MONOTONE SCHEMES FOR DEGENERATE AND NON-DEGENERATE PARABOLIC PROBLEMS

by

MIRNA LIMIC

M.B.A. Rochester Institute of Technology, 2001

B.Sc., Economics and Business, 1999

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

December 2006

Abstract

This thesis analyses several monotone schemes for degenerate second order parabolic PDEs over two-dimensional domains which is in most cases equal $(0,1)^2$. For such problems the degeneracy means that in addition to the axis spanned by the standard basis vectors, there exists another pair of orthogonal cordinate axes such that the spatial difference operator is of the second order in one of the directions, and of the first order in the remaining directions. The direction of one axis along which the spatial difference operator is of the second order we call the direction of diffusion.

The thesis considers only constant coefficient PDE's and therefore the degeneracy can be easily determined by solving the eigenvalue problem for diffusion tensor.

We analyse the impact of the direction of diffusion on the convergence of a scheme. Previous work on a second order elliptic problem has shown that central differences taken in the direction of diffusion produce a convergent scheme, wherease disalignment between the two result in non-convergent schemes. One of our aims was to check the validity of this finding and possibly improve the construction of schemes. As a result we present a novel approach to building monotone schemes from the diffusion tensor by taking spatial step sizes of different length in order to align the second order central differences with the direction of diffusion. We give a step by step algorithm and supply all of the findings. Our findings are based on MATLAB m file, and C language implementations.

Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank my supervisior for his guidance. In addition, I would like to express many thanks to Professor Anthony Peirce of the University of British Columbia Mathematics Department for the suggestions he gave as the second reader of my thesis. Finally, I thank everyone else who helped me with their knowledge and support in this thesis research.

To my family whose support enabled me to make this thesis. This is our success.

Chapter 1

Introduction

1.1 Introduction

We focus our study on monotone schemes for a class of degenerate linear parabolic partial differential equations. For the sake of comparison, monotone schemes for some non-degenerate problems are also analysed, such as the heat equation. Our goals are to

(1) investigate the efficiency of monotone schemes in generating a "good" numerical approximation, and

(2) confirm or disprove through numerical calculations and mathematical tools the claim that monotonicity has impact on the convergence of a scheme.

We consider two-dimensional problems on bounded domains. We limit ourselves to a square two-dimensional domain, which is in most cases $(0,1)^2$, and sometimes $(-1,1)^2$.

For such domains the degeneracy means that there exists another pair of orthogonal coordinate axes such that the spatial differential operator is of the second order in one of the directions, and of the first order in the remaining directions. Therefore, for degenerate equations on the defined domains we use terminology *in the direction of diffusion*, meaning, in the direction of one axis along which the diffusion operator is of the second order.

The monotone schemes are interesting because of the following simple fact. If stable and consistent, they are convergent, as proved by Barles and Souganidis [5]. Our findings confirm the work of Oberman [2] which states that central difference approximations to second order terms taken in the direction of diffusion result in a convergent monotone scheme for the corresponding elliptic problem. To that result our next goal was to create a method for building monotone schemes for degenerate and non-degenerate parabolic equations where the central difference approximations are taken in the direction of diffusion. The idea rests on taking discretization steps of different size along axes in order to "align" the derivative approximations with the diffusion direction.

Spatial discretization of linear PDEs gives a system of linear ODEs

$$\frac{d}{dt}\,\mathbf{u}(t) \;+\; A\,\mathbf{u}(t) \;=\; \mathbf{0},$$

where the matrix A is called the *system matrix*. For monotone schemes the system matrix has a particular structure. It has positive values along the diagonal and negative or zero values on the off-diagonal places. If we decide to solve the above system of ODEs using the explicit Euler method this special property of A allows us to compute the Courant-Friedrichs-Levy bound on the time step, τ. In some cases (like the case of the two-dimensional heat equation) this bound can be found analytically, while in general it must be computed numerically. If we decide to solve the system of ODEs using the implicit Euler, or some other implicit method, the described structure of A enables very easy implementation.

It is important to note that there is a 1-to-1 relationship between monotone schemes and matrices with above specified structure. Therefore, in order to build a monotone scheme for a linear PDE, one is in effect creating a system matrix of ODE's which has positive values along the diagonal and non-positive or zero values on the off-diagonal positions.

A class of linear PDEs to which we limit our research are given in Section 1.2. The schemes for these equations are built from degenerate schemes of the individual terms in the respective equations. In [2] the author studied transformations preserving monotonicity. For instance, the sum of monotone functions is a monotone function. Therefore, we could also say that we built our monotone schemes by summing the monotone schemes of each individual term in a respective equation. For example, we sum a monotone scheme for a convective part of an equation with a monotone scheme for the difussive part of that equation thus obtaining a monotone scheme for the equation. The monotone shemes we use are explained in Section 2.2.1.

So far we have presented some benefits of using a degenerate, i.e. monotone difference scheme. However due to the monotonicity requirement the accuracy of a monotone difference scheme is at most of the first order [2]. This fact was first explained by Barles and Souganidis [5]. Higher than first order schemes such as ENO, and WENO as given in Osher, Fedkiw [15] result in matrices that do not have the special property forementioned, and are therefore not monotone. Since we study monotone schemes we restrict ourselves to first order integration and use Forward Euler in deriving our solutions.

The software implementations are present in two programming languages. One is the MATLAB m language, and the other is C programming language. The files written in the m language are designed to work with a software package, the Toolbox of Level Set Methods, version 1.1, [1]. The Toolbox of Level Set Methods 1.1 is a set of routines implementing level set methods which solve time-dependent Hamilton-Jacobi partial differential equations. The C implementations are designed as stand-alone programs. The difference and reasons for two different implementations are given in Section 5.

1.2 Motivation

The analysis in this thesis is motivated by the research presented in [2], where degenerate elliptic and parabolic equations of the second order are examined for the purpose of constructing their numerical or approximate solutions. The emphasis is put on numerical methods which are called *monotone schemes*. For their definition refer to Section 2.1. The analysis of monotone schemes in this thesis focusses on three initial value problems on the

square domain, $D = (0,1) \times (0,1)$. We define the problems for a function f as follows

Problem 1. Heat equation	$f_t - \sigma_x^2 f_{xx} - \sigma_y^2 f_{yy} = 0$
Problem 2. Degenerate heat equation with advection	$f_t + \mathbf{v}\nabla f - \sigma^2 f_{xx} = 0$
Problem 3. General degenerate parabolic equation	$f_t - \sum_{ij=1}^{d} a_{ij}\,\partial_i\,\partial_j f = 0$
Problem 4. General degenerate parabolic equation with advection	$f_t + \mathbf{v}\nabla f - \sum_{ij=1}^{d} a_{ij}\,\partial_i\,\partial_j f = 0$

We implemented an example of Heat equation to observe behaviour of numerical solution in time when the scheme used is monotone. However, since Heat equation is discussed in many differential equation books, for example Boyce, DiPrima [9], we skip its theoretical analysis. To execute the implemented example see Appendix.

We supply initial and boundary conditions to the above problems in later chapters, yet we do not defer the definition of the parameters present in the above equations. The positive numbers $\sigma_x^2, \sigma_y^2, \sigma^2$ are usually called diffusion constants. Continuous functions $\mathbf{v}$ have two components v_x, v_y and define velocity fields on $D = (0,1)^2$. Continuous functions a_{ij} define a matrix $a = \{a_{ij}\}_{11}^{22}$ which is called *diffusion tensor*.

At this point we are obliged to explain our matrix notation. A matrix $a = \{a_{ij}\}_{11}^{22}$ is a matrix with indices (i,j) going from $(1,1)$ to $(2,2)$.

The matrix a must be symmetric at each point of D and must be either positive definite or positive semidefinite. In the former case it has two positive eigenvalues and Problem 3. is non-degenerate. In the latter case the matrix a has one zero eigenvalue and one positive eigenvalue and the corresponding Problem 3. is degenerate. Actually the degeneracy does not need to be realized at each point of D. It is sufficient that the degeneracy happens at a single point of $\overline{D} = [0,1]^2$.

Problem 3. can be rewritten in another form, which is utilized in Game theory,

Problem 3. $\qquad f_t - \mathrm{trace}[LDR] = 0$

where L, R are 2×2 matrices and D is Hessian of f. The matrices L, and R cannot be arbitrary since trace$[LDR]$ must coincide with $\sum_{ij=1}^{d} a_{ij}\,\partial_i\,\partial_j f$.

In order to introduce monotone schemes we start our analysis with Heat equation. Next, Degenerate heat equation is considered as a degenerate version of Heat equation. Monotone schemes which are developed for Heat equation are applied to Degenerate heat equation. An extension of investigation onto General degenerate heat equation is necessary

in order to draw conclusion on the efficiency of monotone schemes based on the results of our research. In addition, in the research of General degenerate heat equation we confirmed that the direction in which the central differences are taken has an impact on the convergence of a scheme. Chapter 4 presents a new method, created by M. Limic, for building monotone schemes using spatial step sizes of different length. This method builds monotone schemes from a diffusion tensor of the degenrate elliptic differential operator. We conclude our analysis of monotone schemes with General degenerate parabolic equation with advection taking into account the importance of the direction of central differences. In the analysis we use the method

The best way to demonstrate the efficiency of monotone schemes would be to compare the solutions f of the considered problem and its numerical approximation f_{app}. Here we are presented with an obvious difficulty. Solutions are not known and this fact is the basic reason which forces us to look for numerical or approximate solutions. Apart from theoretical results telling us about the convergence of numerical solutions towards solutions f, the efficiency can be studied by using examples for which the unique solutions f are known. We say that solution f is known in a closed form. In this report we analyze numerical solutions using examples for which solutions f are known in closed forms.

1.3 Outline

We have organized the material into eight chapters. Here we give a brief outline of the issues discussed in each chapter.

(i) In chapter 1 we give the motivation and goals of this thesis. We also introduce the topic of monotone schemes, and note some important facts regarding the system matrices associated to monotone schemes and how schemes are built.

(ii) Chapter 2 contains the theory of the degenerate elliptic and parabolic equations and corresponding numerical schemes. Following the theoretical background is a description of the central differences we had to construct to derive monotone schemes for problems considered. Finally we give our derivation of the CFL condition for monotone schemes and its comparison to the CFL condition implemented in the Toolbox of Level Set Methods 1.1.

(iii) Chapter 3 is devoted to a detailed study of monotone schemes by analysing the problems of Section 1.2. We present the derivation of analytical solution to three examples of Degenerate heat equation, as well as solutions to the examples of General degenerate parabolic equation. We consider each problem in turn giving our findings as the concluding discussion. While Degenerate heat equation serves exclusively for the analysis of the efficiency of montone schemes, General degenerate parabolic equation is additionally used for an analysis of the importance of the direction in which the central differences are taken on the convergence of a scheme.

(iv) Chapter 4 presents a method we developed for computing monotone schemes for the case of degenerate and non-degenerate diffusion tensors. This is achieved by taking numerical step sizes of different length along two coordinate axes. We present the

algorithm for deriving the step size in y-axis direction given the number of nodes in the x-axis direction. Our findings are based on the implementation of several examples of General degenerate parabolic equation to this setting.

(v) Chapter 5 is concerned with the implementation details of m files only. Since the C programming language implementations were not required at any time during this thesis research, their implementation will not be discussed here. They were done as an aid to help overcome the disadvantages of slowness and memory overconsumption of the m language implementations. If however, the reader is interested in obtaining the implementation details of C files as well at the C implementations themselves, they can contact me at `mirnalim@cs.ubc.ca`.

(vi) Based on the work completed, in Chapter 6 we propose some topics that can be taken as further research.

(vii) Appendix serves as a small manual for compiling and executing the programs. It contains a listing of the programs that implement the problems considered in the course of the thesis research.

As an end note to this chapter it should be said that this thesis is written to be a learning tool just as much as a presentation of the published work and our research conducted in the area of degenerate monotone equations and their schemes.

Chapter 2

Theoretical background

The theory we look at and develop in this thesis rests mainly on the published work in the field of degenerate elliptic and parabolic equations. Methods of the eigefunction theory are used for the purpose of deriving analytical solutions to PDEs.

2.1 Degenerate equations

In the work of Barles and Souganidis [5] it is shown that monotone, stable, and consistent schemes are also convergent. The only drawback in building such schemes is the fact that they are of the first order of convergence. If the coefficients of a PDE are smooth enough, schemes with higher order of convergence can be successfuly used. However, at points where the coefficients are not smooth one prefers to use monotone schemes due to the stability they have.

Monotone schemes are not only used for linear PDE problems, but also for non-linear PDEs. In effect, they are usually defined for non-linear PDEs. Spatial discretization of an initial value problem (IVP) for a PDE gives a system of ODEs of the form

$$\frac{d}{dt}u_i(t) \;=\; F^i(t,\mathbf{u}(t)), \quad i \;=\; 1,2,\ldots,N, \tag{2.1}$$

where $u(t,x)$ are the solutions of the IVP for a PDE. These solutions are at grid nodes x_i approximated by functions $u_i(t)$. These functions define a column vector $\mathbf{u}(t)$ of approximate solutions. The functions F^i result from spatial discretization and can depend explicitly on t, as well as on the approximate solutions u_i. The constructed system (2.1) of ODEs must be solved using discretization in the time variable. This means that we consider discrete times $t_0 = 0, t_1, \ldots, t_M = T$ and approximate solutions $u_i(t_k)$ by numbers $u_i^{(k)}$. Thus the columns $\mathbf{u}(t_k)$ are approximated by colums $\mathbf{u}^{(k)}$. In the case of the Forward Euler method applied to (2.1) we obtain

$$u_i^{(k+1)} \;=\; u_i^{(k)} \;+\; \tau F^i(t_k,\mathbf{u}^{(k)}) \;:=\; H^i(\tau,t_k,\mathbf{u}^{(k)}). \tag{2.2}$$

Based on the definition of monotonicity as given in [14] we give the following definition.

DEFINITION **2.1.1 (Monotone scheme)** *We say that a scheme is monotone if the functions* $H^i(\tau, t_k, \mathbf{u}^{(k)})$ *do not decrease when their variables* $u_i^{(k)}$ *increase.*

It is important to note that a scheme is not monotone for all values of time increment τ. This fact implies a necessity to analyse the monotonicity of a scheme with respect to the choice of the time step. The aim here is to find the biggest time step for which the scheme is monotone. This issue is discussed in Section 2.3.

For example, take the parabolic equation $u_t - u_{xx} = 0$. After carrying out the central difference spatial discretization of u_{xx} we get the ODE system

$$\frac{d}{dt}\mathbf{u}(t) \; - \; F[\mathbf{u}(t)] = 0,$$

where the functions

$$F^i(u) = \frac{1}{dx}(a_{i-1} + a_{i+1})$$

are defined by

$$a_{i-1} = \frac{u_i^n - u_{i-1}^n}{dx},$$
$$a_{i+1} = \frac{u_i^n - u_{i+1}^n}{dx}.$$

Then, in the present case, the system of difference equations (2.2) has the form

$$u_i^{(k+1)} \; = \; \left(1 - \frac{2\tau}{dx^2}\right) u_i^{(k)} \; + \; \frac{\tau}{dx^2}\left(u_{i+1}^{(k)} \; + \; u_{i-1}^{(k)}\right),$$

defining a scheme. Looking at the definition of monotonicity 2.1.1 we see that this scheme is monotone for all $0 < \tau \leq dx^2/2$. We computed this result by noting that we must have

$$1 - \frac{2\tau}{dx^2} \geq 0 \quad \rightarrow \quad \tau \leq \frac{dx^2}{2},$$
$$\frac{\tau}{dx^2} \geq 0 \quad \rightarrow \quad \tau \geq 0$$

in order for the considered scheme to be monotone.

Next we give the definition of degenerate ellipticity for which we note that it includes the definition of monotonicity. For each node i the symbol $N(i)$ denotes the neighbourhood of node i.

DEFINITION **2.1.2** *We say that a scheme*

$$F^i[u] = F^i\left(u_i, \left.\frac{u_i - u_j}{|x_i - x_j|}\right|_{j \in N(i)}\right), i = 1, \ldots, N$$

is *degenerate elliptic* if each F^i is a non-decreasing function in each of its arguments.

Given the above definition of degenerate ellipticity it follows that every degenerate scheme is also monotone.

Next we would like to state an important theorem taken from [2] and class notes of [6] for our discussion on stability and monotonicity.

THEOREM **2.1** *A scheme is monotone and non-expansive in the l^∞ norm if and only if it is degenerate elliptic.*

THEOREM **2.2** *The accuracy of a monotone finite difference scheme is at most first order.*

The three problems defined in Section 1.2 must be supplied with boundary and initial conditions. So defined we call them initial boundary value problems for linear PDEs.

In the remaining part of this section we describe a procedure for constructing monotone schemes. First we have to define conditions on the coefficients a_{ij}, v_i, and c of a general linear elliptic differential operator of the second order

$$A(\mathbf{x}) \;=\; -\sum_{i,j=1}^{d} a_{ij}(\mathbf{x})\partial_i\partial_j \;+\; \sum_{i=1}^{d} v_i(\mathbf{x})\partial_i \;+\; c(\mathbf{x}). \tag{2.3}$$

DEFINITION **2.1.3** *The differential operator is called degenerate elliptic if the following three conditions are valid*

(a) The functions $a_{ij} = a_{ji}$, v_i $(i, j = 1, 2, \ldots, d)$ and c satisfy the following conditions:

$$|v_i|, |c| \quad \leq M, \quad c \leq 0, \quad a_{ij}, v_i, c \quad \textit{uniformly continuous on } \mathbb{R}^d. \tag{2.4}$$

(b) The differential operator (2.3) is elliptic on $\mathbb{R}^d$, i.e there exists a positive number $\overline{M}$ such that

$$\overline{M}\,|z|_2^2 \;\geq\; \sum_{i,j=1}^{d} a_{ij}(\mathbf{x}) z_i \bar{z}_j \geq 0, \quad \mathbf{x} \in \mathbb{R}^d, \tag{2.5}$$

where $z_i \in \mathbb{C}^d$ are complex numbers and $|z|_2$ the corresponding l_2-norm. In other words, diffusion tensor is positive semidefinite.

(c) There is a point $\mathbf{x} \in \mathbb{R}^d$ such that $\sum_{i,j=1}^{d} a_{ij}(\mathbf{x}) z_i \bar{z}_j = 0$.

We say that the diffusion tensor is defined by functions a_{ij}, convection is defined by velocities v_i and the function c is called the killing rate. For the problems considered in this thesis the killing rate is zero.

Examples of this thesis have either homogeneous or non-homogeneous Dirichlet boundary conditions or periodic boundary conditions. Homogeneous Dirichlet boundary condition is specified by zero value of solution, non-homogeneous by non-zero value of solution.

For an IVP defined by

$$\begin{aligned} \partial_t u(t,\mathbf{x}) \;-\; A(\mathbf{x})\, u(t,\mathbf{x}) \;&=\; 0, \\ u(0,\mathbf{x}) \;&=\; u_0(\mathbf{x}), \\ u(t,\mathbf{x})\,\partial D \;&=\; 0, \end{aligned} \tag{2.6}$$

we use those spatial discretizations for which we get monotone schemes. In Section 2.2.1 some of the possibilities are described in detail. A discretization of the differential operator $A(\mathbf{x})$ results in a system matrix A with entries A_{ij}, where indices i, j refer to grid nodes. Discretizations leading to monotone schemes are easily characterized using the notion of matrices of *positive type* [4], which we define next.

DEFINITION **2.1.4** *A matrix A with entries A_{ij} is said to be of positive type if it has positive diagonal entries, non-positive off-diagonal entries and positive or zero row sums,*

$$A_{ii} > 0, \quad A_{ij} \leq 0 \quad for \quad i \neq j, \quad A_j = \sum_{i \in I} A_{ij} \geq 0.$$

A matrix of positive type with zero row sums, $A_j = 0 \ \ \forall j \in I$, is called conservative.

If we wish to construct a monotone scheme for the IVP (2.6) we need to follow certain principles

RULES OF DISCRETIZATION

Spatial discretization of the differential operator $A(\mathbf{x})$ must be carried out using the following two rules in order to ensure the monotonicity of the resulting scheme

1) The convection term $\sum_{i=1}^{d} v_i(\mathbf{x})\partial_i u$ is discretized using the upwinding scheme.
2) The diffusion term $-\sum_{i,j=1}^{d} a_{ij}(\mathbf{x})\partial_i\partial_j u$ must be discretized in a way to make the resulting system matrix of positive type.

If we abide to these rules, we build a system of ODEs of the form

$$\frac{d}{dt}\,\mathbf{u}(t) \ + \ A\,\mathbf{u}(t) \ = \ \mathbf{0},$$

where the matrix A is of positive type. This matrix, as mentioned before, is called the *system matrix*. A system matrix of positive type results in a monotone scheme, and a monotone scheme always has a system matrix of positive type. Therefore we could laicaly say that system matrices of positive type and monotone schemes are in a $1-$to-1 relationship.

Let us define a diagonal matrix D with entries $D_{ii} = A_{ii}$ and a matrix B with entries $B_{ij} = -A_{ij} \ \ i \neq j$. Then

$$A = D - B.$$

If we use Forward Euler method, the constructed ODE is converted into the following system of difference equations

$$\mathbf{u}^{(k+1)} \ = \ (I \ - \ \tau D)\,\mathbf{u}^{(k)} \ + \ \tau B\,\mathbf{u}^{(k)}.$$

This scheme is monotone for sufficiently small τ. The computation of τ is given in Section 2.3 where we use the same decomposition of the matrix A onto D and B in order to compute the expression for calculating the upper bound on τ, i.e. the CFL condition.

2.2 Central difference approximations

We would like to introduce a way of defining central differences that is different from the one available in the Toolbox of Level Set Methods 1.1 distribution. Central differences as defined below are the ones used by Oberman [2], and differ from the ones in the Toolbox of Level Set Methods 1.1 in their calculation of the second order mixed partial derivative.

2.2.1 Central differences defined

The orthogonal coordinate system in $\mathbb{R}^d$ is determined by unit vectors $\mathbf{e}_i$ of the canonical basis. Points $\mathbf{x} = h\sum_{l=1}^{d} k_l \mathbf{e}_l$, h is the grid step, $k_l \in \mathbb{Z}$, define a numerical grid G_h on $\mathbb{R}^d$.

The partial differential operators of the first and second orders can be approximated in the usual way

$$\partial_i f(\mathbf{x}) \to \eth_i f(\mathbf{x}) = \frac{1}{2h}\left[f(\mathbf{x}+\mathbf{e}_i h) - f(\mathbf{x}-\mathbf{e}_i h)\right], \tag{2.7}$$

$$\partial_i^2 f(\mathbf{x}) \to \eth_{ii} f(\mathbf{x}) = \frac{1}{h^2}\left[f(\mathbf{x}+\mathbf{e}_i h) - 2f(\mathbf{x}) + f(\mathbf{x}-\mathbf{e}_i h)\right], \tag{2.8}$$

where we use the symbol $\eth$ to indicate the approximation to a first or second order derivative, and symbol $\to$ to show that the operator to the left of the arrow is approximated by the discrete operator to the right of the arrow. Following our rules of discretization we discretize the operators $v_i\partial_i$ by using the upwinding procedure rather than by using the central difference operator $\eth_i f(\mathbf{x})$. Thus the first partial derivative in the x-axis direction would be approximated by one of the following two possibilities

$$\frac{1}{h}\Big[f(\mathbf{x}+\mathbf{e}_i h) - f(\mathbf{x})\Big], \frac{1}{h}\Big[f(\mathbf{x}) - f(\mathbf{x}-\mathbf{e}_i h)\Big],$$

which are called the forward and backward finite difference operators respectively.

