DIPLOMARBEIT

Titel der Diplomarbeit

Numerical Methods For Parabolic Equations

Angestrebter akademischer Grad

Magister der Naturwissenschaften (Mag. rer. nat.)

Verfasser:	Arno Mayrhofer
Matrikel-Nummer:	0504812
Studienrichtung:	A 405 Mathematik
Betreuer:	Univ.-Prof. Dr. Herbert Muthsam

Wien, im Juli 2010

Abstract

The diffusion equation

$$\frac{\partial u}{\partial t} + \nabla \cdot (\kappa \nabla u) = f$$

shows up in the description of a multitude of physical systems. Due to their complexity most of them cannot be solved analytically and thus numerical methods have to be employed. Depending on the problem the computation time can be several weeks resulting in the need for efficient algorithms.

The traditional methods, e.g. the Forward Time Central Space algorithm, for solving the diffusion equation are well known for their strict time-stepping restrictions. In this thesis a new class of methods, named Smoothed Essentially Non-Oscillatory (SENO) schemes, will be implemented. By means of numerical simulations it will be shown that these restrictions are surpassed considerably.

Before giving specific information about the SENO algorithm and the numerical methods involved the diffusion equation is examined in detail. For this several analytical properties of the equation and its solutions are discussed and illustrative proofs are given for the special case of the one dimensional heat equation.

The main idea of the SENO methods is to treat the diffusion equation as a conservation law. This leads to the possibility of using Weighted Essentially Non-Oscillatory (WENO) methods to calculate the spatial discretization. Further enhancements are obtained by a least squares spline approximation which provides a smoothing effect. The most significant feature is that the smoothing only takes place inside the truncation error interval which can be calculated analytically. An additional preconditioning step that consists of a weighted average, similar to the one used for the exact solution, provides even more improvements. Furthermore it is advantageous to incorporate certain analytical properties of the solution into the numerical method. The time integration will be performed by a second order Runge–Kutta algorithm commonly known as Heun's method.

To demonstrate the effectiveness of the algorithm several simulations in one and two dimensions will be shown and analyzed. Included is a complete series of simulations that illustrate the evolution of the algorithm. Two representative initial conditions in 2D are utilized for this series. They are also used for showing the optimality of certain parameters of the SENO method. Additional simulations are conducted to point out possible directions of future research.

Zusammenfassung

Bei der Beschreibung physikalischer Systeme tritt die Diffusionslgeichung

$$\frac{\partial u}{\partial t} + \nabla \cdot (\kappa \nabla u) = f$$

häufig auf. Aufgrund der Komplexität dieser Systeme ist es meistens nicht möglich sie analytisch zu lösen, weshalb numerische Verfahren angewendet werden müssen. Abhängig von dem Problem ist es wichtig, effiziente Algorithmen zu entwickeln, um die Rechenzeit in einem vertretbaren Rahmen zu halten.

Traditionelle Methoden, wie etwa der Vorwärts Euler Algorithmus, die üblicherweise für das Lösen der Diffusionsgleichung verwendet werden, schränken bekanntlich die Zeitschrittweite sehr stark ein, so dass sie in diesem Sinn als unbrauchbar gelten müssen. In dieser Diplomarbeit wird eine neue Klasse von Methoden, genannt Smoothed Essentially Non-Oscillatory (SENO), konstruiert, welche diese Grenzen signifikant erweitern.

Bevor dieser Algorithmus zusammen mit den dazu notwendigen numerischen Methoden eingeführt wird, wird die Diffusionsgleichung eingehend untersucht. Dazu werden mehrere analytische Eigenschaften der Gleichung und ihrer Lösungen aufgeführt. Mehrere Beweise für den Spezialfall der eindimensionalen Wärmeleitungsgleichung illustrieren diese charakteristischen Eigenschaften.

Die Grundidee für die SENO Methoden ist die Diffusionsgleichung als Erhaltungssatz zu sehen um darauf Weighted Essentially Non-Oscillatory (WENO) Methoden anzuwenden, die die örtliche Diskretisierung berechnen. Zusätzliche Verbesserungen werden durch das Lösen eines Spline Approximations Problems auf Basis der Methode der kleinsten Quadrate erzielt, was ein Glätten der Lösung bewirkt. Ein zusätzlicher konditionierungs Schritt, bestehend aus einer gewichteten Mittelung ähnlich der der exakten Lösung, verschafft eine zusätzliche Verbesserung. Außerdem ist es vorteilhaft gewisse analytische Eigenschaften der Lösung miteinzubeziehen. Für die Zeitintegration wird ein Runge–Kutta Verfahren zweiter Ordnung verwendet, welches unter dem Namen Heun's Methode bekannt ist.

Um die Effizienz des neuen Algorithmus zu demonstrieren werden mehrere Simulationen in einer und zwei Raumdimensionen gezeigt und analysiert. Ein Teil der Simulationen zeigt die komplette Evolution des Algorithmus anhand zweier speziell ausgewählter Anfangsbedingungen in 2D. Diese werden außerdem herangezogen um die Optimalität gewisser Parameter zu zeigen, welche für die SENO Methoden verwendet werden. Zusätzliche Simulationen werden abschließend benutzt um mögliche Richtungen zukünftiger Forschung aufzuzeigen.

Danksagung

Bevor der wissenschaftliche Teil dieser Diplomarbeit beginnt, möchte ich die Gelegenheit nutzen mich bei mehreren Personen zu bedanken. Ein besonderer Dank gilt Herrn Professor Herbert J. Muthsam, der das Thema vorgeschlagen hat und diese Arbeit aktiv betreut hat. Insbesondere möchte ich mich für die vielen Ratschläge und die hilfreiche Anleitung bedanken, ebenso wie für die Eingliederung in die Forschungsgruppe ACORE.

Aus dieser Gruppe ist besonders Herr Privatdozent Friedrich Kupka hervorzuheben, der mich ebenfalls von wissenschaftlicher Seite tatkräftig unterstützt hat. Auch dem Rest der Gruppe möchte ich für die freundliche Aufnahme und Zusammenarbeit danken. Das betrifft insbesondere Herrn Priv.-Doz. Othmar Koch und Herrn Mag. Hannes Grimm-Strele welche mir zusätzlich noch mit dem Korrekturlesen geholfen haben. Weitere Unterstützung in Form von Verbesserungsvorschlägen bekam ich von Herrn Andrew P. Rogers wofür ich mich ebenfalls herzlich bedanken möchte.

Der größte Dank soll meinen Eltern zuteil werden, denen ich auch diese Arbeit widmen möchte. Ohne ihre vielfältige Unterstützung während meiner Studienjahre wäre nicht nur dieses Werk nicht entstanden, sondern mir wären auch unzählige andere Wege verschlossen geblieben.

Contents

1 Introduction **1**

2 Parabolic Partial Differential Equations **3**
- 2.1 Basic Definitions . . . 3
- 2.2 The Diffusion Equation . . . 5
  - 2.2.1 Physical Derivation of the Heat Equation . . . 5
- 2.3 Important Analytical Results . . . 6
  - 2.3.1 Exact Solution in 1D . . . 7
  - 2.3.2 Smoothness of Solutions . . . 8
  - 2.3.3 Maximum Principle . . . 9
  - 2.3.4 Uniqueness . . . 10
  - 2.3.5 Periodicity . . . 10

3 Numerical Methods **13**
- 3.1 Forward Time Central Space . . . 13
  - 3.1.1 The Courant Number . . . 14
- 3.2 Heun's Method . . . 15
- 3.3 Weighted Essentially Non-Oscillatory Schemes . . . 17
- 3.4 Truncation Error of WENO coupled with Heun's method . . . 20
- 3.5 Cholesky Decomposition . . . 23
- 3.6 Spline Approximation . . . 23
- 3.7 Smoothed Essentially Non-Oscillatory Schemes . . . 26
  - 3.7.1 In One Dimension . . . 26
  - 3.7.2 In n Dimensions . . . 30

4 Simulations **33**
- 4.1 Simulations in 2D . . . 33
- 4.2 Simulations in 1D . . . 44

5 Conclusion and Outlook **49**

Curriculum Vitae **53**

Chapter 1

Introduction

> The differential equations of the propagation of heat express the most general conditions, and reduce the physical questions to problems of pure analysis, and this is the proper object of theory.
>
> *Joseph Fourier (1768 - 1830)*, [1]

Although Fourier was certainly right in his time it would nowadays be necessary to add "and applied" after the word "pure". It will be this mixture of pure and applied mathematics which will be the object of this thesis.

The main focus will not lie on the heat equation itself but rather on its generalization, the diffusion equation

$$\frac{\partial u}{\partial t} + \nabla \cdot (\kappa \nabla u) = f.$$

This equation shows up in a large number of physical systems, e.g. in the description of the interior dynamics of a star. These systems usually consist of multiple coupled partial differential equations which makes most of them unsolvable analytically. However using today's high performance computers it is possible to obtain approximate solutions. Nevertheless computations still take considerable time and thus efficient algorithms are of importance.

In this work a new class of methods for solving the diffusion equation will be implemented and discussed in detail. According to their origin they will be called Smoothed Essentially Non-Oscillatory (SENO) schemes. Their main goal is to overcome the strict time-stepping restrictions usually present for traditional explicit solvers as for example the forward time central space algorithm. Nevertheless the new schemes should maintain a local and explicit character.

The work is divided into the following chapters.

- After this introduction, Chapter 2 will give an overview of parabolic partial differential equations with special emphasis on the diffusion equation. This theoretical chapter will also discuss several analytical properties of the equation and its solutions.

- Chapter 3 will then introduce the numerical methods which are necessary for implementing the SENO schemes. Amongst them are Runge–Kutta methods, Weighted Essentially Non-Oscillatory schemes and spline approximation techniques. The chapter will finish with an in-depth description of the SENO methods in one and higher dimensions.
- In the following Chapter 4 several simulations will be shown, illustrating the evolution and capabilities of the SENO algorithm. For this purpose multiple simulations were conducted in one and two dimensions.
- Finally the thesis will be concluded in Chapter 5, where the obtained results are summarized and possible directions of future research are pointed out.

Chapter 2

Parabolic Partial Differential Equations

Partial differential equations (PDEs) are of huge importance when describing problems arising in physics. What is more, their complexity makes them interesting for a wide range of mathematical research. Since usually PDEs cannot be solved analytically, numerical methods have to be employed to approximate solutions. However, it is possible to derive analytical properties of solutions without actually knowing them. As will be shown later on this can be of help when designing numerical methods.
In the following a short introduction to PDEs will be given with special emphasis on the diffusion equation. After deriving the equation from physical principles, certain analytical properties of the solutions shall be studied that will be important for the later numerical treatment.

2.1 Basic Definitions

A PDE is an equation

$$F(x, u(x), Du(x), \dots, D^k u(x)) = 0 \qquad \forall\, x \in \Omega, \tag{2.1.1}$$

where

$$u : \Omega \to \mathbb{R}^n, \tag{2.1.2}$$

is the unknown and Ω an open subset of $\mathbb{R}^n$.
Let α be a multi-index then $D^\alpha := \partial_{x_1}^{\alpha_1} \cdots \partial_{x_n}^{\alpha_n}$ and if k is a non-negative integer then $D^k := \{D^\alpha u(x) || \alpha| = k\}$.
If F depends linearly on u and its derivatives the PDE is called *linear*. The *order* of a PDE is given by the order of the highest derivative occurring in F.

In the following we will only consider linear PDEs of second order. Their general form is

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial}{\partial x_i}\left(a_{i,j}\frac{\partial u}{\partial x_j}\right)+\sum_{i=1}^{n}b_i\frac{\partial u}{\partial x_i}+cu+f=0, \tag{2.1.3}$$

where the coefficients $a_{i,j}, b_i, c$ and f all depend on the independent variables $x_1, \dots, x_n$. The most important PDEs of the above form can be further classified into the following three types:

- *Elliptic:* All eigenvalues of the coefficient matrix $a_{i,j}$ are either positive or negative.
- *Hyperbolic:* Either one eigenvalue of the coefficient matrix is positive and the rest are negative or alternatively one is negative and the others are positive.
- *Parabolic:* One eigenvalue is zero and the rest are either all positive or negative.

Solutions u of the PDEs which we will consider lie in so-called Sobolev spaces. To define these spaces we need the formulation of weak derivatives. Let α be a multi-index and u an integrable (L^1) function. Then v ($=: D^\alpha u$) is called the α-th weak derivative of u if it is contained in L^1 and

$$\int_\Omega u D^\alpha \phi = (-1)^{|\alpha|}\int_\Omega v\phi \quad \forall \phi \in C_c^\infty \tag{2.1.4}$$

holds. The space C_c^∞ is the space of all infinitely differentiable functions with compact support contained in Ω, commonly known as test functions. A Sobolev space H^m is defined as a subset of all square integrable functions (L^2) for which all weak derivatives up to order m are again contained in L^2. The order of a weak derivative D^α is equivalent to the absolute value of the multi-index α. Every Sobolev space H^m has a corresponding norm which is defined by

$$\|u\| = \left(\sum_{|\alpha|\le m}\int |D^\alpha u(x)|^2 dx\right)^{1/2}. \tag{2.1.5}$$

H_0^m is the closure of C_c^∞ in H^m with respect to the above norm.
In the rest of this work we will restrict ourselves to parabolic PDEs, or more specifically to the diffusion equation that will be defined in the next section.

2.2 The Diffusion Equation

The *inhomogeneous diffusion equation* is defined as

$$\frac{\partial u}{\partial t} + \nabla \cdot (\kappa \nabla u) = f \qquad \forall (t, x_1, \dots, x_n) \in [0,T] \times \Omega, \tag{2.2.1}$$

where

$$(\nabla)_i = \tfrac{\partial}{\partial x_i} \qquad i = 1, \dots, n, \tag{2.2.2}$$

$$u, \kappa, f : \qquad [0,T] \times \Omega \to \mathbb{R}, \tag{2.2.3}$$

$T > 0$ and $\Omega \subseteq \mathbb{R}^n$.

The function u represents the density of a diffusing material (e.g. temperature), κ is the diffusion coefficient and f describes any external influence on the system. In the following we will limit ourselves to the homogeneous equation, i.e. $f \equiv 0$. Note that it will be necessary to impose certain restrictions upon the diffusion coefficient κ when it comes to proving specific results. In case where κ is a constant greater than zero this equation is commonly known as the *heat equation* [2]

$$\frac{\partial u}{\partial t} - \kappa \Delta u = 0. \tag{2.2.4}$$

Comparing (2.1.3) with (2.2.1) it can easily be seen that the coefficient matrix $a_{i,j}$ is equal to the Identity matrix I_{n+1} but with $a_{1,1} = 0$. Thus one eigenvalue of this matrix is zero whereas the rest is one which shows that this equation clearly is an example of a parabolic PDE.

However this equation has infinitely many solutions. To get a unique solution it is necessary to introduce additional conditions. We define an initial condition for the problem by setting

$$u(0,x) = u_0(x) \quad \forall x \in \Omega. \tag{2.2.5}$$

Again restrictions on u_0 will be required. In addition, the analytical results will require the boundary conditions

$$u(t,x) = 0 \quad \forall (t,x) \in [0,T] \times \partial\Omega. \tag{2.2.6}$$

2.2.1 Physical Derivation of the Heat Equation

In this section a short derivation of the heat equation will be given, based on mathematical [2] and physical [3] principles.

