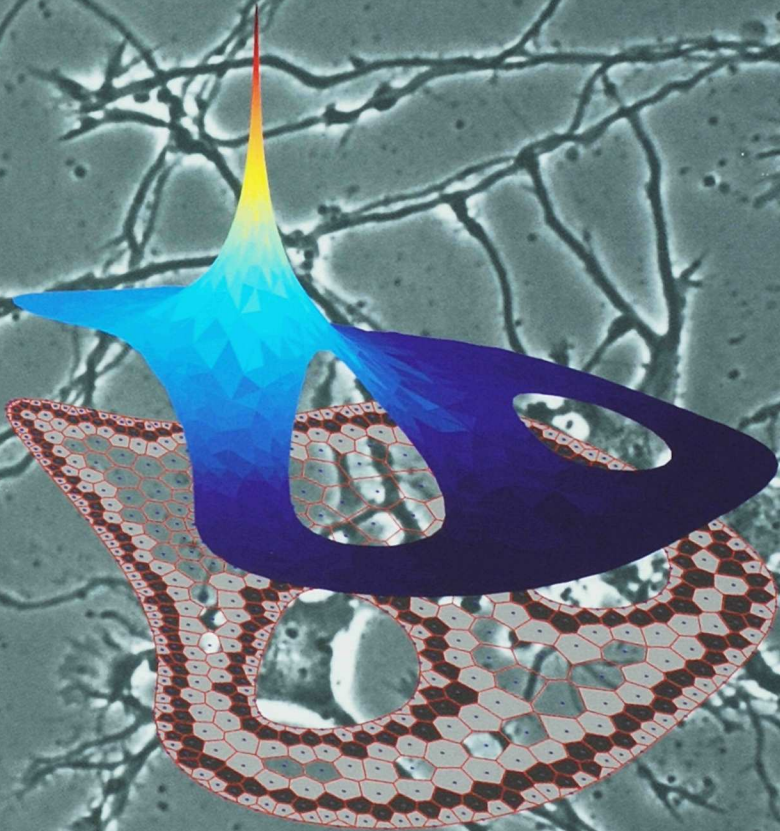


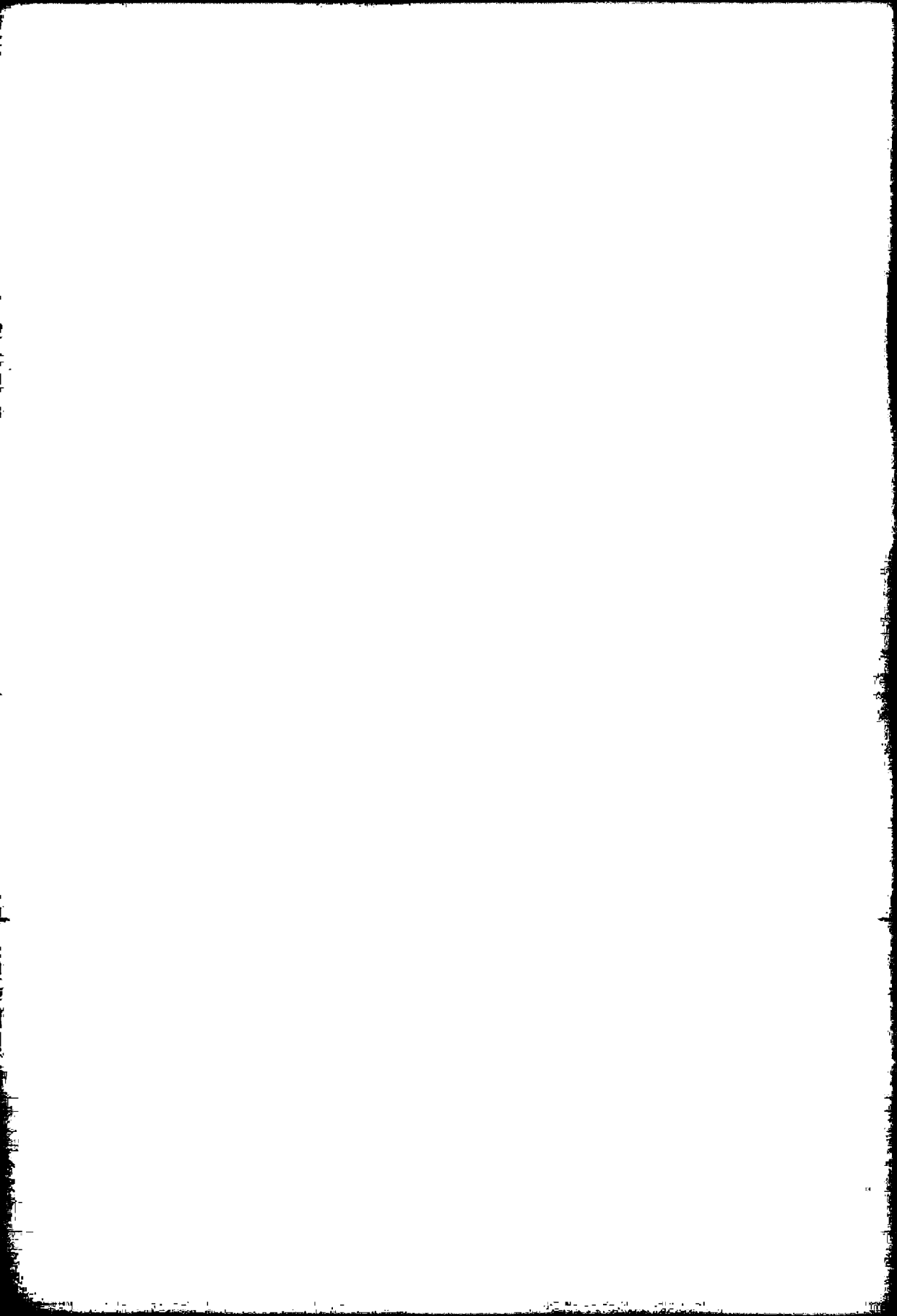
On the numerical solution of
diffusion systems with localized,
gradient-driven, moving sources



Johannes Krottje

On the numerical solution of diffusion systems with
localized, gradient-driven, moving sources

Johannes Krottje



On the numerical solution of diffusion
systems with localized, gradient driven,
moving sources

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. mr. P.F. van der Heijden
ten overstaan van een door
het college voor promoties ingestelde commissie,
in het openbaar te verdedigen in de Aula der Universiteit
op donderdag 17 november 2005, te 12:00 uur

door

Johannes Klaas Krottje

geboren te Sint Annaparochie

Promotiecommissie

Promotor: Prof. dr. J.G. Verwer

Overige leden: Prof. dr. J. Lang
Prof. dr. P.W. Hemker
Prof. dr. A. Doelman
Prof. dr. ir. B. Koren
Dr. A. van Ooyen
Dr. J. van Pelt
Dr. W. Hoffmann

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Dit werk werd financieel gesteund door de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) in het kader van het programma 'Wiskunde Toegepast'.

Preface

This thesis records the research that I conducted between 2001 and 2005, as a member of the cluster Modelling, Analysis and Simulation of the Centrum voor Wiskunde en Informatica (CWI) in Amsterdam.

Numerous people have contributed to the realization of this thesis. In the first place I would like to thank my promotor Jan Verwer for the many things I learned from him, for the freedom he gave me in doing my research and for his advice concerning this work. It has been a real pleasure to work with Jan and I am very grateful he gave me the opportunity to work on this project.

From the Vrije Universiteit I would like to thank Arjen van Ooyen for his discussions on axon guidance modelling, writing a paper together, and letting me experience the communication problems that arise when two specializations start to collaborate. In this context I am also grateful to Jaap van Pelt from the Netherlands Institute for Brain Research. I always felt very welcome when visiting the NIBR for discussions with Jaap and Arjen and when I needed material for a presentation, Jaap sent me very nice movies of growing axons. I also would like to thank Jens Lang of Darmstadt University of Technology for his help on using the software package Kardos. He delivered valuable background information on the architecture of the software. I thank Odo Diekmann for some valuable input concerning the modelling of biological phenomena.

I would like to thank Ger Ramakers of the NIBR for providing the axon pictures that I used for the cover of this thesis.

I enjoyed working at CWI a lot, which can be largely subscribed to the many nice colleagues I had. I should start with my roommate and 'buurman', Jason Frank. In the beginning we shared a room, but when Jason became 'more important' he decided that he needed his own room. As a matter of fact I got my own room also, which is very exceptional for a PhD student at CWI. Therefore it was not a bad development at all and, actually, not much of a change either. Jason kept walking into my room very often, starting to read phrases from his 'taalkalender', writing something on my whiteboard or sharing one of his being-father-experiences with me. Jason, thanks for being my colleague for the past four years, and in particular for your help and advice on numerical issues. In this regard I am also grateful to Ben Sommeijer for helpful discussions in the first year of my PhD study.

In such a job spending hours behind your monitor, some exercise every now and

then can keep you going. Therefore I am very grateful to Mervyn Lewis and Joke Blom for playing table tennis once and a while and to Herman te Riele for organizing the table tennis tournament. Also I would like to thank Pieter Jan 't Hoen, David Echeverria, and many others for playing squash every now and then.

Just for being nice colleagues I thank Carolynne Montijn, Daniel Benden and Michael Smeding and further everybody of MAS.

Finally, I am very grateful to my parents, brothers, Marsha's family and, of course, Marsha, who supported me all the way and made the last four years worthwhile. Marsha, thank you for just being there.

Johannes Krottje

Contents

Preface	i
1 Introduction	1
1.1 Biological phenomena	1
1.2 Modelling framework	2
1.3 Numerical mathematics	3
1.4 Thesis outline	4
2 On the dynamics of a mixed parabolic-gradient system	7
2.1 Introduction	7
2.2 Mixed parabolic-gradient systems	9
2.3 Quasi-steady-state approximation	10
2.4 Source functions	14
2.5 Self-interaction	18
2.6 Numerical tests	21
2.7 Conclusions	24
3 Domain definition with Bézier curves	27
3.1 Bézier curves and paths	27
3.2 Node choosing on curves	28
3.3 Choosing the monitor function M	31
3.4 Number of nodes based on maximal distances	33
3.5 Combining multiple curves	33
3.6 Semi-uniform grids	35
4 Spatial discretization of the field equations	39
4.1 Introduction	39
4.2 Meshfree function approximation	41
4.3 Discretization of the PDE	48
4.4 Practical considerations	51
4.5 Choosing nodes in the domain	53
4.6 Numerical tests	58
4.7 Summary	61

5	A mathematical framework for modelling axon guidance	63
5.1	Introduction	63
5.2	Simulation framework	65
5.3	Underlying mathematical model	68
5.4	Numerical methods	72
5.5	Simulation examples	76
5.6	Discussion	85
6	Numerical solution of the framework's equation systems	91
6.1	Introduction	91
6.2	Simulation framework	92
6.3	Numerical methods in AGTools	96
6.4	Introduction to Kardos	100
6.5	Application of Rosenbrock methods	103
6.6	Comparison between Kardos and AGTools	108
6.7	Summary	122
	Summary	125
	Samenvatting	127

Chapter 1

Introduction

This thesis concerns two major aspects of applied mathematics. On the one hand, the modelling process, in which real-life phenomena are rephrased in terms of mathematical objects, resulting in a so-called mathematical model. On the other hand, the analysis of such a set of objects, resulting in a number of properties of the model that give more understanding of the underlying phenomena.

The phenomena considered in this thesis, which are from the realm of biology, are introduced in Section 1.1. Section 1.2 gives the basic ideas behind the rephrasing of the biology into mathematical objects. For models with a large degree of complexity, the mathematical analysis will for a large part be based on results from the field of numerical mathematics, as is also true for the systems in this thesis. In Section 1.3 an overview is given of some issues that will be encountered in the numerical analysis of the systems at hand. This introduction will then be concluded with an outline of this thesis in Section 1.4.

This thesis has grown out of a co-operative project 'Numerical Modelling of the Formation of Neuronal Connections in the Nervous System' between the CWI and the Netherlands Institute for Brain Research, research group Neurons & Networks, with involved researchers Jaap van Pelt and Arjen van Ooyen (now at VU). The project was supported by NWO, Programme 'Wiskunde Toegepast', No. 613.002.048.

1.1 Biological phenomena

The phenomena that are being modelled in this thesis come from the fields of neurobiology and/or computational neuroscience. A great part of the internal signaling process in a large class of living creatures is handled by a nervous system. Such a system is composed of interconnecting nerve cells, also called neurons, that communicate with each other by sending electrical signals through their connections. Such connections are called axons and the signal sending is called firing. Basically, a cell decides to fire a signal on the basis of the signals it receives from other cells. The signal originates in the cell body, travels along the axon, and reaches other neurons at

their dendrites, i.e., tree-like structures where the axons are connected to the target cells.

There has been done a lot of research into this firing process as well as into the signal transduction along the axon, the underlying mechanisms of the latter being fairly well-known. Models have been proposed that describe the traveling signals and there is simulation software that allows for combining and examining different mechanisms concerning this process. In this thesis the focus will not be on the functioning of such a system, but rather on the development of it. The basic underlying question that is considered here is: how do nerve cells make connections with other nerve cells? We are interested in the mechanisms underlying the guidance of a growing axon toward a target cell, not in why the system decides to make certain connections.

This process of guidance of the axon toward its target is often referred to as axon guidance. A basic explanation of it is that the process belongs to the class of chemotaxis processes. This means that the growth direction of the tip of the axon, which is called the growth cone, is determined by concentration levels of certain molecules that are present in the environment. Such molecules we will refer to by the term guidance molecules. Basically, the growth cone tries to measure the gradients of the concentration fields of guidance molecules, and will grow toward higher concentration of so-called attractant molecules and away from so-called repellent molecules.

It is known that certain species of molecules influence the growth of axons by means of chemotaxis, but the precise mechanisms underlying this process are obscure. As an example serves the combined influence of a group of guidance molecules. Here the question arises how a growth cone does decide on a single growth direction while measuring a number of non-parallel gradients?

Another example of an unknown mechanism concerns the adjustment of the size of the growth cone. It is observed that a growth cone, which has the form of a hand-like structure, can adjust the size of its 'fingers'. If the concentration of guidance molecules in the local environment is low, enlargement of these sensing structures will probably keep the growth cone's measurement of gradients sufficiently reliable. The precise size regulating mechanisms are unclear, as well as how they effect the sensibility for the different guidance molecules.

1.2 Modelling framework

One of the key issues in mathematical modelling is what to incorporate into the model and what to neglect. Choices have to be made on which details will be built in and which are left out because they make the model unnecessarily complicated. These are hard choices that require knowledge of the field in which is modelled as well as mathematical insight. Especially with biological phenomena it is relatively easy to build models that are so complicated that it is very difficult to gain insight using a mathematical analysis of the model equations.

In this thesis the starting point is not a single biological setting but rather a whole class. Not a single model has to be developed to give more understanding of its

underlying phenomena, but it should be possible to build a number of variations of a model and after analysis select the most appropriate one. The goal of the modelling activity here is to learn which combination of elements can explain the observed phenomena.

Given the need for this kind of flexibility a framework has been developed that can be used to model axon guidance. Its basic assumptions are that the systems to be modelled can be regarded as composed of concentration fields of guidance molecules and objects with a certain location that interact with these fields. In the background the environment is modelled by a domain on which the fields are defined and in which the objects are located. A short description will now follow of the domain, fields and objects.

It is important to notice that flexibility with respect to domain geometry is required. It should be possible to work with domains that have strange shapes and, possibly, contain holes. Such holes might represent bone structures or blood vessels that are impenetrable for the guidance molecules, thus forming obstacles around which the axons have to grow. The basic assumption made on the specification of the domain is that it is 2-dimensional and specified by piecewise smooth curves in the plane.

The fields represent the concentration fields of guidance molecules and are non-negative functions defined on the domain. Their dynamics are governed by diffusion, absorption, and excretion. It is assumed that there is no in- or outflow of guidance molecules across the boundaries. An important characteristic is that the excretion takes place through highly localized sources, i.e., at the locations of the objects, that might be moving through the domain. A model choice is to represent the sources by functions with a small support and not by Dirac delta functions. The main reason for this choice is that when using the latter, the resulting fields are singular at the location of the source, possibly leading to ill-defined systems of equations.

The final class of elements in the framework consists of the objects, representing locations of interaction with the fields. As examples serve target cells and growth cones excreting guidance molecules or a point where chemicals are injected into the environment artificially in an experimental setting. In this thesis the objects will also be referred to with the term 'states', because they can be thought of to describe the internal states of the objects. They are finite-dimensional vectors of which the first two elements describe the location. Additional elements of the vectors are used to model the objects further and can contain variables like the sensitivity to certain guidance molecules, growth speed, excretion rate, etc.

1.3 Numerical mathematics

Using the framework described in Section 1.2 the biological setting is translated into a mathematical description. This description consists of two sets of functions, the first set describing the fields by means of real functions of space and time, and the second set describing the states, being functions that map time to finite-dimensional

real vectors. While the dynamics of the fields are given by PDEs (partial differential equations), the dynamics of the states are given by ODEs (ordinary differential equations). The two systems are coupled in two ways. The sources in the field equations depend in their excretion rates and locations on the states. The dynamics of the states depend on the fields in the local vicinity of the states.

The numerical approximation of the time evolution of the fields and states requires a spatial discretization of the domain and the fields and a temporal discretization for the fields and the states. Some properties of the system will now be discussed that are characteristic from a numerical perspective.

Concerning the spatial discretization of the fields, two features of the framework are of importance. First, the domain does not need to be a regular shape to which a grid can be assigned in a straightforward way. Second, the supports of the moving sources are small compared to the domain size, meaning that certain refinement and adaptivity techniques have to be applied.

Based on these two features the class of meshfree methods was taken as a starting point for the selection of a spatial discretization. The underlying idea is that the discretization of the fields consists of arbitrary sets of nodes in the domain, together with values defined on the nodes. This allows for easy refinement by selecting many nodes in the neighborhood of the moving sources. On the other hand it gives the freedom to align the nodes nicely along the boundaries.

A drawback of such a meshfree approach is that it lacks the straightforward definition of neighboring nodes present in grids and triangulations. For every node a selection of neighboring nodes is required for building a local approximation in the vicinity of the node at hand. In practice, some kind of grid structure is often used for this. In this thesis Voronoi diagrams are used for this purpose as well as for building global approximations out of local approximations. As a result, the field discretization, although being meshfree in its ideas, depends heavily on Voronoi diagrams.

Concerning the temporal discretization, an important feature of the coupled systems is that the diffusion operator gives rise to stiffness in the discretization of the PDEs, while the ODEs, which are nonlinearly coupled to the PDEs, are non-stiff. The stiffness of the discretized PDEs would be a reason to use an implicit time integration scheme. However, an implicit scheme for the whole coupled system is very complex due to the typical nature of the coupling, i.e., evaluation of one dependent variable (a field) in another dependent variable (a location of a state). In this thesis Implicit/Explicit (IMEX) schemes are therefore considered for the systems at hand as well as Rosenbrock methods, using an approximate Jacobian.

1.4 Thesis outline

This thesis is based on three articles, a technical report, and an additional chapter on Bézier curves.

Chapter 2 considers some analytical aspects of a typical prototype of the class of systems we are interested in. In this warm-up chapter the main result is the analysis

of a system feature that is called self-interaction. Due to self-interaction presence the use of point sources is prohibited and it turns out that scaling down the supports of the sources will strongly influence the dynamics. The chapter has appeared as:

J.K. Krottje. On the dynamics of a mixed parabolic-gradient system. *Communications on Pure and Applied Analysis*, 2(4):521–537, December 2003.

In Chapter 3 a specification method for the domains is described. It starts with domains that are specified by Bézier paths that determine the outer boundary and a set of internal boundaries. For solving the field equations we will not work with these Bézier paths, however. Instead we will use approximations that consist of a selection of points along the boundaries, where neighboring points are connected by straight line segments. The main part of this chapter is devoted to the selection of nodes by means of an equidistribution principle that is based on arc-length and curvature.

Chapter 4 continues with the presentation of the spatial discretization of the field PDEs. It describes a function approximation technique based on local least-squares approximations and the combination of such approximations into a global approximation. A Voronoi diagram is being used for the selection of neighbors, for the definition of the global approximations, and for the selection of nodes in the domain. In this chapter the discretization is used to solve the steady-state solutions to the field PDEs. It will appear as:

J.K. Krottje. A variational meshfree method for solving time-discrete diffusion equations. *Journal of Computational and Applied Mathematics*, 2005, accepted.

In Chapter 5 the modelling framework is presented and this concerns joint work with Arjen van Ooyen, department of experimental neurophysiology, Free University of Amsterdam. It defines the domain, fields and states and their coupling. It also gives a short overview of the used numerical techniques. A large part of it is devoted to example models that are implemented in the framework. The chapter is expected to appear as:

J.K. Krottje and A. van Ooyen. A mathematical framework for modelling axon guidance. *Bulletin of Mathematical Biology*. In the process of revision.

Chapter 6 compares our numerical approach used for the equation systems that arise in the presented framework with another approach. In this context our numerical approach is referred to as ‘AGTools’ (Axon Guidance Tools). The software package Kardos is taken as a representative of the class of Finite Element Methods and its use for numerically solving the systems at hand is discussed. The chapter presents a number of examples where both approaches are compared, although to a rather limited extent due to time constraints and unexpected difficulties in adjusting the Kardos software for our application. It will appear as a CWI technical report:

J.K. Krottje. Numerical solution of axon guidance framework systems *CWI Technical Report*.

Chapter 2

On the dynamics of a mixed parabolic-gradient system

2.1 Introduction

In a paper of Hentschel & Van Ooyen [21] a mathematical model is presented on the growth of neural connections in the nervous system. The model describes the outgrowth of axons from neurons to targets in a developmental phase for innervation. It is assumed that the growth toward the targets is partly guided by the gradients of concentration fields of certain chemicals which are present in the environment. These concentration fields change in time due to the release of the chemicals by the targets and the growth cones and the processes of diffusion and absorption.

One of the goals of the model is to better understand the observed effects of bundling and debundling of the growth cones. The assumption that the growth cones themselves, besides the targets, also release chemicals that influence the growth of the cones might explain the bundling and debundling effects.

In the model two kinds of variables are used to describe this biological system. First, functions of time $r_\alpha: \mathbb{R} \rightarrow \mathbb{R}^2$, which denote the positions of the growth cones, where α ranges over the number of axons N_d .¹ Second, fields $\rho_\beta: \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}$, which denote the concentration of the chemicals as a function of space and time, where β ranges over the number of concentration fields N_c . The dynamics are described by gradient equations for the growth cone positions

$$\frac{d}{dt}r_\alpha(t) = \sum_{\beta=1}^{N_c} \lambda_{\alpha,\beta} \nabla \rho_\beta(r_\alpha(t), t). \quad (2.1)$$

¹We will regard the growth cones as 'dynamical sources' to which the subscript 'd' refers.

for all $\alpha = 1, \dots, N_d$ and for the concentration fields we have parabolic equations

$$\partial_t \rho_\beta(x, t) = D_\beta \Delta \rho_\beta(x, t) - \kappa_\beta \rho_\beta(x, t) + \sum_{\alpha=1}^{N_d} \sigma_{\beta, \alpha}^d S_{r_\alpha}(t)(x) + \sum_{\gamma=1}^{N_s} \sigma_{\beta, \gamma}^s S_{s_\gamma}(x), \quad (2.2)$$

for all $\beta = 1, \dots, N_c$. Here $x = (x_1, x_2)$, γ ranges over the number of targets N_s and s_γ denotes the position of target γ .² The growth cones and targets act as sources for the concentration fields located at the positions r_α and s_γ . The functions $S_{r_\alpha} : \mathbb{R}^2 \rightarrow \mathbb{R}$, where we left out the argument t , can be considered as source profiles for the growth cone sources, which translate with r_α . The coefficient $\sigma_{\beta, \alpha}^d$ can then be interpreted as the excretion strength of growth cone α with respect to chemical β and this coefficient can be a function of time and of the fields $\rho_1, \dots, \rho_{N_c}$ evaluated at r_α . An analogous interpretation holds for S_{s_γ} and $\sigma_{\beta, \gamma}^s$.

In this chapter we want to gain more insight into the dynamics of this mixed system. Although the ultimate goal is to find a suitable numerical method for the system, most of the chapter will be analytical work. Verwer and Sommeijer [50] used the explicit Runge-Kutta-Chebyshev method and found that the system is highly sensitive in its parameters and source terms with respect to bundling and debundling. A similar conclusion was reported in a second numerical paper by Lastdrager [32]. Therefore we want to gain understanding on the relative importance of parameters, the sensitivity of the dynamics with respect to these parameters and the effects that the choice of used source functions has on the dynamics, where one can think of block, cone or even δ -functions.

Hentschel & Van Ooyen [21] use in their simulations a quasi-steady-state-approximation (QSSA) for the parabolic equations so that the system reduces to a system of ODEs. By using QSSA, the parabolic equations become elliptic equations that can be solved explicitly in some cases. The solutions of the elliptic equations then depend on r_α alone and substitution of these solutions in the gradient equations, results in a closed ODE system. We will discuss to what extent QSSA is profitable by examining when it can be used and what its benefits are. As an example we define a specific 1-dimensional system of the form (2.1)–(2.2), that we can solve analytically by using the QSSA assumption. We will compare this solution to numerical solutions of the full system.

The contents is as follows. We will start with some remarks on the mixed parabolic-gradient system in Section 2 and the QSSA-approximation in Section 3. In Section 4 we will discuss some possible choices for source functions and in Section 5 we will examine an effect that we will call self-interaction. Section 6 is devoted to numerical integration of the system. Here, also, the QSSA solution is compared to the numerical integration of the full system.

²We will regard the targets as ‘static sources’ to which the subscript ‘s’ refers.

2.2 Mixed parabolic-gradient systems

The system consists of N_d gradient equations (2.1) and N_c parabolic equations (2.2). For the domain we will take $t \geq 0$ and $\Omega = [0, 1]^2$ and we will assume periodic boundary conditions because the boundary does not play an essential role here and it is convenient from a numerical point of view. Further we need the initial values for r_α and ρ_β . The coupling between the gradient and parabolic equations occurs through the source terms S_{r_α} in the parabolic equations and of course through the gradient terms in the gradient equations.

2.2.1 Gradient equations

We outline some properties of the gradient equations. With the fields ρ_β as given functions of space and time, equation (2.1) is of the form $\dot{r} = f(r, t)$. If we use the notation $\Phi_\alpha = \sum_{\beta} \lambda_{\alpha, \beta} \rho_\beta$, then equation (2.1) becomes

$$\frac{dr_\alpha}{dt}(t) = \nabla \Phi_\alpha(r_\alpha(t), t). \quad (2.3)$$

If, for all β , ρ_β is twice continuously differentiable with respect to space and continuous with respect to time, $\nabla \Phi_\alpha$ is Lipschitz continuous and existence and uniqueness of solutions is guaranteed. For a fixed, time independent Φ_α , the stationary points are characterized by $\nabla \Phi_\alpha(x) = 0$. For arbitrary solutions $r(t)$ of (2.3), $\Phi_\alpha(r(t))$ is non-decreasing in time and therefore, the local maxima of Φ_α are stable stationary points and the minima and saddle points are unstable stationary points of equation (2.3). The field Φ_α is the weighted sum of the fields $\rho_1, \dots, \rho_{N_c}$. Hence, r_α tends to grow in a direction of increasing ρ_β with $\lambda_{\alpha, \beta} > 0$ and decreasing ρ_β with $\lambda_{\alpha, \beta} < 0$. The former are called fields of attractants whereas the latter are called fields of repellents.

A point that deserves some attention is that the extrema of Φ_α need not be equal to the maxima and minima (for repellent fields) of the separate ρ_β . This means that if we have a set of targets all contributing to an attractant field, then the stable stationary points of equation (2.1) need not be equal to the locations of the targets. In particular, if two targets are close to each other, there might be one stable stationary point in between, instead of two stable stationary points near the locations of the targets.

The existence of the gradients in the points $(r_\alpha(t), t)$ of the functions Φ_α will be discussed in Section 2.4, where we examine the use of different kinds of source functions.

2.2.2 Parameter ranges and the domain

The parameter values are not known exactly, but we will make assumptions on their orders of magnitude. In Table 2.1 some estimated model related quantities are shown. Here v is the growth speed of the growing axon, L_{cones} is the diameter of the growth cone and L_{target} is the distance which the axons have to grow across. Further on we

Parameter	Approximate order
D	10^{-4} mm ² /s
v	10^{-6} -- 10^{-4} mm/s
L_{cones}	10^{-2} mm
L_{target}	10^{-1} -- 1 mm

will use a parameter ℓ that measures the radius of the (circular) support of the source functions S_{r_α} . It seems reasonable to use $\ell = L_{\text{cones}}/2$.

The parameters κ_β , $\sigma_{\beta,\alpha}^d$, $\sigma_{\beta,\gamma}^s$, and $\lambda_{\alpha,\beta}$ are not known and can be used for tuning. However, this does not mean that they can be chosen independently. For instance, the growth speed of axon α at a certain point in time and space is a homogenous function of the parameters $\lambda_{\alpha,1}, \dots, \lambda_{\alpha,N_c}$, as well as the parameters $\sigma_{\beta,\alpha}^d$ and $\sigma_{\beta,\gamma}^s$ for all α , β and γ . Therefore, if the $\sigma_{\beta,\gamma}^s$ and $\sigma_{\beta,\gamma}^s$ are all multiplied by a certain factor, then the $\lambda_{\alpha,\beta}$ should all be divided by approximately the same factor to keep the speed of the axons in the right range.

All parameters and variables will be measured in units of millimeters and seconds.

2.3 Quasi-steady-state approximation

We want to consider the use of a quasi-steady-state approximation instead of the parabolic equations (2.2), as is done in [21]. In this approximation, we use in the gradient equations, not the ρ_β from the parabolic equations, but $\hat{\rho}_\beta$ that are at all times the solutions of the elliptic equations

$$D_\beta \Delta \hat{\rho}_\beta(x) - \kappa_\beta \hat{\rho}_\beta(x) + \sum_{\alpha=1}^{N_d} \sigma_{\beta,\alpha}^d S_{r_\alpha(t)}(x) + \sum_{\gamma=1}^{N_s} \sigma_{\beta,\gamma}^s S_{s_\gamma}(x) = 0. \quad (2.4)$$

for all $\beta = 1, \dots, N_c$, given the values of $r_\alpha(t)$. To solve the $\hat{\rho}_\beta$ simultaneously from this system we have to keep in mind that the $\sigma_{\beta,\alpha}^d$ and $\sigma_{\beta,\gamma}^s$ may depend on the $\hat{\rho}_\beta$ evaluated in the points r_α and s_γ .

For instance, the model that is used by Hentschel and Van Ooyen has three fields of chemicals, namely ρ_1 (attractant), ρ_2 (repellent) and ρ_3 (target attractant). Chemicals ρ_1 and ρ_2 are produced by the moving sources and ρ_3 by the static sources and

their system of parabolic equations is

$$\begin{aligned} \partial_t \rho_1(x, t) &= D_1 \Delta \rho_1(x, t) - \kappa_1 \rho_1(x, t) + \sum_{\alpha=1}^{N_d} \sigma_{1,\alpha}^d S_{r_\alpha(t)}(x), \\ \partial_t \rho_2(x, t) &= D_2 \Delta \rho_2(x, t) - \kappa_2 \rho_2(x, t) + \sum_{\alpha=1}^{N_d} \sigma_{2,\alpha}^d (\rho_3(r_\alpha(t), t)) S_{r_\alpha(t)}(x), \\ \partial_t \rho_3(x, t) &= D_3 \Delta \rho_3(x, t) - \kappa_3 \rho_3(x, t) + \sum_{\gamma=1}^{N_s} \sigma_{3,\gamma}^s S_{s_\gamma}(x), \end{aligned} \quad (2.5)$$

where the $\sigma_{1,\alpha}^d$ and $\sigma_{3,\gamma}^s$ are constants. Its steady state, given the values r_α at time t , can be found by setting $\partial \rho_1 / \partial t = \partial \rho_2 / \partial t = \partial \rho_3 / \partial t = 0$ and dropping the arguments t , which results in system (2.4) for this particular case. The solution of system (2.4), which we denote by $(\hat{\rho}_1, \hat{\rho}_2, \hat{\rho}_3)$, is a steady state of system (2.5) with fixed r_α and when the S_{r_α} and S_{s_γ} are smooth functions or δ -functions (in case of point sources) this state is also globally attracting in the sense that for every set of start functions (ρ_1, ρ_2, ρ_3) the solution of system (2.5) with fixed r_α tends to $(\hat{\rho}_1, \hat{\rho}_2, \hat{\rho}_3)$ in the L_∞ -norm. This is intuitively clear because of the fact that the first and last equation are independent of the equation for ρ_2 . Therefore, ρ_1 and ρ_3 approach $\hat{\rho}_1$ and $\hat{\rho}_3$, respectively, so that for $t \rightarrow \infty$, the equation for ρ_2 gets constant source terms and ρ_2 converges to $\hat{\rho}_2$.

In general, the coupling between the equations of system (2.4) by the functions $\sigma_{\beta,\alpha}^d$ and $\sigma_{\beta,\gamma}^s$, might give problems with respect to the existence of steady-state solutions as well as the global attraction of such solutions. We will consider a few different cases illustrating these problems.

Example 1. *Concerning the existence of steady-state solutions we look at a simple 1-dimensional example system*

$$\begin{aligned} \partial_t \rho_1(x, t) &= \partial_x^2 \rho_1(x, t) - \rho_1(x, t) + \sigma_1(\rho_1(r_1), \rho_2(r_1)) \delta(x - r_1), \\ \partial_t \rho_2(x, t) &= \partial_x^2 \rho_2(x, t) - \rho_2(x, t) + \sigma_2(\rho_2(r_2), \rho_1(r_2)) \delta(x - r_2), \end{aligned} \quad (2.6)$$

in which the functions ρ_1, ρ_2 are defined on \mathbb{R} and the $\delta(\cdot)$ stand for the Dirac δ -function. If both equations are in steady-state the $\hat{\rho}_1, \hat{\rho}_2$ have to satisfy

$$\begin{aligned} \hat{\rho}_1(x) &= \frac{1}{2} \sigma_1(\hat{\rho}_1(r_1), \hat{\rho}_2(r_1)) e^{-|x-r_1|}, \\ \hat{\rho}_2(x) &= \frac{1}{2} \sigma_2(\hat{\rho}_2(r_2), \hat{\rho}_1(r_2)) e^{-|x-r_2|}, \end{aligned} \quad (2.7)$$

for all $x \in \mathbb{R}$, which follows from the fact that for arbitrary $a > 0$,

$$\left. \begin{aligned} \frac{\partial^2}{\partial x^2} \mu(x) - \mu(x) + a \delta(x) &= 0, \quad \forall x \in \mathbb{R} \\ \lim_{x \rightarrow \pm\infty} \mu(x) &= 0. \end{aligned} \right\} \implies \mu(x) = \frac{1}{2} a e^{-|x|}.$$

Substitution of r_1 and r_2 in system (2.7) results in a system of four equations:

$$\begin{aligned} \hat{\rho}_1(r_1) &= \frac{1}{2}\sigma_1(\hat{\rho}_1(r_1), \hat{\rho}_2(r_1)) & \hat{\rho}_1(r_2) &= \frac{1}{2}\sigma_1(\hat{\rho}_1(r_1), \hat{\rho}_2(r_1))e^{-|r_2-r_1|} \\ \hat{\rho}_2(r_2) &= \frac{1}{2}\sigma_2(\hat{\rho}_2(r_2), \hat{\rho}_1(r_2)) & \hat{\rho}_2(r_1) &= \frac{1}{2}\sigma_2(\hat{\rho}_2(r_2), \hat{\rho}_1(r_2))e^{-|r_1-r_2|} \end{aligned} \quad (2.8)$$

in four unknowns $\hat{\rho}_i(r_j)$, ($i, j = 1, 2$). A solution of system (2.8) will yield a steady-state solution of system (2.6). However, whether a solution of (2.8) exists depends on the functions σ_1 and σ_2 and on the values of r_1 and r_2 . For instance, if $\sigma_1(x, y) = \sigma_2(x, y) = xy/((1-x)(1-y))$, then for all choices of r_1 and r_2 the only real solution of system (2.8) is $\hat{\rho}_1(r_1) = \hat{\rho}_1(r_2) = \hat{\rho}_2(r_1) = \hat{\rho}_2(r_2) = 0$ and therefore the only steady-state of (2.6) will be $\hat{\rho}_1 \equiv \hat{\rho}_2 \equiv 0$. \square

Even if a steady-state solution exists, it can be non-attracting, so that the system will never approach this state. It then doesn't make sense to use the steady-state approximation for solving the gradient equations. An example of such a system is described next.

Example 2. The system of equations is given by

$$\begin{aligned} \partial_t \rho_1(x, t) &= D\partial_x^2 \rho_1(x, t) - \rho_1(x, t) + \sigma_1(\rho_2(r_1))\delta(x - r_1), \\ \partial_t \rho_2(x, t) &= D\partial_x^2 \rho_2(x, t) - \rho_2(x, t) + \sigma_2(\rho_1(r_2))\delta(x - r_2). \end{aligned} \quad (2.9)$$

with $\rho_{1,2}$ defined on $[0, 1]$, periodic boundary conditions, $D = 0.1$, $r_1 = 0.25$, $r_2 = 0.5$ and

$$\sigma_1(x) = \frac{1}{2} \frac{x^4}{(\frac{1}{2})^4 + x^4}, \quad \sigma_2(x) = \frac{1}{2} \frac{(1-x)^4}{(\frac{1}{2})^4 + (1-x)^4}. \quad (2.10)$$

Using the same technique as in Example 1, one can show that system (2.9) has a steady-state solution. However, numerical experiments show that with the initial condition $\rho_{1,2} \equiv 0$ the system will approach a periodic motion with a period that is around 2. Some pictures of this are shown in Figure 2.1. Thus, system (2.9) has a steady-state solution which doesn't seem to be an attractor in the sense that the solution is approached for $t \rightarrow \infty$. Therefore, for this system in combination with certain gradient equations no quasi-steady-state approximation can be used. \square

In general we can say the following. System (2.4) can be solved if there is a ordering of the fields ρ_β such that the $\sigma_{\beta,\alpha}^d$ and $\sigma_{\beta,\gamma}^s$ do not depend on $\rho_{\beta_i}(r_\alpha)$ and $\rho_{\beta_i}(s_\gamma)$, for all α and all $\beta \leq \beta_i$. Solving the system can be done by solving sequentially for $\beta = 1, \dots, N_c$. We call such a system of equations sequentially dependent. Thus, if the system of parabolic equations is sequentially dependent, then there exists a steady state. In addition, this steady-state is a global attractor of the system.

If, in a mixed parabolic-gradient system, the parabolic equations are sequentially dependent we can use the quasi-steady-state approximation, i.e., the solutions $\hat{\rho}_\beta$ of system (2.4) can be used in the gradients equations to give

$$\frac{d}{dt} r_\alpha(t) = \sum_{\beta=1}^{N_c} \lambda_{\alpha,\beta} \nabla \hat{\rho}_\beta(r_\alpha(t)), \quad (2.11)$$

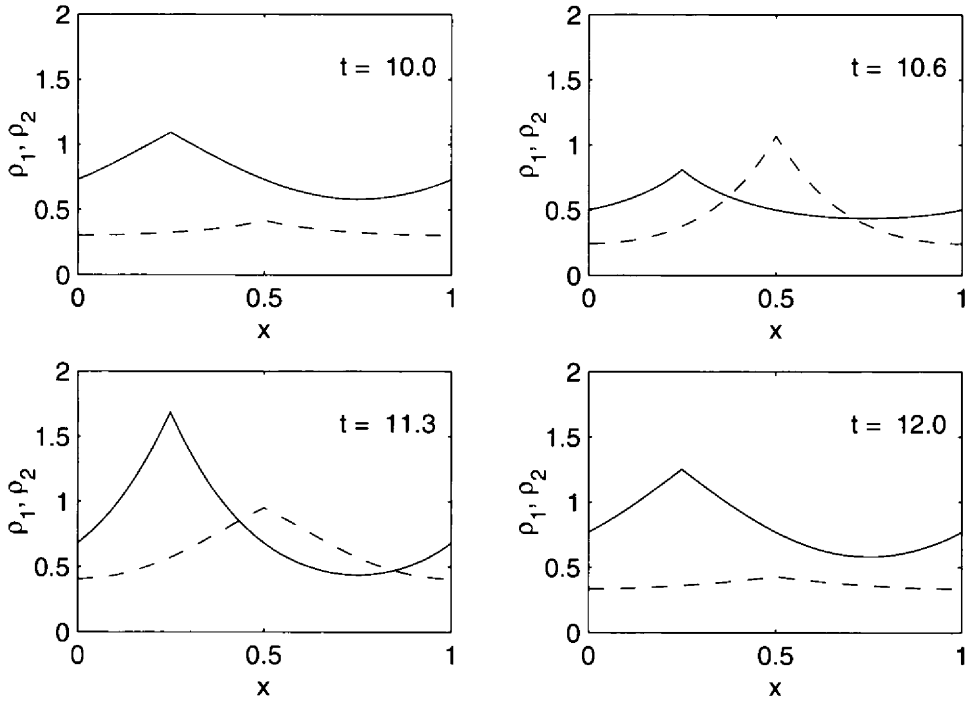


Figure 2.1: Periodic motion of system (2.9). ρ_1 (—) and ρ_2 (---).

for $\alpha = 1, \dots, N_d$. The basic idea behind this approximation is that the dynamics of the parabolic equations is much faster than the dynamics of the gradient equations. However, we will see in Section 2.5 that with the parameter ranges given in Section 2.2 this approximation can become very bad in the sense that the growing speeds r_α can be significantly different when using the quasi-steady-state approximation.

Another point is what kind of source functions to use. If point sources are being used, the system of elliptic equations (2.4) can be solved exactly for all arbitrary combinations of r_1, \dots, r_{N_d} , as is done in [21]. We will examine in the next section the use of point sources and different kinds of source functions.

2.4 Source functions

We want to consider some aspects of the used source functions S_{r_α} in the parabolic equations. From a modeling point of view, the use of point sources for making the model as simple as possible, is appealing. For instance, Hentschel and Van Ooyen [21] use point sources in their simulations of the model. However, as we will show in the next subsection, using point sources gives difficulties with respect to the smoothness and existence of solutions, especially in 2 and 3 dimensions. The alternative is to use source functions that are spread out in the neighborhood of the source position, as is discussed in Subsection 2.4.2. Although extra choices have to be made concerning the form of the source functions, the smoothness and existence of solutions are guaranteed in this case.

2.4.1 Point sources

We will start with point sources in one dimension, so that we are considering the solutions of equations of the form

$$\partial_t \rho_\beta(x, t) = D_\beta \partial_x^2 \rho_\beta(x, t) - \kappa_\beta \rho_\beta(x, t) + \sum_{\alpha=1}^{N_d} \sigma_{\beta, \alpha}^d \delta(x - r_\alpha(t)) + \sum_{\gamma=1}^{N_s} \sigma_{\beta, \gamma}^s \delta(x - s_\gamma), \quad (2.12)$$

where $x \in \mathbb{R}$. Because this equation is linear, the solution for given functions $r_\alpha(t)$ can be found by solving the same equation for the different sources separately, meaning that we have to solve equations of the form

$$\partial_t \rho(x, t) = D \partial_x^2 \rho(x, t) - \kappa \rho(x, t) + \sigma \delta(x - r(t)). \quad (2.13)$$

Smoothness

The solution of equation (2.13) can be written as

$$\rho(x, t) = \frac{\sigma e^{-\kappa t}}{\sqrt{4\pi D t}} \int_{-\infty}^{\infty} \rho_0(\xi) e^{-\frac{|x-\xi|^2}{4Dt}} d\xi + \sigma \int_0^t \frac{e^{-\kappa(t-\tau)}}{\sqrt{4\pi D(t-\tau)}} e^{-\frac{|x-r(\tau)|^2}{4D(t-\tau)}} d\tau, \quad (2.14)$$

for all $t > 0$ and $\rho_0(x) = \rho(x, 0)$, which can be easily found by using Fourier analysis. Given $\rho_0 \in L^p$ ($1 \leq p \leq \infty$), it can be proven that ρ is C^∞ at all points (x, t) with $t > 0$ and $x \neq r(t)$. A proof of this for an analogous equation can be found in [19]. At points $(r(t), t)$, ρ may not be differentiable with respect to x .

Therefore, equation (2.12) has solutions of ρ_β that are smooth, except in points $(r_\alpha(t), t)$ and (s_γ, t) , where the sources are located, as can also be seen in Figure 2.1. For the analogous equations in two and three dimensions the same holds, as can be proven in exactly the same way as in one dimension.

In the complete 1-dimensional mixed parabolic-gradient system, the functions $r(t)$ satisfy gradient equations (2.1). These equations contain terms $\lambda_{\alpha,\beta} \partial_x \rho_\beta(r_\alpha(t), t)$, so that there can occur something which we will call self-interaction. This occurs if there are α and β with $\sigma_{\beta,\alpha}^d \neq 0$ and $\lambda_{\alpha,\beta} \neq 0$, meaning that source α produces ρ_β and the dynamics of r_α is influenced by ρ_β . In other words, there are sources sensing fields which they help produce themselves.

Ill-definedness.

If self-interaction occurs with respect to source α and field β , then the system is not well-defined, because the solution $\rho_\beta(\cdot, t)$ of equation (2.12) is not differentiable at $r_\alpha(t)$, while the term $\partial_x \rho_\beta(r_\alpha(t), t)$ is used in the gradient equation for $r_\alpha(t)$.

In the 2- and 3-dimensional case the situation is similar. The solution ρ_β is everywhere C^∞ , except at the location of the point sources α with $\sigma_{\beta,\alpha} \neq 0$, where it is even singular. Again, this will result in ill-definedness of the problem if self-interaction occurs, in the same way as in the 1-dimensional case. Therefore, if self-interaction occurs, it is impossible to work with point sources.

By defining a generalized gradient $\tilde{\nabla} f(x) = \lim_{h \downarrow 0} (f(x+h) - f(x-h))/(2h)$, we can solve this problem in the 1-dimensional case, because $\tilde{\nabla} \rho_\beta$ exists at the locations of the sources. Further, if f is smooth at x then $\tilde{\nabla} f(x) = \nabla f(x)$. However, it seems that there is not a similar possibility in 2 dimensions due to the fact that the ρ_β are singular at the source locations, which is not the case in 1 dimension. In this case we would like to define a generalized gradient by means of

$$\tilde{\nabla} f(r) = \lim_{h \downarrow 0} \frac{1}{\pi h^3} \int_{B_h(r)} f(x)(x-r) dx, \tag{2.15}$$

with $B_h(r) = \{x \in \mathbb{R}^2 \mid |x-r| = h\}$, for which in case of a smooth function f we can write

$$\begin{aligned} \tilde{\nabla} f(r) &= \lim_{h \downarrow 0} \frac{1}{\pi h^3} \int_{B_h(r)} \left\{ f(r) + \nabla f(r) \cdot (x-r) + \mathcal{O}(h^2) \right\} (x-r) dx \\ &= \lim_{h \downarrow 0} \int_{B_h(r)} \frac{1}{\pi h^3} \left\{ \nabla f(r) \cdot (x-r) \right\} (x-r) dx + \mathcal{O}(h) = \nabla f(r). \end{aligned}$$

However, if we try to use our generalized gradient in two dimensions on a moving source with a constant speed vector \mathbf{v} , yielding equation

$$\partial_t \rho(\mathbf{x}, t) = D \Delta \rho(\mathbf{x}, t) - \kappa \rho(\mathbf{x}, t) + \sigma \delta(\mathbf{x} - r(t)), \tag{2.16}$$

with $r(t) = t\mathbf{v} + r_0$, then for large t the solution approaches

$$\rho(\mathbf{x}, t) = \frac{\sigma}{2\pi D} \exp\left(-\frac{1}{2D}\mathbf{v} \cdot (\mathbf{x} - r(t))\right) K_0\left(\frac{\sqrt{|\mathbf{v}|^2 + 4D\kappa}}{2D}|\mathbf{x} - r(t)|\right). \quad (2.17)$$

This can be found by substituting a moving profile solution $\rho(\mathbf{x}, t) = \hat{\rho}(\mathbf{x} - t\mathbf{v})$ into equation (2.16). Here, the function K_0 is the modified Bessel function of the second kind, for which, for small x , $K_0(x) = -\gamma_E - \ln(x/2) + \mathcal{O}(x)$, where γ_E is Euler's constant. We then can write for $\rho(\mathbf{x}, t)$ in the neighborhood of $r(t)$

$$\begin{aligned} & \rho(r + \xi, t) \\ &= \frac{\sigma}{2\pi D} \left\{ 1 - \frac{1}{2D}\mathbf{v} \cdot \xi + \mathcal{O}(|\xi|^2) \right\} \left\{ -\ln\left(e^{\gamma_E} \frac{\sqrt{|\mathbf{v}|^2 + 4D\kappa}}{4D}|\xi|\right) + \mathcal{O}(|\xi|) \right\} \\ &= A_1 \ln(A_2|\xi|) + A_3(\mathbf{v} \cdot \xi) \ln(A_2|\xi|) + \mathcal{O}(|\xi|), \end{aligned}$$

where A_1 , A_2 and A_3 are real constants. If this expression is substituted in the definition of $\bar{\nabla}\rho$, then this yields

$$\frac{1}{\pi h^3} \int_{B_h(0)} \rho(r + \xi, t) \xi \, d\xi = A_3 \ln(A_2 h) \mathbf{v} + \mathcal{O}(h^0),$$

and therefore the limit in the definition of the generalized gradient does not exist at the location of the source $r(t)$. In three dimensions the same effect occurs and again the analogously defined generalized gradient does not exist at the location of the source.