The second order partial differential operator ∂_i^2 is approximated by two subsequent applications of the first order operators. For example, if the first step of the approximation is the forward finite difference operator

$$\frac{1}{h}\Big[f(\mathbf{x}+\mathbf{e}_i h) - f(\mathbf{x})\Big],$$

then the application of the backward finite difference operator gives us

$$\begin{aligned}&\tfrac{1}{h}\left[\tfrac{1}{h}(f(\mathbf{x}+\mathbf{e}_i h) - f(\mathbf{x})) - \tfrac{1}{h}(f(\mathbf{x}) - f(\mathbf{x}-\mathbf{e}_i h))\right]\\ &\quad = \tfrac{1}{h^2}\Big[f(\mathbf{x}+\mathbf{e}_i h) - 2\,f(\mathbf{x}) + f(\mathbf{x}-\mathbf{e}_i h)\Big].\end{aligned}$$

We would have arrived at the same answer had we applied the directions of the approximation in the reversed order, i.e. first backward, and then forward.

The mixed partial differential operator can be approximated by one of the following four possibilities

$$\begin{aligned}&\partial_i\partial_j f(\mathbf{x}) \to \eth_{ij} f(\mathbf{x}) =\\ &\frac{1}{h^2}\begin{cases} f(\mathbf{x}\pm\mathbf{e}_i h \pm \mathbf{e}_j h) - f(\mathbf{x}\pm\mathbf{e}_i h) - f(\mathbf{x}\pm\mathbf{e}_j h) + f(\mathbf{x}), \\ -f(\mathbf{x}\pm\mathbf{e}_i h \mp \mathbf{e}_j h) + f(\mathbf{x}\pm\mathbf{e}_i h) + f(\mathbf{x}\mp\mathbf{e}_j h) - f(\mathbf{x}).\end{cases}\end{aligned} \tag{2.9}$$

The above operators are derived by subsequent applications of forward finite difference operator and backward finite difference operator in the directions $\mathbf{e}_1$ and $\mathbf{e}_2$.

We now show how one of the four above discretizations for a mixed partial differential operator can be obtained. We take first the backward finite difference operator in the first variable, i,

$$\frac{1}{h}\Big[f(\mathbf{x}) - f(\mathbf{x}-\mathbf{e}_i h))\Big],$$

and then apply the backward finite difference operator in j

$$\frac{1}{h}\Big[f(\mathbf{x}) - f(\mathbf{x} - \mathbf{e}_j h))\Big].$$

The resulting expression is

$$\begin{aligned} &\tfrac{1}{h}\left[\tfrac{1}{h}(f(\mathbf{x}) - f(\mathbf{x} - \mathbf{e}_i h)) - \tfrac{1}{h}(f(\mathbf{x} - \mathbf{e}_j h) - f(\mathbf{x} - \mathbf{e}_i h - \mathbf{e}_j h))\right] \\ &= \tfrac{1}{h^2}\Big[f(\mathbf{x}) - f(\mathbf{x} - \mathbf{e}_i h) - f(\mathbf{x} - \mathbf{e}_j h) + f(\mathbf{x} - \mathbf{e}_i h - \mathbf{e}_j h)\Big]. \end{aligned} \tag{2.10}$$

This is easily recognisable as one of the four possibilities given in (2.9). With a similar procedure one can obtain the remaining three possibilities given in (2.9).

In the problems considered in this thesis we have used upwinding for the approximation of the first order derivatives. When approximating second order non-mixed terms we used operator ∂_{ii}. For mixed partial derivative approximations we used half-sums of the operators defined in the expression (2.9). For example, a half sum of the first two operators of (2.9) is

$$\begin{aligned} &\tfrac{1}{2h^2}\Big[f(\mathbf{x} + \mathbf{e}_i h + \mathbf{e}_j h) - f(\mathbf{x} + \mathbf{e}_i h) - f(\mathbf{x} + \mathbf{e}_j h) + f(\mathbf{x}) + \\ &\quad f(\mathbf{x} - \mathbf{e}_i h - \mathbf{e}_j h) - f(\mathbf{x} - \mathbf{e}_i h) - f(\mathbf{x} - \mathbf{e}_j h) + f(\mathbf{x})\Big]. \end{aligned}$$

Which half-sums are taken is explained in Section 2.2.4. The choice must give us a system matrix of positive type.

2.2.2 Differences of the Toolbox of Level Set Methods 1.1

The difference approximations of second derivatives implemented in the Toolbox of Level Set Methods 1.1 differ from our central differences of Section 2.2.1 in their computation of the mixed partial derivative approximation. In place of upwinding, the Toolbox of Level Set Methods 1.1 implements

$$\frac{1}{2h}(f(\mathbf{x} + \mathbf{e}_i h) - f(\mathbf{x} - \mathbf{e}_i h))$$

as the approximation of the first order term in direction $\mathbf{e}_i$, which we recognize as our first order approximation operator $\partial_i f(\mathbf{x})$.

Let the first order partial derivative approximation in the variable i be as above. Then applying to it the approximation in the variable j yields

$$\begin{aligned} &\tfrac{1}{2h}\left[\tfrac{1}{2h}(f(\mathbf{x} + \mathbf{e}_i h + \mathbf{e}_j h) - f(\mathbf{x} - \mathbf{e}_i h + \mathbf{e}_j h)) - \tfrac{1}{2h}(f(\mathbf{x} + \mathbf{e}_i h - \mathbf{e}_j h) - f(\mathbf{x} - \mathbf{e}_i h - \mathbf{e}_j h))\right] = \\ &\quad \tfrac{1}{4h^2}\Big[f(\mathbf{x} + \mathbf{e}_i h + \mathbf{e}_j h) - f(\mathbf{x} - \mathbf{e}_i h + \mathbf{e}_j h) - f(\mathbf{x} + \mathbf{e}_i h - \mathbf{e}_j h) + f(\mathbf{x} - \mathbf{e}_i h - \mathbf{e}_j h)\Big]. \end{aligned}$$

This difference approximation of the mixed partial derivative is not monotone. Therefore, schemes using these differences in general are not monotone. This is discussed further in Section 3.2.2. The non-monotonicity resulting from using the above non-monotone mixed partial derivative approximation can give a non-convergent scheme. This is discussed further for the case of General degenerate parabolic equation.

2.2.3 Checking derivative approximations

It is well known that the second order difference operator $\boxminus_{ii}$ is the correct approximation of ∂_i^2.

There exists a simple check which demonstrates that the four discretizations of $\partial_1\partial_2$ (2.9) given in Section 2.2.1 are correct and ensure the convergence of numerical solutions to the unique solution of considered IVPs for PDEs. Here we show the calculation for one of possible four approximation of (2.9). A check for the other three approximations can be done using exactly the same procedure. Our check is based on the the necessary and sufficient condition for convergence [8] below.

For any polynomial of the form

$$P(x_1, x_2) \;=\; \omega_1 x_1 \;+\; \omega_2 x_2 \;+\; \omega_3 x_1 x_2$$

expressions $\partial_1\partial_2 P(x_1, x_2)$ and $\boxminus_{12} P(x_1, x_2)$ must have the same value equal to ω_3.

It is obvious that $\partial_1\partial_2 P(x_1, x_2)$ is ω_3, so let us check that $\boxminus_{12} P(x_1, x_2)$ has value ω_3 for the example of Section 2.2.1 where two applications of the backward mixed partial difference operator were applied to derive the operator $\boxminus_{ij}$, $i \neq j$. First we apply the backward differential operator in the first variable

$$\begin{gathered}\tfrac{1}{h}\Big(P(\mathbf{x}) - P(\mathbf{x} - \mathbf{e}_1 h)\Big) = \\ = \tfrac{1}{h}\Big(\omega_1 x_1 + \omega_2 x_2 + \omega_3 x_1 x_2 - \Big(\omega_1(x_1 - h) + \omega_2 x_2 + \omega_3(x_1 - h)x_2\Big) \\ = \tfrac{1}{h}\Big(\omega_1 h + \omega_3 x_2 h\Big)\end{gathered}$$

and then the backward difference operator in the second variable. This gives

$$\begin{gathered}\tfrac{1}{h}\Big[\tfrac{1}{h}(\omega_1 h + \omega_3 x_2 h) - \tfrac{1}{h}(\omega_1 h + \omega_3(x_2 - h)h)\Big] \\ = \tfrac{1}{h^2}\Big[\omega_3 h^2\Big] = \omega_3.\end{gathered}$$

A similar check can be done for any of the other three possibilities from (2.9). Therefore, we conclude that the operator $\boxminus_{ij}$ is a good approximation of the operator $\partial_i\partial_j$ for $i \neq j$.

2.2.4 Difference approximations and matrices of positive type

Using our discretizations, the quadratic operator $a_{11}(\partial_1)^2 + a_{22}(\partial_2)^2$ is approximated by the differences $a_{11}\boxminus_{11} + a_{22}\boxminus_{22}$.

Values of a_{ij}, $i \neq j$, dictate the choice of approximation for $a_{ij}\partial_i\partial_j$. If $a_{12} \geq 0$, then $a_{12}\partial_1\partial_2$ should be approximated by the half sum of the first two possibilities of (2.9), and otherwise by the half sum of the second two possibilities of (2.9). Thus when $a_{12} > 0$ we use the following approximation of the mixed partial derivatives

$$\begin{gathered}\tfrac{1}{2h^2}\Big[f(\mathbf{x} + \mathbf{e}_i h + \mathbf{e}_j h) - f(\mathbf{x} + \mathbf{e}_i h) - f(\mathbf{x} + \mathbf{e}_j h) + \\ f(\mathbf{x} - \mathbf{e}_i h - \mathbf{e}_j h) - f(\mathbf{x} - \mathbf{e}_i h) - f(\mathbf{x} - \mathbf{e}_j h) + 2f(\mathbf{x})\Big],\end{gathered}$$

while when $a_{12} < 0$ we use

$$\frac{1}{2h^2}\Big[- f(\mathbf{x} + \mathbf{e}_i h - \mathbf{e}_j h) + f(\mathbf{x} + \mathbf{e}_i h) + f(\mathbf{x} - \mathbf{e}_j h) \\ -f(\mathbf{x} - \mathbf{e}_i h + \mathbf{e}_j h) + f(\mathbf{x} - \mathbf{e}_i h) + f(\mathbf{x} + \mathbf{e}_j h) - 2f(\mathbf{x})\Big].$$

An approximation based on a different choice can result in a non-convergent method. For further discussion on this subject see Section 3.2.

Discussed approximations of first and second order terms give us a system of ODEs

$$\frac{d}{dt}\mathbf{u}(t) = -A\,\mathbf{u}(t), \quad \mathbf{u}(0) = \mathbf{u}_0$$

where A is the $n \times n$ system matrix, and $\mathbf{u}_0$ is the initial condition. If we were to write out the terms of A, we would see that it has positive diagonal entries and non-positive off-diagonal entries. In other words matrix A is of positive type.

For the two dimensional case where $A = -\sum a_{ij}\partial_i\partial_j$ using centered differences from (2.9) results in entries of the system matrix corresponding to the grid node (k, l) of the following form

$$\begin{aligned} A_{kl\,kl} &= 2h^{-2}[a_{11} + a_{22} - |a_{12}|] \\ A_{kl\,k\pm1l} &= -h^{-2}[a_{11} - |a_{12}|] \\ A_{kl\,kl\pm1} &= -h^{-2}[a_{22} - |a_{12}|] \\ A_{kl\,k\pm1l\pm1} &= -h^{-2}|a_{12}| \quad (= A_{kl\,k\pm1l\mp1}). \end{aligned} \tag{2.11}$$

In the case that values in square brackets are positive the resulting matrix with entries $A_{ij\,kl}$ is of positive type, i.e. it has positive diagonal entries and non-positive off-diagonal entries. Otherwise, schemes are not monotone. Motzkyn and Wasov [4] suggest for non-monotone schemes to perform rotations. There is also another possibility. In order to obtain monotone schemes one can use stencils having different length in x_1 and x_2 directions. This idea is thoroughly pursued in this thesis.

Let us apply the described approximation procedure for the general case of the differential operator A to Degenerate heat equation. For this equation the operator is

$$A = -\sum_{ij=1}^{2} a_{ij}\,\partial_i\,\partial_j + \sum_{i=1}^{2} v_i\,\partial_i\,. \tag{2.12}$$

In the present example the corresponding diffusion tensor $a = \{a_{ij}\}_{11}^{22}$ is

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

The off-diagonal entries in the system matrix corresponding to the grid node (k, l) have the form

$$\begin{aligned} A_{kl\,k\pm1l} &= -h^{-2}a_{11} - h^{-1}\begin{cases} -v^1_{k+1l} & \text{for } v^2_{k+1l} \geq 0 \\ v^1_{k-1l} & \text{for } v^2_{k-1l} \leq 0 \end{cases} \\ A_{kl\,kl\pm1} &= -h^{-1}\begin{cases} -v^2_{kl+1} & \text{for } v^2_{kl+1} \geq 0 \\ v^2_{kl-1} & \text{for } v^2_{kl-1} \leq 0 \end{cases} \end{aligned} \tag{2.13}$$

where $k, l \in \mathbb{N}$ identify nodes on the grid k units in direction $\mathbf{e}_1 h$, and l units in the direction $\mathbf{e}_2 h$, and velocity at a node (k, l) is $\mathbf{v}_{kl} = (v^1_{kl}, v^2_{kl})$.

The above matrix A is of positive type if the velocities are chosen in such a way to ensure that $A_{kl\,k\pm1l}$ and $A_{kl\,kl\pm1}$ are non-negative.

The diagonal entries of the system matrix must be equal to or greater than the negative sum of the off-diagonal entries in order to ensure monotonicity. In this thesis we have chosen to set the diagonal entries equal to the negative sum of the off-diagonal entries.

Let us look at another example. On the elliptic differential operator of General degenerate parabolic equation

$$A \;=\; -\sum_{ij=1}^{2} a_{ij}\,\partial_i\,\partial_j \tag{2.14}$$

one can apply the described approximation procedure when the corresponding diffusion tensor has the form

$$a \;=\; \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Observe that $det(a) = 0$, which means that the problem is degenerate.

We can say that a discretization of General degenerate parabolic equation relies on the appropriate choice of the discretization of the mixed partial derivatives. What "a good choice" is, depends on the diffusion tensor.

2.3 Courant-Friedrichs-Levy (CFL) condition for monotone schemes

We now present our derivation of the CFL condition for monotone schemes.

Let us remind ourselves of the structure of a matrix A of positive type. It must have positive diagonal entries, negative or zero off-diagonal entries and the sum of entries in each row must be either zero or positive. Our differential operator (2.3) must be discretized by a matrix A, where A must be of positive type.

The system matrix of the initial value problem for ODEs

$$\frac{d}{dt}\mathbf{u}(t) \;+\; A\mathbf{u}(t) \;=\; \mathbf{0}$$

can be rewritten in the following form. Let

$$A = D - B,$$

where D is a diagonal matrix with entries $d_{ii} = a_{ii} > 0$, and B is a non-negative matrix with entries $b_{ij} = -a_{ij} \geq 0, b_{ii} = 0$. Set $p = \max_i d_{ii}$ and rewrite the matrix A as

$$A \;=\; pI \;-\; Q,$$

where the matrix Q has entries $q_{ij} = b_{ij}$ for $i \neq j$ and $q_{ii} = p - d_{ii}$. Then the above initial value problem for ODEs has the form

$$\frac{d}{dt}\mathbf{u}(t) \; - \; p\,\mathbf{u}(t) \; + \; Q\,\mathbf{u}(t) \; = \; \mathbf{0}.$$

Using Forward Euler method we get a system for discretizations $\mathbf{u}^{(k)}$ of $\mathbf{u}(t_k)$

$$\mathbf{u}^{(k+1)} \; = \; \big(1 - p\tau\big)\,\mathbf{u}^{(k)} \; + \; \tau\,Q\,\mathbf{u}^{(k)} \; = \; \mathbf{0}. \tag{2.15}$$

Then the CFL condition is determined by the maximal time-interval, τ_{CFL},

$$\tau_{CFL} \; = \; p^{-1}.$$

For matrices A of positive type so determined τ_{CFL} is theoretical, and consequently optimal. The Toolbox of Level Set Methods 1.1 calculates the timestep by considering terms of a partial differential equation separately, meanwhile making sure not to incorrectly set the time step to the minimum of the advection time step and the diffusion time step, as shown in the following example.

Let $A = A_1 + A_2$ where both matrices A_r are of positive type, $A_r = p_r I - Q_r$, and $p_1 \; = \; p_2$. Then a bound of the maximal time interval τ for the system (2.15) from the corresponding time intervals $\tau_r = 1/p_r$ is obtained as follows

$$\tau \; \leq \; \min\Big(\frac{1}{p_1}, \frac{1}{p_2}\Big) = \frac{1}{p}.$$

However, the maximal time interval τ_{CFL} for A is

$$\tau_{CFL} \; \leq \; \frac{1}{2p}.$$

So the actual maximal time step is twice smaller than the time step computed as a minimum of the τ_1 and τ_2. In other words the scheme is unstable for $\tau = \frac{1}{p}$, since $\tau \; > \; \tau_{CFL}$.

Let us now correctly state the above computation of the maximal time step.

LEMMA. **2.1** *Let $A = A_1 + A_2$ where both matrices A_r are of positive type, $A_r = p_r I - Q_r$. Then a bound of the maximal time interval τ_{CFL} for the system (2.15) can be obtained from the corresponding time-intervals $\tau_r = 1/p_r$ as follows*

$$\tau_{CFL} \; \geq \; \Big(\frac{1}{\tau_1} \; + \; \frac{1}{\tau_2}\Big)^{-1}.$$

A proof of this result follows simply from $p \leq p_1 + p_2$. Therefore, taking $\tau \; = \; (\frac{1}{\tau_1} \; + \; \frac{1}{\tau_2})^{-1}$ makes the method stable. We have to point out that this is not the largest possible "stable" time step. It follows that the optimal value of τ_{CFL} should be calculated numerically from the diagonal entries d_{ii} rather than using this lemma.

Let us show an application of this lemma. In the case of level set equation in one dimension the matrix entries d_{ii} are $h^{-1}|v(x_i)|$ so that the associated τ_1 must be

$$\tau_1 \; = \; \max_i \; \frac{h}{|v(x_i)|}.$$

This is the theoretical value of τ_1. This τ_1 depends on the coefficients of differential operator and discretization procedure (upwinding in the considered case). In the case of level set equation in two dimensions we have theoretical values τ_1, τ_2 and a bound on τ_{CFL} from the lemma. However, the theoretical τ_{CFL} is generaly larger and can be obtained directly from the entries d_{ii}.

Let us point out again that τ depends on the coefficients of the elliptic differential operator A defined by (2.3) and the discretization procedure used for the approximation of the first and second order terms. For example, assume that we use discretization that results in the matrix entries (2.11) and additionally that the system matrix is of positive type. Then $\tau_{CFL} = 1/p$ implies

$$\tau_{CFL} = \frac{h^2}{2 \max_{kl} \left[a_{11}(\boldsymbol{x}_{kl}) + a_{22}(\boldsymbol{x}_{kl}) - |a_{12}(\boldsymbol{x}_{kl})| \right]},$$

where $\boldsymbol{x}_{kl}$ are grid nodes. The obtained expression coincides with the well-known expression $\tau_{CFL} = h^2(4\sigma^2)^{-1}$ for $A = -\sigma^2 \sum_{i=1}^{2} \partial_i^2$.

2.4 Consistency of monotone schemes

We have shown in Section 2.3 how to ensure stability of a monotone scheme. Here we discuss the consistency of the monotone schemes investigated in this thesis starting with a definition from Strikwerda, [8].

DEFINITION **2.4.1** *Given a partial differential equation $Pu = f$ and a finite difference scheme, $P_{dt,dx}v = f$, we say the finite difference scheme is consistent with the partial differential equation if for any smooth function $\phi(t, x_1, x_2)$*

$$P\phi - P_{dt,dx}\phi \to 0 \quad as \quad dt \to 0, dx = (dx_1, dx_2) \to 0,$$

the convergence being pointwise convergence at each grid point.

In our case $P = \partial_t + \sum_i v_i \partial_i - \sum_{ij} a_{ij} \partial_i \partial_j$ and

$$\begin{aligned} P_{dt,dx} &= \text{Forward Euler difference operator} + \text{Upwinding difference operator} \\ &- \text{Second order difference operator.} \end{aligned}$$

In the class notes [6] it is shown that explicit Euler difference method, and Upwinding method are consistent. Therefore, we have to demonstrate that our Second order difference operator is consistent with $\sum_{ij} a_{ij} \partial_i \partial_j$. We denote them by $P_{dt,dx}$ and P, respectively. We already know that both operators give the same result when applied to second order polynomials $T_2(x_1, x_2)$. Hence, if

$$\phi(x_1, x_2) = T_2(x_1, x_2) + R_3(dx_1, dx_2), \quad x_1 = hk + dx_1, x_2 = hl + dx_2,$$

at a grid node (hk, hl) where T_2 is a second order Taylor polynomial, then

$$P\phi - P_{dt,dx}\phi = PR_3 - P_{dt,dx}R_3.$$

The remainder R_3 is the series of monomials of arguments dx_1 and dx_2 of the order 3 and higher. Second order partial derivatives of R_3 behave like $\delta = |dx_1| + |dx_2|$ for small values of δ, proving the consistency.

Chapter 3

Analysing Problems 2. and 3.

In this Chapter we focus on Degenerate heat equation and General degenerate parabolic equation using the tools developed in Chapter 2.

3.1 Degenerate heat equation

We write Degenerate heat equation

$$u_t(\mathbf{x}) + \mathbf{v} \bigtriangledown u - u_{xx} = 0, \tag{3.1}$$

where $\mathbf{v}$ is a velocity field. This parabolic equation is interesting since it has a convection and a diffusion term, where the diffusion is present only in the direction of the x-axis. We study the equation for three separate velocity fields for which analytical solutions can be found.

3.1.1 Goal and problem setting

Our goal is the implementation of monotone schemes for the three problems defined below. We consider these three cases in order to investigate the efficiency of monotone schemes for an equation involving constant and vortex velocity fields. We look at two examples of the constant velocity field, since in one of the examples the velocity is aligned with the coordinate axis, while in the other example it is not.

We begin with the definition of our two examples for Degenerate heat equation which we will call Problem A, and Problem B.

Problem A. Consider domain $D = (0,1)^2$, with initial condition

$$\omega(x,y) = g(y)\sqrt{2}\sin(\pi x),$$

where

$$g(y) = \begin{cases} \sin(2\pi y) & \text{for} \quad y \in [0,1/2] \\ 0 & \text{for} \quad y \notin [0,1/2]. \end{cases}$$

The velocity is constant $\mathbf{v} = (0, 1)$, and the solution to (3.1) is

$$u(t, x, y) = \exp(-\pi^2 t) g(y - v_2 t) \sqrt{2} \sin(\pi x).$$

The function $u(t, x, y)$ is actually defined on a strip $(0, 1) \times (-\infty, \infty) \subset \mathbb{R}^2$, and has zero values at $x = 0$ as well as at $x = 1$. In addition, the function u is different from zero only for $0 < y - v_2 t < \frac{1}{2}$. If we consider the problem for $t \in [0, \frac{1}{v_2}]$, then we can claim that solution has zero values on the whole boundary.

