**Delft University of Technology**
**Faculty of Electrical Engineering, Mathematics and Computer Science**
**Delft Institute of Applied Mathematics**

# Fast and robust solution methods for the water quality equations

A thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirements

for the degree

**MASTER OF SCIENCE**
**in**
**APPLIED MATHEMATICS**

**by**

**A. Morais**

**Delft, the Netherlands**
**March 2012**

MSc THESIS APPLIED MATHEMATICS

# "Fast and robust solution methods for the water quality equations"

A. Morais

**Delft University of Technology**

**Daily supervisor**

Dr.ir. M. Borsboom

**Responsible professor**

Prof.dr.ir. C. Vuik

**Other thesis committee members**

Dr. M. Genseberger

Ir. L. Postma

Dr. P. Wilders

March 2012

Delft, the Netherlands

# Contents

# Acknowledgement

I would like to thank some important people who helped me in order to finish my Master Thesis. Without them it was not possible to accomplish my graduation work.

First of all, I would like to thank Kees Vuik for making contact with Deltares to do my thesis at this company and also I am very grateful for his intensive involvement. Week after week we discussed about my work and the results I obtained and that really helped me a lot. Further I like to thank Mart Borsboom, Leo Postma and Menno Genseberger for doing my thesis at Deltares and also for helping me a lot by giving some useful advice to look at/tackle some mathematical problems which I encounter during my project. This leads definitely to better insights into the problems I was studying.

Finally but not least, I want to thank my wife Nadia for giving me all the support I needed during the time I spent on my graduation work. Also my friend Drifa was always there to support me and therefore I am very grateful for her help.

# Chapter 1

# Introduction

Deltares is an independent institute with high knowledge in the field of water, soil and subsoil. The institute is consulted by public authorities, engineering agencies or other companies for solving problems concerning the safety and environment of the society. One particular expertise of Deltares is doing quantitative water quality research. For these assessments models are used to solve the corresponding cases. In this thesis we consider methods how to solve these water quality models. An important objective is to improve the solution methods in order to obtain an efficient and accurate estimation of the water quality.

First we present in the next chapter the water quality model. This model is a mathematical description of the transport and reaction processes of substances in the water. Together with initial and boundary conditions the model for studying the water quality is completely determined.

In Chapter 3 we will explain the problems or difficulties that we encounter for numerical methods in order to solve the water quality model. This will be explained for the time-dependent case. The main objective is to improve the existing numerical methods in their robustness and efficiency.

In Chapter 4 we present the Finite Volume Method for the discretization of the water quality equations. Some requirements will be given for the Finite Volume scheme such that the numerical scheme is mathematically and physically correct. Furthermore will we discuss in Chapter 4 several numerical methods. After that we continue with a more advanced approach based on fluxlimiters.

Since we will deal mainly with implicit numerical schemes an iterative procedure is used to solve the corresponding equations. Different iterative approaches will be treated in Chapter 5. In the subsequently chapter we look at the numerical solutions of these methods and furthermore we analyze the numerical schemes in order to find improvements, both in accuracy and in efficiency.

In the final two chapters we look back at the numerical methods and conclude which approach(es) should be used to achieve the highest accuracy and efficiency. Since not all parts of the water quality model are taken into account in the thesis we will recommend which subjects can be (further) investigated in future research.

# Chapter 2

# The water quality model

In water a lot of different substances can be found. All these substances move along each other or interact with each other. Therefore, for the study of the water quality the description is needed of the behavior of these substances in the water. The behavior is modeled in terms of transport and water quality processes. For the first kind two important types of transport can be considered: the advective transport and the diffusive transport. The former one is the transport due to the motion of the fluid. The substances are carried in the direction of the stream. The latter is transport due to random movement of the molecules and is also called molecular diffusion. The second type of processes contains physical-, chemical- and biological processes for the substances. For each substance all these transport and water quality processes can be expressed in a single equation, the so-called water quality equation. Also sources and/or sinks are included in the equation.

The so-called water quality equations are defined by advection-diffusion-reaction equations. Per relevant substance in the water we have this type of partial differential equation (PDE). This PDE describes the change of a substance due to the transport and water quality processes mentioned above. In practical situations we deal simultaneously with several substances in the water, so the water quality model consists of ditto PDE's.

Let $I$ be the total number of substances we involve in a water quality model. For every substance $i \in I$ we have the following partial differential equation

$$\frac{\partial c_i}{\partial t} - \nabla \cdot (D \nabla c_i) + \nabla(\underline{u} c_i) = p_i, \tag{2.1}$$

defined on $\underline{x} \in \Omega$ for $t \in [0, T]$.

The area of interest is given by $\Omega$, $c_i(\underline{x}, t)$ is the concentration of substance $i$ in the water, $D$ the diffusion coefficient, $\underline{u}$ the velocity vector and $p_i$ represents the water quality processes for the substance in the water. The first term of the equation describes the change in time, the second and third term of the left hand side are the diffusion and advection terms respectively. The minus sign originates from the fact that diffusion causes net transport from higher to lower concentrations.

A wide range of substances can be included in a water quality model (see also Delft3D-WAQ manual [3]), such as:

- conservative substances (salinity, chloride)
- decayable substances
- suspended sediment
- temperature
- nutrients (ammonia, nitrate, phosphate, silicate)
- organic matter
- dissolved oxygen
- algae
- bacteria
- heavy metals
- organic micro-pollutants

The term $p_i$ on the right-hand side of Equation (2.1) consists of source terms $S(t)$ and water quality processes $f_R(c_i, t)$. Changes by sources include the addition of mass by waste loads and the extraction of mass by intakes. Water quality processes convert one substance to another, so there is interaction between several substances. Therefore the function $p_i$ depends on the concentration $c_i$ of substance $i$ and may also depend on the concentration $c_j$ of other substances $j$, with $j \in I$. A wide range of these water quality processes can be given in the model, see also Delft3D-WAQ manual [3]. A few examples are:

- sedimentation
- reaeration of oxygen
- algae growth and mortality
- mineralisation of organic substances
- (de)nitrification
- adsorption of heavy metals
- volatilisation of organic micro-pollutants

Having explained the right-hand side of the water quality equation (2.1) it can be written as

$$\frac{\partial c_i}{\partial t} - \nabla \cdot (D \nabla c_i) + \nabla(\underline{u} c_i) = f_R(c_i, c_j, t) + S(t). \tag{2.2}$$

An example of a simple water quality process is the following first-order decay reaction

$$f_R(c_i, t) = -k c_i, \quad \text{with } k \in \mathbb{R} \setminus \{0\}.$$

To complete the model for the water quality both the initial and the boundary conditions must be specified. The conditions in general formulation are

$$c(\underline{x},0) = c_0(\underline{x}) \qquad (IC), \tag{2.3}$$

$$c|_{\underline{x}\in\partial\Omega} = k_1\frac{\partial c}{\partial n} + k_2 c = g(\underline{x}) \qquad (BC), \tag{2.4}$$

with $k_1(\underline{x},t)$ and $k_2(\underline{x},t)$ given functions and $g(\underline{x})$ a given function defined on the boundary $\partial\Omega$.

# Chapter 3

# Problem formulation

The water quality model is defined by partial differential equations (PDE). These are advection-diffusion-reaction equations and they describe the change in concentration due to the transport of substances in the water and their interaction with other substances. The formulation of the model is presented in the previous chapter. In this chapter we explain in general the use of numerical methods to the water quality model. Further, we deal with restrictions such that problems/difficulties arise. This all will be discussed below for the time-dependent case.

To solve the time-dependent water quality equations numerical methods are used. Together with the numerical model a grid is specified. The numerical scheme depends on the time and therefore time steps must be defined. In general a fine grid will lead to more accurate results than a coarser grid. Likewise for smaller time steps higher accuracy can be obtained than for larger time steps. Due to the highly computational costs caused by refining the grid and/or time step a more efficient approach is desired to obtain sufficient accuracy. Below we will discuss the relation between accuracy and costs for the spatial and time discretization separately.

**Spatial discretization**

There are different numerical methods for discretizing a partial differential equation (PDE) in space. For the spatial discretization one can choose to apply a low order scheme or a high order scheme. Low order schemes are less accurate compared with high order schemes. To achieve the same percentage of accuracy as the high order scheme the low order scheme must use a finer grid. The use of a finer grid may lead to higher costs. High order schemes may have due to its complex structure also high computational costs. So one must consider which numerical scheme is more suitable for solving a PDE.