As a consequence to the first law of thermodynamics the internal energy Q contained inside a volume Ω is proportional to the temperature u. In mathematical terms this reads

$$Q(t) = \int_{\Omega} \kappa_1 u(x,t) dV, \tag{2.2.7}$$

with some material dependent constant κ_1. Subsequently the change of internal energy is given by

$$\frac{dQ}{dt}(t) = \frac{d}{dt} \int_{\Omega} \kappa_1 u(x,t) dV. \tag{2.2.8}$$

On the other hand the rate of change of any quantity enclosed inside a volume Ω is equal to the flux F of this quantity through the boundary $\partial\Omega$. Thus we additionally have

$$\frac{dQ}{dt}(t) = -\int_{\partial\Omega} F(x,t) \nu dS, \tag{2.2.9}$$

where ν is the outward normal of $\partial\Omega$.
The flux F in a thermodynamical system is described by Fourier's law which states that

$$F(x,t) = -\kappa_2 \nabla u(x,t), \tag{2.2.10}$$

where κ_2 is another material dependent constant (possibly dependent on x).
Combining the above equations yields

$$\int_{\Omega} \kappa_1 \frac{\partial u}{\partial t}(x,t) dV = \frac{dQ}{dt}(t) = -\int_{\partial\Omega} F(x,t) \nu dS = \int_{\partial\Omega} \kappa_2 \nabla u(x,t) \nu dS. \tag{2.2.11}$$

Applying the divergence theorem to the surface integral gives

$$\int_{\Omega} \kappa_1 \frac{\partial u}{\partial t}(x,t) dV = \int_{\Omega} \nabla(\kappa_2 \nabla u(x,t)) dV. \tag{2.2.12}$$

Due to Ω being arbitrary we conclude that

$$\frac{\partial u}{\partial t} = \nabla(\kappa \nabla u), \tag{2.2.13}$$

with an appropriate constant κ.

2.3 Important Analytical Results

In this section we will review the most important results for solutions of the diffusion equation. To illustrate the properties we will prove them for the one dimensional case assuming constant κ. The higher dimensional case will only be stated and for the proofs the reader is referred to [2]. As a general guide the ideas presented in [4] will be used.

2.3.1 Exact Solution in 1D

Consider the heat equation (2.2.4) in one dimension. Applying the Fourier transformation with respect to the spatial variable x to this equation we get

$$\frac{\partial \widehat{u}}{\partial t} = -\kappa\omega^2 \widehat{u}. \tag{2.3.1}$$

If the initial condition at time $t = 0$ is $u_0(x)$ we get the solution

$$\widehat{u}(t,\omega) = \mathrm{e}^{-\kappa\omega^2 t}\widehat{u}_0(\omega) \tag{2.3.2}$$

by solving the ordinary differential equation (2.3.1).
To obtain the solution in the original space we use the Fourier inversion formula which yields

$$u(t,x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{e}^{i\omega x} \mathrm{e}^{-\kappa\,\omega^2 t} \widehat{u}_0(\omega) d\omega. \tag{2.3.3}$$

At this point we can already see a distinct property of the solution. Since ω represents the frequency, the $\mathrm{e}^{-\kappa\,\omega^2 t}$ term obviously damps high frequencies of the initial condition. Thus the parabolic operator is called dissipative.
To derive further representations of the exact solution we will need the result below.

Lemma 2.3.1. *For all $t > 0$ the following equation holds:*

$$\frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \mathrm{e}^{i\omega(x-y)} \mathrm{e}^{-\kappa\,\omega^2 t} d\omega = \frac{1}{\sqrt{\kappa t}} \mathrm{e}^{-(x-y)^2/4\kappa t}. \tag{2.3.4}$$

Proof. Starting out with the left hand side of equation (2.3.4) and letting $a := i\omega(x - y) - \kappa\omega^2 t$ we then have

$$\frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \mathrm{e}^{i\omega(x-y)} \mathrm{e}^{-\kappa\,\omega^2 t} d\omega = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \mathrm{e}^{a} d\omega. \tag{2.3.5}$$

Substituting ω by $\xi + \frac{i(x-y)}{2\kappa t}$ the integrand in the equation above does not change with the exception of a. We have

$$\begin{aligned} a &= i(x-y)\left(\xi + \frac{i(x-y)}{2\kappa t}\right) - \kappa t \xi^2 - \xi i(x-y) + \frac{(x-y)^2}{4\kappa t} && (2.3.6)\\ &= -\frac{(x-y)^2}{2\kappa t} + \frac{(x-y)^2}{4\kappa t} - \kappa t\xi^2 && (2.3.7)\\ &= -\frac{(x-y)^2}{4\kappa t} - \kappa t \xi^2, && (2.3.8) \end{aligned}$$

which inserted into (2.3.5) yields

$$\begin{aligned}(2.3.5) &= \frac{1}{\sqrt{\pi}}\mathrm{e}^{-\frac{(x-y)^2}{4\kappa t}}\int_{-\infty}^{\infty}\mathrm{e}^{-\kappa t\xi^2}d\xi && (2.3.9)\\ &= \frac{1}{\sqrt{\pi}}\mathrm{e}^{-\frac{(x-y)^2}{4\kappa t}}\sqrt{\frac{\pi}{\kappa t}} && (2.3.10)\\ &= \frac{1}{\sqrt{\kappa t}}\mathrm{e}^{-\frac{(x-y)^2}{4\kappa t}}, && (2.3.11)\end{aligned}$$

concluding the proof. □

To get an even better view on the solution let us apply the Fourier inversion again, this time on $\widehat{u}_0$. This gives

$$\begin{aligned}u(t,x) &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\mathrm{e}^{i\omega x}\mathrm{e}^{-\kappa\omega^2 t}\left(\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\mathrm{e}^{i\omega y}u_0(y)dy\right)d\omega && (2.3.12)\\ &= \frac{1}{\sqrt{4\pi}}\int_{-\infty}^{\infty}\left(\frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty}\mathrm{e}^{i\omega(x-y)}\mathrm{e}^{-\kappa\omega^2 t}d\omega\right)u_0(y)dy. && (2.3.13)\end{aligned}$$

Applying Lemma 2.3.1 finally provides us with the exact solution of the 1D heat equation

$$u(t,x) = \frac{1}{\sqrt{4\pi\kappa t}}\int_{-\infty}^{\infty}\mathrm{e}^{-\frac{(x-y)^2}{4\kappa t}}u_0(y)dy. \qquad (2.3.14)$$

2.3.2 Smoothness of Solutions

If we consider equation (2.3.14) we can see that the solution at time t is given as a weighted average of u_0. The weighting function $\mathrm{e}^{-\frac{(x-y)^2}{4\kappa t}}$ gets wider as t grows larger and thus we expect that the initial function gets smoothed out. In fact if we differentiate (2.3.3) with respect to t l times and with respect to x m times we obtain

$$\frac{\partial^{l+m}u(t,x)}{\partial t^l\partial x^m} = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\mathrm{e}^{i\omega x}(i\omega)^m(-\kappa\omega^2)^l\mathrm{e}^{-\kappa\omega^2 t}\widehat{u}_0(\omega)d\omega. \qquad (2.3.15)$$

Clearly the derivative is continuous for all l, m if $t > 0$. Furthermore it is possible to use this representation to derive an upper bound on the derivatives.
The following theorem gives a statement about the smoothness of solutions for the homogeneous diffusion equation in case that $\kappa > 0$.

Theorem 2.3.2. *Assume*

$$u_0 \in H^{2m+1}(\Omega) \textit{ and } \nabla(\kappa\nabla u_{k-1}) =: u_k \in H_0^1(\Omega). \qquad (2.3.16)$$

Then

$$\frac{d^k u}{dt^k} \in L^2((0,T];H^{2m+2-2k})(\Omega) \quad \forall k=0,\dots,m+1, \tag{2.3.17}$$

with the estimate

$$\sum_{k=0}^{m+1} \left\| \frac{d^k u}{dt^k} \right\|_{L^2((0,T];H^{2m+2-2k})(\Omega)} \leq C\|u_0\|_{H^{2m+1}(\Omega)}, \tag{2.3.18}$$

where C only depends on m, Ω, T and κ.

2.3.3 Maximum Principle

Consider again equation (2.3.14). Let us look at the supremum of the function u. We get

$$\begin{aligned}
\sup_{(t,x)\in[0,T]\times\Omega} u(t,x) &= \sup_{(t,x)\in[0,T]\times\Omega} \frac{1}{\sqrt{4\pi\kappa t}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x-y)^2}{4\kappa t}} u_0(y)dy && (2.3.19)\\
&\leq \sup_{(t,x)\in[0,T]\times\Omega} \frac{1}{\sqrt{4\pi\kappa t}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x-y)^2}{4\kappa t}} \sup_{z\in\Omega}(u_0(z))dy && (2.3.20)\\
&= \sup_{z\in\Omega}(u_0(z)) \sup_{(t,x)\in[0,T]\times\Omega} \underbrace{\frac{1}{\sqrt{4\pi\kappa t}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x-y)^2}{4\kappa t}} dy}_{1} && (2.3.21)\\
&= \sup_{z\in\Omega}(u_0(z)). && (2.3.22)
\end{aligned}$$

Trivially the same is true for the infimum. Thus the supremum / infimum of $u(t,x)$ is attained at $t=0$. This again seems natural if we consider $u(t,x)$ as weighted average of $u_0(x)$.

To extend this property to higher dimensions we have to assume that $\kappa \in C^1([0,T]\times\Omega)$. In the following let $\Omega_T =]0,T]\times\Omega$ and $\Gamma_T = \partial\Omega_T$.

Theorem 2.3.3. *Let $u \in C_1^2(\Omega_T)\cap C(\overline{\Omega}_T)$ then the* weak maximum principle *holds, i.e.*

$$\max_{\overline{\Omega}_T} u = \max_{\overline{\Gamma}_T} u \tag{2.3.23}$$

and

$$\min_{\overline{\Omega}_T} u = \min_{\overline{\Gamma}_T} u. \tag{2.3.24}$$

However, there is an even more powerful result.

Theorem 2.3.4. *Under the assumptions of the previous theorem and assuming that Ω is connected the* strong maximum principle *holds. It states that u is constant on $\{0\}\times\Omega$ if it attains its minimum or maximum over $\overline{\Omega}_T$ somewhere in Ω_T.*

2.3.4 Uniqueness

As a direct result from the maximum principle we get the uniqueness of the solution. Consider two different solutions u_1, u_2 of the diffusion equation with identical initial and boundary conditions. Let $u = u_1 - u_2$. This is again a solution of the diffusion equation since it is linear. Moreover the initial condition and the boundary conditions are identically zero. This means that the minimum and maximum of the set $\overline{\Gamma}_T$ is zero as well. Due to Theorem 2.3.23 we thus have that the maximum and minimum of u is zero everywhere and thus it is constant zero, yielding $u_1 = u_2$.

2.3.5 Periodicity

Let the domain $\Omega = \mathbb{R}$ and u_0 be a periodic function with period ξ. Starting out with equation (2.3.14) we have

$$u(t, x+\xi) = \frac{1}{\sqrt{4\pi\kappa t}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x+\xi-y)^2}{4\kappa t}} u_0(y)dy. \tag{2.3.25}$$

Substituting y with $z + \xi$ yields

$$\begin{aligned} u(t, x+\xi) &= \frac{1}{\sqrt{4\pi\kappa t}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x+z)^2}{4\kappa t}} u_0(z+\xi)dz & (2.3.26)\\ &= \frac{1}{\sqrt{4\pi\kappa t}} \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{(x+z)^2}{4\kappa t}} u_0(z)dz & (2.3.27)\\ &= u(t,x). & (2.3.28) \end{aligned}$$

This shows that the solution u is again periodic.
The same holds true in higher dimensions as $u(x+\xi)$ $(x, \xi \in \mathbb{R}^n)$ is again a solution of the diffusion equation if $u(x)$ is a solution and κ is also ξ-periodic. Due to the uniqueness of the solution they have to be identical and thus u is periodic.
A consequence of this result is that

$$\int_{x}^{x+\xi} u(y,t)dy = const \quad \forall x \in \mathbb{R}, t > 0. \tag{2.3.29}$$

This is shown by considering the time derivative of the above equation and applying the divergence theorem.

$$\begin{aligned} \frac{\partial}{\partial t}\int_x^{x+\xi} u(y,t)dy &= \int_x^{x+\xi} \frac{\partial}{\partial t}u(y,t)\ dy && (2.3.30)\\ &= \int_x^{x+\xi} \frac{\partial}{\partial y}\cdot(\kappa\frac{\partial}{\partial y}u(y,t))\ dy && (2.3.31)\\ &= \kappa\frac{\partial}{\partial y}u(y,t)\Big|_x^{x+\xi} && (2.3.32)\\ &= 0, && (2.3.33) \end{aligned}$$

where the last step follows from the periodicity of $u(y,t)$. Note that the constant in equation (2.3.29) only depends on the initial condition u_0.

Using the same calculation it can be shown that the result also holds true in arbitrary dimensions. If κ is not constant it is necessary to assume the same periodicity as for u_0.

Chapter 3

Numerical Methods

In this section we review several numerical methods that can be used to solve the diffusion equation. In Section 3.1 the very basic Forward Time Central Space method will be introduced. This is followed by describing Heun's method (3.2), a second order Runge–Kutta algorithm. Weighted Essential Non-Oscillatory Schemes (3.3) are another popular choice for solving PDEs numerically. They will, combined with Spline Approximation (3.6), build the basis for the new Smoothed Essentially Non-Oscillatory Schemes (SENO). They will be introduced in Section 3.7.

3.1 Forward Time Central Space

The derivative of a real-valued function f is defined as

$$\frac{df}{dx}(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}. \tag{3.1.1}$$

In a numerical setting we have our domain divided into a strictly monotonic increasing sequence of nodes x_i with constant spacing. Thus we usually only have the values of the function f available at these nodes. The simplest way of approximating the derivative is consequently

$$\frac{df}{dx}(x_i) \approx \frac{f(x_{i+1})-f(x_i)}{x_{i+1}-x_i}. \tag{3.1.2}$$

Subsequently the second derivative is

$$\frac{d^2f}{dx^2}(x_i) \approx \frac{f(x_{i+1})-2\,f(x_i)+f(x_{i-1})}{(x_{i+1}-x_i)^2}. \tag{3.1.3}$$

If we now insert this ansatz into the heat equation we get

$$\frac{u(t_{i+1},x_i)-u(t_i,x_i)}{\Delta t}=\kappa\,\frac{u(t_i,x_{i+1})-2\,u(t_i,x_i)+u(t_i,x_{i-1})}{\Delta x^2}, \tag{3.1.4}$$

where $\Delta t = t_{i+1} - t_i$ and $\Delta x = x_{i+1} - x_i$ for all i.
A simple calculation then shows that

$$u(t_{i+1},x_i)=u(t_i,x_i)+\frac{\kappa\,\Delta t}{\Delta x^2}\left(u(t_i,x_{i+1})-2\,u(t_i,x_i)+u(t_i,x_{i-1})\right). \tag{3.1.5}$$

This enables the calculation of the solution at time t_{i+1} by using only function values at time t_i. Thus starting with an initial condition $u(0,x_i) = u_0(x_i)$ it is possible to subsequently calculate the solution $u(t_i,x_i)\ \forall t_i > 0$. The truncation error of this method is of the order $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x^2)$.
The biggest drawback of this method is that it only provides useful solutions, i.e. is stable and convergent, if the coefficient $\frac{\kappa\,\Delta t}{\Delta x^2}$ is smaller than $\frac{1}{2}$.
It is straightforward to introduce non-constant coefficients (i.e. solve the diffusion equation) and expand this method to multiple dimensions. Thus we will use this method to calculate reference solutions to test the more complicated methods described in the following sections. For this we will usually set the coefficient $\frac{\kappa\,\Delta t}{\Delta x^2}$ to be $0.01/\kappa$.