Problem B. Consider again domain $D = (0, 1)^2$. where velocity is field given as in [7], $(\partial_y \theta, -\partial_x \theta)$, by the so called stream function

$$\theta = \theta(x, y) = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y).$$

The initial condition is defined by

$$\omega(x, y) = \sin(\pi x) \sin(\pi y),$$

and solution to (3.1) is given by

$$u(t, x, y) = -\exp(-t\pi^2) \omega(x, y).$$

We have the boundary condition 0 everywhere on the boundary.

The analytical solutions to the three problems are constructed by using methods of eigenfunctions [18], [16].

3.1.2 Deriving analytical solutions

In this subsection we explain how we arrived at the analytical solutions of the three examples forementioned.

Method of eigenfunctions

Our objective is a construction of solutions of IVP for PDE

$$u_t + \mathbf{v} \nabla u - u_{xx} = 0 \tag{3.2}$$

where $\mathbf{v}$ is a constant velocity field, $\mathbf{v} = (v_1, v_2)$. The method is known by the name "expansion of solution in terms of eigenfunctions". We use the following differential operators

$$\begin{aligned} H &= \partial_t + \rho^2 - \partial_x^2, \\ K &= \partial_t + v_1 \partial_x - \partial_x^2. \end{aligned}$$

One-dimensional eigenvalue problem

Let us consider the following eigenvalue problem

$$\left(-\partial_x^2 + \rho^2 \right) \phi = \lambda \phi, \qquad \phi(0) = \phi(1) = 0. \tag{3.3}$$

Solutions are

$$\phi_k(x) \;=\; 2^{1/2}\,\sin(k\pi x), \quad \lambda_k \;=\; \rho^2 \,+\, k^2\pi^2.$$

The functions ϕ_k make an orthonormal basis of $L_2(0,1)$. The orthogonality can be easily checked. Consider first the scalar product $(\phi_k \,—\, \phi_l)$ for $k \neq l$,

$$(\phi_k|\phi_l) \;=\; 2\int_0^1 \sin(k\pi x)\sin(l\pi x)dx$$

$$= 2\int_0^1 \frac{1}{2}\Big[\cos(k\pi x - l\pi x) - \cos(k\pi x + l\pi x)\Big]dx$$

$$= \Big|_0^1 \frac{1}{\pi(k-l)}\sin((k-l)\pi x) - \Big|_0^1 \frac{1}{\pi(k+l)}\sin((k+l)\pi x) \;=\; 0.$$

Now let's look at the case $k \,=\, l$.

$$(\phi_k|\phi_l) \;=\; 2\int_0^1 \sin(k\pi x)\sin(k\pi x)dx$$

$$= 2\int_0^1 \sin^2(k\pi x)dx = 2\int_0^1 \frac{1 \,-\, \cos(2k\pi x)}{2}dx = 2\Big|_0^1 \frac{1}{2}x + \frac{1}{4k\pi}\Big|_0^1 \sin(2k\pi x) \;=\; 1.$$

In the above calculations we used the trigonometry formulas for the product of two sin functions [21]. In terms of the inner product $(\cdot\,|\,\cdot)$ [10] this means $(\phi_k|\phi_l) = \delta_{kl}$.

It can be proved that any L_2-function can be approximated arbitrarily well by a finite linear combination of basis functions ϕ_k [16]. Therefore, "any" f on $(0,1)$ can be represented by the series

$$f(x) \;=\; \sum_k c_k\,\phi_k(x),$$

where $c_k = (\phi_k|f)$ are called Fourier coefficients. Equality of two sides holds point wise for sufficiently smooth f. Therefore, we try to represent a solution of $Hu(t,x) = 0$ as a series

$$u(t,x) \;=\; \sum_k \alpha_k(t)\,\phi_k(x),$$

where $\alpha_k(t)$ can be considered as the the Fourier coefficients of $u(t,\cdot)$. If we apply H from the left onto u, and demand $Hu = 0$ we get

$$\Big(\partial_t \,+\, \lambda_k\Big)\,\alpha_k(t) \;=\; 0,$$

giving us the only possibility $\alpha_k(t) \,=\, c_k \exp\big(\,-\,\lambda_k t\big)$ where $c_k \,\in\, \mathbb{R}$. In a step by step computation we derive this result in the following way

$$\partial_t u(t,x) = \sum_k \partial_t\alpha_k(t)\phi_k(x),$$

$$\rho^2 u(t,x) = \sum_k \rho^2\alpha_k(t)\phi_k(x),$$

$$-\,\partial_x^2 u(t,x) = -\sum_k \alpha_k(t)\partial_x^2\phi_k(x) = \sum_k \alpha_k(t)k^2\pi^2\phi_k(x),$$

$$Hu = \sum_k \Big[\partial_t \alpha_k(t) + \lambda_k \alpha_k(t)\Big]\phi_k(x) = 0.$$

This gives us an ordinary differential equation in $\alpha(t)$

$$\partial_t \alpha(t) + \lambda_k \alpha_k(t) = 0,$$

which has the solution $\alpha_k(t) = c_k \exp(-\lambda_k t)$.
Thus

$$u(t,x) \;=\; \sum_k c_k \exp\big(-\lambda_k\, t\big)\,\phi_k(x),$$

is a general solution of $Hu = 0$.

Next we consider an extension

$$\Big(-\partial_x^2 + v_1\,\partial_x\Big)\,\psi \;=\; \lambda\,\psi, \qquad \psi(0) \;=\; \psi(1) \;=\; 0. \tag{3.4}$$

and try to apply the same construction. Let us make an ansatz $\psi(x) = \exp(v_1 x/2)\phi(x)$. If we plug $\psi(x)$ into the formulated eigenvalue problem (3.4), we get the problem (3.3) for ϕ where $\rho^2 = v_1^2/4$. Hence, the eigenfunctions are $\psi_k(x) = \exp(v_1 x/2)\phi_k(x)$ and the corresponding eigenvalues are as before. Therefore, we now write u as a Fourier series of the new eigen-basis

$$u(t,x) = \sum_k \beta_k(t)\phi_k(x)\exp(\frac{v_1 x}{2}),$$

and try to solve $Ku = 0$.

$$\partial_t u(t,x) = \sum_k \partial_t \beta(t)\exp(\frac{v_1 x}{2})\phi_k(x),$$

$$\partial_x u(t,x) = \sum_k \beta_k(t)\Big[\frac{v_1}{2}\exp(\frac{v_1 x}{2})\phi_k(x) + \exp(\frac{v_1 x}{2})\phi_x'(x)\Big] = \sum_k \beta_k(t)\exp(\frac{v_1 x}{2})\Big[\frac{v_1}{2}\phi_k(x) + \phi_k'(x)\Big],$$

$$\partial_{xx} u(t,x) = \sum_k \beta_k(t)\frac{v_1}{2}\exp(\frac{v_1 x}{2})\Big[\frac{v_1}{2}\phi_k(x) + \phi_k'(x)\Big] + \beta_k(t)\exp(\frac{v_1 x}{2})\Big[\frac{v_1}{2}\phi_k'(x) + \phi_k''(x)\Big],$$

$$Ku = \sum_k \exp(\frac{v_1 x}{2})\Big[\partial_t \beta_k(t)\phi_k(x) + \beta_k(t)\Big(\frac{v_1^2}{4}\phi_k(x) - \phi_k''(x)\Big)\Big] = 0. \tag{3.5}$$

Since we know that $\phi_k'' = -\sqrt{2}k^2\pi^2\sin(k\pi x) = -k^2\pi^2\phi_k(x)$, the above calculation (3.5) can be written as

$$Ku = \sum_k \exp(\frac{v_1 x}{2})\Big[\partial_t \beta_k(t) + \beta_k(t)\Big(\frac{v_1^2}{4} + k^2\pi^2\Big)\Big]\phi_k(x) = 0.$$

If we identify $\frac{v_1^2}{4}$ as ρ^2, then the equation (3.5) can be rewritten as

$$Ku = \sum_k \exp(\frac{v_1 x}{2})\Big[\partial_t \beta_k(t) + \lambda_k \beta_k(t)\Big]\phi_k(x) = 0.$$

Similarily, as we did for $\alpha_k(t)$ we now have an ordinary differential equation in $\beta_k(t)$, with solution

$$\beta_k(t) = c_k \exp(-\lambda_k t),$$

and thus

$$u(t,x) = \sum_k \exp(-\lambda_k t)\exp(v_1 x/2)\phi_k(x). \tag{3.6}$$

The eigenfunctions $\psi_k(x)$ are not orthonormal. Yet they define a basis, because the function exp in front of ϕ_k is bounded from bellow and above by the numbers $\exp(\mp|v_1|/2)$. This gives us a possibility to built the above solutions of $Ku = 0$ in terms of ψ_k. In order to show that ψ_k are not orthonormal it is enough to show that either $\int_0^1 \psi_k^2\,dx \neq 1$, or $\int_0^1 \psi_k\psi_l\,dx \neq 0$. Here we show that $\int_0^1 \psi_k^2\,dx \neq 1$ in general. Again, for the integration of a product of exp and sin we consulted [21].

$$\begin{aligned}
(\psi_k|\psi_k) &= 2\int_0^1 \exp(v_1 x)\sin^2(k\pi x)dx \\
&= 2\int_0^1 \exp(v_1 x)\frac{1-\cos(2k\pi x)}{2}dx \\
&= \int_0^1 \exp(v_1 x) - \exp(v_1 x)\cos(2k\pi x)dx \\
&= \left\{\frac{1}{v_1}\exp(v_1 x)\Big|_0^1 - \left[\frac{\exp(v_1 x)}{v_1^2+4k^2\pi^2}(v_1\cos(2k\pi x)+2k\pi\sin(2k\pi x))\right]_0^1\right\} \\
&= \left\{\frac{1}{v_1}(\exp(v_1)-1) - \left[\frac{v_1(\exp(v_1)-1)}{v_1^2+4k^2\pi^2}\right]\right\}.
\end{aligned}$$

The construction of solutions in terms of the eigenfunctions ϕ_k of the boundary value problem (3.4) is general so it does not depend on a particular choice of the boundary conditions. As an alternative to using the boundary conditions of (3.4) one can impose zero derivatives at the end points of the interval (0,1). Also, zero value can be set for a subset of the end-points of the domain, while zero derivative on the remaining end-points. This would, for example, require eigenfunctions ϕ_k with zero derivative at $x = 0$ and zero value at $x = 1$. For this particular case the sequence of orthonormal eigenfunctions is

$$\phi_k(x) = \sqrt{2}\,\cos\Big(\frac{(2k+1)\pi x}{2}\Big), \quad k = 0, 1, \ldots$$

and the corresponding eigenvalues are

$$\lambda_k = \Big(\frac{(2k+1)\pi}{2}\Big)^2, \quad k = 0, 1, \ldots .$$

3.1.3 Two-dimensional eigenvalue problems

To solve the 2-D problem (3.2) we utilize the same technique as for one-dimensional problems. Thus we start with

$$u(t,x,y) = \sum_{k=1}^{\infty} f_k(t,y)\,\exp(v_1 x/2)\,\phi_k(x),$$

where $f_k(t,y)$ are the new Fourier coefficients, and apply from the left the differential operator

$$M \;=\; K \,+\, v_2\partial_y \;=\; \partial_t \,+\, v_1\partial_x \,+\, v_2\partial_y \,-\, \partial_x^2$$

in order to calculate $Mu = 0$. The following result is obtained:

$$\begin{aligned} M\,u(t,x,y) \;&=\; \Big(\partial_t \,+\, v_2\partial_y\Big)\,u(t,x,y) \,+\, \Big(v_1\partial_x \,-\, \partial_x^2\Big)\,u(t,x,y) \\ &=\; \Big(\partial_t \,+\, v_2\partial_y\Big)\,u(t,x,y) \,+\, \lambda_k\,u(t,x,y) \qquad\qquad = 0 \\ &=\; \textstyle\sum_k \Big(\partial_t \,+\, v_2\partial_y \,+\, \lambda_k\Big)\;\; f_k(t,y)\,\exp(v_1 x/2)\,\phi_k(x) \end{aligned}$$

implying

$$\Big(\partial_t \,+\, v_2\partial_y \,+\, \lambda_k\Big)\,f_k(t,y) \;=\; 0.$$

The only non-trivial solution of this problem is

$$f_k(t,y) \;=\; \exp\big(-\lambda_k\,t\big)\,g_k(y - tv_2),$$

where g_k is a function on $\mathbb{R}$. Hence,

$$u(t,x,y) \;=\; \sum_{k=1}^{\infty} \exp\big(-\lambda_k\,t\big)\,g_k(y - tv_2)\;\exp(v_1 x/2)\,\phi_k(x). \tag{3.7}$$

Apparently, the constructed solution is on the set $[0,\infty)\times(0,1)\times\mathbb{R}$.

3.1.4 Solutions on squares

If all the functions g_k have supports in a set $[0,a]\subset[0,1]$ then for times $0 \le t \le (1-a)/v_2$ the packages g_k travel over the domain $(0,1)^2$ from their initial supports towards 1 and eventully hit the right end-point at $T = (1-a)/v_2$. Within $t\in[0,T]$ solution (3.7) has zero values at the boundary of $[0,1]^2$.

An example function is

$$u(t,x,y) \;=\; \exp\big(-\lambda_k\,t\big)\,\chi(y - v_2 t)\;\exp(v_1 x/2)\,\phi_k(x),$$

where

$$\chi(x) \;=\; \begin{cases} \sin(2\pi x) & \text{for} \quad x\in[0,1/2] \\ 0 & \text{for} \quad x\notin[0,1/2]. \end{cases}$$

Therefore, when executing the code for the constant velocity field for Degenerate heat equation the running time of the simulation can be set to at most 0.5.

3.1.5 Solution with the vortex velocity field

We consider an IVP for the second order PDE of the following general form

$$\begin{array}{lll} \partial_t\,u(t,\boldsymbol{x}) + \boldsymbol{v}(\boldsymbol{x})\nabla u(t,\boldsymbol{x}) + A\,u(t,\boldsymbol{x}) = 0 & \text{for} \quad t\ge 0, \quad \boldsymbol{x}\in D, \\ u(t,\cdot)|\partial D \;=\; 0 & \text{for} \quad t\ge 0, \\ u(0,\boldsymbol{x}) \;=\; u_0(\boldsymbol{x}) & \text{for} \quad \boldsymbol{x}\in D, \end{array} \tag{3.8}$$

where $D \subset \mathbb{R}^2$ is a bounded domain with the boundary ∂D, $\boldsymbol{x} \mapsto \boldsymbol{v}(\boldsymbol{x})$ is a divergence free velocity field on D, u_0 is a function on D and A is a second order differential operator (degenerate or non-degenetate) of the following form

$$A \;=\; -\sum_{ij=1}^{2} a_{ij}\,\partial_i\,\partial_j.$$

with a constant diffusion tensor a_{ij}.

One of possibilites for the construction of solutions $u(t,\boldsymbol{x})$ in terms of known functions is given by the following two rules

1) Solution has the general form $u(t,\boldsymbol{x}) = f_1(t)f_2(\boldsymbol{x})$.

2) The function f_2 has the property $\boldsymbol{v}(\boldsymbol{x})\nabla f_2(\boldsymbol{x}) = 0$ and $Af_2(\boldsymbol{x}) = \lambda f_2(\boldsymbol{x})$.

Let us point out that $Af_2 = \lambda f_2$ means that f_2 is an eigenfunction of A with the corresponding eigenvalue λ.

When one inserts the assumed form of the solution into the PDE the following result is obtained

$$\begin{aligned} \partial_t\, f_1(t) \;+\; \lambda\, f_1(t) \;&=\; 0, \\ u_0(\boldsymbol{x}) \;&=\; f_2(\boldsymbol{x}). \end{aligned}$$

Thus, $f_1(t) = \exp(-\lambda t)$, the function f_2 must be an eigenfunction of A and the demanded equality $\boldsymbol{v}\nabla f_2 = 0$ is ensured by choosing the divergence free velocity field of the form

$$v_1(\boldsymbol{x}) \;=\; \partial_2\, f_2(\boldsymbol{x}), \qquad v_2(\boldsymbol{x}) \;=\; -\partial_1\, f_2(\boldsymbol{x}).$$

We can conclude this brief description by the following conclusion. One has to construct explicitly in closed form an eigenfunction of the problem

$$\begin{aligned} -\textstyle\sum_{ij=1}^{2}\, a_{ij}\,\partial_i\,\partial_j\,\phi(\boldsymbol{x}) \;&=\; \lambda\,\phi(\boldsymbol{x}), \\ \phi(\boldsymbol{x}) \;&=\; 0, \quad \text{for } \boldsymbol{x} \in \partial D. \end{aligned}$$

For simpler domains such as a square or rectangle and diagonal diffusion tensors a_{ij} this problem can be easily solved. Let us consider the case $D = (0,1)^2$. Any such eigenfunction is the product of two functions, $\phi(\boldsymbol{x}) = \phi_1(x_1)\phi_2(x_2)$, and the functions ϕ_i must have zero values at 0 and 1. In addition there must hold $\partial^2\phi = \text{number} \times \phi$. Only sin functions satisfy these conditions.

Now one can easily utilize this procedure. Given vortex velocity field $\mathbf{v} = (-\partial_2\psi, \partial_1\psi)$, where

$$\psi(x,y) = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y),$$

a solution to (3.8) is the following function

$$u(t,x,y) \;=\; \exp(-\pi^2 t)\sin(\pi x)\sin(\pi y).$$

The following simple fact is used in the constuction of a solution to (3.8). If $u(t,x,y) = \exp(-\lambda t)\chi(x,y)$ is a solution to $Af_2 = \lambda f_2$ defined in 2) then we have $\boldsymbol{v}\nabla u = 0$ for any velocity field $\boldsymbol{v} = (\partial_2\psi, -\partial_1\psi)$ where the stream function is a polynomial of the function χ. In our case we chose the second order power of χ, i.e. $\psi = \chi^2/\pi$.

3.1.6 Findings

The list of programs implementing Degenerate heat equation can be found in Appendix. For a detailed explanation of the m language implementation the user can consult the manual of the Toolbox of Level Set Methods 1.1 and visit Section 5.1 of this thesis report. The figures pertaining to our findings are included at the end of this section.

The CFL bound computed in the C implementation is the theoretical bound as discussed in Section 2.3. Therefore, we use C implementation to generate errors and figures of analytical and numerical solutions at different time points of the time interval $[0, T]$.

For the constant velocity field examples, the final time T, has to be $\leq 1/2$ in order for the boundary conditions to be satisfied. Note that for Problem B the final time can be any in $\mathbb{R}^+$.

For both problems the number of nodes taken in each spatial direction is 100, and two sets of four images display the analytical and the numerical solutions at different times.

Figures 3.1 and 3.2 show the results of the implementation of Problem A while Figures 3.3 and 3.4 show the results of the implementation of Problem B.

The errors for the two problems, A and B, are calculated in the l_∞ norm and are displayed to the console for a user-defined number of time points of the time interval $[0, T]$. Each line of the program's output contains three numbers. The first number is the time, the second number is the absolute error, and the third number is the relative error.

```
For Problem A errors are
maximum error at time 0.000000 is 0.000000 (in %) 0.00000000
maximum error at time 0.166668 is 0.011709 (in %) 8.57906057
maximum error at time 0.333336 is 0.002948 (in %) 11.19121327
maximum error at time 0.500004 is 0.000665 (in %) 13.08010965
For the vortex velocity case errors are
maximum error at time 0.000000 is 0.000000 (in %) 0.00000596
maximum error at time 0.166663 is 0.000349 (in %) 0.18079901
maximum error at time 0.333327 is 0.000139 (in %) 0.37364951
maximum error at time 0.499990 is 0.000041 (in %) 0.56455006
```

The solutions go quickly to 0 as time increases, since they contain $\exp(-t\pi^2)$ term. The time step taken is 1e−5 using CFL number 0.8, and the initial conditions for the two problems are their respective solutions at time $t = 0$.

Numerical errors shown above, and illustrated by graphs, enable us to say that the monotone scheme of Degenerate heat equation appears to be better suited for an approximation of problems with the vortex velocity field than the constant velocity field. The error of 13% for Problem A seems rather large. Looking at this figure alone we could doubt the quality of accuracy of monotone schemes. Especially since taking a smaller time step produces almost the same error. Still, the error of 0.56% of Problem B leaves us in the need for further investigation of their accuracy.

What can be said with certainty is that the maximal time step calculated from the τ_{CFL} bound produces a stable method in each of these three problems.

Figure 3.1: Analytical solution of Problem A computed for the final time 0.5 , boundary condition 0, velocity $\mathbf{v} = (0, 1)$, grid size $[100 \times 100]$, and initial condition $\sin(2\pi y)\sin(\pi x)/\sqrt{2}\ \forall x \in$ domain D, and $\forall y \in [0, 1/2]$.

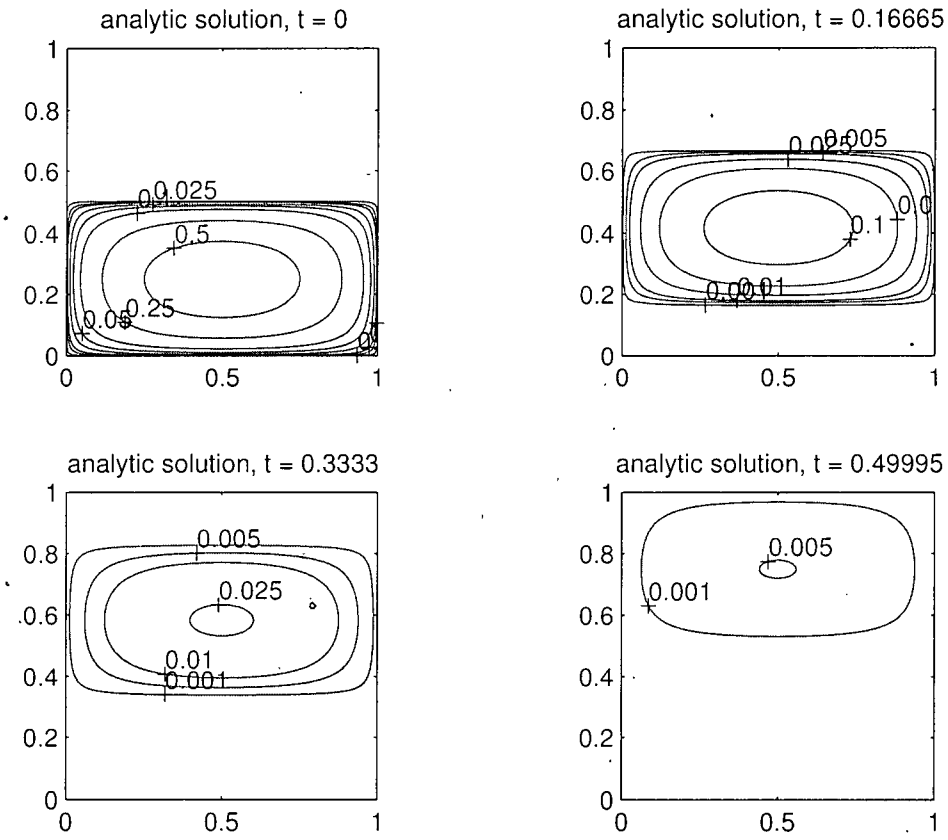


Figure 3.2: Numerical solution of Problem A computed for the final time 0.5 boundary condition 0, velocity $\mathbf{v} = (0, 1)$, grid size $[100 \times 100]$, and initial condition $\sin(2\pi y)\sin(\pi x)/\sqrt{2}$ $\forall x \in$ domain D, and $\forall y \in [0, 1/2]$.