Making use of some kind of generalized gradient doesn't seem to make it possible to combine self-interaction with point sources in the 2- and 3-dimensional case. Because self-interaction is an important feature of the model, we will disregard the use of point sources and concentrate in the next section on sources that are spread-out in space.

2.4.2 Spread-out sources

If using spread-out sources, certain choices have to be made regarding the form of the source function $S_r: \Omega \rightarrow \mathbb{R}$. In general, we will define the function S_{r_α} by $S_{r_\alpha}(x) = S(|x - r_\alpha|)$, where $S: \mathbb{R}^+ \rightarrow \mathbb{R}$. Further we would like to have in most cases a compactly supported source function, meaning that $\text{supp}(S) = [0, \ell]$ for some $\ell > 0$. We assume also that S is non-increasing, piecewise smooth and $\int_\Omega S_{r_\alpha}(\xi) \, d\xi = 1$. In an analogous way we will define S_{s_α} .

The most simple source function we can think of is the one defined by

$$S(x) = \begin{cases} C_{dim} \cdot x & x \leq \ell \\ 0 & x > \ell \end{cases}, \quad C_1 = \frac{1}{2\ell}, \quad C_2 = \frac{1}{\pi\ell^2}.$$

This function already shows an important feature of spread-out sources. Namely, the source functions S_{r_α} defined using the S above, will have discontinuities at points

$x \in \Omega$ with $|x - r_\alpha| = \ell$. This can result in higher order¹ discontinuities at the same points in the solutions ρ_β of the parabolic equations.

Example 3. As an example in one dimension we consider solutions of the equation

$$\partial_t \rho(x, t) = D \partial_x^2 \rho(x, t) - \kappa \rho(x, t) + \sigma S(|x - vt|), \quad (2.18)$$

on \mathbb{R} , which describes the dynamics of a concentration field ρ caused by a moving source with location $r(t) = vt$. One of the solutions is a translating profile $\rho(x, t) = \hat{\rho}(x - vt)$, that is shown for $t = 0$ in the left picture of Figure 2.2. At $t = 0$, the source is located at the origin and moves to the right with speed v . Here, we used $v = 0.5$, $D = 0.1$, $\kappa = 1$ and $\ell = 0.5$. In the middle and right picture of the figure $\partial_x \hat{\rho}$ and $\partial_x^2 \hat{\rho}$ are shown, respectively. We can see that the second order derivative is discontinuous for $|x| = \ell$. \square

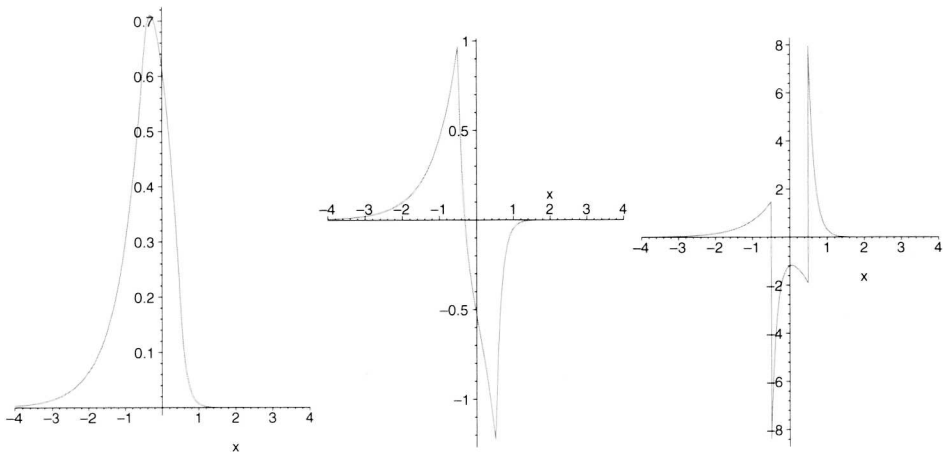


Figure 2.2: Moving source profile $\hat{\rho}$ in one dimension and its derivatives $\partial_x \hat{\rho}$ and $\partial_x^2 \hat{\rho}$.


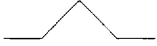


In general, solutions of

$$\partial_t \rho(x, t) = D \partial_x^2 \rho(x, t) - \kappa \rho(x, t) + f(x, t), \quad (2.19)$$

with $f(x, t)$ piecewise smooth and having discontinuities of order $r_k(t)$ at points $x_k(t)$, are smooth everywhere, except for the points $x_k(t)$, where they have discontinuities of order $r_k(t) + 2$. Therefore we can choose other source functions to guarantee a certain smoothness, some of which are shown Table 2.2.

¹With a discontinuity of order r in x we mean that the r^{th} -order derivative is discontinuous in x while the $(r - 1)^{\text{th}}$ derivative is continuous. By a discontinuity order of ∞ we mean that all derivatives in x are continuous.

Table 2.2: Source functions and orders of discontinuity.

$S(x)$	C_1	C_2	profile	disc. order of ρ at r	disc. order of ρ at $B(r; \ell)$
C_{dim}	$\frac{2}{\ell}$	$\frac{1}{\pi\ell^2}$		∞	2
$C_{dim}(\ell - x)$	$\frac{1}{\ell^2}$	$\frac{3}{\pi\ell^3}$		3	3
$C_{dim}(x - \ell)^2(x - \frac{1}{2}\ell)$	$\frac{2}{\ell^4}$	$\frac{20}{3\pi\ell^5}$		5	4
$C_{dim} \cos^2(\frac{\pi x}{2\ell})$	$\frac{1}{\ell}$	$\frac{2\pi}{(\pi^2 - 4)\ell^2}$		∞	4

Another interesting feature of the moving profile of Example 3 is the fact that its maximum is not attained at the center of the source, although it is very close to it: namely, the maximum is attained in $\text{supp}(S_{r(t)})$, which consists of all x with $|x - r(t)| < \ell$. We can prove that this is true for moving profiles in general, by noticing that we must have for $\hat{\rho}$, with $\rho(\mathbf{x}, t) = \hat{\rho}(\mathbf{x} - t\mathbf{v})$,

$$D\Delta\hat{\rho}(\mathbf{x}) - \kappa\hat{\rho}(\mathbf{x}) + \mathbf{v} \cdot \nabla\hat{\rho}(\mathbf{x}) = 0, \quad (2.20)$$

for all $\mathbf{x} \in \Omega \setminus \overline{\text{supp}(S_0)}$, where \overline{A} is the closure of an arbitrary set $A \in \Omega$. If $\hat{\rho}$ would have a maximum at $\mathbf{x} \in \Omega \setminus \overline{\text{supp}(S_0)}$, then $\nabla\hat{\rho}(\mathbf{x}) = 0$ and $\Delta\hat{\rho}(\mathbf{x}) < 0$, because all eigenvalues of the Hessian matrix of $\hat{\rho}$ evaluated in \mathbf{x} are negative and $\Delta\hat{\rho}(\mathbf{x})$ is the sum of these eigenvalues. However, equation (2.20) yields, by $\hat{\rho}(\mathbf{x}) > 0$ and $\nabla\hat{\rho}(\mathbf{x}) = 0$, that $\Delta\hat{\rho}(\mathbf{x}) > 0$, giving a contradiction. Therefore a maximum of $\hat{\rho}$ lies in $\text{supp}(S_0)$ and thus for the maximum of ρ we have that $|\mathbf{x}_{max} - r(t)| \leq \ell$.

2.5 Self-interaction

It will often occur that the gradient equation of a source contains gradients of one of the fields ρ_β that it produces, which is a property that we called self-interaction in Subsection 2.4.1. In this section we will see that self-interaction has a diminishing effect on the speed of a moving source in case that the self-interaction field is an attractant for the source. If the self-interaction field is a repellent, then the speed of the moving source will be greater. Besides this we will examine how the width of the sources influences the self-interaction.

In Example 3, in case of block source functions, the gradient of the field ρ that the source produces, evaluated at the source position, is equal to

$$\partial_x \hat{\rho}(vt) = -\frac{\sigma}{\ell\sqrt{v^2 + 4D\kappa}} \exp\left(-\frac{\sqrt{v^2 + 4D\kappa}}{2D}\ell\right) \sinh\left(\frac{v}{2D}\ell\right). \quad (2.21)$$

The gradient $\partial_x \hat{\rho}(vt)$ is negative (positive), for positive (negative) v , and is decreasing (increasing) with ℓ , as can be proven by noticing that $\sqrt{v^2 + 4D\kappa}/(2D)$ is always greater than $|v|/(2D)$. This means that for decreasing source width ℓ , the source will sense the fields that it produces in an increasing way. For $\ell \downarrow 0$ this will result in

$$\lim_{\ell \downarrow 0} \partial_x \hat{\rho}(vt) = -\frac{\sigma v}{2D\sqrt{v^2 + 4D\kappa}}. \quad (2.22)$$

To get an idea of how big the influence can be on the time derivatives \dot{r} in the gradient equations, we consider an example.

Example 4. *In this example we consider a moving source (position r) and a static source (position s) on \mathbb{R} , secreting a substance ρ to which the moving source is attracted, yielding equations*

$$\partial_t \rho(x, t) = D\partial_x^2 \rho(x, t) - \kappa \rho(x, t) + \sigma S(|x - r(t)|) + \sigma S(|x - s|). \quad (2.23)$$

$$\dot{r}(t) = \lambda \partial_x \rho(r(t), t). \quad (2.24)$$

We can write the solution of equation (2.23) as $\rho(x, t) = \rho_s(x) + \rho_r(x, t)$, where ρ_s and ρ_r satisfy

$$0 = D\partial_x^2 \rho_s(x) - \kappa \rho_s(x) + \sigma S(|x - s|),$$

$$\partial_t \rho_r(x, t) = D\partial_x^2 \rho_r(x, t) - \kappa \rho_r(x, t) + \sigma S(|x - r(t)|),$$

so that for equation (2.24) we have

$$\dot{r}(t) = \lambda \partial_x \rho_s(r(t)) + \lambda \partial_x \rho_r(r(t), t). \quad (2.25)$$

In a quasi steady-state approximation the term $\partial_x \rho_r(r(t), t)$ will vanish, however in this case it is approximately equal to the expression in equation (2.21), which yields after substitution and developing a Taylor series with respect to \dot{r} ,

$$\dot{r} \approx v + \xi \dot{r} + \eta \dot{r}^3 + \mathcal{O}(\dot{r}^4), \quad (2.26)$$

with $v = \lambda \partial_x \rho_s(r)$ and

$$\xi = -\left(\frac{\lambda}{4} \frac{\sigma}{D^2 \mu} e^{\mu \ell}\right), \quad \eta = -\left(\frac{\lambda}{96} \frac{\sigma(\mu^2 \ell^2 + 3\mu \ell - 3)}{D^4 \mu^3} e^{\mu \ell}\right), \quad \mu = \sqrt{\frac{\kappa}{D}}.$$

Solving the approximate equation (2.26) yields then

$$\dot{r} \approx \frac{1}{1 - \xi} v + \frac{\eta}{(1 - \xi)^4} v^3 + \mathcal{O}(v^4). \quad (2.27)$$

By using parameter values $D = 1.0 \cdot 10^{-4}$, $\kappa = 1.0 \cdot 10^{-4}$, $\sigma = 1.0 \cdot 10^{-3}$, $\ell = 1.0 \cdot 10^{-2}$, and $\lambda = 1.0 \cdot 10^{-5}$, where we have that $0 \leq v \leq 5.0 \cdot 10^{-5}$, $(1 - \xi)^{-1} \approx 0.57$ and $\eta(1 - \xi)^{-4} \approx 1.0 \cdot 10^6$, this results in $\dot{r} \approx (0.57) \cdot \lambda \partial_x \rho_s(r)$.

Without self-interaction, as in the case with QSSA, then $\dot{r} = \lambda \partial_x \rho_s(r)$. Hence, according to this estimation, the self-interaction causes a decrease in source speed of 43%. It has to be noted that the quality of this estimation depends on the acceleration of the point source. The used equation (2.21) is valid for constant $v (= \dot{r})$ and $t \rightarrow \infty$ and therefore, if v does not change too rapidly, the convergence of the measured gradient to equation (2.21) can be faster than the speed with which v changes. \square

2.5.1 Self-interaction in two dimensions

We next want to examine the effect of the width of source functions on the self-interaction in two dimensions. For this, we define $S^\ell: \mathbb{R}^2 \rightarrow \mathbb{R}$, by $S^\ell(\mathbf{x}) = \frac{1}{\ell^2} S(\frac{1}{\ell}|\mathbf{x}|)$, with $\text{supp } S = [0, 1]$, so that

$$\int_{B_\ell(0)} S^\ell(\mathbf{x}) d\mathbf{x} = \int_{B_1(0)} S(\mathbf{x}) d\mathbf{x}, \quad (2.28)$$

where $B_\ell(0) = \{\mathbf{x} \in \mathbb{R}^2 \mid |\mathbf{x}| \leq \ell\}$. Again we consider the moving profile for a source moving with a constant speed \mathbf{v} , i.e. $\rho(\mathbf{x}, t) = \hat{\rho}(\mathbf{x} - t\mathbf{v})$. The equation for the field is given by

$$\partial_t \rho(\mathbf{x}, t) = D \Delta \rho(\mathbf{x}, t) - \kappa \rho(\mathbf{x}, t) + \sigma S^\ell(\mathbf{x} - t\mathbf{v}), \quad (2.29)$$

for which the moving profile will be determined by the equation

$$D \Delta \hat{\rho} + \mathbf{v} \cdot \nabla \hat{\rho} - \kappa \hat{\rho} + \sigma S^\ell = 0, \quad (2.30)$$

where $\hat{\rho}$ and S^ℓ are functions of \mathbf{x} only. We will solve this equation by using a Green's function $\hat{\rho}_G$, which is the solution of equation (2.30) with a Dirac distribution δ instead of the function S^ℓ . For these functions we have the expression

$$\hat{\rho}_G(\mathbf{x}) = A_1 e^{(-A_2 \mathbf{v} \cdot \mathbf{x})} K_0(A_3 |\mathbf{x}|) \quad (2.31)$$

$$A_1 = \frac{\sigma}{2\pi D}, \quad A_2 = \frac{1}{2D}, \quad A_3 = \frac{\sqrt{|\mathbf{v}|^2 + 4D\kappa}}{2D} \quad (2.32)$$

and the solution of equation (2.30) is then equal to

$$\hat{\rho}(\mathbf{x}) = \int_{\mathbb{R}^2} \hat{\rho}_G(\mathbf{x} - \xi) S^\ell(\xi) d\xi. \quad (2.33)$$

To consider the self-interaction, we have to calculate $\nabla \rho(t\mathbf{v}, t) = \nabla \hat{\rho}(0)$. Because of the fact that S^ℓ has support $B_\ell(0)$ and is symmetric at 0, we can write

$$\nabla \hat{\rho}(0) = \int_{B_\ell(0)} \nabla \hat{\rho}_G(\xi) S^\ell(\xi) d\xi.$$

The series expansion of $\nabla\hat{\rho}_G$ in the neighbourhood of 0 is equal to

$$\begin{aligned}\nabla\hat{\rho}_G(\mathbf{x}) &= -\left\{A_2A_1e^{(-A_2\mathbf{v}\cdot\mathbf{x})}K_0(A_3|\mathbf{x}|)\right\}\mathbf{v} - \left\{\frac{A_3}{|\mathbf{x}|}A_1e^{(-A_2\mathbf{v}\cdot\mathbf{x})}K_1(A_3|\mathbf{x}|)\right\}\mathbf{x} \\ &= \left\{A_1A_2\ln(e^{\gamma E}A_3|\mathbf{x}|) - A_1A_2^2(\mathbf{v}\cdot\mathbf{x})\ln(e^{\gamma E}A_3|\mathbf{x}|)\right\}\mathbf{v} \\ &\quad + \left\{-\frac{A_1}{|\mathbf{x}|^2} + A_1A_2\frac{\mathbf{v}\cdot\mathbf{x}}{|\mathbf{x}|^2}\right\}\mathbf{x} + \mathcal{O}(|\mathbf{x}|),\end{aligned}$$

so that we have for $\nabla\hat{\rho}(0)$,

$$\nabla\hat{\rho}(0) = A_1A_2\left(\int_{B_1(0)}\ln\left(e^{\gamma E+\frac{1}{2}}A_3|\mathbf{x}|\right)S(\mathbf{x})d\mathbf{x} + \ln(\ell)\right)\mathbf{v} + \mathcal{O}(\ell). \quad (2.34)$$

With this expression of the gradient at the position of the source of a moving profile solution we can estimate the effect of self-interaction in case of the general mixed parabolic-gradient system (2.1) and (2.2). Selecting a moving source r_α and a field ρ_β for which we have self-interaction, we split the field ρ into two parts; one part ρ_r , produced by the source itself, and the other part ρ_e , produced by other sources, giving $\mathbf{v}_e = \lambda\nabla\rho_e(r(t), t)$. Here we dropped the subscripts α and β for convenience. Using the first term in the equation (2.34), we can write $\nabla\rho_r(r) \approx \nabla\hat{\rho}(0) \approx A_4(\ell)\mathbf{v}$. For the speed of the source we then have by the gradient equation $\dot{r} \approx \lambda A_4(\ell)\dot{r} + \mathbf{v}_e$, yielding $\dot{r} \approx (1 - \lambda A_4(\ell))^{-1}\mathbf{v}_e$. With parameter choices of Example 4 we then get $\dot{r} \approx (0.76)\mathbf{v}_e$.

Clearly, the effect of self-interaction can become infinitely large for small source widths ℓ . The source width is therefore a critical parameter of the system. Further, the sign of λ determines whether the self-interaction field acts as an attractant or as a repellent. We have shown an example of an attractant field. For $\lambda < 0$, hence a repellent field, the speed of the moving source will be greater instead of smaller.

2.6 Numerical tests

In the previous sections we found by analytic means some properties of the mixed parabolic-gradient systems. In this section we will do some numerical tests to illustrate some of these findings. For this, we use a simple numerical method, which is first order accurate in time and second order accurate in space and serves for showing the effects of self-interaction.

We will concentrate on a 1-dimensional example system (the system from Example 4). The system is

$$\partial_t\rho(x, t) = D\partial_x^2\rho(x, t) - \kappa\rho(x, t) + \sigma_r S(|x - r(t)|) + \sigma_s S(|x - s|), \quad (2.35)$$

$$\dot{r}(t) = \lambda\partial_x\rho(r(t), t), \quad (2.36)$$

which is the most simple system that shows self-interaction. The moving source (position $r(t)$) and the static source (position s) both produce ρ , and the moving

source is growing to higher concentrations of ρ . Again, $\text{supp } S = [0, \ell]$, with S one of the functions from Table 2.2. Further we will assume that the domain is $[0, 1]$ and we impose periodic boundary conditions.

2.6.1 Numerical method

For discretization of the parabolic equation (2.35) we will use the backward-time central-space scheme,

$$\frac{v_m^{n+1} - v_m^n}{k} = D \frac{v_{m+1}^{n+1} - 2v_m^{n+1} + v_{m-1}^{n+1}}{h^2} - \kappa v_m^{n+1} + \sigma_r S(|x_m - r^n|) + \sigma_s S(|x_m - s|) \quad (2.37)$$

on an evenly spaced grid $0 = x_0, \dots, x_M = 1$, with gridsize h in space and step size k in time. Because of the periodic boundary conditions we can work with vectors $\mathbf{v}^n = (v_1^n, \dots, v_M^n)^T$ of length M , such that $v_m^n \approx \rho(x_m, t_n)$. If we denote $S_m(r^n) = \sigma_r S(|x_m - r^n|) + \sigma_s S(|x_m - s|)$, such that $\mathbf{S}(r^n) \in \mathbb{R}^M$, then we can write this scheme as

$$A\mathbf{v}^{n+1} = \mathbf{v}^n + k\mathbf{S}(r^n), \quad (2.38)$$

where A is a periodic tridiagonal matrix. If we denote the projection of the exact solution $\rho(x, t)$ on the grid by $\boldsymbol{\rho}(t)$, then substitution of this solution into (2.38) yields

$$A\boldsymbol{\rho}(t_{n+1}) = \boldsymbol{\rho}(t_n) + k\mathbf{S}(r(t_n)) + k\mathcal{O}(k + h^2), \quad (2.39)$$

and therefore the discretization is first order consistent in time and second order consistent in space. Further, this scheme is unconditionally stable [46] for a given function $r(t)$.

The path $r(t)$ of the moving source will be approximated at discrete time points t_n ($r^n \approx r(t_n)$). For the discretization of the gradient equation (2.36), we need to approximate the gradient at r^n , which is not necessarily a grid point x_m . For this, we need a numerical gradient function $P_{\nabla}: \mathbb{R}^M \times [0, 1] \rightarrow \mathbb{R}$, such that if an arbitrary, smooth function $f: [0, 1] \rightarrow \mathbb{R}$ is projected on the grid $\mathbf{x} \in \mathbb{R}^M$, yielding $\mathbf{f} \in \mathbb{R}^M$, then $P_{\nabla}(\mathbf{f}, r) \approx \partial_x f(r)$. Then we use forward Euler to calculate r^{n+1} from r^n , giving

$$r^{n+1} = r^n + k\lambda P_{\nabla}(\mathbf{v}^n, r^n). \quad (2.40)$$

If we assume that $P_{\nabla}(\mathbf{f}, r) = \partial_x f(r) + \mathcal{O}(h^p)$ for arbitrary, smooth f , then we can substitute the exact solution $r(t)$ into (2.40) to obtain

$$r(t_{n+1}) = r(t_n) + k\lambda P_{\nabla}(\boldsymbol{\rho}(t_n), r(t_n)) + k\mathcal{O}(k + h^p), \quad (2.41)$$

making scheme (2.40) first order accurate in time and p^{th} -order accurate in space.

Our time stepping process now consists of two stages: equation (2.38) together with equation (2.40). But we still need to define the numerical gradient function P_{∇} . The most straightforward way to define a numerical gradient function is to define

$$P_{\nabla}(\mathbf{f}, x_i + \theta h) = \frac{1}{h}(f_{i+1} - f_i), \quad (2.42)$$

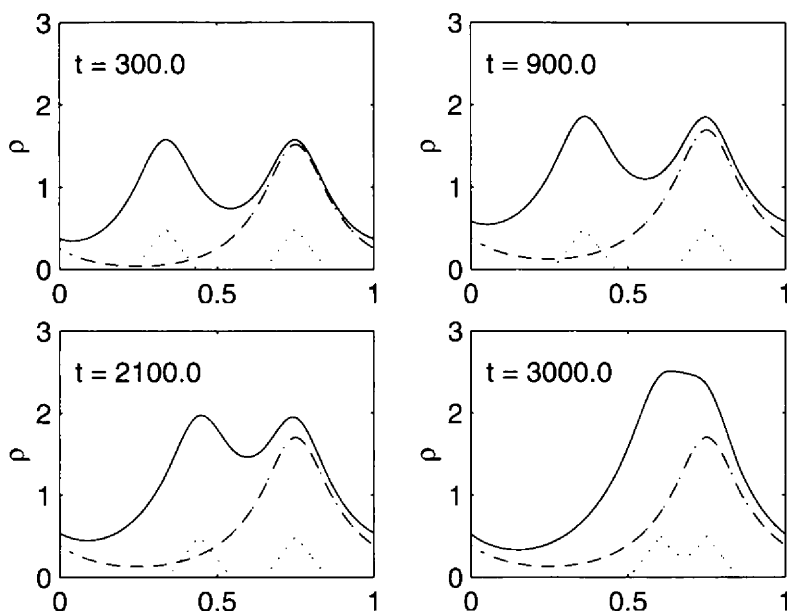


Figure 2.3: Solution of system (2.35)-(2.36) at times $t = 300, 900, 2100, 3000$. Sources (\cdots), ρ_s ($-\cdot-$) and ρ ($-$).

with the unique $\theta \in [0, 1)$ and x_i such that $r = x_i + \theta h$, which gives $P_{\nabla}(\mathbf{f}, r) = \partial_x f(r) + \mathcal{O}(h)$, for arbitrary, smooth functions f . However, it is easy to build higher order gradient functions by using more grid points. For example,

$$P_{\nabla}(\mathbf{f}, x_i + \theta h) = \frac{1}{h} \left(\frac{1}{2} (-f_{i-1} + 3f_i - 3f_{i+1} + f_{i+2})\theta^2 + (f_{i-1} - 2f_i + f_{i+1})\theta + \frac{1}{6} (-2f_{i-1} - 3f_i + 6f_{i+1} - f_{i+2}) \right), \quad (2.43)$$

for which $P_{\nabla}(\mathbf{f}, r) = \partial_x f(r) + \mathcal{O}(h^3)$ for arbitrary, smooth functions f , which is the highest order numerical gradient function possible using four grid points. We now will show some results of an example calculation using equations (2.38), (2.40) and (2.43).

Example 5. We use the parameter values $D = 1.0 \cdot 10^{-4}$, $\kappa = 1.0 \cdot 10^{-4}$, $\sigma = 3.0 \cdot 10^{-3}$, $\lambda = 1.0 \cdot 10^{-4}$ and the cone-like source functions with $\ell = 0.1$. Further we take $s = 3/4$ and as initial values $r(0) = 1/3$ and $\rho(x, 0) = 0$, for all $x \in [0, 1]$ and we will integrate to $t = 3000$. For a calculation with 2000 grid points, both in the x and t -direction ($h = 0.5 \cdot 10^{-3}$ and $k = 1.5$), the results are shown in Figure 2.3. These grid sizes are sufficiently small to approximate exact solutions up to plotting accuracy.

It can be seen that the source at position $r(t)$ moves toward the source at position s . The dash-dotted line shows a ρ -field that is produced solely by the source at s , which we called ρ_s in Example 4. This field isn't used in the calculation, but is shown for illustration purposes. The solid line shows the ρ -field that is used in the calculation. It is the sum of the two fields excreted by the sources. The dotted line displays the scaled source profile functions. They are scaled down by a factor 20 to make them nicely fit into the picture.

In Section 2.5 we made an estimation of the diminishing effect on the moving speed of a source in case of block source functions and for ρ defined on the whole of \mathbb{R} . For our example case, where we have cone source functions and the domain is $[0, 1]$ with periodic boundary conditions, we can do a similar calculation. We then get for ξ in equation (2.27),

$$\xi = \frac{\lambda\sigma}{4D^2\mu} \frac{(1-\ell)\sinh(\mu\ell) - \ell\sinh(\mu(1-\ell))}{\ell(e^\mu - 1)(e^{-\mu} - 1)}.$$

With our choices of parameters, we have $\xi = -0.511$ and equation (2.27) gives $\dot{r} \approx (0.66) \cdot \lambda \partial_x \rho_s(r)$. In the left graph of Figure 2.4 the gradients $\partial_x \rho_s(r(t))$ (dash-dotted line) and $\partial_x \rho(r(t), t)$ (solid line) are shown. Clearly, $\partial_x \rho(r(t), t)$ is much smaller than $\partial_x \rho_s(r(t))$ due to the self-interaction. According to our estimation we should have $\partial_x \rho(r(t), t) / \partial_x \rho_s(r(t)) \approx 0.66$. This ratio is depicted in the right graph. We see that the ratio is a little less than the estimation we made.

Two things might explain this. First, the estimation is based on a moving profile solution moving with constant speed \dot{r} . The fact that \dot{r} is not constant, but increasing, might give some differences. Second, from the gradient of the moving profile solution we only take the first order term in \dot{r} in our estimation. For higher speeds, higher order terms can come into play and they then have to be accounted for.

We see that the self-interaction causes a decrease of about 30% – 40% in moving speed of the source here. If QSSA is used the self-interaction is automatically neglected, because concentration fields in steady-state do have a vanishing gradient at the location of the source. Therefore, sources seem to move faster than they really do with QSSA in this particular problem. In Figure 2.5 the QSSA solution for $r(t)$ (dash-dotted) is shown next to the full integration solution of $r(t)$ just calculated. Clearly, in the QSSA solution the moving source reaches the static source too early.

Using QSSA, we turned the parabolic equation into an elliptic equation by putting $\partial_t \rho = 0$. This equation can be solved analytically and its solution, which depends on $r(t)$ can be used in the gradient equation, resulting in an autonomous ODE. We solved this equation numerically using the classical Runge-Kutta 4th-order integration scheme. \square

2.7 Conclusions

In this chapter we examined a mixed parabolic-gradient system, which is a prototype for such systems arising in neurobiology, where they act as a model for the axonal

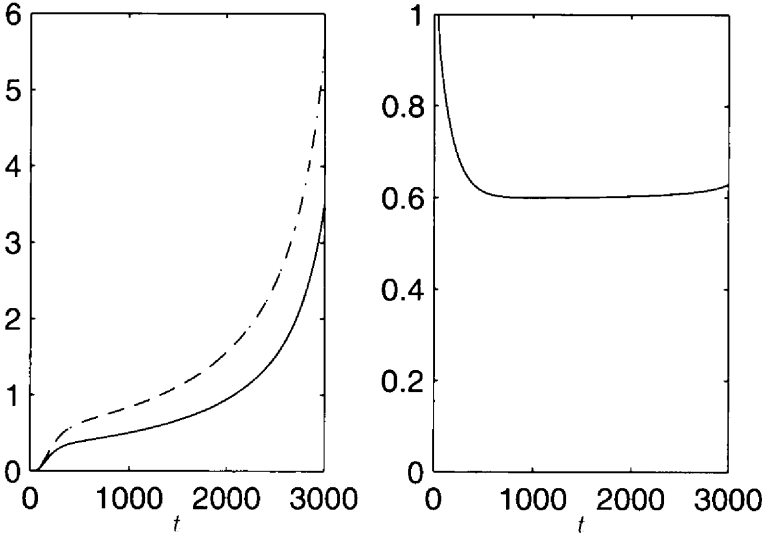


Figure 2.4: (Left) $\partial_x \rho(r(t), t)$ (—) and $\partial_x \rho_s(r(t))$ (- · -) against time. (Right) Ratio of the gradients against time.

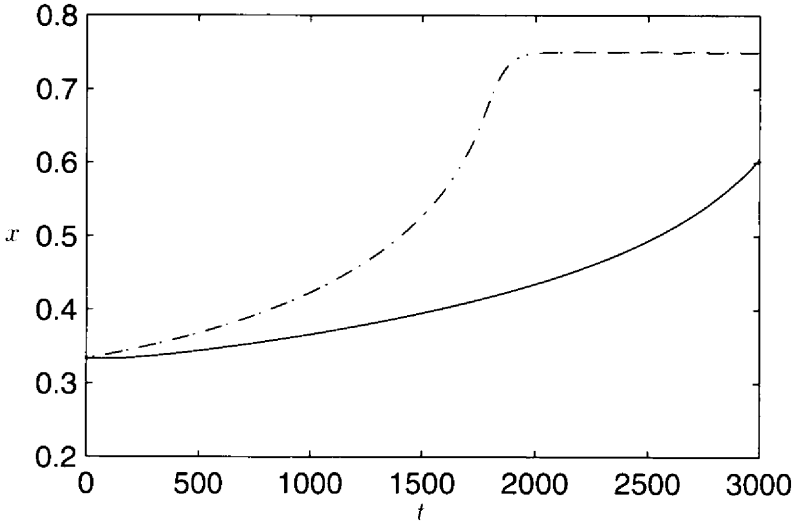


Figure 2.5: Location of the moving source against time computed with QSSA (- · -) and with numerical integration of the full parabolic-gradient system (—).

growth out of neurons. The long term goal is to develop efficient numerical methods for solving such equation systems. Here, we tried to get a better understanding of the equations by analytical and numerical means.

We found that, with the parameters in the estimated ranges, putting the parabolic equations in steady-state gives very different results. Although some reasons to justify the quasi-steady-state approximation exist, see [21], this approximation can give moving speeds of the sources that are significantly wrong in case self-interaction occurs.

A way to estimate the self-interaction effect is found and by using this, it should be possible to give an indication of the quality of the quasi-steady-state approximation in particular cases. In addition, we found as a rule of thumb that decreasing the source width will give a greater self-interaction effect. In one dimension this effect seems to be bounded for decreasing source widths, but in two and three dimensions this effect can become unbounded, resulting in source speeds approaching zero or becoming very large.

This brings us to the use of point sources. With point sources the solutions of the parabolic equations are smooth everywhere except for the locations of the sources. If self-interaction occurs, gradients have to be taken at these locations, making the combination of self-interaction and point sources impossible. In one dimension we can work around this by redefining the gradients, but in two and three dimensions this seems not to be possible. As an alternative, sources that are spread out in space can be used, but then care has to be taken on the smoothness of solutions of the concentration fields.

In doing the numerical tests we found that the number of grid points needed to reach good accuracy is very high, even for the simple problem we used. In future research we will focus on this aspect and search for better ways to discretize these equations.

Other points of interest are how the model can be extended to make it more realistic. For instance, the model relates certain mechanisms (e.g. sensing gradients) to global behaviour (e.g. bundling). However, the dynamics can be such that at a certain moment of time the mechanisms are not realistic anymore and other mechanisms should take over.

Acknowledgments

The present work is a result of collaboration with the Netherlands Institute for Brain Research (NIBR). The author would like to thank Jaap van Pelt and Arjen van Ooyen of the NIBR for discussions and useful insight.

Chapter 3

Domain definition with Bézier curves

In this chapter we will consider the selection of nodes on the boundaries of the domains of the PDEs, where the boundaries are given as a combination of Bézier curves. Such a selection is needed for the spatial discretization. We will start with the definition of Bézier curves and Bézier paths and discuss some of their properties.

3.1 Bézier curves and paths

Given $n + 1$ control points $\mathbf{p}_i \in \mathbb{R}^2$, $i = 0, \dots, n$, a Bézier curve $C: [0, 1] \rightarrow \mathbb{R}^2$ of degree n is defined by

$$\mathbf{c}(t) = \sum_{i=0}^n B_{n,i}(t) \mathbf{p}_i, \quad \text{with } B_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad (3.1)$$

for $t \in [0, 1]$, see [3]. If $n = 1$ this results in a straight line between the points \mathbf{p}_0 and \mathbf{p}_1 . In general \mathbf{c} is a curve which has as its end points \mathbf{p}_0 and \mathbf{p}_n and is entirely contained in the convex hull of the set of control points $\mathbf{p}_0, \dots, \mathbf{p}_n$, see [3]. The first and second order derivative of a Bézier curve \mathbf{c} are given by

$$\mathbf{c}'(t) = \sum_{i=0}^{n-1} n B_{n-1,i}(t) (\mathbf{p}_{i+1} - \mathbf{p}_i), \quad (3.2)$$

$$\mathbf{c}''(t) = \sum_{i=0}^{n-2} n(n-1) B_{n-1,i}(t) (\mathbf{p}_{i+2} - 2\mathbf{p}_{i+1} + \mathbf{p}_i). \quad (3.3)$$

From this follows that, given the end points \mathbf{p}_0 and \mathbf{p}_n , the derivatives at these end points are determined by fixing the points \mathbf{p}_1 and \mathbf{p}_{n-1} , because $\mathbf{c}'(0) = n(\mathbf{p}_1 - \mathbf{p}_0)$

and $\mathbf{c}'(1) = n(\mathbf{p}_n - \mathbf{p}_{n-1})$. Similarly, given these four points that specify the end points and derivatives at the end points, fixing the points \mathbf{p}_2 and \mathbf{p}_{n-2} determines the second order derivatives at the end points.

For the definition of the domains we work with a set of Bézier curves that together form a closed continuous path. Such a path we will refer to as a 'closed Bézier path'. It is C^∞ at its Bézier curves, but might be not even C^1 at its connection points. In the following we will write such Bézier paths as a single function $\gamma: I_T \rightarrow \mathbb{R}^2$, where $I_T = [0, T)$, and γ restricted to subintervals of unit length $\gamma|_{[j, j+1)}$ for integers i represent the independent Bézier curves.

Given a closed Bézier path, we will assign nodes to it that are being used for the discretization of the PDEs of which the path specifies a part of the domain's boundary. Such a distribution of nodes along the path will depend, as we will see in Section 3.2, on the arc-length $|\mathbf{c}'(t)|$ and the curvature κ , see [45], defined by

$$\kappa = \frac{|\mathbf{c}' \times \mathbf{c}''|}{|\mathbf{c}'|^3}. \quad (3.4)$$

Here, $\mathbf{v} \times \mathbf{w} = \det[\mathbf{v}|\mathbf{w}]$ for some arbitrary vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^2$. To get node distributions that behave well in the sense that the distances along the nodes change gradually, we will require that the arc-length and curvature are continuous along the path. Using (3.2) and (3.3) we see that this can be accomplished by letting control points \mathbf{p}_i and \mathbf{q}_j of two consecutive Bézier curves (of the same degree n) obey the following equations

$$\begin{aligned} \mathbf{p}_n - \mathbf{p}_{n-1} &= \mathbf{q}_1 - \mathbf{q}_0 && (C^1\text{-continuity}), \\ \mathbf{p}_n - 2\mathbf{p}_{n-1} + \mathbf{p}_{n-2} &= \mathbf{q}_2 - 2\mathbf{q}_1 + \mathbf{q}_0 && (C^2\text{-continuity}). \end{aligned}$$

at the connection point $\mathbf{p}_n = \mathbf{q}_0$, resulting in a path that is C^2 .

In the left picture of Figure 3.1 a closed Bézier path is displayed that consists of three Bézier curves of degree 4, i.e., they all have 5 control points. The rest of this chapter will be devoted to the assignment of nodes to such Bézier paths or combinations thereof, as is shown in the right picture of Figure 3.1. Such a distribution of nodes should represent the (combination of) curve(s) as closely as possible and should therefore use relatively many nodes where the curves turn and twist most.

3.2 Node choosing on curves

Consider a curve γ which is given by a certain parametrization $\tilde{\gamma}: I_T = [0, T] \rightarrow \mathbb{R}^2$. The path-length s as a function of t is defined by $s = F(t)$ with $F: I_T \rightarrow I_s = [0, L]$ and

$$s = F(t) = \int_0^t |\tilde{\gamma}'(\tau)| d\tau. \quad (3.5)$$

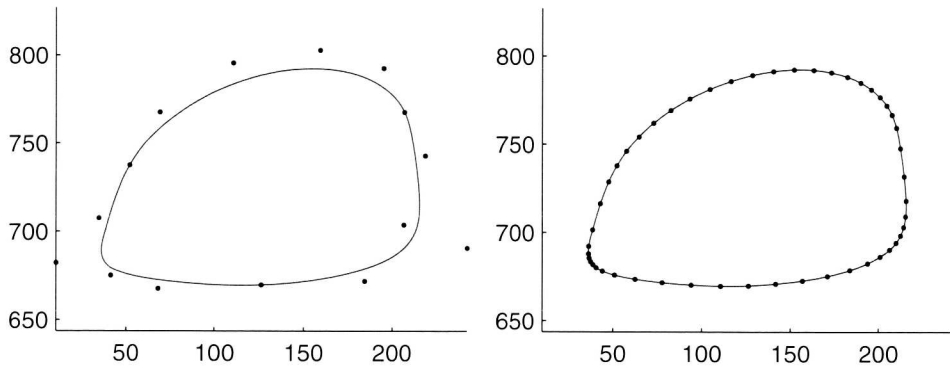


Figure 3.1: (Left) Domain consisting of three coupled bezier curves of degree 4. The black dots are the control points (5 for every curve) that specify the curves. (Right) A selection of nodes on the domain boundary that can be used for discretization.

where $\tilde{\gamma}'$ refers to the derivative $\frac{d}{dt}\tilde{\gamma}(t)$. Here, the curve's total length L is given by $L = F(T)$. The parametrization based on the path length is often called the 'natural parametrization' of the curve, see [45], and is given by

$$\gamma = \tilde{\gamma} \circ F^{-1} : I_s \rightarrow \mathbb{R}^2.$$

In addition we will define a transformation of the path-length parameter s , denoted by $G : I_s \rightarrow I = [0, 1]$, that defines the node distribution along the curve. To this end we will assume that there is a monitor function $M : I_s \rightarrow \mathbb{R}_+ \cup \{0\}$ that yields a relative node density along the curve. Using M the transformation G we define then by

$$z = G(s) = \frac{\int_0^s M(\xi) d\xi}{\int_0^L M(\xi) d\xi}. \quad (3.6)$$

In the left picture of Figure 3.3 a possible monitor function M is shown for the Bézier path of Figure 3.1. The right picture shows the accompanying transformation G . Choosing a suitable monitor function for a given path will be the subject of Section 3.3.

Given a monitor function, which has as its domain I_s , we can also define a reparametrization based on the original domain I_T of $\tilde{\gamma}$ by

$$\tilde{M} = M \circ F : I_T \rightarrow \mathbb{R}_+ \cup \{0\}.$$

It is often the case that we have, instead of function M , the function \tilde{M} available in explicit form, making that we can use M only in the form $\tilde{M} \circ F^{-1}$. Figure 3.2 shows a schematic overview of the function we defined so far.

Using this setting, we will choose a uniform grid in the unit interval I , which we denote by $\{z_i\}$, with $z_i = i/N$ for $i = 0, \dots, N$. The grid then defines a node

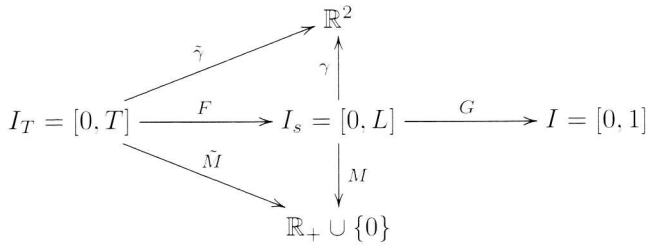


Figure 3.2: Schematic overview of the used functions.

set $\{\mathbf{x}_i\} \in \mathbb{R}^2$ by

$$\mathbf{x}_i = (\gamma \circ G^{-1})(z_i), \quad (3.7)$$

for all $i = 1, \dots, N$. The right picture of Figure 3.3 shows how the transformation G determines the node distribution. The uniform grid $\{z_i\}$ in the unit interval I is mapped by G^{-1} onto a nonuniform grid $\{s_i\}$ in $[0, L]$, which in turn is mapped by γ to the nodes in \mathbb{R}^2 , as shown in the right picture of Figure 3.1.

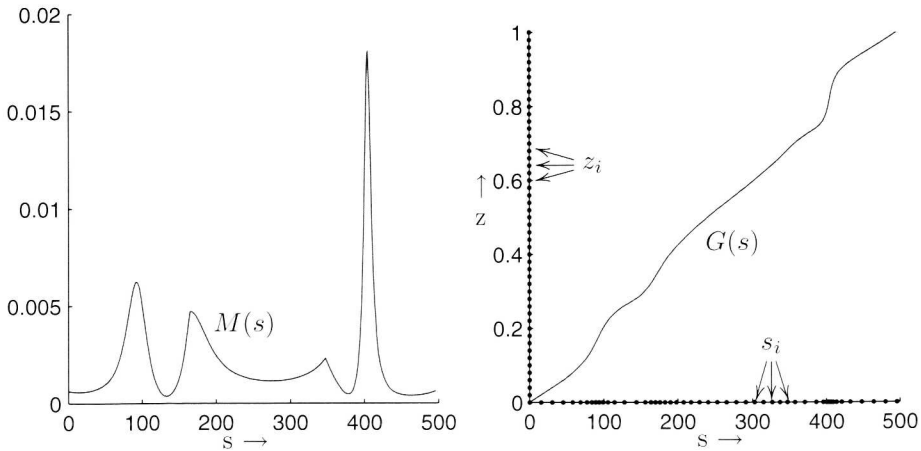


Figure 3.3: (Left) The monitor function $M(s)$. (Right) The transformation $G(s)$ together with the grids $\{z_i\}$ and $\{s_i\}$.

Using $\tilde{\gamma}$ and \tilde{M} The definition of the node set in equation (3.7) uses the functions γ and G . Because often we have been given the curve and an appropriate monitor function not in terms of the natural parametrization γ and M , but in the form $\tilde{\gamma}$ and \tilde{M} , we will define the transformation $\tilde{G} \equiv G \circ F$ and express it in $\tilde{\gamma}$ and \tilde{M} . For

the integrals in (3.6) we have

$$\begin{aligned} \int_0^s M(\xi) d\xi &= \int_0^s (\tilde{M} \circ F^{-1})(\xi) d\xi \stackrel{(\xi=F(\tau))}{=} \int_0^{F^{-1}(s)} \tilde{M}(\tau) F'(\tau) d\tau \\ &= \int_0^{F^{-1}(s)} \tilde{M}(\tau) |\tilde{\gamma}'(\tau)| d\tau. \end{aligned}$$

resulting for \tilde{G} in the expression

$$z = \tilde{G}(t) = (G \circ F)(t) = \frac{\int_0^t \tilde{M}(\tau) |\tilde{\gamma}'(\tau)| d\tau}{\int_0^T \tilde{M}(\tau) |\tilde{\gamma}'(\tau)| d\tau}. \quad (3.8)$$

Given the parametrization $\tilde{\gamma}$ and the monitor \tilde{M} the nodes can now be determined by

$$\mathbf{x}_i = (\gamma \circ G^{-1})(z_i) = (\gamma \circ F \circ F^{-1} \circ G^{-1})(z_i) = (\tilde{\gamma} \circ \tilde{G}^{-1})(z_i). \quad (3.9)$$

Because the integrals in (3.8) can not be calculated analytically, we need for determination of the nodes numerical procedures for integral evaluation and function inversion.

3.3 Choosing the monitor function M

We will now focus on choosing a suitable monitor function for a given γ . A possible choice for the monitor function could be to set

$$M(s) = 1 \quad \implies \quad z = G(s) = s/L \quad (3.10)$$

or any other constant, all resulting in the same transformation G . Given a uniform node distribution in I this will result in a uniform node distribution in I_s . Therefore the resulting transformation only takes into account the distance between the nodes along the curve, making them all equal. In the left picture of Figure 3.4 a node distribution is shown that is the result of this monitor.

As a curve may be more or less straight at some regions, whereas it twists and turns elsewhere, it might be preferable to have relatively more nodes in these latter regions. Consider the more advanced curvature monitor function

$$M(s) = \kappa(s) \quad \implies \quad z = G(s) = \frac{\int_0^s \kappa(\xi) d\xi}{\int_0^L \kappa(\xi) d\xi}. \quad (3.11)$$

A resulting node distribution is displayed in the right picture of Figure 3.4. There are two main reasons why also this one is not such a good monitor function:

- In situations where part of the curve consists of straight lines the curvature κ vanishes. This leads to G being constant at the corresponding parameter ranges, while G should be a globally invertible function.