3.1.1 The Courant Number

It turns out that the ratio between Δt and Δx^2 is of importance no matter which method we consider and this ratio is commonly known as the Courant number C. It usually gives an upper bound on the maximum time-step Δt since the method will become unstable otherwise.
As stated above the Courant number of the FTCS method has to be smaller than $\frac{1}{2}$ which can be proven analytically. However, for more complicated methods (e.g. nonlinear schemes) it often is not possible to obtain analytic bounds. Generally it can be said that higher order methods usually tend to allow lower Courant numbers since they are more prone towards oscillations. Secondly the Courant number is normally influenced by the coefficient κ. For constant κ we have

$$C=\frac{\kappa\,\Delta t}{\Delta x^2} \tag{3.1.6}$$

which in two dimensions becomes

$$C=\frac{\kappa\,\Delta t}{\Delta x^2}+\frac{\kappa\,\Delta t}{\Delta y^2}. \tag{3.1.7}$$

The goal of this thesis is to establish a new method for solving the diffusion equation that can achieve time-steps as large as possible. Although more complicated methods usually demand more computation time this might not seem to be that interesting on a first glance. But physical systems usually consist of systems of PDEs and only one PDE can lower the time-stepping restriction of the whole system. Thus an additional overhead can be justified under certain circumstances.

3.2 Heun's Method

Assuming that we have a method that provides us with suitable derivatives in spatial direction, time discretization is the last step to a full discretization. For this we will use a Runge–Kutta scheme with certain optimal features. A general explicit Runge–Kutta method for the PDE $u_t = L(u)$ can be written as

$$\begin{aligned} u^{(i)} &= \sum_{j=0}^{i-1} \left(\alpha_{ij} u^{(j)} + \Delta t \beta_{ij} L(u^{(j)}) \right) \quad \forall i = 1, \ldots, m, \qquad \textit{with} && (3.2.1) \\ u^{(0)} &= u_n(x) \qquad \textit{and} \\ u_{n+1}(x) &= u^{(m)}. && (3.2.2) \end{aligned}$$

The total variation of the numerical solution is defined as

$$TV(u_i) = \sum_j \left| u_{i,j+1} - u_{i,j} \right|. \tag{3.2.3}$$

Since we already know that our solution will be smooth we want that the variation decreases with time. Thus we want to have a method that is *total variation diminishing* (TVD), i.e.

$$TV(u_{n+1}) \leq TV(u_n). \tag{3.2.4}$$

If we restrict ourselves further to second order Runge–Kutta schemes and demand that we can choose the largest possible time-step and still get the TVD property, then there is only one choice of coefficients $\alpha_{ij} \geq 0$ and $\beta_{ij} \geq 0$ that satisfies these constraints [5]. This is the so called *Heun's method* which can be written using the form (3.2.1) as

$$\begin{aligned} u^{(1)} &= u_n + \Delta t \; L(u_n) && (3.2.5) \\ u_{n+1} &= \frac{1}{2} u_n + \frac{1}{2} u^{(1)} + \frac{1}{2} \Delta t \; L(u^{(1)}). \end{aligned}$$

This can be rewritten to a form more often encountered in literature

$$\begin{aligned} u^{(1)} &= u_n + \Delta t \; L(u_n) \\ u_{n+1} &= u_n + \frac{1}{2}\Delta t \left(L(u_n) + L(u^{(1)}) \right) . \end{aligned} \tag{3.2.6}$$

This schemes is often used in combination with TVD as well as TVB (total variation bounded) or (W)ENO ((Weighted) Essentially Non-Oscillatory) spatial discretization methods. In the next section we will present the WENO method and since it will form the basis for the final method this time-stepping algorithm is highly suitable.
Moreover it can be shown that Heun's method is also strong-stability preserving [6]. A method is called *strong-stability preserving* (SSP) in a given norm $\|.\|$ if

$$\|u_{n+1}\| \leq \|u_n\| , \tag{3.2.7}$$

assuming an appropriate time-step restriction.
If we define $\|u\|_{TV} = TV(u)$ this is actually a norm, so TVD means nothing else than SSP in the TVD norm.
The reason why we assumed the coefficients α_{ij} and β_{ij} to be non-negative is that Heun's method can then be written as convex combination of forward Euler steps. This ensures that the method is SSP in every norm in which the forward Euler method is SSP.
Although the method does not provide a minimum truncation error the SSP-optimality is of much higher value. The truncation error ϵ can be estimated as [6]

$$|\epsilon| \leq \frac{h^3}{3} C_1 C_2 \tag{3.2.8}$$

where

$$C_2 = 2 \left| \frac{3}{4\beta_{2,1}} - 1 \right| + 1 \tag{3.2.9}$$

and C_1 depends only on the differential operator L. Obviously the minimum truncation error is achieved when $\beta_{2,1} = \frac{3}{4}$. Since the coefficient $\beta_{2,1}$ in Heun's method is $\frac{1}{2}$ the truncation is two times larger than the smallest possible truncation error. We will discuss the truncation error in greater detail later on.

3.3 Weighted Essentially Non-Oscillatory Schemes

Weighted Essentially Non-Oscillatory Schemes (WENO) were first introduced by Liu, Osher and Chan [7] to solve hyperbolic conservation laws of the form

$$\frac{\partial u}{\partial t} + \nabla F(u) = 0, \tag{3.3.1}$$

where F is the so called flux. They are based on ENO (essentially non-oscillatory) schemes by Harten et al. [8] and were refined by Jiang and Shu in [9]. The latter paper will serve as a basis for the implementation of WENO schemes for spatial discretization. In the following we will limit the discussion to the fifth order accurate scheme.
Let us consider an unknown function $u(x)$ for which we would like to approximate $\nabla F(u)$. When dealing with WENO schemes in higher dimensions it will be shown that it is sufficient to use it in each spatial direction separately. Thus we will restrict this chapter to the case $x \in \mathbb{R}$.
To calculate the derivative we will use the second order central difference scheme

$$\frac{dF(u)}{dx} = \frac{F(\widehat{u}_{i+1/2}) - F(\widehat{u}_{i-1/2})}{\Delta x}, \tag{3.3.2}$$

where $\widehat{u}_{i\pm1/2}$ is the result of an interpolation process. Furthermore $\widehat{u}_{i+1/2}$ is a linear combination of an approximation from the left ($\widehat{u}^l$) and one from the right ($\widehat{u}^r$). Since both approximations are exactly the same up to reflection at the point $x_{i+1/2}$ we will only describe the calculation of $\widehat{u}^l$.
Let $r = 3$ and a candidate stencil S_k be defined as

$$S_k = (x_{i+k-r+1}, x_{i+k-r+2}, \dots, x_{i+k}) \quad \forall k = 0, \dots, r-1. \tag{3.3.3}$$

We can use each stencil S_k to get an approximate value for $\widehat{u}^l$ which we denote as $\widehat{u}^l_k$. The idea of the ENO schemes is to choose the stencil S_k that has the smoothest function values of the initial function u. This stencil will then solely contribute to the approximation $\widehat{u}^l$. It turns out that this provides a method that is r-th order accurate.
On the other hand if we choose

$$\widehat{u}^l = \sum_{k=0}^{r-1} \omega_k \widehat{u}^l_k, \tag{3.3.4}$$

with

$$\sum_{k=0}^{r-1} \omega_k = 1, \tag{3.3.5}$$

the resulting method has an order of $2r$-1 under certain choices of $\widehat{u}_k^l$ and ω_k.
To calculate $\widehat{u}_k^l$ we use interpolating polynomials and it can be shown that they are uniquely determined by the formula

$$\widehat{u}_k^l = \sum_{l=0}^{r-1} a_{k,l}^r u_{i+k-r+l+1}, \tag{3.3.6}$$

where the coefficients $a_{k,l}^r$ are defined in Table 3.1.

The second set of variables that we have to define are the so called weights ω_k. We want to stick close to the ENO idea of choosing the smoothest stencil. So we would like to have that the weights are some sort of smoothness measure for u and they should sum up to one.

Table 3.1 Coefficients $a_{k,l}^r$

k	l=0	l=1	l=2
0	1/3	-7/6	11/6
1	-1/6	5/6	1/3
2	1/3	5/6	-1/6

If we look at the problem again from an interpolation point of view there would be so called optimal weights $\omega_k = C_k^r$ (cf. Table 3.2) to achieve the best interpolation. This yields

$$\widehat{u}^l = \sum_{k=0}^{r-1} C_k^r u_{i+k-r+l+1} + \sum_{k=0}^{r-1} (\omega_k - C_k^r) u_{i+k-r+l+1}. \tag{3.3.7}$$

Now the last term is very small under the assumption that $\omega_k = C_k^r + \mathcal{O}(h^{r-1})$. As we additionally require the ω_k to sum up to unity we define ω_k by

Table 3.2 Coefficients C_k^r $(r = 3)$

k	0	1	2
	1/10	6/10	3/10

$$\omega_k = \frac{\alpha_k}{\alpha_0 + \ldots + \alpha_{r-1}}, \tag{3.3.8}$$

where

$$\alpha_k = \frac{C_k^r}{(\epsilon + IS_k)^p}. \tag{3.3.9}$$

Here ϵ is a small non-negative regularization parameter which prevents the denominator from becoming zero. In all simulations we will use $\epsilon = 10^{-20}$ as suggested in [10]. The value of p is nowhere strictly defined, based on numerical experience we conclude that $p = 2$ provides the best performance (see also [9]).

The values of IS_k indicate how smooth a certain stencil is and they are defined by

$$\begin{aligned} IS_0 &= \frac{13}{12}(u_{i-2} - 2u_{i-1} + u_i)^2 + \frac{1}{4}(u_{i-2} - 4u_{i-1} + 3u_i)^{,} && (3.3.10) \\ IS_1 &= \frac{13}{12}(u_{i-1} - 2u_i + u_{i+1})^2 + \frac{1}{4}(u_{i-1} - u_{i+1})^2, && (3.3.11) \\ IS_2 &= \frac{13}{12}(u_i - 2u_{i+1} + u_{i+2})^2 + \frac{1}{4}(u_i - 4u_{i+1} + 3u_{i+2})^2. && (3.3.12) \end{aligned}$$

Taylor expansion of the equations above yields

$$\begin{aligned} IS_0 &= \frac{13}{12}(u''h^2)^2 + \frac{1}{4}(2u'h - \frac{2}{3}u'''h^3)^2 + O(h^6), && (3.3.13) \\ IS_1 &= \frac{13}{12}(u''h^2)^2 + \frac{1}{4}(2u'h + \frac{2}{3}u'''h^3)^2 + O(h^6), && (3.3.14) \\ IS_2 &= \frac{13}{12}(u''h^2)^2 + \frac{1}{4}(2u'h - \frac{2}{3}u'''h^3)^2 + O(h^6). && (3.3.15) \end{aligned}$$

By analyzing these expansions it can be seen that the method is fifth-order accurate. This concludes the calculation of $\widehat{u}^l$ and $\widehat{u}^r$. All that is left is the linear combination of the two to obtain an approximation for $F(u)$. This is usually known as flux splitting and there are various methods that describe how to do it. During the course of the investigations a new way of combining the two turned out to provide superior results to the others usually used. We set

$$F(u_{i+1/2}) = \omega_l \; F(\widehat{u}^l_{i+1/2}) + \omega_r \; F(\widehat{u}^r_{i+1/2}), \tag{3.3.16}$$

where

$$\omega_l = \frac{\widehat{\omega}_l}{\widehat{\omega}_l + \widehat{\omega}_r} \tag{3.3.17}$$

and

$$\widehat{\omega}_l = \frac{1}{0.01 + |F(\widehat{u}^l_{i+1/2})|}. \tag{3.3.18}$$

As can be seen this is direclty motivated from the WENO interpolation weights since it ensures that the smaller fluxes are prefered. When considering a disconitinuity in the initial condition it can be seen that only the lower flux should contribute to the solution which is why this weighting was chosen. The divergence of the flux is then finally obtained by using equation (3.3.2).

In general there are two different ways of approximating a scalar conservation law using WENO. The method described above is a finite volume scheme. It uses u at the integer grid to get an approximation of u at the half-integer grid via the WENO scheme. This is then used to calculate the flux F in equation (3.3.1).

Alternatively it would be possible to calculate the flux F at the integer grid via a finite difference scheme. A WENO scheme is then applied to F to obtain the flux at the half-integer grid. This is commonly referred to as the finite difference scheme for WENO methods [11].

In higher dimensions finite volume schemes are usually not the best choice since they induce higher computational cost. However this is not true in the case of the diffusion equation. Since we need to approximate Δu we can split the calculation into an uncoupled system of n one dimensional equations and recombine them via the tensor product. This also implies that the cost grows only linearly with the dimension and not exponentially.

Finally I want to point the interested reader to the above mentioned paper by Shu [11] which gives an excellent overview over state of the art WENO schemes.

3.4 Truncation Error of WENO coupled with Heun's method

Since the calculation of the truncation error is quite cumbersome the process shall be divided into two steps. First of all a demonstration of the detailed calculation of the truncation error for Heun's method will be presented. This established principle is then used to calculate the complete truncation error for WENO schemes coupled with Heun's method. However, only the results of the latter will be presented.

We start with calculating the truncation error for Heun's method [12]. To do this consider the ordinary differential equation (ODE)

$$\frac{du}{dt}(t) = f(t, u(t)), \tag{3.4.1}$$

with initial condition $u(t_0) = y_0$.

The Taylor series gives us an expression for $u(t_0 + \Delta t)$, namely

$$u(t_0 + \Delta t) = u(t_0) + u^{(1)}(t_0)\Delta t + \frac{1}{2}u^{(2)}(t_0)\Delta t^2 + \frac{1}{6}u^{(3)}(\tau)\Delta t^3, \tag{3.4.2}$$

for $\tau \in [t_0, t_0 + \Delta t]$. Heun's method is also used for approximating $u(t_0 + \Delta t)$ and written in one formula (see eq. (3.2.6)) it reads

$$u(t_0 + \Delta t) = u(t_0) + \frac{1}{2}\left[f(t_0, u(t_0)) + f(t_0 + \Delta t, u(t_0) + \Delta t\, f(t_0, u(t_0)))\right]\Delta t + \epsilon_{rk}, \tag{3.4.3}$$

where ϵ_{rk} is the truncation error.
Subtracting the two equations above yields

$$\begin{aligned}\epsilon_{rk} &= u^{(1)}(t_0)\Delta t + \frac{1}{2}u^{(2)}(t_0)\Delta t^2 + \frac{1}{6}u^{(3)}(\tau)\Delta t^3 \qquad (3.4.4)\\ &\quad -\frac{1}{2}\left[f(t_0, u(t_0)) + f(t_0+\Delta t, u(t_0) + \Delta t\, f(t_0, u(t_0)))\right]\Delta t\\ &= u^{(1)}(t_0)\Delta t + \frac{1}{2}u^{(2)}(t_0)\Delta t^2 + \frac{1}{6}u^{(3)}(\tau)\Delta t^3 \qquad (3.4.5)\\ &\quad -f(t_0, u(t_0))\Delta t - \frac{1}{2}\left[-f(t_0, u(t_0)) + f(t_0+\Delta t, u(t_0) + \Delta t\, f(t_0, u(t_0)))\right]\Delta t.\end{aligned}$$

The initial ODE gives us that $u^{(1)}(t_0) = f(t_0, u(t_0))$ and thus

$$\begin{aligned}\epsilon_{rk} &= \frac{1}{2}u^{(2)}(t_0)\Delta t^2 + \frac{1}{6}u^{(3)}(\tau)\Delta t^3 \qquad (3.4.6)\\ &\quad -\frac{1}{2}\left[f(t_0+\Delta t, u(t_0) + \Delta t\, f(t_0, u(t_0))) - u^{(1)}(t_0)\right]\Delta t.\end{aligned}$$

Let us now consider the second argument in the remaining f term. Again due to the initial ODE it can easily be seen that it is a first order approximation to $u(t_0 + \Delta t)$, i.e.