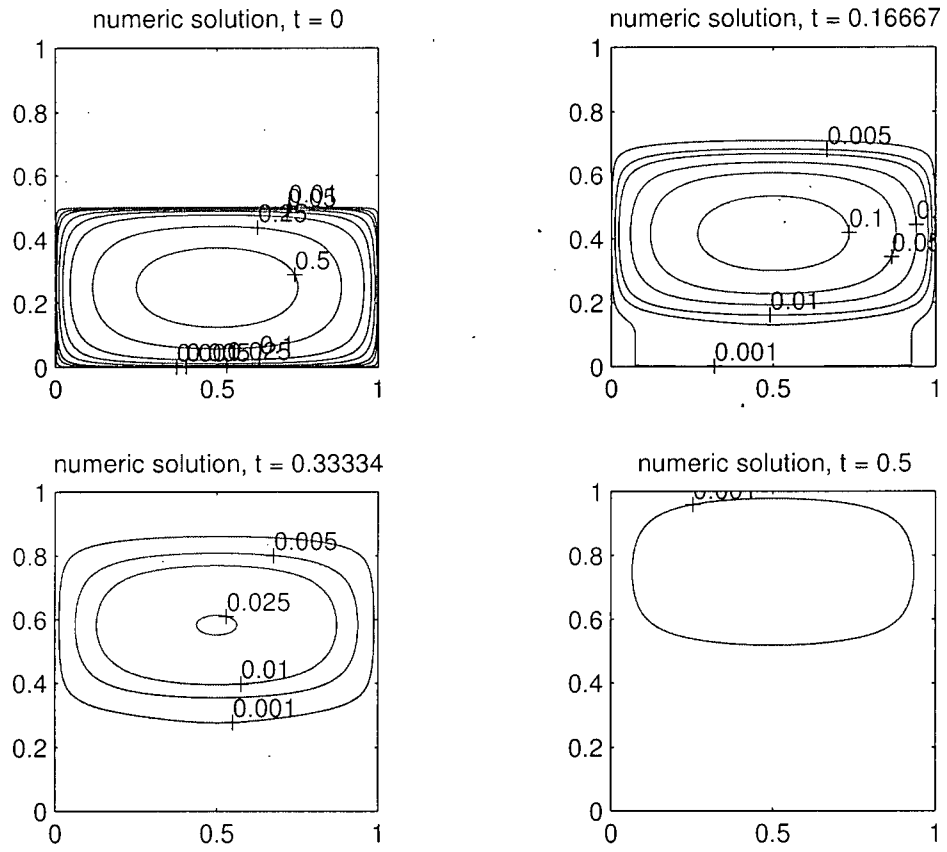


Figure 3.3: Analytical solution of Problem B computed for the final time 0.5 , boundary condition 0, vortex velocity field, grid size [100 × 100], and initial condition $\sin(\pi x)\sin(\pi y)$ $\forall x, y \in$ domain D.

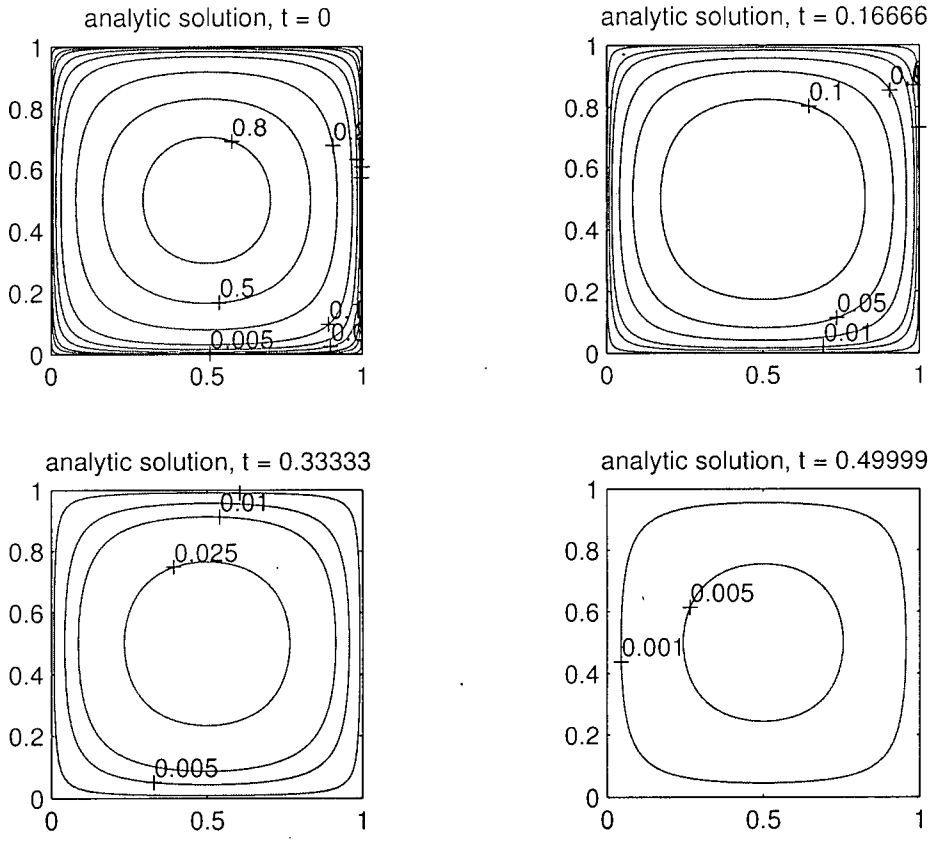
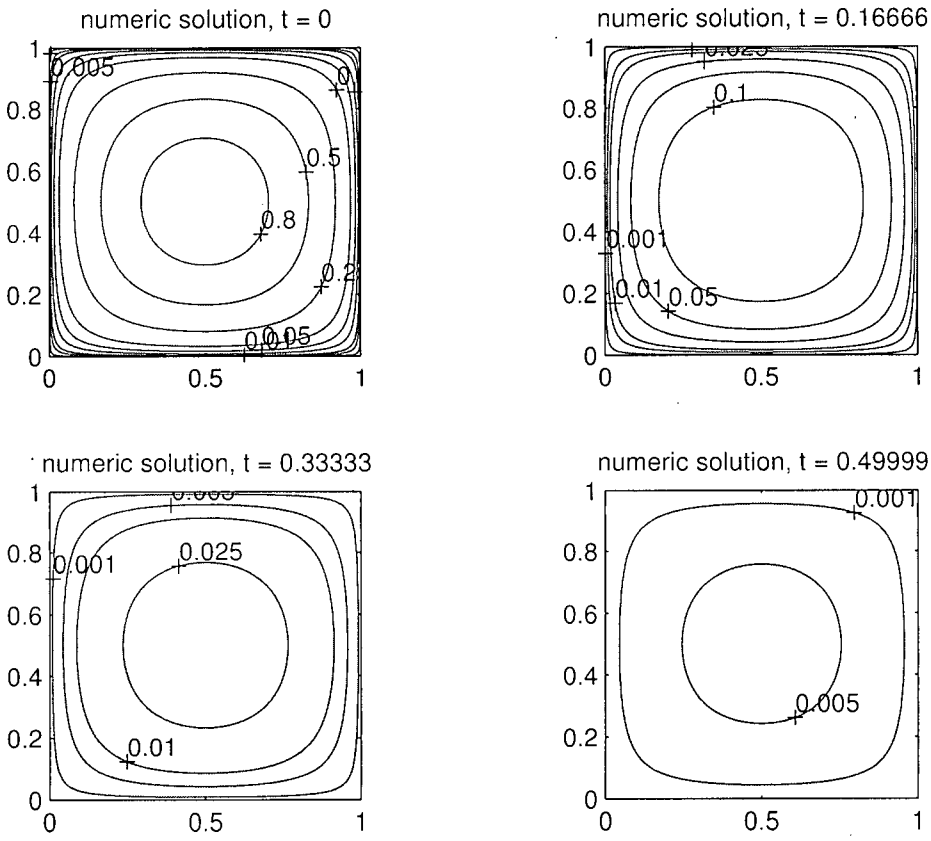


Figure 3.4: Numerical solution of Problem B computed for the final time 0.5 boundary condition 0, vortex velocity field, grid size $[100 \times 100]$, and initial condition $\sin(\pi x)\sin(\pi x)\ \forall x, y \in$ domain D.



3.2 General degenerate parabolic equation

Next we consider the third equation of interest,

$$u_t(\mathbf{x}) - u_{xx}(\mathbf{x}) - 2u_{xy}(\mathbf{x}) - u_{xx}(\mathbf{x}) = 0. \tag{3.9}$$

This parabolic equation was obtained by adding time dependency to a particular example,

$$-\left(u_{xx} + 2u_{xy} + u_{yy}\right) = -\frac{d^2u}{dv^2} = 0, \quad v = (1,1) \tag{3.10}$$

of [2].

The example (3.10) was designed to show how choice of a difference scheme has an impact on the convergence of a method. The author of [2] demonstrates that the difference scheme

$$\frac{1}{h^2}(u(x+h, y+h) - 2u(x,y) + u(x-h, y-h)) \tag{3.11}$$

produces a convergent method of (3.10), while the scheme

$$\begin{array}{c} \frac{1}{h^2}(2u(x+h,y) + 2u(x,y+h) + 2u(x-h,y) + \\ 2u(x,y-h) - 6u(x,y) - u(x+h,y-h) - u(x-h,y+h)) \end{array} \tag{3.12}$$

results in a non-convergent method. Both schemes are central difference schemes. The first scheme takes central differences in the direction $(1,1)$, while the second in the direction $(-1,1)$.

Oberman in [2] suggests an additional way to derive a convergent scheme for the above problem via coordinate axis rotation. This is an alternative, less elegant than central differences.

3.2.1 Goal and problem setting

Our goal is to confirm or contradict the findings of [2] by analysing associated parabolic equation (3.9). The findings of [2] state that the direction in which the central differences are taken has impact on the convergence of a method. Again, we created implementations in both C programming and MATLAB's m languages. The m language implementation is a routine in the Toolbox of Level Set Methods 1.1 and therefore can not be run as a stand-alone program.

The starting point is construction of an analytical solution. It is easily obtainable from the boundary conditions given in [2]. Therefore, we can now postulate the initial boundary value problem for General degenerate parabolic equation.

Consider PDE (3.9) on the domain $D = (0,1)^2$, with initial condition

$$u(0,x,y) = 0,$$

in the interior of the domain, and boundary condition

$$\sin(6\pi\,(x-y)) \quad \forall (x,y) \in \partial_D,\, t > 0$$

Apparently, the approximation converges to the classical solution

$$u(t,x,y) = \sin(6\pi\,(x-y))$$

as $t \to \infty$.

We must bring into the correspondence the schemes (3.11) and (3.12) with the schemes of Section 2.2.1.

Recall that (3.9) coincides with General degenerate parabolic equation, where the degenerate differential operator is $A(\mathbf{x}) = -\sum_{i,j=1}^{d} a_{ij}(\mathbf{x})\partial_i\partial_j$, and the elliptic diffusion tensor is $a_{ij} = 1$ for all $i,j \in \{1,2\}$.

In accordance with the proposed discretizations of Section 2.2.1 we have at our disposal four possibilites for $\partial_1\partial_2$, defined by

$$\partial_{ij}u \;=\; \begin{cases} \frac{1}{2h^2}u(\mathbf{x}\pm\mathbf{e}_ih\pm\mathbf{e}_jh) - u(\mathbf{x}\pm\mathbf{e}_ih) - u(\mathbf{x}\pm\mathbf{e}_jh) + u(\mathbf{x}), \\ \frac{1}{2h^2} - u(\mathbf{x}\pm\mathbf{e}_ih\mp\mathbf{e}_jh) + u(\mathbf{x}\pm\mathbf{e}_ih) + u(\mathbf{x}\mp\mathbf{e}_jh) - u(\mathbf{x}). \end{cases}$$

Due to $a_{12} > 0$ we have to use the first possibility,

$$\frac{1}{2h^2}\Big[u(\mathbf{x}\pm\mathbf{e}_ih\pm\mathbf{e}_jh) - u(\mathbf{x}\pm\mathbf{e}_ih) - u(\mathbf{x}\pm\mathbf{e}_jh) + u(\mathbf{x})\Big],$$

to arrive at a converging method. When we apply the prescribed discretization to the differential operator $\sum_{i,j} a_{ij}\partial_i\partial_j$ we get

$$\begin{aligned}
&\partial_{ii}u(\mathbf{x}) \;+\; 2\,\partial_{ij}u(\mathbf{x}) + \partial_{jj}u(\mathbf{x}) = \\
&= \tfrac{1}{h^2}\Big[u(\mathbf{x}+\mathbf{e}_ih) - 2u(\mathbf{x}) \;+ u(\mathbf{x}-\mathbf{e}_ih)\Big] + \\
&+ 2\,\tfrac{1}{2h^2}\Big[u(\mathbf{x}+\mathbf{e}_ih+\mathbf{e}_jh) \;+\; u(\mathbf{x}-\mathbf{e}_ih-\mathbf{e}_jh) - u(\mathbf{x}+\mathbf{e}_ih) \\
&-u(\mathbf{x}-\mathbf{e}_ih) \;-\; u(\mathbf{x}+\mathbf{e}_jh) - u(\mathbf{x}-\mathbf{e}_jh) \;+\; 2u(\mathbf{x})\Big] + \\
&+\tfrac{1}{h^2}\Big[u(\mathbf{x}+\mathbf{e}_jh) - 2u(\mathbf{x}) + u(\mathbf{x}-\mathbf{e}_jh)\Big] = \\
&= \tfrac{1}{h}\Big[u(\mathbf{x}+\mathbf{e}_ih+\mathbf{e}_jh) \;-\; 2\,u(\mathbf{x}) \;+\; u(\mathbf{x}-\mathbf{e}_ih-\mathbf{e}_jh)\Big].
\end{aligned}$$

This is precisely the convergent scheme (3.11).

If we use intentionally the second possibility to discretize the operator $\partial_i\partial_j$, $i \neq j$, we get a scheme which is not monotone

$$\begin{aligned}
&\partial_{ii}u(\mathbf{x}) \;+\; 2\,\partial_{ij}u(\mathbf{x}) + \partial_{jj}u(\mathbf{x}) = \\
&= \tfrac{1}{h^2}\Big[u(\mathbf{x}+\mathbf{e}_ih) - 2u(\mathbf{x}) \;+ u(\mathbf{x}-\mathbf{e}_ih)\Big] + \\
&+ 2\,\tfrac{1}{2h^2}\Big[-u(\mathbf{x}+\mathbf{e}_ih-\mathbf{e}_jh) \;-\; u(\mathbf{x}-\mathbf{e}_ih+\mathbf{e}_jh) \;+\; u(\mathbf{x}+\mathbf{e}_ih) \\
&+u(\mathbf{x}-\mathbf{e}_ih) \;+\; u(\mathbf{x}-\mathbf{e}_jh) + u(\mathbf{x}+\mathbf{e}_jh) \;-\; 2u(\mathbf{x})\Big] + \\
&+\tfrac{1}{h^2}\Big[u(\mathbf{x}+\mathbf{e}_jh) - 2u(\mathbf{x}) + u(\mathbf{x}-\mathbf{e}_jh)\Big] = \\
&= \tfrac{1}{h}\Big[2\,u(\mathbf{x}+\mathbf{e}_ih) + 2\,u(\mathbf{x}-\mathbf{e}_ih) + 2\,u(\mathbf{x}+\mathbf{e}_jh) + 2\,u(\mathbf{x}-\mathbf{e}_jh) + \\
&+6\,u(\mathbf{x}) \;+\; -u(\mathbf{x}+\mathbf{e}_ih-\mathbf{e}_jh) \;-\; u(\mathbf{x}-\mathbf{e}_ih+\mathbf{e}_jh)\Big].
\end{aligned}$$

Actually this is the non-convergent scheme (3.12). The absence of monotonicity results in the method being non-convergent. For further discussion on monotonicity see the next section.

The code in the Toolbox of Level Set Methods 1.1 handles a completely separate discretization of mixed partial derivatives, discussed in Section 2.2.2. It applies operator $\boxminus_i f$ first in the variable i and then in the variable j. The resulting approximation operator is

$$\frac{1}{4h^2}\Big(u(\mathbf{x}+\mathbf{e}_i h+\mathbf{e}_j h)-u(\mathbf{x}-\mathbf{e}_i h+\mathbf{e}_j h)-u(\mathbf{x}+\mathbf{e}_i h-\mathbf{e}_j h)+u(\mathbf{x}-\mathbf{e}_i h-\mathbf{e}_j h)\Big)$$

Using this operator the discretization of the considered $\sum_{ij} a_{ij}\partial_i\partial_j$ is

$$\begin{array}{l}\boxminus_{ii}u(\mathbf{x})\ +\ 2\,\boxminus_{ij}u(\mathbf{x})+\boxminus_{jj}u(\mathbf{x})= \\ =\frac{1}{h^2}\Big[u(\mathbf{x}+\mathbf{e}_i h)-2u(\mathbf{x})\ +u(\mathbf{x}-\mathbf{e}_i h)\Big]+ \\ \frac{1}{4h^2}\Big(u(\mathbf{x}+\mathbf{e}_i h+\mathbf{e}_j h)-u(\mathbf{x}-\mathbf{e}_i h+\mathbf{e}_j h)- \\ -u(\mathbf{x}+\mathbf{e}_i h-\mathbf{e}_j h)+u(\mathbf{x}-\mathbf{e}_i h-\mathbf{e}_j h)\Big)+ \\ +\frac{1}{h^2}\Big[u(\mathbf{x}+\mathbf{e}_j h)-2u(\mathbf{x})+u(\mathbf{x}-\mathbf{e}_j h)\Big].\end{array} \tag{3.13}$$

It is easy to check that the resulting scheme is not monotone.

3.2.2 Monotonicity of the schemes

In this section we consider monotonicity of the schemes (3.11), (3.12), and (3.13) and show that only (3.11) is monotone, while (3.12) and (3.13) are not monotone. The analysed elliptic problem is defined by (3.10). For the definition of monotonicity refer to Section 2.1.

We first consider scheme (3.11) to approximate the equation (3.10). In order to check the monotonicity of the approximation we write $u_{i,j}$ as a function of its neighbouring nodes, i.e. its arguments, $u_{i,j}=f(u_{l,k}\mid u_{l,k}\in N(u_{i,j}))$,

$$2u_{i,j}=u_{i+1,j+1}+u_{i-1,j-1}.$$

Since both coefficients next to $u_{i+1,j+1}$ and $u_{i-1,j-1}$ are positive, $u_{i,j}$ is nondecreasing in each of its arguments, and therefore scheme (3.11) is monotone.

Next we write the scheme (3.12) in the same form

$$6u_{i,j}=2u_{i+1,j}+2u_{i,j+1}+2u_{i-1,j}+2u_{i,j-1}-u_{i+1,j-1}-u_{i-1,j+1}.$$

Since $u_{i,j}$ is decreasing in $u_{i+1,j-1}$ and $u_{i-1,j+1}$, the scheme is not monotone.

Finally, we look at (3.13)

$$16u_{i,j}=4u_{i+1,j}+4u_{i-1,j}+u_{i+1,j+1}-u_{i-1,j+1}-u_{i+1,j-1}+u_{i-1,j-1}+4u_{i,j+1}+4u_{i,j-1}.$$

This scheme is non monotone since $u_{i,j}$ is decreasing in $u_{i-1,j+1}$ and $u_{i+1,j-1}$.

The following can be said for the mixed partial derivative approximations implemented in the Toolbox of Level Set Methods 1.1. Any scheme implemented in the Toolbox

which uses Toolbox's mixed partial derivative approximation is not guaranteed to be monotone.

This comes from the fact that schemes examined are built from schemes of individual terms. For example, Heat equation is build from two individual schemes. One for the u_{xx} term, and one for u_{yy} term each of which happen to be monotone. Therefore, the scheme for Heat equation which adds these two monotone schemes is monotone.

Let us now examine the individual schemes existing in the Toolbox of Level Set Methods 1.1. The Toolbox of Level Set Methods uses upwinding for the approximation of first order derivatives, which is a monotone approximation. It uses the differences approximation operator, $\exists_{ii}$, for the non-mixed second order term, which is also monotone. Yet, it uses the following expression

$$2a_{12}\,\frac{1}{4h^2}(u_{i+1,j+1} - u_{i-1,j+1} - u_{i+1,j-1} + u_{i-1,j-1})$$

for the approximation of the mixed partial derivatives for the diffusion tensor $a = \{a_{ij}\}_{11}^{22}$. As readily visible, depending on the sign of a_{12} two of the terms always have positive values, while the other two always have negative values. Therefore, once this approximation is combined with monotone approximations for upwinding and $\exists_{ii}$, the result is a non-monotone approximation.

3.2.3 Rotation of coordinate axes for General degenerate parabolic equation

As mentioned in Section 2.2.4 a way to obtain a monotone scheme would be to perform a coordinate axes rotation of a problem [4]. Here we consider this approach.

Let us recall the definition of the elliptic differential operator for a two dimensional problem

$$A \;=\; -\sum_{ij=1}^{2} a_{ij}\,\partial_i\,\partial_j, \tag{3.14}$$

where the diffusion tensor $a = \{a_{ij}\}_{11}^{22}$ is positive semidefinite. Thus, either it is definite and has two real positive eigenvalues or it is semidefinite with one eigenvalue equal to zero implying $det(a) = 0$. In the original coordinates, which are denoted here by x_1, x_2, the partial differentiation operators are denoted by ∂_i.

We define new coordinates, x_1', x_2' as

$$\begin{aligned} x_1' &= t_{11}\,x_1 + t_{12}\,x_2, \\ x_2' &= t_{21}\,x_1 + t_{22}\,x_2, \end{aligned}$$

and the corresponding partial differentiation operators ∂_i'. The following transformation between differentations in the original and new variables can be established

$$\begin{aligned} \partial_1 &= t_{11}\,\partial_1' + t_{21}\,\partial_2', \\ \partial_2 &= t_{12}\,\partial_1' + t_{22}\,\partial_2'. \end{aligned}$$

This can be rewritten in matrix form

$$\begin{bmatrix} \partial_1 \\ \partial_2 \end{bmatrix} = T^T \begin{bmatrix} \partial'_1 \\ \partial'_2 \end{bmatrix}.$$

To obtain the elliptic operator A in new coordinates, we have to use these transformations of partial differential operators. Therefore, we insert expressions ∂_i in terms of ∂'_i into (3.14) to obtain

$$A = -\sum_{ijkl=1}^{2} a_{ij}\, t_{ki}\, \partial'_k\, t_{lj}\, \partial'_l = -\sum_{kl=1}^{2} a'_{kl}\, \partial'_k\, \partial'_l,$$

where the new diffusion tensor has entries

$$a'_{kl} = \sum_{ij} t_{ki}\, a_{ij}\, t_{lj} = (TaT^T)_{kl}.$$

In the case of $det(a) = 0$ we have $det(a') = det(T)det(a)det(T^T) = 0$, i.e. A is degenerate in the new coordinates as well. We know from Strang [10] that there exists a rotation defined by

$$T(\alpha) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix},$$

such that the matrix a' has one of the following forms:

$$\begin{bmatrix} \rho & 0 \\ 0 & 0 \end{bmatrix}, \qquad \begin{bmatrix} 0 & 0 \\ 0 & \rho \end{bmatrix}.$$

For non-degenarate diffusion tensors there exist difference schemes which discretize the elliptic differential operator (3.14) by matrices A_n of positive type. The idea is given in [4]. There are many ellaborations such as [20]. In order to get discretizations A_n of positive type one has to use stencils of different length in the x_1 and x_2 directions. This idea we use for degenerate schemes to develop the method of Chapter 4.

