.e. a situation for with the distance between a predator and a prey have an inflect. Based on that, the kernel will only be implemented in this term. From the generalized system (2a)-(2b) this term is the function $B(u, v) = \eta(u)v$, expressing the feeding rate of the predator on the prey. In the following only the dependence of u and v on space will be emphasized, and so the dependence on time will be suppressed. The amount of prey that encounters a predator localized at position (x, y) can be described by an integral over the entire space domain Ω [2].

$$(\mathcal{U}u)(x,y) = \int_\Omega \int_\Omega u(\bar{x},\bar{y})\Phi(\bar{x},\bar{y},x,y)\mathrm{d}\bar{x}\mathrm{d}\bar{y} \tag{51}$$

Equivalently the above equation is thus the amount of prey a predator at position (x, y) have access to. Φ is called the feeding kernel and this describes how the attention of a predator at position (x, y) is distributed around it in space.

$\Phi(\bar{x}, \bar{y}, x, y)$ can also be seen as the probability of a predator at position (x, y) meeting a prey at position $(\bar{x}, \bar{y})$. When the Gaussian function, Equation (50), is implemented as the feeding kernel, the nearest neighboring prey will have the greatest influence, i.e. will face the biggest risk of being eaten. This influence will decrease with increasing distance from the predator located at position (x, y).

The feeding rate, or interaction rate, consists of two parts, the *attack rate* and the *encounter rate* [2].

$$\eta(u)v = \underbrace{\frac{1}{1+u}}_{\text{attack rate}} \cdot \underbrace{auv}_{\text{encounter rate}} \tag{52}$$

where $\eta(u)$ as earlier mentioned is the functional response, see Section 2.1. The first part, the *attack rate*, defines how likely the predator is to attack. This term incorporates a satiety factor for the predator, meaning that when the amount of prey is very large, the predator will not attack as much as if the amount of prey is smaller. This can be explained as the predator being satiated when there is a large amount of prey, but becoming more and more hungry, and thereby more likely to attack, when the amount of prey is decreasing.

The encounter rate describes the 'meeting' between the two species, and for the former local definition, this is the part that tells that only if the two species are at the exact same position, will they meet.

$$au(\bar{x},\bar{y})\delta(\bar{x}-x,\bar{y}-y)v(x,y) \tag{53}$$

The above equation details this definition, since the 'meeting' between the predator localized at position (x,y) and the prey at position $(\bar{x},\bar{y})$, is given by the amount of predator and prey at the two positions respectively, times the δ-function. This means, that only when the the positions are equal, $(x,y)=(\bar{x},\bar{y})$, will the expression in Equation (53) be positive.

For the new non-local definition a new encounter rate is defined.

$$au(\bar{x},\bar{y})\Phi(\bar{x},\bar{y},x,y)v(x,y) \tag{54}$$

Here the kernel function is incorporated as the distance between predator and prey instead of the δ-function. This leads to the definition of a predator at position (x,y) being able to eat prey in an area around it and so the model is no longer completely localized.

The new attack rate is in combination with the integral from Equation (51) defined by

$$\frac{1}{1+(\mathcal{U}u)(x,y)} \tag{55}$$

This allows the attacking prey at position (x,y), to eat prey within a neighboring area, since the amount of prey is now distributed in an area around the prey due to the integral from Equation (51).

To summarize, the new interacting/feeding term becomes

$$h(\bar{x},\bar{y},x,y)=\underbrace{\frac{1}{1+(\mathcal{U}u)(x,y)}}_{\text{new attack rate}}\cdot\underbrace{au(\bar{x},\bar{y})\Phi(\bar{x},\bar{y},x,y)v(x,y)}_{\text{new encounter rate}} \tag{56}$$

This expresses the feeding rate of the predators at position (x,y) on the prey located at position $(\bar{x},\bar{y})$.

The total amount of prey eaten at position $(\bar{x},\bar{y})$ is found by integrating the new feeding rate, Equation (56), over all predator positions (x,y).

$$\begin{aligned} H_u(\bar{x},\bar{y}) &= \int_\Omega\int_\Omega h(\bar{x},\bar{y},x,y)\mathrm{d}x\mathrm{d}y \\ &= au(\bar{x},\bar{y})\int_\Omega\int_\Omega \frac{1}{1+(\mathcal{U}u)(x,y)}\Phi(\bar{x},\bar{y},x,y)v(x,y)\mathrm{d}x\mathrm{d}y \end{aligned} \tag{57}$$

The feeding rate of the predator at position (x, y) on the prey located at position $(\bar{x}, \bar{y})$ is found by integrating the new feeding rate, Equation (56), over all prey positions $(\bar{x}, \bar{y})$.

$$\begin{aligned} H_v(x,y) &= \int_\Omega \int_\Omega h(\bar{x},\bar{y},x,y)\mathrm{d}\bar{x}\mathrm{d}\bar{y} \\ &= av(x,y)\int_\Omega \int_\Omega \frac{1}{1+(\mathcal{U}u)(x,y)} u(\bar{x},\bar{y})\Phi(\bar{x},\bar{y},x,y)\mathrm{d}\bar{x}\mathrm{d}\bar{y} \qquad (58) \end{aligned}$$

$H_u(\bar{x}, \bar{y})$ must be equal to $H_v(x, y)$, since the feeding rate must be the same. Implementing the two new feeding rates $H_u(\bar{x}, \bar{y})$ and $H_v(x, y)$ instead of $\eta(u)v$ in the equations for the net growth rates, Equation (5a)-(5b), for the prey and predator respectively, results in the following [2].

$$\begin{aligned} \mathcal{F}(u,v)(\bar{x},\bar{y})u(\bar{x},\bar{y}) &= ru(\bar{x},\bar{y})\left(1-\frac{u(\bar{x},\bar{y})}{K}\right) \\ &\quad - au(\bar{x},\bar{y})\int_\Omega\int_\Omega \frac{\Phi(\bar{x},\bar{y},x,y)v(x,y)}{1+(\mathcal{U}u)(x,y)}\mathrm{d}x\mathrm{d}y \qquad (59a) \\ \mathcal{G}(u,v)(x,y)v(x,y) &= av(x,y)\frac{(\mathcal{U}u)(x,y)}{1+(\mathcal{U}u)(x,y)} \\ &\quad - v(x,y) - cv(x,y)^2 \qquad (59b) \end{aligned}$$

The growth rates are now expressed by the operators $\mathcal{F}(u, v)$ and $\mathcal{G}(u, v)$, instead of the previous two functions $f(u, v)$ and $g(u, v)$ respectively. In $\mathcal{G}(u, v)$ the amount of prey that the predator grows with will be smoothed, so that the prey near the predator will have an affect. The system can be written in a more compact way

$$\mathcal{F}(u,v) = r\left(1-\frac{u}{K}\right) - a\mathcal{K}_1\left(\frac{v}{1+\mathcal{K}_2(u)}\right) \qquad (60a)$$

$$\mathcal{G}(u,v) = \frac{a\mathcal{K}_2(u)}{1+\mathcal{K}_2(u)} - 1 - cv \qquad (60b)$$

where the two linear operators $\mathcal{K}_1$ and $\mathcal{K}_2$ are defined by

$$\begin{aligned} (\mathcal{K}_1 w)(\bar{x},\bar{y}) &= \int_\Omega\int_\Omega w(x,y)\Phi(\bar{x},\bar{y},x,y)\mathrm{d}x\mathrm{d}y \\ (\mathcal{K}_2 w)(x,y) &= \int_\Omega\int_\Omega w(\bar{x},\bar{y})\Phi(\bar{x},\bar{y},x,y)\mathrm{d}\bar{x}\mathrm{d}\bar{y} \end{aligned}$$

here w is the solution, i.e. u or v [2].
Since the kernel is symmetric, meaning $\Phi(\bar{x}, \bar{y}, x, y) = \Phi(x, y, \bar{x}, \bar{y})$, then $\mathcal{K}_1 = \mathcal{K}_2 = \mathcal{K}$ [2]. The kernel is a function of the distance between the predator and the prey and this implies that $\Phi(\bar{x}, \bar{y}, x, y) = \phi(\bar{x} - x, \bar{y} - y)$. This means that the operator $\mathcal{K}$ obtains the

form of a convolution operation, defined by $'*'$. Convolution implies taking two signals to produce a third signal. In this conjunction, the input signal (typically the position of the prey, u) convoluted with the impulse response (the Gaussian function) is equal to the output signal, a smoothing of the input signal. The convolution is given by

$$(\mathcal{K}w)(\bar{x},\bar{y}) = (w * \phi)(\bar{x},\bar{y}) = \int_{\mathbb{R}}\int_{\mathbb{R}} w(x,y)\phi(\bar{x}-x,\bar{y}-y)\mathrm{d}x\mathrm{d}y \tag{61}$$

Such a model incorporates long range effects through the kernel ϕ. If the peak of the kernel is high, i.e. the kernel tends to zero quickly, then the long range effects will be weak. If, on the other hand the peak is small, then the kernel tends slowly towards zero, and the long range effects will then be strong.

Finally the modified, non local dynamics, becomes

$$\mathcal{F}(u,v) = r\left(1-\frac{u}{K}\right) - a\mathcal{K}\left(\frac{v}{1+\mathcal{K}(u)}\right) \tag{62a}$$

$$\mathcal{G}(u,v) = \frac{a\mathcal{K}(u)}{1+\mathcal{K}(u)} - 1 - cv \tag{62b}$$

and the full fitness taxis model with the kernel function implemented now becomes [2].

$$\frac{\partial u}{\partial t} = \mathcal{F}(u,v)u - \gamma_u\nabla\cdot u\nabla\mathcal{F}(u,v) + D_u\nabla^2 u \tag{63a}$$

$$\frac{\partial v}{\partial t} = \mathcal{G}(u,v)v - \gamma_v\nabla\cdot v\nabla\mathcal{G}(u,v) + D_v\nabla^2 v \tag{63b}$$

The kernel could have been implemented such that the entire expressions, $f(u,v)u$ and $g(u,v)v$ for the local dynamics (1a)-(1b) was smoothed out. Thus, in conjunction with a biological perspective, implementing the kernel only in the interaction term, as done here, makes far more sense.

5.3 Stability of the New Model

Just as done in section 2.3.1 for the full fitness taxis model, a stability analysis for the model with the integration kernel must also be made, in order to investigate conditions for pattern formation. First of all it should be noted that as long as $\sigma > 0$ the model is well-posed. Stability of the model with the integration kernel (63a)-(63b) will not give rise to new areas with patterns and the same conditions for generating patterns as explained in Section 2.3.2 will apply. This means that if the model without the kernel, i.e. the fitness taxis model (16a)-(16b), predicts a pattern formation and the model with the implemented kernel have a 'small enough' kernel, then this model will also predict patterns.

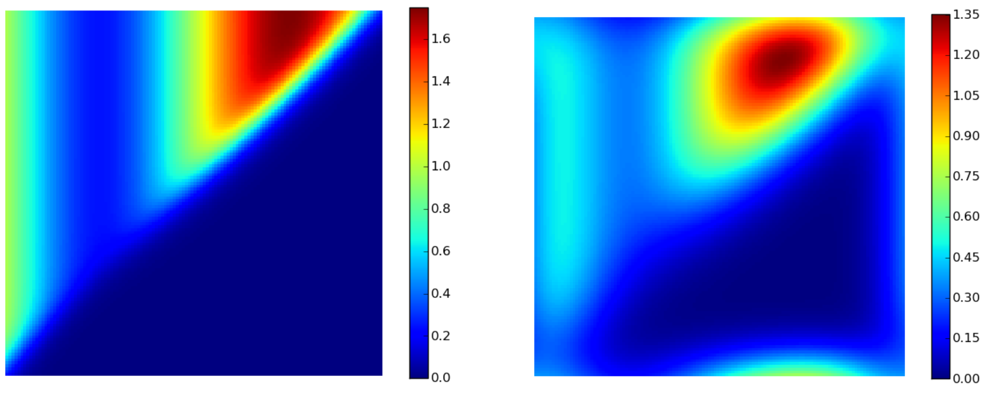
The width of the kernel will thus be of significance in the search for pattern formation since broadening the kernel can cause a lowering in the requirements for which patterns are able to form. The trace conditions will for instance be weakened by an increase in the kernel width, σ. The trace will at some point become negative for all $k^2 > 0$ when the Gaussian bell is broadened. This means that the system will always move towards the stable equilibrium point for all $k^2 > 0$. From the stability analysis of the kernel it can be seen that the implementation of the kernel damps the long range instability effects. A further examination of the stability of the model with the integration kernel is presented in [2].

6 Simulations with Integration Kernel

This chapter will address simulations of the fitness taxis model performed in Julia with the integration kernel implemented. This allows for simulations where more advection are added compared to the previous result, see Section 4. It is divided in to seven parts where the effect of the kernel initially will be presented in the first two sections. This is followed up in the third section by simulations regarding `CASE 3` as previously shown. Thereafter, most importantly, simulations regarding the remaining two cases, `CASE 2` and `CASE 1` as well as the overlapping region of these two cases are considered in the fourth, fifth and sixth section. At last a summary of the results seen in this chapter will be given in the seventh section.

6.1 Integration Kernel

The effect of the kernel can be visualized, which is done by plotting the blurred initial distribution of the prey as an example. This can be seen in the following figure where Figure 22a shows the initial prey density and Figure 22b shows the filtered image of the initial prey density. This is computed as expressed in Equation (61), where the kernel function is the Gaussian distribution, see Equation (50), and for this example, $\sigma = 5.0$. When the size of the kernel, as for this example, is broad, the influence on the filter, the integration kernel, is high, and the filtered positions will be smoothed a lot.



(a) The initial distrubiton of prey

(b) Filtered initial condition of the prey with $\sigma = 5.0$.

Figure 22: Filtering with Gaussian Kernel

When the initial distribution is smoothed, the distribution will be more flat, i.e. the peak from the non-smoothed distribution is a lot higher than for the smoothed distribution. The filter is periodic, so the distribution is smoothed out in all direction, and thereby no amount of prey is lost when filtered.

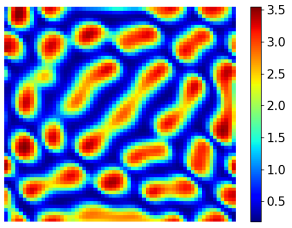
In relevance to the new non-local formulation of the population dynamics the model (63a)-(63b), Figure 22b illustrates the amount of prey available for the predator. It is thereby not a change of the initial distribution of the prey, since this will still be as in Figure 22a.

Simulating with a large standard deviation results in a broad Gaussian function. Convolution with a broad kernel is very computationally costly, but a broader kernel is required when more taxis is added, since the neighboring effect will be greater, and thereby the influence on the long term effects will play a bigger role, which is required in order for the local model (16a)-(16b) to be well posed.

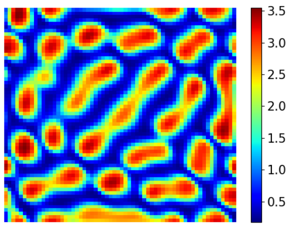
6.2 Comparison *with* and *without* Kernel

Results from Section 4.3.2 will be used for the purpose of comparing results computed with and without the integration kernel. These results regard the scenario for which γ_u and $\gamma_v = 0.5$. For this exact scenario the width of the kernel can be chosen as desired since no kernel is actually needed in order to compute results. It is thus important to notice, that the grid size must be sufficiently fine in each spatial direction in order to represent the Gauss function for the given standard deviation.

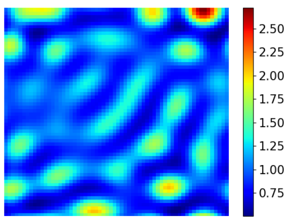
Three different situations will be shown, one computed with the standard fitness taxis model (16a)-(16a) and the remaining two with the model where the integration kernel is implemented (63a)-(63b). These last two situations will regard the values $\sigma = 0.2$ and $\sigma = 0.7$. For $\sigma = 0.2$ the results are seen in Figure 23b and Figure 23e for the prey and predator respectively. For $\sigma = 0.7$ the results are seen in Figure 23c and Figure 23f for the prey and predator respectively. The first column in Figure 23 are the results without the integration kernel for the prey and predator respectively. All three situations are show for time $t = 30$.



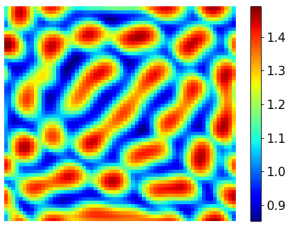
(a) Distribution of prey at time $t = 30$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.5$.



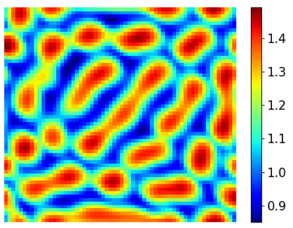
(b) Distribution of prey at time $t = 30$ for the model (63a)-(63b) with $\gamma_u = 0.5$,$\gamma_v = 0.5$ and $\sigma = 0.2$.



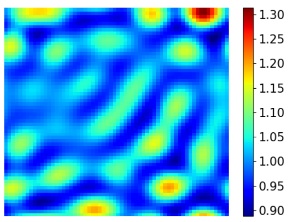
(c) Distribution of prey at time $t = 30$ for the model (63a)-(63b) with $\gamma_u = 0.5$,$\gamma_v = 0.5$ and $\sigma = 0.7$.



(d) Distribution of predator at time $t = 30$ for the model (16a)-(16b) with $\gamma_u = 0.5$,$\gamma_v = 0.5$.



(e) Distribution of predator at time $t = 30$ for the model (63a)-(63b) with $\gamma_u = 0.5$,$\gamma_v = 0.5$ and $\sigma = 0.2$.

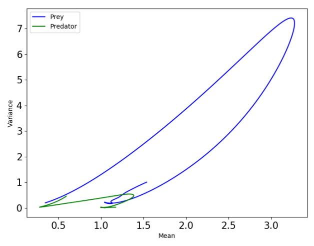


(f) Distribution of predator at time $t = 30$ for the model (63a)-(63b) with $\gamma_u = 0.5$,$\gamma_v = 0.5$ and $\sigma = 0.7$.