$$u(t_0+\Delta t) \approx u(t_0) + \Delta t\, f(t_0, u(t_0)). \qquad (3.4.7)$$

It would be possible to give a more precise formula containing the second derivative of u but since it is an argument inside the non-linear function f this will not help us at all. Thus the truncation error now becomes

$$\begin{aligned}\epsilon_{rk} &\approx \frac{1}{2}u^{(2)}(t_0)\Delta t^2 - \frac{1}{2}\left[f(t_0+\Delta t, u(t_0+\Delta t)) - u^{(1)}(t_0)\right]\Delta t + \frac{1}{6}u^{(3)}(\tau)\Delta t^3 \qquad (3.4.8)\\ &\overset{(3.4.1)}{=} \frac{1}{2}u^{(2)}(t_0)\Delta t^2 - \frac{1}{2}\left[u^{(1)}(t_0+\Delta t) - u^{(1)}(t_0)\right]\Delta t + \frac{1}{6}u^{(3)}(\tau)\Delta t^3 \qquad (3.4.9)\\ &= \frac{1}{2}u^{(2)}(t_0)\Delta t^2 - \frac{1}{2}\frac{u^{(1)}(t_0+\Delta t) - u^{(1)}(t_0)}{\Delta t}\Delta t^2 + \frac{1}{6}u^{(3)}(\tau)\Delta t^3. \qquad (3.4.10)\end{aligned}$$

Again using Taylor expansion we have

$$\begin{aligned}u^{(1)}(t_0+\Delta t) &= u^{(1)}(t_0) + u^{(2)}(t_0)\Delta t + \frac{1}{2}u^{(3)}(\tau)\Delta t^2 \qquad (3.4.11)\\ \Leftrightarrow \frac{u^{(1)}(t_0+\Delta t) - u^{(1)}(t_0)}{\Delta t} &= u^{(2)}(t_0) + \frac{1}{2}u^{(3)}(\tau)\Delta t. \qquad (3.4.12)\end{aligned}$$

Note that the τ in equations (3.4.2) and (3.4.11) are identical. Thus inserting equation (3.4.11) into (3.4.10) gives

$$\begin{aligned}
\epsilon_{rk} &= \frac{1}{2}u^{(2)}(t_0)\Delta t^2 - \frac{1}{2}\left[u^{(2)}(t_0) + \frac{1}{2}u^{(3)}(\tau)\Delta t\right]\Delta t^2 + \frac{1}{6}u^{(3)}(\tau)\Delta t^3 &\quad (3.4.13)\\
&= -\frac{1}{12}u^{(3)}(\tau)\Delta t^3. &\quad (3.4.14)
\end{aligned}$$

If we let $\|u^{(3)}\|_\infty = \sup\limits_{t\in[t_0,t_0+\Delta t]} |u^{(3)}(t)|$ we can estimate the absolute value of the truncation error as

$$|\epsilon_{rk}| \le \frac{1}{12}\|u^{(3)}\|_\infty \Delta t^3. \quad (3.4.15)$$

This completes the first step and we can now move on to describing how this can be combined with the truncation error of WENO schemes. Since we will not give a detailed derivation of the truncation error the reader is referred to [13] for details. Note however that due to the different flux splitting approach the results do not match.

To include the truncation error of the WENO schemes denoted by ϵ_w we replace f by $\nabla(\kappa\nabla u) + \epsilon_w$. Let $L = \nabla(\kappa\nabla u)$ and $L^{n+1} = L(L^n)$. The truncation error after the first Runge–Kutta step is then given by

$$\epsilon_{rk,1} = \epsilon_{w,1} - \frac{\Delta t}{2}L^2(u). \quad (3.4.16)$$

The truncation error after the second step is

$$\epsilon_{rk,2} = \Delta t\left(\epsilon_{w,1} + \frac{\epsilon_{w,1}+\epsilon_{w,2}}{2}\right) + \Delta t^3\frac{1}{12}L^3(u), \quad (3.4.17)$$

where $\epsilon_{w,1}$ (resp. $\epsilon_{w,2}$) is the truncation error of the WENO scheme in the first (resp. second) Runge–Kutta step. Put into formula we have

$$\begin{aligned}
\epsilon_w(i) = \frac{1}{12}\Delta x\frac{\partial^3 u}{\partial x^3}[\quad & \omega_l(&& \omega_{l,1,i-3/2} - \omega_{l,1,i-1/2} - \omega_{l,1,i+1/2} + \omega_{l,1,i+3/2} \quad (3.4.18)\\
& - && \omega_{l,0,i-3/2} + \omega_{l,0,i-1/2} + \omega_{l,0,i+1/2} - \omega_{l,0,i+3/2})\\
- \quad & \omega_r(&& \omega_{r,1,i-3/2} - \omega_{r,1,i-1/2} - \omega_{r,1,i+1/2} + \omega_{r,1,i+3/2}\\
& - && \omega_{r,2,i-3/2} + \omega_{r,2,i-1/2} + \omega_{r,2,i+1/2} - \omega_{r,2,i+3/2})],
\end{aligned}$$

where $\omega_{l/r,k,j}$ is the weight ω_k in the WENO approximation of a point x_j from the left/right side.

3.5 Cholesky Decomposition

To solve the spline approximation problem in Section 3.6 we will have to be able to solve a linear system of equations $Ax = b$. Since the Cholesky decomposition will be the appropriate tool to do so, we will give a short overview here.
Let A be a real, symmetric and positive definite matrix, i.e.

$$A = A^T \text{ and } x^T A x > 0 \; \forall x \neq 0. \tag{3.5.1}$$

The easiest method to solve $Ax = b$ is the LR decomposition coupled with a forward and a backward substitution. Due to the specific structure of A a much faster algorithm can be derived based on the LR decomposition [14]. The main results shall be stated briefly, for proofs and in-depth discussion cf. [14, 15].

Theorem 3.5.1. *For every symmetric and positive definite matrix A there exists a unique decomposition such that*

$$A = LDL^T, \tag{3.5.2}$$

where L is a lower triangular matrix and D is a positive diagonal matrix.

Corollary 3.5.2. *Due to D being positive there exists $D^{\frac{1}{2}} = \left(\sqrt{d_{i,j}}\right)_{i,j}$ such that A can be written as*

$$A = LD^{\frac{1}{2}} D^{\frac{1}{2}} L^T = LD^{\frac{1}{2}} \left(LD^{\frac{1}{2}}\right)^T = \widetilde{L}\widetilde{L}^T, \tag{3.5.3}$$

which is the so called Cholesky decomposition.
The entries of the matrix $\widetilde{L} = (l_{i,j})_{i,j}$ can be calculated explicitly by the formula below:

$$l_{i,i} = \sqrt{a_{i,i} - \sum_{j=1}^{i-1} l_{i,j}^2}, \tag{3.5.4}$$

$$l_{j,i} = \frac{1}{l_{i,i}} \left(a_{j,i} - \sum_{k=1}^{i-1} l_{j,k} l_{i,k} \right) \quad \forall j = i+1, \ldots, n. \tag{3.5.5}$$

The system $Ax = b$ can thus be rewritten as $\widetilde{L}\widetilde{L}^T x = b$. Since $\widetilde{L}$ is a lower triangular matrix, the solution can be calculated by first solving $\widetilde{L}z = b$ and then $\widetilde{L}^T x = z$ by simple back-substitution.

3.6 Spline Approximation

When faced with the task of approximating or interpolating a smooth function, splines will most certainly be an option to consider. Splines have certain properties that make them ideal for the task. Amongst others they consist of piecewise polynomials with specific global smoothness conditions.

Let n and d be integers and $t = \{t_1, \ldots, t_{n+d+1}\}$ a set of real values called *knots* that satisfy $t_{i+d+1} > t_i$ for $i = 1, \ldots, n$. A *basis B-spline* $B_{i,d}$ $(1 \leq i \leq n)$ of degree d is then defined recursively by the *Cox-de Boor formula*

$$B_{i,1}(t) := \begin{cases} 1 & \text{if} \quad t_i \leq t < t_{i+1}, \\ 0 & \quad \text{otherwise}, \end{cases} \tag{3.6.1}$$

$$B_{i,d}(t) := \frac{t - t_i}{t_{i+d} - t_i} B_{i,d-1}(t) + \frac{t_{i+d+1} - t}{t_{i+d+1} - t_{i+1}} B_{i+1,d-1}(t). \tag{3.6.2}$$

An n dimensional *spline space* $\mathbb{S}_{d,t}$ of degree d and with knots t is defined as

$$\mathbb{S}_{d,t} = \{g : \mathbb{R} \to \mathbb{R} \mid g(x) = \sum_{i=1}^{n} c_i \, B_{i,d}(x), c \in \mathbb{R}^n\}. \tag{3.6.3}$$

To approximate a given set of data a method of measuring the error is needed. To keep things simple and efficient we will use a weighted least squares norm. That means that the approximation problem can be stated as follows.

Let $(x_i, y_i)_{i=1}^m$ (with $x_1 < \ldots < x_m$) be a given set of data, $(\omega_i)_{i=1}^m$ a set of real numbers and $\mathbb{S}_{d,t}$ an n dimensional spline space. Then the best approximation $g \in \mathbb{S}_{d,t}$ in the weighted least squares sense is defined via the *argmin*[1] as

$$g = \underset{h \in \mathbb{S}_{d,t}}{\operatorname{argmin}} \sum_{i=1}^{m} \omega_i (y_i - h(x_i))^2. \tag{3.6.4}$$

At this point we should make an important observation that will be exploited later on. If we knew that the value y_i for some i is exact we would not want to approximate the function at this point but rather interpolate it. This can be achieved by letting ω_i be large compared to the other weights. Thus if we have higher confidence for certain data points we will assign larger weights to them. Furthermore it is necessary to assume $m > n$ to avoid interpolation.

The proofs of the following statements can be found in [16].

Lemma 3.6.1. *Let $A = (a)_{i,j} \in \mathbb{R}^{m,n}$ and $b \in \mathbb{R}^m$ with components*

$$a_{i,j} = \sqrt{\omega_i} \, B_{j,d}(x_i) \quad \textit{and} \quad b_i = \sqrt{\omega_i} \, y_i. \tag{3.6.5}$$

---

[1] Stands for *argument of the minimum*, i.e. $\underset{x \in S}{\operatorname{argmin}} f(x) := \{x | \forall y \in S : f(y) \geq f(x)\}$

Then the approximation problem (3.6.4) *is equivalent to the linear least squares problem*

$$c = \underset{d \in \mathbb{R}^n}{argmin} \|A\, d - b\|_2. \tag{3.6.6}$$

The next Lemma will show why the least squares approach is so efficient.

Lemma 3.6.2. *Let A, b and c be given as in Lemma 3.6.1. Then c is equal to the solution c^* of the linear equation*

$$A^T A c^* = A^T b. \tag{3.6.7}$$

Furthermore $N = A^T A$ is symmetric and positive semi-definite. N is positive definite and thus non-singular and invertible if and only if $dim(ker(A)) = 0$. Note that in this case the solution is unique.

Thus if it is possible to fulfil the last requirement in the Lemma we could easily solve the problem. Indeed the following Theorem holds true.

Theorem 3.6.3. *There is a unique spline $g \in \mathbb{S}_{d,t}$ if and only if there is a sub-sequence $(x_{i_l})_{l=1}^n$ such that $B_{l,d}(x_{i_l}) \neq 0$ for $l = 1, \ldots, n$.*

Going back to the last section we see that it is possible to use the Cholesky decomposition to solve the problem if we only choose the knots t appropriately. There is no strict rule on how to define them but the quality of the approximation depends critically upon the knot sequence as shown in [17]. The paper also shows a way of obtaining the best possible knot distribution, however with considerable computational cost. Since we will apply the spline approximation $n \cdot N$ times per time-step this method cannot be justified. Furthermore the paper deals with a large number of function values which is something we need to avoid in our case since that would mean using a broad stencil. Thus a fixed knot vector has to be used.

The choice presented in the following is in no way optimal but has proven to give decent results as will be shown in Chapter 3. First of all we fix the size of the stencil around a point x_i by defining $m = 9$, i.e. $x = (x_j)_{j=i-4}^{i+4}$. The easiest way to secure the assumption of Theorem 3.6.3 is that the $B_{j,d}$ form a partition of unity. If we also limit our knots to the interior of the data points we thus need knots of multiplicity 4 at x_{i-4} and at x_{i+4}. This already requires at least 8 knots. Finally we add a knot at the mid-point of the domain, i.e. x_0. We thus end up with 9 knots, i.e.

$$t = (x_{i-4}, x_{i-4}, x_{i-4}, x_{i-4}, x_i, x_{i+4}, x_{i+4}, x_{i+4}, x_{i+4}), \tag{3.6.8}$$

which results in a 5 dimensional spline space $\mathbb{S}_{3,t}$. The resulting basis splines can be seen in Fig. 3.1.

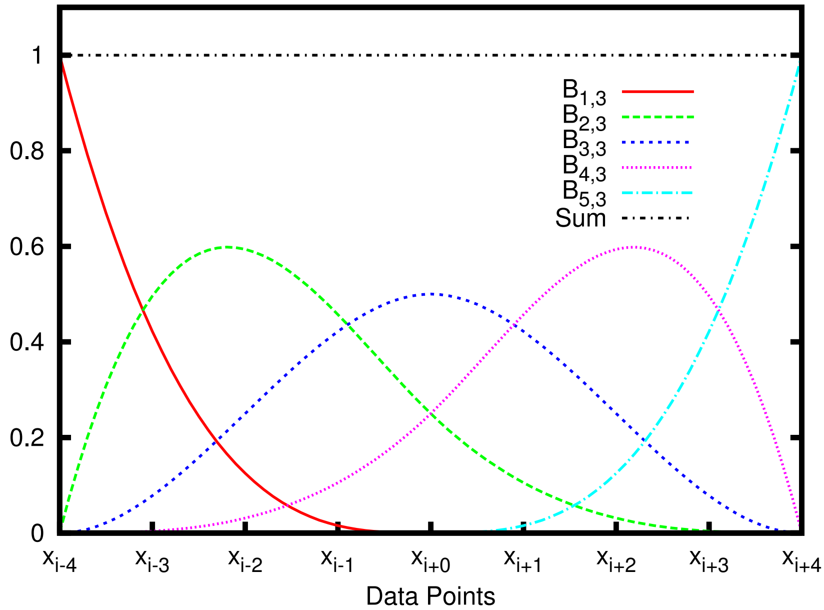

Figure 3.1: Basis B-Splines of third order

3.7 Smoothed Essentially Non-Oscillatory Schemes

Now we have gathered all the prerequisites necessary to implement a new explicit scheme for solving the diffusion equation. The three main ingredients are Heun's method, WENO schemes and spline approximation. In the following we will first describe the method for the one dimensional case (Section 3.7.1). This will then be extended to arbitrary dimensions in Section 3.7.2.

3.7.1 In One Dimension

The main idea of the *Smoothed Essentially Non-Oscillatory* (SENO) method is to look at the diffusion equation not as a parabolic problem but rather akin to a hyperbolic conservation law. This means that we think of the diffusion equation (2.2.1) as

$$\frac{\partial u}{\partial t} + \nabla F(u) = 0 \tag{3.7.1}$$

with

$$F(u) = \kappa(\nabla u). \tag{3.7.2}$$

As discussed in Section 3.3 we can use traditional WENO methods to approximate the divergence of the flux. However as can be seen above the flux F depends on u in a non-linear fashion. In fact we need to replace equation (3.7.2) by

$$F(u(x_{i+1/2})) = \kappa(x_i)\frac{u(x_{i+3/2}) - u(x_{i-1/2})}{2\,\Delta x}. \tag{3.7.3}$$

Remember that the WENO scheme gives us the values of u at the half integer grid and that we also want to calculate the flux at the half integer grid (cf. equation (3.3.16)). Furthermore we need a suitable time integration method. Due to the advantageous properties of Heun's method we will employ this RK scheme.

An approach similar to this one was already pursued by Obertscheider in [18]. However he used a different WENO scheme and the results showed that he could only use Courant numbers up to 0.5. Moreover there are certain difficulties arising in higher dimensions and it will be shown that WENO schemes alone are completely unstable in two dimensions.