As explained above computational costs are a very important aspect. The number of grid cells has a large effect on the computation time. More cells corresponds with higher costs. To obtain sufficient accuracy a specific cell size is necessary to obtain the desired accuracy. How large these cells must be can be determined by two parameters and will be explained below for the one-dimensional case. The size of the cells is denoted by $\Delta x$. The grid size and its parameters are related by

$$\Delta x = \epsilon_L L,$$

where $L$ is the length scale of the process (e.g. transport or chemical processes) we are studying and $\epsilon_L \ll 1$ is a parameter which depends on the demanded accuracy and the accuracy of the discretization in space. The length scale depends on what level the process takes place, e.g. on molecule level. Furthermore, the length scale also varies in time. For instance, the length scale increases when the processes will reach their steady state.

To reduce the computation time a larger value for $\epsilon_L$ is needed such that larger grid cells can be used. Since we know from above that this value depends on the accuracy of the spatial discretization a larger value can be reached by using a higher order spatial discretization method, e.g. high order upwind. For the discretization of the spatial terms high order schemes will have in general a larger value for $\epsilon_L$ than for low order schemes in order to have sufficient accuracy for the numerical solution.

**Time discretization**

For the time discretization a similar analysis can be done. Distinction can be made between the use of explicit and implicit numerical schemes. For explicit schemes no system of equations has to be solved whereas for implicit schemes a (non-)linear system of equations must be solved. But on the other hand, for the latter case larger time steps can be used than for the explicit case. This can be explained by the fact that for the explicit case a CFL condition must be fulfilled in order to obtain stability for the numerical solution. For the implicit case no (severe) restriction on the time step is given, hence large time steps are taken. So due to computational costs we must consider again which approach is appropriate for solving the PDE.

For the advection-diffusion equation the CFL condition depends on the following terms

$$\frac{|u|\Delta t}{\Delta x} \text{ and } \frac{D\Delta t}{\Delta x^2}. \tag{3.1}$$

This condition holds for each of the directions in space. From the CFL condition we know that when a fine grid is taken to obtain sufficient accuracy, then the time step must be very small so that the numerical solution is stable. This means that with smaller time steps more computation time is necessary to solve the PDE.

It is obviously that the time step $\Delta t$ has a large effect on the computation time. How large this time step must be can be determined by two parameters. The time step and its parameters are related by

$$\Delta t = \epsilon_T T, \tag{3.2}$$

where $T$ is the time scale of the process we are studying and $\epsilon_T \ll 1$ is a parameter which depends on the required accuracy. The time scale is for example smaller for the region near a discharge due to fast changes in the concentration, i.e. steep gradients for the concentration profile. Also it varies in time, since the time scale increases when steady state will be reached, i.e. flat gradients due to slow changes in concentration profile.

For numerical schemes we have to restrict to Condition (3.2). For the explicit schemes in particular we also deal with Condition (3.1) to determine the time step. The involvement of the latter condition may for the numerical method be not ideal, since more restrictions holds for the time step. So we have for explicit schemes

$$\Delta t \ll \epsilon_T T, \tag{3.3}$$

which means that the time step must be smaller than is needed for sufficient accuracy.

In order to have low computational costs we will use explicit schemes for the relatively larger cells, since the time step is for these cases not too small. Furthermore, no equations have to be solved. If very small time steps are necessary due to small grid cells then we prefer an implicit scheme, since then larger time steps can be taken. With this approach we can reduce the computational costs.

**Conclusion**
We may conclude that for solving the water quality model one must consider which spatial and time discretization method is appropriate such that sufficient accuracy can obtained, but not at the expense of very high computational costs. With appropriate we mean that larger values for $\epsilon_T$ and $\epsilon_L$ are desired in order to permit larger time steps and larger grid cells respectively while keeping the accuracy sufficiently high. One way to accomplish this is by using a high-order spatial discretization method and by using an explicit scheme as much as possible and implicit schemes when necessary. This will probably lead to an accurate and computationally efficient numerical solution. Our goal is to construct a numerical method that is both robust (stable and accurate) and efficient.

# Chapter 4

# Numerical methods

In Chapter 2 the water quality model is described by means of partial differential equations. Because the coefficients in the water quality model (2.1) are space and time dependent the PDE can not be solved analytically. Therefore in this thesis we will look at numerical methods to deal with this problem. One particular discretization procedure is the Finite Volume Method (FVM), which deals with the integral formulation of equation (2.1). Other important numerical methods for solving PDE's is the Finite Difference Method or the Finite Element Method. In the rest of this thesis we will only focus on FVM schemes. The Finite Volume Method will be discussed below.

## 4.1 Finite Volume Method

To apply the Finite Volume Method we first divide the domain of interest $\Omega$ completely into disjoint volume cells $V_i \subset \Omega$ . This is the first step in the Finite Volume Method and is called grid generation.

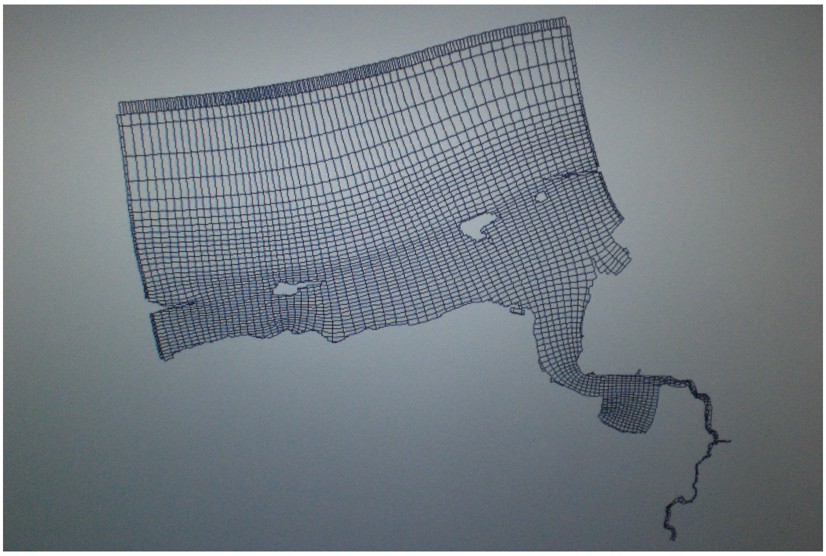

Figure 4.1: Example of a grid generation for the Eems-Dollard region

In the next step we integrate the water quality equation (2.1) piecewise over each volume cell $V_i$

$$\int_{V_i} [\frac{\partial c}{\partial t} - \nabla \cdot (D\nabla c) + \nabla(\underline{u}c)]\, dV = \int_{V_i} p\, dV, \quad i = 1, \ldots, N,$$

where $V_i$ is the volume of cell $i$ in the domain $\Omega$.

Since the time-derivative and integral can be interchanged the equation can be rewritten as

$$\frac{d}{dt}\int_{V_i} c\, dV - \int_{V_i} [\nabla \cdot (D\nabla c) - \nabla(\underline{u}c)]\, dV = \int_{V_i} p\, dV, \quad i = 1, \ldots, N.$$

Applying the Gauss' divergence theorem to the second term in the previous equation leads to

$$\frac{d}{dt}\int_{V_i} c\, dV - \oint_{\Gamma_i} [D\nabla c \ - \underline{u}c\,] \cdot \underline{n}\, d\Gamma = \int_{V_i} p\, dV, \quad i = 1, \ldots, N, \tag{4.1}$$

where $\Gamma_i$ represents the total surface area of the cell and $\underline{n}$ is the unit vector normal to the surface pointing outward.

An important property of the FVM is that due to the piecewise integration of the PDE, the equation is expressed in average values. The quantities for the water quality case are defined as

$$\begin{aligned} c_i &= \frac{1}{|V_i|}\int_{V_i} c(\underline{x}, t)\, dV, && (4.2)\\ c_{ij} &= \frac{1}{|\Gamma_{ij}|}\int_{\Gamma_{ij}} c(\underline{x}, t)\, d\Gamma, && (4.3)\\ u_{ij} &= \frac{1}{|\Gamma_{ij}|}\int_{\Gamma_{ij}} \underline{u}(\underline{x}, t)\, d\Gamma, && (4.4)\\ p_i &= \frac{1}{|V_i|}\int_{V_i} p(\underline{x}, t)\, dV. && (4.5) \end{aligned}$$

In the first equation we defined the average concentration over the $i^{th}$ cell at time $t$, in the second and third equation the average concentration and average velocity respectively over the interface of the $i^{th}$ and $j^{th}$ cell at time $t$ and in the final equation the average water quality process over the $i^{th}$ cell at time $t$. Here $|\cdot|$ represents the volume/area of the cell/boundary and $\Gamma_{ij}$ is the joint-boundary/interface of cell $i$ and $j$.