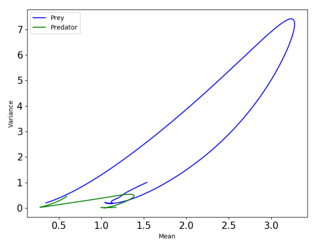
Figure 23: Distribution of prey (top) and predator (bottom) at time $t = 30$ for three different scenarios. To the left the model (16a)-(16b). In the middle and to the right the model (63a)-(63b) with $\sigma = 0.2$ and $\sigma = 0.7$ respectively. All simulations regarding with $\gamma_u = 0.5$,$\gamma_v = 0.5$

The effect of the kernel is shown very clearly in these three simulations. The last example with the kernel width $\sigma = 0.7$ shows how the patterns are smoothed so the inhomogeneity is less prominent. As a result the width of the Gaussian bell can become as broad, so that the disturbed system will just relax back to its equilibrium solution (u^*, v^*). This is an important observation for further simulations, since adding more taxis in most cases needs a broad kernel, but this kernel might also be as broad that no patterns are formed. The results computed with the model without integration kernel can be reproduced with the model where the kernel is implemented. With $\sigma = 0.01$ this there is effectively no kernel in the model (63a)-(63a). The kernel is too small, which means that this corresponds to $\sigma = 0$ which reduces the model with the non local dynamics (63a)-(63b) to the fitness taxis model without the integration kernel (16a)-

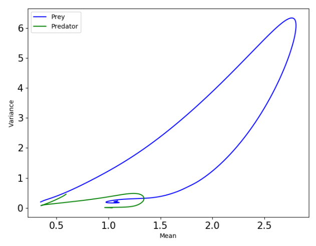
(16b). Computing results with both models, where $\sigma = 0.01$ for the former, the results are the exact same, but the computations with the fitness taxis model with no kernel (16a)-(16b) is way faster. The results are thus the same, but it is computationally costly to use the model (63a)-(63b).
Following plots are the mean and variance plotted over time for the three situations above, i.e. with no integration kernel, $\sigma = 0.2$ and $\sigma = 0.7$, but with the exact same advection parameters.



(a) The mean and variance for each time step without integration kernel



(b) The mean and variance for each time step with $\sigma = 0.2$



(c) The mean and variance for each time step with $\sigma = 0.7$

Figure 24: The mean and variance for each time step computed without integration kernel (left), with $\sigma = 0.2$ (middle) and $\sigma = 0.7$ (right).

Initially both the mean and variance in all three situations are small. Since the population blossoms, both the mean and variance will increase rapidly only to decrease again after a short amount of time. This is seen for both population densities, but is most significant for the prey population. For the simulations with the broadest kernel the values for the mean and variance do not vary as much as for the remaining two situations which is in good correspondence with the observations from Figure 23. It is also seen, especially for the prey population, that when the values for the mean and variance have decreased to approximately a mean value equal to one and variance equal to zero, both values starts to increase slowly again. For the third situation with the broad kernel, this is not the case, since the population will just relax to the equilibrium solution resulting in a constant mean value equal to one and the variance equal to zero.

In the first two situations the patterns will form and both mean and variance will increase. The mean value will increase due to an increase in the population density and the variance will increase due to the inhomogeneous spatial distribution of the population densities.

6.3 Fitness Taxis in `CASE 3`

Specifying the taxis coefficients in order for the full system (63a)-(63b) to fulfill the conditions regarding `CASE 3`, the system is said to be *diffusion* driven, see Section 2.3.2. For simplicity the bifurcation diagram from Section 2.3.2 is shown again.

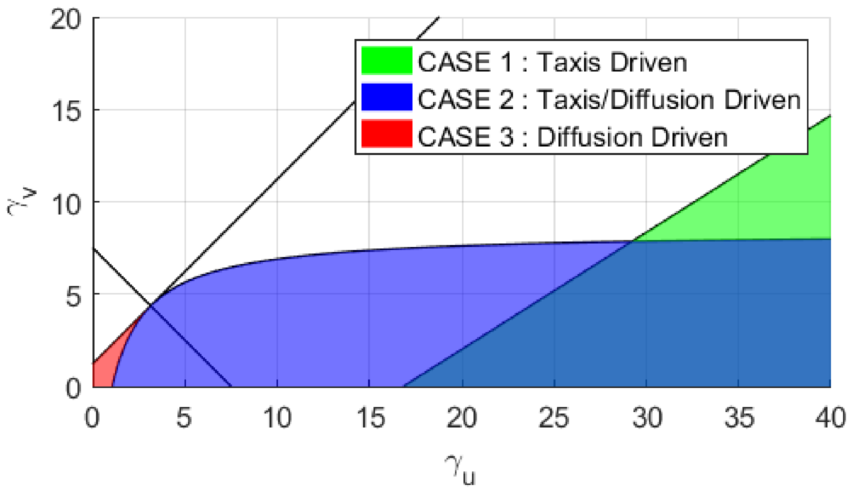
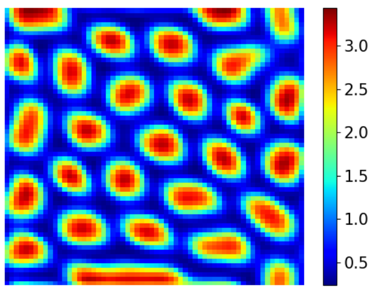


Figure 25: Sketch of the bifurcation diagram for the advection parameters γ_u, γ_v with $D_u = 1$ and $D_v = 15$.

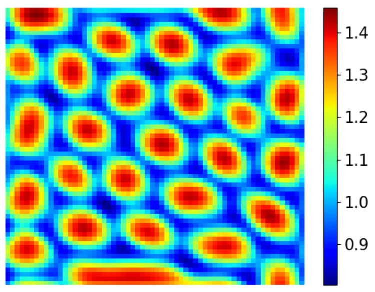
From Section 4 it was experienced that the fitness taxis model with local population dynamics (16a)-(16b) for some values of γ_u and γ_v within this reaction-diffusion like area, failed to give results. Already for these small values of the advection parameters, the model (16a)-(16b) loses its well-posedness. This means that no solutions can be found within this area. In this section a scenario unsolvable for the model without the integration kernel is thus simulated with the kernel implemented.

6.3.1 Scenario $\gamma_u = 1.0, \gamma_v = 1.0$

Different kernel sizes have been tested, and for this situation the kernel needs to have the size $\sigma \geq 0.5$ in order to give results. For this specific values, the results are shown for the prey distribution at time $t = 100$ in Figure 26a and for the predator distribution at time $t = 100$ in Figure 26b.



(a) Distribution of prey at time t = 100 for the model (63a)-(63b) with γ_u = 1.0,γ_v = 1.0 and $\sigma = 0.5$.



(b) Distribution of predator at time t = 100 for the model (63a)-(63b) with γ_u = 1.0,γ_v = 1.0 and $\sigma = 0.5$.

Figure 26: Distribution of prey (left) and predator (right) at time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 1.0$,$\gamma_v = 1.0$ and $\sigma = 0.5$.

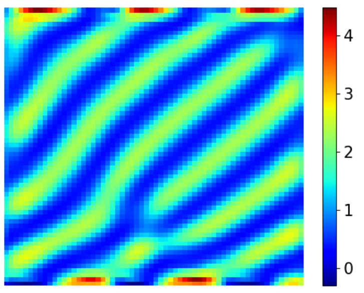
What Figure 26 shows is that the scenario gives rise to pattern formation similar to the ones seen in Section 4.3, in particular for the first two of those scenarios. This means that the new model with the integration kernel is capable of solving the system, where the former model failed. A spatio-temporal development of this scenario from time $t = [1 : 50]$ can be seen in Appendix E.2. With this result the remaining two areas in the bifurcation diagram seen in Figure 25 regarding instability in the full model can now be investigated.

6.4 Fitness Taxis in CASE 2

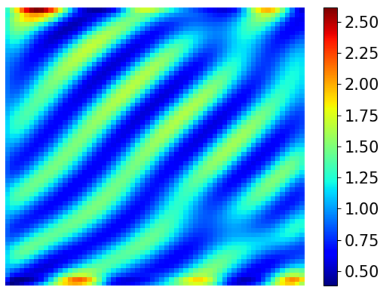
When the values of the two advection parameters falls within the area governed by the conditions for CASE 2, see Section 2.3.2, the system is said to be *taxis/diffusion* driven. In this section two scenarios within this area will be simulated and visualized. The first with an equal value of the two advection parameters but computed with two different kernel widths and the second scenario will be for an increasing amount of prey taxis.

6.4.1 Scenario $\gamma_u = 5.0, \gamma_v = 5.0$

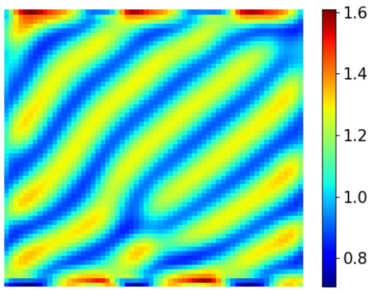
For the first scenario, the value of the two advection parameters is equal. What is previously seen is that the relation between the two taxis parameters might be the reason for the emergence of stripes in the contour plot of the population densities at time $t = 100$. The size of the kernel is also, as mentioned in Section 6.2, of great importance in order for the system to form patterns and not just to revert to the equilibrium solution. In the light of this the model with the specified taxis parameters $\gamma_u = 5.0$ and $\gamma_v = 5.0$ is computed for two different situations, with $\sigma = 0.5$ and $\sigma = 0.7$. The results for the distribution of prey and predator at time $t = 100$ can be seen below, where the first column, namely Figure 27a and Figure 27c shows the results computed with $\sigma = 0.5$ and Figure 27b and Figure 27d shows the results computed with $\sigma = 0.7$.



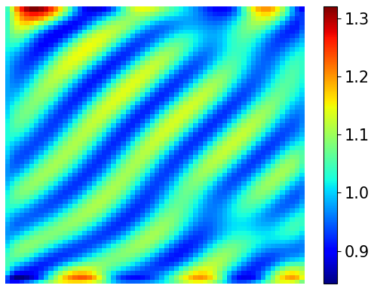
(a) Distribution of prey at time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 5.0$, $\gamma_v = 5.0$ and $\sigma = 0.5$.



(b) Distribution of prey at time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 5.0$, $\gamma_v = 5.0$ and $\sigma = 0.7$.



(c) Distribution of predator at time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 5.0$, $\gamma_v = 5.0$ and $\sigma = 0.5$.



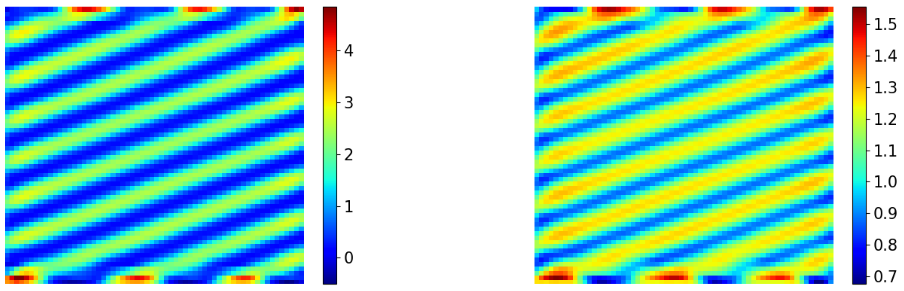
(d) Distribution of predator at time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 5.0$, $\gamma_v = 5.0$ and $\sigma = 0.7$.

Figure 27: Distribution of prey (top) and predator (bottom) at time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 5.0$, $\gamma_v = 5.0$ and $\sigma = 0.5$ and $\sigma = 0.7$, respectively

Both situations shows pattern formation, but here stripes emerge for simulations with both $\sigma = 0.5$ and $\sigma = 0.7$. This indicates that not just the relation between the advection parameters but also a higher amount of taxis all in all affects the formation of stripes. These stripes furthermore no longer spans the entire domain as seen for previous simulations in Section 4.3.3. What is again seen is that a broader kernel reduces the variation in the population density for both species, and so the pattern is flattened. The spatio-temporal development for the simulations with $\sigma = 0.5$ for time $t = [1 : 50]$ are shown in Appendix E.2.

6.4.2 Scenario $\gamma_u = 10.0, \gamma_v = 5.0$

Adding more advection within this will only contribute to the same results regarding the striped patterns seen above. This is seen in the following figure where the distribution of prey at time $t = 100$ is seen in Figure 28a and the distribution of predator at time $t = 100$ is seen in Figure 28a. The width of the kernel is $\sigma = 0.7$ which was found to be the smallest possible size in order to give results for these simulations.



(a) Distribution of prey at time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 10.0$, $\gamma_v = 5.0$ and $\sigma = 0.7$.

(b) Distribution of predator at time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 10.0$, $\gamma_v = 5.0$ and $\sigma = 0.7$.

Figure 28: Distribution of prey and predator at time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 5.0$, $\gamma_v = 5.0$ and $\sigma = 0.7$, respectively

These above results compared to the results from Figure 27 shows that the emerged stripes are both smaller and more straight in this situation. The number of stripes is also higher for these simulations than seen in the previous scenario. This indicates that when the value of the advection parameters are increased, more stripes appears in the formed pattern. The spatio-temporal development for these simulations for time $t = [1 : 50]$ are shown in Appendix E.2.

6.5 Fitness Taxis in CASE 1

This is the third case for which the instability can lead to the formation of inhomogeneous spatial patterns. When the fitness taxis model meets the condition for this area, see

Equation (30), the system is said to be *taxis driven.* This is also the area the model falls within, if no diffusion is present, i.e. completely taxis driven as seen in the bifurcation diagram in Figure 7.

As a common denominator for all simulations where the conditions for this area are met, a broader Gaussian kernel than previously used is required. This means that the neighboring influence will be greater, i.e. the kernel will have a greater influence on the long time effects which is needed in order for the model to give results. The consequence of broadening the Gaussian bell for all test situations regarding this area is thus that the system will relax to the equilibrium solution $(u^*, v^*) = (1, 1)$.

For that reason none of the results at time $t = 100$ is relevant to show, since this is just a plot of the prey and predator population equal to one in the entire domain. Instead only the early spatio-temporal development is included here.

Three test scenarios have been investigated where the first regards the model with the specified values $\gamma_u = 35.0, \gamma_v = 10.0$ and $\gamma_u = 30.0, \gamma_v = 8.0$. Both scenarios are simulated with the smallest possible kernel size in order to give results, which for these simulations can not be smaller than $\sigma = 1.2$. The third scenario differs in the sense that no diffusion is present. The specified values for the advection parameters for this scenario are $\gamma_u = 2.5$ and $\gamma_v = 2.5$ and the kernel width is $\sigma = 4.5$.

The results can be seen below, where Figure 29 shows the development of the prey population for the three scenarios and for the predator population in Figure 30. In each figure the first column is the simulations with values $\gamma_u = 35.0, \gamma_v = 10.0$, the second column shows simulations for the values $\gamma_u = 30.0, \gamma_v = 8.0$ and the third column shows the diffusion free simulations with $\gamma_u = 2.5, \gamma_v = 2.5$. The results shown are for the time $t = [2, 5, 7, 10, 15]$.

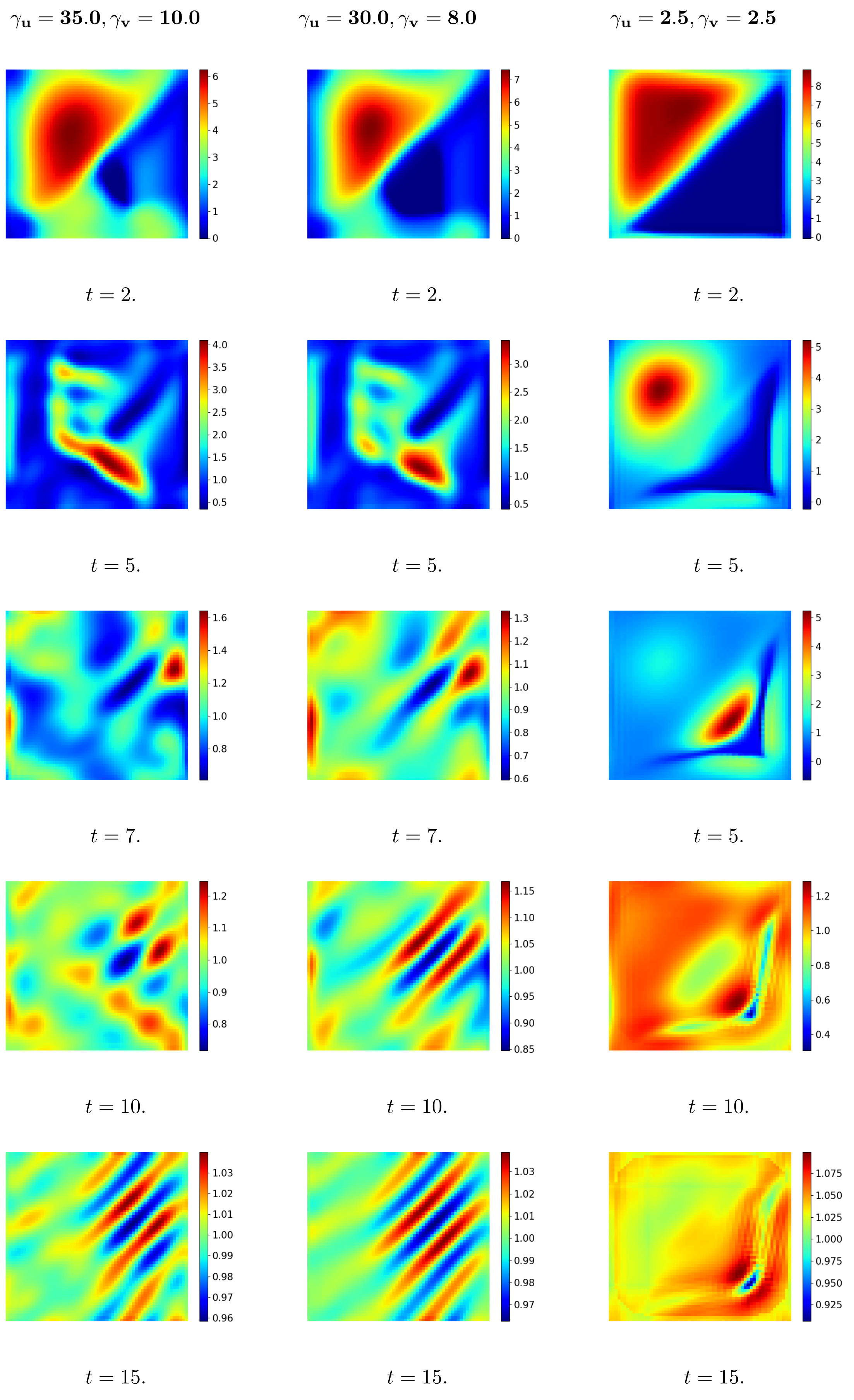


Figure 29: Spatio-temporal development of prey for the three CASE 1 scenarios at time $t = [2, 5, 7, 10, 15]$