We will now describe what type of calculation will be done in each RK step.

The WENO scheme mentioned above will be utilized with another small change in the form of a preconditioning step. The idea came from the representation of the solution as weighted average of the initial condition as discussed in Section 2.3.1. We use a stencil of length $2r+1$ and consider the solution at time n to be the initial condition. Then we obtain a predicted solution as

$$u^*_{n+1,i} = \sum_{j=-r}^{r} \frac{1}{\widehat{e}}\,\mathrm{e}^{-\frac{(j\,\Delta x)^2}{\Delta t}} u_{n,i+j}, \tag{3.7.4}$$

where

$$\widehat{e} = \sum_{k=-r}^{r} \mathrm{e}^{-\frac{(k\,\Delta x)^2}{\Delta t}} \tag{3.7.5}$$

Obviously equation (3.7.4) is the discrete analogy to the exact solution described in equation (2.3.14). Note however that the coefficient κ is not included. The predicted solution is then used as input for the WENO scheme mentioned above. In the algorithm we use $r = 2$ since it proves to be the best compromise between stencil width and gain in timestep.

Of course also with the adapted WENO approach described above, oscillations will show up. Since we know that the solution has to be smooth (cf. Section 2.3.2) the idea is to apply some sort of smoothing technique. Moreover we already know the truncation error ϵ_{rk} of the WENO & RK combination (cf. Section 3.4). Thus we require that the smoothed curve lies within the interval $[u_w(x) - \epsilon_{rk}(x), u_w(x) + \epsilon_{rk}(x)]$, where u_w is the temporary solution. But this also means that we have a confidence measure for the solution. This leads us to the assumption that using the spline approximation technique described in Section 3.6 would be appropriate. Indeed it turns out that applying a specific spline smoothing algorithm is giving us exactly what we need.

Before this algorithm is described in detail another feature of the SENO algorithm will be described. After obtaining the flux from the WENO method we calculate a temporary solution by applying the appropriate RK step. Now we use the maximum principle that was shown in Section 2.3.3. We modify it slightly so that it states that the temporary solution may not exceed the maximum and minimum of the solution of the previous step. Additionally this also has to hold true for the interval describing the truncation error. Although this does not improve the maximum Courant number it provides an important speed-up.

Now we want to consider the spline approximation in greater detail. Let us consider the algorithm at a point x_i assuming a large enough truncation error. We initialize the weights of the weighted least squares approximation to one. We then solve the spline approximation problem and determine whether the smoothed curve is completely inside the truncation error interval. If this holds, the algorithm moves on to the next node x_{i+1}, otherwise the weights are adapted according to the following rules. Let $m^- = i-(m-1)/2$ and $m^+ = i + (m-1)/2$. Note that m is an odd integer since we require a symmetric stencil.

- If all weights $(\omega)_{m^-}^{m^+}$ are smaller than δ_1 then the new weights are calculated according to the formula

  $$\omega_j = \omega_j \left(1 + \frac{|u_j^* - u_{n,j}|}{\epsilon}\right) \quad \forall j = m^-, \dots, m^+, \tag{3.7.6}$$

  where u_j^* is the solution of the spline approximation problem and

  $$\epsilon = \sum_{k=m^-}^{m^+} |u_k^* - u_{n,k}|. \tag{3.7.7}$$

- If all weights are smaller than δ_2, only the weight ω_i is changed by multiplication with 4.

- Otherwise the algorithm forces the value u_i^* to be on the boundary of the truncation error interval depending on which boundary value is closer to the original approximation.

The values for δ_1 and δ_2 can be chosen freely. However one should try to balance the values with respect to accuracy and computational cost. In the experiments we will use

$$\delta_1 = 10^7 \quad \text{and} \quad \delta_2 = 10^{10}. \tag{3.7.8}$$

One slight drawback of this method is that the function decays slightly faster when spline smoothing is applied. To compensate this effect the maximum and minimum of u before smoothing is saved (u_o^+, u_o^-). After the smoothing is applied maximum and minimum of u are determined again (u_n^+, u_n^-) and the function u is changed according to

$$u = (u - u_n^-)\frac{|u_o^+ - u_o^-|}{|u_n^+ - u_n^-|} + u_o^-. \tag{3.7.9}$$

To compensate further numerical errors we enforce that the integral over u is constant. To achieve this the value of the integral at the beginning is saved (U_0) and the current integral U_n is calculated. The function u is then multiplied by the ratio U_0/U_n. This method does have an important drawback that needs to be mentioned here. We only showed that the property of constant integral holds for periodic initial conditions. However if we assume a problem with non-periodic boundary conditions, things get a bit more complicated. In case of Neumann boundary conditions, i.e. $\frac{\partial u}{\partial n} = 0$, equation (2.3.29) still holds true due to the divergence theorem. However, care must be exercised when using Dirichlet boundary conditions. In theory it should still be possible to adapt the condition by considering the flux at the Dirichlet boundary.

These are all methods employed in the SENO scheme for high Courant numbers.

As it turns out there is a possibility of small oscillations when using low Courant numbers and discontinuous initial conditions. To avoid this the spline smoothing algorithm is slightly modified. Instead of only using the data values provided by the WENO and RK combination $(\widetilde{u})$ the smoothed values immediately replace the old ones. This means that after calculating u_i from the values $\widetilde{u}$, $\widetilde{u}_i$ is substituted by u_i. Of course there is now a necessity for choosing a starting point. Obviously it would be best if the starting point does not require any smoothing, i.e. the truncation error is minimal. The direction of smoothing is then determined by the value of the truncation error of the points neighbouring the initial starting point. Additionally we assume that the smoothing gives us more accurate function values and thus we resize the truncation error by dividing it by 10.

It is important to note that this feature, which shall be called *dependent spline smoothing*, somewhat destroys the local character of the SENO scheme presented above. In case that the code would be rewritten for parallel computing one needs to take this into account.

This concludes the description of the SENO method in one dimension.

Table 3.3 Stencil widths

Method	**Stencil width s**	**r** $\left(= \frac{s-1}{2}\right)$
Preconditioning	5	2
WENO	9	4
Spline Smoothing	9	4
1 RK step	21	10
SENO	41	20

As mentioned above some properties depend on the Courant number. In the following paragraph we will show the Courant numbers C which are important for the adaptive algorithm. The values for these bounds will be justified in Chapter 4.

- $C \leq 0.3$: RK time integration with sixth order central difference formula to calculate the second derivative, i.e.

$$\begin{aligned} \frac{\partial u}{\partial x}(x_i) &= \frac{1}{360\ \Delta x^2}(2u_{i-3} - 27u_{i-2} + 270u_{i-1} - 490u_i \\ &+ 270u_{i+1} - 27u_{i+2} + 2u_{i+3}) + \mathcal{O}(\Delta x^6) \end{aligned} \qquad (3.7.10)$$

- $C \leq 1.1$: SENO scheme with dependent spline smoothing
- $C > 1.1$: SENO scheme without dependent spline smoothing

Before we move on to higher dimensions we want to analyze the width of the stencil that results from the above method for $C > 0.3$.

In Table 3.3 the different methods employed for the SENO scheme are listed together with their respective stencil width. Furthermore in the third column the value r is shown which describes how many points on one specific side influence a node. Adding up the stencils we see that the stencil for one RK step in the SENO method is 21 points wide. This is a considerable disadvantage of the method. Note that the full method has a stencil that contains 41 points.

The latter would actually render the method unusable for parallel programming. However it is common to have communication after each RK step and thus the result is not so bad after all.

3.7.2 In n Dimensions

Before moving on to the results of the new method we want to consider an extension to higher dimensions. Since most interesting problems are posed in three dimensions and today's computational power is sufficient for such simulations this is an important step. However it would be devastating if the algorithm needed to consider all directions at once. This would mean that the calculation time depends exponentially upon the dimension of the problem, i.e. the time is $\mathcal{O}(N^d)$, where N is the number of spatial discretization points and d the dimension.

Firstly note that the diffusion equation does not have any mixed derivatives. This implies subsequently that we can use the WENO scheme described previously in each direction separately. This already gives us a hint that we can really use the 1D algorithm and apply it for each dimension, resulting in a calculation time of the order $\mathcal{O}(d\,N)$.

Indeed the preconditioning step can also be employed in this fashion. The only thing remaining is the spline smoothing algorithm. Although it might be advisable to use higher dimensional splines to get an even stronger and more continuous effect, they were not really considered. First of all splines in dimensions larger than two are rarely used as they are quite complicated and enlarge the system of linear equations significantly. Secondly due to the weight adapting algorithm that is used in the SENO method the number of weights would increase greatly, thus introducing even more unknowns.

Summarising the above it can be said that the algorithm can simplify the problem into an array of one dimensional problems that can be solved consecutively by the algorithm described in the previous section. The solution is then obtained by applying the tensor product to the solutions. This allows an easy and efficient generalization of the problem to higher dimensions.

Chapter 4

Simulations

In this chapter we will demonstrate the capability of the SENO method. Due to the fact that there are additional difficulties when considering 2D problems, the main focus of this chapter will lie on these. We start by explaining the different aspects of the SENO schemes by simulating two model problems. The simulations will evolve step by step from the WENO method to address the advantages of each part of the SENO scheme. The first section will be concluded by a few more examples in 2D showing also a distinct disadvantage of the approach.

In the last section a few simulations in one dimension will be presented.

The SENO algorithm was implemented in Fortran 90 and the code will be released under the GNU general public license in summer 2011 on `www.amconception.de`.

4.1 Simulations in 2D

Before starting with the program outlined above a short presentation of the two model equations shall be given. They are characterized by the respective initial condition u_0 and the domain of computation. All simulations have been carried out assuming periodic boundaries as well as the time t ranging from 0 to 10. For now all simulations use 128 nodes in each direction.

The first equation is given by the initial condition

$$u_0(x, y) = \cos(x) \cdot \cos(y) \quad \forall\ (x, y) \in [-\pi, \pi]^2. \tag{4.1.1}$$

Since the initial value is already smooth this is an important reference when it comes to the accuracy of the method. We expect that this model will behave much more nicely than the second one when it comes to oscillations.

The second model is defined by

$$u_0(x,y) \quad := \quad \begin{cases} 1 & \text{if } x^2+y^2 \leq 1, \\ 0 & \text{otherwise,} \end{cases} \tag{4.1.2}$$

where $(x,y) \in [-2,2]^2$. This model will be called *cylinder* in the following. It features a discontinuous initial condition and will thus be more prone towards oscillations. However it is much more realistic when it comes to real-world applications.

The initial conditions of the two model equations can be seen in Fig. 4.1.

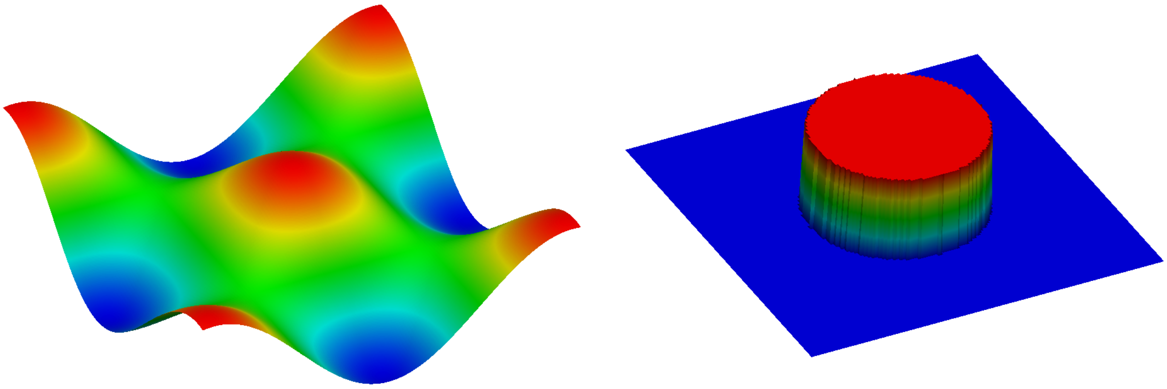

Figure 4.1: Initial conditions of the 2D models

Now the way is paved for the first set of simulations which are presented in Fig. 4.2. Before we analyze the figures the general structure of them shall be explained. Each figure contains four panels in two rows and two columns. The top column corresponds to the cosine model whereas the lower one shows the results from the cylinder. The left row describes the error in L^2 norm and the right row displays the computation time needed for the whole simulation. The upper right panel also contains the legend which applies for all panels.

All computations were run on two identical Mac Pro with 8 processors each. Each processor consists of an Intel Xeon CPU with 2.8 GHz and each machine has 2 GB of RAM. They are dedicated to high performance computing and use Gentoo as operating system with kernel 2.6.31. The code was compiled with the Intel Fortran Compiler version 10.0 using several parameters for optimization. Every processor ran its own simulation, usually one method with varying Courant numbers (Courant step size was strictly smaller than 0.1).

Consider now Fig. 4.2 with the first simulations. The following three methods are compared in this figure

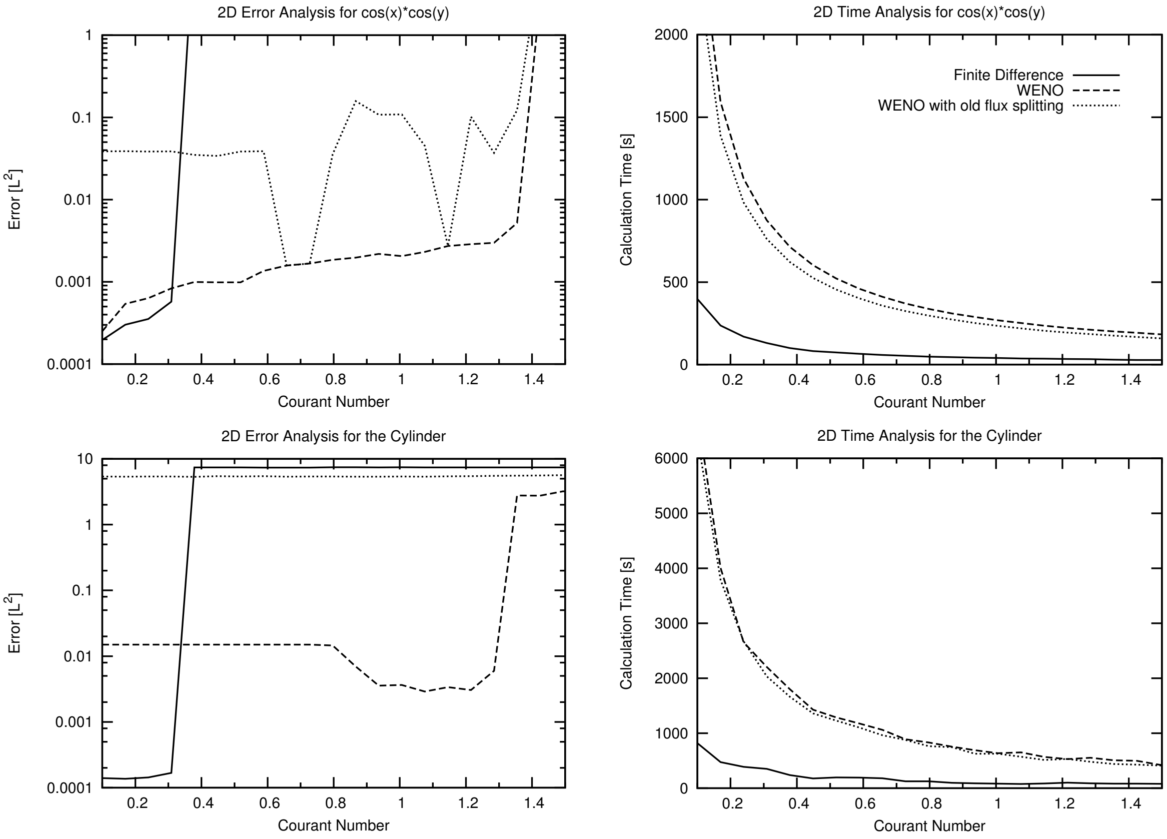

Figure 4.2: Weighted Essentially Non-Oscillatory Schemes

- *Finite Difference:* The 6th order central difference stencil described in equation (3.7.11) is used in combination with Heun's method.
- *WENO:* The WENO scheme as presented in Section 3.3 with the new flux splitting method.
- *WENO with old flux splitting:* As above but this time using the following flux splitting method:

$$F(\widehat{u}_{i+1/2}) \quad := \quad \begin{cases} F(\widehat{u}^{l}_{i+1/2}) & \text{if } \frac{\partial u}{\partial x}(x_{i+1/2}) \geq 0, \\ F(\widehat{u}^{r}_{i+1/2}) & \quad \text{otherwise.} \end{cases} \tag{4.1.3}$$

Looking at the Finite Difference simulation it can be seen that it is only stable up to $C = 0.3$. The only advantage being the considerably faster computation. Thus we will use this method in the low Courant region in the adaptive SENO scheme. It is also clear that the flux splitting presented in (3.3) is superior to the one mentioned above. Note that the traditional flux splitting methods like Godunov, Lax-Friedrichs, Engquist-Osher [11] cannot be used in this case. This is due to the fact that F is not only a function of u at one node but at two different nodes. In the 1D case the two different flux splitting

methods are more similar, but still the new flux splitting is approximately one order better, yet slightly slower.
One more feature can be seen in the lower left panel, namely the comparably high error for the WENO scheme in low Courant number regions. This is the aforementioned stability issue (cf. 3.7.1) that only arises in two dimensions. The comparable square wave simulation in 1D did not display this problem.