3.2.4 Findings

The CFL bound on the schemes which are monotone was computed as given in Section 2.3. The CFL bound for the non-monotone schemes cannot be computed using our described procedure. Still, we used the same CFL bound as the one for monotone schemes, namely

$$\frac{h^2}{2\, \max_{kl} \left[a_{11}(\boldsymbol{x}_{kl}) + a_{22}(\boldsymbol{x}_{kl}) - |a_{12}(\boldsymbol{x}_{kl})| \right]}.$$

This time step when used with non-monotone schemes produces erroneous results which show non-convergence of these schemes.

We must note that for parabolic problems with the zero initial condition and nonhomogeneous boundary conditions we do not show error values. Instead we show the difference between the numerical solution and the analytical solution at a point in time. In addition,

when we say that a solution is analytical, in some cases it is really an asymptotic solution. These cases are the ones for which the initial condition is not equal to the analytical solution at time zero.

Figure 3.5 shows the result of using discretization (3.11) when running the implementation up to time 0.2 for grid size 100×100. The solution driven by the boundary conditions is quickly approaching the analytic solution as can be seen in the figure. The differences in l_∞ norm are

```
maximum difference at time 0.000000 is 0.998867 (in %) 100.00000000
maximum difference at time 0.024997 is 0.919144 (in %) 92.01866571
maximum difference at time 0.049995 is 0.706837 (in %) 70.76385770
maximum difference at time 0.074992 is 0.529321 (in %) 52.99214182
maximum difference at time 0.099990 is 0.395380 (in %) 39.58280635
maximum difference at time 0.124987 is 0.296519 (in %) 29.68551002
maximum difference at time 0.149985 is 0.222830 (in %) 22.30829146
maximum difference at time 0.174982 is 0.167454 (in %) 16.76441874
maximum difference at time 0.199980 is 0.125842 (in %) 12.59842562
```

Figure 3.6 shows the result at time 1.0 for both the analytic and numeric solutions. The differences are

```
maximum difference at time 0.000000 is 0.998867 (in %) 100.00000000
maximum difference at time 1.000026 is 0.000135 (in %) 0.01350979
```

An additional test was run with the initial condition being the solution in order to assert the validity of the implementation, and thus potentially observe the calculational error on a 100×100 grid with final time 0.5.

```
maximum error at time 0.000000 is 0.000000 (in %) 0.00000000
maximum error at time 0.124987 is 0.000000 (in %) 0.00004028
maximum error at time 0.249974 is 0.000000 (in %) 0.00004028
maximum error at time 0.374962 is 0.000000 (in %) 0.00004028
maximum error at time 0.499949 is 0.000000 (in %) 0.00004028
maximum error at time 0.624936 is 0.000000 (in %) 0.00004028
maximum error at time 0.749923 is 0.000000 (in %) 0.00004028
maximum error at time 0.874911 is 0.000000 (in %) 0.00004028
maximum error at time 0.999898 is 0.000000 (in %) 0.00004028
```

We conclude that the 4e−5 percent error is due to round off error.

Figure 3.7 shows the same setting as in Figure 3.5 using discretization (3.12) on a 100×100 nodes grid for final time 1.2. It is readily visible how the non-convergence is materialized and remains with the running time. The differences obtained are

```
maximum difference at time 0.150000 is 0.826923 (in %) 82.7861089311
```

```
maximum difference at time 0.300000 is 0.906501 (in %) 90.7528988305
maximum difference at time 0.450000 is 0.900932 (in %) 90.1953603932
maximum difference at time 0.600000 is 0.861522 (in %) 86.2498673530
maximum difference at time 0.750000 is 0.665061 (in %) 66.5815557683
maximum difference at time 0.900000 is 0.946972 (in %) 94.8045391399
maximum difference at time 1.050000 is 1.081382 (in %) 108.2608615755
maximum difference at time 1.200000 is 0.941579 (in %) 94.2646321863
```

When compared to the results of discretiazation (3.11) it is plain that discretization (3.11) produces a convergent method, while discretization (3.12) doesn't.

Figure 3.8 shows the differences obtained by using the discretization (3.13). The non-convergence happens due to the scheme being non-monotone. The differences for a grid with 100×100 nodes and final time 1.2 are

```
maximum difference at time 0.150000 is 0.492477 (in %) 49.3035721126
maximum difference at time 0.300000 is 0.456759 (in %) 45.7276746742
maximum difference at time 0.450000 is 0.454310 (in %) 45.4825515735
maximum difference at time 0.600000 is 0.451359 (in %) 45.1870767231
maximum difference at time 0.750000 is 0.450665 (in %) 45.1176407953
maximum difference at time 0.900000 is 0.461395 (in %) 46.1918515911
maximum difference at time 1.050000 is 0.465639 (in %) 46.6167256667
maximum difference at time 1.200000 is 0.546891 (in %) 54.7510753188
```

Let us point out once more that schemes (3.12) and (3.13), which are not monotone, result in a non-convergence, while the monotone scheme (3.11) is convergent.

We also investigated what would happen on a grid with periodic boundary conditions when domain is $(-1,1)^2$ using the Toolbox of Level Set Methods 1.1 implementation of the boundary calculation, `addGhostPeriodic.m`. The initial condition is $\sin(6\pi(x-y))$ everywhere within the domain, and the central differences approximation is (3.11). The result for the final time 1.2 on a grid with 100×100 nodes using the discretization (3.11) surprisingly produces a non-convergent scheme. This can be seen from the percentage errors computed at nine different time points from 0 to 1.2

```
maximum error at time 0.150000 is 0.521210 (in %) 52.1801004934
maximum error at time 0.300000 is 0.514805 (in %) 51.5388485854
maximum error at time 0.450000 is 0.376802 (in %) 37.7229136800
maximum error at time 0.600000 is 0.377520 (in %) 37.7948549638
maximum error at time 0.750000 is 0.428266 (in %) 42.8751591476
maximum error at time 0.900000 is 0.522602 (in %) 52.3194115335
maximum error at time 1.050000 is 0.395414 (in %) 39.5862203809
maximum error at time 1.200000 is 0.555725 (in %) 55.6355194335
```

Figure 3.9 shows the above example at four time points.

Further on, we tested the discretization (3.13) with the periodic boundary condi-

tions of the Toolbox of Level Set Methods 1.1. For a grid with 100×100 nodes, and final time 0.2, the result is shown in Figure 3.10. The errors generated are

```
maximum error at time 0.025000 is 0.565357 (in %) 56.5998184766
maximum error at time 0.050000 is 0.961642 (in %) 96.2732027284
maximum error at time 0.075000 is 0.952116 (in %) 95.3195222936
maximum error at time 0.100000 is 1.004376 (in %) 100.5514906582
maximum error at time 0.125000 is 1.256236 (in %) 125.7660333185
maximum error at time 0.150000 is 1.016355 (in %) 101.7507094230
maximum error at time 0.175000 is 1.133993 (in %) 113.5278499173
maximum error at time 0.200000 is 1.076474 (in %) 107.7694694336
```

We conducted a test with boundary conditions $\sin(6\pi(x+y))$ on a grid of 100×100 nodes for final time 1.0 with discretizations (3.11) and (3.12). We expected that if we were to use discretization (3.11) we would obtain a non-convergent method, while if we were to use discretization (3.12) the method would be convergent. Our results depicted in Figures 3.11, 3.12 show our findings, and the following difference figures confirm our expectation.

Differences for the discretization (3.12) are

```
maximum difference at time 0.000000 is 0.998867 (in %) 100.00000000
maximum difference at time 0.333333 is 0.027840 (in %) 2.78714085
maximum difference at time 0.666667 is 0.000715 (in %) 0.07162458
maximum difference at time 1.000000 is 0.000135 (in %) 0.01350979
```

while those for discretization (3.11) are

```
maximum difference at time 0.125000 is 0.709033 (in %) 70.9837158757
maximum difference at time 0.250000 is 0.894754 (in %) 89.5768376666
maximum difference at time 0.375000 is 1.061009 (in %) 106.2211725151
maximum difference at time 0.500000 is 0.937179 (in %) 93.8241215025
maximum difference at time 0.625000 is 0.697497 (in %) 69.8287873810
maximum difference at time 0.750000 is 1.031804 (in %) 103.2973533648
maximum difference at time 0.875000 is 0.703944 (in %) 70.4741875121
maximum difference at time 1.000000 is 0.864096 (in %) 86.5075695024
```

We conducted convergence studies of the examples with asymptotic solution $\sin(6\pi(x-y))$, $\sin(6\pi(x-2y))$, and $\sin(6\pi(3x-2y))$. The results obtained in convergence studies show non-convergence of the first two examples, and a jump in the convergence of the third example.

The negative values appear due to the fact that the numerical solution at a time point is compared to an asymptotic solution, rather than the analytic solution. In order to obtain true convergence studies' values, we would have to let time go to infinity. Since we execute the implementations up to a finite time t, both convergent, and divergent results are possible.

Our findings show that montone schemes are stable and in most cases very accurate approximations of the analytical solutions.

Figure 3.5: Numerical solution for the final time 0.2 for the discretization (3.11), boundary condition $\sin(6\pi(x-y))$, grid size $[100 \times 100]$, and initial condition $0\ \forall x, y$.

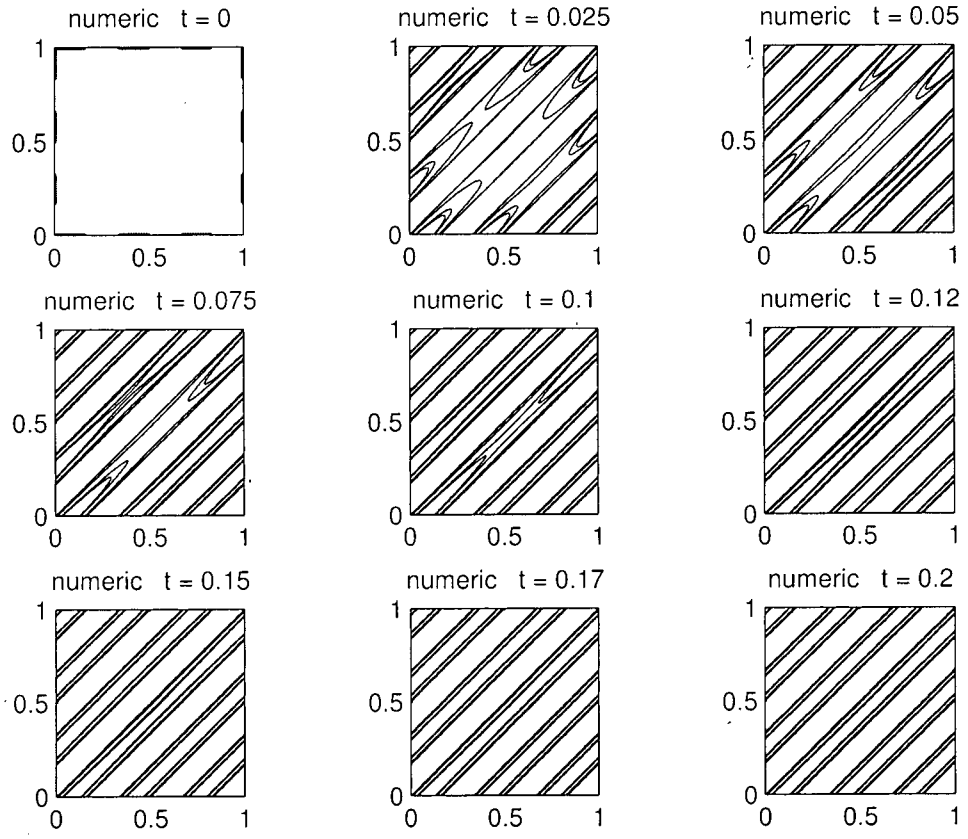


Figure 3.6: Analytical and numerical solutions for the final time 1.0 for the discretization (3.11), boundary condition $\sin(6\pi(x-y))$, grid size $[100 \times 100]$, and initial condition zero.

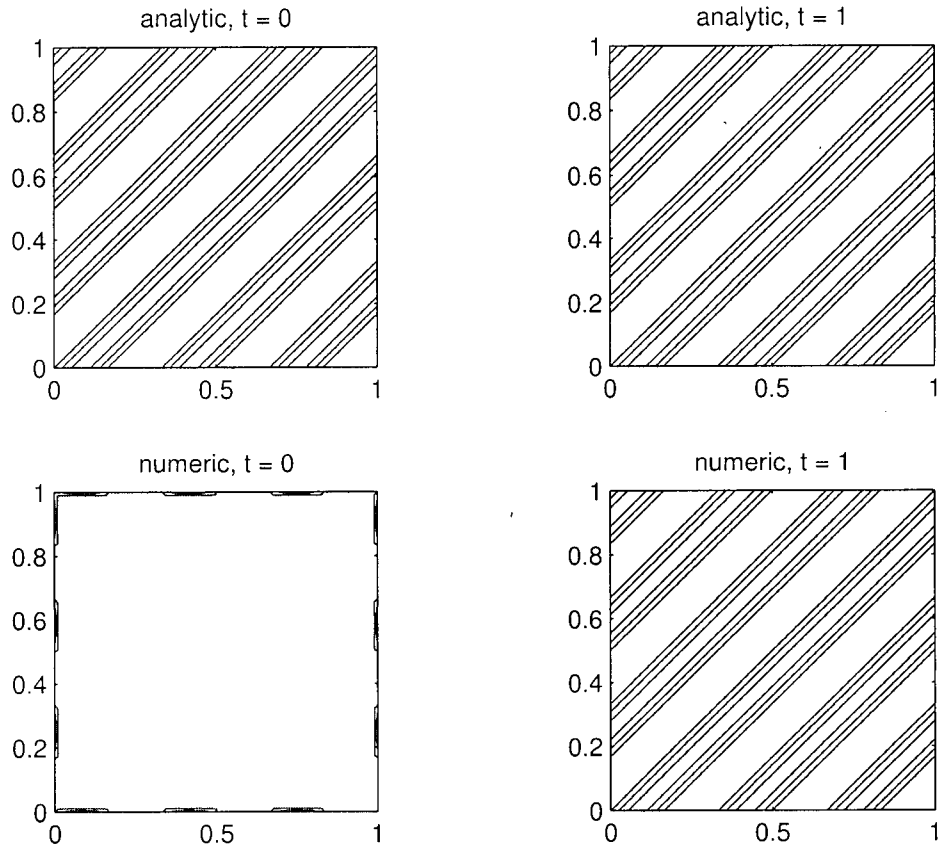


Figure 3.7: Final time 1.2 for the discretization (3.12), boundary condition $\sin(6\pi(x-y))$, grid size $[100 \times 100]$, and initial condition $0\ \forall x, y$.

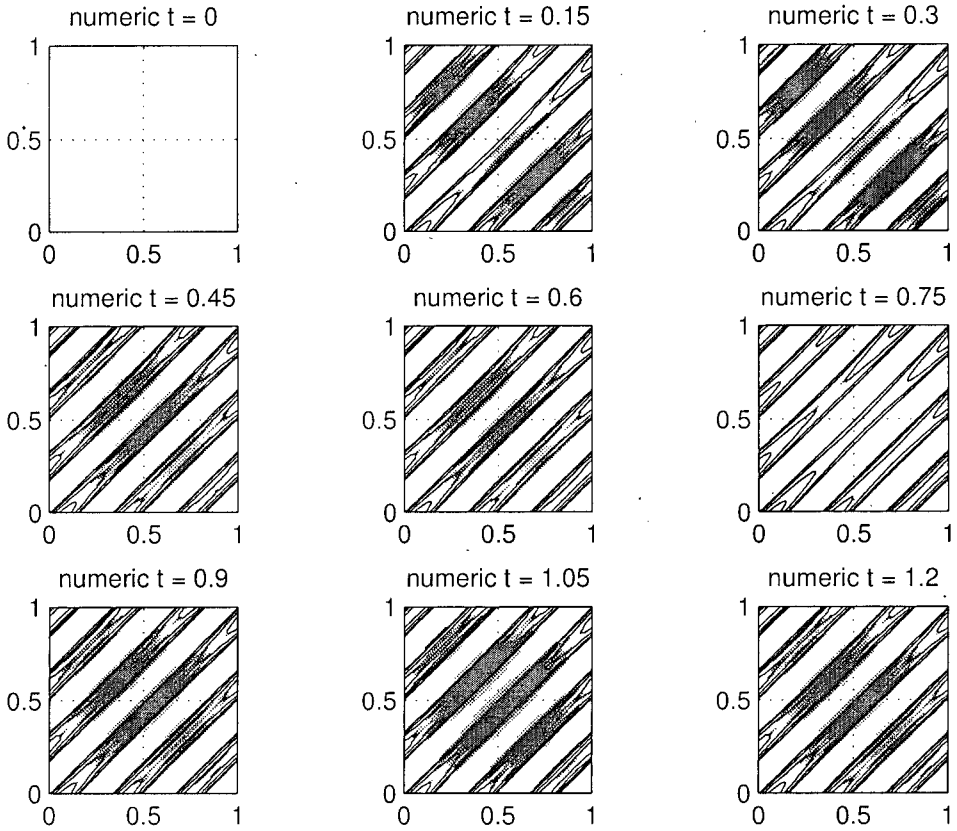


Figure 3.8: Final time 1.2 for the discretization (3.13), boundary condition $\sin(6\pi(x-y))$, grid size $[100 \times 100]$, and initial condition zero for all x, y.

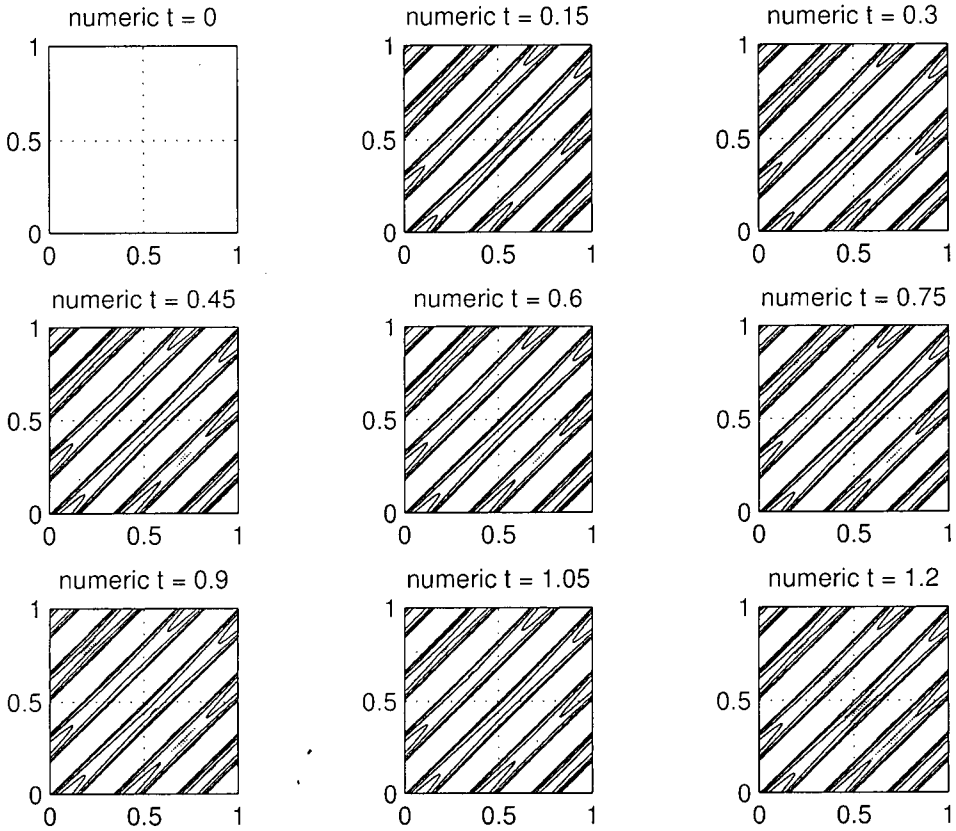


Figure 3.9: Final time 0.2 for the discretization (3.11), periodic boundary conditions, grid size [100 × 100], and initial condition $sin(6\pi(x-y))$.

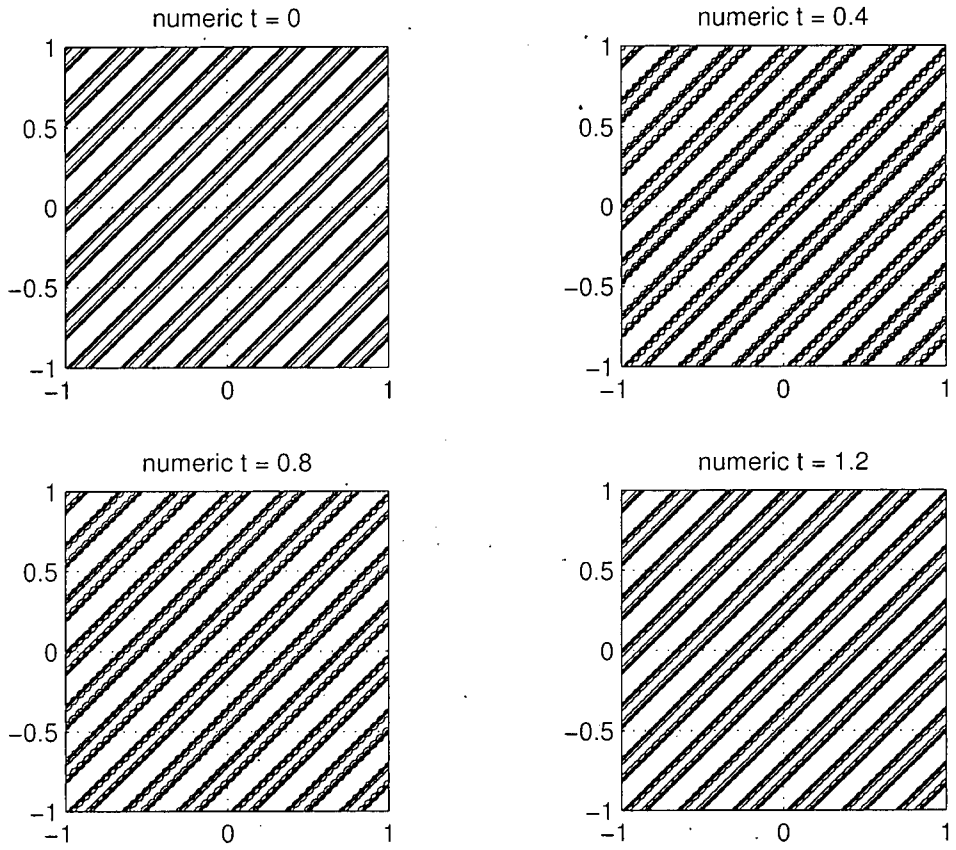


Figure 3.10: Final time 0.2 for the discretization (3.13), periodic boundary conditions, grid size $[100 \times 100]$, and initial condition $\sin(6\pi(x-y))$.