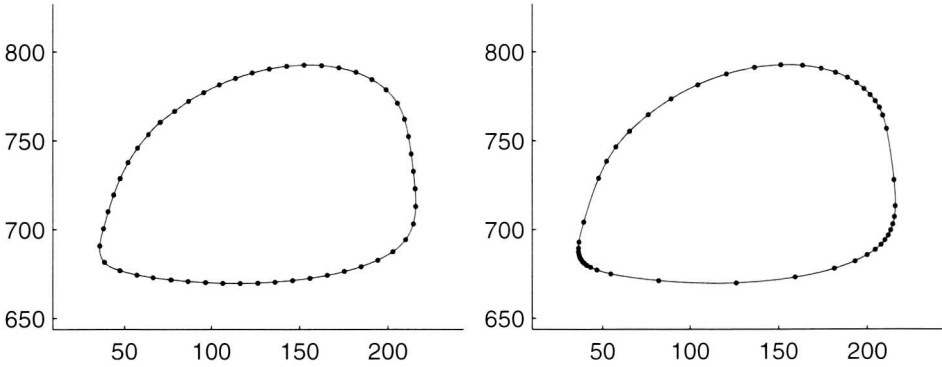


Figure 3.4: (Left) Node distribution according to the arc-length monitor (3.10). (Right) Node distribution according to the curvature monitor (3.11).

- The curvature monitor tends to assign too few nodes to the relatively straight parts of the curve, and too many to the more curving parts. This leads to rather abrupt changes in node distances, which in turn can result in ill-conditioned matrices when being used for discretization.

It seems favourable to have a combination of the two methods, where the curvature is used to select the parts of the curve where relatively many nodes are needed, while at the other hand we have some mechanism smearing out nodes over the relatively straight parts.

In general, we might assume that the monitor function M is a convex combination of a set of normalized monitor functions M_i ,

$$M = \sum_i \alpha_i \frac{M_i}{\int_0^L M_i} \quad \text{with} \quad \sum_i \alpha_i = 1. \quad (3.12)$$

Each weight α_i represents the relative contribution of the monitor i to the total monitor M . For the total transformation G this results in

$$z = G(s) = \frac{\int_0^s \sum_i \alpha_i (M_i / \int_0^L M_i)}{\int_0^L \sum_j \alpha_j (M_j / \int_0^L M_j)} = \sum_i \alpha_i \frac{\int_0^s M_i}{\int_0^L M_i} = \sum_i \alpha_i G_i(s). \quad (3.13)$$

Applying this technique and taking a combination of the curvature monitor and the arc-length monitor gives

$$z = G(s) = (1 - \alpha) \frac{\int_0^s \kappa(\xi) d\xi}{\int_0^L \kappa(\xi) d\xi} + \alpha \frac{s}{L}, \quad (3.14)$$

which with respect to the original parameter space I_T has the form

$$z = (G \circ F)(t) = (1 - \alpha) \frac{\int_0^t \tilde{\kappa}(\tau) |\tilde{\gamma}'(\tau)| d\tau}{\int_0^T \tilde{\kappa}(\tau) |\tilde{\gamma}'(\tau)| d\tau} + \alpha \frac{\int_0^t |\tilde{\gamma}'(\tau)| d\tau}{\int_0^T |\tilde{\gamma}'(\tau)| d\tau}. \quad (3.15)$$

The node distribution in the right picture of Figure 3.1 is the result of this monitor function with $\alpha = 0.5$.

3.4 Number of nodes based on maximal distances

In the previous sections a method is used that translates a uniform grid on the unit interval in a set of nodes along the boundaries. In this section we will consider a criterium for choosing the total number of nodes, N , being used for representing the boundary. It is based on the arc-length and, essentially, it specifies a maximal distance $(\Delta s)_{\max}$ along the boundary between two consecutive nodes.

Let us assume that we work with a monitor function that is a convex combination of monitors, as in (3.12), for which the first monitor equals the arc-length monitor (3.10). To achieve that the maximal distance between two consecutive nodes is bounded by $(\Delta s)_{\max}$, we can choose for the number of nodes

$$N = \lceil L/(\alpha_1(\Delta s)_{\max}) \rceil, \quad (3.16)$$

which is the smallest integer greater or equal than $L/(\alpha_1(\Delta s)_{\max})$.

That this will result in a node set of which every node distance is bounded by $(\Delta s)_{\max}$, can be seen as follows. Writing transformation (3.13) as

$$z = G(s) = \alpha_1 \frac{s}{L} + \sum_{j \neq 1} \alpha_j G_j(s), \quad (3.17)$$

we see that for two consecutive nodes, denoted by z_i and z_{i+1} , we have

$$z_{i+1} - z_i = \alpha_1 \frac{s_{i+1} - s_i}{L} + \sum_{j \neq 1} \alpha_j (G_j(s_{i+1}) - G_j(s_i)). \quad (3.18)$$

Because the functions G_j are increasing, this gives for the distance $(s_{i+1} - s_i)$ along the boundary between the two consecutive nodes

$$(s_{i+1} - s_i) \leq \frac{L}{\alpha_1} (z_{i+1} - z_i) \leq N(\Delta s)_{\max} \cdot N^{-1} = (\Delta s)_{\max}, \quad (3.19)$$

because $L/\alpha_1 \leq N(\Delta s)_{\max}$ and $z_{i+1} - z_i = N^{-1}$.

3.5 Combining multiple curves

In Section 3.3 we considered the combination of different monitor functions. Here, we will consider the combination of different curves, each curve coming with its own monitor function. We want to use the same methodology to distribute a number of nodes on this set of curves, where we have to take into account the following issues.

- The distances between the nodes should be comparable for all the curves. Having a domain with a number of holes, we would like to have the nodes distributed on the hole boundaries in a similar fashion as on the outer boundary.
- If a curve has a point where the curve does not have a continuous derivative, like a sharp angle, we would like to be able to force a node on this point. This can be accomplished by splitting the curve at the point, resulting in two curves. When doing this, one still wants the nodes to be distributed in a similar fashion on both sides of the point.

To incorporate both issues we will use the approach of splitting the path at points where they are not C^1 and glueing all resulting parametrizations together, while keeping track of the connection points.

Let us assume that after such a splitting step, we have for $j = 1, \dots, n$, at our disposal natural parametrizations $\gamma_j: I_s^j \rightarrow \mathbb{R}^2$, monitor functions $M_j: I_s^j \rightarrow \mathbb{R}_+ \cup \{0\}$ and corresponding transformations $G_j: I_s^j \rightarrow I$. Then we define the overall parametrization by

$$\gamma(s) = \begin{cases} \gamma_1(s), & 0 < s \leq s_1^*, \\ \gamma_2(s - s_1^*), & s_1^* < s \leq s_2^*, \\ \vdots & \vdots \\ \gamma_n(s - s_{n-1}^*), & s_{n-1}^* < s \leq s_n^*. \end{cases} \quad (3.20)$$

and the overall monitor M similarly. Here, the s_j^* denote the connections points along the curve, given by $s_1^* = L_1$, $s_2^* = L_1 + L_2$, \dots , $s_n^* = \sum_{j=1}^n L_j$. Using these overall γ , M and G , results in a single-curve problem instead of the multi-curve problem that we started with.

However, there is a significant difference with the single-curve problems used in Section 3.2. The nodes should be chosen such that the corresponding grid $\{s_i\}$ contains the set of connection points $\{s_j^*\}$. Using the technique from Section 3.2, one would choose a uniform grid $\{z_i\}$ in the unit interval I , which is mapped on $\{s_i\} = G^{-1}(\{z_i\})$. The resulting grid $\{s_i\}$ will then in general not obey this requirement. To solve the problem we do not start with a uniform grid in I , but with a grid that contains the nodes $z_j^* \equiv G(s_j^*)$, for all $j = 1, \dots, n$, and is as close as possible to a uniform grid. In the next section we will examine the construction of such needed semi-uniform grids $\{z_i\}$.

In the left picture of Figure 3.5 we see an example of a domain with holes and a boundary that is not C^1 . At the right the total transformation is shown. It consists of four parts, which are separated by the horizontal and vertical lines. The first two parts represent the outer boundary and the latter two represent the two holes. The nodes on the vertical axis represent the 'almost' uniform grid $\{z_i\}$ which is mapped on the non-uniform grid $\{s_i\}$.

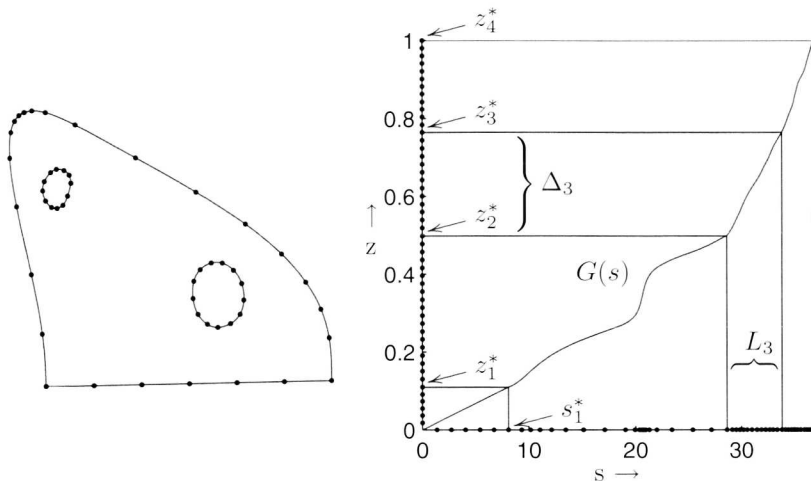


Figure 3.5: (Left) Node set selection for a domain with two holes and outer boundary that is not C^1 . (Right) Used transformation G with grids $\{z_i\}$ and $\{s_i\}$.

3.6 Semi-uniform grids

In this section we consider two approaches for finding the semi-uniform grids in I , one using maximal distances between the nodes and another using fixed number of nodes.

Maximal distance principle If we want to base the node choice on the maximal distance principle of Section 3.4, we can treat every part of the transformation separately, assigning

$$N_j = \left\lceil \Delta_j \frac{L}{\alpha_1 (\Delta s)_{\max}} \right\rceil \quad (3.21)$$

nodes to it. Here, α_1 is again the scalar multiplying the arc-length monitor in the convex combination of monitors, L is the total length $\sum_{j=1}^n L_j$ of the curve and $\Delta_j = z_j^* - z_{j-1}^*$, where we set $z_0^* \equiv 0$. The Δ_j can be expressed in the monitor functions giving

$$\Delta_j = \frac{\int_{s_{j-1}^*}^{s_j^*} M(\xi) d\xi}{\int_0^L M(\xi) d\xi} = \frac{\int_0^{L_j} M_j(\xi) d\xi}{\int_0^L M(\xi) d\xi}. \quad (3.22)$$

Fixed number of nodes Given that we want to distribute N nodes over the boundary, we have to find n integers N_j , with $N = \sum_{j=1}^n N_j$, such that

$$\frac{N_j}{N} \approx \Delta_j, \quad \Delta_j = z_j^* - z_{j-1}^*. \quad (3.23)$$

To formulate this more precisely we consider the problem of finding the $N_j \in \mathbb{N}$ that

$$\text{minimize } \sum_{j=1}^n f \left(\left| \frac{N_j}{N} - \Delta_j \right| \right), \quad \text{with constraint: } \sum_{j=1}^n N_j = N. \quad (3.24)$$

where

- $f: \mathbb{R}_+ \cup \{0\} \rightarrow \mathbb{R}$ is a monotonically increasing, convex function.
- $\Delta_j \in (0, 1]$, with $\sum_{j=1}^n \Delta_j = 1$ and $N \in \mathbb{N}$.

According to Theorem 1 below, the solution must have $N_j \in \{[\Delta_j N], \lfloor \Delta_j N \rfloor\}$, so that for small numbers n we can consider all 2^n possibilities. ($\lceil x \rceil$ is the smallest integer greater than or equal to x and $\lfloor x \rfloor$ is the largest integer smaller than or equal to x .) This would be the case when we have for example a C^1 outer boundary with a small number of holes with C^1 boundaries, giving that n equals the number of holes plus one. On the other hand, for boundaries with a lot of C^1 -discontinuities, like complex polygonal domains, this approach could become unfeasible.

The semi-uniform grid $\{z_i\}$ in Figure 3.5 is based on a fixed number of nodes and the minimization problem above. It contains the nodes $\mathbf{z}_1^* = 0.1097$, $\mathbf{z}_2^* = \mathbf{z}_1^* + 0.3886$, $\mathbf{z}_3^* = \mathbf{z}_2^* + 0.2658$ and $\mathbf{z}_4^* = \mathbf{z}_3^* + 0.2359 = 1$. In total it has 50 nodes, distributed over the four parts with node numbers 6, 19, 13 and 12, respectively, in each part giving a uniform distribution, with node distances equal to 0.0183, 0.0205, 0.0204 and 0.0194, respectively. The node numbers are calculated using a function $f(x) = |x|$ in the minimization problem. The function G is (3.14) with $\alpha = 0.7$.

We will end this chapter with a theorem on the minimization problem (3.24).

Theorem 1. *If $\{N_j\}$ is a solution of the minimization problem (3.24), then for every j , $N_j \in \{[\Delta_j N], \lfloor \Delta_j N \rfloor\}$.*

Proof. We consider an equivalent minimization problem. Find disjoint index sets \mathcal{I}_- , \mathcal{I}_0 , \mathcal{I}_+ , with $\mathcal{I}_- \cup \mathcal{I}_0 \cup \mathcal{I}_+ = \{1, \dots, n\}$, and sets of numbers and $n_j \in \mathbb{N} \cup \{0\}$, that

$$\text{minimize } \sum_{i=1}^n f \left(\frac{\epsilon_j + n_j}{N} \right), \quad \text{with constraints: } \begin{cases} \sum_{j \in \mathcal{I}_-} \epsilon_j + n_j = \sum_{j \in \mathcal{I}_-} \epsilon_j + n_j, \\ n_j = 0, \quad \text{for all } j \in \mathcal{I}_0. \end{cases} \quad (3.25)$$

where

$$\epsilon_j = \begin{cases} \Delta_j N - \lfloor \Delta_j N \rfloor, & j \in \mathcal{I}_-, \\ 0, & j \in \mathcal{I}_0, \\ \lceil \Delta_j N \rceil - \Delta_j N, & j \in \mathcal{I}_+. \end{cases}$$

The two minimization problems are equivalent with respect to the map

$$(\mathcal{I}_-, \mathcal{I}_0, \mathcal{I}_+, (n_j)) \rightarrow (N_j), \quad N_j = \begin{cases} \Delta_j N - \epsilon_j - n_j, & j \in \mathcal{I}_-, \\ \Delta_j N, & j \in \mathcal{I}_0, \\ \Delta_j N + \epsilon_j + n_j, & j \in \mathcal{I}_+. \end{cases}$$

which is bijective if we take the constraints into account.

We will prove the theorem by showing that for the minimizer of the latter problem all $n_j = 0$. In the following the term 'cost function' refers to the function to be minimized. To proceed let us consider some choice of $(\mathcal{I}_-, \mathcal{I}_0, \mathcal{I}_+, (n_j))$, with for some of the i , $n_j > 0$. We will search for choices that have a smaller cost function value.

If we have $j \in \mathcal{I}_-$, with $n_j \geq 1$ and $k \in \mathcal{I}_+$ with $n_k \geq 1$, then we can subtract 1 from both n_j and n_k . The change in cost function will be

$$f\left(\frac{\epsilon_j + n_j - 1}{N}\right) + f\left(\frac{\epsilon_k + n_k - 1}{N}\right) - \left(f\left(\frac{\epsilon_j + n_j}{N}\right) + f\left(\frac{\epsilon_k + n_k}{N}\right)\right) < 0,$$

while the constraint in (3.25) continues to be satisfied. This we can repeat several times until for one of the index sets all $n_j = 0$. Let us assume that this index set is \mathcal{I}_- . We can then continue in the following way.

Select arbitrary $i \in \mathcal{I}_-$ (has $n_j = 0$) and $j \in \mathcal{I}_+$ with $n_k \geq 1$, we can subtract 1 from n_k and move j from \mathcal{I}_- to \mathcal{I}_+ , while replacing ϵ_j with $1 - \epsilon_j$. The change in the cost function will be

$$\begin{aligned} & f\left(\frac{1 - \epsilon_j}{N}\right) + f\left(\frac{\epsilon_k + n_k - 1}{N}\right) - \left(f\left(\frac{\epsilon_j}{N}\right) + f\left(\frac{\epsilon_k + n_k}{N}\right)\right) \\ & < f\left(\frac{1 - \epsilon_j}{N}\right) + f\left(\frac{\epsilon_k + n_k - |1 - 2\epsilon_j|}{N}\right) - \left(f\left(\frac{\epsilon_j}{N}\right) + f\left(\frac{\epsilon_k + n_k}{N}\right)\right) \quad (f \text{ increasing}) \\ & \leq \left(f\left(\frac{\max(1 - \epsilon_j, \epsilon_j)}{N}\right) - f\left(\frac{\max(1 - \epsilon_j, \epsilon_j)}{N} - \frac{|1 - 2\epsilon_j|}{N}\right)\right) - \left(f\left(\frac{\epsilon_k + n_k}{N}\right) - f\left(\frac{\epsilon_k + n_k}{N} - \frac{|1 - 2\epsilon_j|}{N}\right)\right) \\ & \leq 0 \quad (f \text{ convex}) \end{aligned}$$

This process can be repeated until all $n_j = 0$. For if there is only one element left in \mathcal{I}_- , all $n_j = 0$ because of the constraint. We now have constructed a new choice of $(\mathcal{I}_-, \mathcal{I}_0, \mathcal{I}_+, (n_j))$, with a cost function value that is lower than the cost function value of the original choice.

If after the first step the index set with all $n_j = 0$ is \mathcal{I}_+ instead of \mathcal{I}_- , we can interchange the roles of both index sets in the second step, arriving also at a state for which all n_j vanish. \square

Chapter 4

Spatial discretization of the field equations

4.1 Introduction

In the present chapter a meshfree method is presented for solving time-discrete diffusion equations. This method is meant to be used for the simulation of certain models used in brain research. Such models describe mechanisms behind the development of the nervous system and in particular the formation of the connections between the nerve cells [21]. The resulting equations are constituted by two systems. One of the systems is a set of diffusion equations for certain chemicals (attractants and repellents) that is coupled to the other system which consists of nonlinear ODEs describing the growth of the connection forming structures, i.e., axons. The diffusion equations contain moving sources which are small compared to the domain and their strength and movement may depend on the solution of the ODEs. The nonlinear ODEs depend on the solutions of the diffusion equations, and gradients thereof, evaluated along solution paths in the space domain.

For the sake of clarity, we consider an example consisting of one diffusion equation and one ODE ([28], Chapter 2).

$$\begin{aligned}\frac{\partial}{\partial t}\rho(\mathbf{x}, t) &= (d\Delta - \kappa)\rho(\mathbf{x}, t) + S(\mathbf{x} - \mathbf{r}_0) + S(\mathbf{x} - \mathbf{r}(t)), \\ \frac{d}{dt}\mathbf{r}(t) &= \nabla\rho(\mathbf{r}(t), t),\end{aligned}$$

where ρ is some concentration, S is a source profile with compact support, and \mathbf{r}_0 and $\mathbf{r}(t)$ are two source locations of which $\mathbf{r}(t)$ moves in the direction of higher concentrations of ρ . A first-order discretization in time of this model, where the ODE is

handled explicitly and the diffusion equation implicitly, is

$$\begin{aligned} (1 - \delta t(d\Delta - \kappa))\rho^{n+1}(\mathbf{x}) &= \rho^n(\mathbf{x}) + \delta t(S(\mathbf{x} - \mathbf{r}_0) + S(\mathbf{x} - \mathbf{r}^{n+1})), \\ \mathbf{r}^{n+1} &= \mathbf{r}^n + \delta t(\nabla\rho^n(\mathbf{r}^n)), \end{aligned}$$

where the superscripts n and $n + 1$ denote different time levels and δt is the size of a time step. Clearly, we have to solve in every time step an elliptic equation and this step will comprise the majority of the work. Time discretizations of Runge-Kutta type will require the calculation of several of such equations per time step.

This chapter is focused on the numerical solution of these elliptic equations. The combined challenges of small, moving sources and flexible domain geometry suggests the use of a meshfree approach. The basic idea behind meshfree methods is to work with an arbitrary set of nodes instead of a grid. While this makes it easy to handle refinement and flexible domain geometries, function approximation and solvability of resulting systems become more complicated.

In most meshfree methods solving elliptic equations starts with the definition of an approximation space in which a best approximation of the solution is sought. This space is defined by selecting a set of basis functions, which in all cases forms a partition of unity to guarantee at least first-order approximation. While in finite element methods the basis functions are chosen with respect to a partition of the domain, meshfree methods form a basis by assigning functions with compact supports to each node of an arbitrary node set. Here the function's supports have to cover the whole domain, while the overlap has to be minimal so that the resulting linear equation systems become as sparse as possible. More on meshfree methods can be found in the overview articles [4, 12, 33].

In the method developed in this chapter the approximation space is not defined through a set of basis functions but as the image of a linear map. This map assigns to every combination of function values on the nodes a piecewise smooth function on the domain by using a least squares approximation locally. In this way the function values on the nodes parametrize the approximation space that will be used in a Galerkin procedure. Because the approximating functions are piecewise multinomials the integrals in the resulting matrices can be evaluated exactly. This is in contrast to methods like DEM [36] and EFG [5] or variants thereof, where a quadrature rule is needed because of the use of moving least squares interpolants.

A Voronoi diagram [7, 11] based on the nodes is used for finding neighboring nodes and for glueing local approximations together to form a global approximation. Some other meshfree methods that use Voronoi diagrams (or the related Delaunay triangulations) are the Natural Element Method [47] and the Meshless Finite Element Method [27]. Both methods, however, use an approximation space constructed by choosing a set of basis functions.

For the construction of suitable node sets different techniques are available, see for example [34, 6]. Here an algorithm is presented that makes use of Voronoi diagrams and shifting nodes to 'regularize' node sets.

The contents of the chapter is as follows. We start in Section 4.2 with meshfree function approximation based on a local least squares approximation. This is followed by a description of the discretization of the equation in Section 4.3. In Section 4.4 more practical considerations concerning the computation of the discretization are discussed. A way for dealing with different domains and refinement in a meshfree context is examined in Section 4.5, followed by Section 4.6 with a numerical test example. Finally, Section 4.7 summarizes the chapter.

4.2 Meshfree function approximation

This section deals with the meshfree function approximation used in the discretization of the equation. We will start with a description of local least squares approximation, examine its convergence and use it for a global approximation.

4.2.1 Local least squares approximation

Given a function $f: \mathbb{R}^2 \rightarrow \mathbb{R}$ and some node set $\{\mathbf{x}_0 + h\mathbf{x}_1, \dots, \mathbf{x}_0 + h\mathbf{x}_n\}$, we want to approximate the function in the disc with center \mathbf{x}_0 and radius $h > 0$, using the values $\{f(\mathbf{x}_0 + h\mathbf{x}_1), \dots, f(\mathbf{x}_0 + h\mathbf{x}_n)\}$. We choose the approximant to be a linear combination of multinomials which are maximal of second order, i.e.,

$$p^h(\mathbf{x}) = \boldsymbol{\alpha} \cdot \mathbf{b}(\mathbf{x}) \quad \text{with} \quad \mathbf{b}(\mathbf{x}) = (1 \quad x \quad y \quad x^2 \quad xy \quad y^2)^T, \quad (4.1)$$

where $\mathbf{x} = (x, y)$ and $\boldsymbol{\alpha} \in \mathbb{R}^6$ is determined by minimizing

$$\sum_{i=1}^n |p^h(\mathbf{x}_0 + h\mathbf{x}_i) - f(\mathbf{x}_0 + h\mathbf{x}_i)|^2. \quad (4.2)$$

From now on we will assume that $\mathbf{x}_0 = \mathbf{0}$. A particular choice of \mathbf{x}_0 will not influence any approximation properties, because a translation in the domain of the least squares problem can be viewed as a linear change of basis functions, meaning that the approximation is in the same function space.

If we define

$$B(h) = \left(\mathbf{b}(h\mathbf{x}_1) \mid \dots \mid \mathbf{b}(h\mathbf{x}_n) \right) \quad \text{and} \quad \mathbf{F}(h) = \begin{pmatrix} f(h\mathbf{x}_1) \\ \vdots \\ f(h\mathbf{x}_n) \end{pmatrix},$$

substitution of (4.1) into (4.2), will result in

$$\boldsymbol{\alpha}^T (B(h)B(h)^T) \boldsymbol{\alpha} - 2\boldsymbol{\alpha}^T B(h)\mathbf{F}(h) + \mathbf{F}(h)^T \mathbf{F}(h), \quad (4.3)$$

which has to be minimized over $\boldsymbol{\alpha}$. The 6×6 matrix $B(h)B(h)^T$ is positive semi-definite and positive definite if $\det(B(h)B(h)^T) \neq 0$. In the latter case a unique minimum exists that is determined by

$$(B(h)B(h)^T) \boldsymbol{\alpha} = B(h)\mathbf{F}(h). \quad (4.4)$$

The entries of the matrix $B(h)B(h)^T$ can be written as

$$(B(h)B(h)^T)_{i,j} = \sum_{k=1}^n b_i(h\mathbf{x}_k)b_j(h\mathbf{x}_k). \quad (4.5)$$

Ill-conditioning

For finding the local approximation we have to solve equation (4.4). It turns out that for small h the matrix $B(h)B(h)^T$ is ill-conditioned for all sets $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$. To obtain a rough lower bound for the condition number we can use the fact that the diagonal elements of a symmetric matrix are bounded by the smallest and largest eigenvalue, which can be easily deduced using the *interlacing eigenvalues theorem for bordered matrices* [24].

The (1, 1)-entry of $B(h)B(h)^T$ is equal to $\sum_{i=1}^n 1 = n$, while we also have, by using (4.5) and considering the diagonal elements $(B(h)B(h)^T)_{i,i}$, $i = 4, 5, 6$,

$$4\lambda_{\min} \leq h^4 \sum_{j=1}^n |\mathbf{x}_j|^4 \leq nh^4 \max_j |\mathbf{x}_j|^4 \implies \frac{4}{h^4 \max_j |\mathbf{x}_j|^4} \leq \frac{\lambda_{\max}}{\lambda_{\min}}. \quad (4.6)$$

Therefore $\text{cond}(B(h)B(h)^T) \geq \mathcal{O}(h^{-4})$ and direct calculation of the approximation could be an error-prone procedure. A more stable way of calculating the approximation is to use scaled basis functions $\hat{b}_i(\mathbf{x}) = b_i(\frac{1}{h}\mathbf{x})$, $i = 1, \dots, 6$. Then the matrices in equation (4.4) become independent of h and the ill-conditioning for small h will disappear.

4.2.2 Convergence properties

To examine the convergence order of such an approximation, we will assume from now on that all points $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ are situated in the unit circle, that at least for one point $\|\mathbf{x}_i\|_2 = 1$, and that the resulting matrix $B(1)B(1)^T$ is invertible. An example configuration of points is shown in Figure 4.1. For an arbitrary point \mathbf{x} in the unit circle we will examine now $|p^h(h\mathbf{x}) - f(h\mathbf{x})|$.

First we define $S(h) = \text{diag}(1, h, h, h^2, h^2, h^2)$ and $B = B(1)$, so that we can write $\mathbf{b}(h\mathbf{x}) = S(h)\mathbf{b}(\mathbf{x})$ and $B(h) = S(h)B$. Using this we have from (4.4)

$$S(h)(BB^T)S(h)\boldsymbol{\alpha}(h) = S(h)B\mathbf{F}(h) \implies S(h)\boldsymbol{\alpha}(h) = (BB^T)^{-1}B\mathbf{F}(h)$$

and the approximation $p^h(h\mathbf{x})$ can be written as

$$p^h(h\mathbf{x}) = \boldsymbol{\alpha}(h) \cdot \mathbf{b}(h\mathbf{x}) = \mathbf{b}(\mathbf{x})^T S(h)\boldsymbol{\alpha}(h) = \mathbf{b}(\mathbf{x})^T (BB^T)^{-1}B\mathbf{F}(h).$$

For the difference $p^h(h\mathbf{x}) - f(h\mathbf{x})$ a Taylor series expansion can be written down by making Taylor series of $\mathbf{F}(h)$ and $f(h\mathbf{x})$. It turns out that the lower order coefficients

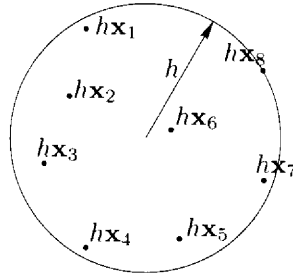


Figure 4.1: Example configuration of points and the related circle.

cancel out due to the following equalities.

$$\begin{aligned} \mathbf{v}^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} &= 1, & \mathbf{v}^T \begin{pmatrix} Df(\mathbf{0})(\mathbf{x}_1) \\ \vdots \\ Df(\mathbf{0})(\mathbf{x}_n) \end{pmatrix} &= Df(\mathbf{0})(\mathbf{x}), \\ \mathbf{v}^T \begin{pmatrix} D^2f(\mathbf{0})(\mathbf{x}_1, \mathbf{x}_1) \\ \vdots \\ D^2f(\mathbf{0})(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} &= D^2f(\mathbf{0})(\mathbf{x}, \mathbf{x}). \end{aligned} \quad (4.7)$$

where \mathbf{v} is defined by $\mathbf{v}^T \equiv \mathbf{b}(\mathbf{x})^T (BB^T)^{-1} B$. Here we used the m -th order Fréchet derivative $D^m(f)(\mathbf{0}): (\mathbb{R}^2)^m \rightarrow \mathbb{R}$ of f at $\mathbf{0}$ which is a multi-linear operator and defined by

$$D^m f(\mathbf{0})(\boldsymbol{\eta}^1, \dots, \boldsymbol{\eta}^m) = \left(\prod_{i=1}^m (\boldsymbol{\eta}_1^i \partial_1 + \boldsymbol{\eta}_2^i \partial_2) \right) f(\mathbf{z}) \Big|_{\mathbf{z}=\mathbf{0}}.$$

Each of these equations can be associated with a least squares approximation problem that has an exact solution because the approximated function is a low order multinomial.

It follows that

$$p^h(h\mathbf{x}) = f(h\mathbf{x}) + \frac{1}{6}Ch^3 + \mathcal{O}(h^4)$$

with

$$C = \mathbf{b}(\mathbf{x})^T (BB^T)^{-1} B \begin{pmatrix} D^3f(\mathbf{0})(\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_1) \\ \vdots \\ D^3f(\mathbf{0})(\mathbf{x}_n, \mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} - D^3f(\mathbf{0})(\mathbf{x}, \mathbf{x}, \mathbf{x}). \quad (4.8)$$

For arbitrary $\mathbf{z} \in \mathbb{R}^2$ with $|\mathbf{z}| \leq 1$, we have $\|\mathbf{b}(\mathbf{z})\|_2 \leq \sqrt{6}\|\mathbf{b}(\mathbf{z})\|_\infty \leq \sqrt{6}$ and

$$\begin{aligned} D^3f(\mathbf{0})(\mathbf{z}, \mathbf{z}, \mathbf{z}) &= (z_1\partial_1 + z_2\partial_2)^3 f(z_1, z_2) \Big|_{z_1=0, z_2=0} \\ &\leq 8 \max_{i,j,k=1,2} |\partial_i \partial_j \partial_k f(\mathbf{0})|. \end{aligned}$$

which, because $|\mathbf{x}|, |\mathbf{x}_1|, \dots, |\mathbf{x}_n| \leq 1$, yield together

$$|C| \leq (48n \|(BB^T)^{-1}\|_\infty + 8) \max_{i,j,k=1,2} |\partial_i \partial_j \partial_k f(\mathbf{0})|.$$

For a real square $m \times m$ -matrix A we have the inequality of Hadamard [51] which states that $|\det(A)| \leq (m+1)^{(m+1)/2} / 2^m$. Applying the general expression for matrix inverses using cofactors on BB^T gives

$$((BB^T)^{-1})_{i,j} = \frac{1}{\det(BB^T)} (-1)^{i+j} \det(BB^T[j, i]).$$

where $BB^T[j, i]$ is a matrix BB^T with row i and column j deleted. One can now estimate

$$\|(BB^T)^{-1}\|_\infty \leq \frac{81}{2n \det(\frac{1}{n}BB^T)},$$

which by denoting $D(0; h) = \{|\mathbf{x}| \leq 1\}$, results finally in

$$\sup_{\mathbf{x} \in D(0; h)} |p^h(\mathbf{x}) - f(\mathbf{x})| \leq \left(\frac{324}{\det(\frac{1}{n}BB^T)} + \frac{4}{3} \right) \max_{i,j,k=1,2} |\partial_i \partial_j \partial_k f(\mathbf{0})| h^3, \quad (4.9)$$

for h small enough.

Therefore we can state that for node sets in the domain of which the subsets used for local approximation can be enclosed in circles of radius h and for which all $\det(\frac{1}{n}BB^T)$ are bounded from below by some constant, the approximation is of third order. These used subsets we will call the local node sets.

On the other hand, for arbitrary local node sets, $\det(\frac{1}{n}BB^T)$ can be arbitrary close to zero, making the third order constant larger, possibly resulting in a bad approximation. Because the idea of meshfree methods is to start with arbitrary node sets, we will after choosing subsets of the global node set, test their approximation ability by evaluating $\det(\frac{1}{n}BB^T)$. For determinant values too small we will add more points from the global node set, repeating this procedure until the determinant is above the required constant.

Derivative approximation

For the approximation of derivatives we expect similar behavior with one order lower accuracy, i.e., second order. To examine how well the derivatives are approximated, we define the matrices $D_1, D_2 \in \mathbb{R}^{6 \times 6}$ by

$$D_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad D_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{pmatrix}.$$

These represent the actions of the partial derivatives with respect to the basis functions b_1, \dots, b_6 in such a way that for functions $g: \mathbb{R}^2 \rightarrow \mathbb{R}$ that are arbitrary linear combinations of the basis functions $g(\mathbf{x}) = \boldsymbol{\alpha} \cdot \mathbf{b}(\mathbf{x})$ we have

$$\partial_i g(\mathbf{x}) = \boldsymbol{\alpha} \cdot D_i \mathbf{b}(\mathbf{x}) \quad (i = 1, 2).$$

With respect to the matrix $S(h)$ these matrices obey $D_i S(h) = \frac{1}{h} S(h) D_i$. This is a direct consequence of the fact that for arbitrary functions $f: \mathbb{R} \rightarrow \mathbb{R}$ with $f(hx) = h^p f(x)$ for some constant p , it follows that $Df(hx) = h^{p-1} Df(x)$.

For the approximation of the i -th partial derivative we can write now

$$\partial_i p^h(h\mathbf{x}) = \boldsymbol{\alpha}(h) \cdot D_i \mathbf{b}(h\mathbf{x}) = D_i^T \mathbf{b}(\mathbf{x})^T \frac{1}{h} S(h) \boldsymbol{\alpha}(h) = D_i^T \mathbf{b}(\mathbf{x})^T (BB^T)^{-1} B \frac{1}{h} \mathbf{F}(h).$$

Differentiating the equations (4.7) gives

$$\begin{aligned} D_i^T v^T \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} &= \frac{\partial}{\partial x_i} 1 = 0, & D_i^T v^T \begin{pmatrix} Df(\mathbf{0})(\mathbf{x}_1) \\ \vdots \\ Df(\mathbf{0})(\mathbf{x}_n) \end{pmatrix} &= \frac{\partial}{\partial x_i} Df(\mathbf{0})(\mathbf{x}) = \partial_i f(\mathbf{0}), \\ D_i^T v^T \begin{pmatrix} D^2 f(\mathbf{0})(\mathbf{x}_1, \mathbf{x}_1) \\ \vdots \\ D^2 f(\mathbf{0})(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} &= \frac{\partial}{\partial x_i} D^2 f(\mathbf{0})(\mathbf{x}, \mathbf{x}) = 2D(\partial_i f)(\mathbf{0})(\mathbf{x}). \end{aligned}$$

which again leads to vanishing coefficients in the Taylor series expansion of $\partial_i p^h(h\mathbf{x}) - \partial_i f(h\mathbf{x})$, resulting in

$$\partial_i p^h(h\mathbf{x}) = \partial_i f(h\mathbf{x}) + \frac{1}{6} C'_i h^2 + \mathcal{O}(h^3)$$

with

$$C'_i = D_i^T \mathbf{b}(\mathbf{x})^T (BB^T)^{-1} B \begin{pmatrix} D^3 f(\mathbf{0})(\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_1) \\ \vdots \\ D^3 f(\mathbf{0})(\mathbf{x}_n, \mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} - D^3 f(\mathbf{0})(\mathbf{x}, \mathbf{x}, \mathbf{x}). \quad (4.10)$$

A similar estimation as in the non-derivative case yields for h small enough

$$\sup_{\mathbf{x} \in D(0;h)} |\partial_i p^h(\mathbf{x}) - \partial_i f(\mathbf{x})| \leq \left(\frac{648}{\det(\frac{1}{n} BB^T)} + \frac{4}{3} \right) \max_{i,j,k=1,2} |\partial_i \partial_j \partial_k f(\mathbf{0})| h^2. \quad (4.11)$$

4.2.3 Quality of local node sets

Above it was stated that we use $\det(\frac{1}{n} BB^T)$ of a local node set as a measure of the approximation quality. In this paragraph we want to investigate this further.

In the calculation of the determinant a circle of radius h was used which contains all nodes of the local node set with the restriction that at least one of the nodes is on the circle. The convergence results (4.9) and (4.11) then hold for arbitrary points

in the circle. In most cases the area in which we use the approximation is actually much smaller than the circle (e.g. a Voronoi tile containing the central node, See Figure 4.2) and probably somewhere in the middle of it. This means that there are many possibilities for choosing such a circle, as is illustrated in Figure 4.2. Here, a local node set is shown together with a polygonal area in which we are interested. The three circles, having radii 0.97 (solid circle), 1.00 and 1.10, respectively, are all suitable choices.

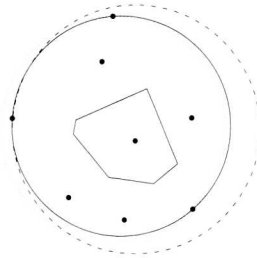


Figure 4.2: Three possible choices for the circle

Because the approximation is independent of the choice of the circle, we want the quality measure of the local node set to be also independent of this choice, which means that we have to make a particular choice. Before showing what a good choice is we will examine how $\det(\frac{1}{n}BB^T)$ responds to translations, rotations and scaling of the local node set. For this we introduce a translation vector $\mathbf{v} \in \mathbb{R}^2$, an orthogonal rotation matrix $Q \in \mathbb{R}^{2 \times 2}$ and some real scaling constant $r > 0$. If we compare now $\det(\frac{1}{n}BB^T)$ for a node set $\{\mathbf{x}_i\}$, $i = 1, \dots, n$ with $\det(\frac{1}{n}\tilde{B}\tilde{B}^T)$, where \tilde{B} is made out of vectors $\mathbf{b}(rQ\mathbf{x}_i + \mathbf{v})$ instead of vectors $\mathbf{b}(\mathbf{x}_i)$ as with B , then it can be shown that

$$\det(\frac{1}{n}\tilde{B}\tilde{B}^T) = r^{16} \det(\frac{1}{n}BB^T).$$

This means that the measurement is invariant with respect to rotation and translation, but that the radius of the chosen circle strongly affects the value of the determinant. For example, choosing a circle which is twice as large, means that the distances between the nodes in the local node set measured relative to the circle radius h will be twice as small, yielding $r = 0.5$. This will result in a determinant which is $(0.5)^{16} = 1.5 \cdot 10^{-5}$ times as large. The determinants in Figure 4.2 are in ratio $1 : 0.61 : 0.13$.

A good circle choice is the smallest enclosing circle of the given local node set, so that the determinant is maximal. From now on, we will use this choice and assume that the area in which we will use our approximation is inside the smallest enclosing circle. Actually, the solid circle in Figure 4.2 is the smallest enclosing circle.

An algorithm for calculation of the smallest enclosing circle is described in [7], which is a randomized incremental algorithm with an expected time complexity of linear order in the number of nodes in the local node set and hence is very efficient.

4.2.4 Global approximation

In the previous subsections a method for local function approximation is discussed. Here we will use it for making a global approximation of a function, given an arbitrary set of nodes in our global domain. The basic idea is to divide the domain into disjoint sub-domains in which we use then the local approximation.

Such a division can be made out of our set of nodes by calculating the related Voronoi diagram [7]. In such a diagram every node has its own Voronoi tile, which consists of all points of the domain which are closer to the node associated with the tile than to every other node of the node set. This will make all the tiles disjoint and their union equal to \mathbb{R}^2 except for a set of which the points are equally close to two or more nodes. This set is called the Voronoi diagram. In the left picture of Figure 4.3 the Voronoi diagram of a particular node set is shown. We will use the unbounded Voronoi diagram to make a division for our bounded domain by connecting all nodes which are on the boundary of the domain by straight lines. This gives us an approximation of our domain which is of second order (with respect to integrals) in the distance between the nodes on the boundaries in case the boundaries are curved. The right picture of Figure 4.3 shows the division of the domain.

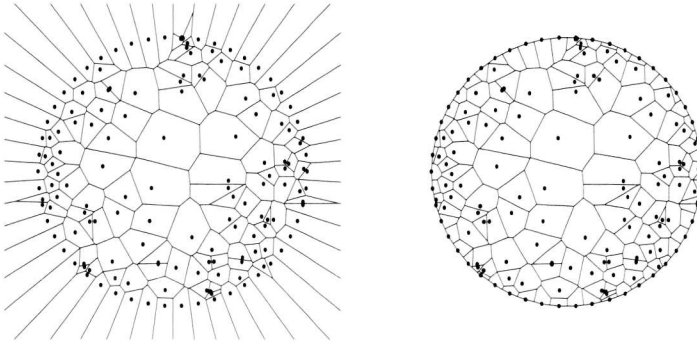


Figure 4.3: Voronoi diagram of the node set and the domain decomposition based on it.

Now for all the Voronoi tiles we have a local approximation giving us a global approximation in our domain except for the Voronoi diagram, which is of measure zero and therefore irrelevant, because we will use the global approximation for evaluating integrals.

Finding neighboring nodes

For finding the local approximation in a Voronoi tile a local set of nodes has to be found. Using the diagram, we can define nodes to be neighbors if and only if their Voronoi tiles have a common Voronoi edge of the diagram. In this way we can find

a ring of neighbors of a node and also a second ring of neighbors of neighbors of a node. Proceeding until we have at least the number of neighbors we need, which is minimally six nodes in case of six basis functions, and the determinant $\det(\frac{1}{n}BB^T)$ is above some threshold, will result in the local node set.

Global approximation mapping

We are now ready to define a mapping which will map a function defined on the set of nodes \mathcal{N} into the space $L^1(\Omega)$, where Ω is our domain. This mapping will be used in the subsequent section on the discretization of the PDE. We define the mapping $F_{\mathcal{N}}: \mathbb{R}^N \rightarrow L^1(\Omega)$ such that for a given node function $\mathbf{f} \in \mathbb{R}^N$, a tile Ω_i and its local approximation p_{Ω_i} , we have $F_{\mathcal{N}}(\mathbf{f}) = p_{\Omega_i}$ in tile Ω_i . This defines $F_{\mathcal{N}}(\mathbf{f})$ in every tile of the domain. What is left is the Voronoi diagram, which is a set with measure zero and because functions that differ on a set of zero measure are identical in L^1 , the definition is complete.

If we define also a restriction map $G_{\mathcal{N}}: C^1(\Omega) \rightarrow \mathbb{R}^N$ by $G_{\mathcal{N}}(u) = u|_{\mathcal{N}}$ (pointwise restriction), we can write for an arbitrary function $u \in C^3(\Omega)$, and h defined as the maximal radius of all circles used in the local approximation,

$$\|u - F_{\mathcal{N}} \circ G_{\mathcal{N}}(u)\|_{\infty} \leq Kh^3, \quad (4.12)$$

for h small enough and where K is some positive constant. In a similar way we have for $i = 1, 2$,

$$\|\partial_i u - \partial_i(F_{\mathcal{N}} \circ G_{\mathcal{N}}(u))\|_{\infty} \leq K'h^2, \quad (4.13)$$

4.3 Discretization of the PDE

To discretize the PDE problem we will formulate it as a minimization problem and use the approximation technique from the previous section to end up with a discrete minimization problem. We then consider the solvability of the linear system that has to be solved for finding the minimum.

4.3.1 Minimization problem

We will consider the elliptic problem

$$\begin{aligned} (d\Delta - \kappa)u + f &= 0, & \mathbf{x} \in \Omega, \\ \nabla u \cdot \mathbf{n} &= 0, & \mathbf{x} \in \partial\Omega. \end{aligned} \quad (4.14)$$

with $d, \kappa > 0$ and $f \in L^2(\Omega)$. This boundary value problem can be written as the variational problem

$$K[u] = \min_{w \in H^1} K[w], \quad \text{with} \quad \begin{cases} K[w] = A(w, w) - L(f, w), \\ A(v, w) = \int_{\Omega} \frac{1}{2} d \nabla v \cdot \nabla w + \frac{1}{2} \kappa v w \, dx, \\ L(f, w) = \int_{\Omega} f w \, dx. \end{cases} \quad (4.15)$$

which can be found in textbooks on elliptic PDEs, for example [14].

To find a numerical solution of the minimization problem we will calculate first an approximation of the integral for a given node function $\mathbf{w} \in \mathbb{R}^N$. We do this by plugging $F_{\mathcal{N}}[\mathbf{w}]$ and $F_{\mathcal{N}}[\mathbf{f}]$ into integrals $A(w, w)$ and $L(f, w)$ of (4.15), where we define $\mathbf{f} = G_{\mathcal{N}}(f)$, yielding

$$\begin{aligned} A(F[\mathbf{w}], F[\mathbf{w}]) &= \int_{\Omega} \frac{1}{2} d \nabla(F[\mathbf{w}]) \cdot \nabla(F[\mathbf{w}]) + \frac{1}{2} \kappa F[\mathbf{w}]^2 \, dx, \\ L(F[\mathbf{f}], F[\mathbf{w}]) &= \int_{\Omega} F[\mathbf{f}] F[\mathbf{w}] \, dx, \end{aligned} \quad (4.16)$$

where we have dropped the subscript \mathcal{N} for convenience. In order to write

$$\frac{1}{2} \mathbf{w}^T \hat{A} \mathbf{w} = A(F[\mathbf{w}], F[\mathbf{w}]) \quad \text{and} \quad \mathbf{f}^T \hat{L} \mathbf{w} = L(F[\mathbf{f}], F[\mathbf{w}]),$$

for certain matrices \hat{A} and \hat{L} , we define linear operators $P_i: \mathbb{R}^N \rightarrow \mathbb{R}^{n_i}$, such that for an arbitrary node function $\mathbf{w} \in \mathbb{R}^N$, $P_i \mathbf{w} \in \mathbb{R}^{n_i}$ equals the node function restricted to \mathcal{N}_i (with the ordering inherited from \mathcal{N}).