In Fig. 4.3 three different states of the simulation can be seen. The function values are the simulated values minus those of the exact solution. The left panel is the evolution after 0.1 s followed by the simulation at 1.6 s and the final state after 10 s. It can be seen that there is a considerable error immediately after the first output step.

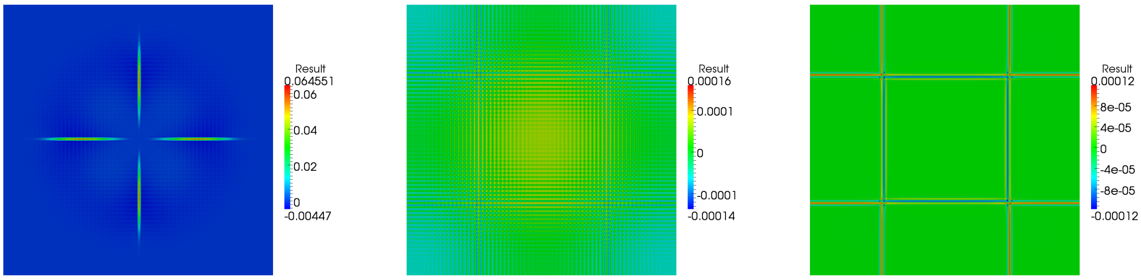

Figure 4.3: WENO stability issues.

The problem can be identified when considering Fig. 4.4. The plot on the top panel shows the cross section of the cylinder at the gray line. It resembles very much a delta spike, which as we will discuss at the end of this section is the source of the difficulties. This is mainly due to the fact that WENO is not able to resolve these structures.

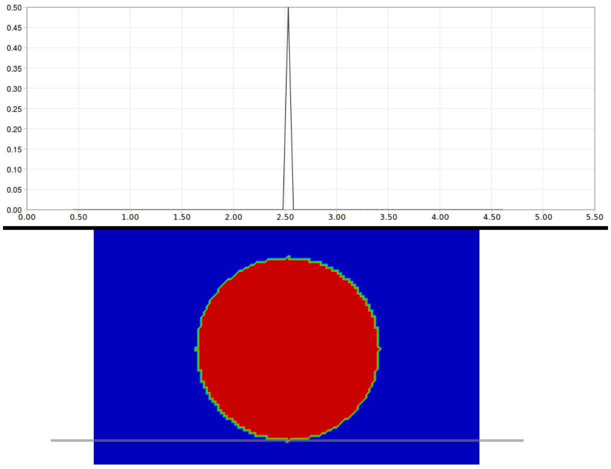

Figure 4.4: Cross section of the cylinder.

Due to that the cross visible in Fig. 4.3 on the left panel can be explained. Although

this error does actually disappear it still is saved in the simulation. As can be seen in the middle panel this manifests itself after about 1.6 s and eventually grows until it reaches a stable solution displayed in the right panel.
If one additionally requires that $u_0(x, y) = 1$ if $|y| \leq 1$ then obviously the two dimensional structure is destroyed and problems should only occur parallel to the y axis. Indeed this holds true which proves that these one dimensional delta spikes are the source of the problem.
Note that even due to the fact that it is possible to get rid of the oscillations the method still provides a fairly bad error when it comes to low Courant numbers. This point will be discussed further later in this chapter.

The first step towards the new SENO method was to apply some sort of smoothing to the resulting function after every RK step. In Section 3.6 it was already elaborated why using splines with a weighted least squares approximation is optimal for the task. In Fig. 4.5 the result of applying spline smoothing can be seen. The three graphs correspond to the following methods.

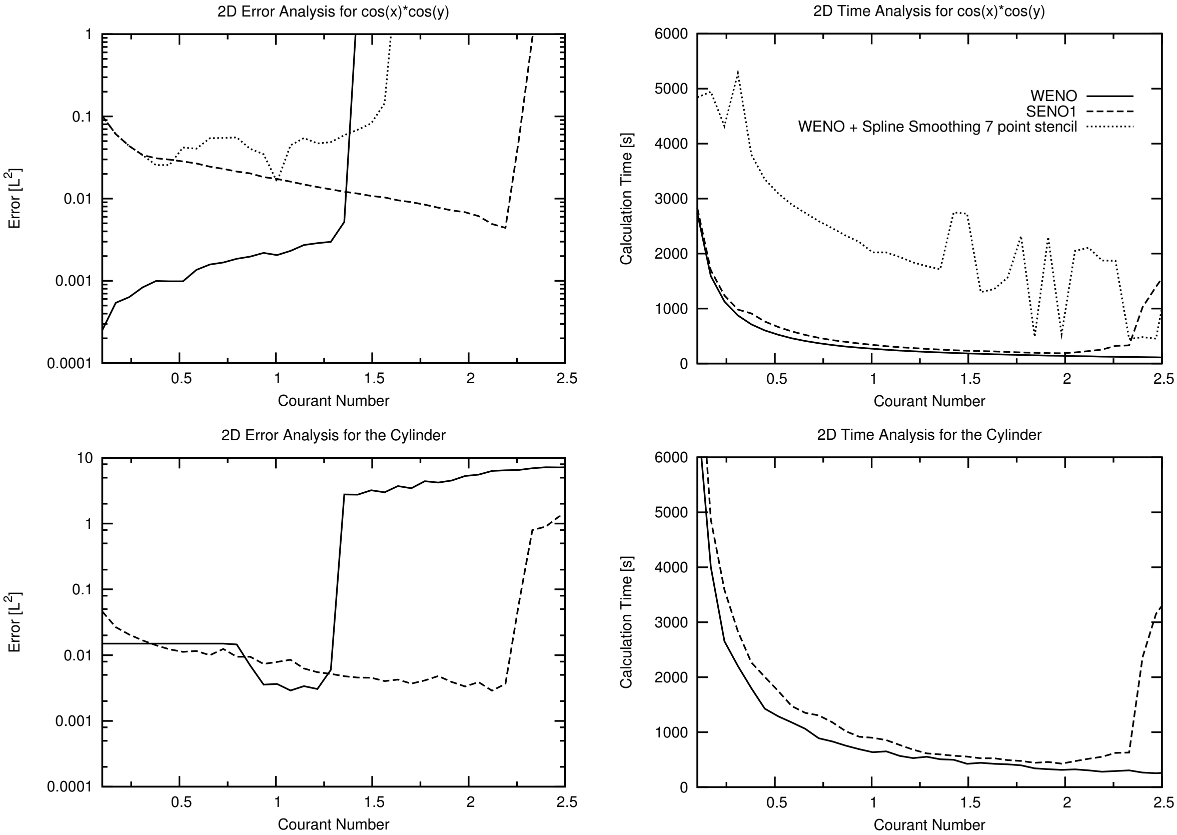

Figure 4.5: WENO with Spline Smoothing

- *WENO:* as above.

- *SENO1:* WENO including the spline smoothing using 9 function values for the approximation.
- *WENO + 7 point:* WENO with spline smoothing, however using only 7 function values for the approximation.

Although using fewer function values for the spline approximation would result in a smaller stencil, the results clearly show that it is necessary to use a stencil at least 9 points wide. The smaller stencil would be of advantage when it comes to parallel programming. Even though we will not pursue this here it is very important in applications.
As it can be seen in Fig. 4.5 there is no 7 point stencil simulation for the cylinder model. The reason for this is that the results were actually unreliable and the computation times too high. Additionally it is clear that the 9 point stencil provides a much larger gain in Courant numbers, the maximum now being around 2.2.
Despite that, the cosine model shows that the error increases by up to two orders of magnitude. The reason for this has already been mentioned, namely the faster decay of the smoothed function. Before moving on to compensate for this error a closer look at the spline smoothing shall be taken.

The simulations in Fig. 4.6 show slightly altered spline smoothing methods. They differ in the boundaries set for the following three features:

(i) B_l : If the truncation error at a point x_i is lower than B_l no spline smoothing is applied to save time. Instead the value u_i is redefined as a weighted average of itself and its neighbours by

$$u_i = \frac{u_{i-1} + 100\ u_i + u_{i+1}}{102} \tag{4.1.4}$$

which takes care of very small oscillations that could cause instabilities.

(ii) δ_1 : The first boundary for the weights (cf. equation (3.7.8)).

(iii) δ_2 : The second boundary for the weights.

The values used for the simulations in Fig. 4.6 are shown in Table 4.1.
The results indicate that the choice of weights for the SENO1 method is optimal in two ways. Compared to the weak boundaries the courant number is slightly larger with only minor additional effort. Secondly when weighting it against the strong boundaries, large spikes in computation time do not show up and for the discontinuous model the error is approximately the same as the maximum Courant number for both models.

It was already discussed that the error increases significantly as soon as spline smoothing is introduced. In Fig. 4.7 two strategies on lowering these errors are presented. The first

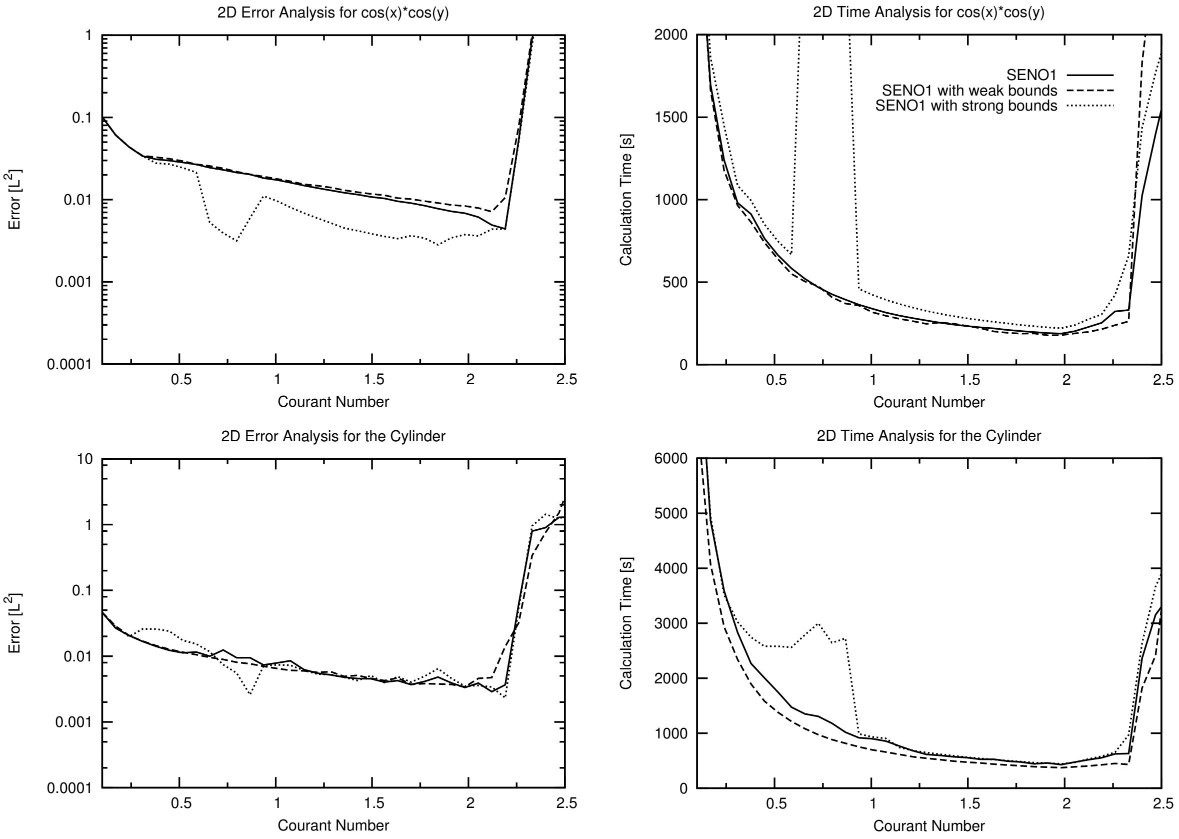

Figure 4.6: SENO1 with different bounds

Table 4.1 Values for the specific Boundaries

Method	log B_l	log δ_1	log δ_2
SENO1	-4	7	10
Weak Bounds	-3	3	5
Strong Bounds	-5	10	15

one is based on the assumption that the minimum and maximum are calculated correctly by the WENO + RK scheme. The exact formula can be seen in equation (3.7.9) and the corresponding graph is labeled *SENO1 + Smoothing Compensation*. The second refinement corresponds to an analytical property of the solution, namely the constant integral as proven in Section 2.3.5. The *SENO2* method combines both the smoothing and integral compensations.