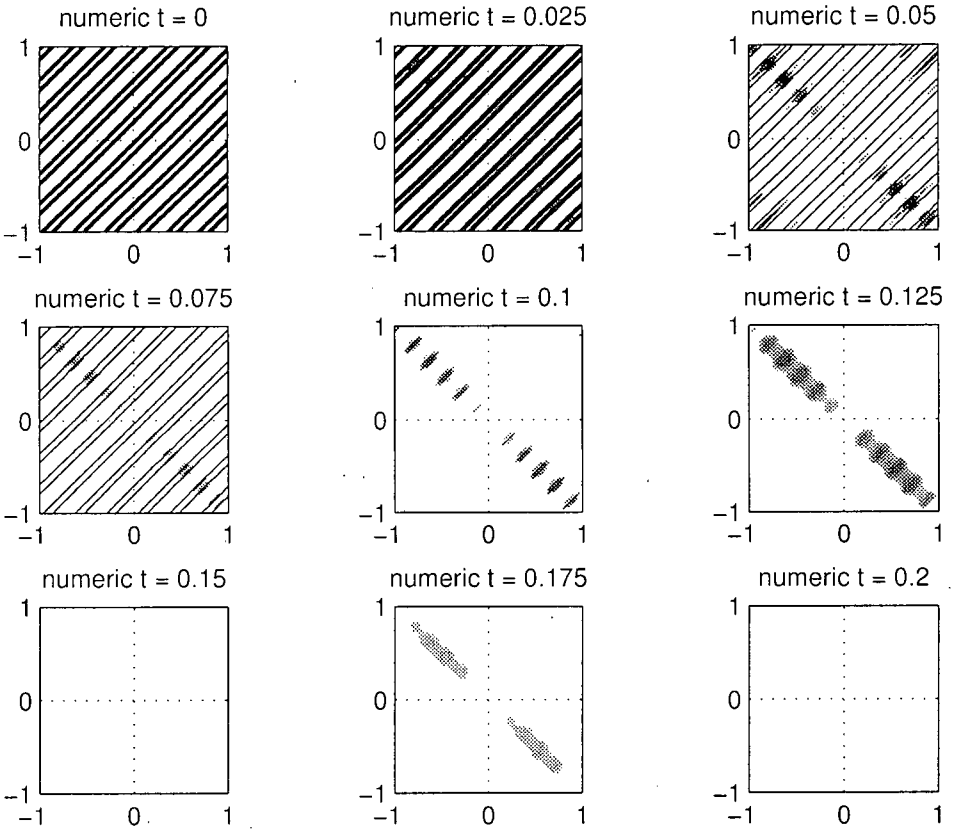


Figure 3.11: Numerical solution for the final time 1.0 for the discretization (3.12), boundary condition $\sin(6\pi(x+y))$, grid size $[100 \times 100]$, and initial condition $0\ \forall x, y \in D$.

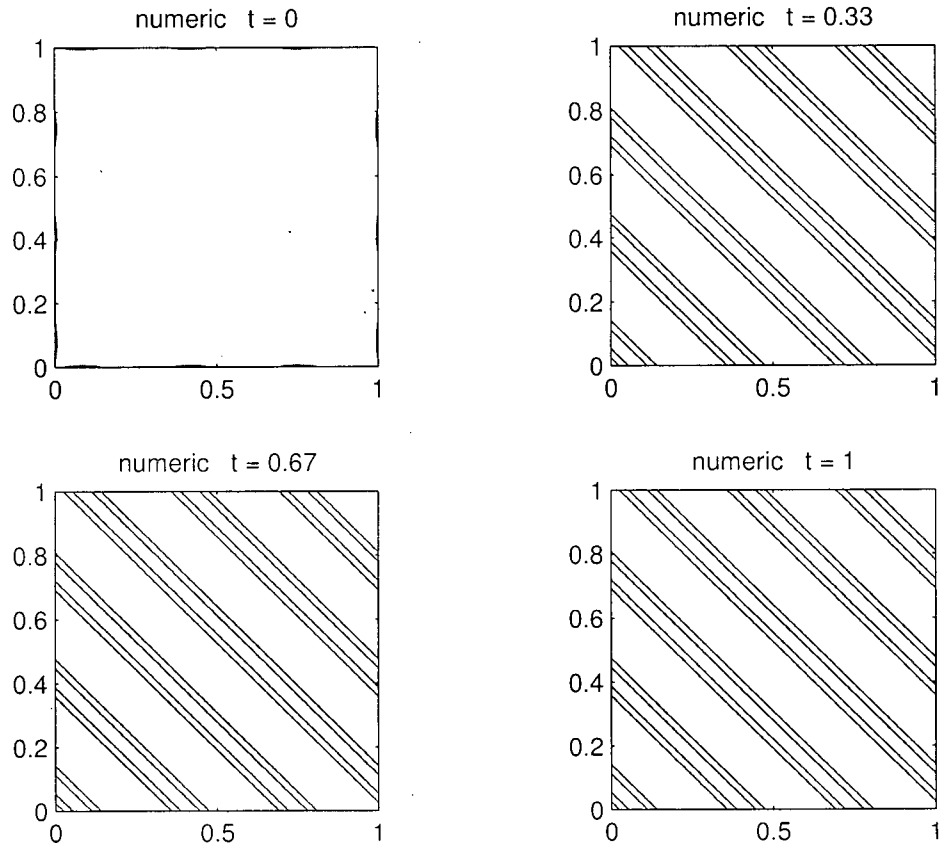
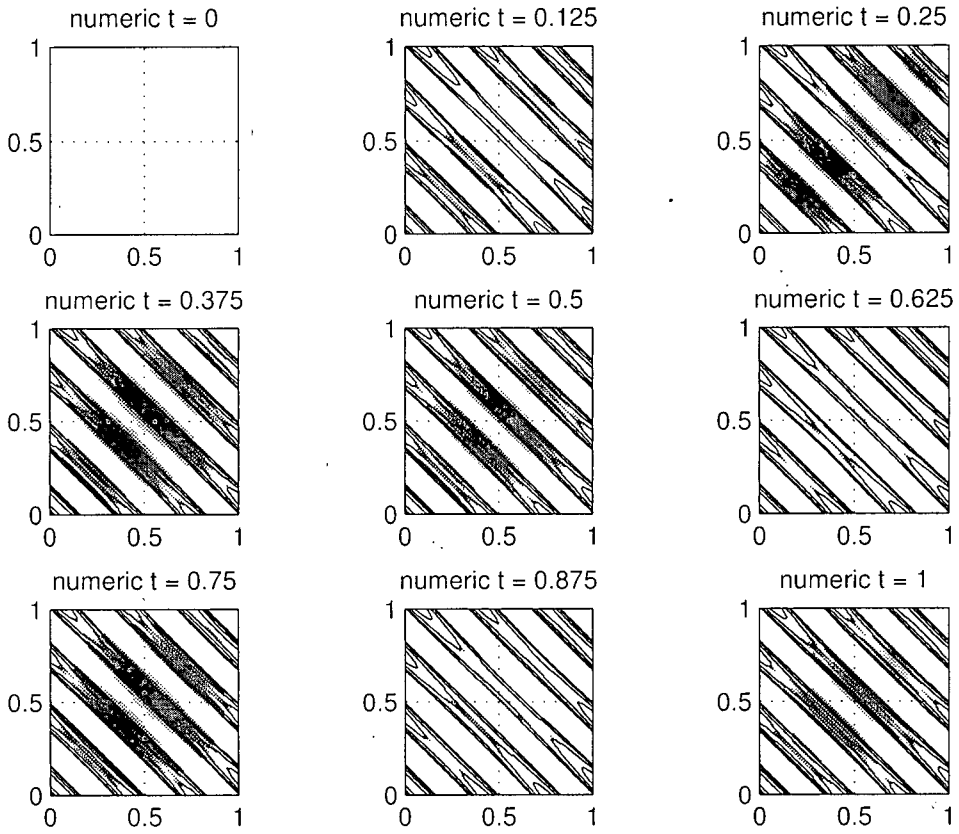


Figure 3.12: Numerical and analytical solutions for the final time 1.2 for the discretization (3.11), boundary condition $\sin(6\pi(x+y))$, grid size $[100\times100]$, and initial condition $0\ \forall x, y \in D$.



Chapter 4

Building monotone schemes

In this chapter we present our general procedure for building monotone schemes from positive semidefinite and positive definite diffusion tensors. We restrict our research to PDEs with purely diffusive terms such as General degenerate parabolic equation.

4.1 Central difference approximations

Let σ be an $m \times n$ matrix and D be the Hessian, i.e. the square matrix of second order derivatives. Then we have

$$trace(\sigma D \sigma^T) \;=\; trace(\sigma^T \sigma D) \;=\; \sum_{ij=1}^{2} a_{ij}\partial_i\partial_j,$$

where the entries of the matrix $a = \sigma^T\sigma$ are denoted by a_{ij}, and the matrix a is positive definite or semidefinite.

Discretizations of differential operators in Section 2.2.1 are given for the case of equal spatial step size h in both discretization directions. We now define these discretizations with spatial steps h_1 and h_2, where h_1 is the step size in the x-axis direction, while h_2 is the step size in the y-axis direction. The corresponding expressions for the forward and backward difference operators are

$$\partial_i f(\mathbf{x}) \to \frac{1}{h_i}\big(f(\mathbf{x}) + \mathbf{e}_i h_i) \;-\; f(\mathbf{x})\big),$$
$$\partial_i f(\mathbf{x}) \to \frac{1}{h_i}\big(f(\mathbf{x}) \;-\; f(\mathbf{x} - \mathbf{e}_i h_i)\big).$$

Then $\eth_{ii} f(\mathbf{x})$ has the form as before with h_i instead of h. Similarily, the mixed partial differential operator can be approximated by one of the following four possibilities

$$\partial_i\partial_j f(\mathbf{x}) \to \eth_{ij} f(\mathbf{x}) =$$
$$\frac{1}{h_1 h_2}\begin{cases} f(\mathbf{x} \pm \mathbf{e}_i h_i \pm \mathbf{e}_j h_j) - f(\mathbf{x} \pm \mathbf{e}_i h_i) - f(\mathbf{x} \pm \mathbf{e}_j h_j) + f(\mathbf{x}), \\ -f(\mathbf{x} \pm \mathbf{e}_i h_i \mp \mathbf{e}_j h_j) + f(\mathbf{x} \pm \mathbf{e}_i h_i) + f(\mathbf{x} \mp \mathbf{e}_j h_j) - f(\mathbf{x}). \end{cases} \tag{4.1}$$

We construct discretizations of the operator $a_{11}(\partial_1)^2 + 2a_{12}\partial_1\partial_2 + a_{22}(\partial_2)^2$ by using $a_{11}\eth_{11} + 2a_{12}\eth_{12} + a_{22}\eth_{22}$ as in the previous case. The resulting entries of the system matrix are

now

$$\begin{aligned} A_{kl\,kl} &= -2\left[h_1^{-2}a_{11} + h_2^{-2}a_{22} - h_1^{-1}h_2^{-1}|a_{12}|\right] \\ A_{kl\,k\pm1l} &= h_1^{-1}[h_1^{-1}a_{11} - h_2^{-1}|a_{12}|] \\ A_{kl\,kl\pm1} &= h_2^{-1}[h_2^{-1}a_{22} - h_1^{-1}|a_{12}|] \\ A_{kl\,k\pm1l\pm1} &= h_1^{-1}h_2^{-1}|a_{12}| \quad (= A_{kl\,k\pm1l\mp1}) \end{aligned} \tag{4.2}$$

In order to get monotone schemes we must have system matrices of positive type. This implies that the following conditions must be satisfied

$$\begin{aligned} h_1^{-1}a_{11} - h_2^{-1}|a_{12}| &\geq 0, \\ h_2^{-1}a_{22} - h_1^{-1}|a_{12}| &\geq 0. \end{aligned} \tag{4.3}$$

Let us analyse these conditions. We know that the matrix a is positive definite or semidefinite. In the former case we have $a_{12}^2 < a_{11}a_{22}$ and in the latter case $a_{12}^2 = a_{11}a_{22}$. This conclusion follows from the conditions $det(a) > 0$ and $det(a) = 0$, respectively. If the matrix a is positive definite, there exist plenty of possibilities of choosing the pairs h_1, h_2 in order to satisfy (4.3). For instance if we fix $h_1 > 0$, then there are as many h_2 as there are real numbers. If a is semidefinite, then the choices are very restricted. If we fix $h_1 > 0$, then h_2 is also fixed since it is calculated from the expression

$$\frac{h_2}{h_1} = \frac{|a_{12}|}{a_{11}}, \tag{4.4}$$

assuming that $a_{11} \neq 0$. The above ratio was computed knowing that in case of a degeneracy the equations (4.3) are one and the same equation for which a strict equality holds as the following computation shows.

Take (4.3) and plug into each one $|a_{12}| = \sqrt{a_{11}a_{22}}$,

$$\frac{a_{11}}{h_1} - \frac{\sqrt{a_{11}a_{22}}}{h_2} \geq 0 \qquad \frac{a_{22}}{h_2} - \frac{\sqrt{a_{11}a_{22}}}{h_1} \geq 0$$

Now divide the one on the left with $\sqrt{a_{11}}$, and one on the right with $\sqrt{a_{22}}$,

$$\frac{\sqrt{a_{11}}}{h_1} - \frac{\sqrt{a_{22}}}{h_2} \geq 0 \qquad \frac{\sqrt{a_{22}}}{h_2} - \frac{\sqrt{a_{11}}}{h_1} \geq 0,$$

$$\frac{\sqrt{a_{11}}}{h_1} \geq \frac{\sqrt{a_{22}}}{h_2} \qquad \frac{\sqrt{a_{22}}}{h_2} \geq \frac{\sqrt{a_{11}}}{h_1}.$$

In other words

$$\frac{\sqrt{a_{11}}}{h_1} \geq \frac{\sqrt{a_{22}}}{h_2} \geq \frac{\sqrt{a_{11}}}{h_1} \quad \rightarrow \quad \frac{\sqrt{a_{11}}}{h_1} = \frac{\sqrt{a_{22}}}{h_2}.$$

But this means that

$$\begin{aligned} h_1^{-1}a_{11} - h_2^{-1}|a_{12}| &= 0, \\ h_2^{-1}a_{22} - h_1^{-1}|a_{12}| &= 0. \end{aligned}$$

The reason for equation (4.4) is simple. There exists only one rotation such that in the new coordinates the diffusion tensor has the form

$$a = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

Thus we can draw the following conclusion. *Instead of rotations we use the original coordinate system and spatial grid steps of different sizes.*

Let $\mathbf{x} = (k, l)$ be a grid node. Then the system matrix has non-trivial entries for those grid nodes which are on the cross through $\mathbf{x}$ and additional two grid-knots as illustrated in Figure 4.1. These sets can be called numerical neighbourhoods.

4.2 Computational procedure

1. Given σ calculate $a = \sigma^T \sigma$.

2. Let $M1$ be a given number of grid intervals in the x-axis direction, which means that there are $M1+1$ grid nodes in the $\mathbf{e}_1$ direction. Then $h_1 = 1/M1$ is the corresponding grid step size.

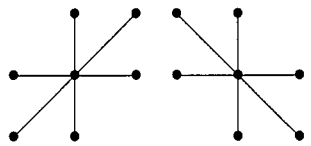
3. Calculate
$$r_1 = \frac{|a_{12}|}{a_{11}}, \quad r_2 = \frac{|a_{12}|}{a_{22}}.$$
If both $r_k \leq 1$, then the spatial differential operator is of the second order in more than one coordinate directions so take $h_2 = h_1$ and construct the system matrix.

4. Otherwise, calculate the number
$$g = \sqrt{\frac{a_{22}}{a_{11}}},$$
and set $h_2 = gh_1$.
The number g is calculated from (4.3)

5. The number $\mu = 1/h_2$ is not an integer so one needs to take the nearest integer $M2$. This integer determines the number of grid intervals in the second direction. In this way the domain D is equal to a rectangle of sides $h_1 M1$ and $h_2 M2$. This rectangle is very close to the unit square for fine grids.

6. If $a_{12} \geq 0$ the system matrix is constructed by using the numerical neighbourhoods depicted in Figure 4.1(a), and for $a_{12} \leq 0$ one uses the numerical neighbourhoods depicted in Figure 4.1(b).

Figure 4.1: The numerical neighbourhoods used in the calculation of the terms of the system matrix A in (4.2).



(a): $a_{12} \geq 0$ *(b):* $a_{12} \leq 0$

4.3 Problem definition

We would like to test out implementation on three problems. Two degenerate and one non-degenerate. We call them *Problem C*, *Problem D*, and *Problem E*. All three problems are examples of General degenerate parabolic equation. The PDE analysed in Problem C is

$$u_t - (4u_{xx} + 4u_{xy} + u_{yy}) = 0, \tag{4.5}$$

while the one for Problem D is

$$u_t - (4u_{xx} + 12u_{x,y} + 9u_{yy}) = 0. \tag{4.6}$$

We define these two problems next.

Problem C. Consider domain $D = (0,1)^2$, equation (4.5) with analytical solution

$$u(t,x,y) = \sin(6\pi(x-2y)),$$

where boundary condition is given by the function

$$u(0,x,y) = \sin(6\pi(x-2y)),$$

and zero initial condition. We analyse numerical solution as $t \to \infty$.

Problem D. The domain is $D = (0,1)^2$, the equation is (4.6), the analytical solution is

$$u(t,x,y) = \sin(6\pi(3x-2y)),$$

the boundary condition is given by the function

$$u(0,x,y) = \sin(6\pi(3x-2y)),$$

and the initial condition is zero. Again, we analyse numerical solution as $t \to \infty$.

The difference between the above two problems and Problem E is in the fact that Problem E is designed to test our implementation with a positive definite diffusion tensor. The PDE studied in Problem E is

$$u_t - (a_{11}u_{xx} + 2a_{12}\,u_{x,y} + a_{22}u_{yy}) = 0, \tag{4.7}$$

with the corresponding diffusion tensor $a = \{a_{ij}\}_{11}^{22}$,

$$\begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

We chose to write the PDE in this way to point out that we could have used any other positive definite diffusion tensor of the form

$$\begin{bmatrix} a & b \\ b & a \end{bmatrix},$$

for the solution

$$u(t,x,y) = -x\frac{(1-x)}{a_{11}} + y\frac{(1-y)}{a_{22}}$$

to still hold.

Given our choice of the diffusion tensor we define Problem E as follows.

Problem E. Consider again the domain $D = (0,1)^2$, equation (4.7) with the solution $\forall t$ is

$$u(t,x,y) = -x(1-x) + y(1-y). \tag{4.8}$$

Numerical boundary condition on the top and bottom borders of the domain is $-x(1-x)$, while the one on the left and right borders of the domain is $y(1-y)$. We analyse numerical solution as $t \to \infty$.

4.4 Findings

All three problems were tested on the domain discretized by $N1 = 100$ grid nodes in the x-axis direction. The number of nodes in the y-axis direction is computed from $N1$ as discussed in Section 4.2.

For Problems C and D we generate the results using both non-equally and equally spaced grids. Problem E we will investigate only on an equally spaced grid since its associated diffusion tensor requires an equally spaced grid.

We must note that for problems C, D, and E the difference figures do not show the actual numerical error. Instead they show the difference between the numerical solution and the analytical solution at a point in time. This is why the percentage change decreases with the increase in the running time.

The differences of Problem C generated for unequal grid step sizes and final time 0.1 are

```
maximum difference at time 0.000000 is 1.000000 (in %) 100.00000000
maximum difference at time 0.033330 is 0.341236 (in %) 34.12359953
maximum difference at time 0.066660 is 0.091538 (in %) 9.15384889
maximum difference at time 0.099990 is 0.024577 (in %) 2.45770812
```

As the final running time of the simulation increases, the difference decreases. Figures 4.2 and 4.3 depict the numerical and the analytical solutions for Problem C respectively for the final time 0.1. Figure 4.4 shows the result obtained when using equally spaced grid for final time 0.25. We include the following differences generated by the equally spaced grid scheme for the final time 0.25.

```
maximum difference at time 0.000000 is 0.998867 (in %) 100.00000000
maximum difference at time 0.031247 is 0.493938 (in %) 49.44984886
maximum difference at time 0.062494 is 0.341225 (in %) 34.16121397
maximum difference at time 0.093740 is 0.310776 (in %) 31.11286388
maximum difference at time 0.124987 is 0.304707 (in %) 30.50522153
```

```
maximum difference at time 0.156234 is 0.303492 (in %) 30.38361548
maximum difference at time 0.187481 is 0.303254 (in %) 30.35983013
maximum difference at time 0.218728 is 0.303245 (in %) 30.35888731
maximum difference at time 0.249975 is 0.303245 (in %) 30.35888731
```

The relative difference remains close to 30.35%, as illustrated in Figure 4.4 and evident from the following differences

```
maximum difference at time 0.000000 is 0.998867 (in %) 100.00000000
maximum difference at time 0.333333 is 0.303245 (in %) 30.35888731
maximum difference at time 0.666667 is 0.303245 (in %) 30.35888731
maximum difference at time 1.000000 is 0.303245 (in %) 30.35888731
```

Problem D on an unequally spaced grid is shown in Figure 4.5. The difference figures for the Problem are shown below for final time 0.25.

```
maximum difference at time 0.000000 is 0.999998 (in %) 100.00000000
maximum difference at time 0.083333 is 0.003905 (in %) 0.39048283
maximum difference at time 0.166667 is 0.003208 (in %) 0.32080485
maximum difference at time 0.250000 is 0.003208 (in %) 0.32080485
```

Figure 4.6 illustrates the numerical solution to Problem D computed on an equally spaced grid. Again, the equally spaced grid scheme produces a non-convergent method as can be seen from the following differences.

```
maximum difference at time 0.000000 is 0.998867 (in %) 100.00000000
maximum difference at time 0.083333 is 0.689878 (in %) 69.06607524
maximum difference at time 0.166667 is 0.689878 (in %) 69.06607524
maximum difference at time 0.250000 is 0.689878 (in %) 69.06607524
```

This is also visible in Figure 4.6. We could say that a lower bound on difference is 69.06%.

Figures 4.7 and 4.8 show the numerical and analytical solution to Problem E for final time 0.05. In Figure 4.7 it can be seen how the solution is being formed. We ran the same setup for final time 0.1 to obtain the following difference values

```
maximum difference at time 0.000000 is 0.239976 (in %) 100.00000000
maximum difference at time 0.033330 is 0.006672 (in %) 2.78036850
maximum difference at time 0.066660 is 0.000312 (in %) 0.12986445
maximum difference at time 0.099990 is 0.000015 (in %) 0.00604490
```

From the presented tests we conclude that methods with unequal step sizes give far better results than methods with equal step sizes.

Figure 4.2: Numerical solution for the final time 0.1 using unequal grid step sizes discretization, boundary condition $\sin(6\pi(x-2y))$, grid size $[100 \times 201]$, and initial condition $0\ \forall x, y \in D$.