If we denote $\mathcal{N}_i = \{\mathbf{x}_1^i, \dots, \mathbf{x}_{n_i}^i\}$, and define the matrix $B_i = (\mathbf{b}(\mathbf{x}_1^i) | \dots | \mathbf{b}(\mathbf{x}_{n_i}^i))$, we can write the least squares approximation $p_{\Omega_i}[\mathbf{w}]$ on Ω_i as

$$p_{\Omega_i}[\mathbf{w}](\mathbf{x}) = [(B_i B_i^T)^{-1} B_i P_i \mathbf{w}] \cdot \mathbf{b}(\mathbf{x}). \quad (4.17)$$

Let us write $\tilde{B}_i = (B_i B_i^T)^{-1} B_i$, so that we have $p_{\Omega_i}[\mathbf{w}](\mathbf{x}) = \tilde{B}_i P_i \mathbf{w} \cdot \mathbf{b}(\mathbf{x})$. Taking the gradient of such a function will result in an expression like

$$\nabla(p_{\Omega_i}[\mathbf{w}])(\mathbf{x}) = \begin{pmatrix} D_1^T \tilde{B}_i P_i \mathbf{w} \cdot \mathbf{b}(\mathbf{x}) \\ D_2^T \tilde{B}_i P_i \mathbf{w} \cdot \mathbf{b}(\mathbf{x}) \end{pmatrix},$$

where $D_{1,2}$ are the 6×6 -matrices defined in Section 4.2, yielding

$$\begin{aligned} A(F[\mathbf{w}], F[\mathbf{w}]) &= \sum_i \int_{\Omega_i} \frac{1}{2} d \|\nabla(p_{\Omega_i}[\mathbf{w}])\|^2 + \frac{1}{2} \kappa (p_{\Omega_i}[\mathbf{w}])^2 \, dx \\ &= \sum_i \int_{\Omega_i} \left\{ \frac{1}{2} d \mathbf{w}^T P_i^T \tilde{B}_i^T \left(D_1 \mathbf{b}(\mathbf{x}) \mathbf{b}(\mathbf{x})^T D_1^T + D_2 \mathbf{b}(\mathbf{x}) \mathbf{b}(\mathbf{x})^T D_2^T \right) \tilde{B}_i P_i \mathbf{w} \right. \\ &\quad \left. + \frac{1}{2} \kappa \mathbf{w}^T P_i^T \tilde{B}_i^T \mathbf{b}(\mathbf{x}) \mathbf{b}(\mathbf{x})^T \tilde{B}_i P_i \mathbf{w} \right\} dx. \end{aligned}$$

By defining $I_{\Omega_i} = \int_{\Omega_i} \mathbf{b}(\mathbf{x})\mathbf{b}(\mathbf{x})^T d\mathbf{x}$, this can be written as

$$\begin{aligned} A(F[\mathbf{w}], F[\mathbf{w}]) &= \frac{1}{2} \mathbf{w}^T \left(\sum_{i=1}^N P_i^T \tilde{B}_i^T (d(D_1 I_{\Omega_i}, D_1^T + D_2 I_{\Omega_i}, D_2^T) + \kappa I_{\Omega_i}) \tilde{B}_i P_i \right) \mathbf{w} \\ &= \frac{1}{2} \mathbf{w}^T \left(\sum_{i=1}^N P_i^T M_i^A P_i \right) \mathbf{w}, \end{aligned} \quad (4.18)$$

where we have written $M_i^A = \tilde{B}_i^T (d(D_1 I_{\Omega_i}, D_1^T + D_2 I_{\Omega_i}, D_2^T) + \kappa I_{\Omega_i}) \tilde{B}_i$, which is an $n_i \times n_i$ -matrix. Therefore $\hat{A} = \sum_{i=1}^N P_i^T M_i^A P_i$. The $N \times N$ -matrix $P_i^T M_i^A P_i$ is a large sparse matrix with the entries of matrix M_i^A put at locations dependent on the location of the nodes \mathcal{N}_i in the ordering of \mathcal{N} .

In a similar way we have that

$$L(F[\mathbf{f}], F[\mathbf{w}]) = \mathbf{f}^T \left(\sum_{i=1}^N \tilde{P}_i^T \tilde{B}_i^T I_{\Omega_i} \tilde{B}_i P_i \right) \mathbf{w} = \mathbf{f}^T \left(\sum_{i=1}^N \tilde{P}_i^T M_i^L P_i \right) \mathbf{w}, \quad (4.19)$$

implying $\hat{L} = \sum_{i=1}^N \tilde{P}_i^T I_{\Omega_i} \tilde{B}_i$ so that \hat{L} has the same sparsity structure as \hat{A} . Further, I_{Ω_i} is a symmetric matrix for every node i and therefore \hat{A} and \hat{L} are also symmetric. Our continuous minimization problem (4.15) has now been translated into a discrete minimization problem

$$\hat{K}[\mathbf{u}] = \min_{\mathbf{w} \in \mathbb{R}^N} \hat{K}[\mathbf{w}], \quad \text{with} \quad \hat{K}[\mathbf{w}] = \frac{1}{2} \mathbf{w}^T \hat{A} \mathbf{w} - \mathbf{f}^T \hat{L} \mathbf{w}. \quad (4.20)$$

4.3.2 Solving the discrete problem

If we assume that \hat{A} is invertible, then we can rewrite $\hat{K}[\mathbf{w}]$ as

$$\hat{K}[\mathbf{w}] = \frac{1}{2} (\mathbf{w} - \hat{A}^{-1} \hat{L} \mathbf{f})^T \hat{A} (\mathbf{w} - \hat{A}^{-1} \hat{L} \mathbf{f}) - \mathbf{f}^T \hat{L}^2 \mathbf{f}. \quad (4.21)$$

The matrix \hat{A} is positive semi-definite because $\mathbf{v}^T \hat{A} \mathbf{v} = 2A(F[\mathbf{v}], F[\mathbf{v}]) \geq 0$. Being also invertible would give positive-definiteness, which means that $\hat{K}[\mathbf{w}]$ is minimal for $\mathbf{w} = \mathbf{u}$, with $\mathbf{u} = \hat{A}^{-1} \hat{L} \mathbf{f}$ and $\hat{K}[\mathbf{u}] = -\mathbf{f}^T \hat{L}^2 \mathbf{f}$.

Invertibility of \hat{A}

If the linear function $F: \mathbb{R}^N \rightarrow L^1(\Omega)$ is injective, meaning that $\dim(\text{Im}(F)) = N$, then \hat{A} must be invertible. If so we can define a norm on \mathbb{R}^N by $\|\mathbf{v}\| \equiv \|F[\mathbf{v}]\|_{L^1}$. In finite dimension this norm is equivalent to the standard norm $\|\cdot\|_2$, meaning that there must be a constant $C_{\mathcal{X}} > 0$, such that for arbitrary $\mathbf{v} \in \mathbb{R}^N$, $\|F[\mathbf{v}]\|_{L^1} \geq C_{\mathcal{X}} \|\mathbf{v}\|_2$.

This yields

$$\begin{aligned} \mathbf{v}^T \hat{A} \mathbf{v} &= 2A(F[\mathbf{v}], F[\mathbf{v}]) = 2 \sum_{i=1}^N \int_{\Omega_i} d|\nabla F[\mathbf{v}]|^2 + \kappa F[\mathbf{v}]^2 dx \\ &\geq 2\kappa \sum_{i=1}^N \int_{\Omega_i} F[\mathbf{v}]^2 dx \geq 2\kappa \int_{\Omega} dx \|F[\mathbf{v}]\|_{L^1}^2 \geq (2\kappa C_N^2 \int_{\Omega} dx) \mathbf{v}^T \mathbf{v}. \end{aligned}$$

Therefore all eigenvalues of \hat{A} are greater than zero and \hat{A} is invertible.

In this chapter we will take a practical approach and assume from now on that F is injective (and do not examine under which conditions this is true.) If, given a certain set of nodes, the choice of the local nodes sets is such that \mathcal{N}_i consists of precisely 6 points and $\det(\frac{1}{n} B_i B^T) \neq 0$, then every local approximation multinomial is the unique interpolation multinomial of the six nodes and their function values. In this case F will be certainly injective. In the cases we will consider, the number of nodes used in the local nodes sets will be slightly over 6 and we didn't encounter any non-invertible \hat{A} .

Meshfree-ness

At this point one could wonder whether we should call the method meshfree as the discretization involves Voronoi diagrams. Here we persist in classifying the method as such, because the basic idea of the method is to build a discretization on an arbitrary set of nodes, which is taken as starting point because it will facilitate refinement around the localized sources. Also, despite the use of Voronoi diagrams for the selection of neighboring nodes, building local approximations around nodes does not need a mesh in itself. There are other ways to define neighboring nodes of a certain node. But having used the diagram for this purpose, it also perfectly serves to glue the local approximations together to form a global approximation which is used to build the discrete operators. Clearly this last step makes that the method cannot be characterized as truly meshfree, although its main ideas are of meshfree nature.

4.4 Practical considerations

In this section we will treat some practical issues encountered in calculating and solving the discrete minimization problem.

4.4.1 Calculation and storage of Voronoi diagrams

For calculation of the Voronoi diagram we use the sweep algorithm of Fortune [11], which has a complexity of $\mathcal{O}(N \log N)$. Some code is available for this, but because we need to store the result in some other way than just a list of edges, a new code has been written, which uses the sweep-line algorithm but stores the diagram in a suitable data structure which is called the 'doubly-connected edge list'.

In such a data structure it is easy to find all neighboring edges, vertices and faces of a face. Because in our case a face is just a Voronoi tile and every tile represents exactly one node, we can find neighboring nodes in this way and also a list of vertices, making up the polygon which describes the Voronoi tile of a node. The algorithms involved in finding these neighboring elements are of complexity $\mathcal{O}(N^0)$. More on doubly-connected edge lists can be found in the book [7].

4.4.2 Integrals of multinomials on polygons

When for a given node i the neighboring nodes are found we have to calculate the matrices $(B_i B_i^T)^{-1} B_i$ in equation (4.17). To find the matrices M_i^A and M_i^L also the calculation of I_{Ω_i} is required.

To see how this can be done we assume that we want to integrate a function $f: \mathbb{R}^2 \rightarrow \mathbb{R}$, with $f(x, y) = x^n y^m$, on a polygon given by the points \mathbf{r}_j , $j = 1, \dots, N$, describing in counter clockwise order the polygon Ω_i . (Here the N is different from the one used earlier, which denoted the total number of nodes.) If we define

$$F(x, y) = \begin{pmatrix} x^{n+1} y^m \\ \frac{x^{n+1} y^m}{n+1} \\ 0 \end{pmatrix}, \quad \nabla \cdot F(x, y) = f(x, y).$$

then, by using Gauss' divergence theorem,

$$\int_{\Omega_i} f \, d\mathbf{x} = \int_{\Omega_i} \nabla \cdot F \, d\mathbf{x} = \int_{\partial\Omega_i} F(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) \, d\mathbf{x} = \sum_{j=1}^N \int_{\partial\Omega_i^j} F(\mathbf{x}) \cdot \mathbf{n}_j \, d\mathbf{x},$$

where $\partial\Omega_i^j$ is the line segment $\mathbf{r}_j \mathbf{r}_{j+1}$ for $j \leq N$, $\partial\Omega_i^N = \mathbf{r}_N \mathbf{r}_1$, and \mathbf{n}_j is the normal vector pointing outward. When using the parameterizations $\gamma_j: [0, 1] \rightarrow \mathbb{R}^2$, with $\gamma_j(t) = \mathbf{r}_j + t\Delta\mathbf{r}_j$, where $\Delta\mathbf{r}_j = \mathbf{r}_{j+1} - \mathbf{r}_j$, we have $\|\dot{\gamma}_j'(t)\| = \|\Delta\mathbf{r}_j\|$ and $\mathbf{n}_j = \frac{1}{\|\Delta\mathbf{r}_j\|} [\Delta y_j, -\Delta x_j]^T$, implying

$$\int_{\partial\Omega_i^j} F(\mathbf{x}) \cdot \mathbf{n}_j \, d\mathbf{x} = \int_0^1 \begin{pmatrix} x_j(t)^{n+1} y_j(t)^m \\ \frac{x_j(t)^{n+1} y_j(t)^m}{n+1} \\ 0 \end{pmatrix} \cdot \begin{pmatrix} \Delta y_j \\ -\Delta x_j \end{pmatrix} dt.$$

Consequently,

$$\int_{\Omega} x^n y^m \, d\mathbf{x} = \sum_{j=1}^N \frac{\Delta y_j}{n+1} \int_0^1 (x_j + t\Delta x_j)^{(n+1)} (y_j + t\Delta y_j)^m dt.$$

All entries of I_{Ω_i} have this form and they differ only in the choice of n and m .

4.4.3 Solving the linear system

Once we have built the matrices \hat{A} and \hat{L} , using a sparse matrix structure, we have to solve the system

$$\hat{A}\mathbf{u} = \hat{L}\mathbf{f}.$$

Although the matrices \hat{A} and \hat{L} are sparse, they do not have a band structure in general. Due to the use of arbitrary nodes, the non-zero entries are actually scattered throughout the whole matrix. For solving the system by using an LU -decomposition, it would be advantageous to have a band structure, because then the number of non-zeros in the L and U matrices is also limited [16].

Fortunately, the sweep-line algorithm for calculation of the Voronoi diagram makes use of an ordering of the nodes that can be used here, because it renders our matrices into band structured matrices. The ordering is a lexicographical ordering, where the nodes are ordered on the basis of their coordinates such that

$$(x_1, y_1) \leq (x_2, y_2) \iff y_1 \leq y_2 \quad \text{or} \quad x_1 \leq x_2 \text{ and } y_1 = y_2.$$

Because nodes used in a local approximation are close to each other, the matrices will have the desired band structure after applying this ordering. In Figure 4.4 we see the sparsity structure of the matrix \hat{A} for the node set of the left picture, using the original ordering (middle picture) and the ordering used by the sweep-line algorithm (right picture). In the original ordering the interior points are chosen randomly, while the boundary nodes are ordered along the boundary. This gives the pattern in the matrix as can be seen in the middle picture.

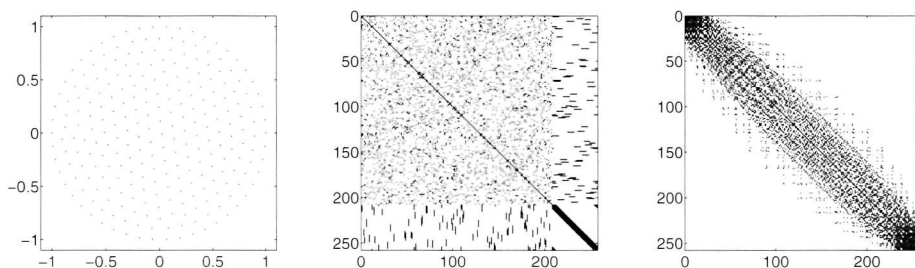


Figure 4.4: (Left) Node set \mathcal{N} consisting of 257 nodes. (Middle) Sparsity structure of \hat{A} and \hat{L} with original ordering of \mathcal{N} . (Right) Sparsity structure of \hat{A} and \hat{L} with ordering of \mathcal{N} used by the Voronoi diagram calculation. The number of non-zero entries equals 5409.

4.5 Choosing nodes in the domain

In this section we will discuss an algorithm for choosing nodes in the domain. As said in the introduction we will work with very diverse domain geometries. Starting with a domain of which the boundary is piecewise smooth and which could have some holes, we need to put nodes in its interior and on the boundary.

Putting nodes on the boundary is relatively easy if we assume that the boundary is given by some parametrization. Given a number of nodes, we can distribute them

over the boundary, possibly taking into account the curvature, so that to regions with high curvature comparatively many nodes are being assigned.

To put nodes in the interior is harder because testing whether a node is inside or outside a certain domain can be expensive in case of complicated geometries. For example, the method described in [15] requires that for every node an integral over the boundary has to be calculated. This seems to be expensive if some kind of adaptation is involved and for every time step a new set of nodes is needed. Aside from this problem, there is also the problem of making a distribution of nodes that suits well function approximation and solving PDEs. As it turns out, the set of nodes one gets by randomly distributing points into a domain will show a lot of clustering, which is not optimal for function approximation and which will give also ill-conditioning problems when used for solving PDEs. Another issue is that we want to be able to impose some variation in the local node density, so that sufficiently many nodes are used in the neighborhoods of sources and on the boundaries and fewer at some distance of them.

4.5.1 Lloyd's method

We will now discuss an algorithm to put nodes into the interior, assuming that there are nodes placed on the boundary already. The boundary nodes are connected by straight lines, transforming our domain into one which has a polygonal boundary. The first step is to find a rectangular region which lies entirely in the interior of the domain. The user of the algorithm has to find it by inspection of the domain.

In this rectangular region nodes are assigned in an arbitrary way. The rest of the algorithm consists of an alternating sequence of the following two steps.

Step 1. *Calculation of the Voronoi diagram.* Given the set of nodes, calculate the Voronoi diagram. The tiles Ω_i at the boundaries are cut off by the straight lines connecting the boundary nodes. This results in a tessellation of the polygonal domain, in which every tile has one node in it. An example has been shown in Figure 4.3. \square

Step 2. *Node replacement using mass centroids.* Given the tessellation of the domain, shift every node to the mass centroid of the tile it is in, except for the nodes on the boundary. The mass centroid of a tile Ω_i is defined as

$$\mathbf{z}_i^{\text{centroid}} = \frac{\int_{\Omega_i} \mathbf{x} d\mathbf{x}}{\int_{\Omega_i} d\mathbf{x}}. \quad \square$$

While alternating these steps the boundary nodes stay on the boundary and the interior nodes stay in the interior. The boundary nodes are fixed and thus also the polygonal boundary.

This algorithm is called Lloyd's method and more about it can be found in [10]. The basic idea behind it is that it tries to make all tiles equally large and spreads out all nodes into the domain while avoiding clustering. One might wonder whether the iteration converges or if the possibility exists that it will run into some cycle. To make this somewhat clearer, we have the following theorem.

Theorem 2. *Suppose we are given an arbitrary disjoint node set $\{\mathbf{x}_i, i = 1, \dots, N\}$ which is bounded by a piecewise linear boundary, consisting of a part of $\{\mathbf{x}_i\}$ connected by straight lines. Then any step in the alternating sequence of calculation of Voronoi diagrams and node replacement using mass centroids minimizes the functional*

$$F(\{\mathbf{x}_i\}, \{\Omega_i\}) = \sum_{i=1}^N \int_{\Omega_i} \|\mathbf{x}_i - \mathbf{x}\|^2 d\mathbf{x}. \quad (4.22)$$

where $\{\Omega_i, i = 1, \dots, N\}$ is a tiling, such that $\mathbf{x}_i \in \Omega_i$ for all $i = 1, \dots, N$. Step 1 chooses Ω_i to minimize (4.22) for fixed \mathbf{x}_i and step 2 chooses \mathbf{x}_i to minimize (4.22) for fixed Ω_i .

Proof. Denote the set bounded by the piecewise linear boundary by \mathcal{D} , resulting in $\mathcal{D} = \cup_i \bar{\Omega}_i$.

Step 1: to prove the assertion for the Voronoi diagram calculation step, we consider an arbitrary $\mathbf{x} \in \mathcal{D}$. Clearly, the contribution of the area around this point to the functional is determined by the value $\|\mathbf{x} - \mathbf{x}_i\|^2$, where \mathbf{x}_i is some node of \mathcal{N} . The fact that $\|\mathbf{x} - \mathbf{x}_i\|$ is minimal over all $i = 1, \dots, N$ by definition of the Voronoi diagram, makes that $\|\mathbf{x} - \mathbf{x}_i\|^2$ is also minimal. Because this can be done for arbitrary $\mathbf{x} \in \mathcal{D}$, the functional is minimal over all possible tessellations of \mathcal{D} .

Step 2: the assertion can be proved by considering $\int_{\Omega_i} \|\mathbf{z} - \mathbf{x}\|^2 d\mathbf{x}$ for a tile Ω_i and some arbitrary $\mathbf{z} \in \mathbb{R}^2$. To minimize the integral we can set the gradient with respect to \mathbf{z} equal to zero. This results in

$$\int_{\Omega_i} (\mathbf{z} - \mathbf{x}) d\mathbf{x} = 0 \implies \frac{\int_{\Omega_i} \mathbf{x} d\mathbf{x}}{\int_{\Omega_i} d\mathbf{x}}. \quad (4.23)$$

which is exactly the definition of the mass centroid. \square

According to the theorem both steps minimize $F(\{\mathbf{x}_i\}, \{\Omega_i\})$ and therefore during the alternating procedure $F(\{\mathbf{x}_i\}, \{\Omega_i\})$ will be non-increasing. Also, the functional is bounded from below, making the sequence of $F(\{\mathbf{x}_i\}, \{\Omega_i\})$ convergent. As a result no cycling can occur but on the other hand convergence of the node set $\{\mathbf{x}_i\}$ itself is not guaranteed and the found minimal value of F does not need to be a global minimizer. In Figure 4.5 some iterations are shown of a node choosing process which resulted after 200 iterations in the node set shown in the first picture of Figure 4.4.

4.5.2 Adjusting local node density

When using the method of the previous subsection we can get node distributions where neighboring nodes are on a more or less constant distance from each other. To impose some variation in local node density we can use a more general version of the algorithm which makes use of a density function in the evaluation of the centroids [10]. We take a different approach where after replacement of the nodes by the calculated centroids in step 2, we shift them a little. This shift is in the direction where a higher concentration of nodes is needed.

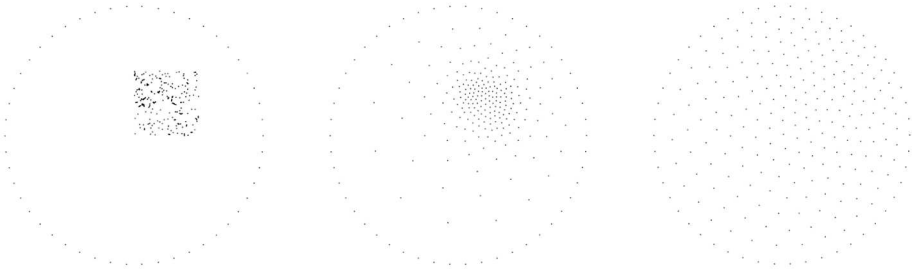


Figure 4.5: A node set at the start, after 10 iterations, and after 100 iterations.

This procedure takes into account that variations in node density should be smooth rather than abrupt. The size of the shift is taken proportional to the size of the tile with respect to the direction of the shift, yielding that the node stays inside the tile. The direction of the shift is determined by attracting neighboring nodes. Given node i and attracting neighboring nodes j , the direction will be close to $\sum_j (\mathbf{x}_j - \mathbf{x}_i)$. The calculation of the shift follows the step of the replacement of the nodes to their mass centroids. To make it more precise we will now give a detailed description of the shifting step.

Step 3. *Node replacement by applying the shift.* Given the tile Ω_i , a set of vertices $\{\mathbf{y}_j\}$ and a set of attracting neighbors $\{\mathbf{x}_{i_k}\}$, first calculate the attracting direction $\mathbf{v} = \sum_k (\mathbf{x}_{i_k} - \mathbf{x}_i)$. Second, determine the minimal and maximal vertices with respect to this direction, i.e., \mathbf{y}_{\min} minimizes and \mathbf{y}_{\max} maximizes $\mathbf{y}_j \cdot \mathbf{v}$. Third, transform the tile using a transformation $(\xi, \eta) \rightarrow (x, y)$,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\mathbf{v} \cdot \mathbf{v}} \begin{pmatrix} v_1 & -v_2 \\ v_2 & v_1 \end{pmatrix} \begin{bmatrix} \mathbf{v} \cdot (\mathbf{y}_{\min} + \frac{1}{c} \log(\xi)(\mathbf{y}_{\max} - \mathbf{y}_{\min})) \\ \eta \end{bmatrix}, \quad (4.24)$$

where $c > 0$ is some constant determining the shift size with respect to the tile size. Finally, for the transformed tile the mass centroid is calculated and the result is transformed back, using the inverse transformation, to give the new location of the node. \square

Figure 4.6 illustrates the process of calculating the shift for an example tile, including the transformation involved. Here $c = 2$ and the direction of \mathbf{v} is given by the arrow in the second picture. The first picture shows the tile and its mass centroid. The second picture adds the coordinate frame of the transformation, while the third picture displays the tile in the transformed coordinate system and its mass centroid with respect to this system. The last picture shows the tile with the original mass centroid and the shifted point.

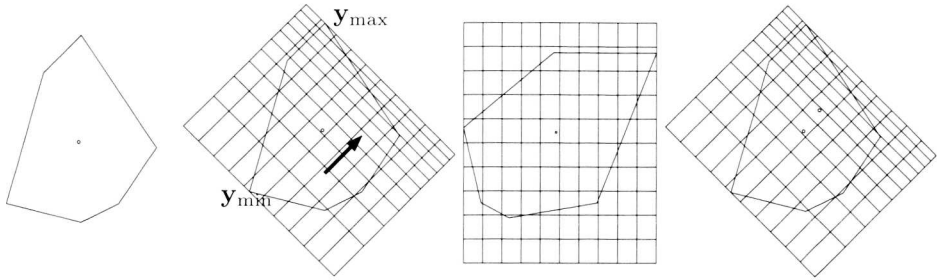


Figure 4.6: Shift calculation by transformation of the tile

Attracting neighboring nodes

To determine how the nodes attract each other all nodes are classified by some integer value. Attraction can then be implemented by defining the attracting neighboring nodes of a node as the neighbors that have a integer value which is higher than their own integer value.

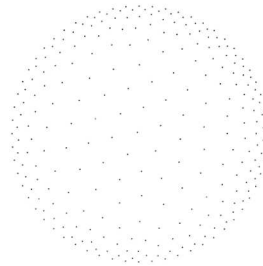


Figure 4.7: Example of refinement near the boundary

Lets us for example assume that we would like to have refinement near the boundary. Then all boundary nodes could be classified by 2, all neighbors of boundary nodes by 1, and the rest by 0. By cycling through the steps: 1. calculation of the Voronoi diagram, 2. giving every node a type, 3. calculation of mass centroids, 4. calculation of shifts, the global nodes would gradually change in a node set which has some refinement near the boundary. The parameter c specifies the maximal spatially decay in the distances between the nodes in a refinement area. The number of different types specifies the size of a refinement area.

In Figure 4.7 a refinement near the boundary is achieved by classifying boundary nodes as 3, their neighbors as 2 and their neighbors' neighbors as 1.

Standard meshing algorithms

One might wonder how the node choosing algorithm described here compares to standard FEM meshing algorithms, like for example the algorithms implemented in Shewchuk's software package Triangle [41]. There, Delaunay triangulation algorithms are being used together with rules on how to deal with holes and how to ensure certain angle and area properties of the produced triangles. Our method seems conceptually simpler than such a method, especially when also some kind of adaptivity is involved.

Although we could use software like Triangle with a standard FEM approach for our problems, we are searching for a method that is focussed on the presence of moving sources, which requires a well-defined form of adaptivity. The combination of the described meshfree method and node choosing algorithm results in a method that gives refinement around the sources in a relatively straightforward way and provides adaptivity at the same time. While meshing algorithms need besides their refinement techniques also rules that specify how coarsening takes place in case of adaptivity, this method moves the nodes along with the moving sources ensuring refinement around each of them during the process. How our method compares to standard meshing algorithms from the perspective of efficiency is an object of current research.

4.6 Numerical tests

In this section we will carry out two numerical tests. We will start with a convergence test on the unit circle where we use a uniform distribution of nodes. I.e., after inserting the nodes randomly in the domain we use the iteration procedure from Section 4.5 with a constant density. We calculate the solution of the elliptic problem (4.14) for the source function

$$f(r, \theta) = \frac{2\pi}{r(\pi^2 - 4)} \cos(\frac{1}{2}\pi r) \left((\pi^2 + 1)r \cos(\frac{1}{2}\pi r) + \pi \sin(\frac{1}{2}\pi r) \right),$$

with r and θ polar coordinates. With $D = 1$ and $\kappa = 1$ the exact solution is

$$u(r, \theta) = \frac{\pi}{\pi^2 - 4} \left(2 \cos^2(\frac{1}{2}\pi r) + \pi^2 \right).$$

Figure 4.8 shows both the source function (left) and the solution (right).

For the test the solution is calculated thirty times. The number of nodes is increased every time, such that the maximal distance between two neighboring nodes Δ_{\max} will vary gradually between 0.2 and 0.02. The maximal local radius h , used in the convergence analysis, will be around twice this distance and will therefore also change with a factor 10. The number of nodes used varies between 106 and 9226.

For each numerical solution we computed the error $\mathbf{e} = \mathbf{u}_{\text{num}} - \mathbf{u}_{\text{exact}}$, its L^2 -norm $\|\mathbf{e}\|_{L^2} = (\mathbf{e}^T \hat{L} \mathbf{e})^{1/2}$, and its maximum error $\|\mathbf{e}\|_{\infty}$. Figure 4.9 shows the results of the test and both errors display a second order convergence.

In the second test we calculate the solution of equation (4.14), where the domain is the unit circle with a hole in it. The source function is formed by two narrow peaks

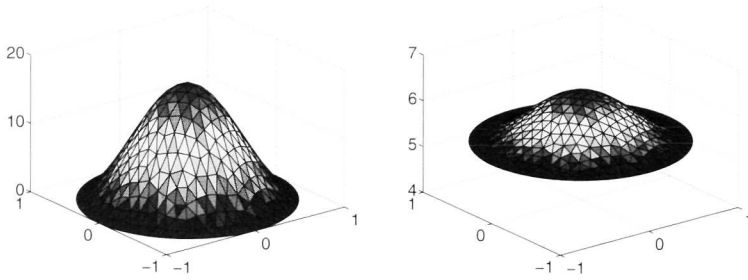
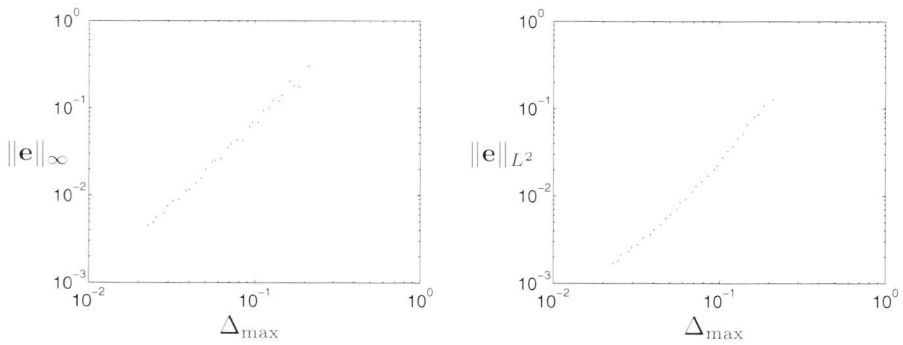


Figure 4.8: source function and solution of equation (4.14)

Figure 4.9: Maximum norm and L^2 -norm errors against Δ_{\max} .

somewhere in the domain. The peaks are circle symmetric with a circular support of radius of $\ell = 0.02$, inside of which they are given by

$$f(r, \phi) = \frac{2\pi}{\ell^2(\pi^2 - 4)} \cos^2\left(\frac{1}{2\ell}\pi r\right),$$

with (r, ϕ) being polar coordinates centered at the location of the peak. Again we take $D = 1$ and $\kappa = 1$.

The refinement strategy is used to put a high concentration of nodes in the neighborhood of the peaks and near the boundaries. With 1890 nodes in total this yields the left picture in Figure 4.10. The right picture shows a magnification around the support of one of the peaks. In such a circular support 70 nodes are being used. To determine the nodes, first the nodes for the peaks and the boundary nodes are determined, after which they are fixed. Then the other nodes are added and the node shifting iterations are done. Here the nodes for the peaks are surrounded with eight rings of attracting nodes, while for the boundaries three and five rings are used, respectively. In Figure 4.11 the numerical solution is shown. For the integral of the

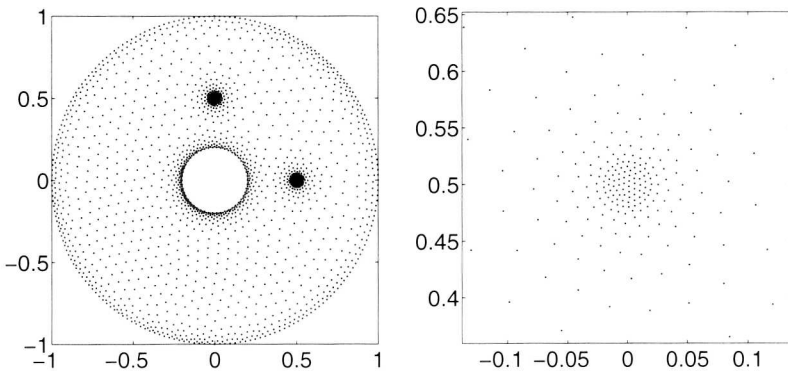


Figure 4.10: (Left) Node distribution. (Right) Refinement around one of the peaks.

solution we have

$$\int_{\Omega} u(\mathbf{x}) d\mathbf{x} = \int_{\Omega} f(\mathbf{x}) d\mathbf{x} = 2.$$

A second order approximation of this integral is $\mathbf{1}^T \hat{L}\mathbf{u} = 1.9875$, where $\mathbf{1}$ is a vector whose entries are all equal to 1. To fill the domain with nodes such that the node density would be equal to the node density as it is in the peak support, would require over 100.000 nodes. We did a similar experiment with peak widths ten times as small as in the test under consideration, but with the same number of nodes. With the number of rings of attracting nodes changed from 8 to 13, the result was $\mathbf{1}^T \hat{L}\mathbf{u} = 1.96$. In that case a node distribution with a uniform node density would require over 10 million nodes.

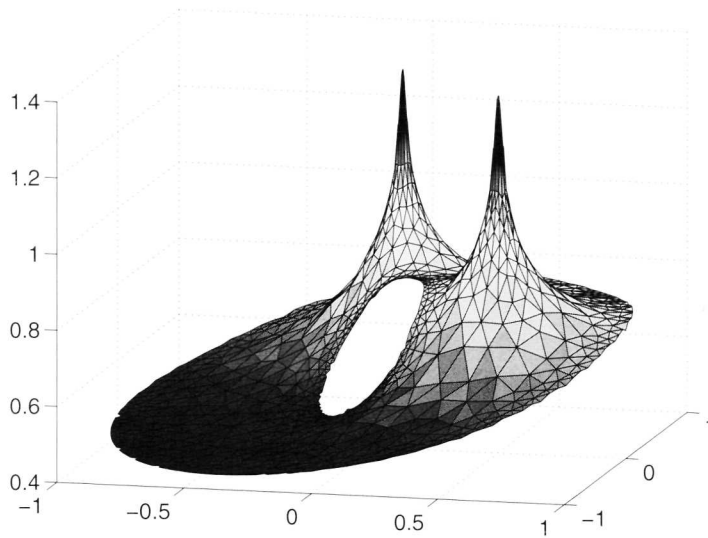


Figure 4.11: Numerically computed peaks

4.7 Summary

In this chapter we developed a meshfree method for solving time-discrete diffusion equations that arise in equation systems used in models from brain research. Important criteria for a suitable method are flexibility with respect to domain geometry and easy refinement possibilities. Both criteria are met when using a meshfree method. The two main results of this chapter are a meshfree discretization of the modified Helmholtz operator and a node choosing algorithm that allows for easy placement of nodes into a given domain while varying node density. Both the discretization and the node choosing algorithm use a Voronoi diagram based on the given node set.

The meshfree discretization uses a Voronoi diagram for finding neighboring nodes of a node and for approximation of an integral on the domain. It is based on a local least squares approximation and the minimization problem in H^1 that is related to the modified Helmholtz equation in combination with the boundary conditions. The minimization problem is discretized by using node functions instead of elements of H^1 . The node choosing algorithm uses a Voronoi diagram for shifting nodes in the right direction. Here the final node distribution tends to be optimal in a certain sense. During the algorithm the nodes repel each other, thereby resulting in some kind of uniformity.

The local least squares approximation underlying the discretization uses a finite number of nodes, called the local node set, to determine a local approximation of a

function. Its convergence in the maximum norm is of second order in the diameter of the local node set, provided that the quality of this set is sufficient. Here the quality is being measured by a determinant based on the set. Numerical experiments show that, when using the node choosing algorithm, the numerical solution of the diffusion equation converges in second order in the maximal diameter of all used local node sets.

An example has been given to show the domain flexibility and refinement possibilities of the node choosing algorithm. Here the method is applied to the modified Helmholtz equation on a circular domain with a hole in it and a source function with very small support compared to the domain.

Chapter 5

A mathematical framework for modelling axon guidance

5.1 Introduction

The proper functioning of the nervous system relies on the formation of correct neuronal connections. During development, neurons project long, thin extensions, called axons, which grow out, often over long distances, to form synaptic connections with appropriate target cells. Axons can find their target cells with remarkable precision by using molecular cues in the extracellular space (for reviews, see Tessier-Lavigne and Goodman [48]; Dickson [8]; Yamamoto et al. [53]). They steer axons by regulating cytoskeletal dynamics in the growth cone (Huber et al. [25]), a highly motile and sensitive structure at the tip of a growing axon. Extracellular cues can either attract or repel growth cones, and can either be relatively fixed or diffuse freely through the extracellular space. Target cells secrete diffusible attractants and create a gradient of increasing concentration, which the growth cone can sense and follow (Goodhill [17]). Cells that the axons have to avoid or grow away from produce repellents. By integrating different molecular cues in their environment, growth cones guide axons along the appropriate pathways and via intermediate targets to their final destination, where they stop growing and form axonal arbors to establish synaptic connections. The responsiveness of growth cones to guidance cues is not static but can change dynamically during navigation. Growth cones can undergo consecutive phases of desensitization and resensitization (Ming et al. [35]), and can respond to the same cue in different ways at different points along their journey (Shirasaki et al. [42]; Zou et al. [56]; Shewan et al. [40]). Through modulation of the internal state of the growth cone, attraction can be converted to repulsion and vice versa (Song et al. [43]; Song and Poo [44]).

Axon guidance is a very active field of research. Several families of molecules have been identified and a few general mechanisms can account for many guidance

phenomena. The major challenge is now to understand, not only qualitatively but also quantitatively, how these molecules and mechanisms act in concert to generate the complex patterns of neuronal connections in the nervous system.

To address this challenge, experimental work needs to be complemented by modelling studies. Unlike for the study of electrical activity in neurons and neuronal networks (e.g., NEURON: Hines and Carnevale [22]), however, there are currently no general simulation tools available for axon guidance.

In Hentschel and Van Ooyen [21] a model is presented in which growing axons on a plain are modelled by means of differential equations for the locations of the growth cones. These equations are coupled to diffusion equations that describe the concentration fields of diffusible chemoattractants and chemorepellents (henceforth referred to as guidance molecules). The system is simplified by using quasi-steady-state approximations for the concentration fields. This approach turns the problem of solving a system consisting of PDEs (partial differential equations) plus ODEs (ordinary differential equations) into a much simpler problem where only ODEs have to be solved. This works fine if the whole plain is used as a domain for the diffusion equations, but we also want to be able to consider more general domains with, for example, areas where diffusion cannot take place ("holes") or with boundaries. Also, Krottje ([28], Chapter 2) showed that in Hentschel and Van Ooyen's approach moving growth cones that secrete diffusible guidance molecules upon which they respond themselves causes the speed of growth to be strongly dependent on the diameter of the growth cone (a phenomenon that was called self-interaction). Using a quasi-steady state approximation will then result in heavily distorted dynamics.

Here we present a general framework for the simulation of axon guidance together with novel numerical methods for carrying out the simulations. The two major ingredients of the modelling framework are the concentration fields of the guidance molecules and the finite-dimensional state vectors representing the growth cones and target neurons. For the latter two, ODEs must be constructed that describe the interaction with the concentration fields. The dynamics of the fields is described by diffusion equations, where we allow for domains with holes or internal boundaries.

Numerical difficulties arise from small, moving sources for the diffusion equations (see Krottje [28], Chapter 2) and from the time integration of a system that is a combination of highly nonlinear, non-stiff ODEs and stiff diffusion equations (see Verwer and Sommeijer [50]). To circumvent this last difficulty we consider the use of quasi-steady-state approximations, and we will discuss some criteria on the validity of such approximations.

The organization of the chapter is as follows. We start with a description of the simulation framework in Section 2. In Section 3 we will discuss some features of the underlying mathematical model and in Section 4 the numerical methods are discussed. Some simulation examples are given in Section 5. We will finish with a discussion in Section 6.

5.2 Simulation framework

In this section we will describe a modelling framework that can be used to model axon guidance. In the models that can be defined within this framework one can incorporate different biological processes and mechanisms, some of which are displayed in Figure 5.1. From a mathematical perspective the framework consists of states, fields and their coupling. We will now discuss these components and their biological interpretation, as well as show how they are related through the model equations.

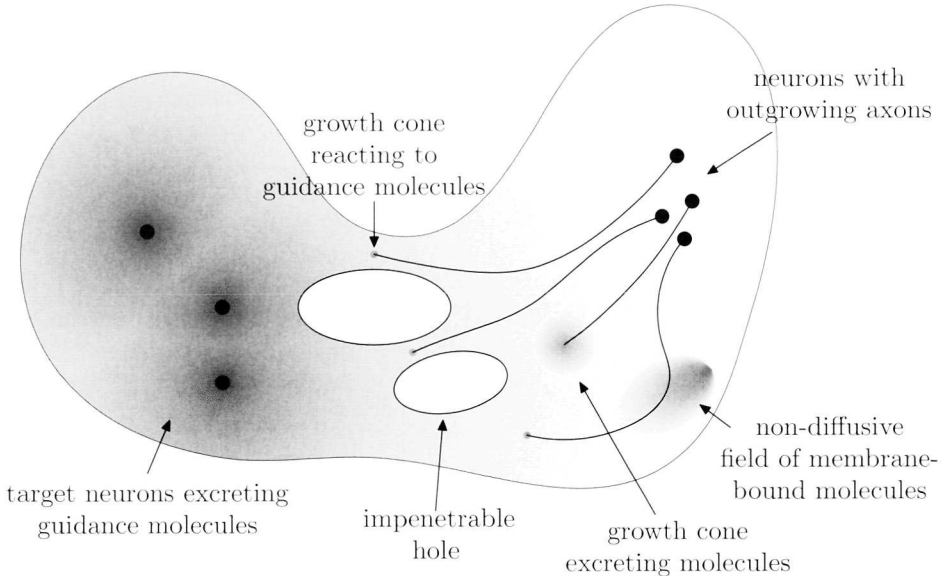


Figure 5.1: Examples of biological concepts that can be incorporated in a model.

States We define states to be finite-dimensional state vectors that represent objects that interact with the concentration fields of guidance molecules. These objects can be, for example, growth cones that move in response to the concentration fields, target neurons that act as sources of guidance molecules, or locations where artificial injection of guidance molecules takes place.

We will assume that the first two variables of the state vector will always represent its 2-dimensional location, which we will denote by \mathbf{r} . Whereas in the model of Hentschel and Van Ooyen a growth cone is completely characterized by its location \mathbf{r} , our description allows for a more general approach in which the state can be extended with a vector \mathbf{s} that further describes the characteristics of the growth cone. Possible characteristics of growth cones and targets that can be modelled with \mathbf{s} are:

Sensitivity Growth cones can respond to different guidance molecules. Their sensitivity to a particular molecule may vary over time (Shewan et al. [40]) and can be influenced by the concentration levels of other guidance molecules as well as by the level of signaling molecules inside the growth cone (Song and Poo [44]).

Growth cone geometry It is known that growth cones can change their size while moving through the environment (Rehder and Kater [38]). The vector \mathbf{s} could model how this process depends on the concentration fields, or it could model the way in which changes in growth cone size change the growth cone's sensitivity or behavior.

Internal state of growth cone Inside a growth cone biochemical reactions take place that determine the growth cone's dynamics (Song et al. [43]; Song and Poo [44]). With \mathbf{s} , the concentrations of the different reactants and their effect on growth cone dynamics and axon guidance can be modelled.

Production rates The rate at which target cells produce guidance molecules may depend on the concentration fields measured at the locations of the targets. The vector \mathbf{s} can be used to describe such dependencies. Alternatively, \mathbf{s} can describe production rates that are given explicitly as functions of time.

For the dynamics of the states we allow for two possibilities. In the first one, the state (\mathbf{r}, \mathbf{s}) is given explicitly as a function of time t and the different concentration levels of guidance molecules ρ_j and their gradients $\nabla\rho_j$ evaluated at position \mathbf{r} ,

$$\begin{pmatrix} \mathbf{r} \\ \mathbf{s} \end{pmatrix} = \begin{pmatrix} G^{\mathbf{r}}(t) \\ G^{\mathbf{s}}(t, \rho_1(\mathbf{r}, t), \nabla\rho_1(\mathbf{r}, t), \dots, \rho_M(\mathbf{r}, t), \nabla\rho_M(\mathbf{r}, t)) \end{pmatrix}. \quad (5.1)$$

In the second possibility an ODE describes the dynamics of the states,

$$\frac{d}{dt} \begin{pmatrix} \mathbf{r} \\ \mathbf{s} \end{pmatrix} = G \left(t, \mathbf{s}, \rho_1(\mathbf{r}, t), \nabla\rho_1(\mathbf{r}, t), \dots, \rho_M(\mathbf{r}, t), \nabla\rho_M(\mathbf{r}, t) \right). \quad (5.2)$$

The functions $G^{\mathbf{r}}$, $G^{\mathbf{s}}$ and G are used to model the different biological processes and mechanisms. We will now discuss the fields ρ_j ($j = 1, \dots, M$).

Fields The fields in our framework represent the concentration fields of the guidance molecules. The dynamics of these fields are determined by diffusion, absorption and some highly localized sources. With ρ the concentration field, d the diffusion coefficient, k the absorption coefficient, and S_{tot} a source term, this results in the diffusion equation

$$\partial_t \rho = d\Delta\rho - k\rho + S_{\text{tot}}, \quad \text{on } \Omega \subset \mathbb{R}^2, \quad \mathbf{n} \cdot \nabla\rho = 0, \quad \text{on } \partial\Omega, \quad (5.3)$$

where the domain Ω may contain several holes (i.e., areas that are impenetrable for guidance molecules) with piecewise smooth boundaries. Thus on the boundary of the

domain we will assume that there is no in- or outflow of guidance molecules. A domain is defined by specifying an outer boundary and possibly several internal boundaries. In our framework all boundaries must be given by parameterizations $\gamma_i: [0, 1) \rightarrow \Omega$.

A number of states is linked to a field. These states determine the total source function S_{tot} , which is the sum of source functions S_i , each of them belonging to a single state $(\mathbf{r}, \mathbf{s})_i$. To further specify the form of the S_i , we make use of a translation operator $T_{\mathbf{y}}$, which can be applied to arbitrary functions $\eta: \Omega \rightarrow \mathbb{R}$ and is defined for $\mathbf{y} \in \Omega$ by $(T_{\mathbf{y}}\eta)(\mathbf{x}) = \eta(\mathbf{x} - \mathbf{y})$ for all $\mathbf{x} \in \Omega$. For the source functions $S_i: \Omega \rightarrow \mathbb{R}$, we make the assumption that $S_i = \sigma_i(\mathbf{s}_i)T_{\mathbf{r}_i}S$. Here, S is some general function profile and $\sigma_i(\mathbf{s}_i) \in \mathbb{R}$ denotes the production rate.

We also allow for the possibility of having fields in steady-state. A reason to incorporate such fields is that the field dynamics might be significantly faster than the dynamics of the growth cones or targets. In this case the fields equation will be

$$d\Delta\rho - k\rho + S_{\text{tot}} = 0, \quad \text{on } \Omega \subset \mathbb{R}^2, \quad \mathbf{n} \cdot \nabla\rho = 0, \quad \text{on } \partial\Omega, \quad (5.4)$$

We will refer to them as quasi-steady-state equations because the source term S_{tot} may depend on time due to time dependent \mathbf{s}_i and \mathbf{r}_i .

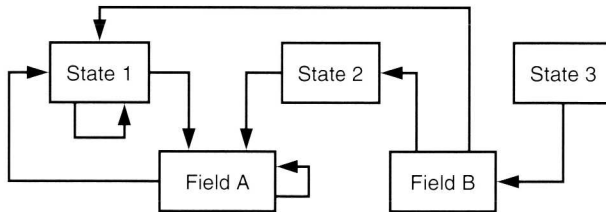


Figure 5.2: Example setting with three dynamic states and two fields.