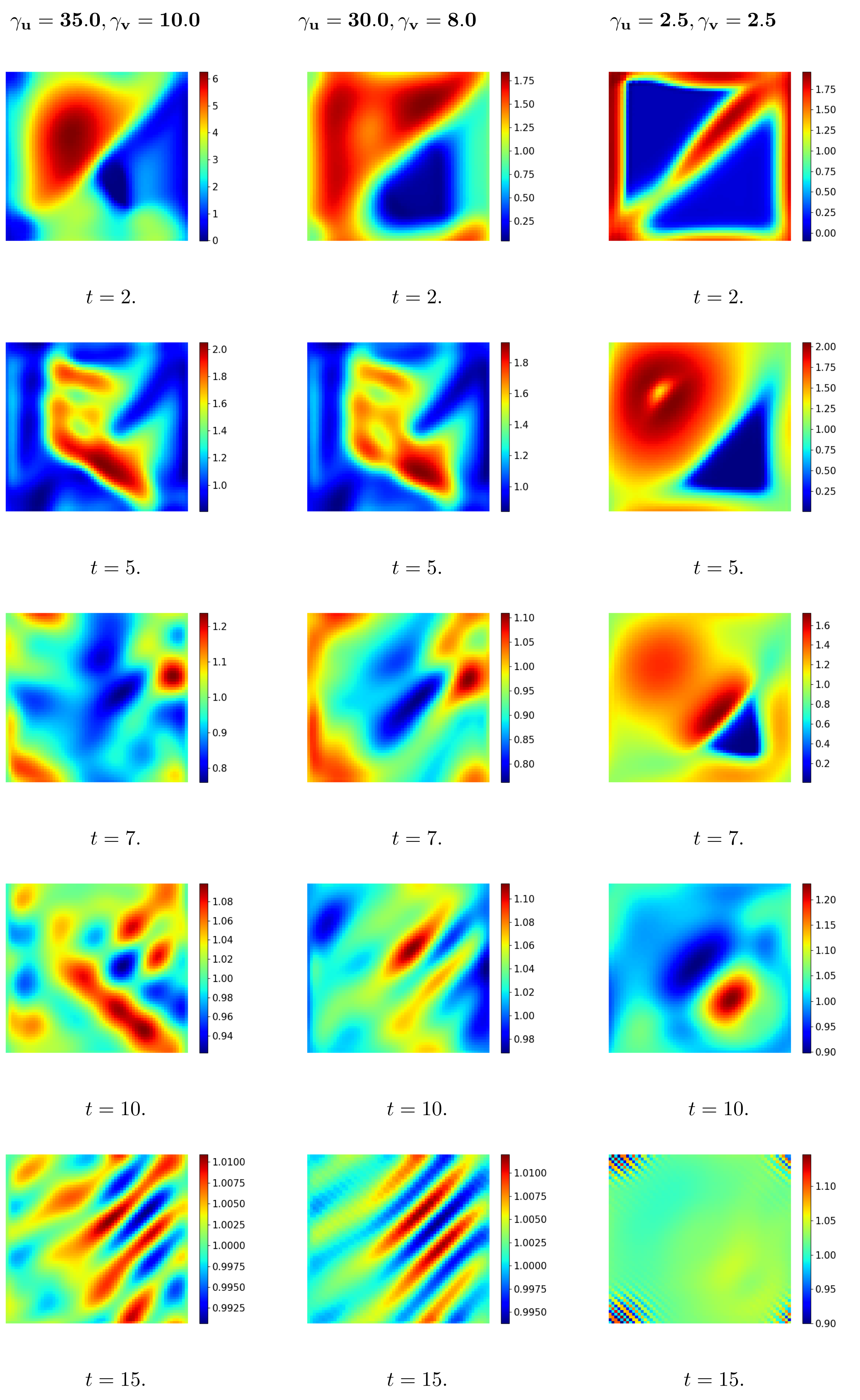


Figure 30: Spatio-temporal development of predator for the three `CASE 1` scenarios at time $t = [2, 5, 7, 10, 15]$

Only the time up until time $t = 15$ is shown which is due to the fact that all three systems already at this time almost have reached the equilibrium solution. The two first situations do not differ a lot from each other. The prey is suddenly capable of moving very fast compared to previous situations, since γ_u is even bigger than the taxis ans diffusion parameter values for the predator.

The first two scenarios are the ones that look the most alike. Only small variations are seen between these two scenarios, but the development differs greatly from the early spatio-temporal development seen in Section 4.5. The prey can now move way faster than previous seen. Already at the initial time steps, the prey population moves away from the upper left half. As seen for all previous scenarios the predator population quickly moves to the upper left half of the domain, since all prey is located here at the beginning. In earlier simulations the escape of the prey from this area is thus first seen at a later time step, but due to the high value of the prey advection parameter in these scenarios, the prey population can now escape faster.

Both species will quickly move towards the center of the domain, but already at time $t = 7$ most fluctuations in the population densities are flattened, and the system will end in its equilibrium solution.

In the last situation, i.e. the diffusion free situation, the movement of the two species is now completely determined from the taxis term. Since the parameter values are equal, the predators do not move faster than the prey. The prey population blossoms up in the entire predator free area in the upper left half of the domain at the initial time steps. The predators, not moving as fast as for all previous simulations, are now located at all boundaries of the prey population. They thereby encapsulate the prey population, which is the reason why the prey population does not move away from this area as is the case in the other two situations for this time step.

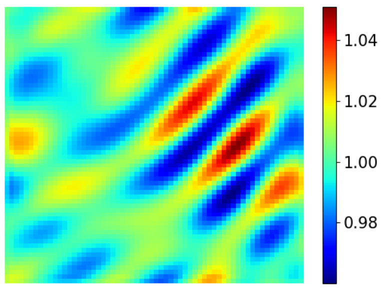
They prey gets eaten, and starts moving away from the predator. Here noting that the width of the kernel is very broad so the predator located at any position in the domain, will see the prey in a great area around, and thereby being able to eat it. The prey starts moving towards the predator-free area in the lower right half of the domain, but with the predator following right after. The populations will reach the equilibrium solution already at time $t = 15$ where the last group of prey is reduced by the predator population, so they are equally distributed in the entire domain. For the specified values this system will also reach the equilibrium solution faster than in the other two situations.

6.6 Fitness Taxis in CASE 1+CASE 2

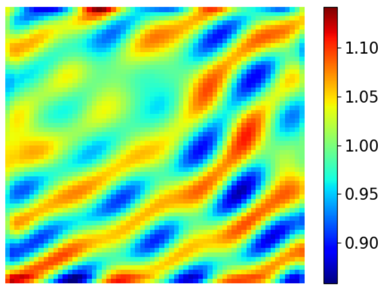
This part regards the overlapping part between the former two cases, i.e. CASE 1 and CASE 2. The situation simulated will be for the smallest amount of taxis possible, still falling within this area. One situation is investigated, but for two different values of the kernel width, σ, namely $\sigma = 1.2$ and $\sigma = 1.1$. These values are almost the same, but the results seen below differs highly from one anther.

6.6.1 Scenario $\gamma_u = 29.0, \gamma_v = 7.0$

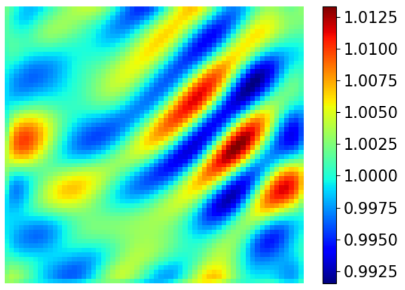
To show how the results for this situation differs with respect to the kernel size, the results at time $t = 20$ are shown below. In Figure 31a and Figure 31c the results computed with the kernel size $\sigma = 1.2$ is shown for the prey and predator population. In Figure 31b and Figure 31d the results computed with the kernel size $\sigma = 1.1$ is shown for the prey and predator population.



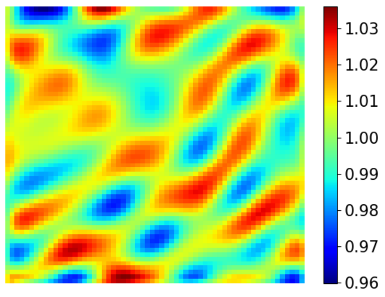
(a) Distribution of prey at time $t = 20$ for the model (63a)-(63b) with $\gamma_u = 29.0$, $\gamma_v = 7.0$ and $\sigma = 1.2$.



(b) Distribution of prey at time $t = 20$ for the model (63a)-(63b) with $\gamma_u = 29.0$, $\gamma_v = 7.0$ and $\sigma = 1.1$.



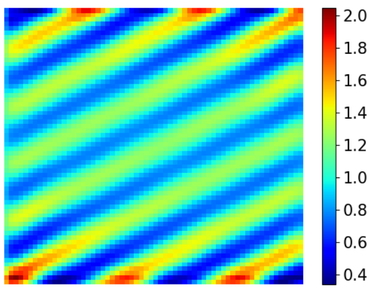
(c) Distribution of predator at time $t = 20$ for the model (63a)-(63b) with $\gamma_u = 29.0$, $\gamma_v = 7.0$ and $\sigma = 1.2$.



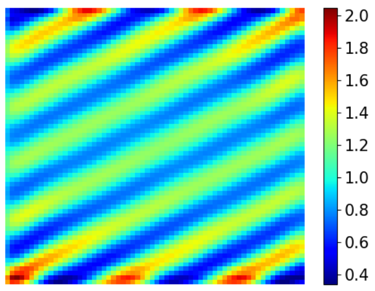
(d) Distribution of predator at time $t = 20$ for the model (63a)-(63b) with $\gamma_u = 29.0$, $\gamma_v = 7.0$ and $\sigma = 1.1$.

Figure 31: Distribution of prey (top) and predator (bottom) at time $t = 20$ for the model (63a)-(63b) with $\gamma_u = 29.0$, $\gamma_v = 7.0$ and $\sigma = 1.2$ and $\sigma = 1.1$, respectively

The picture here is almost the same for both situations. For the computations with the broad kernel the population at this time is thus located slightly closer to the constant equilibrium solution $(u^*, v^*) = (1, 1)$ than for the results computed with the smaller kernel. This small deviation is though enough to generate two completely different results in the end. At time $t = 100$ results computed with kernel size $\sigma = 1.2$ completely reverts to the equilibrium solution. On the other hand for the results computed with kernel size $\sigma = 1.1$ patterns emerge as seen below.



(a) Distribution of prey at time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 29.0$ and $\gamma_v = 7.0$ and $\sigma = 1.1$.



(b) Distribution of predator at time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 29.0$ and $\gamma_v = 7.0$ and $\sigma = 1.1$.

Figure 32: Distribution of prey and predatorat time $t = 100$ for the model (63a)-(63b) with $\gamma_u = 29.0$ and $\gamma_v = 7.0$ and $\sigma = 1.1$, respectively

For these results the width of the Gaussian kernel again showed to have major impact. For the simulations with $\sigma = 1.2$ no pattern is formed. Changing instead the width of the kernel to $\sigma = 1.1$ results on the other hand in the formation of patterns with the form of stripes, seen similarly for the results in Section 6.4. The spatio-temporal development for the simulations with $\sigma = 1.1$ for time $t = [1 : 50]$ are shown in Appendix E.2.

The results from Section 6.4 indicated a connection between the number of stripes in the formed patterns and the value of the advection parameters. A higher value of the advection parameters gave rise to a greater number of stripes. This is not the case for results in this specific scenario with $\gamma_u = 29.0$ and $\gamma_v = 7.0$ and $\sigma = 1.1$. The values for the advection parameters are here much higher than for both of the scenarios in Section 6.4, but the amount of stripes seen in Figure 32 are not higher than those seen in Figure 28. It is important to notice that the width of the kernel is not the same for these simulations. In this specific scenario the kernel is broader than for both scenarios in Section 6.4, which, as already seen, have a great affect on the final result.

When the simulations in Section 6.4.1 are computed with the same kernel size as for this scenario, i.e. with $\sigma = 1.1$, the system will for both scenarios be equal to the equilibrium solution. This means that results across the simulations regarding CASE 2 and the overlapping area CASE 2+CASE 1 can not be compared directly. Crossing the line between the two areas regarding CASE 2 and CASE 2+CASE 1 going from left to right, see the bifurcation diagram in Figure 25, requires a broader kernel. Even for values of the advection parameters close to the boarder between the two areas referred, but within the area for `CASE 2`, the same problem appears. Unfortunately this means that changes in the patterns across these two areas can not be compared directly, since the kernel will affect the emerged patterns.

6.7 Summary

With the implementation of the integration kernel the new model (63a)-(63b) is able to compute results for a greater variety of the two taxis parameters. For none of the results for situations with parameters values outside of the area regarding `CASE 3` are the hexagonal arrangement of spots seen in any of the formed patterns. This is in contrast to what is seen for most of the other simulations within this region. Instead, all patterns emerging for scenarios regarding conditions for `CASE 1` and `CASE 2` shows stripes.

The difference between these striped-formed patterns for the various simulations are also seen. The results indicates that the amount of taxis affects both the width, the number and height of the stripes. Furthermore it is seen that not only do taxis affects the patterns, but the width of the kernel also plays a role.

When the taxis parameters are increased this also requires a broader kernel. This combination causes the system, for all simulations regarding conditions for `CASE 1`, to simply end in the equilibrium solution. The broad kernel is thus capable of damping all instabilities in the system. This behavior of the kernel is predicted from the stability analysis of the model with the integration kernel, briefly described in Section 5.3. The effect is particularly visible for the test cases in Section 6.3 for which the system is solved with and without the integration kernel, and with different kernel sizes.

Computations with a smaller grid size have been tried in order to compute smoother solutions and so to be able to represent the kernel on a finer grid. These computations turned out to be too cumbersome, i.e. no results were generated within a reasonable amount of time. Due to the design of the algorithm, only specific grid sizes are aloud, which is explained in Section 3.3. As consequence, the next accepted value of grid points in each spatial direction increases from 64 to 128 points. This quadruples the number of grid points for the full domain which thereby increases the computational time to an unfavorable long time.

A redesign of the algorithm would be required in order to choose any desired amount of grid points for the full domain. This is due to the fact that several of the matrices build doing computations are reused in the multigrid solver as a consequence of the specific number of grid points. Changing this would require the algorithm to built new matrices when the system is transferred back and fourth between a finer and coarser grid and so this would increase the computational time as well.

7 Conclusion

The aim of this study was to apply the well studied theory behind Turing patterns to the fitness taxis model in order to investigate conditions for the spatio-temporal pattern formation. This resulted in specific conditions categorized into three different cases, `CASE 1`, `CASE 2` and `CASE 3` for which the full system would be unstable to small spatial perturbations.

For small values of the two advection parameters it was possible to meet the conditions in `CASE 3`. For scenarios within this area the formed patterns showed hexagonal arrangement of spots. Increasing the values of the advection parameters turned out to require a reformulation of the fitness taxis model (16a)-(16b), since instabilities in the solution grew to infinity and no results could be computed.

Reformulating the fitness taxis model by implementing the integration kernel turned out to provide solutions with a higher amount of taxis. For values of the advection parameters for which the system satisfies conditions regarding `CASE 2` the resulting pattern showed the formation of stripes. These stripes varied in number, density and width depending on the values of the advection parameters, but all stripes turn out to be located in the same direction. This direction of the patterns, i.e. stripes and spots, was also tested by changing the initial conditions, which all gave rise to the same results.

The change of the form of the stationary patterns from simulations regarding conditions in `CASE 3` and conditions in `CASE 2` showed how the taxis term affected the formation of spots into the formation of stripes. Movement due to a higher amount of taxis thus showed the formation of stripes in the stationary patterns, and showed that these stationary states will be set later.

The values of the two advection parameters turned out not to be the only parameters adjusted in this study that affected the pattern formation. Results computed with the integration kernel implemented showed that the width of the kernel also played a big role in the formation of patterns. As the kernel size became broader, the final patterns became less evident. On behalf of this the smallest possible kernel size giving rise to results is thus desired. No patterns were able to form within the area of `CASE1`, since this required such a broad kernel that all instabilities were damped.

For the results computed in `COMSOL` no stripes were seen for the same scenarios where stripes appeared in the results from Julia. This indicates that there might be some numerical effect in the results from Julia. The stripes might thereby not solely be a consequence of adding taxis to the model, but also some numerical effect in the algorithm playing a greater role when taxis is added.

Good numerical tools were also required in order to compute results within a reasonable amount of time when simulation for long periods of time. All computations with the integration kernel were very time-consuming. This has been a huge disadvantages when using the cloud-based Juliabox version that are run in the web browser. All results regarding simulations with the integration kernel has thus been run locally since connection to the cloud has either been lost due to the complexity and size of the model problem or terminated due to the amount of permitted computational time.

Many parameters in the model can be adjusted which makes it a bit difficult to get an overview of the simulations, and to get some hands on results. The parameters can be changed in such a variety of ways that it is very difficult to control and understand the emergence of a specific pattern. This regards for instance how to control the density and size of spots and stripes and how to control how these are distributed in space. Most results have been computed on the basis of some trial and error approach, and it has been difficult to know what to expect and what to look for.

Biology is complicated and messy, and lack of experimental work and data makes it difficult to connect the theory with actual real-life scenarios. No real conclusion regarding validation of the results can be drawn for this study since neither a true solution to the fitness taxis model nor real life data to underline the obtained results are available.

8 Further Work

For this study there is a numerous field to dive further into. First of all it is seen that computations between `COMSOL` and Julia showed deviation in teh results when an increasing amount of movement due to taxis is present in the system. An approach in order to investigate this deviation further could thus be to implement the kernel in `COMSOL` as well. There is no evidence that results from `COMSOL` is more correct, but big deviation between computations with the integration kernel in `COMSOL` and the results seen from computations in Julia would indicate that further investigation and validation of the model is desired.

All in all digging into the Julia code in order to improve the numerical tools used to compute results for the fitness taxis model would be a field of interest. It could be of great interest to make the computations faster. This could regard different discretization in space and different time stepping methods for instance. Making it less time consuming to produce results would make it easier to investigate patterns for a variety of advection values.