Furthermore we define the deviations for the concentration and the velocity

$$\begin{aligned} \hat{c}_{ij}(\underline{x}, t) &= c(\underline{x}, t) - c_{ij}, \quad \underline{x} \in \Gamma_{ij}, && (4.6)\\ \hat{\underline{u}}_{ij}(\underline{x}, t) &= \underline{u}(\underline{x}, t) - \underline{u}_{ij}, \quad \underline{x} \in \Gamma_{ij}, && (4.7) \end{aligned}$$

Using the definitions (4.2) and (4.5) Equation (4.1) is rewritten as

$$\frac{d|V_i|c_i}{dt} - \sum_{j \in J_i} [\int_{\Gamma_{ij}} (D\nabla c \cdot \underline{n}_{ij} \ - c\underline{u} \cdot \underline{n}_{ij})\, d\Gamma] \ = |V_i| p_i, \quad i = 1, \ldots, N. \tag{4.8}$$

If we substitute next the definitions (4.6)-(4.7) in the advection part of the summation term and use that the average deviation of the average is zero we get

$$\frac{d|V_i|c_i}{dt} - \sum_{j\in J_i}[\int_{\Gamma_{ij}} (D\nabla c\cdot \underline{n}_{ij} - \hat{c}_{ij}\hat{\underline{u}}_{ij}\cdot \underline{n}_{ij})\, d\Gamma \; - |\Gamma_{ij}|c_{ij}\underline{u}_{ij}\cdot \underline{n}_{ij}] \; = |V_i|p_i, \quad i = 1,\ldots,N. \tag{4.9}$$

The integral term in Equation (4.9) represents the new diffusion term which consist of the molecular diffusion and the turbulent diffusion (Postma 2011, [9]). In the turbulent diffusion one has the term $\hat{c}_{ij}\hat{\underline{u}}_{ij}$, which can be seen as non-normalized correlation coefficient i.e. a measure of the linear dependence between two random variables. Remember the correlation coefficient for two random variables X and Y which is defined as

$$\rho_{X,Y} = \frac{\sum_i^N (x_i - \overline{x})(y_i - \overline{y})}{\sigma_x \sigma_y},$$

where $\overline{x}$ and $\overline{y}$ are the mean and $\sigma_x$ and $\sigma_y$ are standard deviations of $X$ and $Y$ respectively. In our particular case the random variables are the deviations for the concentration and the velocity (see Equations (4.2) – (4.7)). The turbulent diffusion is a sub-grid phenomenon (Postma 2011, [9]). The magnitude of this term vanishes as the grid is refined.

For the spatial derivatives in (4.9) numerical difference formulas can be used, such as central difference or one-sided difference. Next we use a time integration method to solve equation (4.9). This numerical solution should lead to an accurate solution of the water quality model (2.1). Below we illustrate the FVM described above for a simple one-dimensional example.

**Example 4.1**

Let's consider the following one-dimensional water quality equation

$$\frac{\partial c}{\partial t} - D(x)\frac{\partial^2 c}{\partial x^2} + u(x)\frac{\partial c}{\partial x} = p(x), \tag{4.10}$$

with x defined on $\Omega$ = [a,b] and $c(x,0) = c_0(x)$.

First we divide our interval $\Omega$ in N subintervals with equidistant cell-size. In each volume cell we define a node at the center of the cell.

Integrating the equation above over a volume $V_i = (x_{i-1/2}, x_{i+1/2})$ results in

$$\frac{d}{dt}\int_{x_{i-1/2}}^{x_{i+1/2}} c\, dx - \int_{x_{i-1/2}}^{x_{i+1/2}} [D\frac{\partial^2 c}{\partial x^2} - u\frac{\partial c}{\partial x}]\, dx = \int_{x_{i-1/2}}^{x_{i+1/2}} p\, dx, \quad i = 1,\ldots,N.$$

Next we apply Gauss' divergence theorem to the previous equation. This yields

$$\frac{d}{dt}\int_{x_{i-1/2}}^{x_{i+1/2}} c\,dx - (D_{i+1/2}\frac{\partial c_{i+1/2}}{\partial x} - D_{i-1/2}\frac{\partial c_{i-1/2}}{\partial x}) + (u_{i+1/2}c_{i+1/2} - u_{i-1/2}c_{i-1/2}) = \int_{x_{i-1/2}}^{x_{i+1/2}} p\,dx.$$

Using the average method described above we obtain

$$\frac{dc_i}{dt}\Delta x - (D_{i+1/2}\frac{\partial c_{i+1/2}}{\partial x} - D_{i-1/2}\frac{\partial c_{i-1/2}}{\partial x}) + (u_{i+1/2}c_{i+1/2} - u_{i-1/2}c_{i-1/2}) = p_i\Delta x,$$

with $c_i$ and $p_i$ given by (4.2) and (4.5) respectively and $D_m = D(x_m, t)$, $u_m = u(x_m, t)$ and $\Delta x = x_{i+1/2} - x_{i-1/2}$ for an equidistant grid.

Next we assume a constant diffusion coefficient $D$ and velocity $u$ for each cell $i$. Using for example central differences for the spatial derivative and central average for the zeroth order derivative we get

$$\frac{dc_i}{dt}\Delta x - D_i\frac{c_{i-1} - 2c_i + c_{i+1}}{\Delta x} + u_i\frac{c_{i+1} - c_{i-1}}{2} = p_i\Delta x. \tag{4.11}$$

A better choice for the difference and average is possible, but at this point this is not relevant since we only illustrate how the PDE can be transformed to a system of equations. After rearrangement equation (4.11) can be written in matrix-vector form as

$$\frac{d\underline{c}}{dt} + S\underline{c} = \underline{f}, \tag{4.12}$$

where S represents a N x N matrix .

Finally one can apply a time integration method to equation (4.12). Using the $\theta-$method we can write the equation in the form

$$(\frac{1}{\Delta t}I - \theta S)\underline{c}^{n+1} = (\frac{1}{\Delta t}I - (1-\theta)S)\underline{c}^n + \underline{f}, \tag{4.13}$$

with $\theta = 0$ for Euler Forward and $\theta = 1$ for Euler Backward. For $\theta = 0.5$ we get the method of Crank-Nicolson which has a higher order of accuracy than the Euler methods (second and first order resp.). The letter $I$ is the identity matrix. The letter $n$ denotes the time $t^n$, i.e. $t^n = n\Delta t$. As starting point we use the initial condition $\underline{c}^0 = c_0(x)$. Hence we are able to solve the discretized water quality equation (4.13) for $\underline{c}$. The value $c_i^n$ is an approximate average value over the $i^{th}$ volume cell at time $t^n$.

In the example above we present a general approach for solving the one-dimensional WQM. Though, for practical situations we deal with non-linear water processes $p(x,t)$ (see Chapter 2) and therefore this approach leads to difficulties. Rather the following fractional-step approach is used to avoid solving complex non-linear equations caused by the term $p$.

Split Equation (4.10) with source term $p(x,t)$ into

$$\frac{\partial c}{\partial t} - D(x)\frac{\partial^2 c}{\partial x^2} + u(x)\frac{\partial c}{\partial x} = 0, \tag{4.14}$$
$$\frac{\partial c}{\partial t} = p(x,t). \tag{4.15}$$

By splitting the general equation one can use standard methods for each of the equations above in order to solve Equation (4.10) with non-linear source term $p(x,t)$. For the first equation we use the FVM approach discussed above and for the second equation we can use a time-integration method such as Euler Forward. Note that for each cell $i$ the equations above are solved in alternating order, i.e. first (4.14) then (4.15).

For Example 4.1 with Euler Forward as time discretization in both equations above we have

$$c_i^* = c_i^n + \Delta t[D_i\frac{c_{i-1}^n - 2c_i^n + c_{i+1}^n}{h^2} - u_i\frac{c_{i+1}^n - c_{i-1}^n}{2h}], \tag{4.16}$$
$$c_i^{n+1} = c_i^* + \Delta t p(c_i^*). \tag{4.17}$$

In the first equation with known $\underline{c}^n$ we determine solution $c_i^*$ and next this solution is used in the second equation to obtain $c_i^{n+1}$. So, solving the original problem is simplified by using this splitting method.