Coupling The coupling between the states and the fields occurs through the arguments $\rho_j(\mathbf{r}_i)$ and $\nabla\rho_j(\mathbf{r}_i)$ in G and the functions $\sigma(\mathbf{s}_i)$ in S_{tot} . An example of the coupling is depicted in Figure 5.2, where we have three states and two concentration fields. Here an arrow from one object to another means that the dynamics of the latter object depend on the former. For example the dynamics of state 1 is determined by itself and the fields A and B, whereas the dynamics of field A depend on the state

1 and 2. The system of equations in this case might be

$$\begin{aligned}
 \frac{d}{dt} \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{s}_1 \end{pmatrix} &= G_1 \left(t, \mathbf{s}_1, \rho_A(\mathbf{r}_1, t), \nabla \rho_A(\mathbf{r}_1, t), \rho_B(\mathbf{r}_1, t), \nabla \rho_B(\mathbf{r}_1, t) \right), \\
 \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{s}_2 \end{pmatrix} &= \begin{pmatrix} G_2^{\mathbf{r}}(t) \\ G_2^{\mathbf{s}} \left(t, \rho_B(\mathbf{r}_2, t), \nabla \rho_B(\mathbf{r}_2, t) \right) \end{pmatrix} \\
 \begin{pmatrix} \mathbf{r}_3 \\ \mathbf{s}_3 \end{pmatrix} &= G_3(t) \equiv \begin{pmatrix} \mathbf{r}_3^0 \\ \mathbf{s}_3^0 \end{pmatrix}. \\
 \partial_t \rho_A &= d\Delta \rho_A - k\rho_A + \sigma_{A,1}(\mathbf{s}_1)T_{\mathbf{r}_1}S + \sigma_{A,2}(\mathbf{s}_2)T_{\mathbf{r}_2}S, \\
 0 &= d\Delta \rho_B - k\rho_B + \sigma_{B,3}(\mathbf{s}_3)T_{\mathbf{r}_3}S.
 \end{aligned} \tag{5.5}$$

We see that only the dynamics of state 1 depends on the state itself, which is reflected in having an ODE for its dynamics, while the dynamics of the other two states are given in a more explicit form.

5.3 Underlying mathematical model

In the framework, the complete simulation model consists of a number of state vectors $\mathbf{u}_i = (\mathbf{r}_i^T \mathbf{s}_i^T)^T$, $i = 1, \dots, N$, with their dynamics determined by (5.1) or (5.2), together with concentration fields ρ_j , $j = 1, \dots, M$, defined by diffusion equations (5.3) or quasi-steady-state equations (5.4). We assume that for the first M_d fields the dynamics are given by the full diffusion equations and that for the other fields the dynamics are given by quasi-steady-state equations. This results in our system of field equations of the form

$$\partial_t \rho_j = L_j \rho_j + \sum_{i=1}^N \sigma_{ji}(\mathbf{s}_i) T_{\mathbf{r}_i} S, \quad \text{on } \Omega, \quad j = 1, \dots, M_d \tag{5.6}$$

$$0 = L_j \rho_j + \sum_{i=1}^N \sigma_{ji}(\mathbf{s}_i) T_{\mathbf{r}_i} S, \quad \text{on } \Omega, \quad j = M_d + 1, \dots, M \tag{5.7}$$

$$\mathbf{n} \cdot \nabla \rho_j = 0, \quad \text{on } \partial\Omega, \quad j = 1, \dots, M. \tag{5.8}$$

where $L_j = d_j \Delta - k_j$. Here we assume that $S: \Omega \rightarrow \mathbb{R}$ is an L_2 -function with compact support with the property that $\int_{\Omega} S(\mathbf{x}) d\mathbf{x} = 1$. This means that we can interpret the σ_{ji} as the production rate of the source attached to state $(\mathbf{r}, \mathbf{s})_i$ with respect to field ρ_j .

We assume that the dynamics of the first N_o state vectors are given by ODEs, i.e., equations of the form (5.2), and that the dynamics of the other vectors are given explicitly as a function of time and the fields. When we make use of the vector notations $\boldsymbol{\rho}(\mathbf{r}_i)$, $\partial_x \boldsymbol{\rho}(\mathbf{r}_i)$, and $\partial_y \boldsymbol{\rho}(\mathbf{r}_i)$, that are defined by

$$\boldsymbol{\rho}(\mathbf{r}_i)_j = \rho_j(\mathbf{r}_i), \quad \partial_x \boldsymbol{\rho}(\mathbf{r}_i)_j = \partial_x \rho_j(\mathbf{r}_i), \quad \partial_y \boldsymbol{\rho}(\mathbf{r}_i)_j = \partial_y \rho_j(\mathbf{r}_i).$$

this results in

$$\partial_t \mathbf{u}_i = G_i(t, \mathbf{u}_i, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i)), \quad i = 1, \dots, N_o \quad (5.9)$$

$$\begin{pmatrix} \mathbf{r}_i \\ \mathbf{s}_i \end{pmatrix} = \begin{pmatrix} G_i^r(t) \\ G_i^s(t, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i)) \end{pmatrix}, \quad i = N_o + 1, \dots, N. \quad (5.10)$$

In the functions G_i we have to implement the different mechanisms that are involved with the behavior of the growth cones and targets when they measure the levels of particular concentration fields and their gradients. To complete the system we have to add initial conditions for the states \mathbf{u}_i and the fields ρ_j .

Typical parameter ranges Goodhill [18] gives some estimates for the ranges of some relevant parameters. Table 5.1 shows a list with parameter ranges. The ratio of

Table 5.1: Parameter ranges

quantity	symbol	order of magnitude	units
- diffusion constant	d_j	$10^{-5}-10^{-4}$	mm^2/s
- production rate	σ_{ji}	10^{-7}	nMol/s
- minimal concentration	ρ_{\min}	$10^{-2}-10^{-1}$	nMol/l
for gradient detection			
- maximal concentration	ρ_{\max}	100	nMol/l
for gradient detection			
- minimal relative	$L_{\text{cone}} \nabla \rho_j /\rho_j$	0.01-0.02	
detectable gradient			
- growth cone diameter	L_{cone}	$10^{-2}-2 \cdot 10^{-2}$	mm
- growth speed	v	$10^{-6}-10^{-4}$	mm/s
- growth range	L_{path}	$10^{-1}-1$	mm

the maximal and minimal concentration for gradient detection ρ_{\max}/ρ_{\min} can be used together with the diffusion constant d to find an upper bound on the possible values of the absorption parameter k_j . Assume that the ratio ρ_{\max}/ρ_{\min} is $100/10^{-2} = 10^4$ and that we have a point source located at the origin that produces the steady-state field $\rho_s(r)$. Then using the assumptions that the maximal distance over which a cone can be guided L_{path} is 1 mm and the growth cone radius equals 0.005 mm, we find

$$\frac{\rho_s(0.005)}{\rho_s(1.0)} = K_0 \left(0.005 \sqrt{\frac{k_j}{d_j}} \right) / K_0 \left(\sqrt{\frac{k_j}{d_j}} \right) \approx 10^4 \quad \Rightarrow \quad \sqrt{\frac{k_j}{d_j}} \approx 60. \quad (5.11)$$

Here we used an expression for ρ_s that is derived in the Appendix. We can derive a lower bound for the absorption constant k_j by considering the ratio $L_{\text{cone}}|\partial_r \rho_s(r)|/\rho_s(r)$ which decreases with r and increases with k_j . If we assume it to be greater than 0.01, for all $r \leq 1$, this yields a bound $\sqrt{k_j/d_j} \geq 0.60$.

Moving sources Our framework also allows for the possibility that guidance molecules are released by the growth cones themselves, i.e. we allow for moving sources. Although the biological evidence for this is less strong than for the release of guidance molecules by target cells, it is certainly not implausible. Growth cones secrete various chemicals that may operate as chemoattractants and chemorepellents. For example, migrating axons are capable of secreting neurotransmitters [54], which have been implicated as chemoattractants [55]. The treatment of moving sources that respond to guidance molecules they themselves secrete is mathematically challenging and will be dealt with in the Appendix.

Quasi-steady-state approximation When we run a simulation using the whole system (5.6)–(5.10), we should use a time integration technique that is suitable for the stiff diffusion equations in combination with the non-stiff ODEs. If the dynamics of all the diffusion equations are fast compared to the state-dynamics, then it is possible to approximate the ρ_j , $j = 1, \dots, M_d$ with solutions of the steady-state equations

$$0 = L_j \rho_j + \sum_{i=1}^N \sigma_{ji}(\mathbf{s}_i) T_{\mathbf{r}_i} S \quad \text{on } \Omega, \quad j = 1, \dots, M_d. \quad (5.6')$$

The original dynamical system, which had as its dependent variables the states \mathbf{u}_i and the fields ρ_j , is now replaced by a dynamical system that has the \mathbf{u}_i as its dependent variables only. Although the system at hand is therefore reduced from an infinite-dimensional to a finite-dimensional system, evaluation of the right hand side still involves solving a infinite-dimensional system. Determination of the values $\rho_j(\mathbf{r}_i)$ requires solving the equations (5.6')–(5.7). From a numerical perspective the advantage is that we do not need a time integrator that can handle the combination of stiff PDEs and non-stiff ODEs, but we can simply make use of a standard explicit time integrator.

To investigate the validity of such an approximation we will consider a diffusion equation (5.12) and its steady-state approximation (5.13)

$$\begin{cases} \partial_t \rho = d \Delta \rho - k \rho + S, & \text{on } \mathbb{R}^2 \\ \rho(0, \mathbf{x}) = \rho_0(\mathbf{x}) \end{cases} \quad (5.12)$$

$$0 = d \Delta \rho - k \rho + S, \quad \text{on } \mathbb{R}^2. \quad (5.13)$$

Some implications of using an approximation like (5.13) for (5.12) are discussed in ([28], Chapter 2). There the case of self-interaction is considered, meaning that for a particular field a source is attached to a state and the dynamics of the state is determined by the same field. Here we want to consider some more general criteria on when such a quasi-steady-state approximation might be valid for different parameter values of the diffusion rate d , the absorption rate k , and the speed a source moves through the domain v .

Hentschel and Van Ooyen [21] used the approximation on the basis of comparing the time scales of growth and diffusion. Here, however, the absorption parameter

plays also a role. To determine criteria that take also k into account we will follow two approaches. In the first approach we consider the time needed for setting-up a concentration field. In the second approach we compare the concentration profile produced by a point source moving with constant speed with its quasi-steady-state approximation.

Field set-up time To examine how the time for setting up the concentration field depends on d and k , we consider the solution of (5.12) with S being a point source at the origin, $S(\mathbf{x}, t) = \delta(\mathbf{x})$, and an initial field $\rho = 0$ at time $t = 0$. The solution is rotation symmetric, making it dependent on the distance to the source r and the time t only, $\rho(r, t)$. In the Appendix it is shown that it approaches a steady-state solution $\rho(r, \infty)$. To see how fast the field approaches the steady state field, we consider

$$c(r, t) = \frac{\rho(r, \infty) - \rho(r, t)}{\rho(r, \infty)},$$

which represents how close the field is to its limit value. For example, a value $c(r, t) = 0.01$, means that at time t the field is for 99% set up, at location r . In the Appendix we derive

$$c(r, t) \sim \frac{1}{2K_0\left(r\sqrt{\frac{k}{d}}\right)} \frac{e^{-kt}}{kt}, \quad (5.14)$$

where K_0 is a modified Bessel function of the Second Kind [1]. This can be used to get an indication of the time scale of the field dynamics. Such an indicator is important if we want to work with fields of which the sources do not move through the domain. In case of moving sources, one might wonder how the speed of a source influences the produced field. To this end we examine the solution of (5.12) with a point source that moves with constant speed.

Field produced by moving source Consider equation (5.12) with a point source that moves with constant speed v along the x -axis in positive direction. i.e., $S(\mathbf{x}, t) = \delta(\mathbf{x} - \mathbf{v}t)$, with $\mathbf{v} = (v, 0)^T$. In the Appendix it is shown that we get a stable constant profile solution that moves also with constant speed \mathbf{v} .

Here we want to compare how close the quasi steady-state-approximation solution ρ_s is to this moving profile solution ρ_p . It turns out that in the vicinity of the source the moving profile is smaller than the steady-state-approximation, and on approaching the location of the source they tend to become equal. In the Appendix it is derived that

$$r \lesssim 2e^{-\gamma\epsilon} \sqrt{\frac{d}{k}} \left(1 + \frac{v^2}{4dk}\right)^{\frac{1}{2(\gamma-1)}} \implies \frac{\rho_p}{\rho_s} \lesssim \gamma. \quad (5.15)$$

If we choose $\gamma = 0.99$, we get an indication of the size of the region around the source, where the difference between the moving profile and the quasi-steady-state solution is less than 1%, given the values of the diffusion rate d , absorption rate k and moving speed v .

5.4 Numerical methods

In this section we will consider the numerical methods we use for solving the equation systems (5.6)–(5.10). We will start with the spatial discretization for solving the field equations. This will be followed by a description of the time integration techniques.

For solving the field equations we use an unstructured spatial discretization based on an arbitrary set of nodes situated in the domain. This approach facilitates dealing with complex domains, refinement and adaptivity; the latter is needed in cases where we have moving sources with small support. A thorough description of the method can be found in ([29], Chapter 4); we will briefly outline it here.

Function approximation Given function values on the nodes, we use a local least-squares approximation technique to determine for every node a second-order multinomial that is a local approximation of the function around that node. For this we use the function values on a number of neighboring nodes. Because every second-order multinomial can be written as the linear combination of six basis functions, we must choose at least five neighbors for every node to determine such an approximating multinomial.

With this procedure a set of function values is mapped onto a set of local approximations around every node. If we assign to every node a part of the domain for which we assume the local approximation to be valid, such that the whole domain is covered, this results in a global approximation. For a given set of function values in a vector $\mathbf{w} \in \mathbb{R}^N$, we denote the global approximation by $F(\mathbf{w}) \in L_1(\Omega)$, where $L_1(\Omega)$ is the space of integrable real functions defined on $\Omega \subset \mathbb{R}^2$.

Voronoi diagrams For choosing neighboring nodes of nodes, as well as for assigning parts of the domain to the nodes, we use the Voronoi diagram [11]. It assigns to every node a Voronoi cell, which is the set of points closer to the node than to every other node, hence dividing the domain and at the same time creating neighbors in a natural way.

Because a Voronoi diagram extends to all of \mathbb{R}^2 , we will truncate it by connecting the nodes on the boundary by straight lines, resulting in a bounded diagram. From now on all our diagrams will be truncated ones, but we here will still refer to them as Voronoi diagrams. Determination of such a diagram can be done in $\mathcal{O}(N \log(N))$ operations, where N is the number of nodes [7]. We store the diagram in a totally disconnected edge list [7], so that searching neighboring nodes for every node becomes a process of $\mathcal{O}(N)$ operations.

Variational problem Solving equations of the form (5.7) can be done by solving the variational problem of minimizing $A(w, w) - L(S, w)$ over $w \in H^1$ [2, 29], where

$$A(v, w) = \int_{\Omega} \frac{1}{2} d\nabla v \cdot \nabla w + \frac{1}{2} \kappa v w \, dx. \quad (5.16)$$

$$L(S, w) = \int_{\Omega} S w \, dx. \quad (5.17)$$

A direct discretization of this problem is to minimize $A(F(\mathbf{w}), F(\mathbf{w})) - L(S, F(\mathbf{w}))$, for all $\mathbf{w} \in \mathbb{R}^N$. It can be shown ([29], Chapter 4) that sparse matrices \hat{A} and \hat{L} exist such that $\frac{1}{2} \mathbf{w}^T \hat{A} \mathbf{w} = A(F(\mathbf{w}), F(\mathbf{w}))$ and $\mathbf{S}^T \hat{L} \mathbf{w} = L(F(\mathbf{S}), F(\mathbf{w}))$. If \hat{A} is non-singular the discrete problem has a unique solution $\mathbf{w} = \hat{A}^{-1} \mathbf{S}$. With the algorithm for finding the Voronoi diagram comes a lexicographical ordering of the nodes that will give the sparse matrices a band structure, which is advantageous when solving the system directly using an LU-decomposition.

Convergence tests show that the solution is 2nd-order convergent in the L^2 -norm, with respect to the maximum distance between neighboring nodes ([29], Chapter 4).

Choosing nodes To distribute nodes appropriately over a domain we make use of Lloyd's algorithm [10]. This algorithm is based upon the determination of Voronoi diagrams and the process of shifting nodes to centroids of Voronoi cells. An alternating sequence of these two operations distributes the nodes equally over the domain, in the sense that distances between neighbors will tend to become equal throughout the diagram.

To achieve refinement at certain points, we use a variation of Lloyd's algorithm. Here, after shifting the nodes to their centroids, an extra shift in the direction of neighboring nodes is added. To determine for a particular node which of its neighbors are attracting this node, all nodes are given an integer type. Nodes will then be attracted to the neighbors with higher type than their own type.

To get refinement around a certain point in the domain, a node is fixed at that point and several rings of decreasing node type are defined around it. The extended Lloyd's algorithm then moves nodes around, which results in a refinement around the fixed node.

In contrast to methods where refinement is based on local error estimation, here refinement takes place around the source locations. This is done because we know in advance that only at those locations, and possibly at the boundary, refinement is required for optimal accuracy. Doing it this way instead of using an error estimation process will then speed up the refinement process.

Having discussed the spatial discretization method we will now focus on the time integration. We will consider three different cases that can be distinguished by the field dynamics in the model.

Time integration with static fields We first consider the case with static fields only. In this case we only have ODEs which need the solutions of the fields for the evaluation of their right hand sides. These fields are determined at the start of the simulation by solving the elliptic equations, giving the approximations to the field solutions ρ_1, \dots, ρ_M . After this the growth cone dynamics can be solved using a standard explicit ODE solver.

For the fields to be static we need a number of static states that make up the sources of the fields. Let us assume that of all the states only the last N_s are static, i.e., $(\mathbf{r}_i, \mathbf{s}_i) = \text{constant}$, and that the rest of the states do not influence the field dynamics. Thus, we must have

$$\sigma_{ji} \equiv 0, \quad \text{for all } \begin{cases} j = 1, \dots, M, & (\text{all fields}) \\ i = 1, \dots, N - N_s & (\text{all dynamic states}). \end{cases} \quad (5.18)$$

Then given the N_s static positions \mathbf{r}_i , $i > N - N_s$, we have to solve

$$\mathbf{s}_i = G_i^s(\boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i)), \quad i = N - N_s + 1, \dots, N. \quad (5.19)$$

$$L_j \rho_j + \sum_{i=N-N_s+1}^N \sigma_{ji}(\mathbf{s}_i) T_{\mathbf{r}_i} S = 0, \quad \text{on } \Omega, \quad j = 1, \dots, M, \quad (5.20)$$

$$\mathbf{n} \cdot \nabla \rho_j = 0, \quad \text{on } \partial\Omega.$$

This system can be solved by solving first the field equations (5.20). Using the inverse operators of L_j with respect to the boundary conditions, we get

$$\rho_j = - \sum_{i=N-N_s+1}^N \sigma_{ji}(\mathbf{s}_i) L_j^{-1} T_{\mathbf{r}_i} S, \quad \text{on } \Omega. \quad (5.21)$$

When combined with equation (5.19), evaluation of these field solutions and their gradients in the given \mathbf{r}_i , results in a closed algebraic system with respect to \mathbf{s}_i , $\rho_j(\mathbf{r}_i)$, $\partial_x \rho_j(\mathbf{r}_i)$ and $\partial_y \rho_j(\mathbf{r}_i)$. We will assume that this nonlinear system can be solved, although the solvability depends on the \mathbf{r}_i and the functions σ_{ji} .

Therefore, to solve numerically the fields ρ_j we first have to solve numerically the fields $L_j^{-1} T_{\mathbf{r}_i} S$, using the spatial discretization above. After evaluation of these fields (i.e., their numerical approximations) and their derivatives in all locations \mathbf{r}_i the algebraic system can be built by substituting (5.21) into (5.19). We can solve this system by using, for example, Newton iterations and use the \mathbf{s}_i to determine the solutions ρ_j .

Once the fields and static states ($i > N - N_s$) are solved we can start solving the non-static states from the equations

$$\partial_t \mathbf{u}_i = G_i(t, \mathbf{u}_i, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i)), \quad i = 1, \dots, N_o \quad (5.22)$$

$$\begin{pmatrix} \mathbf{r}_i \\ \mathbf{s}_i \end{pmatrix} = \begin{pmatrix} G_i^r(t) \\ G_i^s(t, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i)) \end{pmatrix}, \quad i = N_o + 1, \dots, N - N_s. \quad (5.23)$$

For solving the ODEs we choose an explicit integration scheme because the ODEs are non-stiff (and nonlinear). We will use the classical 4th-order RK (see for example [26]) for this. Note that for the function evaluations in the scheme we have to determine local approximations of the fields *and* their gradients. A slight difficulty arises here because the local least-squares approximations are discontinuous from one Voronoi cell to another. Therefore, if the integration process crosses the edge of a cell during a time step, there will be loss of order with respect to the size of the time step. To prevent this we make sure that during a time step we use for every state only one local field approximation for all function evaluations used in the scheme. Because the local field approximation is a multinomial the order of the scheme will be retained.

Quasi-steady-state approximation When using quasi-steady-state approximations for the fields, the system we have to solve is

$$\partial_t \mathbf{u}_i = G_i(t, \mathbf{u}_i, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i)), \quad i = 1, \dots, N_o \quad (5.24)$$

$$\begin{pmatrix} \mathbf{r}_i \\ \mathbf{s}_i \end{pmatrix} = \begin{pmatrix} G_i^r(t) \\ G_i^s(t, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i)) \end{pmatrix}, \quad i = N_o + 1, \dots, N. \quad (5.25)$$

$$\left. \begin{aligned} L_j \rho_j + \sum_{i=1}^N \sigma_{ji}(\mathbf{s}_i) T_{\mathbf{r}_i} S &= 0, & \text{on } \Omega, \\ \mathbf{n} \cdot \nabla \rho_j &= 0, & \text{on } \partial\Omega. \end{aligned} \right\} \quad j = 1, \dots, M. \quad (5.26)$$

Here we use, as in the previous case, an explicit time integrator for the ODEs in (5.24). To evaluate the right hand side of the equations we need to solve the fields ρ_j for given values of $(\mathbf{r}_i, \mathbf{s}_i)_n$, $i = 1, \dots, N_o$, and t_n , where n denotes the time level. To find these we have to determine the fields again by solving a non-linear algebraic system as is done in the case with static fields. Here, the system will have as its unknowns the $\rho_j(\mathbf{r}_i)$, $\partial_x \rho_j(\mathbf{r}_i)$ and $\partial_y \rho_j(\mathbf{r}_i)$ for all combinations of fields ρ_j and states \mathbf{r}_i , together with all \mathbf{s}_i , for $i > N_o$.

In contrast to the case with static fields, every function evaluation in the right hand side of (5.24) requires solving equations (5.26) and evaluations of the resulting solution fields and their gradients. Also, because the source terms in (5.26) depend on the states \mathbf{u}_i , it may be necessary to redefine the nodes used to solve the field equations. Therefore solving such a system is computationally much more expensive than solving a system with static fields only.

Full system Solving the full system, i.e., equations (5.6)–(5.10) requires a numerical method that can deal with both the nonlinear, non-stiff ODEs and the stiff diffusion equations. Verwer and Sommeijer [50] use for a system similar to the combination of (5.6) and (5.9) the RKC method, which is explicit and can deal with moderately stiff systems due to a long narrow stability region around the negative real axis. Lastdrager [32] used a Rosenbrock method with approximate Jacobians for the same system so that effectively the field equations are integrated implicitly and the state equations explicitly, as with IMEX (IMplicit-EXplicit) methods [26].

We use a Runge-Kutta IMEX scheme, in particular an an IMEX-midpoint scheme, which can be seen to be a combination of an implicit and an explicit midpoint step. For a system $\dot{\mathbf{x}} = f_1(t, \mathbf{x}) + f_2(t, \mathbf{x})$ it is given by

$$\begin{aligned} \mathbf{x}_s &= \mathbf{x}_n + \frac{1}{2}\tau f_1(t_n + \frac{1}{2}\tau, \mathbf{x}_s) + \frac{1}{2}\tau f_2(t_n, \mathbf{x}_n). \\ \mathbf{x}_{n+1} &= 2\mathbf{x}_s - \mathbf{x}_n + \tau \left(f_2(t_n + \frac{1}{2}\tau, \mathbf{x}_s) - f_2(t_n, \mathbf{x}_n) \right), \end{aligned} \quad (5.27)$$

where the s in \mathbf{x}_s refers to the intermediate stage. For our system the part f_1 , which is treated implicitly, contains the linear operators L_i from equation (5.6), while the explicit part f_2 contains the source terms of equation (5.6) and the functions G_i from equation (5.9). This is a second-order time integration method and the implicit part, i.e., the implicit midpoint method, is A-stable. Also, using this scheme for the systems at hand never revealed any stability problems.

In the next section we will show some example models. Although our framework can deal with non-static fields (as discussed earlier), in these examples we will only consider cases in which the fields are static.

5.5 Simulation examples

In this section we will discuss simulations of some example models. We want to stress that the models used here are still simple and only serve to show the different possibilities of our framework. To model the growth cones and the sources of the guidance molecules, such as target cells, we have to choose state vectors $(\mathbf{r}_i, \mathbf{s}_i)$ that characterize these objects and accompanying functions G that describe the dynamics through equations (5.1) and (5.2).

Growth cone model As a first example of a growth cone model we consider growth cones characterized by three-dimensional state vectors. To the position $\mathbf{r}_i = (x, y)$ we add a variable representing the orientation angle $\mathbf{s}_i = \phi \in [0, 2\pi)$ of the growth cone. This gives our model growth cone a growth direction, which it has to adjust in order to steer. It gives the opportunity to build in some kind of ‘stiffness’, the inability to undergo instant changes in growth direction

In order to describe the dynamics of the growth direction we need to define a differential equation. We will assume that the growth speed is constant, given as v , and that the cone grows with this speed in the direction given by the orientation angle ϕ , i.e., $\dot{\mathbf{r}}_i = (v \cos(\phi), v \sin(\phi))$. For the dynamics of ϕ , we assume that it is continuously compared with some ideal direction ϕ_g , which we will assume to be a linear combination of the sensed gradients of the fields ρ_j evaluated at location \mathbf{r}_i ,

$$\phi_g = \arg \left(\sum_{j=1}^M \lambda_j(\rho(\mathbf{r}_i)) \nabla \rho_j(\mathbf{r}_i) \right). \quad (5.28)$$

with 'arg' the function that returns the angle between the argument and the positive x -axis. Here the real functions λ_j determine the sensitivity to each of the fields. A positive λ_j will cause the cone to be attracted by the field ρ_j , while a negative λ_j causes repulsion.

To formulate an ODE for ϕ that depends on the value of ϕ_g , we use the mapping $\phi \rightarrow (\sin(\phi), \cos(\phi))$ to view the growth directions as two-dimensional unit vectors, \mathbf{z} and \mathbf{z}_g , respectively. The ideal direction \mathbf{z}_g can be split in a part parallel to the growth direction \mathbf{z} and a part that is perpendicular to it, $\mathbf{z}_g = \mathbf{z}_{\parallel} + \mathbf{z}_{\perp}$. An illustration of this is shown in Figure 5.3. We assume that $\dot{\mathbf{z}} = (v/\ell)\mathbf{z}_{\perp}$. Returning to angles ϕ and ϕ_g this results in $\dot{\phi} = v/\ell \sin(\phi_g - \phi)$.

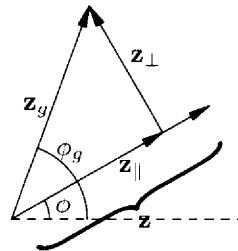


Figure 5.3: An example configuration of the vectors \mathbf{z} and \mathbf{z}_g

Here, the parameter ℓ is a measure for the smallest circle the growth cone can make while turning. This latter fact can be understood by realizing that the maximal value of $\dot{\phi}$ is v/ℓ . If we consider a solution where $\dot{\phi}$ is maximal we get, with $\mathbf{r} = (x, y)$,

$$\frac{d}{dt} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \dot{\phi} \ell \cos(\phi) \\ \dot{\phi} \ell \sin(\phi) \end{pmatrix} \implies \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \ell \sin(\phi(t)) + x_0 \\ -\ell \cos(\phi(t)) + y_0 \end{pmatrix}.$$

meaning that the solution path of $(x(t), y(t))$ is part of the circle with radius ℓ and center (x_0, y_0) . Using the framework the dynamics of state $(\mathbf{r}_i, \mathbf{s}_i)$ are described by

$$\begin{aligned} \frac{d}{dt} \begin{pmatrix} \mathbf{r}_i \\ \mathbf{s}_i \end{pmatrix} &= G_i(t, \mathbf{r}_i, \mathbf{s}_i, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i)) \\ &= \begin{pmatrix} v \cos(\mathbf{s}_i) \\ v \sin(\mathbf{s}_i) \\ v/\ell \sin \left(\arg \left(\sum_{j=1}^M \lambda_j(\boldsymbol{\rho}(\mathbf{r}_i)) \nabla \rho_j(\mathbf{r}_i) \right) - \mathbf{s}_i \right) \end{pmatrix}. \end{aligned} \quad (5.29)$$

Field sources In the examples we will assume that the fields are produced by sources that are not moving and not changing their behavior in time. Therefore it will serve to include in their state vectors only their positions $\mathbf{r}_i \in \mathbb{R}^2$ and keep them constant in time $\mathbf{r}_i = G_i^{\mathbf{r}}(t) \equiv \mathbf{r}_i^0$.

For every source we take a bell shape function S that is translated with \mathbf{r}_i to give the function $T_{\mathbf{r}_i} S$,

$$(T_{\mathbf{r}_i} S)(\mathbf{x}) = \begin{cases} \frac{2\pi}{(\pi^2 - 4)u^2} \cos^2 \left(\frac{\pi}{2u} |\mathbf{x} - \mathbf{r}_i| \right), & |\mathbf{x} - \mathbf{r}_i| \leq u, \\ 0, & \text{otherwise.} \end{cases} \quad (5.30)$$

where u denotes the radius of the source. The σ_{ji} are constants describing the production rate of the source i , with respect to field j . This is reflected in the fact that $\int_{\Omega} \sigma_{ji} (T_{\mathbf{r}_i} S)(\mathbf{x}) d\mathbf{x} = \sigma_{ji}$.

Example 1: Axon guidance in a simple concentration field We will now consider an example simulation, with a single concentration field and a single growth cone. For the domain Ω we take the unit circle and put a source at $(0.5, 0)$. This source produces at a production rate $\sigma_{11} = 1.0 \cdot 10^{-4}$ a field ρ_1 with diffusion coefficient $d_1 = 1.0 \cdot 10^{-4}$ and absorption parameter $k_1 = 1.0 \cdot 10^{-4}$. For the width of the source we take $w = 0.02$.

The growth cone is modelled by using system (5.29) with the functions λ_1 set to $\lambda_1 \equiv 1$, which means that $\phi_g = \arg \nabla \rho_1$. Further we use the parameter values $v = 1.0 \cdot 10^{-5}$ and $\ell = 0.02$. Thus the total system we have to solve becomes:

$$\begin{aligned} 0 &= d_1 \Delta \rho_1(\mathbf{x}) - k_1 \rho_1(\mathbf{x}) + \sigma_{11} T_{\mathbf{r}_1} S(\mathbf{x}), \quad \forall \mathbf{x} \in \Omega, \\ 0 &= \mathbf{n}(\mathbf{x}) \cdot \nabla \rho_1(\mathbf{x}), \quad \forall \mathbf{x} \in \partial \Omega, \end{aligned}$$

$$\mathbf{r}_1 = (0.5, 0) \tag{5.31}$$

$$\frac{d}{dt} \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{s}_2 \end{pmatrix} = \begin{pmatrix} v \cos(\mathbf{s}_2) \\ v \sin(\mathbf{s}_2) \\ v/\ell \sin(\arg(\nabla \rho_1(\mathbf{r}_2)) - \mathbf{s}_2) \end{pmatrix}, \quad \begin{pmatrix} \mathbf{r}_2(0) \\ \mathbf{s}_2(0) \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ \phi_0 \end{pmatrix}$$

In the simulation we solved the diffusion profile using 1514 nodes with six attracting rings and 2 non-attracting rings around the source location. This gives a refinement such that the node density inside the source support is about 100 times higher than far away from the source. Using the field solution we solved the paths of 50 growth cones, where we chose the start positions of the cones (x_0, y_0) randomly inside an initial area. For this we took a circle with radius 0.1 centered at $(-0.5, 0)$. The initial growth directions ϕ_0 were chosen randomly from $[0, 2\pi)$. With the integration done from $t = 0$ to $t = 1.0 \cdot 10^5$, we obtained the set of axon paths shown in Figure 5.4.



Figure 5.4: Axon paths growing toward target

If we compare this result with pictures of similar experiments with real axon

growth (Dodd and Jessell, [9]). we see that real axons often start to grow away from the initial area before they seem to react to the attracting field. This could mean that real axons have a higher stiffness than the stiffness we used in Figure 5.4. We therefore increased the stiffness by setting $\ell = 0.1$. This results in the paths shown in the left panel of Figure 5.5. While this gives a somewhat better result, it seems not realistic to increase the stiffness this far, because one would expect growing axons to make quicker turns.

Another option would be to assume that the neurons in the initial area excrete a repellent. To implement this we define a new field ρ_2 , with a source located at the location of the initial area $\mathbf{r}_b = (-0.5, 0)$. The definition of ϕ_g has to be extended with an extra repellent term; we choose $\phi_g = \arg(\nabla\rho_1 - \nabla\rho_2)$. The resulting system now is

$$\begin{aligned} 0 &= d_1 \Delta \rho_1(\mathbf{x}) - k_1 \rho_1(\mathbf{x}) + \sigma_{11} T_{\mathbf{r}_1} S(\mathbf{x}), & \forall \mathbf{x} \in \Omega, \\ 0 &= d_2 \Delta \rho_2(\mathbf{x}) - k_2 \rho_2(\mathbf{x}) + \sigma_{23} T_{\mathbf{r}_3} S(\mathbf{x}), & \forall \mathbf{x} \in \Omega, \\ 0 &= \mathbf{n}(\mathbf{x}) \cdot \nabla \rho_1(\mathbf{x}) = \mathbf{n}(\mathbf{x}) \cdot \nabla \rho_2(\mathbf{x}), & \forall \mathbf{x} \in \partial\Omega, \end{aligned}$$

$$\mathbf{r}_1 = (0.5, 0) \tag{5.32}$$

$$\frac{d}{dt} \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{s}_2 \end{pmatrix} = \begin{pmatrix} v \cos(\mathbf{s}_2) \\ v \sin(\mathbf{s}_2) \\ v/\ell \sin(\arg(\nabla\rho_1(\mathbf{r}_2) - \nabla\rho_2(\mathbf{r}_2)) - \mathbf{s}_2) \end{pmatrix}, \quad \begin{pmatrix} \mathbf{r}_2(0) \\ \mathbf{s}_2(0) \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ \phi_0 \end{pmatrix}$$

$$\mathbf{r}_3 = (-0.5, 0)$$

with $v = 1.0 \cdot 10^{-5}$ and $\ell = 0.02$. The paths of the growth cones are shown in the right panel of Figure 5.5. This gives paths more similar to the ones observed in the experiments.

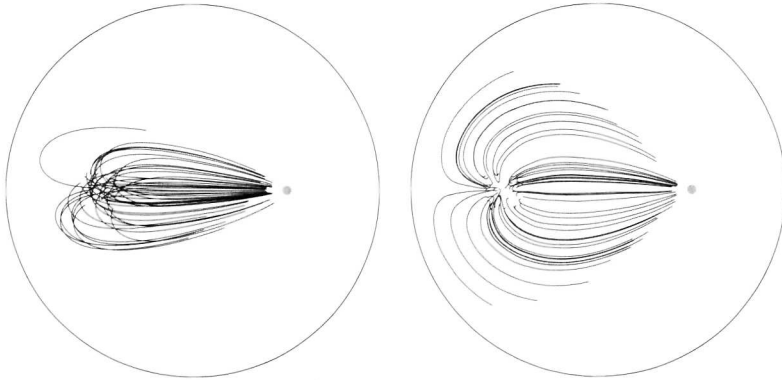


Figure 5.5: Compared with Figure 5.4, axons have a higher stiffness (left) or sense a repellent field secreted in the initial area (right).

Example 2: Axon guidance in a complex concentration field We will now consider a variation of the previous example where the domain has been changed from a simple circular domain to a more complex domain with four holes in it. These holes might represent blood vessels or cells where the axons have to grow around and that are also impenetrable to the diffusive guidance molecules.

In this simulation we again use system (5.31) to model 50 growth cones with randomly chosen initial state vectors $(x_0, y_0, \phi_0) \in [-0.4, -0.2] \times [-0.5, 0.5] \times [0, 2\pi]$. For the field, 2502 nodes were used with refinement around the outer as well as the inner boundaries and around the source location. The results of the simulation are shown in Figure 5.6.

Although in this case the axons grow nicely around the holes, there is actually no mechanical force in the model that prevents the growth cones from entering the holes. Here the growth cone dynamics alone was sufficient to keep the growth cones outside the holes. However, if ℓ is bigger, the growth cones will need more space to turn, and might enter the holes if not stopped by a hard boundary.

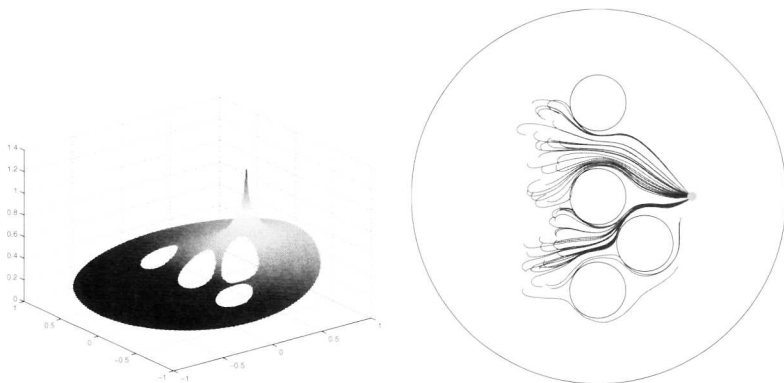


Figure 5.6: (Left) Field on a domain with holes. (Right) Axon paths produced by system (5.31).

Example 3: Axon guidance with internal growth cone dynamics In this example we will extend our cone dynamics by adding another variable. In the previous examples the ideal direction, based on the sensed gradients, is directly translated in a change of direction. In real growth cones, however, signaling pathways inside the growth cone are responsible for this translation. We now incorporate such signaling pathways and represent it by a single variable $\alpha \in [-1, 1]$, where $\alpha < 0$ means steering to the left and $\alpha > 0$ steering to the right. The growth cone translates the ideal direction into the signaling pathway dynamics in a way that is similar to the way that the ideal direction is translated into the direction dynamics in the previous examples.

The state consists now of $(x, y, \phi, \alpha) \in \mathbb{R}^2$ and its dynamics are given by

$$\frac{d}{dt} \begin{pmatrix} x \\ y \\ \phi \\ \alpha \end{pmatrix} = \begin{pmatrix} v \cos(\phi) \\ v \sin(\phi) \\ -v\alpha/\ell \\ c(\sin(\phi_g - \phi) - \alpha) \end{pmatrix}, \quad (5.33)$$

with ϕ_g again defined as in (5.28). Here, parameters are as in the previous example and c is a parameter that determines how fast the steering dynamics is. If the dynamics is fast, i.e., c is big, we have $\alpha \approx \sin(\phi_g - \phi)$, resulting in the previous model. But if c is small, a kind of zig-zag behavior emerges (Figure 5.7, $c = 0.1$) that is also observed in some experiments (Ming et al. [35]). In Ming et al. [35], this behavior was thought to occur as a result of alternating phases of receptor sensitization and desensitization. Our simulation, without such receptor adaptation, shows that oscillatory growth cones paths can already arise as a result of an inertia of the steering dynamics.

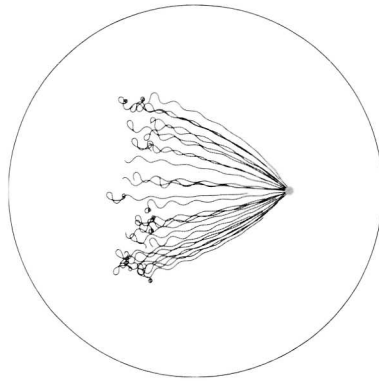


Figure 5.7: Wiggly axon paths produced by system (5.34).

For completeness, the total system in this case is

$$\begin{aligned} 0 &= d_1 \Delta \rho_1(\mathbf{x}) - k_1 \rho_1(\mathbf{x}) + \sigma_{11} T_{\mathbf{r}_1} S(\mathbf{x}), \quad \forall \mathbf{x} \in \Omega, \\ 0 &= \mathbf{n}(\mathbf{x}) \cdot \nabla \rho_1(\mathbf{x}), \quad \forall \mathbf{x} \in \partial\Omega, \end{aligned}$$

$$\mathbf{r}_1 = (0.5, 0) \quad (5.34)$$

$$\frac{d}{dt} \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{s}_2 \end{pmatrix} = \begin{pmatrix} v \cos((\mathbf{s}_2)_1) \\ v \sin((\mathbf{s}_2)_1) \\ -v(\mathbf{s}_2)_2/\ell \\ c(\sin(\arg(\nabla \rho_1(\mathbf{r}_2)) - (\mathbf{s}_2)_1) - (\mathbf{s}_2)_2) \end{pmatrix}, \quad \begin{pmatrix} \mathbf{r}_2(0) \\ \mathbf{s}_2(0) \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ \phi_0 \\ \alpha_0 \end{pmatrix}$$

with $v = 1.0 \cdot 10^{-5}$ and $\ell = 0.02$.

Example 4: Axon guidance with membrane-bound guidance molecules in topographic map formation In our last example we consider a more complicated model of a phenomenon that is called topographic mapping [49]. Many neuronal connections are made so as to form a topographic map of one structure onto another. In other words, neighboring cells in one structure make connections to neighboring cells in the other structure. An example of a topographic map is the direct projection of the retina onto the optic tectum in the brain of non-mammalian vertebrates [13]. One explanation for the formation of topographic maps that has received strong experimental support is that it is based on the matching of gradients of receptors and their ligands [37, 52]. For the retinotectal projection, there is a gradient across the retina in the number of Eph receptors on the growth cones of the retinal neurons. A similar but opposite gradient is found across the tectum in the number of membrane-bound ephrin molecules (the ligands for Eph receptors) on the tectal neurons. Axons grow out so that growth cones with a low number of receptors come to connect to tectal cells with a high number of ligand molecules, and vice-versa.

A simple model for this phenomenon is the following (see also [23]). We use essentially model (5.29), but we extend it with two extra variables, β_x and β_y that represent the levels of two kinds of receptors. These β_x and β_y remain constant during growth and vary with respect to the initial location $\mathbf{r}_0 = (x_0, y_0)$. We take for these

$$\beta_x = \exp(1.39x_0 + 1.18) \quad \text{and} \quad \beta_y = \exp(1.39y_0 + 0.35). \quad (5.35)$$

We will assume that there are five fields of which three are diffusive fields and two are the fields of membrane bound ligands. Fields ρ_1 , ρ_2 and ρ_3 are produced by guidance cells located at $\mathbf{r}_1 = (-0.1, 0)$, $\mathbf{r}_2 = (0.85, 0)$ and $\mathbf{r}_3 = (0.3, 0.85)$, respectively. We use the same diffusion rate $d = 1.0 \cdot 10^{-4}$ and the absorption rate $k = 1.0 \cdot 10^{-4}$ as in the previous examples. The two fields of membrane bound ligands ρ_4 and ρ_5 are described by explicit functions that are given by

$$\rho_4(x, y) = \exp(-1.39x + 0.21) \quad \text{and} \quad \rho_5(x, y) = \exp(-1.39y + 0.14). \quad (5.36)$$

We will assume that the dynamics of the growth cones occurs in two phases. In the first phase the growth cones are attracted by field ρ_1 and they grow toward the guidance cell located at \mathbf{r}_0 . Once they have reached the guidance cell, which we will formalize by $(T_{\mathbf{r}_1}, S)(\mathbf{r}) > 0$, they switch their behavior and phase two will start. The dynamics of the growth cones during the first phase are given by

$$\frac{d}{dt} \begin{pmatrix} x \\ y \\ \phi \\ \beta_x \\ \beta_y \end{pmatrix} = \begin{pmatrix} v \cos(\phi) \\ v \sin(\phi) \\ v/\ell \sin(\phi_g - \phi) \\ 0 \\ 0 \end{pmatrix}, \quad \text{with} \quad \phi_g = \arg(\nabla \rho_1(\mathbf{r})). \quad (5.37)$$

For the dynamics of the growth cones in phase two we need assumptions on the influence of the receptors and ligands on the growth. The basic assumption is the following. For each direction, i.e., x - or y -direction, we have a couple of receptor and

ligand pairs. For growth in either of these directions it is needed that the concentration of the ligand in the neighborhood of the cone is above a certain level, which is determined by the receptor density on the growth cone. We will assume that growth in x -direction is determined by the product $c_x = \beta_x \rho_4(\mathbf{r})$ (and similar $c_y = \beta_y \rho_5(\mathbf{r})$ for the y -direction.) A $c_x \ll 1$ means strong inhibition of growth in x -direction and $c_x \gg 1$ means no inhibition. The dynamics is the same as in the first phase, but now with

$$\phi_g = \arg \left(\text{Sgm}_{20}(c_x) \nabla \rho_2(\mathbf{r}) + \text{Sgm}_{20}(c_y) \nabla \rho_3(\mathbf{r}) \right). \quad (5.38)$$

Here the function Sgm_n is defined by $\text{Sgm}_n(x) = x^n / (1 + x^n)$. Finally we will assume that the growth is completely inhibited if both $c_x < 0.8$ and $c_y < 0.8$.

To summarize, the total system is given by

$$\begin{aligned} 0 &= d_j \Delta \rho_j(\mathbf{x}) - k_j \rho_j(\mathbf{x}) + \sigma_{jj} T_{r_j} S(\mathbf{x}), \quad \forall \mathbf{x} \in \Omega, \quad j = 1, \dots, 3 \\ 0 &= \mathbf{n}(\mathbf{x}) \cdot \nabla \rho_j(\mathbf{x}), \quad \forall \mathbf{x} \in \partial\Omega, \quad j = 1, \dots, 3 \\ \rho_4(\mathbf{x}) &= \exp(-1.39x + 0.21), \quad \forall \mathbf{x} \in \Omega, \\ \rho_5(\mathbf{x}) &= \exp(-1.39y + 0.14), \quad \forall \mathbf{x} \in \Omega, \\ \mathbf{r}_1 &= (-0.1, 0) \\ \mathbf{r}_2 &= (0.85, 0) \\ \mathbf{r}_3 &= (0.3, 0.85) \end{aligned} \quad (5.39)$$

$$\frac{d}{dt} \begin{pmatrix} \mathbf{r}_4 \\ \mathbf{s}_4 \end{pmatrix} = \begin{pmatrix} v \cos((\mathbf{s}_4)_1) \\ v \sin((\mathbf{s}_4)_1) \\ v/\ell \sin(\arg(\phi_g) - (\mathbf{s}_4)_1) \\ 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} \mathbf{r}_4(0) \\ \mathbf{s}_4(0) \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ \phi_0 \\ \exp(1.39x_0 + 1.18) \\ \exp(1.39y_0 + 0.35) \end{pmatrix}$$

with $v = 1.0 \cdot 10^{-5}$ and $\ell = 0.02$.

phase 1: $\phi_g = \arg(\nabla \rho_1(\mathbf{r}_4))$. **if** $(T_{r_1} S(\mathbf{r})) > 0$ **goto phase 2.**

phase 2: $\phi_g = \arg(\text{Sgm}_{20}(\beta_x \rho_4(\mathbf{r}_4)) \nabla \rho_2(\mathbf{r}_4) + \text{Sgm}_{20}(\beta_y \rho_5(\mathbf{r}_4)) \nabla \rho_3(\mathbf{r}_4))$.
if $(\beta_x \rho_4(\mathbf{r}_4) < 0.8$ **or** $\beta_y \rho_5(\mathbf{r}_4) < 0.8)$ **ready.**

In Figure 5.8 we see the fields in a simulation of the topographic mapping model. The three upper panels show the three diffusive fields ρ_1 , ρ_2 and ρ_3 . In Figure 5.9 the axons paths are shown. The left panel shows the paths of 200 growth cones that started at the left with randomly chosen initial positions (x_0, y_0) and orientations ϕ_0 . Clearly all growth cones are attracted by the guidance cell in the middle. Having reached this cell they change their behavior and gain attractivity to the fields ρ_2 and ρ_3 . This attractivity is steered by the fields ρ_4 and ρ_5 , which also determine when growth is completely inhibited.