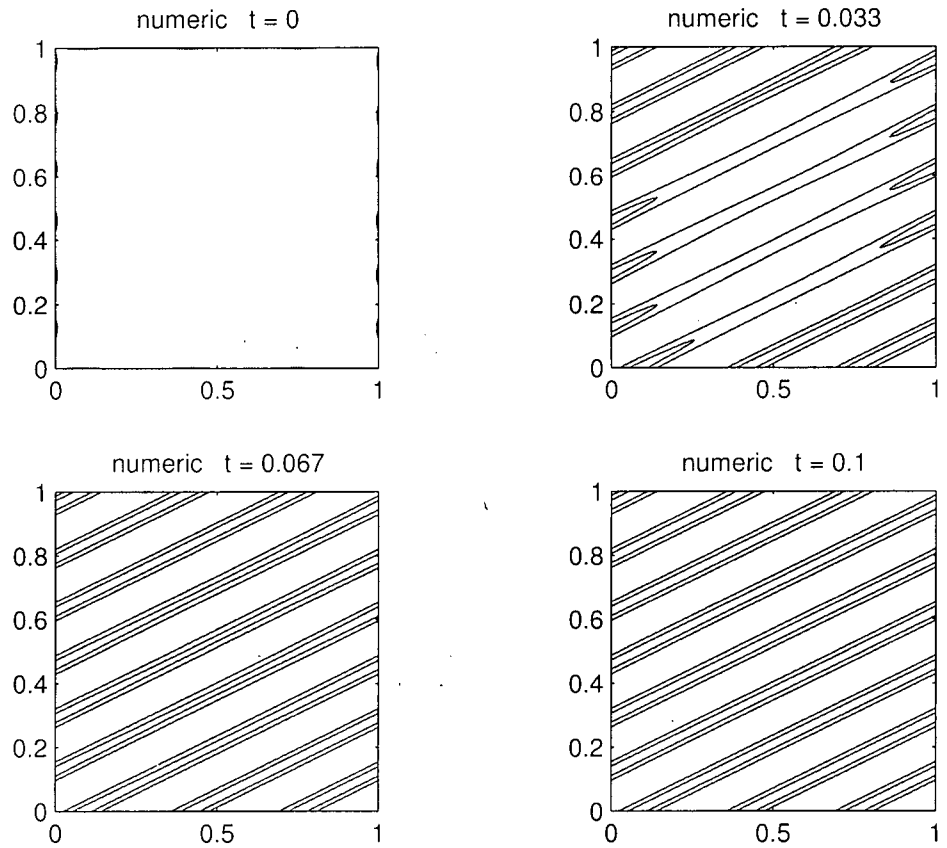


Figure 4.3: Analytical solution for the final time 0.1 using unequal grid step sizes discretization, and grid size [100 × 201].

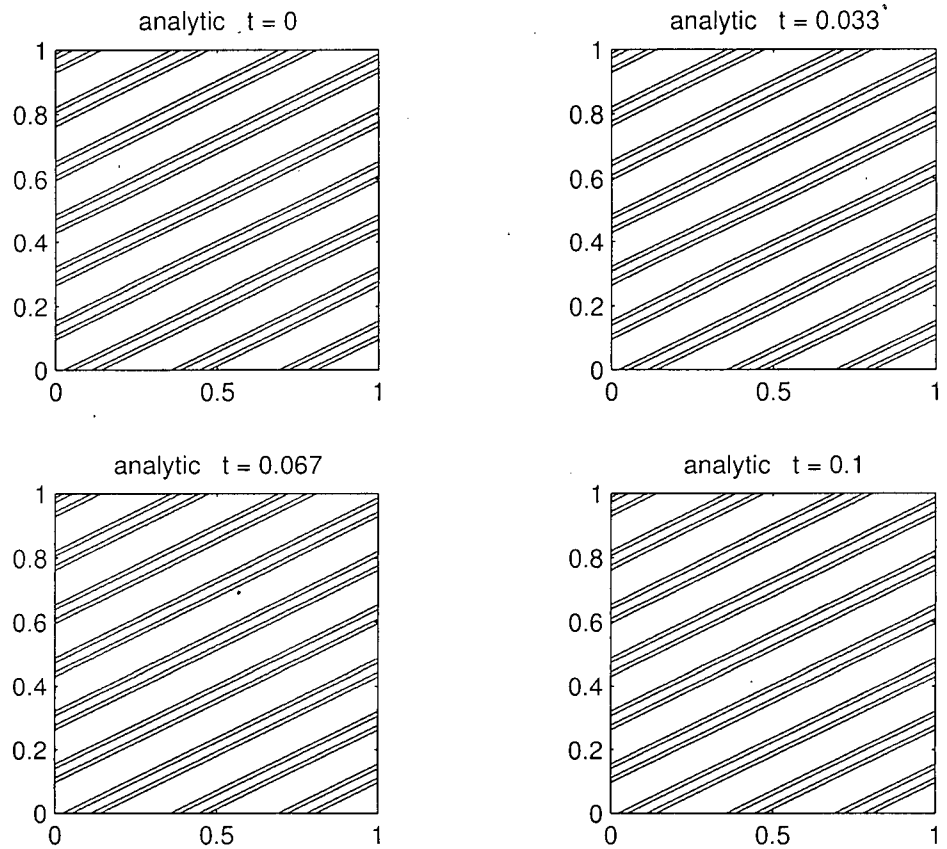


Figure 4.4: Numerical solution for the final time 0.25 using equal grid step sizes discretization, boundary condition $\sin(6\pi(x-2y))$, grid size $[100 \times 100]$, and initial condition $0\ \forall x, y \in D$.

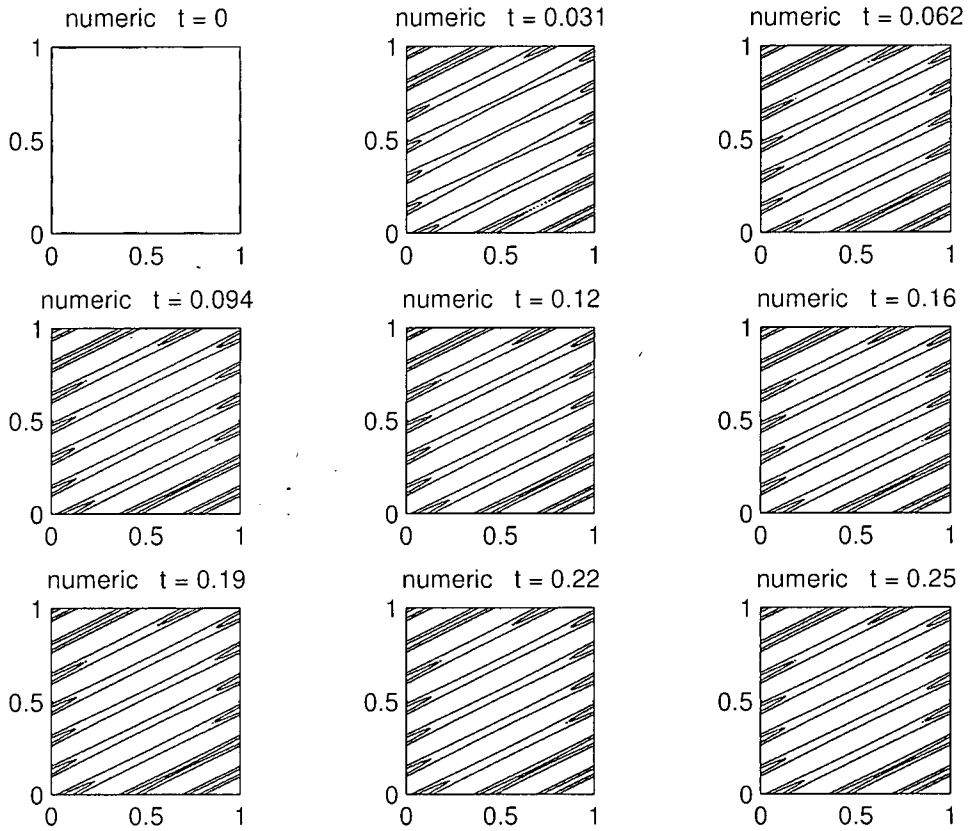


Figure 4.5: Numerical and analytical solutions for the final time 0.25 using unequal grid step sizes discretization, boundary condition $\sin(6\pi(3x-2y))$, grid size $[100\times 68]$, and initial condition 0 $\forall x, y \in D$.

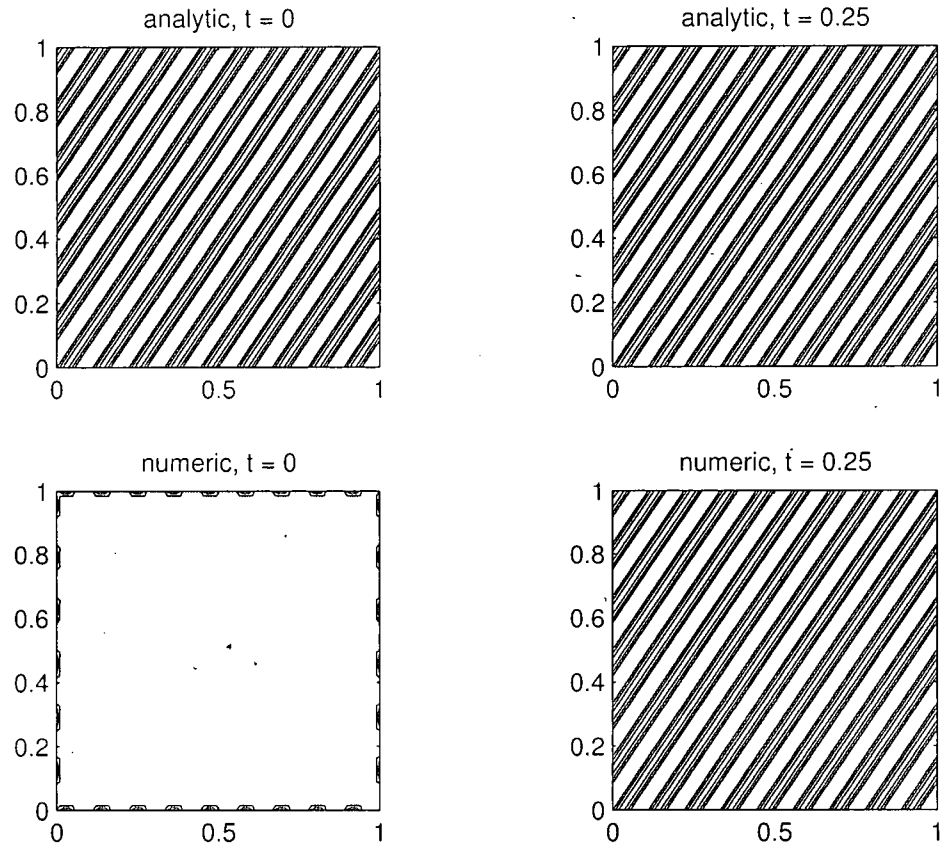


Figure 4.6: Numerical solution for the final time 0.25 using equally spaced grid step sizes, boundary condition $\sin(6\pi(3x-2y))$, grid size $[100\times 100]$, and initial condition $0\ \forall x, y \in D$.

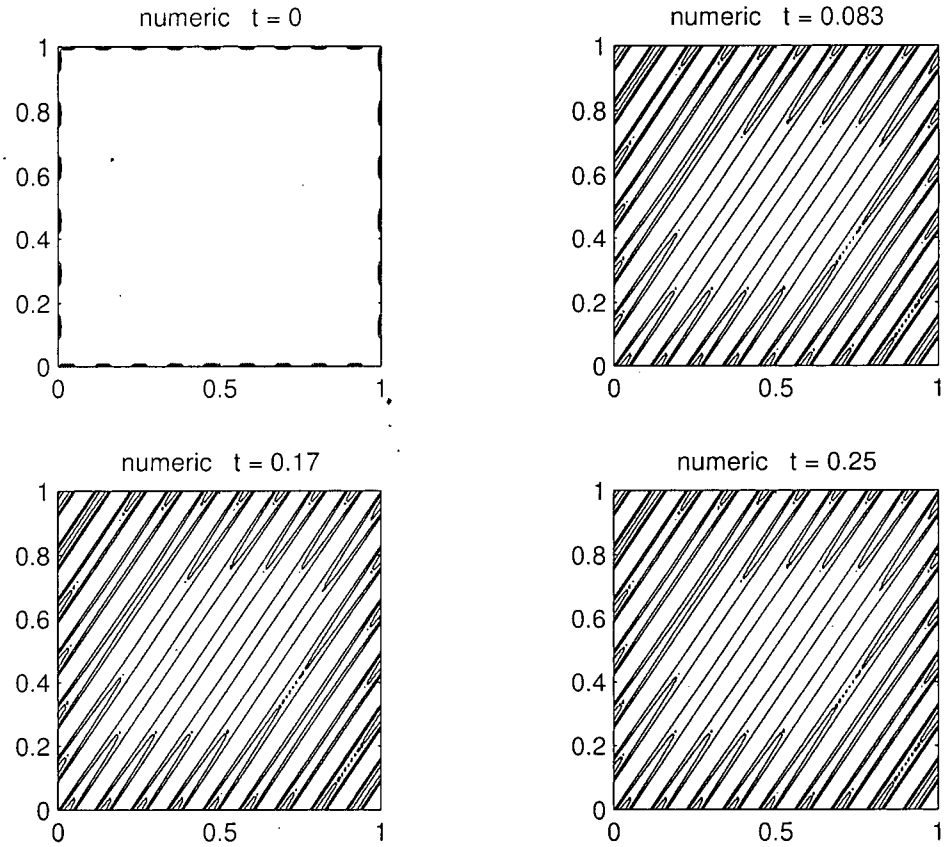


Figure 4.7: Numerical solution for the final time 0.05, with boundary condition $-x\frac{(1-x)}{a_{11}}$ on the top and bottom ends of the domain, $y\frac{(1-y)}{a_{22}}$ on the left and right ends of the domain, and grid size [100 × 100].

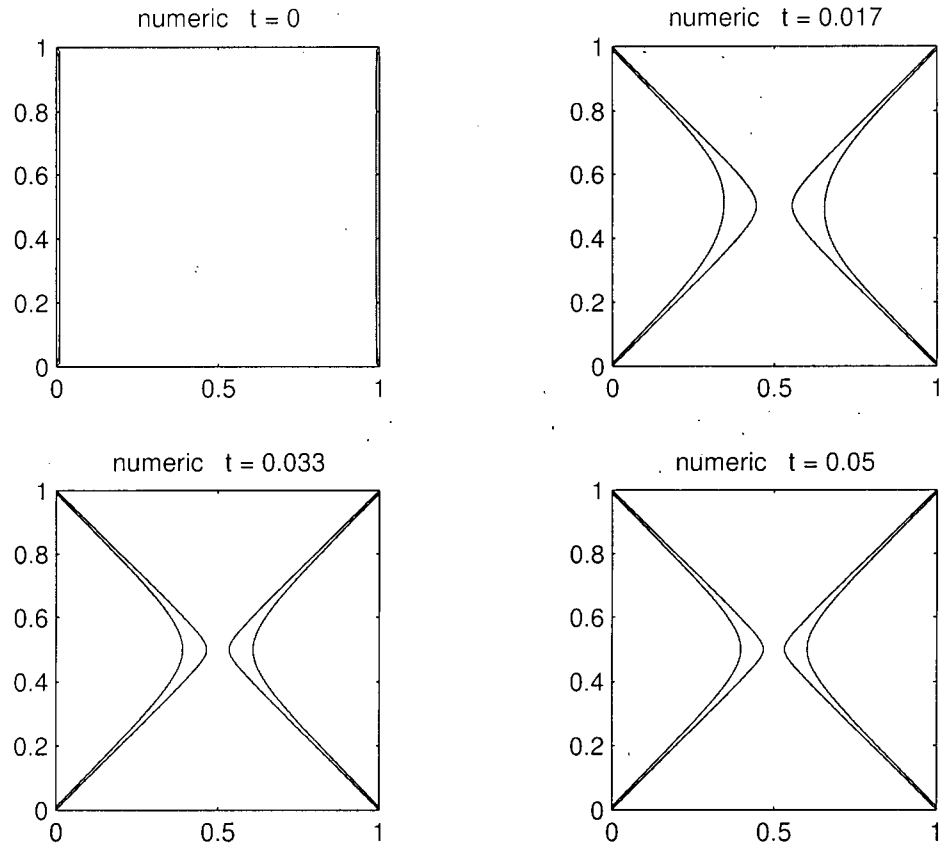
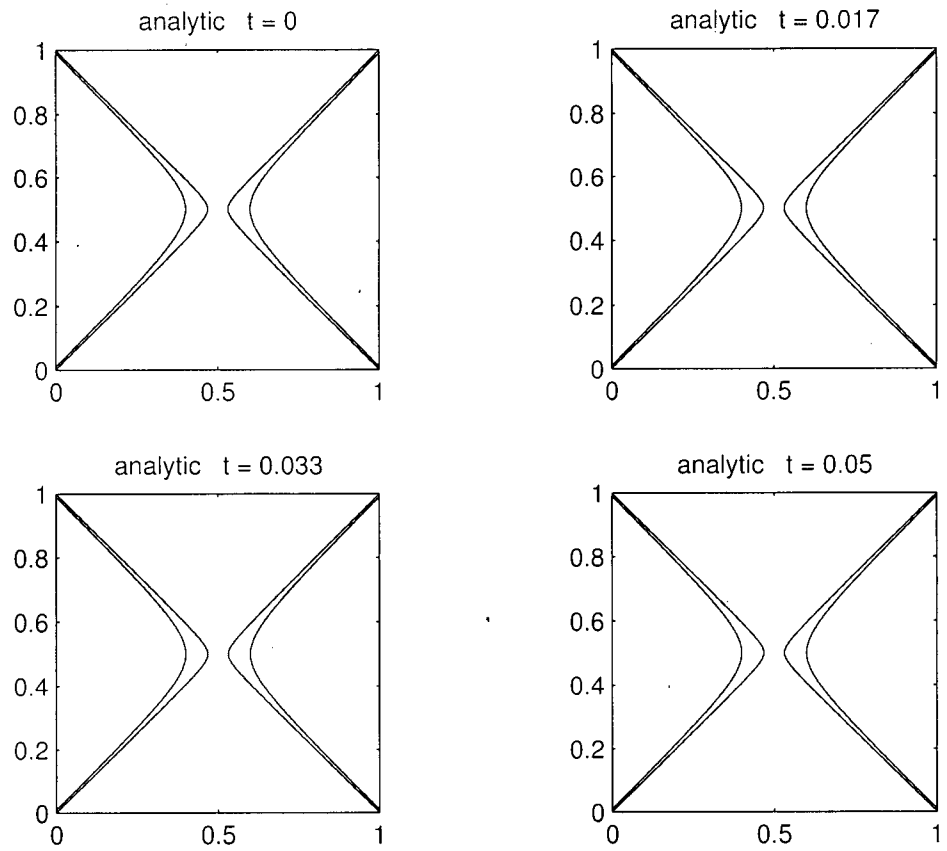


Figure 4.8: Analytical solution for the final time 0.05, with boundary condition $-x\frac{(1-x)}{a_{11}}$ on the top and bottom ends of the domain, $y\frac{(1-y)}{a_{22}}$ on the left and right ends of the domain, and grid size [100 × 100].



4.5 Analysing Problem 4.

Now that we have developed our method of unequal step sizes in we can look at the final equation of interest, General degenerate parabolic equation with advection,

$$u_t(\mathbf{x}) + \mathbf{v} \bigtriangledown u - (u_{xx} + 2u_{xy} + u_{yy}) = 0. \tag{4.9}$$

This equation is interesting since it combines convection with diffusion terms discretized using our unequal step size method. The first order term $\mathbf{v} \bigtriangledown u$ is discretized using upwinding, while the second order terms using $\eth_{ij}$ developed in Section 2.2.1.

4.5.1 Problem Definition

In accordance with our alphabetical naming we call this example Problem F.

Problem F. Consider PDE (4.9) over domain $D = (0,1)^2$. The solution is

$$u(t,x,y) = \sin(6\pi(x - (y - v_2 t))) \quad \text{for } v_1 = 0,$$

where t is time and (x,y) are grid-node coordinates. The boundary condition is computed at every time step from the value of the analytical solution. The velocity is constant $\mathbf{v} = (v_1, v_2)$. We have chosen $\mathbf{v} = (0, 0.5)$.

4.5.2 Findings

We ran the implementation of Problem F until final time 1.0 on D discretized by 100×100 nodes. The differences are

```
maximum difference at time 0.000000 is 0.000000 (in %) 0.00000000
maximum difference at time 0.333333 is 0.091789 (in %) 9.18926738
maximum difference at time 0.666667 is 0.093522 (in %) 9.36283003
maximum difference at time 1.000000 is 0.093567 (in %) 9.36730545
```

The difference is noticeable in Figure 4.9 for $t = 0.33$, for example. A closer look at difference values shows that the magnitude of velocity drives the magnitude of percentage difference. I.e. the larger the velocity is, the larger is the percentage difference between the numeric and analytic solutions. To show this we include the following difference values for final running time 0.5 and velocity $\mathbf{v} = (0,1)$ on D

```
maximum difference at time 0.000000 is 0.000000 (in %) 0.00000000
maximum difference at time 0.166667 is 0.153738 (in %) 15.39127710
maximum difference at time 0.333333 is 0.170976 (in %) 17.11701003
maximum difference at time 0.500000 is 0.171830 (in %) 17.20244873
```

These differences are depicted in Figure 4.10. The difference figures can appear surprising at first. However, the error magnitude of h coming from the discretization of

first order derivatives when multiplied by the velocity $\mathbf{v}$ does not render the differences completely unexpected.

We also note that the percentage differences decrease with the increase in the number of grid-nodes.

Table 4.1 shows the results of the convergence study of Problem F. Since the example does compare the numerical solution at a time point to the actual analytical solution the values in the table confirm the method's convergence.

Table 4.1: Example of the IVP $\sin(6\pi(x-(y-v_2t)))$ for final time 1.0, and $\mathbf{v}=(0,0.5)$.

$N1$	L_∞	r	L_1	r	L_2	r
50	0.172639		0.04204		0.05861	
100	0.093567	0.88	0.02254	0.90	0.03122	0.90
200	0.050461	0.89	0.01505	0.58	0.01839	0.76

Figure 4.9: Numerical solution for the final time 1.0, with boundary condition $\sin(6\pi(x - (y - v_2 t)))$, where $v_2 = 0.5$, grid size $[100 \times 100]$, and initial condition $\sin(6\pi(x - y))$.