Finally trying to show pattern formation within the area for `CASE 1` seen in the bifurcation diagram in Figure 25 is desired. These results are lacking for this study and ways to show how these patterns would look like is also a field of interest. Other formulations of the population dynamics could be implemented, for instance regarding systems with more than one equilibrium point for positive values of the two species. The effect of such different populations dynamics could be investigated in order to see how this would change the pattern formation when the system is perturbed.

References

[1] Ian George Bolton.
Big fish eat little fish.
2015.
URL: http://www.grammar.zone/big-fish-eat-little-fish/ (visited on 10/15/17).

[2] Irene Louise Torpe Heilmann.
"Spatio-temporal Pattern formation in Predator-Prey Systems with Fitness Taxis".
Paper in preparation.
n. y.

[3] Alan M. Turing.
"The Chemical Basis of Morphogenesis".
In: *Philosophical Transactions of the Royal Society of London* 237 (1952), pp. 37–72.

[4] J. D. Murray.
Mathematical Biology, II: Spatial Models and Biomedical Applications.
Third Edition.
Springer Science+Business Media, LLC, 2002.

[5] Lingfa Yang , Milos Dolnik , Anatol M. Zhabotinsky , Irving R. Epstein.
"Turing patterns beyond hexagons and stripes".
In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 16 (2006), pp. 1 –9.

[6] Allen R. Sanderson , Robert M. Kirby , Chris R. Johnson , Lingfa Yang.
"Advanced Reaction-Diffusion Models for Texture Synthesis".
In: *Journal of Graphics Tools* 11, No. 3 (2006), pp. 47–71.

[7] Christopher A. Kennedy , Mark H. Carpenter.
"Hypothesis for origin of planktonic patchiness".
In: *Nature* 259 (1976), p. 659.

[8] M. A Tsyganov , J. Brindley , A. V. Holden , V. N. Biktashev.
"Quasi-soliton interaction of pursuit-evasion waves in a predator-prey system".
In: *Physical review letters* 91 (2003), pp. 1–4.

[9] Willem Hundsdorfer , Jan G. Verwer.
Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations.
First Edition.
Springer-Verlag Berlin Heidelberg New York, 2003.

[10] Alexander D. Bazykin.
Nonlinear Dynamics Of Interacting Populations.
Vol. 11.
World Scientific Publishing Co. Pte. Ltd, 1998.

[11] A. D. Bazykin.
"Structural and Dynamic Stability of Model Predator-Prey Systems".
In: *IIASA Research Memorandum* (1976), pp. 1–38.

[12] James D. Meiss.
Differential Dynamical Systems.
First Edition.
Society for Industrial and Aaplied Mathematics, 2007.

[13] Malay Banerjee , S. Ghorai , Nayana Mukherjee.
"Approximated spiral and target patterns in Bazykin's prey-predator model: multiscale perturbation analysis".
In: *International Journal of Bifurcation and Chaos* (2017), pp. 1–15.

[14] Edward A. McGehee , Noel Schutt , Desiderio A. Vasquez , Enrique Peacock-López.
"Bifurcations, ans temporal and spatial patterns of a modified Lotka-Volterra Model".
In: *Bifurcation Chaos 18* 18 (2007), pp. 2223–2248.

[15] J. D. Murray.
Mathematical Biology, I: An Introduction.
Third Edition.
Springer Science+Business Media, LLC, 2002.

[16] Manuel Schmid.
Efficient solver for systems of reaction-diffusion equations.
Ecole Polytechnique Fédérale de Lausanne – CSE Semester Project.
2015.

[17] Christopher A. Kennedy , Mark H. Carpenter.
"Additive Runge–Kutta schemes for convection–diffusion–reaction equations".
In: *1-2* 44 (2003), pp. 139–181.

[18] Jeff Bezaonson , Alan Edelman , Stefan Karpinski , Viral B. Shah.
"Julia: A fresh approach to numerical computing".
In: *arXiv preprint arXiv:1411.1607v4* (2015), pp. 1–37.

[19] Jeff Bezaonson , Alan Edelman , Stefan Karpinski , Viral B. Shah.
"Julia: A Fast Dynamic Language for Technical Computing".
In: *arXiv preprint arXiv:1209.5145v1* (2012), pp. 1–27.

A Appendix

B Finite Difference Approximations

The following is the fourth-order centered finite difference approximation matrices for the Nabla- and Laplace-operator respectively. They assume periodic boundaries.

B.1 4th order Centered ∇-operator Scheme

The full code for this implementation is listed in Appendix F.3.

$$\nabla f_{ij} \approx \frac{f_{i-2,j} - 8f_{i-1,j} + 8f_{i+1,j} - f_{i+2,j}}{12\Delta x} + \frac{f_{i,j-2} - 8f_{i,j-1} + 8f_{i,j+1} - f_{i,j+2}}{12\Delta y} \tag{B.1}$$

$$\nabla u \approx \mathrm{FD}_A \cdot u = (\mathrm{FD}_{Ax} + \mathrm{FD}_{Ay}) \cdot u$$
$$\nabla v \approx \mathrm{FD}_A \cdot v = (\mathrm{FD}_{Ax} + \mathrm{FD}_{Ay}) \cdot v$$

$$\mathrm{FD}_{Ax} = \frac{1}{12\Delta x}\begin{bmatrix} \mathbf{A}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{A}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \mathbf{A}_{N_y} \end{bmatrix} \tag{B.2}$$

$$\mathbf{A}_i = \begin{bmatrix} 0 & 8 & -1 & 0 & \cdots & 0 & 1 & -8 \\ -8 & 0 & 8 & -1 & \ddots & \ddots & \ddots & 1 \\ 1 & -8 & 0 & 8 & -1 & \ddots & \ddots & 0 \\ 0 & 1 & -8 & 0 & 8 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & 1 & -8 & 0 & 8 & -1 \\ -1 & 0 & \ddots & \ddots & 1 & -8 & 0 & 8 \\ 8 & -1 & 0 & \cdots & 0 & 1 & -8 & 0 \end{bmatrix}_{N_x \times N_x} \tag{B.3}$$

$$
\mathrm{FD}_{Ay} = \frac{1}{12\Delta y}\begin{bmatrix}
0 & \cdots & 8 & \cdots & -1 & \cdots & \cdots & -1 & \cdots & 8 & \cdots & 0 \\
\vdots & 0 & \ddots & 8 & \ddots & -1 & \ddots & \ddots & -1 & \ddots & \ddots & \vdots \\
-8 & \ddots & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 8 \\
\vdots & -8 & \ddots & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
1 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & -1 \\
\vdots & 1 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
1 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & -1 \\
\vdots & 1 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
-8 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 8 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & -8 & \cdots & 1 & \cdots & \cdots & 1 & \cdots & -8 & \cdots & 0
\end{bmatrix}_{N_xN_y\times N_xN_y} \tag{B.4}
$$

B.2 4th order ∇^2-operator Centered Scheme

The matrices for the fourth-order centered finite difference approximation of the ∇^2-operator are built according to the same principle used for building the previous finite difference matrices, see Appendix B.1. The full matrices are thus omitted here, but their generation can be seen in the code in Appendix F.3.

$$
\begin{aligned}
\nabla^2 f_{ij} \approx\ & \frac{-f_{i-2,j} + 16f_{i-1,j} - 30f_{i,j} + 16f_{i+1,j} - f_{i+2,j}}{\Delta x^2} \\
+\ & \frac{-f_{i,j-2} + 16f_{i,j-1} - 30f_{i,j} + 16f_{i,j+1} - f_{i,j+2}}{\Delta y^2}
\end{aligned} \tag{B.5}
$$

$$
\begin{aligned}
\nabla^2 u &\approx \mathrm{FD}_D \cdot u = (\mathrm{FD}_{Dx} + \mathrm{FD}_{Dy}) \cdot u \\
\nabla^2 v &\approx \mathrm{FD}_D \cdot v = (\mathrm{FD}_{Dx} + \mathrm{FD}_{Dy}) \cdot v
\end{aligned}
$$

C The Butcher Tableau

The coefficient are from [17].

C.1 ARK(3)4L[2]SA-ERK

$$
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
\frac{1767732205903}{2027836641118} & \frac{1767732205903}{4055673282236} & 0 & 0 & 0 \\
\frac{3}{5} & \frac{5535828885825}{10492691773637} & \frac{788022342437}{10882634858940} & 0 & 0 \\
1 & \frac{6485989280629}{16251701735622} & \frac{-4246266847089}{9704473918619} & \frac{10755448449292}{10357097424841} & 0 \\
\hline
b_i & \frac{1471266399579}{7840856788654} & \frac{-4482444167858}{7529755066697} & \frac{11266239266428}{11593286722821} & \frac{1767732205903}{4055673282236} \\
\hline
\hat{b}_i & \frac{2756255671327}{12835298489170} & \frac{-10771552573575}{22201958757719} & \frac{9247589265047}{10645013368117} & \frac{2193209047091}{5459859503100}
\end{array}
$$

C.2 ARK(3)4L[2]SA-ESDIRK

$$
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
\frac{1767732205903}{2027836641118} & \frac{1767732205903}{4055673282236} & \frac{1767732205903}{4055673282236} & 0 & 0 \\
\frac{3}{5} & \frac{2746238789719}{10658868560708} & \frac{-640167445237}{6845629431997} & \frac{1767732205903}{4055673282236} & 0 \\
1 & \frac{1471266399579}{7840856788654} & \frac{-4482444167858}{7529755066697} & \frac{11266239266428}{11593286722821} & \frac{1767732205903}{4055673282236} \\
\hline
b_i & \frac{1471266399579}{7840856788654} & \frac{-4482444167858}{7529755066697} & \frac{11266239266428}{11593286722821} & \frac{1767732205903}{4055673282236} \\
\hline
\hat{b}_i & \frac{2756255671327}{12835298489170} & \frac{-10771552573575}{22201958757719} & \frac{9247589265047}{10645013368117} & \frac{2193209047091}{5459859503100}
\end{array}
$$

D Multigrid Solver

```
function MULTIGRID_ITERATION($\mathbf{A}_h, \mathbf{b}_h$)
    for $i \leftarrow 1$ to $\nu_1$ do
        $\mathbf{x} \leftarrow \left(\mathbf{I} - \omega\mathbf{D}^{-1}\mathbf{A}\right)\mathbf{x} + \omega\mathbf{D}^{-1}\mathbf{b}$
    end for
    $\mathbf{r}_h \leftarrow \mathbf{b}_h - \mathbf{A}_h\mathbf{x}$
    $\mathbf{r}_H \leftarrow$ RESTRICTION($\mathbf{r}_h$)
    $\epsilon_H \leftarrow$ MULTIGRID_ITERATION($\mathbf{A}_H, \mathbf{r}_H$)
    $\epsilon_h \leftarrow$ PROLONGATION($\epsilon_H$)
    $\mathbf{x} \leftarrow \mathbf{x} + \epsilon_h$
    for $i \leftarrow 1$ to $\nu_2$ do
        $\mathbf{x} \leftarrow \left(\mathbf{I} - \omega\mathbf{D}^{-1}\mathbf{A}\right)\mathbf{x} + \omega\mathbf{D}^{-1}\mathbf{b}$
    end for
    return $\mathbf{x}$
end function
```

E Simulations

E.1 Spatio-temporal development without Integration Kernel

Visualization of early spatio-temporal development. From Figure 33 to Figure 35 this is shown for the prey in the taxis free scenario from Section 4.2 together with the three different scenarios from Section 4.3.2, Section 4.3.2 and Section 4.3.3 respectively. In Figure 36 to Figure 38 the exact same is shown for the predator for the same scenarios.

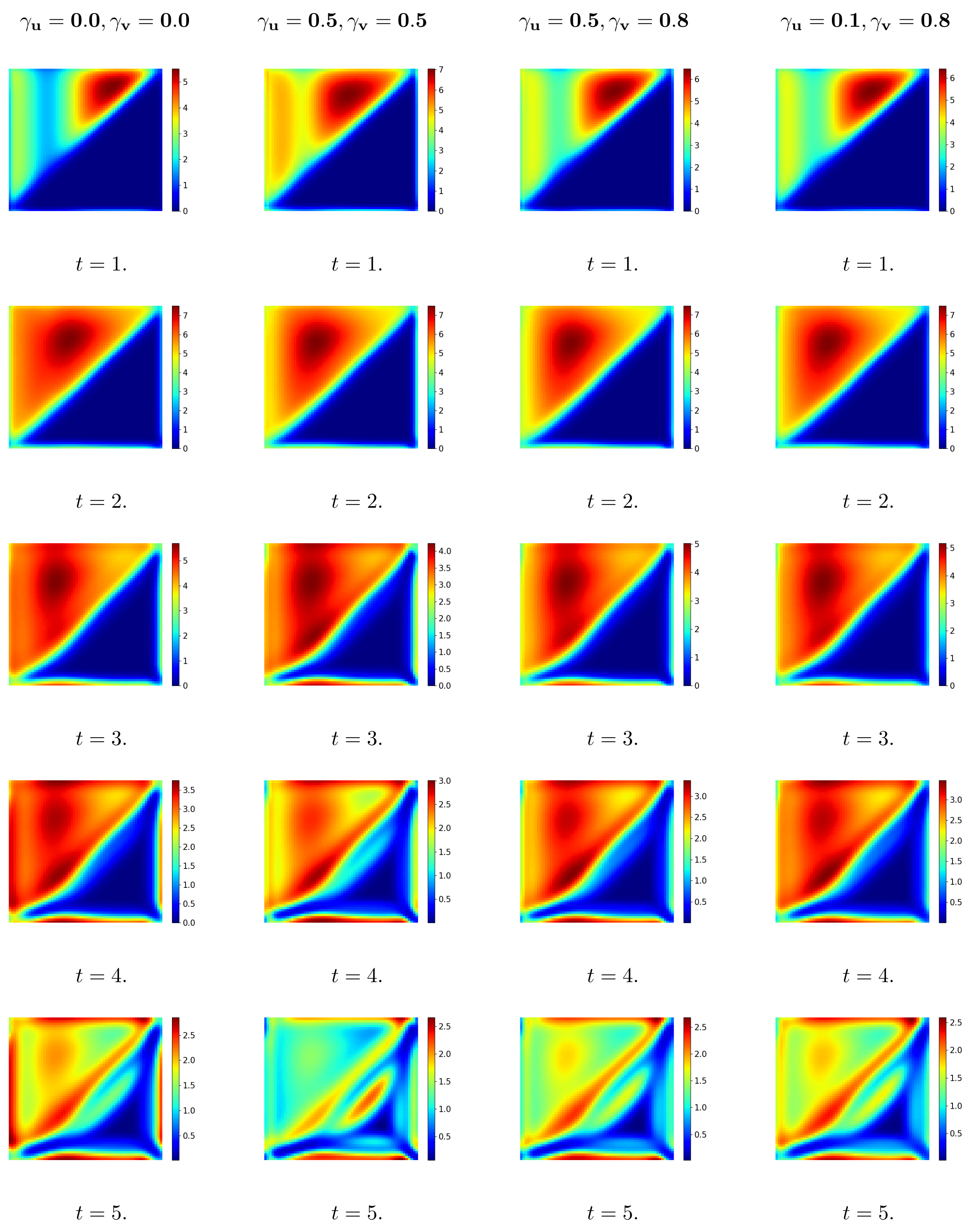


Figure 33: Spatio-temporal development of prey population for time $t = [1 : 5]$

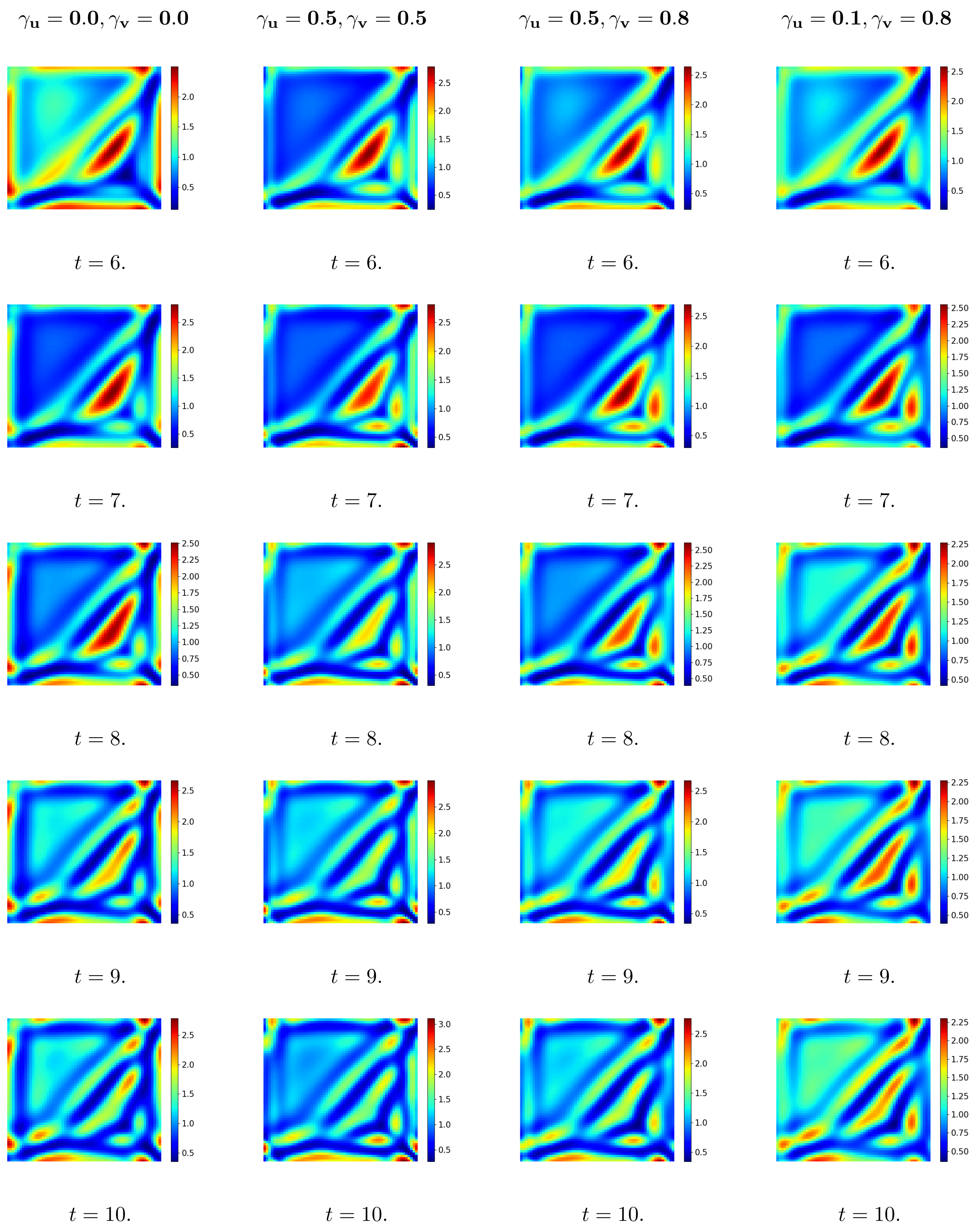


Figure 34: Spatio-temporal development of prey population for time $t = [6 : 10]$