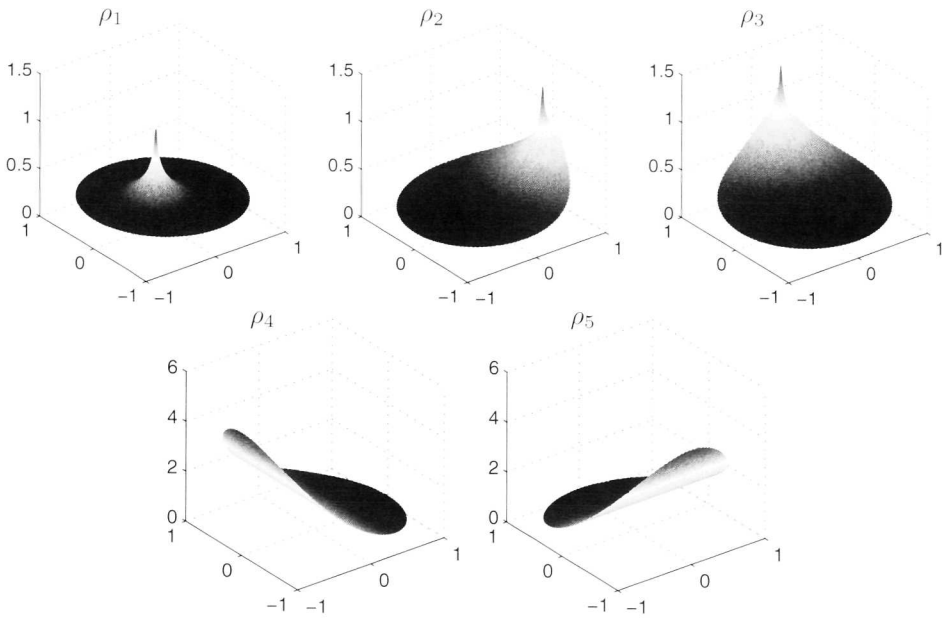


Figure 5.8: Fields in the example of topographic mapping. The three fields in the top row are diffusive fields, and the ones in the bottom row are fields of membrane bound ligands.

To visualize the conservation of spatial order we have color coded the initial locations and end locations of the paths, i.e., begin and end points of a path have the same color. The result of this is displayed in the left panel and it clearly shows that the ‘A’ is transferred from the initial area (at the left) to the final area (at the right).

The combination of the membrane bound ligand fields ρ_4 and ρ_5 with the receptor densities β_x and β_y determines what the topographic mapping will look like. Using a model like this for exploring different possibilities for the concentration fields can give us more insight into the forms of the fields and mechanisms involved in topographic map formation.

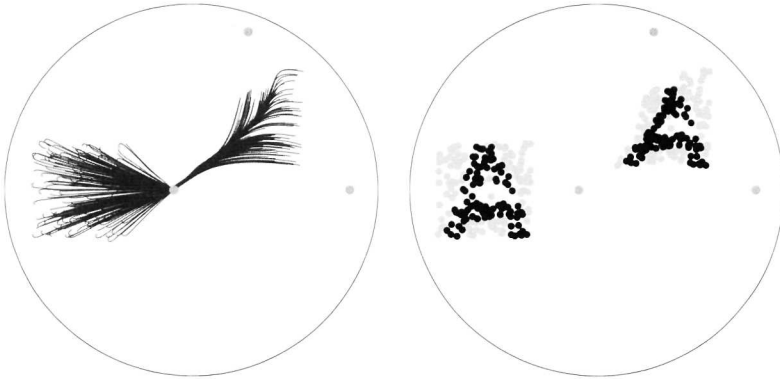


Figure 5.9: Axon paths in the example of topographic mapping. (Left) Resulting axon paths, starting at the left and ending at the right. (Right) Visualization of the conservation of spatial order between the final axon targets and the initial neuron locations.

5.6 Discussion

In this chapter we have presented a framework for the modelling of axon guidance. In contrast to the modelling of electrical activity in neurons and neuronal networks, such a general framework did not exist. Our framework allows for the relatively straightforward and fast modelling and simulation of axon guidance and its underlying mechanisms. For example, mechanisms that ‘translate’ concentration levels of guidance molecules (or gradients thereof) measured at the growth cone’s location into growth speed, sensitivity for certain fields, and growth direction, can easily be incorporated. A major challenge in the study of axon guidance is to understand quantitatively how the many molecules and mechanisms involved in axon guidance act in concert to generate complex patterns of neuronal connections. The framework we developed contributes to this challenge by providing a general simulation tool in which a wide range of models can be implemented and explored.

Our framework has three basic ingredients, which are the domain, the concentration fields and the states. The domain models the physical environment where the neurons, axons, and fields live in; the domain can have a complicated geometry with piecewise smooth boundaries and holes. The fields are defined on the domain and represent the time varying concentration fields of guidance molecules that are subject to diffusion and absorption. The states model the growth cones and targets cells and consist of finite dimensional vectors for which the dynamics are given in the form of ODEs that model the mechanisms involved in axon guidance.

Specific numerical methods have been developed that are suitable for solving the systems of equations that typically arise in models of axon guidance. With respect to time integration for the full system a method is needed that can handle the combination of stiff diffusion equations (describing the concentration fields) and non-stiff, nonlinear differential equations (describing the states). For this a 2nd-order Runge-Kutta IMEX scheme is used. In case of static fields or a quasi-steady-state approximation an explicit time integrator will suffice, for which we use the classical 4th-order Runge-Kutta method.

The spatial discretizations needed for solving the elliptic field equations that arise after discretization in time, are based on arbitrary node sets. Voronoi diagrams are used for the selection of suitable node sets as well as for the discretization of the equations. Refinement and adaptivity of the discretization are based upon the location of the highly localized sources only, to speed up the node selection process.

We have implemented the framework and the numerical algorithms in a set of Matlab programs. In these programs one can simulate a wide range of models by defining appropriate Matlab data-structures and solve them by applying the spatial and temporal numerical solvers. At the moment, the code is typical research code without extensive documentation, but we are working on a more user-friendly version.

Possible extensions of our framework include the incorporation of randomness in the guidance of the axons and the possibility that boundaries (of impenetrable holes, for example) can produce guidance molecules. The latter extension would make it possible to model also tissues, rather than individual cells, that attract or repel axons.

Appendix

Field set-up time To examine how the time for setting up the field depends on d and k , we consider the solution of (5.3) with a point source at the origin, $S(\mathbf{x}, t) = \delta(\mathbf{x})$, and an initial field $\rho = 0$ at time $t = 0$. The field will be radially symmetric, and the concentration, which depends only on the radius r and the time t , is

$$\rho(r, t) = \int_0^t \frac{e^{-\left(ks + \frac{r^2}{4ds}\right)}}{4\pi ds} ds \quad \xrightarrow{t \rightarrow \infty} \quad \frac{1}{2\pi d} K_0 \left(r \sqrt{\frac{k}{d}} \right). \quad (5.40)$$

where the limit of the solution is the steady state solution, which satisfies (5.7), and K_0 is a modified Bessel function of the Second Kind [1]. To see how fast the field

approaches the steady state field, we will investigate

$$c(r, t) = \frac{\rho_\infty(r) - \rho(r, t)}{\rho_\infty(r)} = \frac{1}{2K_0 \left(r\sqrt{\frac{k}{d}} \right)} \int_{\frac{2}{r}\sqrt{dk}t}^{\infty} \frac{e^{-\frac{r}{2}\sqrt{\frac{k}{d}}(s+\frac{1}{s})}}{s} ds.$$

which represents how close the field is to its limit value. For example, a value $c(r, t) = 0.01$, means that at time t the field is for 99% set up, at location r . Using an asymptotic expansion for large t for the integral, we find that

$$c(r, t) \sim \frac{1}{2K_0 \left(r\sqrt{\frac{k}{d}} \right)} \frac{e^{-kt}}{kt}. \quad (5.41)$$

This can be used to get an indication of the time scale of the field dynamics. Such an indicator is important if we want to work with fields of which the sources do not move through the domain. In case of moving sources, one might wonder how the speed of a source influences the produced field. To this end we examine the solution of (5.3) with a point source that moves with constant speed.

Field produced by moving source Consider equation (5.3) with a point source that moves with constant speed v along the x -axis in positive direction, i.e., $S(\mathbf{x}, t) = \delta(\mathbf{x} - \mathbf{v}t)$, with $\mathbf{v} = (v, 0)^T$. If we make the 'ansatz' that the solution $\rho(\mathbf{x}, t)$ is the sum of a solution profile $\hat{\rho}$ that moves with constant speed with the source and a 'residual' solution η ,

$$\rho(\mathbf{x}, t) = \hat{\rho}(\mathbf{x} - \mathbf{v}t) + \eta(\mathbf{x}, t),$$

we can rewrite (5.12) to

$$\frac{\partial}{\partial t} \eta(\mathbf{x}, t) = d\Delta\hat{\rho}(\mathbf{x} - \mathbf{v}t) + \mathbf{v} \cdot \nabla\hat{\rho}(\mathbf{x} - \mathbf{v}t) - k\hat{\rho}(\mathbf{x} - \mathbf{v}t) + \delta(\mathbf{x} - \mathbf{v}t) + d\Delta\eta(\mathbf{x}, t) - k\eta(\mathbf{x}, t). \quad (5.42)$$

If $\hat{\rho}$ satisfies the equation

$$d\Delta\hat{\rho}(\mathbf{x}) + \mathbf{v} \cdot \nabla\hat{\rho}(\mathbf{x}) - k\hat{\rho}(\mathbf{x}) + \delta(\mathbf{x}) = 0. \quad (5.43)$$

we see that equation (5.12) will turn into a equation for η with only diffusion and absorption. Therefore, η will damp out for long times, resulting in $\rho(\mathbf{x}, t) \approx \hat{\rho}(\mathbf{x} - \mathbf{v}t)$. The solution of (5.43) in polar coordinates (r, ϕ) : $x = r \cos(\phi)$, $y = r \sin(\phi)$, is given by

$$\hat{\rho}(r, \phi) = \frac{1}{2\pi d} \exp\left(-\sqrt{\frac{k}{d}} \left(\frac{v}{2\sqrt{kd}}\right) r \cos(\phi)\right) K_0\left(\sqrt{\frac{k}{d}} \left(\sqrt{\left(\frac{v}{2\sqrt{kd}}\right)^2 + 1}\right) r\right). \quad (5.44)$$

This solution we will compare to the steady state solution of (5.13)

$$\rho_s(r) = \frac{1}{2\pi d} K_0 \left(r \sqrt{\frac{k}{d}} \right). \quad (5.45)$$

where the subscript s refers to the steady state. So we will consider the quotient function $q(r, \phi) = \hat{\rho}(r, \phi) / \rho_s(r)$ and we want to investigate the geometry of the region where this quotient is close to 1. For example, given a value $\gamma > 1$, and slightly bigger than one, we could consider the region $\{(r, \phi) \mid \gamma^{-1} \leq \hat{\rho} / \rho_s \leq \gamma\}$. Using a rescaling of $s = r\sqrt{k/d}$ and $\alpha = v/(2\sqrt{dk})$, we get

$$q = \frac{\hat{\rho}}{\rho_s} = e^{-\alpha s \cos(\phi)} \frac{K_0((\sqrt{1+\alpha^2})s)}{K_0(s)}.$$

To analyze q we use the asymptotic expansions of K_0 and K_1 , both modified Bessel functions of the Second kind,

$$K_0(x) = \ln(2) - \ln(x) - \gamma_E + \mathcal{O}(x^2), \quad K_1(x) = \frac{1}{x} + \mathcal{O}(x) \quad (x \downarrow 0), \quad (5.46)$$

$$K_0(x) \sim \sqrt{\frac{\pi}{2x}} e^{-x}, \quad K_1(x) \sim \sqrt{\frac{\pi}{2x}} e^{-x} \quad (x \rightarrow \infty), \quad (5.47)$$

where γ_E is Euler's constant [1].

Close to the source, q is close to 1 as follows from $\lim_{s \downarrow 0} q(s, \phi) = 1$, which can be seen by using the expansion K_0 around 0. To find the behavior around 0, we will examine the derivative of q with respect to s ,

$$\partial_s q = q \left\{ \frac{K_1(s)}{K_0(s)} - \sqrt{1+\alpha^2} \frac{K_1(\sqrt{1+\alpha^2}s)}{K_0(s)} - \alpha \cos(\phi) \right\}.$$

This is equal to q times some factor that is increasing with s and has limit values $-\infty$ at $s = 0$ and $1 - \sqrt{1+\alpha^2} - \alpha \cos(\phi)$ at $s = \infty$. For $\phi = 0$ this limit is negative, while for $\phi = \pi$ this limit is positive. Therefore, there is an interval $[-\phi_t, \phi_t]$ with $\phi_t \in [0, \pi]$ of possible choices of ϕ for which q decreases with s while keeping ϕ constant.

For ϕ outside this interval, i.e., $\phi \in (-\pi, -\phi_t) \cup (\phi_t, \pi]$, there is an $s_\phi > 0$, with $\partial_s q(s_\phi, \phi) = 0$, such that q as a function of s decreases for $s \in (0, s_\phi)$ and increases for $s \in (s_\phi, \infty)$. The function $\phi \rightarrow s_\phi$ itself is decreasing on $(\phi_t, \pi]$ with $\lim_{\phi \downarrow \phi_t} s_\phi = \infty$. To find $\phi_t \in [0, \pi]$, we solve

$$1 - \sqrt{1+\alpha^2} - \alpha \cos(\phi_t) = 0, \quad \implies \quad \cos(\phi_t) = \frac{1 - \sqrt{1+\alpha^2}}{\alpha} < 0,$$

where the last inequality follows from the fact that $\alpha > 0$. Therefore, $\phi_t \in (\frac{1}{2}\pi, \pi)$, which is increasing with α and has limits $\phi_t = \frac{1}{2}\pi$ with $\alpha \downarrow 0$ and $\phi_t = \pi$ for $\alpha \rightarrow \infty$.

We can now conclude that close to the origin there is some region where we have $q \leq 1$. To find an estimate of the size of this region we will use the asymptotic expansion of q for small s ,

$$q = 1 + \frac{\frac{1}{2} \ln(1 + \alpha^2)}{\ln(s) - \ln(2) + \gamma_E} + \mathcal{O}(s).$$

Neglecting the higher-order terms and setting this equal to γ gives

$$s = 2e^{-\gamma E} (1 + \alpha^2)^{\frac{1}{2(\gamma-1)}} \implies r = 2e^{-\gamma E} \sqrt{\frac{d}{k}} \left(1 + \frac{v^2}{4dk}\right)^{\frac{1}{2(\gamma-1)}}. \quad (5.48)$$

If we choose $\gamma = 0.99$, we get an indication for the region around the source, where the difference between the moving profile and the quasi-steady-state solution is smaller than 1%, given the values of the diffusion rate d , absorption rate k and moving speed v .

Chapter 6

Numerical solution of the framework's equation systems

6.1 Introduction

The current chapter considers different methods for numerically solving special systems of coupled PDEs and ODEs/DAEs. The PDEs describe diffusion processes of concentration fields and are nonlinearly coupled to the ODEs/DAEs, that describe the motion of particle-like objects that interact with these fields.

The starting point for our research on such systems is an article of Hentschel and Van Ooyen [21], where they model the outgrowth of axons out of neurons. The growing axons react to different concentration fields of so-called 'attractants' and 'repellents' that are subject to diffusion and absorption processes. The movement of the axon heads, i.e., when growth occurs, is determined by local values and gradients of the fields. At the same time these so called growth cones act as sources for the fields as do the target neurons and messenger cells. In [21] growth cones were modelled by their location, for which ODEs were proposed and point sources were used in the diffusion equations for the fields. The equations were solved by using quasi-steady-state approximation for the fields on an infinite 2-dimensional domain, effectively reducing the system to a finite dimensional system of ODEs, that was solved using standard explicit RK-methods.

To facilitate the research on this so called axon guidance Krottje and Van Ooyen ([30], Chapter 5) developed a simulation framework for a certain class of such systems. The more general approach of this framework allows for the definition of a number of fields and states that are linked to each other. The fields can be defined on domains with piecewise smooth boundaries on which no in- and outflow is assumed. Sources are described by continuous bell-shaped functions with local support instead of point sources which may result in ill-defined systems ([29], Chapter 4). States are defined as objects that interact with the fields having a certain position and are modelled in

a finite-dimensional way, resulting in ODEs/DAEs for their dynamics. They can act as sources for the fields and their movement is determined by local field values and gradients. From a modelling perspective they can be growth cones, target neurons or artificial sources in an experimental setting.

It has turned out that the developed framework has some features which make it numerically challenging. First, there are the small moving sources for the diffusion equations. Efficiently finding a solution approximation of the field equations may need some kind of refinement and adaptivity in the setting of geometrically complex domains with possibly a number of holes. Second, the system consists of diffusion equations giving rise to stiffness and ODEs/DAEs that are non-stiff and nonlinear. This makes that choosing a suitable time integration method is not a trivial task.

We wrote a set of Matlab functions for carrying out simulations of models defined in the framework. To address the first challenge we used a spatial discretization that can handle complex domains as well as refinement and works with a set of independent nodes instead of a grid ([29], Chapter 4). It uses a Voronoi diagram, both for building local approximations and proper placement of nodes. For the time integration we used a second order Runge-Kutta IMEX method based on a combination of the implicit and explicit midpoint rule.

However, the question arises whether instead we could use a standard FE package for solving such systems and, if possible, how it would compare to using our set of Matlab functions if one considers efficiency. To get some insight into these issues, we will therefore examine in this chapter simulation of models in the framework using both our set of Matlab functions as well as a typical FE solver developed for parabolic PDEs. We will pick as a representative solver the program Kardos [31]. Kardos includes an adaptive multilevel finite element package and uses Rosenbrock methods for time integration.

The organization of the chapter is as follows. We start with a description of our simulation framework in Section 2. We will give a short introduction to our own developed Matlab package, AGTools (Axon Guidance Tools) in Section 3 and to Kardos in Section 4. Section 5 is devoted to the application of Rosenbrock time-integration methods within our own framework. In Section 6 we will compare the efficiency of both implementations and we finish with a conclusion in Section 7.

6.2 Simulation framework

From the mathematical point of view the framework consists of a number of diffusion equations, the PDEs, which contain besides the diffusion terms, absorption and source terms. These equations are strongly coupled to nonlinear ODEs/DAEs, of which the righthandsides contain field values and gradients of the fields evaluated at certain locations. We will not go into the biological interpretation here, which can be found in ([30], Chapter 5), but we want to stress that the ODEs/DAEs describe particle-like moving objects that interact with the fields. This interaction occurs by means of field sources associated to the objects as well as by movement of the objects guided by

locally measured field values and gradients.

Model state The models in the framework are from the mathematical perspective infinite-dimensional dynamical systems of which the model state is a combination of fields and finite-dimensional particle states.

The fields are defined on a 2-dimensional domain Ω with a piecewise smooth boundary and are denoted by $\rho_j: \Omega \rightarrow \mathbb{R}$, $j = 1, \dots, M$, where M denotes the number of fields. They will be assumed to be elements of $H^1(\Omega)$, the space of square integrable functions defined on Ω of which also the first-order derivatives are square integrable.

The particle states are finite-dimensional vectors $\mathbf{u}_i \in \mathbb{R}^{n_i}$ of which the first two components denote a location $\mathbf{r}_i \in \Omega$ and the rest of the variables is gathered in a vector $\mathbf{s}_i \in \mathbb{R}^{n_i-2}$ and in the following referred to as the \mathbf{s} -part of the state. This part may be obsolete and therefore the particle states are at least 2-dimensional. We will denote the number of particle states by N .

Concluding, the model state, which we denote by \mathbf{x} , is of the form

$$\mathbf{x} = (\rho_1, \dots, \rho_M, \mathbf{u}_1, \dots, \mathbf{u}_N) \in (H^1(\Omega))^M \times \mathbb{R}^{n_1 + \dots + n_N}. \quad (6.1)$$

Model dynamics To complete the definition of a dynamical system we will add to the model state \mathbf{x} the dynamics in the form of the PDEs, ODEs and algebraic equations. We start with the dynamics of the M fields. For all fields we assume that there is no inflow or outflow across the boundary $\partial\Omega$ of Ω , giving the boundary condition

$$\mathbf{n} \cdot \nabla \rho_j = 0, \quad \text{on } \partial\Omega, \quad j = 1, \dots, M, \quad (6.2)$$

where \mathbf{n} is the outward normal vector. We assume that for the first M_d fields the dynamics is given by full diffusion equations,

$$\partial_t \rho_j = L_j \rho_j + \sum_{i=1}^N \sigma_{ji}(\mathbf{s}_i) T_{\mathbf{r}_i} S, \quad \text{on } \Omega, \quad j = 1, \dots, M_d, \quad (6.3)$$

where $L_j = d_j \Delta - k_j$ for all $j = 1, \dots, M_d$. In each diffusion equation there is a source term $\sigma_{ji}(\mathbf{s}_i) T_{\mathbf{r}_i} S$ associated with every particle state $\mathbf{u}_i = (\mathbf{r}_i, \mathbf{s}_i)$. Here the function $\sigma_{ji}: \mathbb{R}^{n_i} \rightarrow \mathbb{R}$ denotes an excretion rate of the source term and the function $T_{\mathbf{r}_i} S: \Omega \rightarrow \mathbb{R}$ denotes a continuous source profile. The latter is defined by applying a translation operator $T_{\mathbf{r}_i}$ to a general source profile $S: \Omega \rightarrow \mathbb{R}$, where the operator is defined by $(T_{\mathbf{r}_i} S)(\mathbf{x}) = S(\mathbf{x} - \mathbf{r}_i)$ for all \mathbf{x} , $\mathbf{x} - \mathbf{r}_i \in \Omega$. When the \mathbf{s} -part of the particle-state is absent we will assume the σ_{ji} to be constants.

For the rest of the fields we will assume that they are in quasi-steady-state and use for their dynamics the equations

$$0 = L_j \rho_j + \sum_{i=1}^N \sigma_{ji}(\mathbf{s}_i) T_{\mathbf{r}_i} S, \quad \text{on } \Omega, \quad j = M_d + 1, \dots, M. \quad (6.4)$$

being defined similarly as in (6.3). Note that we can still talk about their ‘dynamics’ because the source terms contain the time-dependent particle states.

For the particle states we assume in the same way a division in states governed by real dynamics in the form of ODEs and states determined by quasi-static laws in the form of algebraic equations. For the first N_d particle states we assume that there are functions G_i such that their dynamics are given by

$$\partial_t \mathbf{u}_i = G_i(t, \mathbf{u}_i, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i)), \quad i = 1, \dots, N_d. \quad (6.5)$$

Here we use the vector notation $\boldsymbol{\rho}(\mathbf{r}_i)$, meaning $\boldsymbol{\rho}(\mathbf{r}_i)_j = \rho_j(\mathbf{r}_i)$ for all j , and $\partial_x \boldsymbol{\rho}(\mathbf{r}_i)$, meaning $\partial_x \boldsymbol{\rho}(\mathbf{r}_i)_j = \partial_x \rho_j(\mathbf{r}_i)$ for all j . For the remaining particle states we assume the dynamics to be of the form

$$0 = G_i(t, \mathbf{u}_i, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i)), \quad i = N_d + 1, \dots, N. \quad (6.6)$$

We will assume that the function G_i in equation (6.6), i.e., only for $i = N_d + 1, \dots, N$, can be decomposed in a function for the position, G_i^r , and a function for the \mathbf{s} -part, G_i^s , of the form

$$G_i(t, \mathbf{u}_i, \dots) = \left(G_i^r(t, \mathbf{r}_i), G_i^s(t, \mathbf{s}_i, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i)) \right), \quad i = N_d + 1, \dots, N. \quad (6.7)$$

in such a way that, using these, the \mathbf{r}_i and \mathbf{s}_i are uniquely solvable from equation (6.6) if the time t and fields ρ_j are given.

Abstract formulation Let us, before proceeding, for convenience first define the index sets of the dynamic fields, $J_d = \{1, \dots, M_d\}$, static (dynamic) fields, $J_s = \{M_d + 1, \dots, M\}$, dynamic states, $I_d = \{1, \dots, N_d\}$, and static (dynamic) states, $I_s = \{N_d + 1, \dots, N\}$. The equations (6.2)–(6.6) together constitute the dynamical system behind the model that can be written in the form

$$\begin{aligned} \dot{\mathbf{z}} &= f(t, \mathbf{z}, \mathbf{y}) & \text{with} & & \mathbf{z}(0) &= \mathbf{z}_0 \\ 0 &= g(t, \mathbf{z}, \mathbf{y}) & & & \mathbf{y}(0) &= \mathbf{y}_0. \end{aligned} \quad (6.8)$$

Here, the \mathbf{z} is composed of the dynamic fields and states, i.e., the fields ρ_j for $j \in J_d$ and the states \mathbf{u}_i for $i \in I_d$, and therefore consists of a selection of components of the total model state \mathbf{x} in (6.1). Likewise, the vector \mathbf{y} is composed of the static fields and states, i.e., the fields ρ_j for $j \in J_s$ and the states \mathbf{u}_i for $i \in I_s$, forming the remaining part of the model state \mathbf{x} . The initial condition consisting of the vectors \mathbf{z}_0 and \mathbf{y}_0 has to be chosen in such a way that it obeys $g(0, \mathbf{z}_0, \mathbf{y}_0) = 0$.

The dynamical system (6.8) can be turned into a ‘lower-dimensional’ system for \mathbf{z} only, if we assume that for given values of \mathbf{z} and t we can solve \mathbf{y} uniquely from $g(t, \mathbf{z}, \mathbf{y}) = 0$. This yields \mathbf{y} as a function of t and \mathbf{z} , $\mathbf{y} = h(t, \mathbf{z})$, resulting in the system

$$\dot{\mathbf{z}} = f(t, \mathbf{z}, h(t, \mathbf{z})) \quad \text{with} \quad \mathbf{z}(0) = \mathbf{z}_0. \quad (6.9)$$

We will now examine equation $g(t, \mathbf{z}, \mathbf{y}) = 0$ in more detail. Here, the vector \mathbf{z} is composed of the dynamic fields and states, i.e., ρ_j with $j \in J_d$ and \mathbf{u}_i with $i \in I_d$. Assuming that these and the time t are given, solving \mathbf{y} from $g(t, \mathbf{z}, \mathbf{y}) = 0$ comes down to solving the static fields and states from

$$\begin{aligned} 0 &= L_j \rho_j + \sum_{i \in I_d \cup I_s} \sigma_{ji}(\mathbf{s}_i) T_{\mathbf{r}_i} S, & j \in J_s, \\ 0 &= G_i \left(t, \mathbf{u}_i, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i) \right), & i \in I_s. \end{aligned} \quad (6.10)$$

In the case where we use the general functions G_i we have to solve this full system, which is nonlinear and infinite-dimensional. Using Newton iteration is a possibility, but one that requires solving elliptic equations every iteration step. An appealing alternative exists if we use the extra assumption that the functions G_i are of the special form (6.7). We can then turn the system into a finite-dimensional system for which we do not have to solve elliptic equations for every iteration step, but we have to do it only once.

To this end we first eliminate the elliptic equations from the system (6.10). Using these, the fields ρ_j can be expressed in the \mathbf{s}_i and \mathbf{r}_i by writing

$$\rho_j = - \sum_{k \in I_d \cup I_s} \sigma_{jk}(\mathbf{s}_k) L_j^{-1} T_{\mathbf{r}_k} S, \quad j \in J_s \implies \boldsymbol{\rho}(\mathbf{r}_i) = - \text{diag}([\boldsymbol{\sigma}(\bar{\mathbf{s}})][\mathbf{S}(\bar{\mathbf{r}}, \mathbf{r}_i)]). \quad (6.11)$$

Here, the operators L_j^{-1} , that commute with the scalars σ_{jk} , denote the inverse operators of L_j with respect to the boundary conditions (6.2). The $\bar{\mathbf{s}}$ and $\bar{\mathbf{r}}$ denote the vector $(\mathbf{s}_1^T, \dots, \mathbf{s}_N^T)^T$ and $(\mathbf{r}_1^T, \dots, \mathbf{r}_N^T)^T$, respectively, while the matrices $\boldsymbol{\sigma}$ and \mathbf{S} are defined by $[\boldsymbol{\sigma}(\bar{\mathbf{s}})]_{jk} = \sigma_{jk}(\mathbf{s}_k)$ and $[\mathbf{S}(\bar{\mathbf{r}}, \mathbf{r}_i)]_{kj} = (L_j^{-1} T_{\mathbf{r}_k} S)(\mathbf{r}_i)$, respectively. The function $\text{diag}(\cdot)$ is defined to return the diagonal vector of its argument. Defining the matrix \mathbf{S}_x by $[\mathbf{S}_x(\bar{\mathbf{r}}, \mathbf{r}_i)]_{kj} = \partial_x (L_j^{-1} T_{\mathbf{r}_k} S)(\mathbf{r}_i)$ yields a similar expression for $\partial_x \boldsymbol{\rho}(\mathbf{r}_i)$ with \mathbf{S} replaced with \mathbf{S}_x , while a similar definition of \mathbf{S}_y results in a similar expression for $\partial_y \boldsymbol{\rho}(\mathbf{r}_i)$.

The second equation in system (6.10) can now be written as

$$\begin{aligned} 0 &= G_i \left(t, (\mathbf{r}_i, \mathbf{s}_i), - \text{diag}([\boldsymbol{\sigma}(\bar{\mathbf{s}})][\mathbf{S}(\bar{\mathbf{r}}, \mathbf{r}_i)]), - \text{diag}([\boldsymbol{\sigma}(\bar{\mathbf{s}})][\mathbf{S}_x(\bar{\mathbf{r}}, \mathbf{r}_i)]), \right. \\ &\quad \left. - \text{diag}([\boldsymbol{\sigma}(\bar{\mathbf{s}})][\mathbf{S}_y(\bar{\mathbf{r}}, \mathbf{r}_i)]) \right), \quad i \in I_s, \end{aligned} \quad (6.12)$$

where we wrote the state \mathbf{u}_i as $(\mathbf{r}_i, \mathbf{s}_i)$. In doing this we have replaced the infinite-dimensional system (6.10) with the finite-dimensional system (6.12), where the $\bar{\mathbf{s}}$ and the $\bar{\mathbf{r}}$ are composed of all the \mathbf{s}_i and \mathbf{r}_i , respectively ($i \in I_s \cup I_d$), but only the static \mathbf{s}_i , \mathbf{r}_i are the unknowns. Although the resulting system (6.12) is essentially finite-dimensional, applying Newton iteration requires solving elliptic equations each iteration step, needed for evaluation of the matrices \mathbf{S} , \mathbf{S}_x and \mathbf{S}_y . However, if the G_i are of the form (6.7), the system decouples. It is then possible to solve the static \mathbf{r}_i

first using the functions G_i^f . Afterwards, all \mathbf{r}_i are known and elliptic equations have to be solved to find the \mathbf{S} , \mathbf{S}_x and \mathbf{S}_y . Finally, Newton iteration is used to solve

$$0 = G_i^s \left(t, \mathbf{s}_i, -\text{diag}([\boldsymbol{\sigma}(\bar{\mathbf{s}})][\mathbf{S}]), -\text{diag}([\boldsymbol{\sigma}(\bar{\mathbf{s}})][\mathbf{S}_x]), -\text{diag}([\boldsymbol{\sigma}(\bar{\mathbf{s}})][\mathbf{S}_y]) \right), \quad i \in I_s,$$

for the static s_i , where we left out the arguments of the matrices \mathbf{S} , \mathbf{S}_x and \mathbf{S}_y .

6.3 Numerical methods in AGTools

In this section we will go into the numerical machinery implemented in AGTools for approximating solutions of the systems (6.2)-(6.6). We will start with making some general remarks on the adopted approach for discretizing the model equations.

In general there are two approaches for writing down full discretizations of time dependent PDE systems. The most used one is called the Method of Lines (MOL) approach and starts with a spatial discretization of the dependent fields and their differential equations, turning the system in a large, but finite-dimensional ODE-system, called the semi-discrete system. Then a suitable time integrator is selected for the temporal discretization to yield the fully discrete solution.

An advantage of the MOL approach is that one can choose a suitable method from a large collection of time integration methods for ODEs that are available. The downside is that the semi-discrete systems might become very complicated due to the presence of certain discretization- or interpolation operators. Direct application of, for example Rosenbrock methods, which involve the evaluation of Jacobians, can become cumbersome or even impossible.

The second approach is the so-called Rothe approach [39]. Instead of first choosing a spatial discretization it starts by selecting a time integration method. This will result in a sequence of PDEs in time containing only spatial derivatives (boundary value problems). In this approach the PDEs are often stated as an abstract ODE in a certain Banach space making that the analysis of the used time integration methods moves to the realm of functional analysis and therefore becomes much more difficult

The harder analysis however is accompanied by a number of advantages. First, the approach seems to have a cleaner appearance, not having to deal with difficult ODEs with discontinuities that are the result of spatial discretizations, but instead with elliptic equations coupled to algebraic equations, where everything is still smooth from spatial perspective. Second, it allows for nice error estimators, as is clearly described and illustrated by Lang [31].

We will adopt here the Rothe approach and not consider any functional analytic aspects, but take the practical approach in which we assume that our time integration methods work well for our cases, i.e., do not display any instability behavior.

Time integration

With respect to the time integration it is of importance that system (6.2)-(6.6) consists of stiff and non-stiff parts. The diffusion in the field equations gives rise to

stiffness, while the nonlinear state equations are not necessarily stiff. For the time integration of stiff equations we would like to use an implicit method, but using such a method becomes very complicated for the system at hand and is not really suitable for the non-stiff part as well.

IMEX scheme A class of schemes that seems to be appropriate here, is the class of IMEX (IMPLICIT/EXPLICIT) schemes. Such schemes can be applied to dynamical systems of the form $\dot{\mathbf{z}} = F_1(t, \mathbf{z}) + F_2(t, \mathbf{z})$. One part of the vectorfield $F_1 + F_2$, say F_1 , is treated implicitly by the scheme, while the other, i.e., F_2 , is treated explicitly. Different IMEX schemes have been developed, under which there are the popular IMEX-BDF schemes that are of multistep type [26]. We, however, will use a Runge-Kutta IMEX scheme, because we prefer to work with one-step methods. Especially when working with spatial adaptivity, implementation of multistep methods can become very complicated due to the fact that every time level has its own spatial discretization.

We will use an IMEX-midpoint scheme, which can be seen to be a combination of an implicit and an explicit midpoint step, and is given by

$$\begin{aligned} \mathbf{z}_s &= \mathbf{z}_n + \frac{1}{2}\tau F_1(t_n + \frac{1}{2}\tau, \mathbf{z}_s) + \frac{1}{2}\tau F_2(t_n, \mathbf{z}_n), \\ \mathbf{z}_{n+1} &= 2\mathbf{z}_s - \mathbf{z}_n + \tau \left(F_2(t_n + \frac{1}{2}\tau, \mathbf{z}_s) - F_2(t_n, \mathbf{z}_n) \right), \end{aligned} \quad (6.13)$$

where the s in \mathbf{z}_s refers to the intermediate stage. This is a second order time integration method and the implicit part, i.e., the implicit midpoint method, is A-stable. Also, using this scheme for the systems at hand never revealed any stability problems.

Application of (6.13) In the application of this method to the system (6.2)-(6.6), we use the representation (6.8) and choose the implicit and explicit parts as shown in

$$\begin{aligned} \partial_t \rho_j &= \underbrace{L_j \rho_j}_{F_1} + \sum_{i=1}^N \sigma_{ji}(\mathbf{s}_i) T_{r_i} S, & j \in J_d, \\ \partial_t \mathbf{u}_i &= 0 + \underbrace{G_i(t, \mathbf{u}_i, \boldsymbol{\rho}(\mathbf{r}_i), \partial_x \boldsymbol{\rho}(\mathbf{r}_i), \partial_y \boldsymbol{\rho}(\mathbf{r}_i))}_{F_2}, & i \in I_d. \end{aligned}$$

where these systems only represents the first equation of (6.8). Application of the IMEX-midpoint scheme leads then to the following solution process.

Starting at the beginning of a time step with values ρ_j^n for all j and \mathbf{u}_i^n for all i , the first equation of (6.13) for our system reads

$$(I - \frac{1}{2}\tau L_j) \rho_j^s = \rho_j^n + \frac{1}{2}\tau \sum_{i=1}^N \sigma_{ji}(\mathbf{s}_i^n) T_{r_i} S, \quad j \in J_d. \quad (6.14)$$

$$\mathbf{u}_i^s = \mathbf{u}_i^n + \frac{1}{2}\tau G_i^n, \quad i \in I_d. \quad (6.15)$$

where $G_i^n = G_i^n(t_n, \mathbf{u}_i^n, \boldsymbol{\rho}^n(\mathbf{r}_i^n), \partial_x \boldsymbol{\rho}^n(\mathbf{r}_i^n), \partial_y \boldsymbol{\rho}^n(\mathbf{r}_i^n))$. Equations (6.14) and (6.15) are used to determine the *dynamic* fields ρ_j^s and states \mathbf{u}_i^s , for $j \in J_d$ and $i \in I_d$, respectively. For evaluation of the *static* fields ρ_j and states \mathbf{u}_i , for $j \in J_s$ and $i \in I_s$, the second equation of (6.8), which takes the form of (6.10), will be used and reads

$$0 = L_j \rho_j^s + \sum_{i \in I = I_d \cup I_s} \sigma_{ji}(\mathbf{s}_i^s) T_{\mathbf{r}_i^s} S, \quad j \in J_s, \quad (6.16)$$

$$0 = G_i \left(t_n + \frac{1}{2} \tau, \mathbf{u}_i^s, \boldsymbol{\rho}^s(\mathbf{r}_i^s), \partial_x \boldsymbol{\rho}^s(\mathbf{r}_i^s), \partial_y \boldsymbol{\rho}^s(\mathbf{r}_i^s) \right), \quad i \in I_s, \quad (6.17)$$

Therefore, to determine the intermediate stage values, elliptic equations for the fields have to be solved for every field. For the dynamic fields these are equations (6.14) and for the static fields these are (6.16) and thus part of a larger system that can be solved using the method described at the end of Section 6.2.

Having all fields and states of the intermediate stage determined this way, we turn to the second equation of (6.13), which reads for our system

$$\rho_j^{n+1} = 2\rho_j^s - \rho_j^n + \tau \sum_{i=1}^N (\sigma_{ji}(\mathbf{s}_i^s) T_{\mathbf{r}_i^s} S - \sigma_{ji}(\mathbf{s}_i^n) T_{\mathbf{r}_i^n} S), \quad j \in J_d, \quad (6.18)$$

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n + \tau G_i^s, \quad i \in I_d, \quad (6.19)$$

where G_i^s is defined similarly as G_i^n and the upper index s refers to the intermediate stage. After using equations (6.18) and (6.19) for solving the dynamic fields ρ_j^{n+1} and states \mathbf{u}_i^{n+1} we once more have to solve a system similar to (6.16) and (6.17). Therefore, the whole time stepping procedure amounts to solving one system of linear elliptic equations (6.14) and two systems of the form (6.16) and (6.17).

Spatial discretization

For solving the field equations we use an unstructured (meshfree like) approach based on an arbitrary set of nodes in the domain. This approach facilitates dealing with complex domains, refinement and adaptivity: the latter is needed in cases where we have moving sources with small support. A thorough description of the method can be found in ([29], Chapter 4). We will briefly outline it here.

Function approximation Given function values on the nodes, we use a local least-squares approximation technique to determine for every node a second-order multinomial as a local approximation of the function around that node. For this we use the function values on a number of neighboring nodes. Because every second-order multinomial can be written as the linear combination of six basis functions, we must choose at least five neighbors for every node to determine such an approximating multinomial.

With this procedure a set of function values is mapped onto a set of local approximations around every node. If we assign to every node a part of the domain for which

we assume the local approximation to be valid, such that the whole domain is covered, this results in a global approximation. Let N_n denote the number of nodes, then for a given set of function values in a vector $\mathbf{w} \in \mathbb{R}^{N_n}$, we denote the global approximation by $F_{\text{app}}(\mathbf{w}) \in L_1(\Omega)$, where $L_1(\Omega)$ is the space of integrable real functions defined on $\Omega \subset \mathbb{R}^2$.

Voronoi diagrams For choosing neighboring nodes of nodes, as well as for assigning parts of the domain to the nodes, we use the Voronoi diagram [11]. It assigns to every node a Voronoi cell, which is the set of points closer to the node than to every other node, hence dividing the domain and at the same time creating neighbors in a natural way. Because a Voronoi diagram extends to all of \mathbb{R}^2 , we will truncate it by connecting the nodes on the boundary by straight lines, resulting in a bounded diagram. From now on all our diagrams will be truncated ones, but we will still refer to them as Voronoi diagrams. Determination of such a diagram can be done in $\mathcal{O}(N_n \log(N_n))$ operations, where N_n is the number of nodes [7]. We store the diagram in a totally disconnected edge list [7], so that searching neighboring nodes for every node becomes a process of $\mathcal{O}(N_n)$ operations.

Variational problem The stage equations (6.14) are of the form $(\alpha\Delta - \beta)\rho + f_{\text{rhs}} = 0$, with $\alpha = \frac{1}{2}\tau d_j > 0$, $\beta = 1 + \frac{1}{2}\tau k_j > 0$, f_{rhs} given by the right hand side of (6.14), and the unknown ρ_j^s replaced with ρ . Solving such equations can be done by solving the variational problem of minimizing $A_{\text{var}}(w, w) - L_{\text{var}}(S, w)$ over $w \in H^1(\Omega)$ [2, 29], where

$$A_{\text{var}}(v, w) = \int_{\Omega} \frac{1}{2}\alpha \nabla v \cdot \nabla w + \frac{1}{2}\beta vw \, dx, \quad L_{\text{var}}(f_{\text{rhs}}, w) = \int_{\Omega} f_{\text{rhs}} w \, dx.$$

A direct discretization of this problem is to minimize

$$A_{\text{var}}(F_{\text{app}}(\mathbf{w}), F_{\text{app}}(\mathbf{w})) - L_{\text{var}}(f_{\text{rhs}}, F_{\text{app}}(\mathbf{w})),$$

for all $\mathbf{w} \in \mathbb{R}^N$. After replacement of f_{rhs} with an approximation $F_{\text{app}}(\mathbf{f}_{\text{rhs}})$, where \mathbf{f}_{rhs} is the vector of node values of f_{rhs} , it can be shown ([29], Chapter 4) that sparse matrices \hat{A}_{var} and \hat{L}_{var} exist such that

$$\begin{aligned} \frac{1}{2}\mathbf{w}^T \hat{A}_{\text{var}} \mathbf{w} &= A_{\text{var}}(F_{\text{app}}(\mathbf{w}), F_{\text{app}}(\mathbf{w})), \\ \mathbf{f}_{\text{rhs}}^T \hat{L}_{\text{var}} \mathbf{w} &= L_{\text{var}}(F_{\text{app}}(\mathbf{f}_{\text{rhs}}), F_{\text{app}}(\mathbf{w})). \end{aligned}$$

If \hat{A} is non-singular the discrete problem has a unique solution $\mathbf{w} = \hat{A}_{\text{var}}^{-1} \hat{L}_{\text{var}} \mathbf{f}_{\text{rhs}}$. With the algorithm for finding the Voronoi diagram comes a lexicographical ordering of the nodes that will give the sparse matrices a band structure, which is advantageous when solving the system directly using an LU-decomposition.

Convergence tests show that the numerical solution is 2nd-order convergent in the L^2 -norm, with respect to the maximum distance between neighboring nodes ([29], Chapter 4).

Choosing nodes To distribute nodes appropriately over a domain we make use of Lloyd's algorithm [10]. This algorithm is based upon the determination of Voronoi diagrams and the process of shifting nodes to centroids of Voronoi cells. An alternating sequence of these two operations distributes the nodes equally over the domain, in the sense that distances between neighbors will tend to become equal throughout the diagram. To achieve refinement at certain points, we use a variation of Lloyd's algorithm. Here, after shifting the nodes to their centroids, an extra shift in the direction of neighboring nodes is added. To determine for a particular node which of its neighbors are attracting it, all nodes are given an integer type. Nodes will then be attracted to the neighbors with higher type than their own type. To get refinement around a certain point in the domain, a node is fixed at that point and several rings of decreasing node type are defined around it. The extended Lloyd's algorithm then moves nodes around, which results in a refinement around the fixed node.

6.4 Introduction to Kardos

In short, Kardos [31] is a software package that can be used to approximate solutions of systems of nonlinear parabolic equations. Its main features are that it follows the Rothe approach using Rosenbrock-type time integration methods and multilevel finite element methods. It makes use of a posteriori error estimates for local refinement and adaptivity in space and time. We will now consider both the used Rosenbrock methods and finite elements methods in some detail.

Rosenbrock-type time integration

We first consider Rosenbrock schemes in general for a system $\dot{\mathbf{z}} = F(t, \mathbf{z})$ in \mathbb{R}^m , see [20, 31, 26]. The scheme determines from \mathbf{z}_n given at time level t_n , \mathbf{z}_{n+1} at time level $t_{n+1} = t_n + \tau$. To accomplish this it uses s stage vectors \mathbf{k}_i , $i = 1, \dots, s$, and coupled to that the arguments \mathbf{z}_{n_i} , used for function evaluation. To write down the scheme we use the following notation, which results in a compact way of writing the Rosenbrock formulas (different from [20, 31, 26]):

$$K_s = [\mathbf{k}_1 \dots \mathbf{k}_s], \quad Z_{ns} = [\mathbf{z}_{n1} \dots \mathbf{z}_{ns}], \quad \mathbf{t}_{ns} = [t_{n1} \dots t_{ns}]^T \in \mathbb{R}^s, \\ F_{ns} = [F(t_{n1}, \mathbf{z}_{n1}) \dots F(t_{ns}, \mathbf{z}_{ns})], \quad \mathbf{1} = [1 \dots 1]^T \in \mathbb{R}^s.$$

Note that K_s , Z_{ns} and F_{ns} are matrices in $\mathbb{R}^{m \times s}$. The Rosenbrock method is completely defined by the $s \times s$ coefficient matrices A (strictly lower triangular) and Γ (lower triangular), here the latter having every diagonal entry equal to γ , and the s -dimensional coefficient vector \mathbf{b} . The scheme reads then

$$K_s = F_{ns} + \tau[D_{\mathbf{z}}F]K_s\Gamma^T + \tau[\partial_t F](\Gamma\mathbf{1})^T, \quad Z_{ns} = u_n\mathbf{1}^T + \tau K_s A^T, \\ \mathbf{z}_{n+1} = \mathbf{z}_n + \tau K_s \mathbf{b}, \quad \mathbf{t}_{ns} = t_n\mathbf{1} + \tau A\mathbf{1}.$$

where $[D_{\mathbf{z}}F]$ denotes the Jacobian matrix at (t_n, \mathbf{z}_n) . Likewise, $[\partial_t F]$ denotes the partial derivative at (t_n, \mathbf{z}_n) with respect to time t . Because of the fact that the matrix Γ is lower triangular we can solve successively for the columns of K_s , i.e., the stages \mathbf{k}_i .