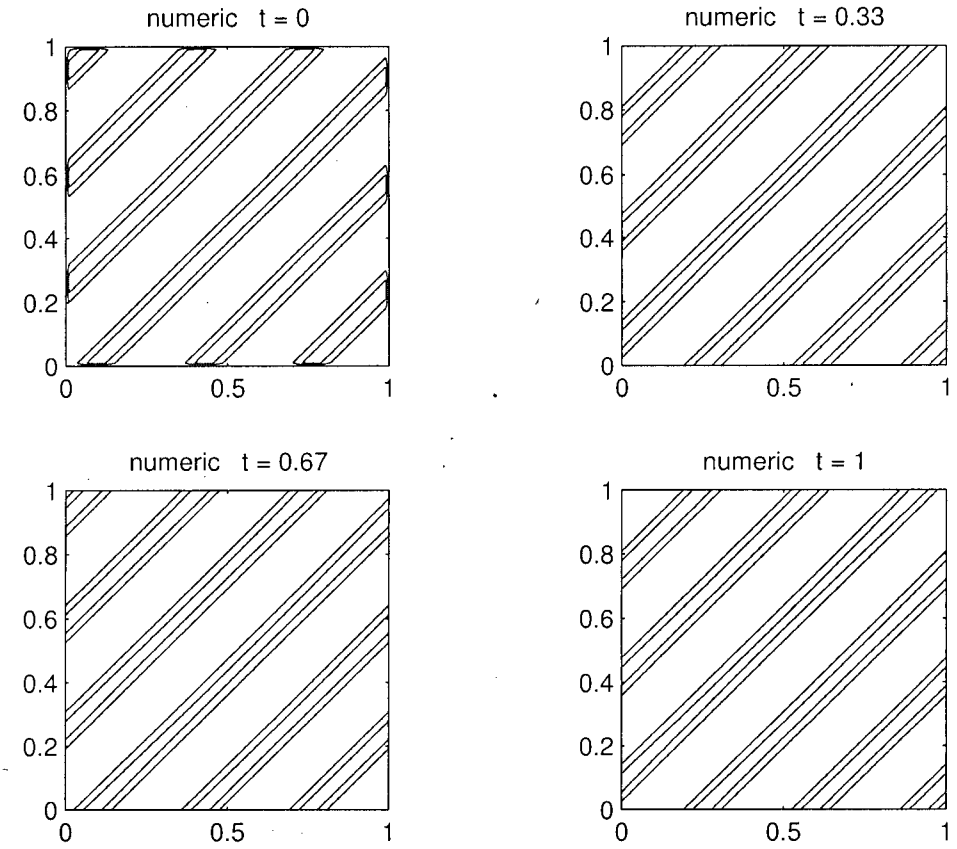
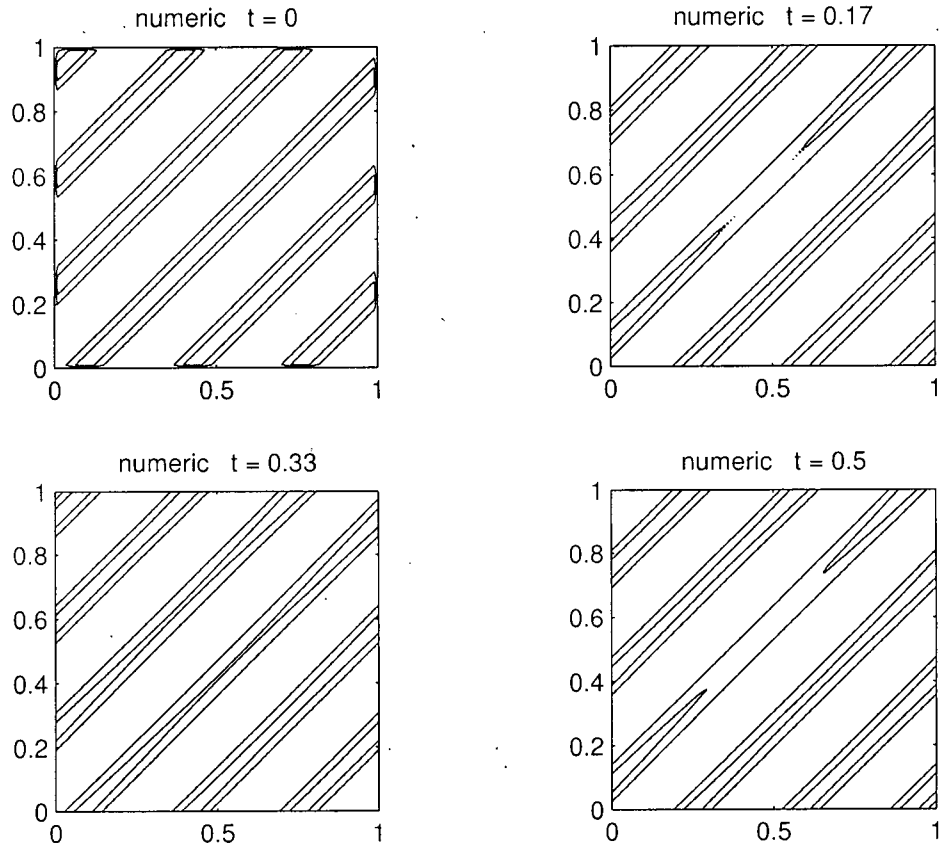


Figure 4.10: Numerical solution for the final time 0.5, with boundary condition $\sin(6\pi(x - (y - v_2 t)))$, where $v_2 = 1$, grid size $[100 \times 100]$, and initial condition $\sin(6\pi(x - y))$.



Chapter 5

Software implementation

In this chapter we take a close look at the way we have implemented our programs in MATLAB's M language. The implementations are located within files with extension m, for example *filename.m*. Therefore, we call such a file *m file*.

A list of programs created in this thesis can be found in Appendix where it is also explained how to execute the programs.

5.1 Implementation details of m files

The Toolbox of Level Set Methods 1.1. package developed by I. Mitchell [1] is a set of m files that are ready to be executed and edited by the user. The package is distributed from `http://www.cs.ubc.ca/ mitchell/ToolboxLS/index.html`
and comes with a manual explaining its contents and usage. In order to understand the m language implementations of this thesis it is important to understand the simple convective flow example explained on the pages 14 to 27 of the manual. The thesis m file implementations follow the layout of the convective flow example. In this section we explain on one example implementation the most imortant parts of the code under separate headings. The other implementations have layout very similar to the one of the example implementation.

Let our example implementation be the m program `GDPdirichlet.m` which implements the monotone discretization (3.11) of General degenerate parabolic equation for the boundary values $\sin(6\pi(x-y))$ and initial condition zero on a cartesian grid. The PDE solved by this problem is (3.9), and an example run of this implementation is depicted in Figure 3.5.

A reader will notice that we use the name of an m file to name the function the file contains. In effect, the name of the function can be any. Execution of MATLAB functions is done by a call to the filename containing the function, not by the function name itself. This, however, is true only if a file contains a single function. Function calls within an m file are done by function names. Most of our m files contain just one function and so when refering to a function we interchange the file name with the function name.

When specifying file names below we use asterisk (*) in the usual UNIX wildcard sense. I.e. * means zero or more characters.

BOUNDARY CALCULATION

The boundary values are computed by using two functions located in two separate m files. The first m file calculates the (x, y) coordinates of the boundary nodes of the grid. It then calls the other m file which contains the function to be used to calculate the value of the numerical solution on the boundary nodes given their (x, y) positions at time t.

In `GDPdirichlet.m` example the function that calculates the (x, y) coordinates of grid-nodes is called `computeBoundary.m`. The function that `computeBoundary.m` calls is named `boundaryFunction.m`. `boundaryFunction.m` contains function $\sin(6\pi(x-y))$ to be used in the calculation of the numerical solution for (x, y) position at time t.

`boundaryFunction.m` is designed to operate on a set of values, rather than individual (x, y) positions to reduce the computation time. A user should not feel the need to edit the `computeBoundary.m` file. This is due to the fact that the Toolbox of Level Set Methods 1.1 works only with rectangular domains without "holes", and changing the `computeBoundary.m` would change the look of the domain. The `boundaryFunction.m` in the `GDPdirichlet.m` implementation is set up to hold the analytical solution to the problem, so it should not be changed either.

COMPARISON OF SOLUTIONS

The analytical solution to a problem is kept in a separate file. In the case of `GDPdirichlet.m` the solution is kept in the file `solutionGDP.m`. The function that actually does the comparison of the numerical and analytical solutions, `compareSols.m`, is called by `GDPdirichlet.m`. Any file containing an analytical solution can be edited by the user. However, the solution for the `GDPdirichlet.m` is set to the one solving the PDE (3.9), so the user should not feel the need to change it.

DERIVATIVE DISCRETIZATION

The discretization of the first and second order terms, $\partial_i, \partial_{ij}$, is computed by a separate function called `hessian*.m`. All of the `hessian*.m` functions follow the same layout. In the case of `GDPdirichlet.m` the file is called `hessianDegenerate.m`. `hessianDegenerate.m` computes the derivative approximations in two directions: bottom-left to top-right, and bottom-right to top-left. Bottom-left to top-right direction was used for PDE (3.9) where the analytical solution is $sin(6\pi(x-y))$. Bottom-right to top-left direction was used to generate the example with the analytical solution $sin(6\pi(x+y))$.

THE term FUNCTIONS*

There are several `term*.m` functions to be found in the Toolbox of Level Set Methods 1.1, and we have build ours following the same layout. The integrator functions can be found in `~/ToolboxLS-1.1/Kernel/ExplicitIntegration/Term` directory, where ~ is the directory into which the Toolbox of Level Level Set Methods 1.1 is installed. A `term*.m` function can be viewed as a driver of the numerical approximation. It is the file that repeatedly calls the discretization and integration routines. We mention it here since at a first glance its role in the implementation might be confusing to a novice user of the Toolbox of Level Set Methods 1.1.

A term function is also responsible for computing the CFL bound on the time step for every iteration.

Finally, a few notes on the integrator function used in the thesis even though the implementation of the integrator function was not a part of this thesis but was already provided with the Toolbox of Level Set Methods 1.1. In all the m file implementations of the thesis, the integrator function is `odeCFL1.m`, which approximates Forward Euler integration. A user could try to use a different integrator (all integrator functions can be found inside of `~/ToolboxLS-1.1/Kernel/ExplicitIntegration/Integrators` directory), but then the resulting system matrix would not be of positive type.

It is noticeable that `GDPdirichlet.m` is implemented in a way so as not to be changeable. So why then do we have separate files containing the analytical solution and the function to be used in the calculation of the boundary values when we could have written the analytical solution and the boundary function in `GDPdirichlet.m`.? We wanted to keep the same layout for all our implementations. Examples of Chapter 4 can all be tested on one m file implementation by changing the m files that contain the solution and the function to be used in the calculation of the boundary values. In this way we made the implementations easier for a user to understand.

Since the Toolbox of Level Set Methods 1.1 is designed to calculate the CFL condition and the derivative approximations at every time step, we had to follow this pattern.

The m files contain comments on the implementation of a numerical example, but not comments on the m language syntax. For questions regarding the m language syntax we refer the user to the MATLAB's online help website `http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html`, or the MATLAB help provided with a MATLAB distribution. In addition, a simple google search (`http://www.google.com`) can return many examples.

5.2 m file v.s. C language implementations

The monotone implementations of Problems 1., 2., 3., and 4. have been implemented both in the m files, as a part of the Toolbox of Level Set Methods 1.1, and as stand-alone C programs. The non-monotone examples exist only as m file implementations.

Some reasons for advocating a C implementation vs. an m file implementation stem from the fact that a C implementation is faster and requires less memory storage. In addition, the CFL condition computed in the C implementations is guaranteed to be the theoretical one. This bound is computed once, before the numerical solver is called. In this way, the time step is calculated only once and used in every iteration of the numerical solver. The arrays that contain the values of the system matrix are also calculated only once, as are the values of the boundary for examples where the calculation of the boundary values is time independent.

Contrary to this, the implementations of the Toolbox of Level Set Methods 1.1 compute a guess on the CFL bound at every time step. Also, the system matrix and the values on the boundary are computed at every time step. Moreover, the boundary is allocated and deallocated at every time step. This means additional storage space is required

to store the values that will be copied into the boundary at every time step once the boundary for that time step gets created. Finally, the passing of structures and arrays to a function written in the MATLAB's m language is a "pass by value" if the passed in argument(s) are changed within the function. This means that every array is copied into a new memory location when being passed to a function that changes it. Thus executing MATLAB m files can quickly appropriates a substantial amount of available memory, depending on the implementation and the size of the arrays.

We include the following comparison of the m files versus C language implementations with respect to memory and speed. An m language implementation of any of the explicit method examples presented in this thesis cannot be run on a 256MB RAM, Pentium 4, machine with 2GB of swap space if the grid has 100×100 nodes, due to thrashing. Yet a grid of 100×100 can be used on the same machine using the C implemetations without problems.

The domain in the Toolbox of Level Set Methods 1.1 can only be a square or a rectangle without "holes". The same holds for the domain in the current C language implementations. However, the C implementations are made in a way to allow a user-specified domain, but then the computation of the diagonal terms of the system matrix would have to be changed.

We note that the implicit Midpoint Runge-Kutta example is only implemented in C language.

5.3 Implementation of C files

The C implementations are not a requirement of this thesis. Therefore, their implementation will not be discussed here. They were done by the principal investigator as an aid in overcoming the memory and running time difficulties of the m file implementations. In effect, most of the results shown in this thesis are obtained using C language implementations. If, however, an interested reader would like to obtain a detailed description of the C language implementations as well as the implementations themselves, they should contact the principal investigator at `mirnalim@cs.ubc.ca`.

Chapter 6

Future work

• **Extension to three and n dimensions**

The work conducted in this thesis involves only two-dimensional problems. One can continue in the reseach of the importance of stencil directions for three and higher dimensional spaces.

The three-dimensional case, we expect, would present more difficulty to program, but not to draw given the MATLAB's graphing capabilities. It is uncertain what demands on programming and displaying would higher than three dimensions pose.

In order to test the implementation in three-dimensional space one would have to obtain an analytical solution to a three-dimensional problem. One could argue that a two dimensional example could be used by setting diffusion to be zero in one of the spatial dimensions. However, the question remains on whether two-dimensional examples can be utilized to test properly a three-dimensional implementation.

Further research could involve the implementation of the method presented by Oberman in [14], and its comparison to the method of Chapter 4. The idea of [14] is to use an equally spaced grid, but a wider stencil, thus allowing for more choices of the direction of discretization. The discretization direction which is closest to the direction of diffusion should be taken since then the central differences approximations will be aligned with the diffusion as much as possible. We did not implement the method of [14] for a two-dimensional case. Once implemented it could be easily compared to the method of varying step sizes given the examples of this thesis.

• **Implicit and implicit-explicit methods**

The Midpoint Runge-Kutta is the only implicit method implemented in this thesis research. We decided not to include it as part of the thesis due to the fact that it was our introduction into the implicit methods that has already been thoroughly researched in published work. Further research of implicit methods could involve implementation of another equation(s) using Midpoint or other implicit Runge-Kutta methods such as Hammer-Hollingsworth and Diagonal implicit Runge-Kutta. Since the implementation of the Midpoint Runge-Kutta was only written in C programming language, the next step would be to write a Toolbox of Level Set Methods 1.1 implementation.

It is known that implementations of implicit schemes take more time to run than

do the implementations of explicit schemes. Speed and memory requirements would need to be investigated when considering implicit implementations within the Toolbox of Level Set Methods 1.1.

Additional research could involve the implicit-explicit (IMEX) methods. The idea is to solve the second order terms in an equation via an implicit solver, and first order terms using an explicit solver. The reason being that the second order terms require a much smaller time step than do the first order terms if solved using an explicit solver. So to have as big of a time step as possible in an IMEX scheme, the second order terms could be solved by using an implicit method, while the first order terms by using an explicit method. The work of Ascher, et. al. [19] develops several Runge-Kutta based IMEX schemes.

The issues remaining to be considered are the running time and memory requirements of IMEX implementations. Since we have not implemented an IMEX scheme we cannot claim any findings. C language implementations would give insight into the speed of IMEX schemes since in C, unlike in MATLAB, pointers to arrays can be passed between functions, and so the running time would be that of the underlying solver rather than that of the solver plus time for memory allocations and deallocations.

Bibliography

[1] author: I. Mitchell, Assistant Professor, *Toolbox of Level Set Methods 1.1*, Department of Computer Science, University of British Columbia, `http://www.cs.ubc.ca/ mitchell/ToolboxLS/index.html`

[2] A. M. Oberman, *Convergent difference schemes for nonlinear elliptic and parabolic equations: Hamilton-Jacobi equations and free boundary problems*, SINUM, 44: No. 2: 879-895, 2006

[3] D. H. Anderson, *Compartmental Modeling and Tracer Kinetics*, Lecture Notes in Biomathematics, vol. 50, Springer-Verlag, Berlin, 1983

[4] T. S. Motzkyn, W. Wasov, *On the approximation of linear elliptic differential equations by difference equations with positive coefficients*, J. Math. Phys. 31: 253-259, 1953

[5] G. Barles, P. E. Souganidis, *Convergence of approximation schemes for fully nonlinear second order equations*, Asymptotic Anal., 4(3):271-283, 1991

[6] Class notes from the course *Numerical methods* taught by U. Ascher, University of British Columbia, 2005-2006

[7] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, *A hybrid particle level set method for improved interface capturing*, J. Comp. Physics, 183:83-116, 2002

[8] J. C. Strikwerda, Finite Difference Schemes and Partial Differential Equations, Wadsworth & Brooks/Cole, 1989

[9] W. E. Boyce, R. C. DiPrima, Elementary Differential Equations and Boundary Value Problems, 7. ed., John Wiley & Sons, 2001

[10] G. Strang, Linear Algebra and its Applications, 3. ed., Brooks/Cole, 1988

[11] U. M. Ascher, L. R. Petzold, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, SIAM, 1998

[12] W. H. Press, W. T. Vetterling, S. A. Teukolsky, B. P. Flannery, Numerical Recipies in C++: The Art of Scientific Computing, 2. ed., 2002

[13] J. Bramble, B. Hubbard, *A theorem on error estimation for finite difference analogues of the dirichlet problem for elliptic problems*, Contrib. Diff. Eq., 2:319-340, 1963

[14] A. M. Oberman, *A convergent monotone difference scheme for motion of level sets by mean curvature*, Numer. Math., 99:365-379, 2004

[15] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer, 2003

[16] M. Spiegel, Fourier analysis with applications to boundary value problems (Shaum's outline series), McGraw-Hill Companies, 1974

[17] S. M. A. Bruin, *seminar talk*, `www.win.tue.nl/casa/meetings/seminar/previous/runge-kutta/talksem.pdf`, 2002

[18] M. S. Gockenbach, Partial Differential Equations, Analytical and Numerial Methods, SIAM, 2002

[19] U. A. Ascher, S. J. Ruuth, R. J. Spiteri, *Implicit- Explicit Runge-Kutta Methods for Time-Dependent Partial Differential Equations*, Appl. Numer. Math., 25(2-3):151-167, 1997

[20] N. LIMIĆ, M. ROGINA, *Explicit stable methods for second order parabolic systems*, Math. Commun. Vol 5, (2000), 97–115

[21] I.N. BRONSTEJN, K. A. SEMENDJAJEV, *Matematicki prirucnik za inzenjere i studente*, Tehnicka knjiga, Zagreb, 1991.

Appendix

The Toolbox of Level Set Methods 1.1 can be downloaded from
`http://www.cs.ubc.ca/ mitchell/ToolboxLS/index.html`
website together with the installation instructions.

The Toolboox of Level Set Methods 1.1 is a set of m files to be executed within a MATLAB process. MATLAB m files do not require compiling. In order to execute an m file type the name of the file followed by the list of arguments enclosed in parentheses at the MATLAB prompt. The following is an example call to a function stored in file `filename.m` which takes arguments $\texttt{arg}_1$ to $\texttt{arg}_n$:
`filename(`$\texttt{arg}_1$`, `$\texttt{arg}_2$`, ..., `$\texttt{arg}_n$`);`

The description of the arguments accepted by a function is located at the beginning of the file that contains the function itself. It is important to note that if the program called does not reside in the current directory (i.e. the directory the user is currently in), the entire path to that program needs to be specified, or the directory in which the program resides must be added to the MATLAB's search path using `addpath()`. Fo example, to run an m file named `filename.m` from outside of the directory in which it resides one would execute
`/path_to_toolbox/ToolboxLS-1.1/Examples/Basic/filename(`$\texttt{arg}_1$`, ..., `$\texttt{arg}_n$`);`
or one could store in a file, `path_file`, the following command
`addpath(genpath('/path_to_toolbox/ToolboxLS-1.1/path_to_desired_directory'));`
and then from a subdirectory call
`run('path_to_path_file/path_file');`
where `path_to_toolbox` and `path_to_path_file` are the absolute paths from the root directory, /, to the directory containing the Toolbox of Level Set Methods 1.1, and the `path_file`, respectively.

Before we list the programs developed as a part of this thesis we would like to comment on the writing conventions used in order to simplify the table listing of the m file implementations.

We will assume that the user has installed the Toolbox of Level Set Methods 1.1 in directory *dir*. Then, the m file implementation of Heat equation (Problem 1) is stored in
`dir/ToolboxLS-1.1/Examples/Basic/p1/`
directory. The m file implementation of Degenerate heat equation (Problem 2) is contained in
`dir/ToolboxLS-1.1/Examples/Basic/p2/`

directory, and those of General degenerate parabolic equation (Problem 3) in
`dir/ToolboxLS-1.1/Examples/Basic/p3/`
directory. The m file implementation of General degenerate parabolic equation with advection (Problem 4) can be found in directory
`dir/ToolboxLS-1.1/Examples/Basic/p4/` .

Having said that, we write program filenames in Table Ā.1 without specifying the path to them. We will also use abreviations of PDE problems

- H for Heat equation,
- DH for Degenerate heat equation,
- GDP for General degenerate parabolic equation,
- GDPĀ gor General degenerate parabolic equation with advection.

Problems Ā-F each fall into one of the above PDE problems

- Ā, B are under DH,
- C, D, E are under GDP,
- F is under GDPĀ.

When specifying file names in the table, we use asterisk (*) in the usual UNIX wildcard sense. I.e. * means zero or more characters. USER identifies the user of a MĀTLĀB session who initiates an execution of a program.

When both numerical and analytical solutions to a problem are graphed, then two separate graphs are used where at a certain time t_1, one graph shows the numerical solution at t_1, while the other graph shows the analytical solution at t_1. Otherwise, only the numerical solution is graphed.

Table A.1 list of m (MATLAB) files implemented in the Toolbox of Level Set Methods 1.1.

Program Name	PDE Problem	Is called by
heat2d.m	H	USER
heat2dSolution.m	H	heat2d.m
hd.m	DH	USER
hdsConstant.m	DH	hd.m
hdsVortex.m	DH	hd.m
addGhostFunction2D.m	GDP	GDP*.m
computeBoundary.m	GDP, GDPA	GDP*.m
boundaryFunction.m	GDP	GDPdirichlet.m
boundaryFunctionMono.m	GDP	GDPmono.m
GDPdirichlet.m	GDP	USER
GDPmono.m	GDP	USER
GDPperiodic.m	GDP	USER
hessianDegenerate.m	GDP	GDPdirichlet.m, GDPperiodic.m
hessianMono.m	GDP	GDPmono.m
solutionGDP.m	GDP	GDPdirichlet.m, GDPperiodic.m
solutionGDPA.m	GDP	GDPdirichlet1.m
solutionMono.m	GDP	GDPmono.m
termDegenerate.m	GDP	GDPdirichlet.m, GDPperiodic.m
termMono.m	GDP	GDPmono.m
GDPdirichlet1.m	GDPA	USER
computeBoundaryt.m	GDPA	GDPdirichlet1.m
boundaryFunction1.m	GDPA	GSPdirichlet1.m
compareSols.m	ALL	ALL called by USER
visualizeLevelSet1.m	ALL	ALL called by USER