In case of the cosine model the additional compensations lower the error to the level of the WENO method while still maintaining the high Courant numbers. Furthermore both models show that the computation time is not significantly larger due to the extra burden. However the situation is not so good in the case of the second model. There is nearly no error reduction in the low Courant number regions. But still the error is about five times

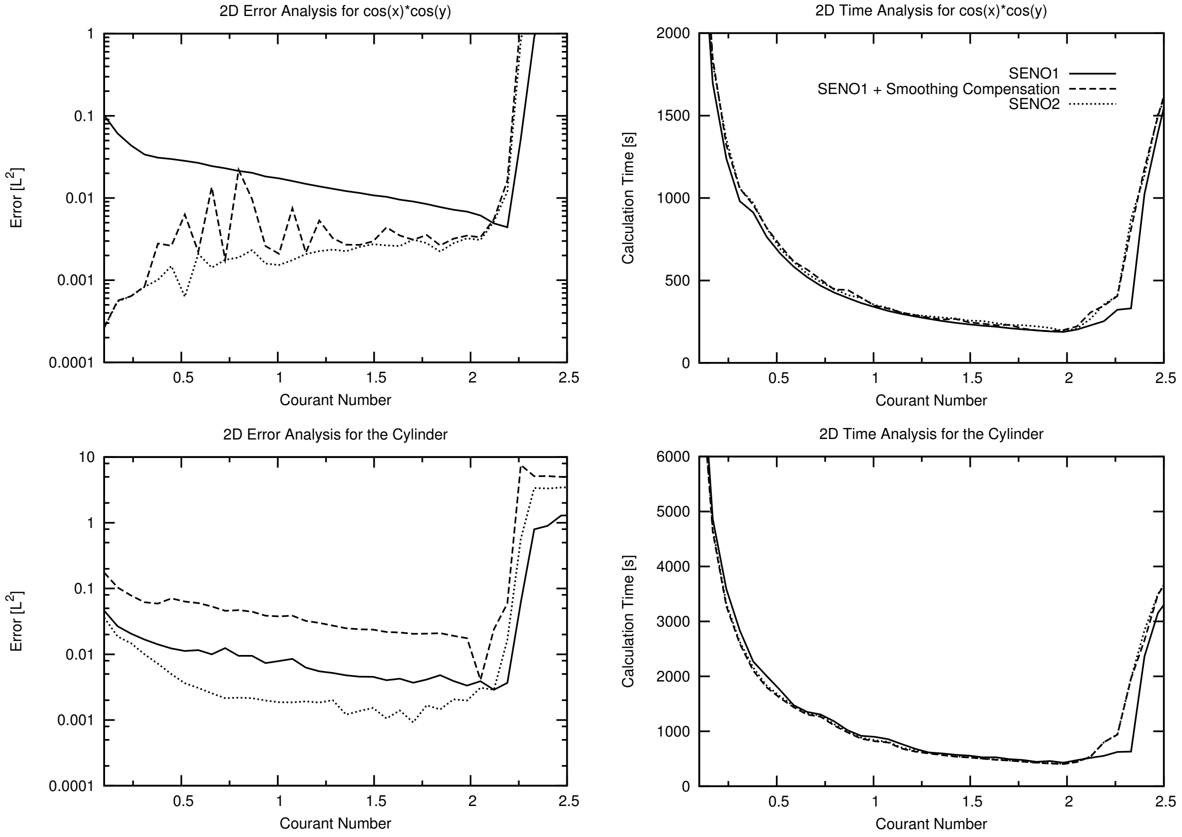

Figure 4.7: SENO1 with compensations

smaller if the Courant number is between 0.5 and 1.75. We also see that the assumption that the WENO method provides the correct extrema is wrong here. This is mostly due to the errors associated with the pure WENO scheme as shown in Fig. 4.3.

The next step in the evolution of the SENO algorithm was to implement some sort of preconditioning. The idea was to use the exact representation of the solution as weighted average of the previous step. The corresponding formula is described in equation (3.7.4). In Fig. 4.8 two different stencils are compared with the previous *SENO2* method. The graph corresponding to *SENO3* represents the preconditioning with the 5 point stencil (i.e. $r = 2$), whereas the third graph displays the 3 point stencil with $r = 1$.
The effect of this preconditioning is indeed profound. Courant numbers up to three times as high as before can now be used while still maintaining a decent level of accuracy. The relative error at the highest Courant numbers (~6.5) is at about 1% for both models. The additional computational effort is again quite low for the smooth cosine model. However this is not true for the low Courant number region in case of the cylinder model. The largest drawback is the error increase for $0.5 \leq C \leq 1.2$. Note also the two spikes that occur in the computation time. During earlier stages of the development they were some-

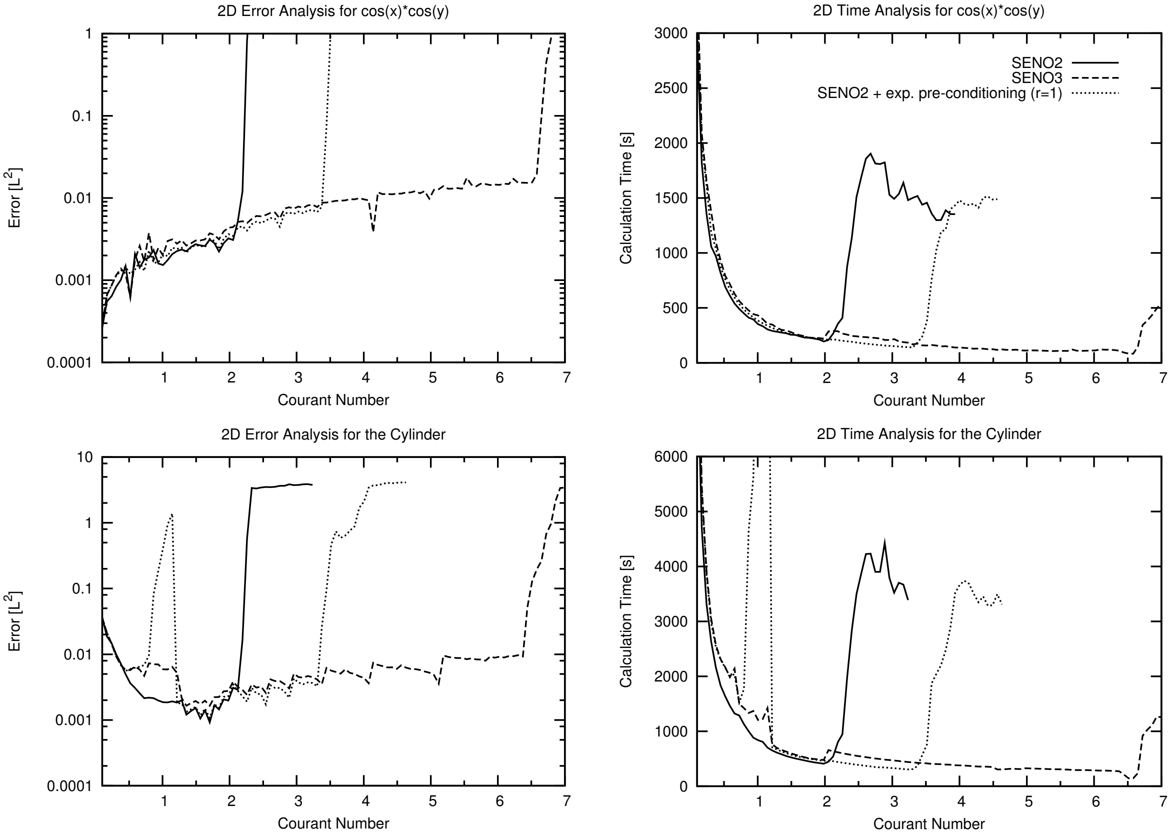

Figure 4.8: SENO2 with preconditioning

times even higher, in the worst case up to ten times more than expected.

The 3 point stencil method shows even more problems for Courant numbers at approximately 1. The source of this issue are small oscillations that spread continuously across the whole domain and which are not reduced by the spline smoothing algorithm. The best way of tackling these is to add dependent spline smoothing to the method as will be shown below.

The 3 point stencil provides a 1.5 times larger maximum Courant number whereas the 5 point stencil gives us a gain by a factor 3. Larger stencils have not been tested due to the already considerably large total stencil.

The final modification to the algorithm is the so-called dependent smoothing. The additional algorithm was described in Section 3.7.1. The following three graphs are presented in Fig. 4.9.

- *SENO3:* The SENO algorithm as above including preconditioning and compensations.
- *SENO4:* SENO3 plus dependent spline smoothing without modification of the truncation error.

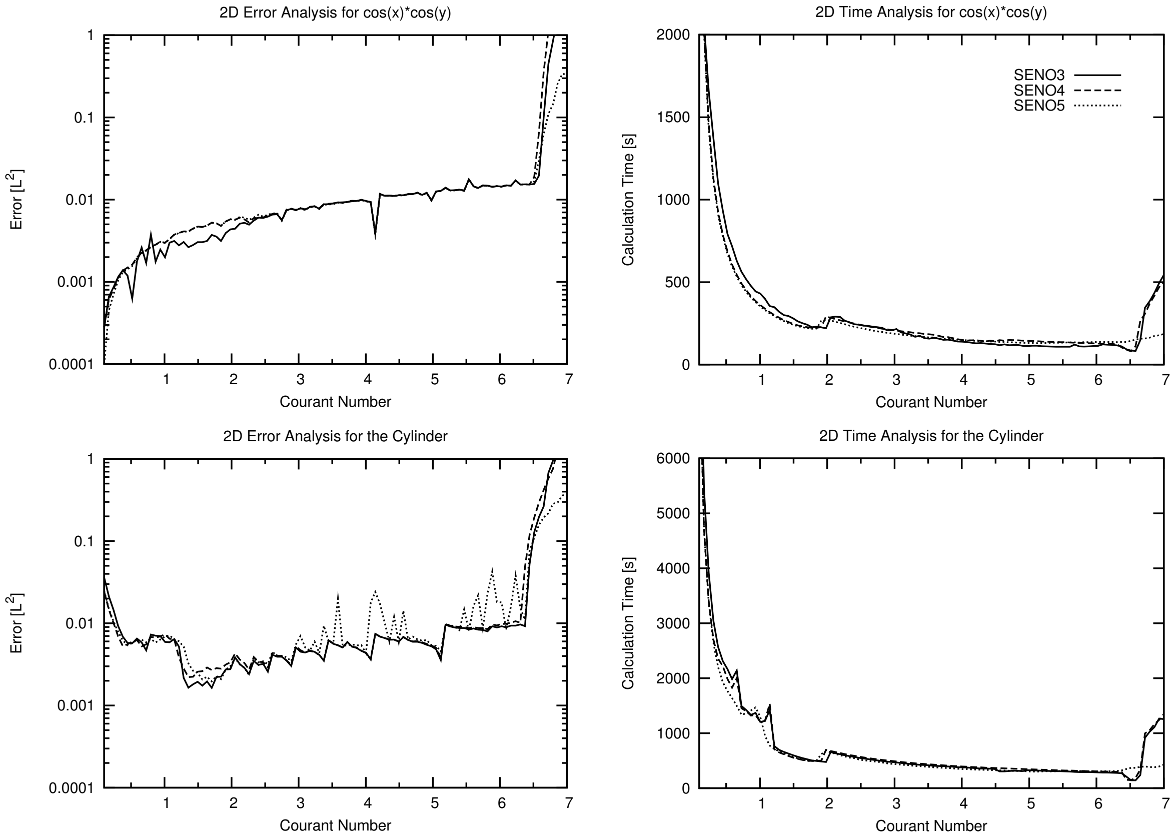

Figure 4.9: SENO3 with dependent smoothing

- *SENO5:* SENO4 including the truncation error reduced by a factor of ten.

As seen in Fig. 4.9 the only advantage of this method is that the computation time is slightly lower. Note specifically that the spikes in the lower right panel at about $C = 0.6$ and $C = 1.1$ completely vanish. However if the Courant number is larger than 3 this additional modification is no longer valuable since the error increases drastically in the discontinuous model. Thus the adaptive SENO method as presented in Section 3.7.1 will be the final step in the development.

Fig. 4.10 is important insofar as it shows how good the convergence of the method is. In this figure several simulations with various resolutions can be seen. Up to now the number of nodes in each direction (nx) was 128. This time graphs with $nx = 64$ and $nx = 256$ were added. Note that this time the computation time is on a logarithmic scale as well. For the analysis let's start with the upper column where the cosine model is used. The errors and times scale nicely across the whole domain of Courant numbers. This means that the distance between the errors is approximately the same which shows that the method appears to be converging. Of course more simulations would be necessary to get an estimate for the convergence speed. But due to the large computation times

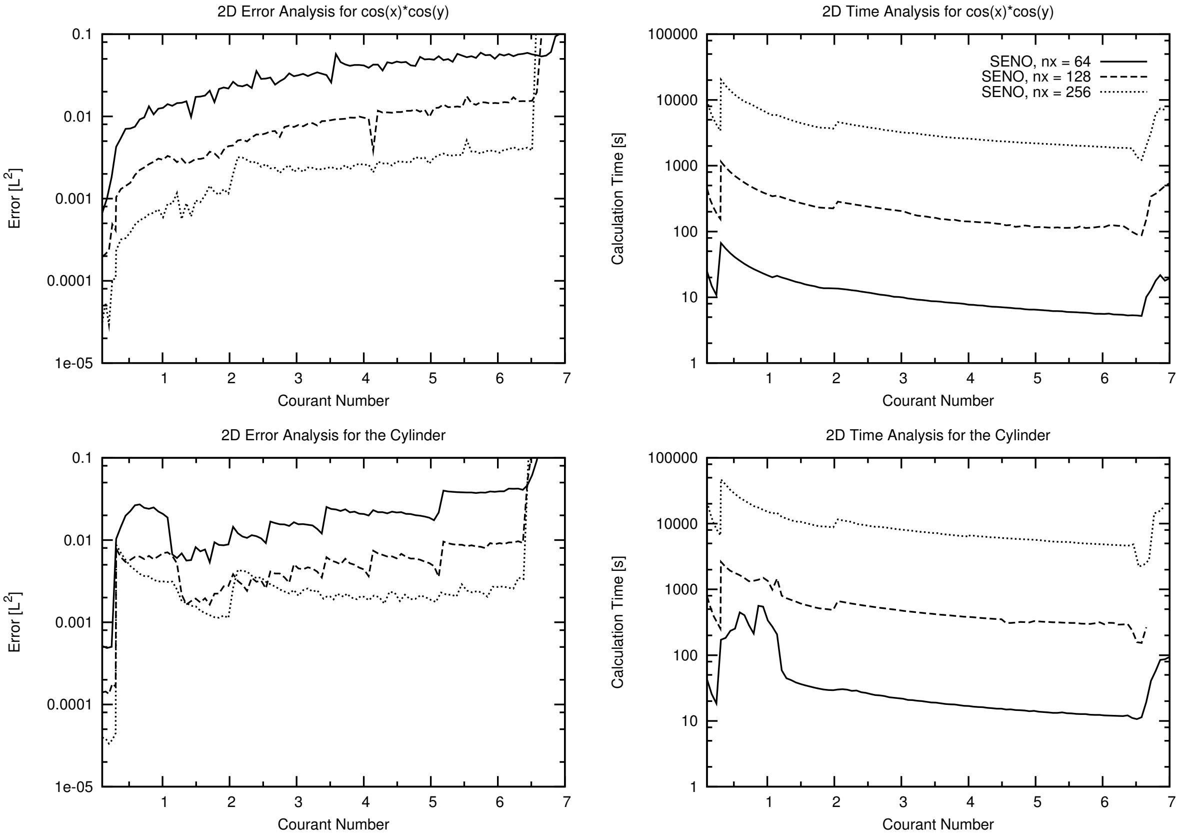

Figure 4.10: SENO with multiple resolutions

this is not possible without parallelizing the code. The only deviation can be seen at $C \approx 2.1$ where the error for the $nx = 256$ model is higher than expected. Note that the maximum Courant numbers are nearly identical, only a slight increase can be seen for lower resolutions.

Also in the cylinder simulations we see a deviation from the expected computation time for the high resolution simulation. Interestingly the errors are considerably larger than expected and there is again a peak at $C \approx 2.1$. It would be interesting to have a simulation with an even higher resolution ($nx = 512$), but due to the large computation time this could not be realized. Since the scaling behaviour is nicer when moving to higher Courant numbers the previous issues can be held accountable for this effect. To support this theory the last simulations in this section should be considered.