However, it has turned out that we can avoid a number of matrix-vector multiplications if we work with stages defined by the columns of the matrix $Z_s = \tau K_s \Gamma^T$, as is well-known. The resulting system is then

$$\begin{aligned} \frac{1}{\tau} Z_s \Gamma^{-T} &= F_{n_s} + [D_{\mathbf{z}}F] Z_s + \tau [\partial_t F] (\Gamma \mathbb{1})^T, & Z_{n_s} &= u_n \mathbb{1}^T + Z_s (A \Gamma^{-1})^T, \\ \mathbf{z}_{n+1} &= \mathbf{z}_n + Z_s (\Gamma^{-T} \mathbf{b}), & \mathbf{t}_{n_s} &= t_n \mathbb{1} + \tau A \mathbb{1}, \end{aligned} \quad (6.20)$$

in which we will denote the columns of the new stage matrix Z_s by \mathbf{z}_{si} . Again the columns can be solved successively after rewriting the equation for the matrix Z_s as

$$\left(\frac{1}{\tau\gamma} I - [D_{\mathbf{z}}F] \right) Z_s = F_{n_s} - \frac{1}{\tau} Z_s \left(\Gamma^{-1} - \frac{1}{\gamma} I \right)^T + \tau [\partial_t F] (\Gamma \mathbb{1})^T,$$

where $(\Gamma^{-1} - \frac{1}{\gamma} I)$ and $(A \Gamma^{-1})$ are strictly lower triangular matrices. Therefore, with \mathbf{e}_i the vector with the i th component equal to one and zero otherwise, we have for all the stages $\mathbf{z}_{si} = Z_s \mathbf{e}_i$ the system

$$\mathbf{z}_{ni} = \mathbf{z}_n + Z_s (\mathbf{e}_i^T A \Gamma^{-1})^T, \quad t_{ni} = t_n + \tau (\mathbf{e}_i^T A \mathbb{1}), \quad (6.21)$$

$$\left(\frac{1}{\tau\gamma} I - [D_{\mathbf{z}}F] \right) \mathbf{z}_{si} = F(t_{ni}, \mathbf{z}_{ni}) - \frac{1}{\tau} Z_s \left(\mathbf{e}_i^T \left(\Gamma^{-1} - \frac{1}{\gamma} I \right) \right)^T + \tau [\partial_t F] (\mathbf{e}_i^T \Gamma \mathbb{1})^T. \quad (6.22)$$

In Kardos actually a more sophisticated Rosenbrock method is being used, which can handle more general systems $H(t, \mathbf{z}) \dot{\mathbf{z}} = F(t, \mathbf{z})$. However, the resulting method will be equivalent to (6.20) for our system $\dot{\mathbf{z}} = F(t, \mathbf{z})$. Therefore we do not describe this method here.

Example: ROS2 As an example consider the 2nd-order method ROS2 [26] defined by

$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad \Gamma = \gamma \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}, \quad (6.23)$$

which is L-stable for $\gamma = 1 \pm \frac{1}{2} \sqrt{2}$. It is also a so-called W-method, meaning it is still of 2nd-order when using arbitrary approximations of the Jacobians $[D_{\mathbf{z}}F]$ and $[\partial_t F]$. We will apply (6.23) as a W-method in Section 6.5. The stage equations from (6.21) and (6.22) combined with the time step equation for \mathbf{z}_{n+1} in (6.20) yield the scheme

$$\begin{aligned} \mathbf{z}_{n1} &= \mathbf{z}_n, & t_{n1} &= t_n, \\ \left(\frac{1}{\tau\gamma} I - [D_{\mathbf{z}}F] \right) \mathbf{z}_{s1} &= F(t_{n1}, \mathbf{z}_{n1}) + \tau\gamma [\partial_t F], \\ \mathbf{z}_{n2} &= \mathbf{z}_n + \frac{1}{\gamma} \mathbf{z}_{s1}, & t_{n2} &= t_n + \tau, \\ \left(\frac{1}{\tau\gamma} I - [D_{\mathbf{z}}F] \right) \mathbf{z}_{s2} &= F(t_{n2}, \mathbf{z}_{n2}) - \frac{2}{\tau\gamma} \mathbf{z}_{s1} - \tau\gamma [\partial_t F], \\ \mathbf{z}_{n+1} &= \mathbf{z}_n + \frac{3}{2\gamma} \mathbf{z}_{s1} + \frac{1}{2\gamma} \mathbf{z}_{s2}. \end{aligned}$$

Multilevel finite elements

To solve the system of linear elliptic boundary problems (6.22) Kardos utilizes a multilevel finite element method. Its main idea is to replace the solution space by a sequence of discrete spaces that have successively increasing dimensions yielding improving approximations of the solution.

Assume there exist an admissible finite element mesh T_h at $t = t_n$ and an associated finite dimensional space S_h consisting of all continuous functions ϕ that are polynomials of order q when restricted to an arbitrary element $T \in T_h$. The standard Galerkin finite element approximation satisfies the equation

$$(\tilde{L}_n \mathbf{z}_{si}^h, \phi) = (r_{ni}, \phi) \text{ for all } \phi \in S_h. \quad (6.24)$$

Here \tilde{L}_n is the weak representation of the differential operator $\frac{1}{\tau\gamma}I - [D_{\mathbf{z}}F]$ on the left-hand-side in (6.22) and r_{ni} stands for the entire right-hand-side in (6.22). Since the operator \tilde{L}_n is independent of the stages i its calculation is required only once within each time step.

Stabilization To overcome the well-known inconvenience that the solutions \mathbf{z}_{si} may suffer from numerical oscillations caused by dominating convective terms, Kardos uses a stabilized discretization by adding locally weighted residuals, resulting in

$$(\tilde{L}_n \mathbf{z}_{si}^h, \phi) + \sum_{T \in T_h} (\tilde{L}_n \mathbf{z}_{si}^h, w(\phi))_T = (r_{ni}, \phi) + \sum_{T \in T_h} (\mathbf{r}_{si}^h, w(\phi))_T \quad (6.25)$$

for all $\phi \in S_h$. Here $w(\phi)$ is defined with respect to the operator \tilde{L}_n . Two important classes of stabilized methods are the streamline diffusion and the more general Galerkin/least-squares finite element method, both of which can be chosen in Kardos.

A posteriori error estimates A posteriori error estimates provide the appropriate framework to determine where a mesh refinement is necessary and where degrees of freedom are no longer needed.

After computing the approximate intermediate values \mathbf{z}_{si}^h a posteriori error estimates can be used to provide specific assessment of the error distribution. Considering a hierarchical decomposition

$$S_h^{q+1} = S_h^q \oplus Z_h^{q+1} \quad (6.26)$$

where Z_h^{q+1} is the subspace that corresponds to the span of all additional basis functions needed to extend the space S_h^q to higher order, an attractive idea of an efficient error estimation is to bound the spatial error by evaluating its components in the space Z_h^{q+1} only. This technique, which is known as hierarchical error estimation, has been carried over to time-dependent nonlinear problems in [31]. Defining an a posteriori error estimator $E_{n+1}^h \in Z_h^{q+1}$ by

$$E_{n+1}^h = E_{n0}^h + \sum_{i=1}^s m_i E_{si}^h \quad (6.27)$$

with $E_{n,0}^h$ approximating the projection error of the initial value u_n in Z_h^{q+1} and E_{si}^h estimating the spatial error of the intermediate value \mathbf{z}_{si}^h , the local spatial error for a finite element $T \in T_h$ can be estimated by $\eta_T \equiv \|E_{n+1}^h\|_T$. The error estimator E_{n+1}^h is computed by linear systems which can be derived from (6.25).

For practical computations the spatially global calculation of E_{n+1}^h is normally approximated by a small element-by-element calculation. This leads to an efficient algorithm for computing a posteriori error estimates which can be used to determine an adaptive strategy to improve the accuracy of the numerical approximation where needed. A rigorous a posteriori error analysis for a Rosenbrock-Galerkin finite element method applied to nonlinear parabolic systems is given in [31].

Refinement In order to produce a nearly optimal mesh, those finite elements T having an error η_T larger than a certain threshold are refined. After the refinement improved finite element solutions \mathbf{z}_{si}^h defined by (6.25) are computed. The whole procedure solve-estimate-refine is applied several times until a prescribed spatial tolerance $\|E_{n+1}^h\| \leq \text{TOL}_r$ is reached. To maintain the nesting property of the finite element subspaces coarsening takes place only after an accepted time step before starting the multilevel process at a new time. Regions of small errors are identified by their η -values.

Linear systems The linear systems that arise in the Galerkin procedure Kardos can be solved by direct or iterative methods. The user can choose from a collection of methods, like for example the direct solver MA28 and the iterative solver BiCGStab with ILU-preconditioning.

6.5 Application of Rosenbrock methods

Because Kardos works with Rosenbrock-type time integration methods, we want to examine whether we can easily apply such methods to the system (6.8). Here, we will assume that $D_{\mathbf{y}}g(t, \mathbf{z}, \mathbf{y})$ is invertible and that therefore we can consider the reduced system (6.9) instead of (6.8). The Rosenbrock methods are directly applicable to this reduced system.

Setting $F(t, \mathbf{z}) = f(t, \mathbf{z}, h(t, \mathbf{z}))$ and differentiating the function F and the equation $0 = g(t, \mathbf{z}, h(t, \mathbf{z}))$ with respect to \mathbf{z} and t gives the expressions

$$D_{\mathbf{z}}F = D_{\mathbf{z}}f - D_{\mathbf{y}}f(D_{\mathbf{y}}g)^{-1}D_{\mathbf{z}}g, \quad \partial_t F = \partial_t f - D_{\mathbf{y}}f(D_{\mathbf{y}}g)^{-1}\partial_t g.$$

Direct application of the Rosenbrock scheme (6.20) gives therefore for the stage matrix Z_s the equation

$$\frac{1}{\tau}Z_s\Gamma^{-T} = F_{n,s} + \left[D_{\mathbf{z}}f - D_{\mathbf{y}}f(D_{\mathbf{y}}g)^{-1}D_{\mathbf{z}}g \right] Z_s + \tau \left[\partial_t f - D_{\mathbf{y}}f(D_{\mathbf{y}}g)^{-1}\partial_t g \right] (\Gamma \mathbb{I})^T.$$

In a finite-dimensional setting solving this equation every stage could be done because the matrix $(D_{\mathbf{y}}g)^{-1}$ can be calculated exactly. Here, in the infinite-dimensional

setting we do not have an explicit form of this inverse and we therefore introduce an additional stage matrix Y_s and replace the system by the equivalent system

$$\begin{aligned} \frac{1}{\tau} Z_s \Gamma^{-T} &= F_{n_s} + [D_{\mathbf{z}} f] Z_s + [D_{\mathbf{y}} f] Y_s + \tau [\partial_t f] (\Gamma \mathbf{1})^T \\ [D_{\mathbf{y}} g] Y_s &= -[D_{\mathbf{z}} g] Z_s - \tau [\partial_t g] (\Gamma \mathbf{1})^T, \end{aligned}$$

which does not involve the inverse operator $(D_{\mathbf{y}} g)^{-1}$ explicitly. ‘Column-wise’ solving requires that we rewrite this as

$$\begin{aligned} \left(\frac{1}{\tau} I - [D_{\mathbf{z}} f] \right) Z_s - [D_{\mathbf{y}} f] Y_s &= F_{n_s} - \frac{1}{\tau} Z_s \left(\Gamma^{-1} - \frac{1}{\gamma} I \right)^T + \tau [\partial_t f] (\Gamma \mathbf{1})^T, \\ -[D_{\mathbf{z}} g] Z_s - [D_{\mathbf{y}} g] Y_s &= \tau [\partial_t g] (\Gamma \mathbf{1})^T. \end{aligned}$$

For a single time step the following has to be done. At the beginning of a step the linear operators $D_{\mathbf{z}} f$, $D_{\mathbf{y}} f$, $D_{\mathbf{z}} g$, $D_{\mathbf{y}} g$, $\partial_t f$ and $\partial_t g$, at time level n have to be solved. During every stage first the vectors \mathbf{z}_{ni} and \mathbf{y}_{ni} from

$$t_{ni} = t_n + \tau (\mathbf{e}_i^T A \mathbf{1}), \quad \mathbf{z}_{ni} = \mathbf{u}_n + Z_s (\mathbf{e}_i^T A \Gamma^{-1})^T, \quad 0 = g(t_{ni}, \mathbf{z}_{ni}, \mathbf{y}_{ni}), \quad (6.28)$$

have to be solved. The stage is completed by solving the vectors $\mathbf{z}_{si} = Z_s \mathbf{e}_i$ and $\mathbf{y}_{si} = Z_s \mathbf{e}_i$ from

$$\begin{bmatrix} \frac{1}{\tau} I - D_{\mathbf{z}} f & -D_{\mathbf{y}} f \\ -D_{\mathbf{z}} g & -D_{\mathbf{y}} g \end{bmatrix} \begin{pmatrix} \mathbf{z}_{si} \\ \mathbf{y}_{si} \end{pmatrix} = \begin{pmatrix} f(t_{ni}, \mathbf{z}_{ni}, \mathbf{y}_{ni}) - \frac{1}{\tau} Z_s C^T \mathbf{e}_i + \tau \gamma_i [\partial_t f] \\ \tau \gamma_i [\partial_t g] \end{pmatrix}, \quad (6.29)$$

where we defined $C = \Gamma^{-1} - \frac{1}{\gamma} I$ and $\gamma_i = (\Gamma \mathbf{1})^T \mathbf{e}_i$. After the last stage the values on time level $n+1$ are obtained by solving

$$t_{n+1} = t_n + \tau, \quad \mathbf{z}_{n+1} = \mathbf{z}_n + Z_s (\Gamma^{-T} \mathbf{b}), \quad 0 = g(t_{n+1}, \mathbf{z}_{n+1}, \mathbf{y}_{n+1}). \quad (6.30)$$

We will now focus on solving the stage system (6.29). This system contains linear operators like $D_{\mathbf{u}} f$ which are represented by Jacobian matrices in the finite-dimensional case, but here we have to consider them as the more general Fréchet derivatives. For example, the Fréchet derivative of f with respect to \mathbf{z} denoted by $[D_{\mathbf{z}} f(t, \mathbf{z}, \mathbf{y})]$ is defined through

$$[D_{\mathbf{z}} f(t, \mathbf{z}, \mathbf{y})] \mathbf{v} = \left. \frac{d}{d\epsilon} f(t, \mathbf{z} + \epsilon \mathbf{v}, \mathbf{y}) \right|_{\epsilon=0},$$

for all variation vectors \mathbf{v} that live in the same space as \mathbf{z} .

Example system To prevent from immediately getting lost in complex formulae when applying this definition to the functions f and g in our general system (6.8), we will first consider an example system that is relatively simple. This system contains only a dynamic part \mathbf{z} , with the dynamics given by f , and lacks the static part \mathbf{y} and accompanying algebraic equations, given by the function g . Further it consists of one

field ρ and one state $\mathbf{u} = (\mathbf{r}^T, \mathbf{s}^T)^T$ only and is not explicitly time dependent. The abstract ODE is then given by

$$\frac{d}{dt} \begin{pmatrix} \rho \\ \mathbf{u} \end{pmatrix} = F(\rho, \mathbf{u}), \quad F(\rho, \mathbf{u}) = \begin{pmatrix} L\rho + \sigma(\mathbf{s})T_{\mathbf{r}}S \\ G(\mathbf{r}, \mathbf{s}, \rho(\mathbf{r}), \partial_x \rho(\mathbf{r}), \partial_y \rho(\mathbf{r})) \end{pmatrix}. \quad (6.31)$$

The Fréchet derivative applied to a variation vector (η, \mathbf{v}) , with $\mathbf{v} = (\mathbf{p}^T, \mathbf{q}^T)^T$, is defined by $\left. \frac{d}{d\epsilon} f(\rho + \epsilon\eta, \mathbf{u} + \epsilon\mathbf{v}) \right|_{\epsilon=0}$ and reads for the first component

$$\begin{aligned} \left. \frac{d}{d\epsilon} F_1(\dots) \right|_{\epsilon=0} &= L\eta - \sigma(\mathbf{s}) [\partial_x T_{\mathbf{r}}S \quad \partial_y T_{\mathbf{r}}S] \mathbf{p} + ([D\sigma(\mathbf{s})]\mathbf{q})T_{\mathbf{r}}S \\ &= L\eta + \left[\begin{array}{cc} -\sigma(\mathbf{s})\partial_x T_{\mathbf{r}}S & -\sigma(\mathbf{s})\partial_y T_{\mathbf{r}}S \end{array} \right] \left[([D\sigma(\mathbf{s}))T_{\mathbf{r}}S] \right] \mathbf{v}, \end{aligned} \quad (6.32)$$

and for the second component

$$\begin{aligned} \left. \frac{d}{d\epsilon} F_2(\dots) \right|_{\epsilon=0} &= [D_{\mathbf{r}}G \quad D_{\mathbf{s}}G] \begin{pmatrix} \mathbf{p} \\ \mathbf{q} \end{pmatrix} + (D_{\rho}G) ([D_{\mathbf{x}}\rho]\mathbf{p} + \eta(\mathbf{r})) \\ &\quad + (D_{\rho_x}G) ([D_{\mathbf{x}}(\partial_x \rho)]\mathbf{p} + (\partial_x \eta)(\mathbf{r})) \\ &\quad + (D_{\rho_y}G) ([D_{\mathbf{x}}(\partial_y \rho)]\mathbf{p} + (\partial_y \eta)(\mathbf{r})) \\ &= \left((D_{\rho}G) P_{\mathbf{r}} + (D_{\rho_x}G) P_{\mathbf{r}}\partial_x + (D_{\rho_y}G) P_{\mathbf{r}}\partial_y \right) \eta + \\ &\quad \left[\begin{array}{l} [(D_{\mathbf{r}}G) + (D_{\rho}G) [D_{\mathbf{x}}\rho] + (D_{\rho_x}G) [D_{\mathbf{x}}(\partial_x \rho)] \\ + (D_{\rho_y}G) [D_{\mathbf{x}}(\partial_y \rho)]] \quad [D_{\mathbf{s}}G] \end{array} \right] \mathbf{v}. \end{aligned} \quad (6.33)$$

Note that F_1 and F_2 denote here the vectorial components of F while in the IMEX scheme (6.13) they denote terms that sum up to F . The linear operator $P_{\mathbf{r}}$ used in the second component is the 'point evaluation' operator defined by $P_{\mathbf{r}}\eta = \eta(\mathbf{r})$ for arbitrary fields η . We use it here because it enables us to write down the linear systems that arise during the Rosenbrock stages (6.29) in a form that is analogous to the matrix notation of finite-dimensional linear equation systems.

For our example function the linear stage system (6.29) lacks the static components \mathbf{y} and the second equation, and is therefore of the form

$$\left[\frac{1}{\gamma\tau} I - [DF(\rho, \mathbf{u})] \right] \begin{pmatrix} \eta \\ \mathbf{v} \end{pmatrix} = F(\rho_{ni}, \mathbf{u}_{ni}) - \frac{1}{\tau} \sum_{j < i} c_{ij} \begin{pmatrix} \rho_{sj} \\ \mathbf{u}_{sj} \end{pmatrix}. \quad (6.34)$$

with (η, \mathbf{v}) denoting $(\rho_{si}, \mathbf{u}_{si})$ and c_{ij} the entries of C . This can be written as

$$\left[\begin{array}{cc} \frac{1}{\gamma\tau} I - L & -\mathbf{S} \\ -\mathbf{a}P_{\mathbf{r}} - \mathbf{a}^x P_{\mathbf{r}}\partial_x - \mathbf{a}^y P_{\mathbf{r}}\partial_y & \frac{1}{\gamma\tau} I - A \end{array} \right] \begin{pmatrix} \eta \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \alpha \\ \mathbf{w} \end{pmatrix}. \quad (6.35)$$

where the matrices \mathbf{S} and A and the vectors \mathbf{a} , \mathbf{a}^x and \mathbf{a}^y are implicitly defined by equations (6.32) and (6.33), and (α, \mathbf{w}) is given by the right hand side of (6.34).

For convenience we will define $\tilde{L} = L - \frac{1}{\gamma\tau}I$ and $\tilde{A} = A - \frac{1}{\gamma\tau}I$. The linear system (6.35) can be solved by first expressing the field as a linear combination of the state components, resulting in

$$\eta = -(\tilde{L}^{-1}\alpha) - [\tilde{L}^{-1}\mathbf{S}]\mathbf{v} \implies \begin{cases} \eta(\mathbf{r}) = -(\tilde{L}^{-1}\alpha)(\mathbf{r}) - [(\tilde{L}^{-1}\mathbf{S})(\mathbf{r})]\mathbf{v}, \\ \partial_x\eta(\mathbf{r}) = -\partial_x(\tilde{L}^{-1}\alpha)(\mathbf{r}) - [\partial_x(\tilde{L}^{-1}\mathbf{S})(\mathbf{r})]\mathbf{v}, \\ \partial_y\eta(\mathbf{r}) = -\partial_y(\tilde{L}^{-1}\alpha)(\mathbf{r}) - [\partial_y(\tilde{L}^{-1}\mathbf{S})(\mathbf{r})]\mathbf{v}. \end{cases} \quad (6.36)$$

Here, \mathbf{S} is a $(1 \times n)$ -matrix ($n = \dim(\mathbf{u})$) of fields and the notation $\tilde{L}^{-1}\mathbf{S}$ stands for $[\tilde{L}^{-1}S_{11} \ \dots \ \tilde{L}^{-1}S_{1n}]$. Evaluation of this vector in \mathbf{r} is defined to be component-wise evaluation in \mathbf{r} . Using these equations we can derive the following finite-dimensional system for the vector $(\eta(\mathbf{r}), \partial_x\eta(\mathbf{r}), \partial_y\eta(\mathbf{r}), \mathbf{v})$,

$$\begin{bmatrix} 1 & 0 & 0 & [(\tilde{L}^{-1}\mathbf{S})(\mathbf{r})] \\ 0 & 1 & 0 & [\partial_x(\tilde{L}^{-1}\mathbf{S})(\mathbf{r})] \\ 0 & 0 & 1 & [\partial_y(\tilde{L}^{-1}\mathbf{S})(\mathbf{r})] \\ \mathbf{a} & \mathbf{a}^x & \mathbf{a}^y & \tilde{A} \end{bmatrix} \begin{pmatrix} \eta(\mathbf{r}) \\ \partial_x\eta(\mathbf{r}) \\ \partial_y\eta(\mathbf{r}) \\ \mathbf{v} \end{pmatrix} = - \begin{pmatrix} \tilde{L}^{-1}\alpha(\mathbf{r}) \\ \partial_x(\tilde{L}^{-1}\alpha)(\mathbf{r}) \\ \partial_y(\tilde{L}^{-1}\alpha)(\mathbf{r}) \\ \mathbf{w} \end{pmatrix}. \quad (6.37)$$

The solution of this system gives the solutions of the fields through (6.36).

General case This procedure can be generalized to the general case where the linear equation (6.29) takes the form

$$\begin{bmatrix} \tilde{L} & \mathbf{S} \\ \left(\begin{pmatrix} V_1 P_{\mathbf{r}_1} \\ \vdots \\ V_N P_{\mathbf{r}_N} \end{pmatrix} + \begin{pmatrix} V_1^x P_{\mathbf{r}_1} \\ \vdots \\ V_N^x P_{\mathbf{r}_N} \end{pmatrix} \partial_x + \begin{pmatrix} V_1^y P_{\mathbf{r}_1} \\ \vdots \\ V_N^y P_{\mathbf{r}_N} \end{pmatrix} \partial_y \right) & \tilde{A} \end{bmatrix} \begin{bmatrix} \eta \\ \bar{\mathbf{v}} \end{bmatrix} = - \begin{bmatrix} \alpha \\ \bar{\mathbf{w}} \end{bmatrix}. \quad (6.38)$$

Equation (6.38) contains the following elements

- The diagonal operator $\tilde{L} = \text{diag}((\tilde{L}_1, \dots, \tilde{L}_M))$, with $\tilde{L}_j = L_j$ for the static fields $j \in J_s$ and $\tilde{L}_j = L_j - \frac{1}{\gamma\tau}I$, for the dynamic fields $j \in J_d$.
- The diagonal matrix $\tilde{A} = \text{diag}((\tilde{A}_1, \dots, \tilde{A}_N))$ with $\tilde{A}_i = A_i$ for the static states $i \in I_s$ and $\tilde{A}_i = A_i - \frac{1}{\gamma\tau}I$, for the dynamic fields $i \in I_d$, and

$$A_i = \left[\begin{array}{c} [D_\rho G_i][D_x \rho] + [D_{\partial_r \rho} G_i][D_x(\partial_x \rho)] + [D_{\partial_y \rho} G_i][D_x(\partial_y \rho)] \\ [D_s, G_i] \end{array} \right]. \quad (6.39)$$

- The full matrix \mathbf{S} composed of $M \times N$ blocks S_{ji} , with dimensions $(1 \times \dim(\mathbf{u}_i))$, defined by

$$S_{ji} = \begin{bmatrix} -\sigma_{ji}(\mathbf{s}_i)(T_{\mathbf{r}_i}, \partial_x S) & -\sigma_{ji}(\mathbf{s}_i)(T_{\mathbf{r}_i}, \partial_y S) & [D\sigma_{ji}(\mathbf{s}_i)](T_{\mathbf{r}_i}, S) \end{bmatrix}. \quad (6.40)$$

- The vector $[\boldsymbol{\alpha}, \bar{\mathbf{w}}]^T$ given by the right hand side of (6.29), where the components have been reordered such that the fields are on top.
- The matrices V_i , V_i^x and V_i^y are given by

$$V_i = [D_{\rho} G_i], \quad V_i^x = [D_{\partial_x \rho} G_i], \quad V_i^y = [D_{\partial_y \rho} G_i]. \quad (6.41)$$

The unknown dynamic components \mathbf{z}_{si} (fields and states) and the unknown static components \mathbf{y}_{si} (fields and states) are denoted by the fields η_j and states \mathbf{v}_i . Compared with equation (6.29) the components are reordered, putting the fields before the states as in the original ordering in the model state \mathbf{x} of (6.1).

System (6.38) can be solved in a way completely analogous to solving system (6.35). Expressing the η_j in the \mathbf{v}_i , using the first M equations results in

$$\eta_j = -(\tilde{L}_j^{-1} \alpha_j) - \sum_{i=1}^N [\tilde{L}_j^{-1} S_{ji}] \mathbf{v}_i \implies \boldsymbol{\eta} = -(\tilde{L}^{-1} \boldsymbol{\alpha}) - [\tilde{L}^{-1} \mathbf{S}] \mathbf{v}. \quad (6.42)$$

such that for all k

$$\begin{aligned} \boldsymbol{\eta}(\mathbf{r}_k) &= -(\tilde{L}^{-1} \boldsymbol{\alpha})(\mathbf{r}_k) - [\tilde{L}^{-1} \mathbf{S}(\mathbf{r}_k)] \mathbf{v}, \\ \partial_x \boldsymbol{\eta}(\mathbf{r}_k) &= -\partial_x (\tilde{L}^{-1} \boldsymbol{\alpha})(\mathbf{r}_k) - [\partial_x (\tilde{L}^{-1} \mathbf{S})(\mathbf{r}_k)] \mathbf{v}, \\ \partial_y \boldsymbol{\eta}(\mathbf{r}_k) &= -\partial_y (\tilde{L}^{-1} \boldsymbol{\alpha})(\mathbf{r}_k) - [\partial_y (\tilde{L}^{-1} \mathbf{S})(\mathbf{r}_k)] \mathbf{v}. \end{aligned} \quad (6.43)$$

We now define the vectors

$$\bar{\boldsymbol{\eta}} = \begin{bmatrix} \boldsymbol{\eta}(\mathbf{r}_1) \\ \vdots \\ \boldsymbol{\eta}(\mathbf{r}_N) \end{bmatrix}, \quad \bar{\boldsymbol{\eta}}_x = \begin{bmatrix} \partial_x \boldsymbol{\eta}(\mathbf{r}_1) \\ \vdots \\ \partial_x \boldsymbol{\eta}(\mathbf{r}_N) \end{bmatrix}, \quad \bar{\boldsymbol{\eta}}_y = \begin{bmatrix} \partial_y \boldsymbol{\eta}(\mathbf{r}_1) \\ \vdots \\ \partial_y \boldsymbol{\eta}(\mathbf{r}_N) \end{bmatrix}, \quad (6.44)$$

all elements of \mathbb{R}^{MN} and the $(MN \times \dim(\mathbf{v}))$ -matrices

$$\bar{\mathbf{S}} = \begin{bmatrix} [\tilde{L}^{-1} \mathbf{S}(\mathbf{r}_1)] \\ \vdots \\ [\tilde{L}^{-1} \mathbf{S}(\mathbf{r}_N)] \end{bmatrix}, \quad \bar{\mathbf{S}}_x = \begin{bmatrix} [\partial_x (\tilde{L}^{-1} \mathbf{S})(\mathbf{r}_1)] \\ \vdots \\ [\partial_x (\tilde{L}^{-1} \mathbf{S})(\mathbf{r}_N)] \end{bmatrix}, \quad \bar{\mathbf{S}}_y = \begin{bmatrix} [\partial_y (\tilde{L}^{-1} \mathbf{S})(\mathbf{r}_1)] \\ \vdots \\ [\partial_y (\tilde{L}^{-1} \mathbf{S})(\mathbf{r}_N)] \end{bmatrix}. \quad (6.45)$$

the MN -dimensional vectors

$$\bar{\boldsymbol{\alpha}} = \begin{bmatrix} [\tilde{L}^{-1} \boldsymbol{\alpha}(\mathbf{r}_1)] \\ \vdots \\ [\tilde{L}^{-1} \boldsymbol{\alpha}(\mathbf{r}_N)] \end{bmatrix}, \quad \bar{\boldsymbol{\alpha}}_x = \begin{bmatrix} [\partial_x (\tilde{L}^{-1} \boldsymbol{\alpha})(\mathbf{r}_1)] \\ \vdots \\ [\partial_x (\tilde{L}^{-1} \boldsymbol{\alpha})(\mathbf{r}_N)] \end{bmatrix}, \quad \bar{\boldsymbol{\alpha}}_y = \begin{bmatrix} [\partial_y (\tilde{L}^{-1} \boldsymbol{\alpha})(\mathbf{r}_1)] \\ \vdots \\ [\partial_y (\tilde{L}^{-1} \boldsymbol{\alpha})(\mathbf{r}_N)] \end{bmatrix}. \quad (6.46)$$

and the block-diagonal matrices $V = \text{diag}((V_1, \dots, V_N))$, $V^x = \text{diag}((V_1^x, \dots, V_N^x))$ and $V^y = \text{diag}((V_1^y, \dots, V_N^y))$. Using these definitions, the analog of system (6.37) for the general case, is

$$\left[\begin{array}{ccc|ccc} I_{MN} & & & \bar{\mathbf{S}} & & \\ & I_{MN} & & \bar{\mathbf{S}}_x & & \\ & & I_{MN} & \bar{\mathbf{S}}_y & & \\ \hline & & & \tilde{A}_1 & & \\ & & & & \ddots & \\ V & V^x & V^y & & & \tilde{A}_N \end{array} \right] \begin{bmatrix} \bar{\boldsymbol{\eta}} \\ \bar{\boldsymbol{\eta}}_x \\ \bar{\boldsymbol{\eta}}_y \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} = - \begin{bmatrix} \bar{\boldsymbol{\alpha}} \\ \bar{\boldsymbol{\alpha}}_x \\ \bar{\boldsymbol{\alpha}}_y \\ \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_N \end{bmatrix}. \quad (6.47)$$

Concluding, every time step s linear stage systems (6.29), which are of the form (6.38), have to be solved, all of which have the same linear operator. Solving such a system requires a reformulation (6.42) and (6.47). As the matrix in the linear system (6.47) consists only of elements of the original operator (6.38), its construction is also needed once per time step only.

The most expensive part of the construction of the matrix in (6.38) is evaluation of the matrices \mathbf{S} , \mathbf{S}_x and \mathbf{S}_y through equations (6.45). For these, elliptic equations have to be solved, whereas evaluation of the matrices V , V^x , V^y , and $\tilde{A}_1, \dots, \tilde{A}_N$ doesn't require the solution of PDEs. From the perspective of efficiency it would be advantageous if the evaluation of \mathbf{S} , \mathbf{S}_x and \mathbf{S}_y could be omitted.

When using a member of a subclass of Rosenbrock methods, called the W-methods [26], we can retain order of consistency, while using approximations of the Jacobian operators instead of the exact Jacobian operators. In our case we could ignore \mathbf{S} , \mathbf{S}_x and \mathbf{S}_y and replace these operators by the zero operator. This is equivalent to replacing the operator \mathbf{S} with the zero operator in the stage system (6.38). This will greatly reduce the amount of work to be done. When applying the ROS2 method, which is also a W-method, in the next section, we will take this even a step further and also replace the matrices V_i , V_i^x , V_i^y and \tilde{A} with zero matrices.

6.6 Comparison between Kardos and AGTools

In this section we will examine how Kardos and AGTools can be used on a set of test problems. We will start with some general aspects of using Kardos for simulation of the systems at hand.

Domain definition For the definition of the domains of the equations (6.3) and (6.4) AGTools uses a set of closed paths of cubic Bézier curves [51]. A single path describes the boundary and additional others can be used to specify holes. Using this technique one can specify rather complex domains with relatively little points, while ensuring that the total boundary is C^1 .

An example is shown in the left picture of Figure 6.1. The numbers along the axes do not have any physical meaning here and only provide a frame of reference.

The control points that specify the Bézier curves are shown in Figure 6.2. The three paths consist out of 4, 3 and 5 points, respectively.

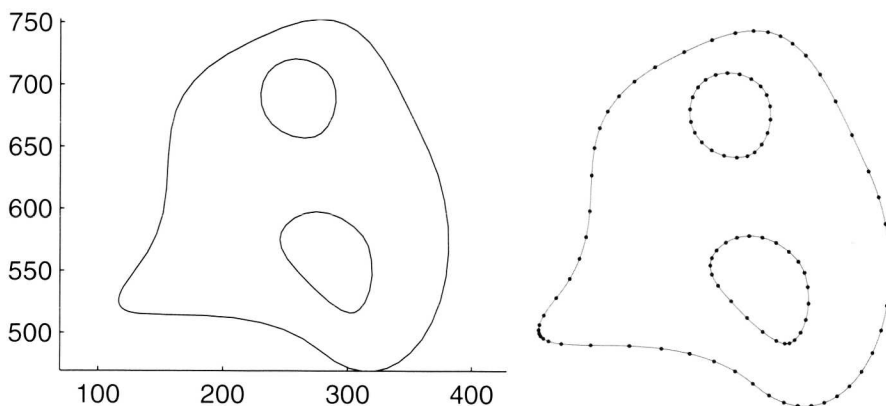


Figure 6.1: Domain and nodes assigned to the boundary

Outer boundary:

(130, 550)—(074, 481)—(208, 545)—(279, 485)
 (279, 485)—(350, 425)—(412, 545)—(365, 647)
 (365, 647)—(317, 749)—(303, 779)—(211, 729)
 (211, 729)—(119, 679)—(185, 618)—(130, 550)

Boundary hole 1:

(273, 658)—(257, 653)—(228, 666)—(230, 694)
 (230, 694)—(232, 722)—(262, 728)—(280, 713)
 (280, 713)—(299, 698)—(289, 663)—(273, 658)

Boundary of hole 2:

(261, 548)—(251, 559)—(237, 574)—(251, 588)
 (251, 588)—(265, 602)—(286, 600)—(302, 588)
 (302, 588)—(318, 576)—(318, 563)—(319, 553)
 (319, 553)—(320, 544)—(317, 522)—(306, 517)
 (306, 517)—(295, 511)—(271, 537)—(261, 548)

Figure 6.2: The three Bézier paths making up the boundary.

We cannot directly work with Bézier curves for the domain specification in Kardos. The domain has to be specified as a set of boundary nodes. Kardos then uses the software package Triangle [41] to produce a Delaunay triangulation based on these given nodes.

To produce a set of boundary nodes based on the Bézier curves we use the algorithm from Chapter 3 that takes into account the arc-length as well as the curvature along a curve and use an equidistribution principle for the assignment of the nodes. Taking 100 nodes and using the transformation (3.14) with $\alpha = 0.5$, the algorithm produces the node set given in the right picture of Figure 6.1. This set is used in both Kardos and AGTools.

Triangulation Kardos uses Triangle to produce an initial triangulation of the domain. It gives the possibility to specify the minimal angle that can occur inside a

triangle or the maximal area of certain triangles. Further, one can add nodes in the interior to force the presence of certain vertices in the triangulation. More details about the algorithm can be found in [41]. For the given domain and node set of Figure 6.1 the resulting triangulation, produced by setting the minimal angle to 34° , is shown in Figure 6.3. It consists of 384 points, 1038 edges and 653 triangles.

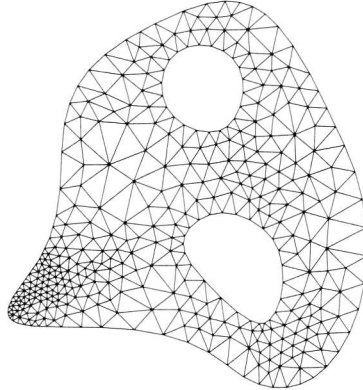


Figure 6.3: Triangulation determined by Kardos with minimal angle 34° .

Due to the very small support of the sources the discretization based on such a triangulation will in general 'not see' the sources. For example, in the left picture of Figure 6.4 a part of a triangulation is shown. The black dots are the actual points that are used in the FE discretization for calculation of integrals. The gray circle denotes the support of a source, which cannot be seen by the discretization. In such a situation, starting with a problem where the initial fields are zero, Kardos will never sense the sources and, as a result, the fields will stay zero. If a source support contains integration points, Kardos will start refinement routines to resolve the source profile properly.

Using the fact that Triangle can incorporate specific vertices, the source locations may be incorporated, as in the middle picture of Figure 6.4. Still, this will not solve the problem for very small supports. By adjusting also the cubature rules used by Kardos to evaluate integrals, integration points can be forced to coincide with source locations. See the right picture of Figure 6.4. A disadvantage is that in general this will result in less efficient cubature rules. For example, both cubature rules in Figure 6.4 are of 5-th order of accuracy, while the old rule uses 7 points and the new rule needs 10 points.

Problem 1: steady-state solutions

Our first test problem is the most simple one and it will serve for the comparison of the refinement capabilities of both methods. We will consider steady-state solutions

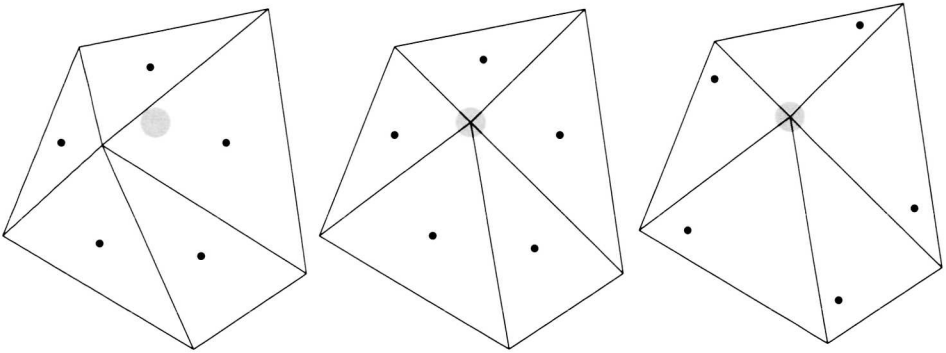


Figure 6.4: (Left) Independent triangulation and source location. (Middle) Triangulation based on source locations. (Right) Cubature rule that includes vertices of triangles.

of a field, thus no time integration issues are involved. Let us first define the problem.

Definition We take the domain of Figure 6.1 and set a single source at location $(207, 568)$ without internal behaviour modelled by the \mathbf{s} -part. We consider a single field ρ_1 with diffusion coefficient $d_1 = 1.0$, absorption coefficient $k_1 = 1.0e - 4$ and a constant production rate $\sigma_{11} = 1.0$. In this setting, the field solution approaches a steady-state solution for $t \rightarrow \infty$, which we will approximate using Kardos as well as AGTools. The system we consider is simply

$$\begin{aligned} 0 &= L_1 \rho_1 + \sigma_{11} T_{\mathbf{r}_1} S, & \text{on } \Omega, \\ 0 &= \mathbf{n} \cdot \nabla \rho_1, & \text{on } \partial\Omega, \\ \mathbf{0} &= \begin{pmatrix} 207 \\ 568 \end{pmatrix} - \mathbf{r}_2, \end{aligned} \tag{6.48}$$

with $L_1 = d_1 \Delta - k_1$. Although we deal here with a single field and a single state we persist in using the subscript notations to stay as closely as possible to the notation of Equations (6.3)–(6.6). The source profile function S is defined by

$$S(\mathbf{x}) = \begin{cases} \frac{2\pi}{(\pi^2 - 4)\ell^2} \cos^2\left(\frac{\pi}{2\ell} |\mathbf{x}|\right), & |\mathbf{x}| \leq \ell, \\ 0, & \text{otherwise,} \end{cases} \tag{6.49}$$

where we take the radius of the source support $\ell = 1$.

Solution by Kardos As Kardos only handles time-dependent, parabolic equations, we need to take some special actions to solve the steady-state equations. Because the steady-state equations are essentially of linear nature (ignoring for the moment the

nonlinearity incorporated through the coupling with the states), we can use the ROS1 time-integration scheme [26]. Using this in combination with a single fixed time step, while adjusting diffusion and absorption coefficients with respect to the step size, will give the solution.

To find the appropriate coefficients we consider first the linear ODE, $\dot{\mathbf{w}} = A\mathbf{w} + \mathbf{s}$, where A is an arbitrary linear operator that is invertible. The steady-state solution is equal to $\mathbf{w}_s = -A^{-1}\mathbf{s}$. Applying ROS1 to an equation that has an adjusted operator \tilde{A} , gives

$$\mathbf{w}_{n+1} = \mathbf{w}_n + (I - \gamma\tau\tilde{A})^{-1}\tau(\tilde{A}\mathbf{w}_n + \mathbf{s}).$$

Setting $n = 0$, $\mathbf{w}_0 = 0$, this gives $\mathbf{w}_1 = ((1/\tau)I - \gamma\tilde{A})^{-1}\mathbf{s}$, and also requiring $\mathbf{w}_s = \mathbf{w}_1$ leads to the condition

$$\tilde{A} = \frac{1}{\gamma}A + \frac{1}{\tau\gamma}I.$$

In the infinite-dimensional analog we have $A = d\Delta - kI$, resulting in

$$\tilde{A} = \frac{d}{\gamma}\Delta - \left(\frac{k}{\gamma} - \frac{1}{\tau\gamma}\right)I$$

or instead of d and k we have to work with $\tilde{d} = d/\gamma$ and $\tilde{k} = k/\gamma - 1/(\tau\gamma)$.

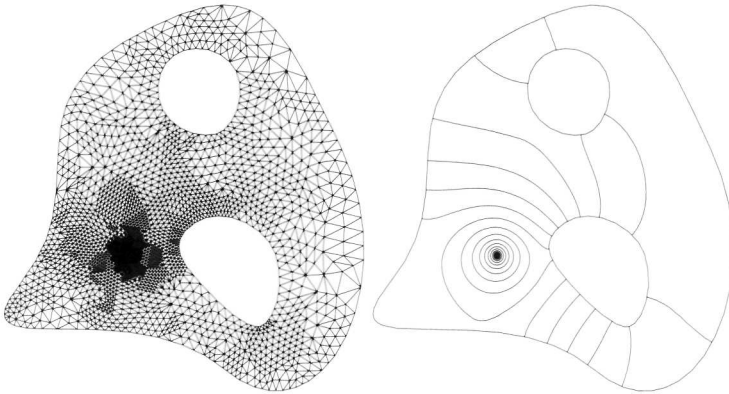


Figure 6.5: (Left) Triangulation used by Kardos for representing the approximation of solution. It consists of 2539 points, 7396 edges and 4856 triangles. (Right) Solution represented by 20 concentration level lines.

Now, for solving the system we start with the initial triangulation from Figure 6.3. Setting the relative tolerance to $1e-3$, Kardos will produce a triangulation that is shown in the left picture of Figure 6.5 and a solution of which a representation in level lines is shown in the right picture of the same figure.

To reach this solution Kardos uses 4 steps of refinement. It subsequently determines solutions based on triangulations with a number of points and an estimated

error, as given in the following table.

#Points:	384	496	679	1149	2539
Estimated error:	$1.373e-2$	$6.725e-3$	$2.940e-3$	$1.244e-3$	$5.030e-4$

Solution by AGTools For the solution by AGTools we use the same domain, given by the 100 boundary points. We use 3 rings of attraction at the boundary and 11 around the source location and 2539 nodes in total. In Figure 6.6 the resulting node set and the solution are displayed. Figure 6.7 shows in the left picture the underlying Voronoi diagram with the used refinement rings at the boundary. In the right picture the refinement rings around the source location are shown.

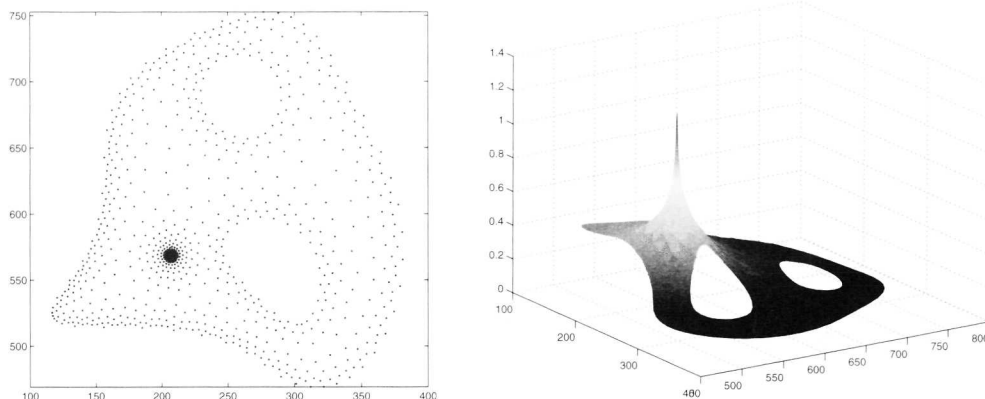


Figure 6.6: (Left) Node set used for the solution of Problem 1. (Right) The solution of Problem 1.

Comparison One of the main differences between the two approaches of Kardos and AGTools is that in Kardos the selection of a discretization and the actual solving of the equation are coupled, while in AGTools this is not so. To retrieve a solution from Kardos the required input is a certain error level. The program will then automatically generate a suitable discretization and solution by repeatedly solving the equation, estimating the error and adapting the discretization, until the solution falls below the prescribed error level.

Using such an approach, the resulting triangulation will have a strong refinement around the location of the sources and will be relatively coarse everywhere else. This is due to the fact that the diffusion processes tend to give smooth solutions, which possess their largest gradients near the locations of the sources. AGTools uses this information to produce, a priori, node sets that are suitable for the equations.

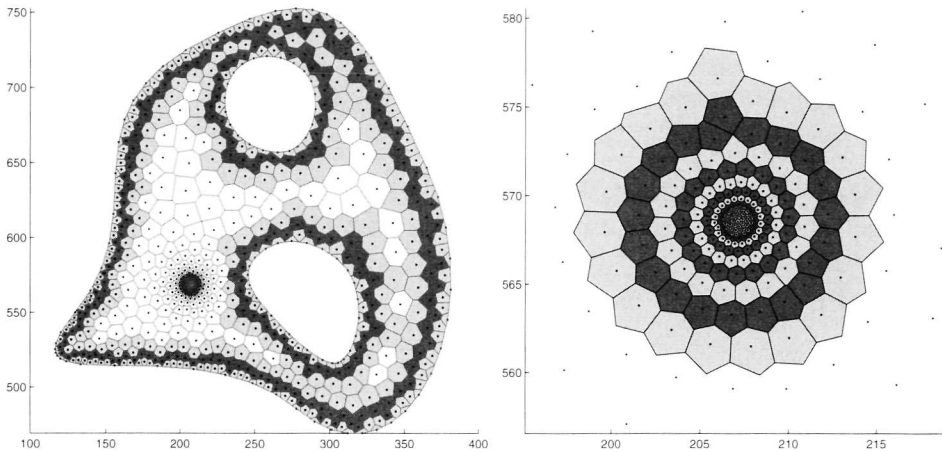


Figure 6.7: (Left) Rings of attraction at the boundaries. (Right) Rings of attraction around the source location.