## 4.2 Solution properties

### 4.2.1 Convergence of FVM scheme

The FVM leads to a discrete numerical model of a partial differential equation. Several definitions will be introduced below which are important in order to measure the quality of the model.

An important requirement of a FVM scheme (or any other numerical scheme) is that local errors do not grow catastrophically and hence a bound on the global error can be obtained in terms of these local errors. If this description holds for the FVM scheme, then the numerical method is called stable. First we present the definitions for the global and local error.

The global error at a time $t^n$ is given by

$$E^n = q(x, t^n) - c(x, t^n),$$

with $q$ the approximation by the FVM scheme and $c$ the true value. The local truncation error at time $t^n$ is defined as

$$\tau^n = \frac{\mathcal{N}(c(x, t^{n-1})) - c(x, t^n)}{\Delta t},$$

where $\mathcal{N}(\cdot)$ represents the numerical operator (Leveque 2002, [8]).

Stability for a numerical method is given in the following definition (Leveque 2002, [8]).

**Definition 4.2**
Let $||\cdot||$ be some norm, then a numerical method is stable if

$$||E^n|| \leq C||E^0||, \quad \text{for all } n,$$

where $C$ is some constant.

If Definition 4.2 holds, then the global error is bounded for each time step. We have due to the boundedness that a small perturbation in the initial condition leads to a small change in the solution.

Next we discuss when the FVM scheme is consistent with the PDE. This means that the local truncation error vanishes as the grid is refined.

**Definition 4.3**
A method is called consistent with the partial differential equation if the local truncation error at time $t^n$ in some norm satisfies

$$\lim_{\Delta t \to 0, \Delta x \to 0} ||\tau^n|| = 0, \quad \text{with } x \text{ fixed.}$$

The dominant term of the truncation error determines the order of accuracy of the numerical method. We say that a method is accurate of order $s_1$ in time and accurate of order $s_2$ in space if

$$||E^N|| = O(\Delta t^{s_1}) + O(\Delta x^{s_2}), \tag{4.18}$$

where $N$ is the total number of time steps, i.e. $N\Delta t = T$ with $T$ the total time.

But what can be said about the numerical solution of a FVM scheme? Does this solution approximates sufficiently well the real unknown solution? Having stability and consistency for a numerical method we have indeed convergence of the numerical solution to the real solution according to the Fundamental Theorem of numerical methods for PDE's (Leveque 2002, [5]). This theorem can be summarized as

$$\text{consistency} \ + \ \text{stability} \implies \text{convergence.}$$

**Definition 4.4**
A method is convergent at time $t^n$ in some norm if the global error satisfies

$$\lim_{\Delta t \to 0, \Delta x \to 0} ||E^n|| = 0, \quad \text{with } x \text{ fixed.}$$

### 4.2.2 Positivity and monotonicity of FVM scheme

For the computation of the numerical solution of the water quality model it is important to obtain non-negative values. Negative values are unphysical and may cause instability for its solution.

The one-dimensional water quality model reads

$$\frac{dc}{dt} - D(x)\frac{\partial^2 c}{\partial x^2} + u(x)\frac{\partial c}{\partial x} = p(x). \tag{4.19}$$

Discretizing the equation above in space by the FVM approach we obtain an ordinary differential equation (ODE). In order to get a positive solution for the ODE it is necessary for the ODE system to be positive. In here it is important that the right-hand side of the ODE remains positive, i.e. the spatial discretization guarantees positive values. Definitions are given below by Veldhuizen 2009, [11].

**Definition 4.5**
An ODE system $\frac{dc}{dt} = F(c(t))$ is called positive, or non-negative if $c(0) \geq 0$ (component-wise) implies $c(t) \geq 0$ for all $t > 0$.

The ODE system can be solved numerically by applying a time-integration method. The definition mentioned above have to be translated to time-integration methods, such that the numerical method can be called positive.

**Definition 4.6**
A time integration method $c^{n+1} = \phi(c^n)$ is called positive if for all n $\geq 0$ holds

$$c^n \geq 0 \Longrightarrow c^{n+1} \geq 0.$$

If Definition 4.6 holds then we have a positive numerical solution which is important for water quality models. At last, the numerical FVM scheme should ensure monotonicity. This means that non-physically behavior of the numerical solution is not desired.

**Definition 4.7**
A numerical scheme is monotonicity preserving if for every non-decreasing part of the initial condition the numerical solution at later time steps remains non-decreasing for these parts. A similar definition holds for non-increasing parts.

If Definition 4.7 holds then we are ensured that the solution will maintain its proper form such that no oscillations will occur.

## 4.3 Current numerical schemes

The numerical methods we will consider must deal with steep gradients in order to describe a sudden release of a concentration in water. First we will discuss two different methods and observe that these methods cannot deal sufficiently with this behaviour. The failure of these two methods is caused by the advection term in the (discrete) water quality model which is not very capable in describing the steep gradients. Though, one of these methods will be important in the design for higher accurate numerical methods to tackle the steep gradients. Due to problems by the advection term the numerical methods will first be applied to the one-dimensional homogeneous advection equation given by

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0, \quad x \in [0,10],\ t \geq 0, \tag{4.20}$$

where $u > 0$ (substances flow from left to right) is constant and with periodic boundary conditions

$$c(0,t) = c(10,t), \quad t \geq 0, \tag{4.21}$$

and with initial condition

$$c(x,0) = 1_{[2,4]}(x)\frac{1}{2}(1-\cos(\pi x)) + 1_{[6,8]}(x), \quad x \in [0,10]. \tag{4.22}$$

Since the water quality model is a conservative system the discretization of the equation above is presented in terms of fluxes coming in and going out. Discretizing Equation (4.20) by FVM and Euler Forward leads to the following system of equations

$$c_i^{n+1} = c_i^n - \frac{\Delta t}{\Delta x_i}(F_{i+1/2}^n - F_{i-1/2}^n) \qquad i = 1, \ldots, N \tag{4.23}$$

where the term $F_{i-1/2}^n$ is some approximation of the average flux at time $t^n$ along the left boundary $x = x_{i-1/2}$ of control volume $V_i$. Further $c_i$ represents the average concentration of volume cell $V_i$. The FVM is chosen to be written in this form since we are dealing with a conservative system, hence Equation (4.23) is a conservative scheme. Therefore the fluxes at the boundaries are important to study.

### 4.3.1 The upwind method

First of all we use a simple approximation for this average flux before continuing with more accurate methods. The simplest one is the upwind method, which only uses information coming from one side, dependent on the direction of the stream.

For the upwind method for the advection term we have $F^n_{i-1/2} = uc^n_{i-1}$ and $F^n_{i+1/2} = uc^n_i$. The upwind method is first order accurate. By using the Taylor series expansion for the upwind scheme it can be shown that this numerical method introduces diffusive behavior. The modified equation yields

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = \frac{1}{2}u\Delta x(1 - u\frac{\Delta t}{\Delta x})\frac{\partial^2 c}{\partial x^2}, \tag{4.24}$$

where the right-hand side represents a diffusion term.

It is necessary for the upwind method to satisfy the CFL condition $u\frac{\Delta t}{\Delta x_i} \leq 1$, such that the numerical upwind scheme is stable and converge to the solution of the differential equation as the grid is refined. The main advantage of using the upwind method is that the solution is monotone and negative values do not appear. A disadvantage is that the upwind method leads to severe damping of the numerical solution. This is caused by the artificial numerical diffusion term in Equation (4.24). In the figure below we can observe the features mentioned above for Problem (4.20)-(4.22).