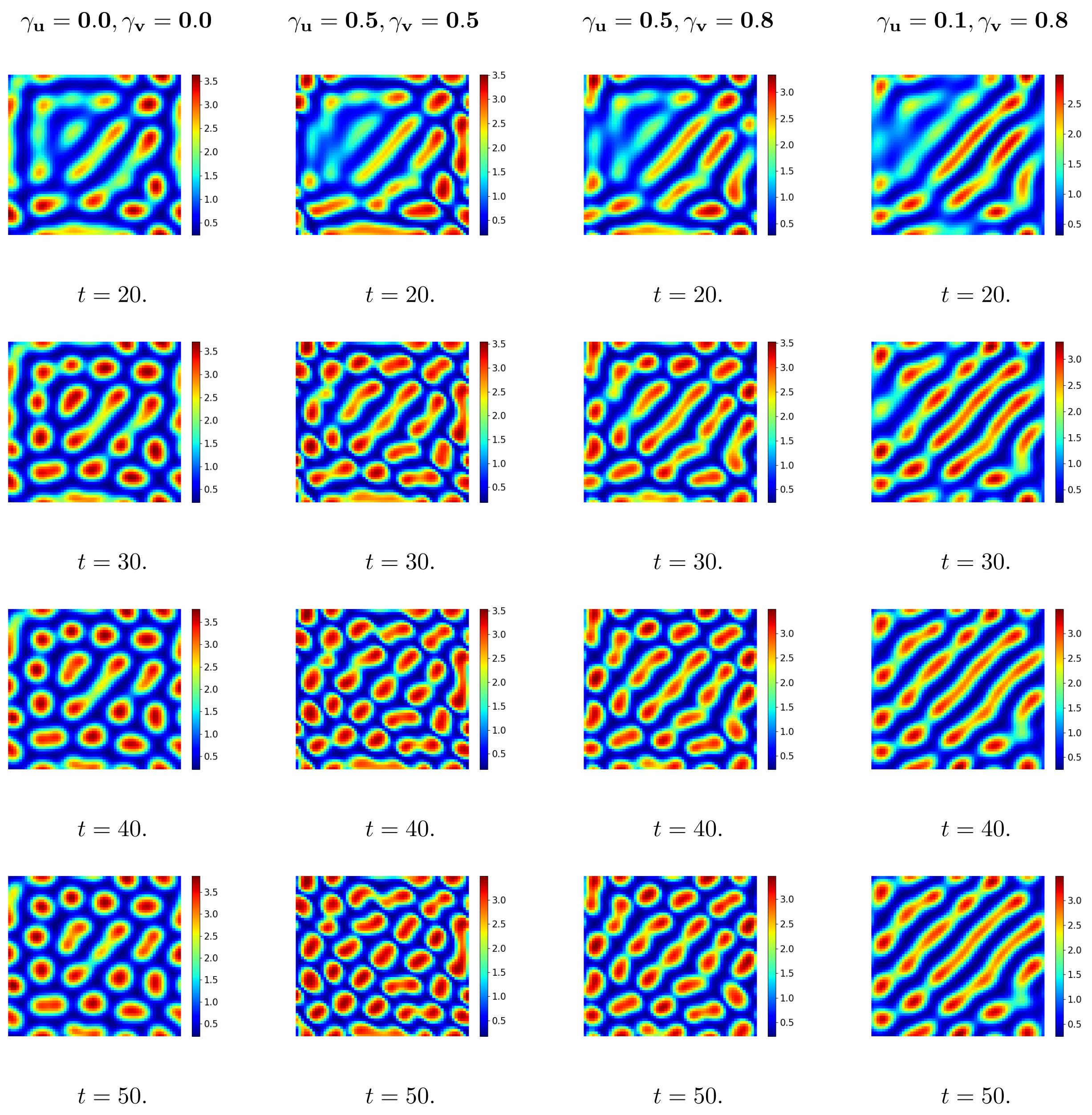


Figure 35: Spatio-temporal development of prey population for time $t = [20, 30, 40, 50]$

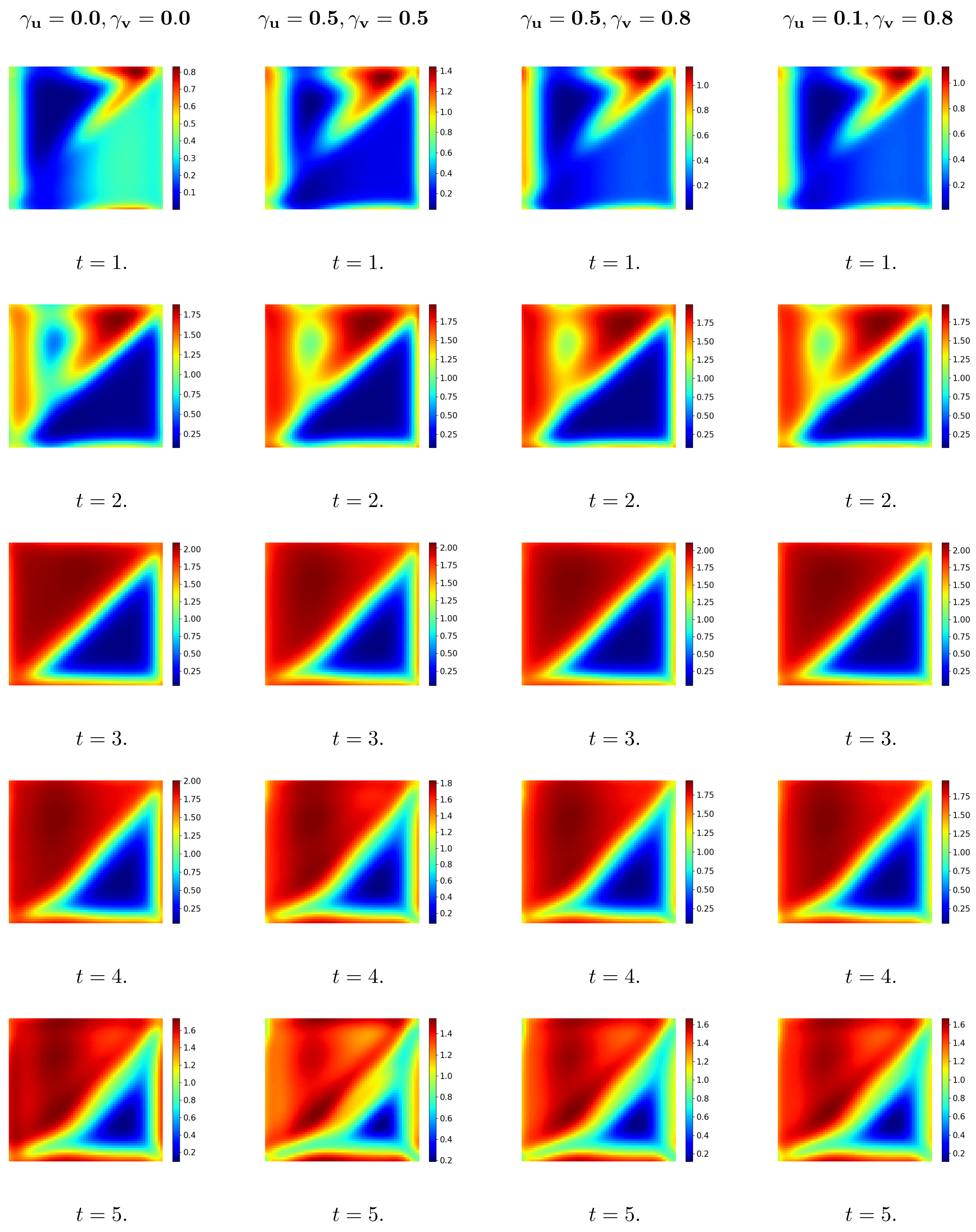


Figure 36: Spatio-temporal development of predator population for time $t = [1 : 5]$

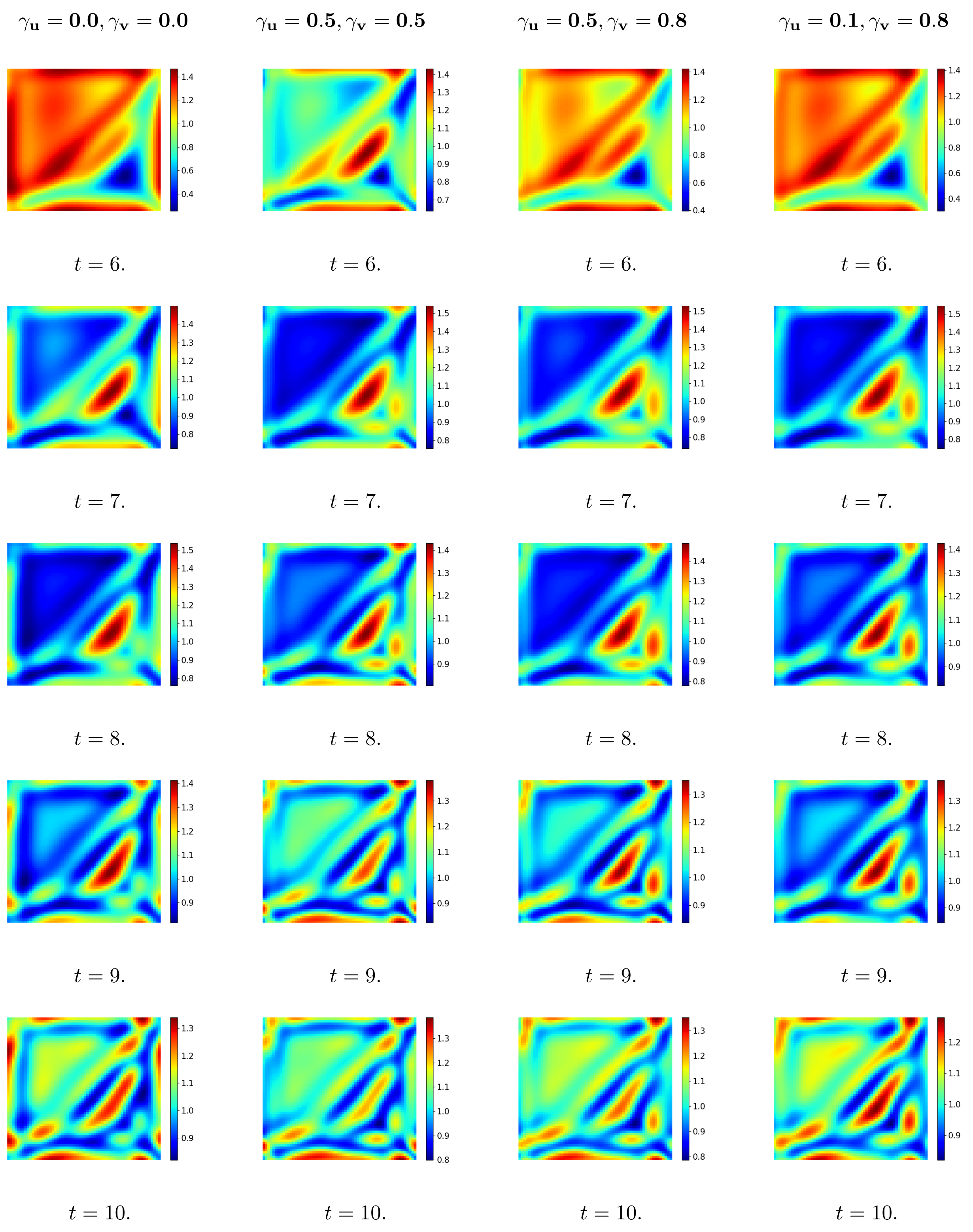


Figure 37: Spatio-temporal development of predator population for time $t = [6 : 10]$

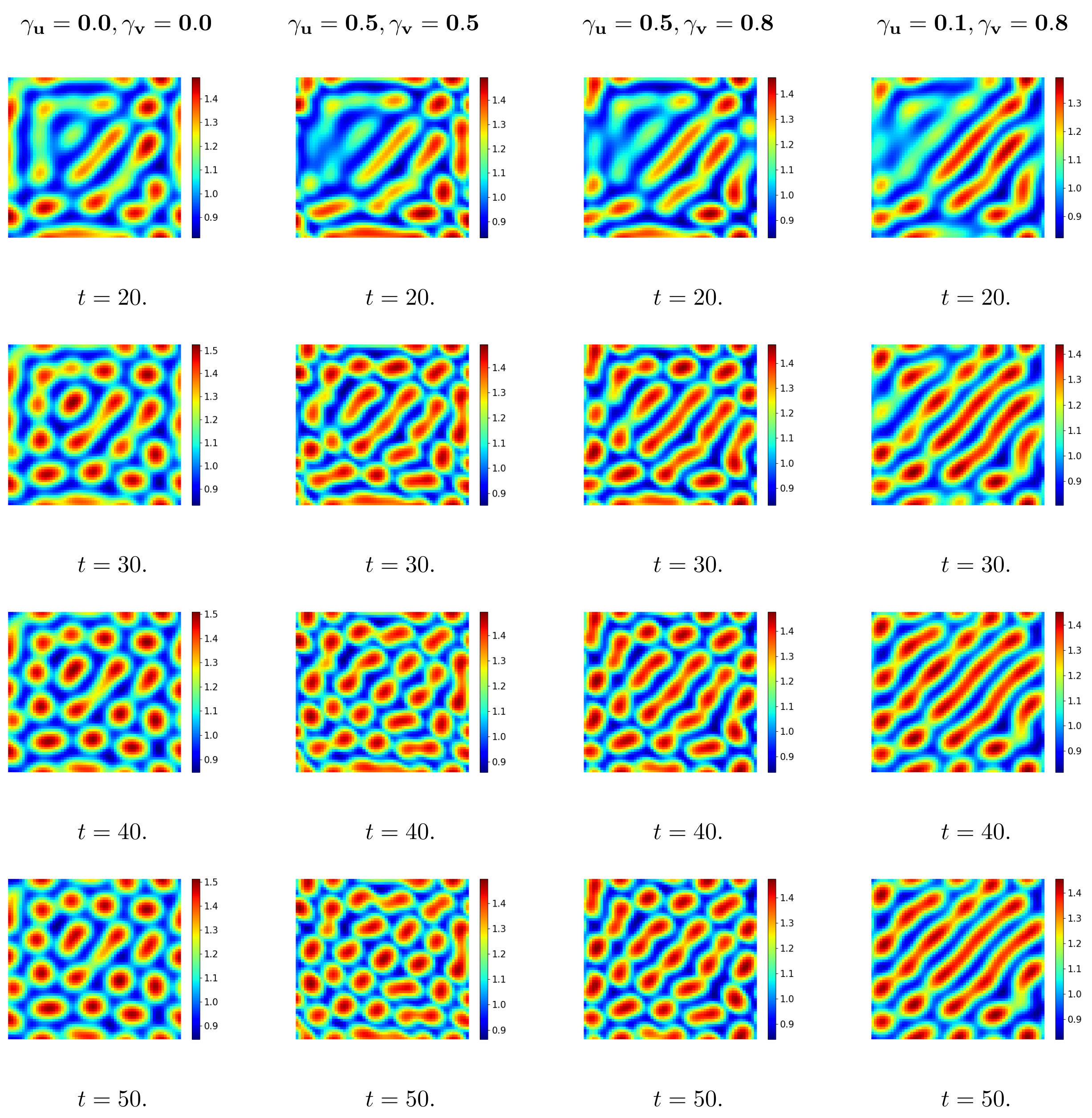


Figure 38: Spatio-temporal development of predator population for time $t = [20, 30, 40, 50]$

E.2 Spatio-temporal development whit Integration Kernel

Simulations with the integration kernel that gave rise to patterns are shown here as the spatio-temporal development from $t = [1 : 50]$. Four scenarios are shown where the first is the results from Section 6.3 with $\gamma_u = 1.0, \gamma_v = 1.0$ and $\sigma = 0.5$. The second and third column shows results from the two different scenarios computed in Section 6.4 respectively. The values for these two are $\gamma_u = 5.0, \gamma_v = 5.0$ and $\sigma = 0.5$ and $\gamma_u = 10.0, \gamma_v = 5.0$ and $\sigma = 0.7$. The last of the four columns shows the results obtained in Section 6.6 for which $\gamma_u = 29.0, \gamma_v = 7.0$ and $\sigma = 1.1$. All these simulations are shown in Figure 39 to Figure 41 for the prey population and for the predator population in Figure 36 to Figure 38.