A comparison of the triangulation used by Kardos, Figure 6.5, and the node set used by AGTools, Figure 6.7, shows that the refinement area of the node set seems to be more regular than the one of the triangulation. In the latter a kind of irregularity seems to be the result of the refinement technique used by Kardos, which splits triangles using ‘Red’ and ‘Green’ refinement, See [31].

For a precise comparison of the errors of both solutions an exact solution or an approximation with higher accuracy is needed. However, an easier, less accurate, way is to evaluate the AGTools solution in the vertices of the Kardos triangulation and compare the result with the Kardos solution. Such an approach shows that the pointwise difference of the two solutions away from the source location is about 0.1%, while the maximum, which is reached near the source location, is around 1%.

Problem 2: static sources

The second problem we consider is a problem where the sources still do not move through the environment, but where, in contrast to the first problem, the fields are dynamic and the sources possess extra behavior modelled by \mathbf{s}_1 and \mathbf{s}_2 . With this problem we want to examine how to implement a combination of field equations and ODEs (DAEs) in Kardos, which is developed to deal with a system of PDEs only.

Definition The system consists of two fields and three states. Two of the states represent the two non-moving sources, one for each field. The third state models an object that moves according to the gradients it senses. For both sources the excretion rate depends on the other field’s concentration at the source its location. The domain

and the diffusion and absorption coefficients are the same as in Problem 1. The equations are

$$\begin{aligned}
 \partial_t \rho_1 &= L_1 \rho_1 + \sigma_{11}(\mathbf{s}_1) T_{\mathbf{r}_1} S, & \text{on } \Omega, & \quad \rho_1(0, \mathbf{x}) = 0, \quad \forall \mathbf{x} \in \Omega \\
 \partial_t \rho_2 &= L_2 \rho_2 + \sigma_{22}(\mathbf{s}_2) T_{\mathbf{r}_2} S, & \text{on } \Omega, & \quad \rho_2(0, \mathbf{x}) = 0, \quad \forall \mathbf{x} \in \Omega \\
 0 &= \mathbf{n} \cdot \nabla \rho_1 = \mathbf{n} \cdot \nabla \rho_2, & \text{on } \partial\Omega, & \\
 \mathbf{0} &= \left(\begin{array}{c} \left(\frac{207}{568} \right) - \mathbf{r}_1 \\ 10^{-5} \\ \frac{10^{-5}}{10^{-5} + (\rho_2(\mathbf{r}_1))^4} - \mathbf{s}_1 \end{array} \right) & & (6.50) \\
 \mathbf{0} &= \left(\begin{array}{c} \left(\frac{350}{600} \right) - \mathbf{r}_2 \\ (\rho_2(\mathbf{r}_1))^4 \\ \frac{10^{-5}}{10^{-5} + (\rho_2(\mathbf{r}_1))^4} - \mathbf{s}_2 \end{array} \right) \\
 \dot{\mathbf{r}}_3 &= \lambda \frac{\nabla \rho_1(\mathbf{r}_3) + \nabla \rho_2(\mathbf{r}_3)}{\|\nabla \rho_1(\mathbf{r}_3) + \nabla \rho_2(\mathbf{r}_3)\|}, & \mathbf{r}_3(0) &= \begin{pmatrix} 257 \\ 618 \end{pmatrix},
 \end{aligned}$$

with $\sigma_{11}(\mathbf{s}_1) = \mathbf{s}_1$ and $\sigma_{22}(\mathbf{s}_2) = \mathbf{s}_2$. The $\mathbf{0}$ represents the three-dimensional vector with all components equal to 0. Note that \mathbf{s}_1 and \mathbf{s}_2 are scalars, despite being typeset in boldface.

Simulation with AGTools The results of a simulation with AGTools are shown in Figure 6.8 and Figure 6.9. This simulation ran for a time $T = 12\text{e}+4$ and used 400 time steps with the IMEX-midpoint scheme. The node set used for the discretization consists of 3261 nodes and is shown in the right picture of Figure 6.9. In this case the two fields share the same node set for simplicity. However, AGTools allows for the fields to have their own node sets, which would be more efficient here.

In the top panel of Figure 6.8 the evolution of the variables \mathbf{s}_1 and \mathbf{s}_2 is shown. These variables represent the excretion rates of the two fields. The middle panel shows the field values at the location of the sources. It can be clearly seen that these values are driven by the values of \mathbf{s}_1 and \mathbf{s}_2 , because they follow a similar pattern. In turn, the values of \mathbf{s}_1 and \mathbf{s}_2 are driven by the values of $\rho_2(\mathbf{r}_1)$ and $\rho_1(\mathbf{r}_2)$, respectively, the latter being shown in the bottom panel of the figure.

An intuitive description of the oscillation goes like this. At $t = 0$ both fields are zero. Therefore source 1 its excretion rate equals 1, while the excretion rate of source 2 is zero. As a consequence only field ρ_1 starts to develop.

1. (Around $t = 1.0\text{e}+4$) The rising value $\rho_1(\mathbf{r}_2)$ triggers source 2, resulting in a rising \mathbf{s}_2 . As a consequence field ρ_2 starts to develop.
2. (Around $t = 2.5\text{e}+4$) A rising field value $\rho_2(\mathbf{r}_1)$ will inhibit the production for field ρ_1 . Therefore field ρ_1 starts to decay, because of the absorption.

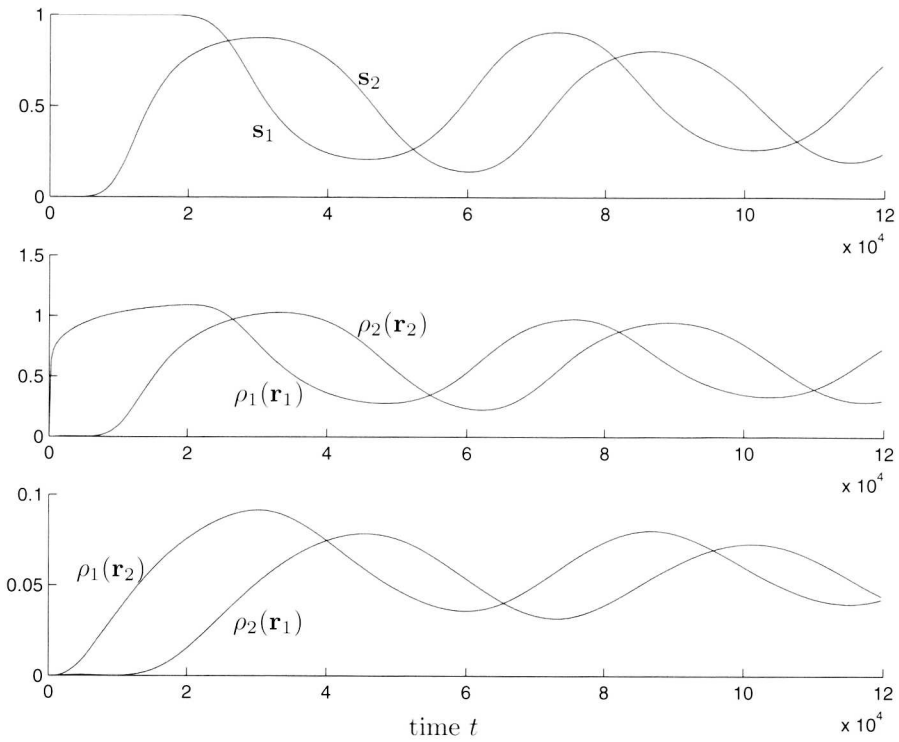


Figure 6.8: Problem 2: State variables and field values against time.

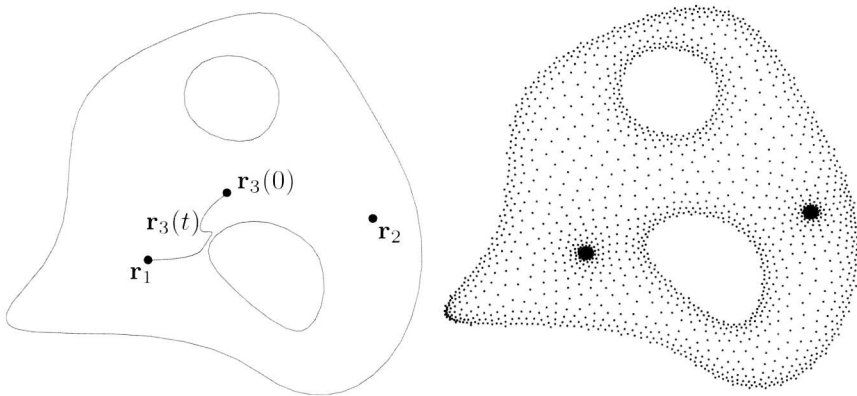


Figure 6.9: Problem 2: (Left) Locations of the initial states (dots) and the path of \mathbf{r}_3 . (Right) Used node set for the discretization of the fields.

3. (Around $t = 4.0e+4$) A declining field value $\rho_1(\mathbf{r}_2)$ will inhibit the production for field ρ_2 . Therefore field ρ_2 starts to decay. because of the absorption.
4. (Around $t = 5.0e+4$) A declining field value $\rho_2(\mathbf{r}_1)$ will trigger the production for field ρ_1 . Therefore field ρ_1 starts to develop again.

From here, the process continues at step 1 again, resulting in an oscillating pattern. We didn't consider the stability of this pattern. It might be very well the case that the oscillations damp out or that they tend to increase over time.

During this process \mathbf{r}_3 moves with constant speed through the domain in the direction of a linear combination of the gradients of the two fields. As a result of the oscillating fields the path of \mathbf{r}_3 displays two sharp turns, as can be clearly seen in Figure 6.9.

Implementation in Kardos Kardos is designed for systems of nonlinear parabolic equations. The systems that we consider in this chapter include besides a number of field equations also a number of ODEs and/or algebraic equations. In Section 6.5 it was shown how these systems can be solved using Rosenbrock time integration methods, as are used by Kardos. It turns out that adjusting Kardos for making it possible to solve these hybrid systems in general is a very complex and time consuming task. This seems not to be the best direction to take, because the underlying idea of trying to use Kardos for system (6.2)–(6.6), is that it might be, as an existing software package, easily extensible as to incorporate the simulations of our hybrid systems.

A far more simpler option is to restrict our use of Kardos to using only the implemented W-methods. Such methods do allow for replacement of the exact Jacobian matrix by an approximation of it, while retaining the order of accuracy of the method. The earlier given example of a Rosenbrock method, ROS2, is a representative of this class of W-methods. If we use this method and an approximation of the Jacobian in which we only incorporate the stiff parts, i.e., those parts that are related to the diffusion operators, then we can implement our systems with a relatively little amount of work.

To show the basic principles behind this approach, consider a simple system consisting of two components $\boldsymbol{\rho}$, \mathbf{u} , of which the dynamics are determined by

$$\dot{\boldsymbol{\rho}} = f_1(\boldsymbol{\rho}, \mathbf{u}), \quad \dot{\mathbf{u}} = f_2(\boldsymbol{\rho}, \mathbf{u}), \quad (6.51)$$

where the Jacobian operator $D_{\boldsymbol{\rho}}f_1$ gives rise to stiffness and the other Jacobian operators $D_{\mathbf{u}}f_1$, $D_{\boldsymbol{\rho}}f_2$ and $D_{\mathbf{u}}f_2$ do not. When using a W-method, replacement of the three non-stiff Jacobian operators, by zero-operators, will retain the order of accuracy and will in general not harm so much the stability of the method. We will now compare the application of this approach, using ROS2, to this system as well as to the first equation of this system only (containing only the $\boldsymbol{\rho}$ component).

We start with the system consisting of the $\boldsymbol{\rho}$ -component only. The variable \mathbf{u} is

then considered as a parameter of the system. Application of ROS2 yields

$$\begin{aligned} \left(\frac{1}{\gamma\tau} - [D_{\rho}f_1(\rho_n, \mathbf{u})]\right) \boldsymbol{\eta} &= f_1(\rho_n, \mathbf{u}) \\ \left(\frac{1}{\gamma\tau} - [D_{\rho}f_1(\rho_n, \mathbf{u})]\right) \boldsymbol{\xi} &= f_1\left(\rho_n + \frac{1}{\gamma}\boldsymbol{\eta}, \mathbf{u}\right) - \frac{2}{\gamma\tau}\boldsymbol{\eta} \\ \rho_{n+1} &= \rho_n + \frac{3}{2\gamma}\boldsymbol{\eta} + \frac{1}{2\gamma}\boldsymbol{\xi}. \end{aligned} \tag{6.52}$$

Application of ROS2 to the complete system, with the use of the approximation of the Jacobian, yields

$$\begin{aligned} \left(\frac{1}{\gamma\tau} - [D_{\rho}f_1(\rho_n, \mathbf{u}_n)]\right) \boldsymbol{\eta} &= f_1(\rho_n, \mathbf{u}_n) \\ \frac{1}{\gamma\tau}\mathbf{v} &= f_2(\rho_n, \mathbf{u}_n) \\ \left(\frac{1}{\gamma\tau} - [D_{\rho}f_1(\rho_n, \mathbf{u}_n)]\right) \boldsymbol{\xi} &= f_1\left(\rho_n + \frac{1}{\gamma}\boldsymbol{\eta}, \mathbf{u}_n + \frac{1}{\gamma}\mathbf{v}\right) - \frac{2}{\gamma\tau}\boldsymbol{\eta} \\ \frac{1}{\gamma\tau}\mathbf{w} &= f_2\left(\rho_n + \frac{1}{\gamma}\boldsymbol{\eta}, \mathbf{u}_n + \frac{1}{\gamma}\mathbf{v}\right) - \frac{2}{\gamma\tau}\mathbf{v} \\ \rho_{n+1} &= \rho_n + \frac{3}{2\gamma}\boldsymbol{\eta} + \frac{1}{2\gamma}\boldsymbol{\xi} \\ \mathbf{u}_{n+1} &= \mathbf{u}_n + \frac{3}{2\gamma}\mathbf{v} + \frac{1}{2\gamma}\mathbf{w}. \end{aligned} \tag{6.53}$$

Comparing the systems (6.52) and (6.53), we see that the application of ROS2 to a stiff system (6.52) can be extended to the application of ROS2 to a larger system (6.53), by inserting a number of actions between the stages. In the following scheme the actions performed by Kardos and the actions that are to be inserted are displayed.

<p style="text-align: center;">Kardos</p> $J = [D_{\rho}f_1(\rho_n, \mathbf{u})]$ $\left(\frac{1}{\gamma\tau} - J\right) \boldsymbol{\eta} = f_1(\rho_n, \mathbf{u})$ $\left(\frac{1}{\gamma\tau} - J\right) \boldsymbol{\xi} = f_1\left(\rho_n + \frac{1}{\gamma}\boldsymbol{\eta}, \mathbf{u}\right) - \frac{2}{\gamma\tau}\boldsymbol{\eta}$ $\rho_{n+1} = \rho_n + \frac{3}{2\gamma}\boldsymbol{\eta} + \frac{1}{2\gamma}\boldsymbol{\xi}$	<p style="text-align: center;">extra implemented</p> $\mathbf{u} = \mathbf{u}_n$ $\frac{1}{\gamma\tau}\mathbf{v} = f_2(\rho_n, \mathbf{u})$ $\mathbf{u} = \mathbf{u}_n + \frac{1}{\gamma\tau}\mathbf{v}$ $\frac{1}{\gamma\tau}\mathbf{w} = f_2\left(\rho_n + \frac{1}{\gamma}\boldsymbol{\eta}, \mathbf{u}\right) - \frac{2}{\gamma\tau}\mathbf{v}$ $\mathbf{u}_{n+1} = \mathbf{u}_n + \frac{3}{2\gamma}\mathbf{v} + \frac{1}{2\gamma}\mathbf{w}.$
--	--

Implementation of this approach can be done by using the built-in event mechanism of Kardos. It requires the definition of new events before and between the different stages of a time step. They trigger the calls for the suitable subroutines for performing the actions at the right side of (6.54).

Simulation with Kardos Using the technique described above, Kardos was used to approximate the solution to system (6.50). With error tolerances for time set to 0.01 and for space to 0.001, the resulting solutions of the states and the fields evaluated in the states' locations is within 1% of the AGTools solutions shown in Figure 6.8 and Figure 6.9.

An initial triangulation, similar to the one of Figure 6.3, was used, which has 402 vertices and was forced to contain nodes with locations \mathbf{r}_1 and \mathbf{r}_2 . During the simulation Kardos uses a refined triangulation, where refinement takes place around these two points. The maximum number of vertices reached during the triangulation is 4904. The triangulation at the end of the simulation is shown in Figure 6.10, which has 1540 vertices.

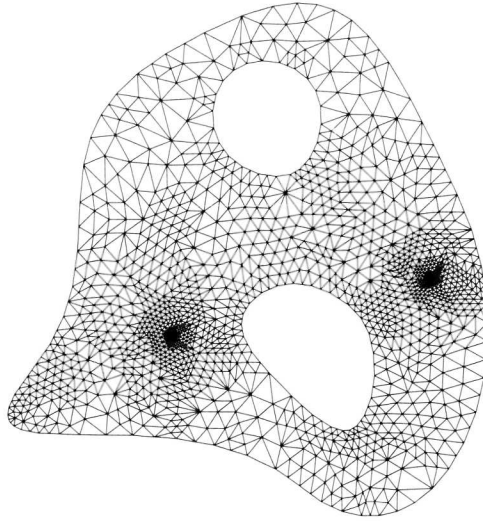


Figure 6.10: The triangulation used by Kardos at time $T = 12e+4$.

Figure 6.11 displays the time step size and the number of vertices in the triangulation against the time step index. The number of time steps equals 175, where the time step size is very small in the beginning of the simulation as the fields have to be developed and time derivatives are relatively large. Especially in the beginning of the simulation several time steps are rejected and smaller time steps are taken instead. For example, while the initial time step size is set to 1000, the first accepted time step, after three reductions, is equal to 0.182. For every reduction 4 to 6 refinement iterations are carried out, each with their own linear systems to be solved. After the initial phase, time step reductions do occur in smaller numbers, while the time step size increases gradually to around 1300. At time level 80, at more or less half of the computational work, only 10% of the total time interval is reached.

We want to conclude the treatment of this problem by making a few remarks.

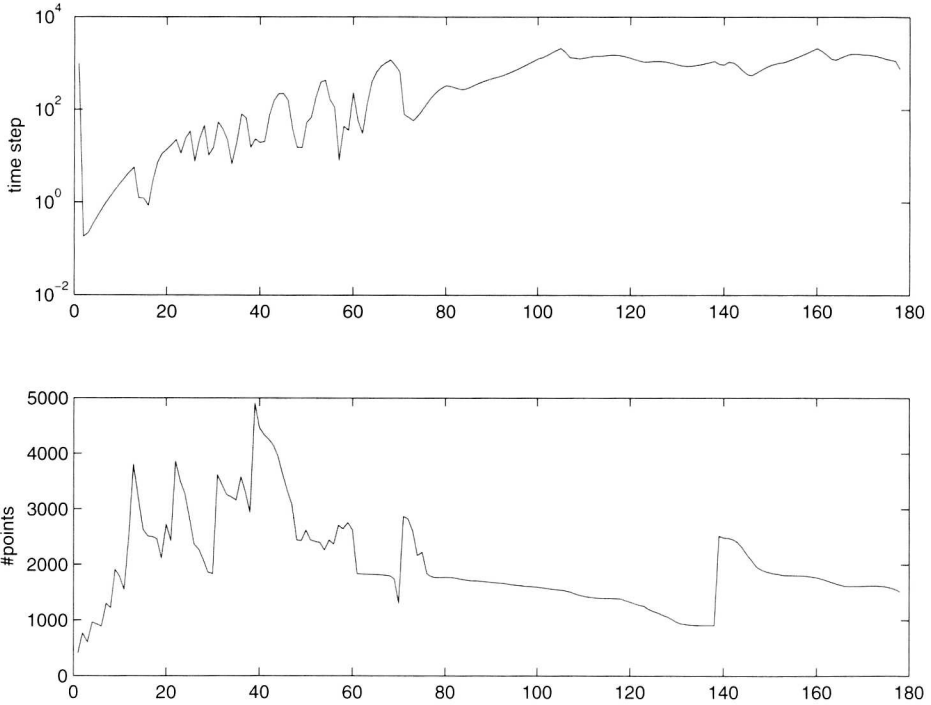


Figure 6.11: Problem 2: The time step size (top) and the number of vertices in the triangulation (bottom) against the time level used by Kardos.

First, due to the fact that the handling of the ODEs is ‘hacked’ into Kardos by means of the event mechanism, the adaptivity routines completely ignore the ODEs. Therefore, sharp gradients in the dynamics of the states will not enforce smaller time steps. Second, the solution is not flexible with respect to the implementation of different ODEs. As a result, changing the dimensions as well as changing the dynamics of the states can be very error-prone.

Problem 3: moving sources

Our final test problem concerns a system where we have two fields, both produced by their own sources that are moving through the domain. For both sources applies that their movement is determined by the field that they do not produce for.

Definition The system is defined by

$$\begin{aligned}
 \partial_t \rho_1 &= L_1 \rho_1 + \sigma_{11} T_{\mathbf{r}_1} S, & \text{on } \Omega, & & \rho_1(0, \mathbf{x}) &= 0, & \forall \mathbf{x} \in \Omega \\
 \partial_t \rho_2 &= L_2 \rho_2 + \sigma_{22} T_{\mathbf{r}_2} S, & \text{on } \Omega, & & \rho_2(0, \mathbf{x}) &= 0, & \forall \mathbf{x} \in \Omega \\
 0 &= \mathbf{n} \cdot \nabla \rho_1 = \mathbf{n} \cdot \nabla \rho_2, & \text{on } \partial\Omega, & & & & \\
 \mathbf{r}_1 &= \lambda \frac{\nabla \rho_2(\mathbf{r}_1)}{\|\nabla \rho_2(\mathbf{r}_1)\|} & & & \mathbf{r}_1(0) &= \begin{pmatrix} 207 \\ 568 \end{pmatrix} & (6.55) \\
 \mathbf{r}_2 &= \lambda \frac{\nabla \rho_1(\mathbf{r}_2)}{\|\nabla \rho_1(\mathbf{r}_2)\|} & & & \mathbf{r}_2(0) &= \begin{pmatrix} 350 \\ 600 \end{pmatrix}
 \end{aligned}$$

with $L_1 = L_2 = d\Delta - k$, $\sigma_{11} = 1$ and $\sigma_{12} = 1$. The domain Ω , diffusion coefficient d and the absorption coefficient k are taken the same as in Problem 1 and 2 and $\lambda = 8.0\text{e-}4$. In the resulting dynamics the two sources move toward each other.

Simulation with AGTools Simulation of a problem like Problem 3 is computationally more expensive because of the moving sources. Every time step a new node set has to be generated and by using interpolation the solution has to be transferred to it.

Figure 6.12 shows the result of a simulation with AGTools for $t \in [0, 1\text{e}+5]$, where 300 time steps with the IMEX-midpoint scheme were used. The left picture of the figure displays the paths of \mathbf{r}_1 and \mathbf{r}_2 . Clearly can be seen that the sources move toward each other, while growing around the hole in the domain. In the right picture of the figure the used node set at $t = 7.4\text{e}+4$ is shown.

Implementation in Kardos The implementation in Kardos of this problem was done in a way similar to Problem 2. We used the same simulation parameters as there and the solution gave paths that were close to the paths of Figure 6.12. The maximal euclidian distance between the two solutions over the whole integration interval was 9.3, in the units of Figure 6.1.

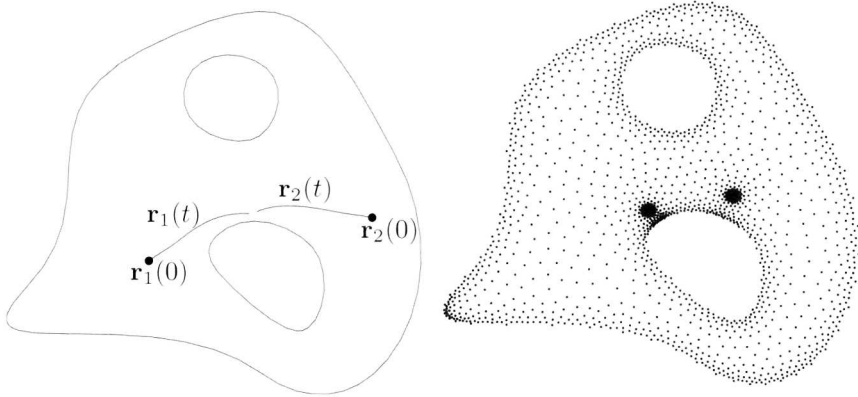


Figure 6.12: Problem 3: (Left) Locations of the initial states (dots) and the paths of \mathbf{r}_1 and \mathbf{r}_2 . (Right) Used node set for the discretization of the fields halfway during the simulation.

In Figure 6.13 the time step size and the number of vertices in the triangulation are shown against the time level. One of the differences with Problem 2 is that the sources are moving here. Due to these moving sources the maximal number of vertices in the used triangulations is here almost twice as high as in Problem 2. The maximal time step size however, is larger as in Problem 2. This can be probably explained by the fact that the fields are not produced by constant sources but that their excretion rates depend on field values.

We want to conclude Problem 3 by mentioning that we did not carry out a comparison between the use of AGTools and Kardos with respect to efficiency. This we did not do because of the great differences between the two approaches and because of the advanced error control routines present in Kardos, while not available in AGTools.

6.7 Summary

This chapter concerns the numerical approximation of the behavior of the dynamical systems that are present in the AGTools framework. These systems are composed of parabolic and elliptic PDEs that are strongly coupled to a system of ODEs and algebraic equations. A presentation of the used numerical methods in AGTools is combined with a discussion of the possibility of using an existing software package for the solution of the systems at hand, that is designed for solving systems of PDEs. As a representative was taken the software package Kardos.

The chapter starts with an overview of the AGTools framework; giving both an abstract formulation of the equation systems, as well as a presentation of the used numerical methods for approximating these systems numerically. This is followed by

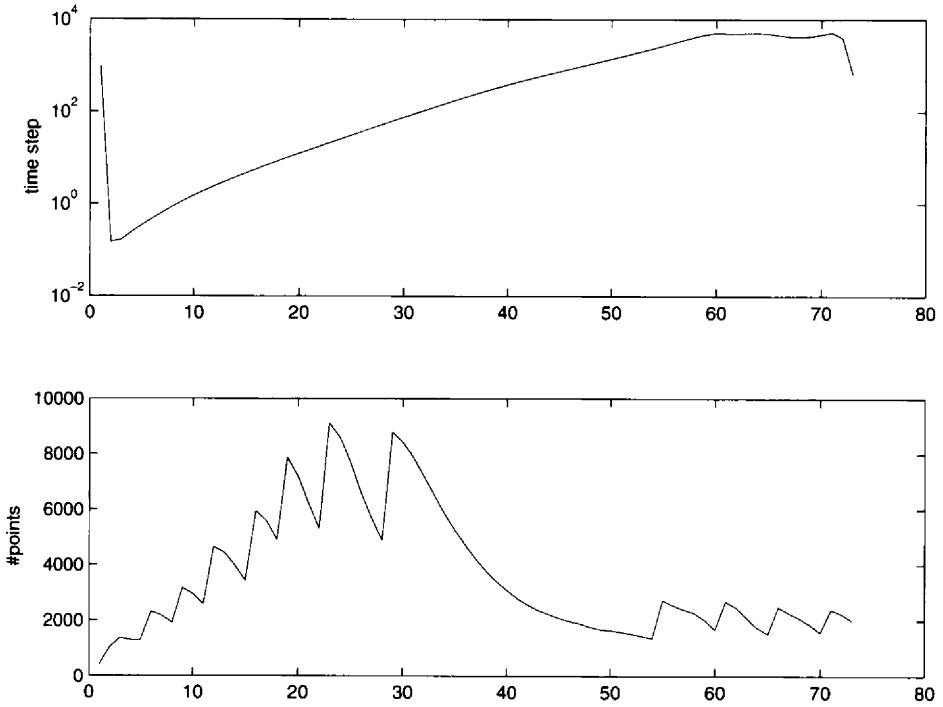


Figure 6.13: Problem 3: The time step size (top) and the number of vertices in the triangulation (bottom) against the time level used by Kardos.

a description of Kardos and its main features.

In Section 6.5 the application of Rosenbrock time integration methods to the framework's equation systems is discussed. Due to the nature of these systems this is not straightforward. It is shown that the emerging linear stage systems in the Rosenbrock methods contain point evaluation operators. A method for solving such linear systems is described.

As the use of general Rosenbrock methods for the systems at hand is rather complex and because of the way that Kardos builds its discretizations for solving PDEs, a general adjustment of Kardos for incorporating these systems seems not practical. A simpler alternative is to restrict our use of Kardos to the use of W-methods. These time integration methods allow for simplifying the solution process by approximating the Jacobian operators. It is shown that the use of the built-in event mechanisms in Kardos can be used to implement the solution of certain systems in a more ad-hoc approach.

Kardos uses an initial coarse triangulation that is adaptively refined on the basis of a posteriori error estimators. It is shown that because of the small supports of the sources this initial triangulation has to be based on the initial locations of the sources and that (at least for the first time step) a special cubature rule is used for the numerical approximation of integrals needed for the discretizations of the finite element method.

Three example problems are discussed, together with the issues that are encountered when implementing the problems in Kardos. An advantage of using Kardos over AGTools is that it has a sophisticated adaptation scheme in space and in time and can produce solutions within a predetermined error range. On the other hand, AGTools makes better use of the a priori knowledge that the refinement areas are around the source locations. This leads to an efficient discretization of the PDEs and no iterative solving and error estimation is needed.

A direct comparison with respect to efficiency was not performed due to the great differences in the two approaches. Concerning the question whether Kardos might be used for the equation systems of the framework, we can say that a structural adjustment to incorporate such models into Kardos is very difficult and requires a lot of programming activity. An easy, more ad-hoc, approach is possible, but requires still some serious amount of error-prone coding. Also, if using this latter approach, error control routines do not take into account the ODE dynamics.

Acknowledgement The author wants to thank Prof. Dr. J. Lang of Darmstadt University of Technology for his help on using the software package Kardos. He delivered valuable background information on the architecture of the software and for the implementation of described problems.

Summary

This thesis is a treatment on the simulation of growing nerve cells during the development of the nervous system. While there is a basic understanding of how the connection forming axons find their target neurons by means of chemotaxis, the underlying precise mechanisms are far from clear. To complement the experimental research by the use of computational models, a framework for modelling these growth processes mathematically is presented, together with numerical methods for use in simulation. A basic assumption is that it should be possible to set up a model and carry out the simulations without extensive programming activity.

The framework consists of a set of finite-dimensional vectors and a two-dimensional domain with fields defined on it. The vectors are referred to as states, and can be interpreted as objects that move through the domain. The fields are subject to diffusion, absorption, and excretion processes, the latter being the result of highly localized source terms that are situated at the locations of the states.

As a start an examination of a representative equation system is given. It serves to get acquainted with some specific features of the systems at hand, in particular the occurrence of a feature called self-interaction. When present, the use of point sources is prohibited and the resulting dynamics are very sensitive to the width of the source supports. Also, the use of quasi steady-state approximations for fields with fast diffusion may cause large changes in the dynamics.

What follows is an exposition on the domains of models in the framework can be specified. To make the definition of complex domains easy, the boundaries of the domains are specified by means of Bézier paths. However, the numerical methods for solving the field equations work with boundaries that are specified as sets of points connected by straight lines. It is shown how point sets on the Bézier paths that represent the boundaries well can be selected, making use of the arc-length and the curvature of the paths.

After that, the spatial discretization of the field equations is discussed. Two properties of the systems at hand are of importance. First, the sources for the diffusion equations are made up of highly localized source terms of which the locations may change over time. Second, the domains can be rather complex, making the use of regular grids difficult. To address these two issues, an unstructured discretization is presented in which the fields are discretized on the basis of arbitrary node sets. This approach is flexible with respect to domain geometries and allows for easy re-

finement and adaptation needed for the moving sources. It uses Voronoi diagrams for discretizing the operators as well as for choosing the underlying set of nodes.

The thesis continues with a description of the modelling framework. After an introduction to its basic concepts the underlying mathematical model is presented. Both the field and state dynamics can occur in two forms that are referred to as dynamic and static, respectively. Dynamic fields give rise to parabolic partial differential equations (PDEs), while static fields produce elliptic PDEs. Likewise, dynamic states give ordinary differential equations (ODEs) and static states give algebraic equations. In total, the model can be a system of coupled parabolic PDEs, elliptic PDEs, ODEs, and algebraic equations.

A description is given of different regimes that are based on the combination of types of equations that make up the models. As a result, these regimes are connected to the numerical methods needed for the simulations. Included is a presentation of example models that show how the framework can be used to test different mechanisms. As the focus lies on the modelling aspect, the discussed models all fall in a regime for which the simplest set of numerical methods is needed. Those models merely encompass static fields that are in their steady-states.

The final chapter concerns the simulation of models in which all types of equations are present. It discusses the numerical methods used by AGTools; a set of Matlab scripts written especially for use with the framework. Its most important characteristics are the use of a Runge-Kutta IMEX-method for time integration and the earlier presented spatial discretization technique based on Voronoi diagrams. In addition, the use of a standard finite element package is considered, where the package Kardos was taken as a representative. It is examined how Kardos can be used for the simulation of the framework models. In particular, the use of Rosenbrock time integration for the models is discussed. For three example models the use of AGTools and the use of Kardos are compared.

Samenvatting

Dit proefschrift handelt over de simulatie van groeiende zenuwcellen tijdens de ontwikkeling van het zenuwstelsel. Het betreft het modelleren van groeiende axonen en de mechanismen die ervoor zorgen dat ze groeien in de richting van de neuronen waarmee ze verbinding maken. Ondanks dat de principes die hieraan ten grondslag liggen duidelijk zijn, zijn de onderliggende, precieze mechanismen onbekend. Om het experimentele onderzoek te complementeren met het gebruik van computationele modellen, is een framework ontwikkeld voor het wiskundig modelleren van deze groeiprocessen. Dit framework wordt gepresenteerd samen met numerieke methoden die gebruikt worden voor de simulaties. Een uitgangspunt bij de ontwikkeling hiervan was dat het mogelijk moet zijn om een model te implementeren en simulaties uit te voeren zonder daar uitvoerig programmeerwerk voor te verrichten.

Het framework bestaat uit een collectie van eindig-dimensionale vectoren en een twee-dimensionaal domein met daarop velden gedefinieerd. Deze vectoren worden 'states' genoemd en kunnen worden beschouwd als objecten die door het domein bewegen. De velden zijn onderhevig aan diffusie, absorptie en uitscheiding door bronnen. Deze bronnen hebben een kleine afmeting en zijn gesitueerd op de locaties van de states.

Het proefschrift begint met de beschrijving van een representatief systeem van vergelijkingen. Hiermee wordt een introductie gegeven tot een paar specifieke eigenschappen van de systemen die in het framework voorkomen. In het bijzonder wordt ingegaan op een specifieke eigenschap. Als deze eigenschap, genoemd self-interaction, aanwezig is in het systeem, dan is het gebruik van puntbronnen niet mogelijk en de dynamica van het systeem zeer gevoelig voor de afstand van de bronnen. Ook het gebruik van quasi steady-state benaderingen voor velden met een snelle diffusie kan tot een dynamica leiden die zeer verschilt van de originele dynamica.

Wat volgt is een uiteenzetting van de domeinspecificatie voor modellen in het framework. Om een makkelijke definitie van complexe domeinen mogelijk te maken, worden de randen van de domeinen vastgelegd door middel van Bézierpaden. Echter, de numerieke methoden voor het oplossen van de veldvergelijkingen werken met randen die gespecificeerd zijn als verzamelingen van punten, onderling verbonden door middel van rechte lijnen. Er wordt getoond hoe punten op de Bézierpaden kunnen worden geselecteerd, zodanig dat de randen goed gerepresenteerd zijn. Hierbij is gebruik gemaakt van de 'arc-length' en de 'curvature' van de Bézierpaden.

Het daarop volgende hoofdstuk gaat over de ruimtelijke discretisatie van de veldvergelijkingen. Twee eigenschappen van de systemen zijn belangrijk. Ten eerste worden de bronnen van de diffusievergelijkingen gevormd door brontermen die allen een zeer klein support hebben en kunnen bewegen door het domein. Ten tweede kunnen de domeinen een complexe geometrie hebben, waardoor het gebruik van reguliere grids moeilijk is. Met deze twee eigenschappen in het achterhoofd is een discretisatie ontwikkeld, waarbij de velden worden gediscrètiseerd op basis van een verzameling van willekeurige punten in het domein. Deze benadering is flexibel met betrekking tot de vorm van het domein en staat daarnaast een makkelijke vorm van verfijning en adaptatie toe, nodig vanwege de bewegende bronnen. Voor de discretisatie van de diffusieoperatoren en voor het kiezen van de onderliggende puntverzamelingen wordt gebruikt gemaakt van Voronoi diagrammen.

Het proefschrift gaat verder met een uitgebreide beschrijving van het framework. Na een inleiding tot de gebruikte concepten wordt het onderliggende wiskundige model besproken. Zowel de velden als de states kunnen voorkomen in twee vormen, te weten dynamisch en statisch. Dynamische velden resulteren in parabolische partiële differentiaalvergelijkingen (pdv's), terwijl statische velden leiden tot elliptische pdv's. Op dezelfde wijze geven dynamische states gewone differentiaalvergelijkingen (gdv's) en statische states, algebraïsche vergelijkingen. Het totale model bestaat dan uit een systeem van gekoppelde parabolische en elliptische pdv's, gdv's en algebraïsche vergelijkingen.

Er wordt tevens een onderscheid gemaakt in regimes van modellen, waarbij een regime bestaat uit modellen waarin overeenkomstige typen vergelijkingen voorkomen. Hierdoor heeft ieder regime zijn eigen numerieke methoden, nodig voor het uitvoeren van de simulaties van de modellen in het desbetreffende regime. Inbegrepen is een presentatie van voorbeeldmodellen die laat zien hoe het framework kan worden gebruikt voor het testen van verschillende mechanismen. Omdat de focus ligt op het modelleeraspect, vallen deze modellen in het regime met de minst uitgebreide numerieke methoden. Deze modellen bevatten alleen statische velden die tevens in hun steady-state verkeren.

Het laatste hoofdstuk gaat over de simulatie van de regimes waarin alle soorten vergelijkingen aanwezig zijn. Het behandelt de numerieke methoden die worden gebruikt bij AGTools, een verzameling van Matlab scripts speciaal geschreven voor gebruik van het framework. De belangrijkste karakteristieken zijn de toepassing van een Runge-Kutta IMEX methode voor tijdsintegratie en de eerder besproken ruimtelijke discretisatiemethode gebaseerd op Voronoi diagrammen. Tevens wordt het gebruik van een standaard eindige-elementen pakket overwogen, waarbij Kardos als representatief pakket is gebruikt voor de simulatie van de modellen in het framework. In het bijzonder wordt het gebruik van de Rosenbrock tijdsintegratiemethoden besproken. Voor drie voorbeeldmodellen is het gebruik van AGTools en het gebruik van Kardos vergeleken.

Bibliography

- [1] M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions*. Dover Publications, New York, 1964.
- [2] K. Atkinson and W. Han. *Theoretical Numerical Analysis*. Number 39 in Texts in Applied Mathematics. Springer-Verlag, New York, 2001.
- [3] Richard H. Bartels, John C. Beatty, and Brian A. Barsky. *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, Palo Alto, CA, 1987. With forewords by Pierre Bézier and A. Robin Forrest.
- [4] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering, Special Issue on Meshless Methods*, 139:3-47, 1996.
- [5] T. Belytschko, Y.Y. Lu, and L. Gu. Element-free galerkin methods. *International Journal for Numerical Methods in Engineering*, 37:229-256, 1994.
- [6] Y.J. Choi and S.J. Kim. Node generation scheme for the meshfree method by Voronoi diagram and weighted bubble packing. Fifth U.S. National Congress on Computational Mechanics. Boulder, CO, 1999.
- [7] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry*. Springer-Verlag, 2nd edition, 2000.
- [8] B. J. Dickson. Molecular mechanisms of axon guidance. *Science*, 298:1959-1964, 2002.
- [9] J. Dodd and T.M. Jessell. Axon guidance and the patterning of neuronal projections in vertebrates. *Science*, 242:692-699, 1988.
- [10] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637-676, 1999.
- [11] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153-174, 1987.

- [12] T.P. Fries and H.G. Matthies. Classification and overview of meshfree methods. Informatikbericht 2003-03, Institute of Scientific Computing, Technical University Braunschweig, Brunswick, Germany, 2003.
- [13] R.M. Gaze. The representation of the retina on the optic lobe of the frog. *Quart. J. Exp. Physiol.*, 43:209–224, 1958.
- [14] D. Gilbarg and N.S. Trudinger. *Elliptic Partial Differential Equations of Second Order*. Springer Verlag, 2001.
- [15] G.S. Gipson. Use of the residue theorem in locating points within an arbitrary multiply-connected region. *Adv. Eng. Software*, 8(2):73–80, 1986.
- [16] G.H. Golub and C.F. van Loan. *Matrix Computations*. The John Hopkins University Press, 3rd edition, 1996.
- [17] G.J. Goodhill. Diffusion in axon guidance. *Eur. J. Neurosci.*, 9:1414–1421, 1997.
- [18] G.J. Goodhill. A mathematical model of axon guidance by diffusible factors. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages 159–165. MIT Press, 1998.
- [19] K.E. Gustafson. *Introduction to Partial Differential Equations and Hilbert Space Methods*. John Wiley & Sons, 2nd edition, 1987.
- [20] E. Hairer and G. Wanner. *Solving ordinary differential equations II: Stiff and differential-algebraic problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1991.
- [21] H.G.E. Hentschel and A. van Ooyen. Models of axon guidance and bundling during development. *Proc. R. Soc. Lond. B.*, 266:2231–2238, 1999.
- [22] M.L. Hines and N.T. Carnevale. The neuron simulation environment. *Neural Comput.*, 9:1179–1209, 1997.
- [23] H. Honda. Topographic mapping in the retinotectal projection by means of complementary ligand and receptor gradients: a computer simulation study. *J. Theor. Biol.*, 192:235–246, 1998.
- [24] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.
- [25] A.B. Huber, A.L. Kolodkin, D.D. Ginty, and J.-F. Cloutier. Signaling at the growth cone: ligand-receptor complexes and the control of axon growth and guidance. *Annu. Rev. Neurosci.*, 26:509–563, 2003.
- [26] W. Hundsdorfer and J.G. Verwer. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*. Springer, 2003.

- [27] S.R. Idelsohn, E. Oñate, F. Del Pin, and N. Calvo. The meshless finite element method. *Int. J. Numer. Methods Eng.*, 2001.
- [28] J.K. Krottje. On the dynamics of a mixed parabolic-gradient system. *Communications on Pure and Applied Analysis*, 2(4):521–537, December 2003.
- [29] J.K. Krottje. A variational meshfree method for solving time-discrete diffusion equations. Technical Report MAS-E0319, Centrum voor Wiskunde en Informatica, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, December 2003.
- [30] J.K. Krottje and A. van Ooyen. A mathematical framework for modelling axon guidance. *Bulletin for Mathematical Biology*, Submitted, May 2005.
- [31] J. Lang. *Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems*. Number 16 in Lecture Notes in Computational Science and Engineering. Springer, 2001.
- [32] B. Lastdrager. Numerical solution of mixed gradient-diffusion equations modelling axon growth. Technical Report MAS-R0203, Centrum voor Wiskunde en Informatica, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, January 2002.
- [33] Shaofan Li and Wing Kam Liu. Meshfree and particle methods and their applications. *Applied Mechanics Review*, 55:1–34, 2002.
- [34] X.-Y. Li, S.-H. Teng, and A. Üngör. Point placement for meshless methods using sphere packing and advancing front methods. Technical report, University of Illinois at Urbana-Champaign, 2000.
- [35] G.-L. Ming, S.T. Wong, J. Henley, X.-B. Yuan, H.-J. Song, N.C. Spitzer, and M.-M. Poo. Adaptation in the chemotactic guidance of nerve growth cones. *Nature*, 417:411–418, 2002.
- [36] B. Nayroles, G. Touzet, and P. Villon. Generalizing the finite element method: Diffuse approximation and diffuse elements. *Comp. Mech.*, 10:307–318, 1992.
- [37] D.D.M. O’Leary and D.G. Wilkinson. Eph receptors and ephrins in neural development. *Current Opinion Neurobiology*, 9:55–73, 1999.
- [38] V. Rehder and S. B. Kater. Filopodia on neuronal growth cones: multi-functional structures with sensory and motor capabilities. *Sem. Neurosci.*, 8:81–88, 1996.
- [39] K. Rektorys. *The method of discretization in time and partial differential equations*. volume 4 of *Mathematics and Its Applications (East European Series)*. D. Reidel Publishing Co., Dordrecht, 1982.
- [40] D. Shewan, A. Dwivedy, R. Anderson, and C.E. Holt. Age-related changes underlie switch in netrin-1 responsiveness as growth cones advance along visual pathway. *Nature Neurosci*, 5:955–962, 2002.

- [41] J.R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors. *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.
- [42] R. Shirasaki, R. Katsumata, and F. Murakami. Change in chemoattractant responsiveness of developing axons at an intermediate target. *Science*, 279:105–107, 1998.
- [43] H. Song, G. Ming, Z. He, M. Lehmann, M. Tessier-Lavigne, and M.-M. Poo. Conversion of neuronal growth cone responses from repulsion to attraction by cyclic nucleotides. *Science*, 281:1515–1518, 1998.
- [44] H.-J. Song and M.-M. Poo. Signal transduction underlying growth cone guidance by diffusible factors. *Current Opinion in Neurobiology*, 9:355–363, 1999.
- [45] M. Spivak. *A Comprehensive Introduction to Differential Geometry. Vol. II*. Publish or Perish Inc., Wilmington, Del., second edition, 1979.
- [46] J.C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Chapman & Hall, 1989.
- [47] N. Sukumar, B. Moran, A. Yu Semenov, and V.V. Belikov. Natural neighbour galerkin methods. *International Journal for Numerical Methods in Engineering*, (50):1–27, 2001.
- [48] M. Tessier-Lavigne and C.S. Goodman. The molecular biology of axon guidance. *Science*, 274:1123–1133, 1996.
- [49] A. van Ooyen, editor. *Modeling Neural Development*. MIT Press, 2003.
- [50] J.G. Verwer and B.P. Sommeijer. A numerical study of mixed parabolic-gradient systems. *J. Comp. Appl. Math.*, 132:191–210, 2001.
- [51] E.W. Weisstein. *Concise Encyclopedia of Mathematics*. CRC Press, 2nd edition, 2002.
- [52] D.G. Wilkinson. Multiple roles of eph receptors and ephrins in neural development. *Nature Neuroscience Reviews*, 2:155–164, 2001.
- [53] N. Yamamoto, A. Tamada, and F. Murakami. Wiring up the brain by a range of guidance cues. *Progress in Neurobiology*, 68:393–407, 2003.
- [54] S. Young and N.M. Poo. Spontaneous release of transmitter from growth cones of embryonic neurons. *Nature*, 305:634–637, 1983.
- [55] J. Q. Zheng, M. Felder, J.A. Connor, and M.M. Poo. Turning of nerve growth cones induced by neurotransmitters. *Nature*, 368:140–144, 1994.

- [56] Y. Zou, E. Stoeckli, H. Chen, and M. Tessier-Lavigne. Squeezing axons out of the gray matter: a role for slit and semaphorin proteins from midline and ventral spinal cord. *Cell*, 102:363-375, 2000.