The first simulation shown in Fig. 4.11 has a delta spike as initial condition, i.e.

$$u_0(x,y) \quad = \quad \begin{cases} 1 & \text{if } (x,y) = (0,0), \\ 0 & \text{otherwise,} \end{cases} \tag{4.1.5}$$

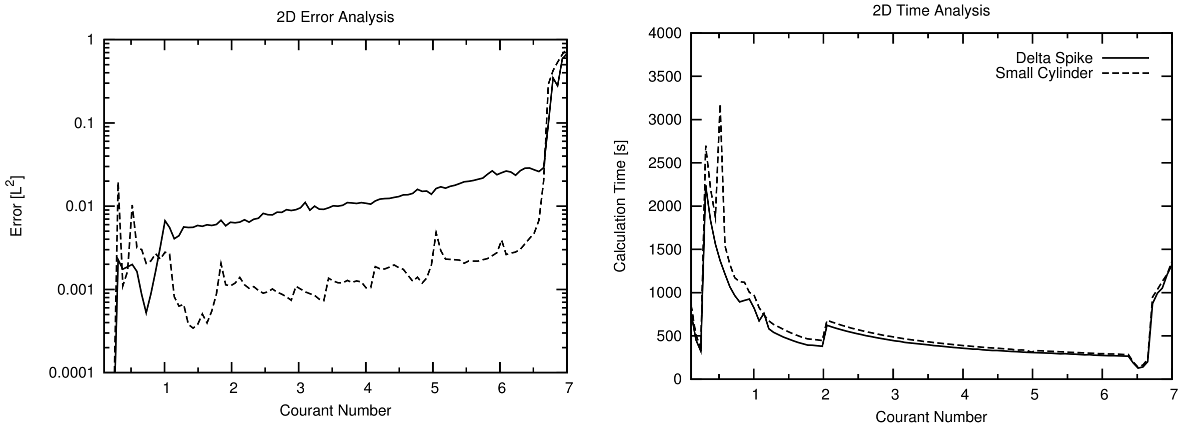

Figure 4.11: Delta spike and small cylinder simulated with SENO

for $(x, y) \in [-2, 2]^2$.

The second simulation features a cylinder with a radius of five nodes.

As mentioned in the previous section it is expected that there are difficulties for the SENO method in the low Courant regions. For the small cylinder it can indeed be seen in Fig. 4.11 that the error is very large for $C < 1.1$. What is true moreover is that the error only becomes uniform as soon as $C > 2$, which coincides with the spike in computation time. Compared to that the delta spike behaves much nicer although it features a considerably larger overall error. It is thus possible to conclude that the delta spike does not cool with the correct speed since WENO methods are not able to resolve such structures correctly. On the other hand WENO schemes are designed for maintaining discontinuities and due to the weaker preconditioning for low Courant numbers this results in a larger error for the small cylinder in this regime.

If further work is done on the SENO methods then this would be an important model to consider. Especially in the low Courant number regime ($C < 2$) further investigations have to be undertaken in order to optimize the performance of the newly presented schemes.

4.2 Simulations in 1D

In this section a few simulations in 1D shall be analyzed. Note that the simulations are actually conducted with the 2D code but with $u_0(x, \cdot) = const.$ The number of subdivisions in y direction were fixed as 60. As a result the errors presented below are actually larger than for a real one dimensional simulation (by a factor of 60).

The first objective of this section is to show the relation of the maximum Courant number to the value κ.

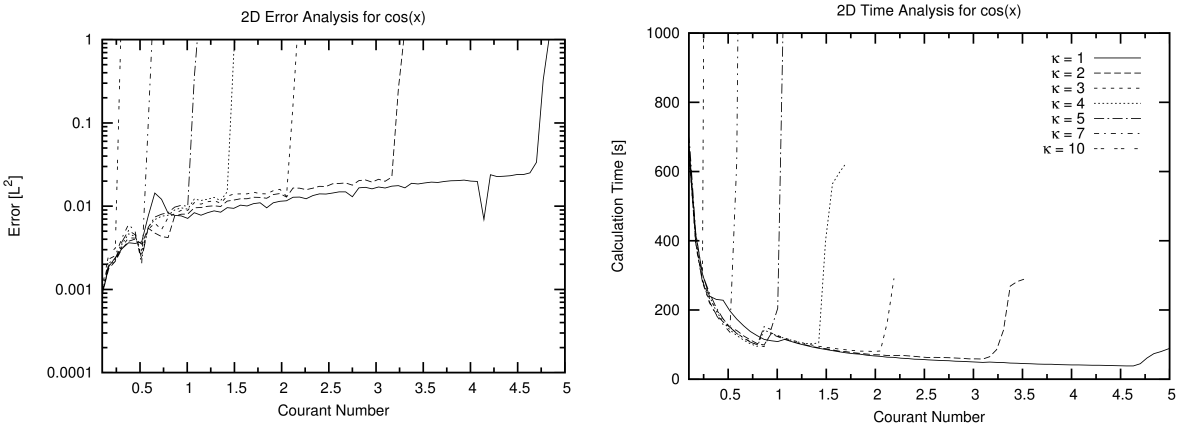

Figure 4.12: Relation between κ and C

The stability criterion for the FTCS scheme is

$$\frac{\kappa \Delta t}{\Delta x^2} \leq a \tag{4.2.1}$$

for constant κ and $a = 1/2$. The question now is whether the same holds true for the SENO scheme if only the constant a is modified. To analyze this we present several simulations in Fig. 4.12 varying κ and using

$$u_0(x, y) = \cos(x) \tag{4.2.2}$$

as initial condition. Note that we do not apply the adaptive algorithm here since it would distort the results. Thus we use the *SENO3* algorithm presented in the previous section which allowed the largest Courant numbers. For the rest of this section the Courant number will only represent the ratio $\Delta t/\Delta x^2$.

The first notable difference is that the maximum Courant number is lower than the one for the 2 D simulations. This is a result of the different definitions of the Courant number in different dimensions. For $\kappa = 1$ we now have C_∞ (i.e. the maximum Courant number) equal to 4.7.

Referring to Table 4.2 we can see that the predicted Courant numbers ($= 4.7/\kappa$) are actually lower than the ones calculated (with accuracy of 0.035) for κ smaller than 5. However it is clear that the calculated C_∞ do not decrease linearly. However for larger κ the maximum possible Courant number no longer reaches the expected values. This is an issue which will require further attention.

To conclude this section we will show two more simulations. The first one will be the 1D analogy to the cylinder model presented in Section 4.1, namely a square wave. The initial

Table 4.2 Predicted and actual values for C_∞

κ	Predicted C_∞	Calculated C_∞
1		4.7
2	2.35	3.17
3	1.57	2.05
4	1.18	1.42
5	0.94	0.94
7	0.67	0.52
10	0.47	0.24

condition is defined as

$$u_0 = \operatorname{sign}(\cos(x)) \quad x \in [-\pi, \pi] \tag{4.2.3}$$

again using 128 points in x and 60 in y direction.

Again the most outstanding feature of the plots in Fig. 4.13 is the large error and computation time for low Courant numbers. This corresponds to the issue that was already discussed in Section 4.1. Note that here the maximum Courant number C_∞ is somewhat smaller than for the smooth cosine model.

The last simulation uses a sawtooth function, i.e.

$$u_0 = x - \operatorname{floor}(x) \quad x \in [0, 1] \tag{4.2.4}$$

as initial condition. To get data which is more comparable to the rest of this work, we extended the domain to $[-2, 2]$ and stretched the initial condition such as to get only one period inside the domain.

Again as we have seen for all the other discontinuous models there is a significant spike in the low Courant regions with considerable deviations in the L^2 error. Compared to the

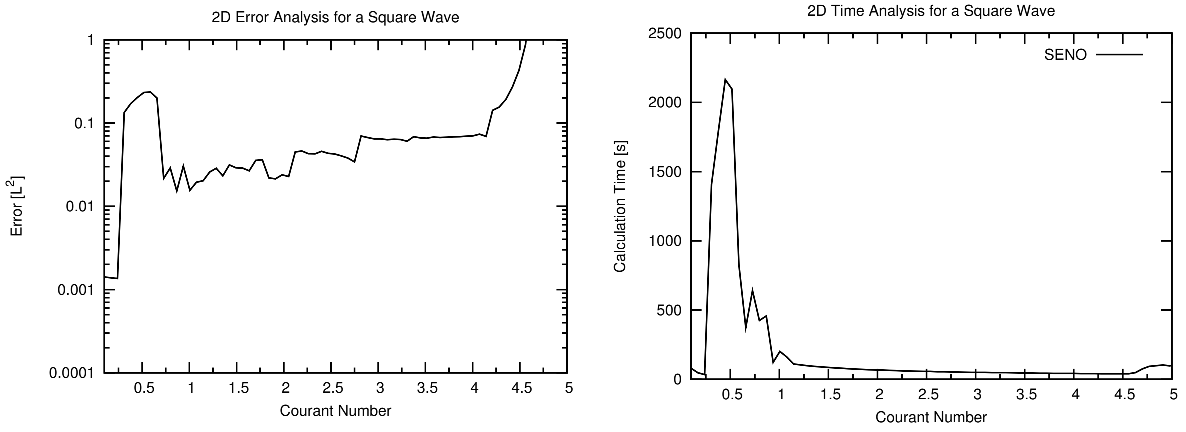

Figure 4.13: Simulation of a square wave

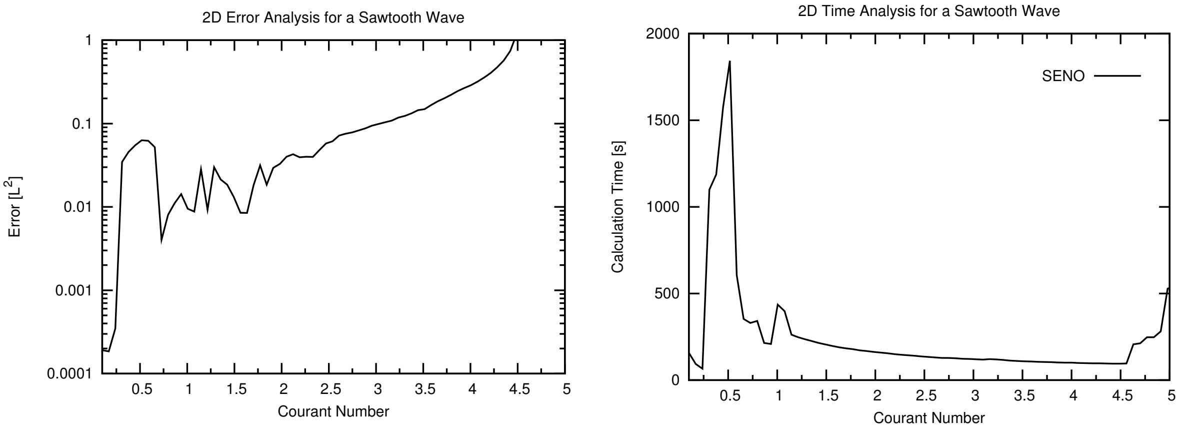

Figure 4.14: Simulation of a sawtooth wave

square wave model the error grows faster and more evenly which can be expected due to different type of discontinuity, i.e. $\frac{du_0}{dx} = 1/4 > 0$ on either side.

Chapter 5

Conclusion and Outlook

The main goal of this thesis was to increase the maximum possible time-step for a high order, local and explicit solver for the diffusion equation. After examining multiple properties of the equation and its solutions by means of theoretical analysis several numerical methods were presented in Chapter 3, eventually leading to the description of the Smoothed Essentially Non-Oscillatory schemes. The main idea was to treat the diffusion equation as conservation law and use a combination of WENO methods and Runge–Kutta algorithms for approximating the solution. In the chapter where the simulations were presented the effect of adding a preconditioning and a spline smoothing algorithm was discussed in detail. Furthermore it was shown that the maximum Courant number could be increased by a factor of approximately 20 when compared to the sixth order central difference stencil with Heun's method as time integrator.

Although so far the results are highly promising there are a few issues which will need to be addressed as part of future research. First of all there is the issue that discontinuities are prone to induce large errors in the low Courant number region. Additional focus is also required on the relation between κ and C_∞. It should be possible to obtain an analytical lower bound for the maximum Courant number depending upon κ.

Besides these more major issues a few optimizations should be investigated. First of all the 41 point stencil of the SENO method is quite large. Maybe it is possible to reduce it by adapting the preconditioning so that it only affects the points contained in the WENO stencil. Secondly these weights would need some further attention to analyze whether they really offer the best possible values. It might be possible to minimize the problems showing up for discontinuous initial conditions when using different weights.

The most time consuming algorithm is by far the spline smoothing. This is due to the fact that it needs several iterations until the weights converge. If it were possible to obtain an approximate relationship between the weights and the truncation error, some sort of preconditioning could be implemented resulting in fewer iterations.

Furthermore it has been noted that introducing Dirichlet boundary conditions will alter the integral compensation algorithm. The possibility of modifying this algorithm for arbitrary boundary conditions should be explored in future work.
However, compared to the initial time-stepping restrictions these remaining issues are only minor. Thus it is safe to say that the SENO methods provide a powerful and efficient algorithm for solving the diffusion equation.

Bibliography

[1] Fourier, J. *The Analytical Theory of Heat* (Cambridge University Press, Cambridge, 1878).

[2] Evans, L. C. *Partial Differential Equations* (American Mathematical Society, Providence, 2002).

[3] Demtroeder, W. *Experimentalphysik 1* (Springer, Berlin, 2006).

[4] Strikwerda, J. C. *Finite Difference Schemes and Partial Differential Equations* (Wadsworth & Brooks/Cole, Pacific Grove, 1989).

[5] Gottlieb, S. & Shu, C.-W. Total variation diminishing Runge–Kutta schemes. *Math. Comput.* **67**, 73–85 (1998).

[6] Ketcheson, D. I. & Robinson, A. C. On the practical importance of the SSP property for Runge–Kutta time integrators for some common Godunov-type schemes. *Int. J. Numer. Methods Fluids* **48**, 271–303 (2005).

[7] Liu, X.-D., Osher, S. & Chan, T. Weighted essentially non-oscillatory schemes. *J. Comput. Phys.* **115**, 200 (1994).

[8] Harten, A., Engquist, B., Osher, S. & Chakravarthy, S. R. Uniformly high order accurate essentially non-oscillatory schemes, III. *J. Comput. Phys.* **131**, 3–47 (1987).

[9] Jiang, G.-S. & Shu, C.-W. Efficient implementation of weighted ENO schemes. *J. Comput. Phys.* **126**, 202–228 (1996).

[10] Titarev, V. & Toro, E. WENO schemes based on upwind and centred TVD fluxes. *Comput. Fluids* **34**, 705–720 (2005).

[11] Shu, C.-W. High order weighted non-oscillatory schemes for convection dominated problems. *SIAM Rev.* **51**, 82–126 (2009).

[12] Harder, D. W. Numerical analysis for engineering (2010). URL `http://www.ece.uwaterloo.ca/~dwharder/NumericalAnalysis/`.

[13] Oliveira, M., Su, J., Xie, P. & Liu, C. Truncation error, dissipation and dispersion terms of fifth order WENO and of WCS for 1d conservation law. *Int. J. Comput. Math.* 1–14 (2008).

[14] Deuflhard, P. & Hohmann, A. *Numerische Mathematik 1* (Walter de Gruyter & Co., Berlin, 1993).

[15] Hanke-Bourgeois, M. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*, vol. 2 (Teubner, Wiesbaden, 2006).

[16] Lyche, T. & Morken, K. Spline methods. Lecture Notes (2005). URL `http://folk.uio.no/in329/komp.html`.

[17] Beliakov, G. Least squares splines with free knots: global optimization approach. *Appl. Math. Comput.* **149**, 783–798 (2004).

[18] Obertscheider, C. *Essentially Non-Oscillatory Verfahren zur numerischen Lösung der Diffusionsgleichung.* Master's thesis, University of Vienna (2002).

Curriculum Vitae

Name: Arno Mayrhofer
Birth: 13th of August, 1986, in Bregenz, Austria
Citizenship: Austria
High school diploma: Bundesoberstufenrealgymnasium Lauterach,
6923 Lauterach, Montfortplatz 16A, Austria
21st of June, 2004
Studies: University of Vienna:
  Mathematics: 10.2005 - 10.2010
  Physics: 10.2005 - 02.2007
University College Cork:
  Mathematics: 09.2008 - 05.2009 (Erasmus)
Academic Positions: 04.2010 - 08.2010: Research Assistant, University of Vienna, Austria
11.2009: Visiting Researcher, University of Manchester, United Kingdom
09.2009: Visiting Researcher, University of Vigo, Spain
06.2009 - 08.2009: Research Assistant, University of Limerick, Ireland