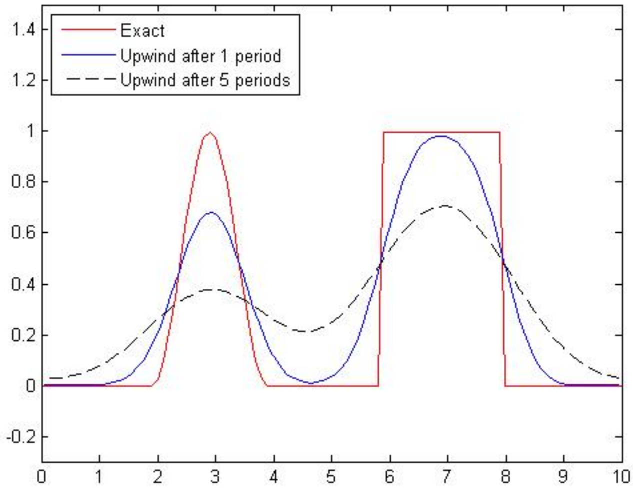

Figure 4.2: Upwind method for $\Delta x_i = 0.1$, $u = 1$ and CFL = 0.8

### 4.3.2 Lax-Wendroff method

Also more accurate flux approximations can be used such as the Lax-Wendroff method, which is a second-order accurate method. This method has an extra term to correct for the diffusive upwind part. For the Lax-Wendroff method applied to the one-dimensional advection equation with $u > 0$ we have the following flux approximation

$$F^n_{i-1/2} = \frac{1}{2}u(c^n_{i-1} + c^n_i) - \frac{1}{2}\frac{\Delta t}{\Delta x_i}u^2(c^n_i - c^n_{i-1}),$$
$$F^n_{i+1/2} = \frac{1}{2}u(c^n_i + c^n_{i+1}) - \frac{1}{2}\frac{\Delta t}{\Delta x_i}u^2(c^n_{i+1} - c^n_i).$$

The equations above can be rewritten as

$$\begin{aligned} F^n_{i-1/2} &= uc^n_{i-1} + \frac{1}{2}u(c^n_i - c^n_{i-1})(1 - u\frac{\Delta t}{\Delta x_i}), \\ F^n_{i+1/2} &= uc^n_i + \frac{1}{2}u(c^n_{i+1} - c^n_i)(1 - u\frac{\Delta t}{\Delta x_i}). \end{aligned}$$

The equations present an extra term which is absent for the upwind term. Note that this term corresponds with the artificial diffusion term in Equation (4.24). This extra term is also called an anti-diffusion term since it makes sure that the numerical diffusion caused by the upwind part is vanished, i.e. zero numerical diffusion. So we have no smearing of the numerical solution. Also for this case the CFL condition $u\frac{\Delta t}{\Delta x_i} \leq 1$ must be satisfied. The advantage of this method is that the numerical solution is more accurate than the upwind method, especially for smooth parts. On the other hand it gives wiggles near steep gradients. As a consequence we may get negative values for the numerical solution which is not desired.

In the figure below we present the features mentioned above for Problem (4.20)-(4.22).

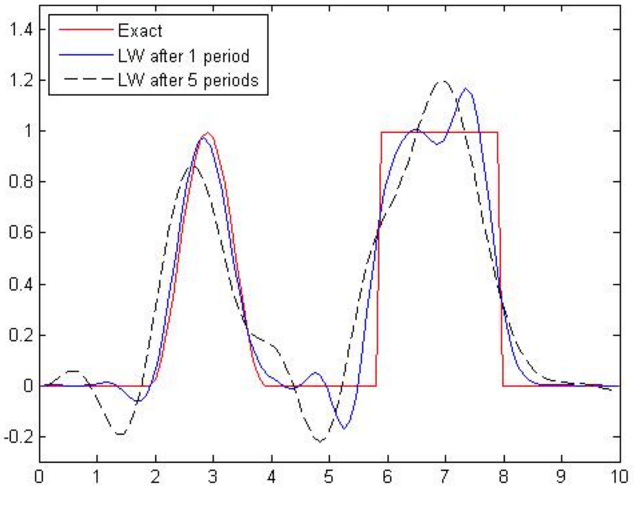

Figure 4.3: Lax-Wendroff method for $\Delta x_i = 0.1$, $u = 1$ and CFL $= 0.8$

### 4.3.3 The FCT method

Both the upwind- and Lax-Wendroff method have their limitations for dealing with the advective term. The former introduces extra diffusion of the solution, while the latter one introduces oscillations in the solution. Therefore we introduce flux limiters, which optimize the numerical solution by combining the best features of both the first-order method and the second-order method. This optimized solution can be obtained by a convex combination between a first-order flux and an high-order flux, e.g. Lax Wendroff. So in general the flux limiter method is defined as

$$F^n_{i\pm1/2} = F_L(c^n_i, c^n_{i\pm1}) + l^n_{i\pm1/2}(F_H(c^n_i, c^n_{i\pm1}) - F_L(c^n_i, c^n_{i\pm1})), \tag{4.25}$$

which is a convex combination of a low-order flux formula $F_L$ and an high-order flux formula $F_H$. The value for $l^n_{ij} \in [0, 1]$ determines the weight over the two formulas and is also called the limiter. The value for the limiter is chosen such that high accuracy is maintained as much as possible while positivity is preserved and no wiggles occur. A limiter of 0 corresponds with the low-order method and for a limiter of 1 we obtain the high-order method. The term $F_H(c^n_i, c^n_{i\pm1}) - F_L(c^n_i, c^n_{i\pm1})$ is an anti-diffusion/flux-correction term. This term corrects for the diffusive behavior made by the first-order flux method, i.e. the upwind method. Combined with a suitable choice for $l^n_{i\pm1/2}$, the upwind method with a second-order method will lead to a monotonous and positive solution with no or a less diffusive behavior. An example will be given below.

In the literature different methods are available for the computation of the limiter. One flux-limiter method is called the Total Variation Diminishing (TVD) method (Leveque 2002, [8]). Another important flux-limiter method is the Flux Corrected Transport (FCT) method (Kuzmin et al. 2004, [6]). An essential condition for the WQM is that the solution is positive. The FCT method has properties like positivity and monotonicity, whereas the TVD method has only the total-variation diminishing property. Since we will make use of implicit numerical schemes the former method is more appropriate since one can prove that non-negative solutions can be obtained. For TVD methods on the other hand this is not possible. So, in the rest of this section we will consider the FCT method .

We will present the FCT method developed by Boris and Book for the one-dimensional advection equation. The flux-limiter method is an algorithm to determine the value of the limiter $l^n_{i\pm1/2}$ in equation (4.25). The purpose of the limiter is to maintain the accuracy of the high-order method as much as possible while positivity and monotonicity are fullfilled. In fact the FCT algorithm is a method to optimize the accuracy of the solution under certain conditions.

First we discuss the method for the explicit situation. Later on we will focus on the more important implicit case. The starting point is an explicitly discretized equation for the one-dimensional advection equation (with $u > 0$ constant), which is of the form

$$c^{n+1}_i = c^n_i - \frac{\Delta t}{\Delta x_i}(F^n_{i+1/2} - F^n_{i-1/2}), \qquad i = 1, \ldots, N \tag{4.26}$$

where $F^n_{i-1/2}$ and $F^n_{i+1/2}$ is some numerical advective flux along the left and right boundary of cell $i$ respectively and with initial condition $c^0(x) = c(x, 0)$.

**The FCT method** :
Below we present the FCT method as formulated by Zalesak 1979 [14], which can easily be extended to multi dimensions. The FCT method will be applied to the one-dimensional advection equation. The algorithm is presented with upwind as first-order method and Lax-Wendroff as high-order method. Naturally, also other discretizations are possible, e.g. central spatial discretization.

1. Given the solution $c_i^n$ a time $t^n$ we first compute a first-order approximation $\tilde{c}_i^{n+1}$ of the solution at the new time step $t^{n+1}$ with $l_{ij}^n = 0$ in (4.25), i.e.

$$\tilde{c}_i^{n+1} = c_i^n - \frac{\Delta t}{\Delta x_i}(F_L(c_i^n, c_{i+1}^n) - F_L(c_{i-1}^n, c_i^n)),$$

where $F_L$ is a low-order flux function, i.e. first-order upwind. The formulas of $F_L$ are in our case given by

$$\begin{aligned} F_L(c_{i-1}^n, c_i^n) &= u_{i-\frac{1}{2}} c_{i-1}^n, \quad u_{i-\frac{1}{2}} \geq 0, \\ F_L(c_{i-1}^n, c_i^n) &= u_{i-\frac{1}{2}} c_i^n, \quad u_{i-\frac{1}{2}} < 0. \end{aligned}$$

Note that the obtained solution $\tilde{c}_i^{n+1}$ is monotone (guaranteed to give monotonic results at time $t^{n+1}$) and it will be corrected by adding an anti-diffusion term (will be derived below).