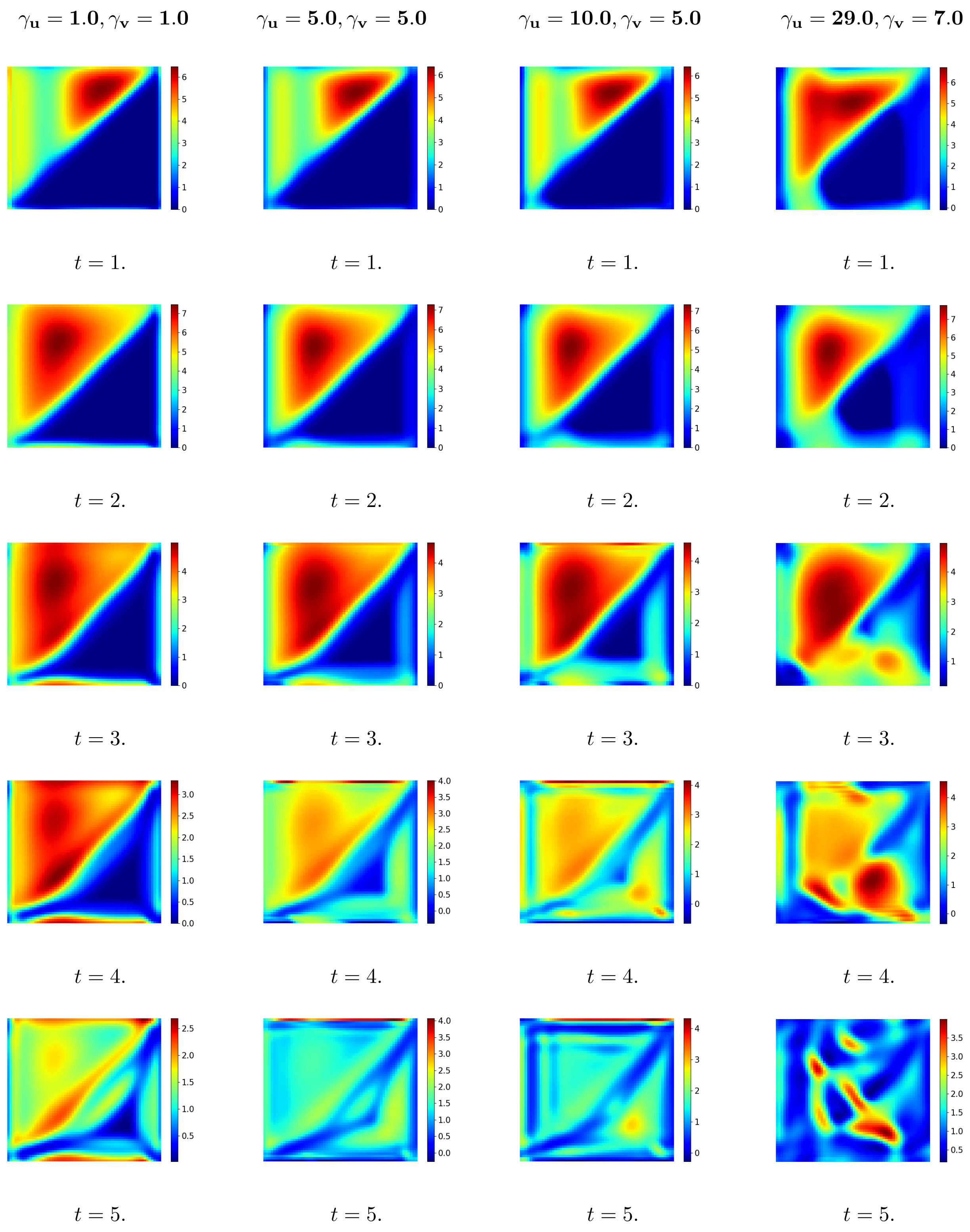


Figure 39: Spatio-temporal development of prey population for time $t = [1 : 5]$

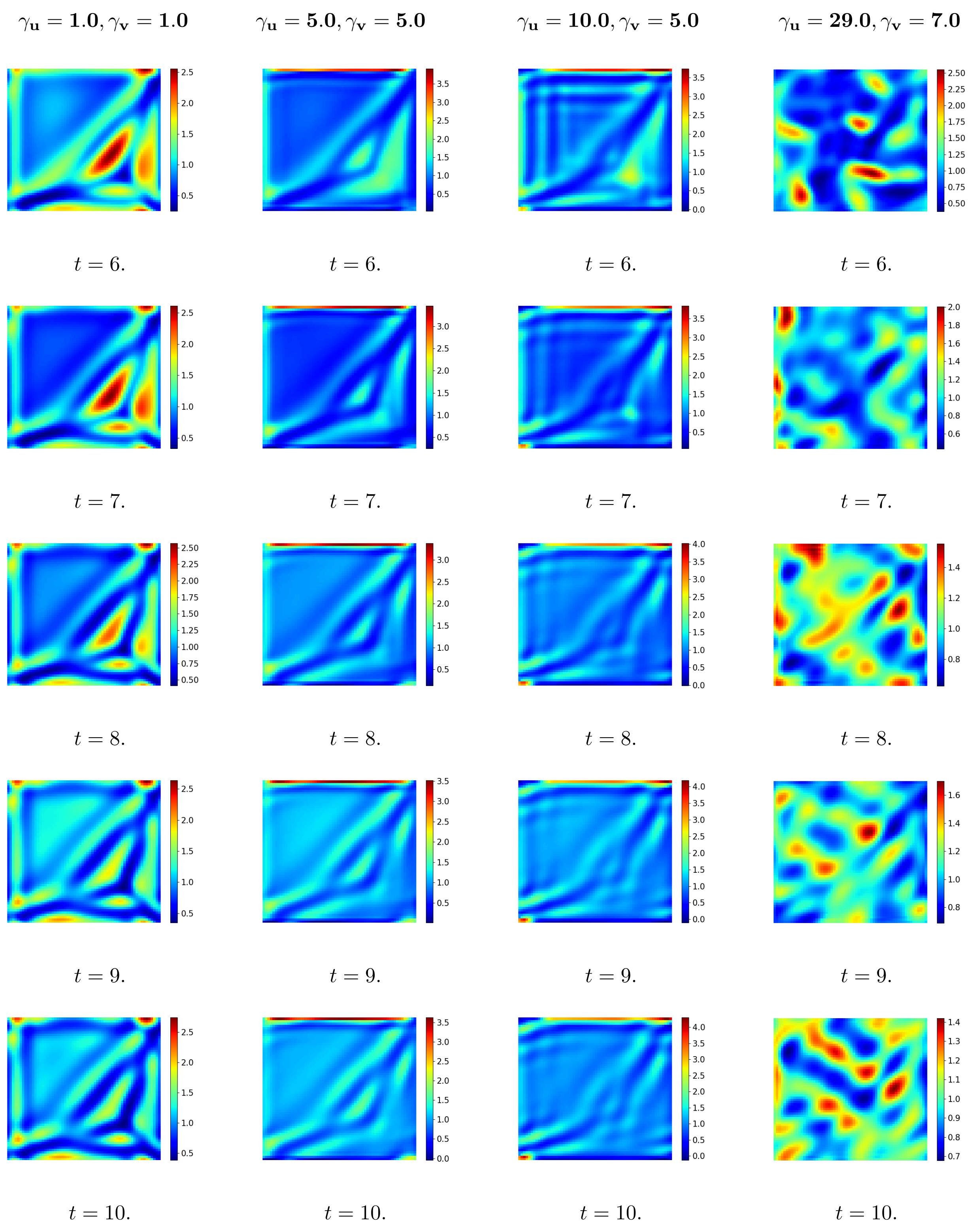


Figure 40: Spatio-temporal development of prey population for time $t = [6 : 10]$

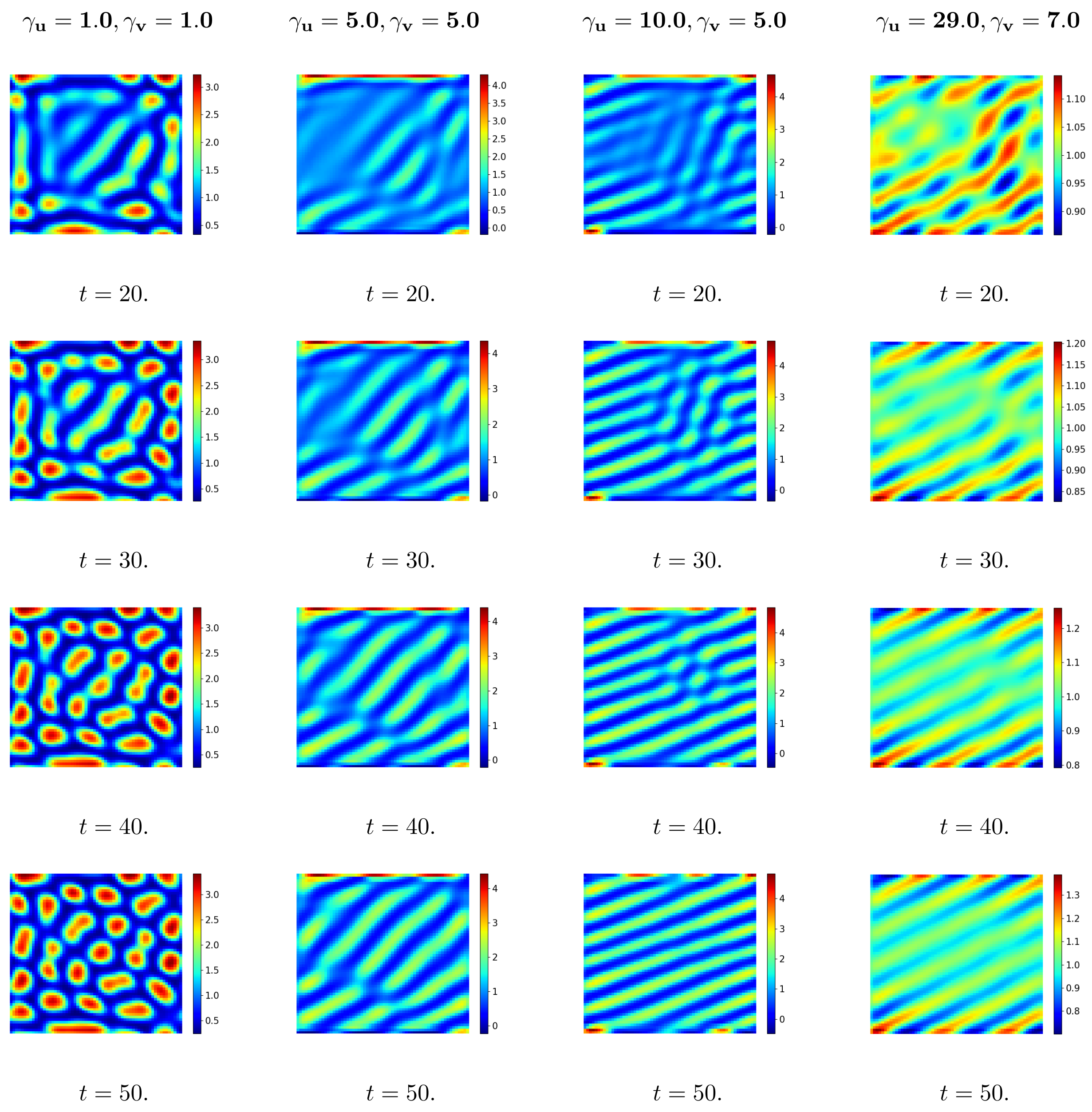


Figure 41: Spatio-temporal development of prey population for time $t = [20, 30, 40, 50]$

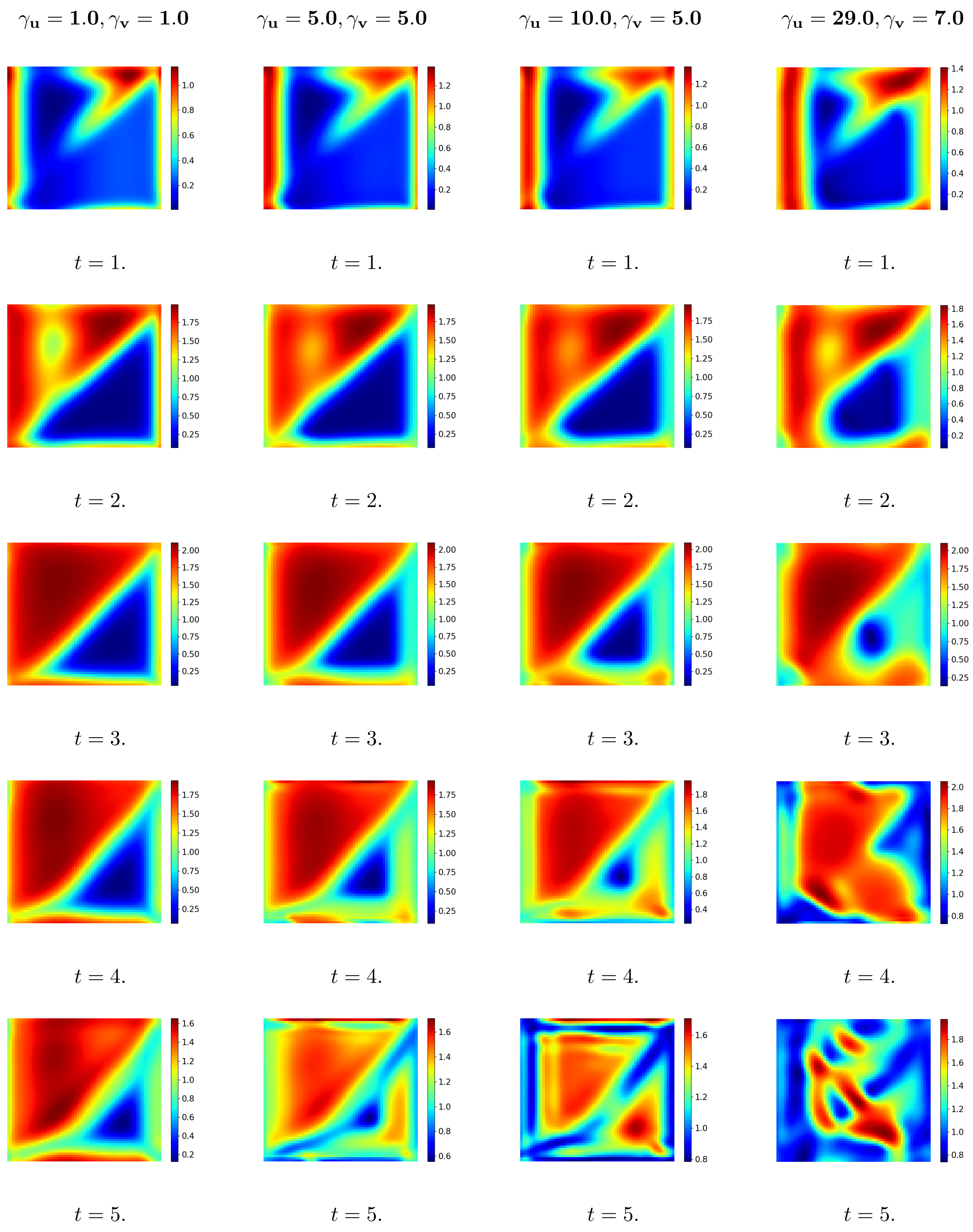


Figure 42: Spatio-temporal development of predator population for time $t = [1 : 5]$

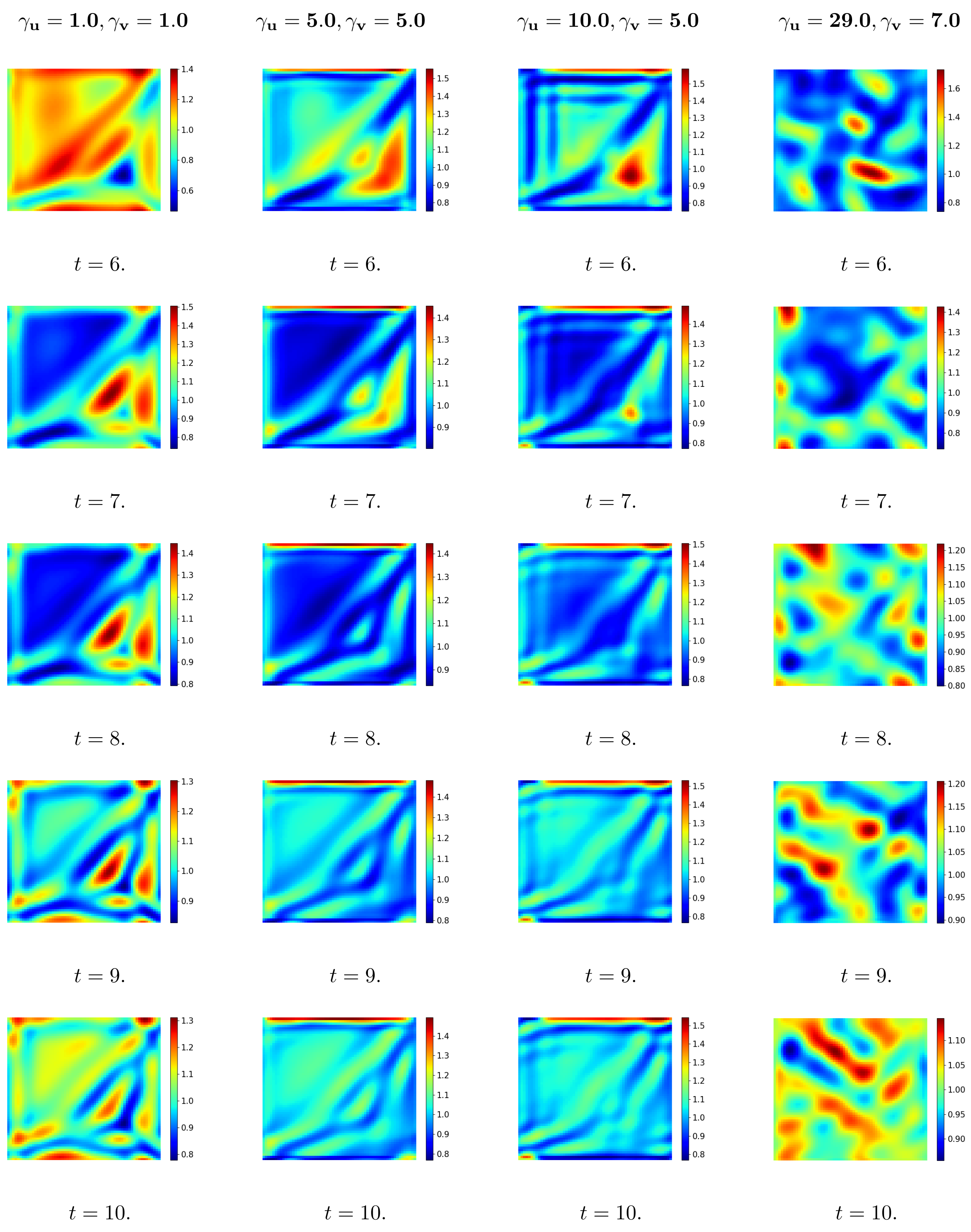


Figure 43: Spatio-temporal development of predator population for time $t = [6 : 10]$