2. Next we compute the flux $F_H$ at time $t^n$ by an high-order scheme. A scheme we mentioned before is the Lax-Wendroff method. The formula $F_H$ is in this case given by

$$F_H(c_{i-1}^n, c_i^n) = \frac{u_{i-\frac{1}{2}}}{2}(c_{i-1}^n + c_i^n) - \frac{\Delta t}{\Delta x_i} u_{i-\frac{1}{2}}^2 (c_i^n - c_{i-1}^n).$$

3. Then we determine the flux correction by

$$\begin{aligned} \Delta F_{i-1/2}^n &= F_H(c_{i-1}^n, c_i^n) - F_L(c_{i-1}^n, c_i^n) \\ &= \frac{u_{i-\frac{1}{2}}}{2}(c_i^n - c_{i-1}^n)(1 - \frac{\Delta t}{\Delta x_i} u_{i-\frac{1}{2}}), \end{aligned}$$

where $F_L$ and $F_H$ are the low-order and high-order flux function respectively used in step 1 and 2. The flux correction $\Delta F_{i-1/2}^n$ is referred to as anti-diffusion, since it corrects the numerical diffusion of the low-order flux. It is important to note that the CFL condition must be satisfied, i.e. $\frac{u \Delta t}{\Delta x_i} \leq 1$.

4. In the next step we apply a pre-limiting step. Since the term $\Delta F_{ij}^n$ corrects the diffusive first-order flux it should not be diffusive. Hence we set $\Delta F_{ij}^n = 0$ if $\Delta F_{ij}^n(\tilde{c}_i^{n+1} - \tilde{c}_j^{n+1}) > 0$, with $j \in [i-1, i, i+1]$ and $\tilde{c}_i^{n+1}$ the first-order approximation for $c_i^{n+1}$ determined in step 1.

5. An important property for the solution is monotonicity, hence no new local maximum or minimum must be created or accentuated. Therefore we determine an upper and lower bound for $\tilde{c}_i^{n+1}$ computed by step 1

$$c_i^{\max} = \max_{j \in J_i} \tilde{c}_j^{n+1},$$

$$c_i^{\min} = \min_{j \in J_i} \tilde{c}_j^{n+1},$$

where $J_i$ consists of node $i$ and its nearest neighbors.

These quantities will be used in the next step.

6. Next we define the amount of mass that flows into cell $V_i$

$$P_i^+ = \max(0, \Delta F_{i-1/2}^n) - \min(0, \Delta F_{i+1/2}^n).$$

The allowed mass increase is

$$Q_i^+ = |V_i|(c_i^{\max} - \tilde{c}_i^{n+1}),$$

where $|V_i|$ is the volume of cell $i$.

The fraction of mass that is allowed to flow into the cell is given by

$$R_i^+ = \begin{cases} \min(1, \frac{Q_i^+}{P_i^+}), & P_i^+ > 0, \\ 0, & P_i^+ = 0. \end{cases}$$

For mass decrease we can define in a similar way the following quantities:

$$\begin{aligned} P_i^- &= \max(0, \Delta F_{i,i+1}^n) - \min(0, \Delta F_{i,i-1}^n), \\ Q_i^- &= |V_i|(\tilde{c}_i^{n+1} - c_i^{\min}), \\ R_i^- &= \begin{cases} \min(1, \frac{Q_i^-}{P_i^-}), & P_i^- > 0, \\ 0, & P_i^- = 0. \end{cases} \end{aligned}$$

The values $R_i^+$ and $R_i^-$ guarantees no overshoot and undershoot in cell $i$ respectively.

7. In the next step we determine the limiter which is the mass fraction that is allowed by both adjacent cells

$$l_{i-1/2}^n = \begin{cases} \min(R_i^+, R_{i-1}^-), & \Delta F_{i-1/2}^n \geq 0 \\ \min(R_{i-1}^+, R_i^-), & \Delta F_{i-1/2}^n < 0. \end{cases}$$

Note that two neighboring cells have equal limiter at the cell interface, i.e. $l_{i-1,i}^n = l_{i-1/2}^n = l_{i,i-1}^n$ at $x = x_{i-1/2}$.

8. With the previous step we finally update the solution by

$$c_i^{n+1} = \tilde{c}_i^{n+1} - \frac{\Delta t}{\Delta x_i}(F_{i+1/2}^n - F_{i-1/2}^n), \tag{4.27}$$

with

$$F_{i-1/2}^n = l_{i-1/2}^n \Delta F_{i-1/2}^n = l_{i-1/2}^n \frac{u_{i-1/2}}{2}(c_i^n - c_{i-1}^n)(1 - \frac{\Delta t}{\Delta x_i} u_{i-\frac{1}{2}}). \tag{4.28}$$

Below we illustrate the result of applying the FCT method above to the 1D advection problem (4.20 - 4.22). As low order method we used first-order upwind and as high-order method we used Lax-Wendroff. The upwind and Lax Wendroff method are presented in the figure below for comparison.

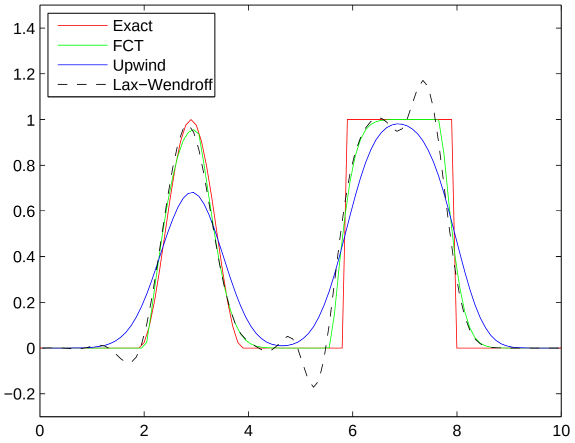

Figure 4.4: Numerical solutions with $\Delta x_i = 0.1$, $u = 1$ and CFL $= 0.8$

The results are obtained with a velocity value $u = 1$, a cfl value of 0.8, an uniform grid with 100 cells and a total of 125 time steps. We observe that the FCT solution by Zalesak gives a more accurate solution then upwind and Lax Wendroff. The wiggles are removed by the limiter, the monotonicity is maintained and the solution is less diffusive compared to the upwind solution.

### 4.3.4 FCT method extended to implicit schemes

In this section we extend the FCT method to implicit schemes, whereas in the previous section only explicit schemes are studied. We will briefly present the numerical schemes and the corresponding difficulties that arise. This methodology is discussed in more detail in Slingerland, 2007 [10].

In order to apply the FCT method one first needs an implicit representation for the advection equation. One useful description is by the theta approach. This method uses a $\theta$-parameter which determines the degree in which information is used from the new time level. The formulation in terms of fluxes is for the 1D case as follows

$$\frac{c_i^{n+1}-c_i^n}{\Delta t} = -\frac{1}{\Delta x_i}[(1-\theta)(F_{i+1/2}^n - F_{i-1/2}^n) + \theta(F_{i+1/2}^{n+1} - F_{i-1/2}^{n+1})], \tag{4.29}$$

where $\theta \in [0,1]$ and $\Delta t = t_{n+1} - t_n$. For $\theta = 0$ we have the original explicit scheme (4.26), which is Euler Explicit. For other values such as $\theta = \frac{1}{2}$ and $\theta = 1$ we have Crank-Nicolson and Euler Implicit respectively.

Taking for example an upwind flux in the equation above, i.e. $F_{i-1/2}^n = uc_{i-1}^n$ for $u \geq 0$, we get an upwind implicit scheme ($\theta \neq 0$). For central fluxes we have $F_{i-1/2}^n = uc_{i-1/2}^n = u\frac{c_{i-1}^n + c_i^n}{2}$, which leads to a central implicit method ($\theta \neq 0$). In a similar way we can use fluxes according to Lax-Wendroff.

With the help of a Taylor-series expansion we can derive for each choice for the fluxes mentioned above a modified equation for the advection equation. This modified equation is important, since it describes more accurately the behavior of the numerical solution than the actual advection equation. By inserting a function $\nu(x,t)$ in the numerical method instead of $c$ and expanding next these terms in Taylor series around $(x_i, t^n)$ we can derive the modified equation.

The reason we discuss the modified equations is that in the FCT method a convex combination is used between the upwind flux and a high-order flux, e.g. central fluxes. As consequence we have more insight in the effect of a specific value for theta on the numerical solution, i.e. the presence and the amount of numerical (anti-)diffusion in the numerical solution of the advection equation.

Below we present three modified equations with upwind, central and Lax-Wendroff fluxes respectively.