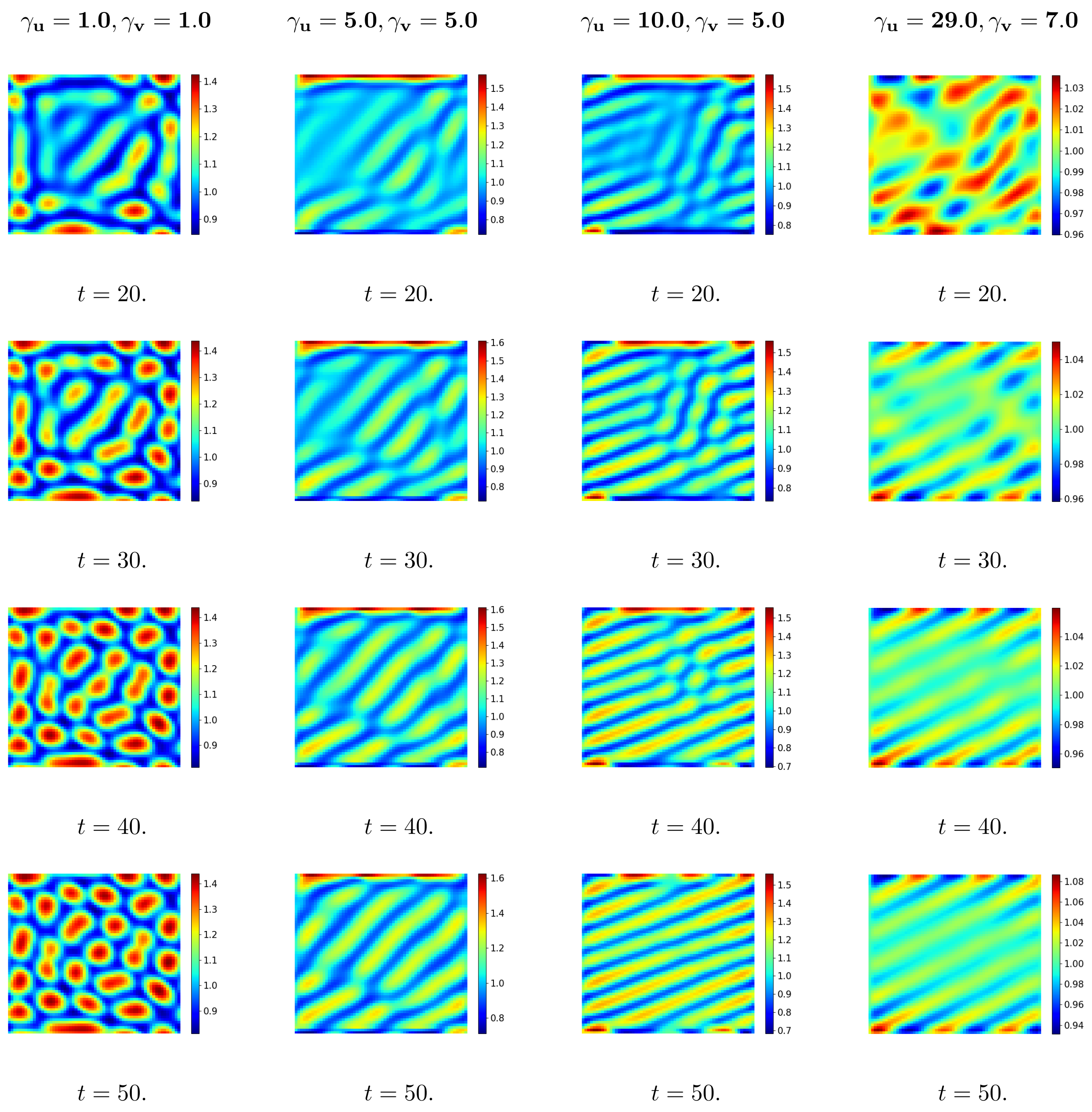


Figure 44: Spatio-temporal development of predator population for time $t = [20, 30, 40, 50]$

F Julia Code

All code from Julia used to compute solutions to the fitness taxis model is listed below.

F.1 Cases

This code is handed out by Irene, developed as a part of the article [2]. It has the task of defining which of the three cases, See Section 2.3.1, the model falls within, for some specified values of D_u, D_v, γ_u and γ_v. It is here implemented in Julia.

```
function CheckCases(A, Dif, Tax)
# Determine if any of the three cases for pattern formation occurs.
# Input is 'A', a 2x2 matrix with linearized dynamics of local model,
# the diffusion coefficients 'Dif' and the taxis coefficients 'Tax'.
# Output is a logcial vector 'C' indicating which cases/conditions are
# met, the critical wave numbers squared 'k2', the corresponding
# waveliengths 'L' and values 'z' for the bifurcation lines.

# Diffusion coefficients
Du = Dif[1];
Dv = Dif[2];
# Taxis coefficients
Tu = Tax[1];
Tv = Tax[2];

# Determinant
A_det = det(A);

C = zeros(Bool,5)
z = zeros(7,1)
z = complex(z)
k2 = zeros(3,1)
k2 = complex(k2)
# The conditions for the different cases:
# Case 1
C[1] = ( Tu*A[1,1] + Tv*A[2,2] > Du + Dv );
# Case 2
C[2] = ( Dv*Tu*A[1,1] + Du*Tv*A[2,2] > Tu*Tv*A_det + Du*Dv );
```

```
# Case 3 has three condtions
C3a = ( Dv*A[1,1] + Du*A[2,2] > (Tu+Tv)*A_det );
C3b = ( (Tv-Tu) < (Dv*A[1,1]-Du*A[2,2] - 2*sqrt(-A[1,2]*A[2,1]*Du*Dv) )/A_det
);

# Case 3
C[3] = ( !C[2] ) & C3a & C3b  ;
# The separate conditions b and c for case 3
C[4] = C3a;
C[5] = C3b;

# Values for drawing bifurcation diagram: z1,z2,z3,z4,z5
z[1] = (Du + Dv)/A[1,1];
z[2] = Du/A[1,1];
z[3] = (A[1,1]*Dv + A[2,2]*Du)/A_det;
z[4] = A[1,1]*Dv/A_det;
z[5] = (A[1,1]*Dv - A[2,2]*Du - 2*sqrt(-A[1,2]*A[2,1]*Du*Dv))/A_det;
z[6] = A[1,2]*A[2,1]*Du*Dv / A_det^2;
z[7] = A[2,2]*Du/A_det;

# Wave numbers, squared:
# Definition of value S and R, see notes.
S = A_det*(Tu+Tv) - A[1,1]*Dv - A[2,2]*Du;
# The determinant of matrix (T-D)
TmD_det =  A_det*Tu*Tv - A[1,1]*Dv*Tu - A[2,2]*Du*Tv + Du*Dv;
R = S^2 - 4*A_det*TmD_det;
# Critical wave number for case 1: k_c^2
k2[1] = - ( A[1,1] + A[2,2] )/( A[1,1]*Tu + A[2,2]*Tv - Du - Dv);
# The critical wave numbers for case 2 and 3
k2[2] = (-S - sqrt(complex(R)))/(2*TmD_det);
k2[3] = (-S + sqrt(complex(R)))/(2*TmD_det);

# Wave lengths, corresponding to wave numbers.
L = 2*pi./sqrt(k2);

# Eigenvalues as function of wavenumber squared

# Trace function p(k^2)
p(k2) = A[1,1]+A[2,2] + k2*( A[1,1]*Tu+A[2,2]*Tv - Du - Dv);
```

```
# Determinant function
q(k2) = TmD_det*k2.^2 + S*k2 + A_det;

Eig1_fun(k2) = 0.5*p(k2) +  0.5*sqrt(p(k2).^2 - 4*q(k2));
Eig2_fun(k2) = 0.5*p(k2) -  0.5*sqrt(p(k2).^2 - 4*q(k2));

    return C', z, L, k2, Eig1_fun, Eig2_fun
end
```

F.2 Draw

This code is handed out by Irene, developed as a part of the article [2]. This code is used to generate the bifurcation plot that arises for some specified values of D_u, D_v, γ_u and γ_v. It is here implemented in Julia.

```
function Draw(A, Dif)
# Diffusion coefficients
    Du = Dif[1];
    Dv = Dif[2];
    # Linearisation matrix A
    a11 = A[1,1]; a12 = A[1,2];
    a21 = A[2,1]; a22 = A[2,2];
    A_det = det(A); # Determinant
z = zeros(8,1);
z = complex(z);
# Values for drawing bifurcation diagram: z1,z2,z3,z4,z5
z[1] = (Du + Dv)/a11;
z[2] = Du/a11;
z[3] = (a11*Dv + a22*Du)/A_det;
z[4] = a11*Dv/A_det;
z[5] = (a11*Dv - a22*Du - 2*sqrt(complex(-a12*a21*Du*Dv)))/A_det;
z[6] = a12*a21*Du*Dv / A_det^2;
z[7] = a22*Du/A_det;
z[8] = -a11/a22;
# Calcute points on lines
x = linspace(0,50, 400); # gamma_u
# Condition 1
a = z[8];    # Slope
```

```
b = (Du + Dv)/a22; # Intercept   = - z(1)*a
# Line
y1 = a*x + b;
# Condition 2
a = z[7];
b = z[4];
c = z[6];
# Line
y2 = b + c./(x-a);
# Condition 3a
y3a = z[3] - x;
# Condition 3b
y3b = z[5] + x;
rc("xtick", labelsize=10)
rc("ytick", labelsize=10)
plot(x, y1)
plot(x, y2)
plot(x, y3a)
plot(x, y3b)
ax = gca()
ax[:set_ylim]([0,20])
ax[:set_xlim]([0,50])
xlabel("\gamma_u")
ylabel("\gamma_v")
end
```

F.3 Finite Difference matrices

This code generates the fourth order centered finite difference scheme for the Laplace operator in two dimensions. This function is unmodified and taken from [16].

```
function fdmatrix4(grid::Grid2D; EBC::Bool=false)

    # number of unknowns in x and y direction
    ux = (grid.Nx-1)
    uy = (grid.Ny-1)

    FDx_local = spdiagm((-ones(ux-2), 16*ones(ux-1),
    -30*ones(ux), 16*ones(ux-1), -ones(ux-2)),(-2,-1,0,1,2))
    if EBC
        FDx_local[1,1:4] = [-20,6,4,-1]
        FDx_local[end,end-3:end] = [-1,4,6,-20]
    else
        FDx_local[1,end-1:end] = [-1, 16]
        FDx_local[2,end] = -1
        FDx_local[end-1,1] = -1
        FDx_local[end,1:2] = [16,-1]
    end
    FDx = blkdiag(Any[FDx_local for i in 1:uy]...)

    if EBC
        FDy = spdiagm((-ones(ux*(uy-2)), 16*ones(ux*(uy-1)),
        -30*ones(ux*uy), 16*ones(ux*(uy-1)), -ones(ux*(uy-2))),
            (-2*ux,-ux,0,ux,2*ux))
        for i=1:ux
            FDy[i,[i,i+ux,i+2*ux,i+3*ux]] = [-20, 6, 4, -1]
            FDy[(end-ux)+i,(end-ux)+i-[3,2,1,0]*ux] = [-1, 4, 6, -20]
        end
    else
        FDy = spdiagm((16*ones(ux), -ones(2*ux),
            -ones(ux*(uy-2)), 16*ones(ux*(uy-1)), -30*ones(ux*uy),
            16*ones(ux*(uy-1)), -ones(ux*(uy-2)),
            -ones(2*ux), 16*ones(ux)),(-ux*(uy-1),-ux*(uy-2),-2*ux,-ux,
            0,ux,2*ux,ux*(uy-2),ux*(uy-1)))
    end
    # return combined matrix
    1/(12*grid.dx^2) * FDx + 1/(12*grid.dy^2) * FDy
end
```

This code generates the fourth order centered finite difference scheme for the Nabla operator in two dimensions.

```
function fdmatrix4UP(grid::Grid2D; EBC::Bool=false)
    # number of unknowns in x and y direction
    ux = (grid.Nx-1)
    uy = (grid.Ny-1)

    FDx_local = spdiagm((ones(ux-2), -8*ones(ux-1), zeros(ux),
     8*ones(ux-1), -ones(ux-2)),(-2,-1,0,1,2))
```

```
        FDx_local[1,end-1:end] = [1, -8]
        FDx_local[2,end] = 1
        FDx_local[end-1,1] = -1
        FDx_local[end,1:2] = [8,-1]
    FDx = blkdiag(Any[FDx_local for i in 1:uy]...)

    FDy = spdiagm((-8*ones(ux), ones(2*ux),
ones(ux*(uy-2)), -8*ones(ux*(uy-1)), zeros(ux*uy), 8*ones(ux*(uy-1)),
 -ones(ux*(uy-2)),
        -ones(2*ux), 8*ones(ux)),(-ux*(uy-1),-ux*(uy-2),-2*ux,-ux,0,
        ux,2*ux,ux*(uy-2),ux*(uy-1)))

    # return combined matrix
1/(12*grid.dx) * FDx + 1/(12*grid.dy) * FDy
end
```

F.4 Grid

This code generates the two dimensional grid. The code is unmodified and taken from [16].

```
immutable Grid2D

    x_start::Float64
    x_end::Float64
    y_start::Float64
    y_end::Float64
    Nx::Int
    Ny::Int
    dx::Float64
    dy::Float64
    levels::Int

    function Grid2D(x_start, x_end, y_start, y_end, Nx, Ny, levels)

        if Nx/2^(levels-1) != round(Int,Nx/2^(levels-1))
            error("ERROR: The number of intervals
            in x-direction is not divisible by 2^i")
        end
        if Ny/2^(levels-1) != round(Int,Ny/2^(levels-1))
            error("ERROR: The number of intervals
            in y-direction is not divisible by 2^i")
        end

        new(x_start, x_end, y_start, y_end, Nx, Ny,
        (x_end-x_start)/Nx, (y_end-y_start)/Ny, levels)
    end
end
```

F.5 Integration Kernel

This code generates the Gaussian function from Equation (50) and the integration function from Equation (61) respectively.

```
function PeriodicKernel(X::Array{Float64,2},Y::Array{Float64,2},
    x0::Float64,y0::Float64,sigma::Float64,Lx::Float64,Ly::Float64)
        A1 = 1/(sqrt(2*pi)*sigma); A2 = 1/(sqrt(2*pi)*sigma);
    return A1*exp(-(mod(x0-X+(Lx/2),Lx)-Lx/2).^2/
    (2*sigma^2))*A2*exp(-(mod(y0-Y+(Ly/2),Ly)-Ly/2).^2/(2*sigma^2));
end
function FilterIT(u::Array{Float64,1},Z::Array{Float64,3},r,c)
    Cu1 = reshape(u,g.Nx-1,g.Ny-1);
Ff = zeros((r-2)*(c-2),1);
for i = 1:((r-2)*(c-2))
    F_ = sum(Cu1.*Z[:,:,i])
        Ff[i] = F_
end
    return vec(Ff)
end
```

F.6 Time Integration

These two functions regards the time integration of the system. First function solves the problem in each time step and the second function defines the scheme stated in Section 3.3. Both functions are modified for this specific model study.

```
function integrate(x::Array{Float64,2},
integration_step::Function, time::TimeSteps; tol::Float64=1e-4,
controller::Integer=0)
    t::Float64 = time.t_start
    dt::Float64 = time.dt
    it::Int = 0
    progress::Int = 0
    next_print::Float64 = 0
    dt_old::Float64 = dt
    err::Float64 = tol
    err_1::Float64 = tol
    err_2::Float64 = tol
    alpha::Float64 = 1.0
    beta::Float64 = 1.0
    gamma::Float64 = 1.0
    omega::Float64 = 1.0
    kI = 0.25
    kP = 0.14
    kD = 0.10
    T = zeros(50000,1)
    r,c = size(x)
    X = zeros(r,c,50000)
    T[1] = t
    X[:,:,1] = x
    while t+dt < time.t_end

        # print progress
        while t > next_print
            println("$(progress)% done")
            progress += 5
            next_print = time.t_start +
            (time.t_end-time.t_start) * 0.01*progress
        end

        it += 1
        T[it+1] = T[it]+dt
        #println("calling RK for time $(t) with Delta t=$(dt)")
        x_new, xe = integration_step(x, t, dt)
        err = maximum(abs(xe-x_new))
        X[:,:,it+1] = x_new

        if controller == 0
            q = 1
        elseif controller == 1
            # I controller
            q = (tol/err)^(1/3)
        elseif controller == 2
            # PC controller
            q = (tol/err*err_1/err)^(1/3) * dt/dt_old
            dt_old = dt
        end

        if q > 2
            println("error small (q=$(q)),
            step accepted and Delta t reduced")
            x = x_new
            t += dt
            dt *= q
        elseif q < 0.5
            println("error too large (q=$(q)),
            step rejected and Delta t increased")
            dt *= q
        else
            #println("error normal (q=$(q)), step accepted")
            x = x_new
            t += dt
        end

        err_2 = err_1
        err_1 = err
    end
    x, xe = integration_step(x, t, time.t_end-t) # x is u
    println("integration completed with $(it) iterations")
    return x,T,X
end
```

```
function gen_ARK324L2SAadv(grid::Grid2D, matrix_generator::Function,
fdmatrixUP::Function, solver::Function, D::Array{Float64,1},
    gammag::Array{Float64,1}, reaction::Function; p::Int = 3)

    FD = matrix_generator(grid)
    NUP = fdmatrixUP(grid)
    DD = spdiagm(D)
    Gamma = spdiagm(gammag)

    # coefficients for both ERK and ESDIRK
    sigma = 1767732205903/4055673282236
    c = [0 1767732205903/2027836641118 3/5 1] # not used
    b = [1471266399579/7840856788654 -4482444167858/7529755066697
    11266239266428/11593286722821 sigma]
    be = [2756255671327/12835298489170 -10771552573575/22201958757719
    9247589265047/10645013368117 2193209047091/5459859503100]
    a1 = [0 0 0 0]

    # coefficients for ERK
    a2e = [sigma 0 0 0]
    a3e = [5535828885825/10492691773637 788022342437/10882634858940 0 0]
    a4e = [6485989280629/16251701735622 -4246266847089/9704473918619
    10755448449292/10357097424841 0]

    # coefficients for ESDIRK
    a2i = [sigma sigma 0 0]
    a3i = [2746238789719/10658868560708
    -640167445237/6845629431997 sigma 0]
    a4i = b

    # keep p last results to construct initial guess for next one
    U = [zeros((grid.Nx-1)*(grid.Ny-1), p) for i=1:length(D)]

    function ARK324L2SAadv(u::Array{Float64,2}, t::Float64, dt::Float64)

        # QR factorization of last solutions, used for initial guesses
        QR = [qr(ui) for ui=U]

        xi1  = u
        k1i = dt * FD * xi1 * DD
        k1e = dt * (reaction(xi1).* xi1 -
        NUP*(NUP*reaction(xi1).*xi1)*Gamma) #+ dt * f(t + c[1]*dt)

        xi2  = u + sigma*(k1i+k1e)
        for i=1:length(D)
            xi2[:,i] = solver(dt, i, xi2[:,i], QR[i][1])
        end
        k2i = dt * FD * xi2 * DD
        k2e = dt * (reaction(xi2).* xi2 -
        NUP*(NUP*reaction(xi2).*xi2)*Gamma) #+ dt * f(t + c[2]*dt)

        xi3 = u + a3i[1] * k1i + a3i[2] * k2i + a3e[1] * k1e + a3e[2] * k2e
        for i=1:length(D)
            xi3[:,i] = solver(dt, i, xi3[:,i], QR[i][1])
        end
        k3i = dt * FD * xi3 * DD
        k3e = dt * (reaction(xi3).* xi3 -
        NUP*(NUP*reaction(xi3).*xi3)*Gamma) #+ dt * f(t + c[3]*dt)

        xi4 = u + a4i[1] * k1i + a4i[2] * k2i + a4i[3] * k3i + a4e[1] * k1e + a4e[2
        for i=1:length(D)
            xi4[:,i] = solver(dt, i, xi4[:,i], QR[i][1])
        end
        k4i = dt * FD * xi4 * DD
        k4e = dt * (reaction(xi4).* xi4 -
        NUP*(NUP*reaction(xi4).*xi4)*Gamma) #+ dt * f(t + c[4]*dt)

        ue = u + be[1] * (k1i + k1e) + be[2] * (k2i + k2e) +
        be[3] * (k3i + k3e) + be[4] * (k4i + k4e)
        u += b[1] * (k1i + k1e) + b[2] * (k2i + k2e) +
        b[3] * (k3i + k3e) + b[4] * (k4i + k4e)

        # add new step to last solutions
        for i=1:length(D)
            U[i] = hcat(u[:,i], U[i][:,1:end-1])
        end

        return u, ue
    end
```

```
    return ARK324L2SAadv
end
```

F.7 Multigrid Functions

All following three functions are unmodified and taken from [16]. They all regards the multigrid solver, where the first function defines the multigrid matrices. The second function defines the restriction matrices. Finally the third function is the multigrid solver that creates the restriction matrices used to transfer the model from coarser to finer grid and vice versa and that solves the problem and the error on a given level projecting the problem back and fourth. All three functions are unmodified and taken from [16].

```
type MgMatrices

    jacobi_weight::Float64
    dt::Float64
    D::Float64
    sigma::Float64
    grid::Grid2D

    FD::SparseMatrixCSC{Float64,Int}
    A::SparseMatrixCSC{Float64,Int}
    J1::SparseMatrixCSC{Float64,Int}
    J2::SparseMatrixCSC{Float64,Int}

    function MgMatrices(dt::Float64, D::Float64, sigma::Float64,
    grid::Grid2D, fdmatrix::Function, restriction::Function)

        jacobi_weight = 2/3

        FD = fdmatrix(grid)
        A  = speye(FD) - dt*sigma*D*FD
        J2 = spdiagm(jacobi_weight./diag(A))
        J1 = speye(A) - J2 * A

        new(jacobi_weight, dt, D, sigma, grid, FD, A, J1, J2)
    end
```

```julia
end
```

```julia
function restrictionmatrix(grid::Grid2D)
    ux_old = grid.Nx - 1
    uy_old = grid.Ny - 1
    ux_new = div(grid.Nx,2) - 1
    uy_new = div(grid.Ny,2) - 1
    R = spzeros(ux_new*uy_new, ux_old*uy_old)
    for j = 1:uy_new
        for i = 1:ux_new
            i_new = ux_new*(j-1) + i
            R[i_new, ux_old*(2*j-1)     + 2*i    ] = 4
            R[i_new, ux_old*(2*j-1)     + 2*i - 1] = 2
            R[i_new, ux_old*(2*j-1)     + 2*i + 1] = 2
            R[i_new, ux_old*(2*j-1 - 1) + 2*i    ] = 2
            R[i_new, ux_old*(2*j-1 - 1) + 2*i - 1] = 1
            R[i_new, ux_old*(2*j-1 - 1) + 2*i + 1] = 1
            R[i_new, ux_old*(2*j-1 + 1) + 2*i    ] = 2
            R[i_new, ux_old*(2*j-1 + 1) + 2*i - 1] = 1
            R[i_new, ux_old*(2*j-1 + 1) + 2*i + 1] = 1
        end
    end
    R /= 16
end
```

```julia
function gen_mgsolver(grid::Grid2D, time::TimeSteps,
D::Array{Float64, 1}, sigma::Float64, fdmatrix::Function, restriction::Function,
    v1::Int, v2::Int; pcg::Bool = false,
    tol::Float64=1e-8, max_it::Int = 100)
    # The MG-solver solves the equation (I-dt*D_i*sigma*FD)x=b

    # create main matrices for all levels
    MGM = [MgMatrices(time.dt, D_, sigma, level(grid,i),
    fdmatrix, restriction) for i=1:grid.levels, D_=D]

    # create restriction matrices
    R = [restriction(level(grid,i)) for i=1:grid.levels-1]

    function mgsolver(dt::Real, iD::Int, b::Array,
    x0::Array{Float64} = zeros(b); level::Int = 1)
        #println("mgsolver on level $(level)")
                # regenerate matrices if time step has changed
        if MGM[level, iD].dt != dt
            update_mgmatrices(MGM[level, iD], dt)
        end

        if level == grid.levels
            return MGM[level, iD].A \ b
        end
        # solve reduced system if a projection matrix is passed as x0
        if size(x0,2) > 1
            alpha = (transpose(x0)*MGM[1, iD].A*x0) \ (transpose(x0)*b)
            x0 = x0*alpha
        end

        x = vec(x0) # make sure x is a 1d array
            for it = 1:max_it

            # smoothen problem on current level
            for i = 1:v1
                x = MGM[level, iD].J1 * x + MGM[level, iD].J2 * b
            end

            # restrict residual to next level and solve recursively
            r = R[level] * (b - MGM[level, iD].A * x)
            e = mgsolver(dt, iD, r, level=level+1)

            # prolongate error to current level and add to solution
            x += 4*R[level].' * e

            #post-smoothening on current level
            for i = 1:v2
                x = MGM[level, iD].J1 * x + MGM[level, iD].J2 * b
            end

            if level > 1
                return x
            end
            # only one cycle if preconditioner
            if pcg
                return x
            end

            if norm(b - MGM[level, iD].A * x) < tol
                return x
            end
        end
        println("multi-grid solver reached maximum iterations")
        return x
    end

    function pcgsolver(dt::Float64, iD::Int, b::Array{Float64,2},
    x0::Array{Float64,2} = zeros(b))

        # regenerate matrices if time step has changed
        if MGM[1, iD].dt != dt
            update_mgmatrices(MGM[1, iD], dt)
        end

        # solve reduced system if a projection matrix is passed as x0
        if size(x0,2) > 1
            alpha = (transpose(x0)*MGM[1, iD].A*x0) \ (transpose(x0)*b)
            x0 = x0*alpha
        end

        # conjugate gradient preparation
        x = vec(x0) # make sure x is a 1d array
        r_old = b - MGM[1, iD].A * x
        Mr_old = mgsolver(dt, iD, r_old, zeros(r_old))
        p = Mr_old

        for it = 1:max_it

            # test convergence
            if norm(r_old) < tol
                return x
            end

            # conjugate gradient step
            Ap = MGM[1, iD].A * p
            alpha = dot(r_old, Mr_old) / dot(p, Ap)
            x += alpha * p
            r = r_old - alpha*Ap
            Mr = mgsolver(dt, iD, r, zeros(r))
            beta = dot(r, Mr) / dot(r_old, Mr_old)
            p = Mr + beta * p
            r_old = r
            Mr_old = Mr

        end
        println("conjugate gradient solver reached maximum iterations")
        return x
    end

    if pcg
        return pcgsolver
    else
        return mgsolver
    end
end
```

F.8 Helper Functions

Following three functions are all to define timesteps, get coarser level of a given grid and to update the matrices used, when the multigrid solver is called respectively. All three functions are unmodified and taken from [16].

```
immutable TimeSteps
    t_start::Float64
    t_end::Float64
    Nt::Int
    dt::Float64

    TimeSteps(t_start, t_end, Nt) = new(t_start, t_end, Nt,
    (t_end-t_start)/Nt)
```

```
end
# helper function to get coarser levels of a grid
function level(grid::Grid2D, lvl::Int)
    if lvl > grid.levels
        error("ERROR: Grid doesn't support level $(level).")
    end
    Grid2D(grid.x_start, grid.x_end, grid.y_start, grid.y_end,
     round(Int,grid.Nx/2^(lvl-1)), round(Int,grid.Ny/2^(lvl-1)),
     grid.levels-(lvl-1))
end
function update_mgmatrices(mgm::MgMatrices, dt::Float64)
    mgm.A  = speye(mgm.FD) - dt*mgm.sigma*mgm.D*mgm.FD
    mgm.J2 = spdiagm(mgm.jacobi_weight ./ diag(mgm.A))
    mgm.J1 = speye(mgm.J2) - mgm.J2 * mgm.A
end
```

F.9 Main Script

This is the main script where the model problem is defined. All parameters are specified, and all above functions are included. Visualizations of the computed results is generated here as well.

```
using PyPlot
using NBInclude
nbinclude("Grid2D.ipynb")
nbinclude("Meshgrid.ipynb")
nbinclude("FDmatrix4.ipynb")
nbinclude("FDmatrix4Nabla.ipynb")
nbinclude("RestrictionMatrix.ipynb")
nbinclude("MGMatrices.ipynb")
nbinclude("Coordinates.ipynb")
nbinclude("Kernel.ipynb")
nbinclude("Functions.ipynb")
nbinclude("MGsolver.ipynb") # FD MG solver
nbinclude("RungeKuttaSolver.ipynb")
nbinclude("Integrate.ipynb")
## Model Problem
final_time = 10
# Generate Grid
```

```
    order_space = 6
    Lx = 20*pi
    Ly = 20*pi
    g = Grid2D(0,Lx,0,Ly,2^order_space,2^order_space,order_space-1)
    t = TimeSteps(0, final_time, round(Int,100*final_time))
    X,Y = get_coordinates(g)
    sigma 1 = 0.5

# Gauss Kernel
    r,c = size(X);
    p = hcat(vcat(X[2:end-1,2:end-1]...), vcat(Y[2:end-1,2:end-1]...))
    Zz = zeros(r-2,c-2,(r-2)*(c-2));
    for i = 1:((r-2)*(c-2))
    weight = PeriodicKernel(X[2:end-1,2:end-1],
    Y[2:end-1,2:end-1],p[i,1],p[i,2],sigma 1,Lx,Ly)/
    sum(sum(PeriodicKernel(
    X[2:end-1,2:end-1],Y[2:end-1,2:end-1]
    ,p[i,1],p[i,2],sigma 1,Lx,Ly)));
    pep = weight;
    pep[find(weight .<=10.0^-4)]=0;
    Pp = sparse(pep);
    Zz[:,:,i]=Pp;
    end
# Reaction parameters
    a_ = 5
    c_ = 1.5
    r_ = 2.8
    K_ = 28/3
    A_ = vcat(hcat((-r_/K_)+(a_/4),-a_/2),hcat(a_/4,-c_))
# linearized system

    ff(u_,v_) = r_.*(1-(u_./K_))-((a_.*v_)./(u_+1));
    gg(u_,v_) = ((a_.*u_)./(u_+1))-1-(c_.*v_);

    F(u_,v_) = r_.*(1-(u_./K_))
    -(a_.*FilterIT(v_./(1+FilterIT(u_,Zz,r,c)),Zz,r,c));
    G(u_,v_) = ((a_.*FilterIT(u_,Zz,r,c))
    ./(1+FilterIT(u_,Zz,r,c)))-1-(c_.*v_);
    # reaction terms of Model
    reaction(u) = hcat(ff(u[:,1],u[:,2]),gg(u[:,1],u[:,2]));
    reactionKERNEL(u) = hcat(F(u[:,1],u[:,2]),G(u[:,1],u[:,2]));

# define taxis and diffusion parameters
    yg = [0.5, 0.5]
    D  = [1.0, 15.0]

# prepare initial conditions: equilibrium points plus perturbation
    k_x = 2*pi/Lx
    k_y = 2*pi/Ly
    B_  = -0.75
    u0  = Float64[((1+tanh(((x-26)-(y-24))/4))/2)
    for x=g.x_start:g.dx:g.x_end, y=g.y_start:g.dy:g.y_end];
    v0  = Float64[((1+tanh((-(x-24)+(y-26))/4))/2)
    for x=g.x_start:g.dx:g.x_end, y=g.y_start:g.dy:g.y_end];
    Cc1  = Float64[-sin(k_y.*y)
    for x=g.x_start:g.dx:g.x_end, y=g.y_start:g.dy:g.y_end];
    Cc2  = Float64[sin(k_x.*y)
    for x=g.x_start:g.dx:g.x_end, y=g.y_start:g.dy:g.y_end];
    C1  = u0.*(1+B_*Cc1)
    C2  = v0.*(1+B_*Cc2)

    sigma = 1767732205903/4055673282236
    fdmatrix(g) = fdmatrix4(g, EBC=false)
    fdmatrixadv(g) = fdmatrix4UP(g, EBC=false)

    # solve unstable problem
    x = hcat(vcat(C1[2:end-1,2:end-1]...), vcat(C2[2:end-1,2:end-1]...))
    # inital u and v ;
    solver = gen_mgsolver(g, t, D, sigma, fdmatrix, restrictionmatrix,
    3, 3, pcg=false, tol=1e-6, max_it=100);
    integration_step = gen_ARK324L2SAadv(g, fdmatrix, fdmatrixadv, solver,
    D, yg, reactionKERNEL, p=3)
    x,Tt,U = @time integrate(x, integration_step, t, tol=1e-3, controller=1)
    C1_unstable = reshape(x[:,1],g.Nx-1,g.Ny-1);
    C2_unstable = reshape(x[:,2],g.Nx-1,g.Ny-1);
order_space = 6
Lx = 20*pi
```

```
Ly = 20*pi
g = Grid2D(0,Lx,0,Ly,2^order_space,2^order_space,order_space-1)
t = TimeSteps(0, final_time, round(Int,100*final_time))
X,Y = get_coordinates(g)
swag = hcat(vcat(X[2:end-1,2:end-1]...))
Jj = 920 # timestep
point = Int(round(3969/2))+3
C1_unstable = reshape(U[:,1,Jj],g.Nx-1,g.Ny-1);
Tid = Int(round(Tt[Jj]))
rc("xtick",labelsize=15)
rc("ytick",labelsize=15)
pcolormesh(Y[2:(end-1),2:(end-1)],X[2:(end-1),2:(end-1)],C1_unstable)
title("After time $Tid")
axis("off")
colorbar()
Uu = zeros(63,63,Jj)
Uv = zeros(63,63,Jj)
for i = 1:Jj
    Uu[:,:,i] = reshape(U[:,1,i],g.Nx-1,g.Ny-1);
    Uv[:,:,i] = reshape(U[:,2,i],g.Nx-1,g.Ny-1);
end
BoelgeU = zeros(63,Jj)
BoelgeV = zeros(63,Jj)
for i=1:Jj
    BoelgeU[:,i] = Uu[:,31,i]
    BoelgeV[:,i] = Uv[:,31,i]
end
## Visualization of Model Problem
for i = 100:50:Jj
    rc("xtick",labelsize=15)
rc("ytick",labelsize=15)
    plot(BoelgeU[:,i])
    hold("on")
end
ax = gca()
ax[:set_xlim]([0,Lx-g.dx])
xlabel("y");ylabel("Prey Density");
r,c,n = size(U)
sumU = zeros(1,n)
```

```
meanU = zeros(1,n)
varU = zeros(1,n)
pointU = zeros(1,n)
for i = 1:n
    sumU[i] = sum(U[:,1,i])
    meanU[i] = mean(U[:,1,i])
    varU[i] = var(U[:,1,i])
    XU = U[:,1,i]
    pointU[i] = XU[point]
end

sumV = zeros(1,n)
meanV = zeros(1,n)
varV = zeros(1,n)
pointV = zeros(1,n)
for i = 1:n
    sumV[i] = sum(U[:,2,i])
    meanV[i] = mean(U[:,2,i])
    varV[i] = var(U[:,2,i])
    XV = U[:,2,i]
    pointV[i] = XV[point]
end
P = Jj
W = round(swag[point],2)
figure(1)
fig = figure(figsize=(8,6))
rc("xtick",labelsize=15)
rc("ytick",labelsize=15)
ax = gca()
ax[:plot](Tt[1:P],sumU[1:P],label="Prey")
hold(true)
ax[:plot](Tt[1:P],sumV[1:P],label="Predator")
ax[:legend](loc="upper right")
xlabel("Time");ylabel("Sum")
figure(2)
rc("xtick",labelsize=15)
rc("ytick",labelsize=15)
fig = figure(figsize=(8,6))
ax = gca()
```

```
ax[:plot](Tt[1:P],meanU[1:P],label="Prey")
hold(true)
ax[:plot](Tt[1:P],meanV[1:P],label="Predator")
ax[:legend](loc="upper right")
xlabel("Time");ylabel("Mean")
figure(3)
rc("xtick",labelsize=15)
rc("ytick",labelsize=15)
fig = figure(figsize=(8,6))
ax = gca()
ax[:plot](Tt[1:P],pointU[1:P],label="Prey")
hold(true)
ax[:plot](Tt[1:P],pointV[1:P],label="Predator")
ax[:legend](loc="upper right")
xlabel("Time");ylabel("Density")
figure(1)
fig = figure(figsize=(8,6))
rc("xtick",labelsize=15)
rc("ytick",labelsize=15)
ax = gca()
ax[:plot](Tt[1:P],varU[1:P],label="Prey")
hold(true)
ax[:plot](Tt[1:P],varV[1:P],label="Predator")
ax[:legend](loc="upper right")
xlabel("Time");ylabel("Variance")
figure(2)
fig = figure(figsize=(8,6))
rc("xtick",labelsize=15)
rc("ytick",labelsize=15)
ax = gca()
ax[:plot](meanU[1:P],varU[1:P],label="Prey")
hold(true)
ax[:plot](meanV[1:P],varV[1:P],label="Predator")
ax[:legend](loc="upper left")
xlabel("Mean");ylabel("Variance")
```