1. Combining the theta scheme (4.29) with upwind fluxes leads to the following equation

$$\frac{\partial \nu}{\partial t} + u\frac{\partial \nu}{\partial x} = \frac{u\Delta x}{2}(1 - (1-2\theta)\frac{u\Delta t}{\Delta x})\frac{\partial^2 \nu}{\partial x^2}. \tag{4.30}$$

The term given on the right-hand side of the modified equation represents the numerical diffusion. For $\theta = 0$, the explicit case, the numerical solution will be diffusive provided that the CFL-condition is satisfied. For $\theta \geq \frac{1}{2}(1 - \frac{\Delta x}{u\Delta t})$ the numerical solution is diffusive, independent of the size of the time-step and cell-size. Furthermore we can note that the amount of numerical diffusion increases with theta. But on the other hand, refining the grid ($\Delta x \to 0$) leads to vanishing of the numerical diffusion.

2. Combining the theta scheme with central fluxes leads to the following equation

$$\frac{\partial \nu}{\partial t} + u\frac{\partial \nu}{\partial x} = (\theta - \frac{1}{2})u^2 \Delta t \frac{\partial^2 \nu}{\partial x^2}. \tag{4.31}$$

For the central discretization case the numerical diffusion is diffusive for $\theta > \frac{1}{2}$ and anti-diffusive for $\theta < \frac{1}{2}$. For $\theta$ equal to $\frac{1}{2}$ the numerical diffusion is zero. Also in this case the numerical diffusion increases with theta.

3. Combining the theta scheme with Lax-Wendroff for the fluxes leads to the following equation

$$\frac{\partial \nu}{\partial t} + u\frac{\partial \nu}{\partial x} = \theta u^2 \Delta t \frac{\partial^2 \nu}{\partial x^2}. \tag{4.32}$$

The numerical diffusion term is only zero for the Euler Forward method ($\theta = 0$). For larger values of theta the numerical solution is diffusive. Similar as in the other modified equations, the amount of numerical diffusion increases with theta.

Next we insert the limiter in the fluxes $F^n_{i\pm1/2}$ and $F^{n+1}_{i\pm1/2}$ of Equation (4.29). The fluxes are formulated as

$$\begin{align} F^n_{i-1/2} &= F_L(c^n_{i-1}, c^n_i) + l^n_{i-1/2}(F_H(c^n_{i-1}, c^n_i) - F_L(c^n_{i-1}, c^n_i)), \tag{4.33}\\ F^{n+1}_{i-1/2} &= F_L(c^{n+1}_{i-1}, c^{n+1}_i) + l^{n+1}_{i-1/2}(F_H(c^{n+1}_{i-1}, c^{n+1}_i) - F_L(c^{n+1}_{i-1}, c^{n+1}_i)), \tag{4.34} \end{align}$$

where the limiter $l_{ij}$ is determined according to Zalesak (Section 4.3.3) and $F_H$ and $F_L$ correspond with a high-order and upwind-flux function respectively. Using for example upwind and Lax-Wendroff for the low-order and high-order fluxes respectively and $u_{i-1/2} > 0$ we get

$$\begin{align} F^n_{i-1/2} &= F_L(c^n_{i-1}, c^n_i) + l^n_{i-1/2}(F_H(c^n_{i-1}, c^n_i) - F_L(c^n_{i-1}, c^n_i)), \tag{4.35}\\ &= u_{i-1/2}c^n_{i-1} + l^n_{i,i-1}\frac{u_{i-1/2}}{2}(c^n_i - c^n_{i-1})(1 - \frac{\Delta t}{\Delta x_i}u_{i-\frac{1}{2}}), \tag{4.36} \end{align}$$

and

$$\begin{align} F^{n+1}_{i-1/2} &= F_L(c^{n+1}_{i-1}, c^{n+1}_i) + l^{n+1}_{i-1/2}(F_H(c^{n+1}_{i-1}, c^{n+1}_i) - F_L(c^{n+1}_{i-1}, c^{n+1}_i)), \tag{4.37}\\ &= u_{i-1/2}c^{n+1}_{i-1} + l^{n+1}_{i,i-1}\frac{u_{i-1/2}}{2}(c^{n+1}_i - c^{n+1}_{i-1})(1 - \frac{\Delta t}{\Delta x_i}u_{i-\frac{1}{2}}). \tag{4.38} \end{align}$$

After substitution of these fluxes in Equation (4.29) we get an implicit FCT scheme. Note that in this case we are dealing with limiters on two different time-levels, i.e. $l^n_{i-1/2}$ and $l^{n+1}_{i-1/2}$. So a direct application of the FCT methodology described in the previous section is not obvious. A simple approach according to Slingerland 2007 [10] will be discussed below.

**Implicit FCT method** :
1. Given the solution $c^n_i$ a time $t^n$ we first compute a first-order approximation $\tilde{c}^{n+1}_i$ of the

solution at the new time step $t^{n+1}$ with $l^n_{ij} = l^{n+1}_{ij} = 0$ in (4.33)-(4.34), i.e.

$$\tilde{c}^{n+1}_i = c^n_i - \frac{\Delta t}{\Delta x_i}[\{(1-\theta)F_L(c^n_i, c^n_{i+1}) + \theta F_L(\tilde{c}^{n+1}_i, \tilde{c}^{n+1}_{i+1})\} - \{(1-\theta)F_L(c^n_{i-1}, c^n_i) + \theta F_L(\tilde{c}^{n+1}_{i-1}, \tilde{c}^{n+1}_i)\}].$$

Note that the unknown solution $c^{n+1}_i$(belonging to the implicit part) is approximated/replaced by $\tilde{c}^{n+1}_i$. Since $F_L$ is a linear function, the equation above can simply be solved.

2. Next we define the flux correction by

$$\Delta F^n_{i-1/2} = (1-\theta)(F_H(c^n_{i-1}, c^n_i) - F_L(c^n_{i-1}, c^n_i)) + \theta(F_H(\tilde{c}^{n+1}_{i-1}, \tilde{c}^{n+1}_i) - F_L(\tilde{c}^{n+1}_{i-1}, \tilde{c}^{n+1}_i)). \quad (4.39)$$

Note that instead of the unknown solution $c^{n+1}_i$ the first order approximation $\tilde{c}^{n+1}_i$ is used in the second term of the right-hand side.

3. Next we apply steps 4-7 of the FCT method given in Section 4.3.3 to obtain the limiter with $l^{n+1}_{i-1/2} = l^n_{i-1/2}$.

4. Finally we update the solution by

$$c^{n+1}_i = \tilde{c}^{n+1}_i - \frac{\Delta t}{\Delta x_i} l^n_{i-1/2}(\Delta F^n_{i+1/2} - \Delta F^n_{i-1/2}), \quad (4.40)$$

where $\Delta F^n_{i\pm1/2}$ is defined in step 2.

The implicit FCT method is simple since the first-order approximation $\tilde{c}^{n+1}_i$ is used instead of the actual solution $c^{n+1}_i$. Though, reasonable accuracy can be obtained for the advection equation by this approach (Slingerland, 2007 [10]). In the final section of this chapter another method will be discussed for updating the numerical solution to the next time-step. This approach will be more complex, but higher accuracy is possible than with the implicit FCT method discussed above.

### 4.3.5 Local-theta scheme

In the previous section we observed from the modified equations that increasing the value of theta leads to a larger amount of numerical diffusion. This implies that using an implicit scheme leads to a decrease in the accuracy of the numerical solution. Though, the benefit of applying an implicit scheme is that larger time-steps are allowed, which results in a lower computation time. This implies that an optimal theta value must be available, such that the numerical diffusion is minimized but large enough to allow larger values for the time-step.

In (Slingerland, 2007 [10]) a strategy was defined for determining the optimal $\theta$. This strategy says that $\theta$ must be chosen as small as possible to minimize the amount of numerical diffusion, but large enough to ensure that the scheme is stable, positivity preserving and non-oscillatory. In this way the accuracy of the theta scheme (4.29) is improved without loss of robustness. The strategy for the one-dimensional case is given by

$$\theta \geq 1 - \frac{\Delta x_i}{u\Delta t}.$$

For a non-uniform grid with a constant theta we conclude from the formula above that the value for $\theta$ in the entire computational domain is determined by the smallest cell. This means that for larger cells the value for $\theta$ is much larger than necessary. An unnecessarily larger value for $\theta$ corresponds with an unnecessarily larger amount of numerical diffusion for these cells. Therefore a value per volume interface is used, i.e. $\theta$ is space dependent. In (Slingerland, 2007 [10]) this strategy is given by

$$\theta_{i+1/2} = \max\{0, \theta_i, \theta_{i+1}\}, \tag{4.41}$$

with

$$\theta_i = 1 - \frac{\Delta x_i}{u\Delta t}, \tag{4.42}$$

where $\theta_{i+1/2}$ is the value for theta at the boundary interface of cell $i$ and $i+1$ and $\theta_i$ represents the value for the $i$-th cell. In the fully implicit case ($\theta = 1$) any size for the time-step can be taken, since in this situation Equation (4.42) is unconditionally satisfied. For further details of the derivation of the strategy described above we refer to [10].

Using a local theta value per flux instead of a constant theta value gives us the local-theta method. For the one-dimensional advection equation this is given by

$$\frac{c_i^{n+1} - c_i^n}{\Delta t} = -\frac{1}{\Delta x_i}[((1-\theta_{i+1/2})F_{i+1/2}^n - (1-\theta_{i-1/2})F_{i-1/2}^n) + (\theta_{i+1/2}F_{i+1/2}^{n+1} - \theta_{i-1/2}F_{i-1/2}^{n+1})], \tag{4.43}$$

where $F_{i-1/2}^n$ and $F_{i+1/2}^n$ are some numerical advective fluxes along the left and right boundary of cell $i$ respectively. For a constant theta we retrieve the original theta scheme (4.29).

The local-theta scheme presented above can be improved by incorporating flux-limiters computed by the FCT method. The flux-limiters make sure that a larger reduction of the numerical diffusion can be obtained. The numerical fluxes $F_{i-1/2}^n$ and $F_{i-1/2}^{n+1}$ are defined by

$$\begin{aligned} F_{i-1/2}^n &= F_L(c_{i-1}^n, c_i^n) + l_{i-1/2}^n(F_H(c_{i-1}^n, c_i^n) - F_L(c_{i-1}^n, c_i^n)), & (4.44)\\ F_{i-1/2}^{n+1} &= F_L(c_{i-1}^{n+1}, c_i^{n+1}) + l_{i-1/2}^{n+1}(F_H(c_{i-1}^{n+1}, c_i^{n+1}) - F_L(c_{i-1}^{n+1}, c_i^{n+1})), & (4.45)\end{aligned}$$

where the limiters $l_{i-1/2}^n$ and $l_{i-1/2}^{n+1}$ can be determined by the implicit FCT method described in the previous section. Furthermore, $F_H$ and $F_L$ corresponds with a high-order and a low-order flux function respectively.

### 4.3.6 Numerical methods compared

In the previous sections some numerical schemes were presented for solving the 1D advection equation. In this section we discuss the advantages and disadvantages of the different types of numerical schemes presented in this chapter. In the table below a comparison between the methods are based on the following properties: accuracy, diffusiveness, monotonicity, positivity and efficiency. The table can be seen as a global summary of the numerical methods. All the numerical methods are based on the local-theta scheme (4.43) and differs only in the formulation of the fluxes.

Table 4.1: Numerical methods applied to the 1D problem (4.20)-(4.22)

| *Numerical scheme* | *Accuracy* | *Diffusive* | *Monotone* and positive | *Efficient* |
|---|---|---|---|---|
| Upwind explicit | $O(\Delta t)$,<br>$O(\Delta x)$ | Yes | Yes | Only for small timescale |
| Lax-Wendroff explicit | $O(\Delta t^2)$,<br>$O(\Delta x^2)$ | No | No | Only for small timescale |
| FCT explicit | $O(\Delta t)$ - $O(\Delta t^2)$ ,<br>$O(\Delta x)$ - $O(\Delta x^2)$ | Only with LW fluxes | Yes | Only for small timescale |
| Upwind implicit | $O(\Delta t)$ - $O(\Delta t^2)$,<br>$O(\Delta x)$ | Yes | Yes | Not for small timescale |
| Central implicit | $O(\Delta t)$ - $O(\Delta t^2)$,<br>$O(\Delta x^2)$ | No | No | Not for small timescale |
| FCT implicit | $O(\Delta t)$ - $O(\Delta t^2)$,<br>$O(\Delta x)$ - $O(\Delta x^2)$ | Only for $\theta \geq 0.5$ | Yes | Not for small timescale |

The first three mentioned methods are explicit schemes and are only applicable under the CFL condition which must be fulfilled for stability purposes. The FCT explicit method uses upwind fluxes as low-order flux function and Lax-Wendroff fluxes or central fluxes as high-order flux function. Lax-Wendroff fluxes do not cause any numerical diffusion, whereas central fluxes cause negative numerical diffusion, hence anti-diffusion (see also Section 4.3.4 for the modified equations). Upwind fluxes is the only method which causes numerical diffusion, so diffusiveness of the explicit FCT scheme only occurs if Lax-Wendroff fluxes are used.

The latter three methods are implicit schemes. An important benefit of these methods is that the CFL condition does not have to be fulfilled. In the row of the "FCT implicit" method we see that diffusiveness is not always the case. For $\theta \in [0, 1/2)$ the numerical solution is anti-diffusive, for $\theta = 1/2$ it has zero diffusion and for $\theta \in (1/2, 1]$ the numerical solution is diffusive. The FCT implicit method uses upwind fluxes as low-order flux function and central fluxes as high order flux function.

All the methods presented in the table are robust, where for the explicit cases the CFL condition must hold. For the definition of the accuracy we refer to Section 4.2. The results in Table 4.1 are only valid for the specified problem (4.20)-(4.22). According to the results in the table the FCT implicit scheme is preferred for this 1D problem.

## 4.4 The iterative FCT approach by Kuzmin

In the end of Section 4.3.4 we mentioned that another approach can be used to update the numerical solution to the next time level. This method, which will be presented below, uses an iterative approach and was designed by D. Kuzmin, M. Möller and S. Turek 2004 [6]. This method is not implemented in Delft3D and therefore the main objective of this thesis is to investigate this new approach.

In Section 4.3 we described a simplified approach for updating the numerical solution based on information at the old time level and on a first-order approximate solution at the new time level. In the new approach we will make use of a high-order approximate solution at the new time level. Below we will see that in this case a non-linear equation must be solved in order to obtain the solution $c_i^{n+1}$. To solve the non-linear equation we will use the iterative algorithm by Kuzmin et al. 2004 [6]. The method of Kuzmin takes for each iteration step into account all the anti-diffusive fluxes of the previous iteration steps, whereas normally only the anti-diffusive fluxes of the current iteration step are important. More will be clear after reading the rest of this section.

The method of Kuzmin uses a matrix-vector notation and in this section we keep this formulation. To update the numerical solution to the next time level the following main steps have to be performed for each single time step:

**Implicit FCT method** :
Given the solution $c_i^n$ for all $i$ at time $t^n$ we proceed by

1. Determine the low and high order matrix operator $K^L$ and $K^H$ respectively
2. Determine a low-order approximation $\tilde{c}_i^{n+1}$

   Start of the iteration process, with $c_i^{n+1,0} = c_i^0$

3. Define the anti-diffusive fluxes
4. Apply a pre-limiting step
5. Define the correction factors $\alpha$
6. Update the solution to $c_i^{n+1,m+1}$
7. Repeat steps 3 to 6 if stop-criterion is *not* fulfilled

   End of the iteration process

8. Set $c_i^{n+1} = c_i^{n+1,m+1}$, which is the solution at time step $t^{n+1}$

The method of Kuzmin will be discussed below in more detail for the one-dimensional advection equation with homogeneous boundary conditions. The equation for the homogeneous one-dimensional advection equation reads

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0. \tag{4.46}$$

In order to apply the FCT method we discretize the equation by the theta-method. Using a matrix-vector notation we get

$$\underline{c}^{n+1} - \underline{c}^{n} = \Delta t[(1-\theta)(\underline{F}^{n}_{i+1/2} - \underline{F}^{n}_{i-1/2}) + \theta(\underline{F}^{n+1}_{i+1/2} - \underline{F}^{n+1}_{i-1/2})], \tag{4.47}$$

where $\underline{F}_{i+1/2}$ and $\underline{F}_{i-1/2}$ are vectors representing the fluxes at the right and left boundary of the volume cells respectively. The net fluxes are defined by

$$\begin{aligned}
\underline{F}^{n}_{i+1/2} - \underline{F}^{n}_{i-1/2} &= (K^L + \alpha\cdot(K^H - K^L))\underline{c}^{n}, &(4.48)\\
\underline{F}^{n+1}_{i+1/2} - \underline{F}^{n+1}_{i-1/2} &= (K^L + \alpha\cdot(K^H - K^L))\underline{c}^{n+1}, &(4.49)
\end{aligned}$$

where $\alpha$ is an N x N matrix containing the limiters. $K^L$ (N x N matrix) is the discrete transport operator of the low order method and $K^H$ (N x N matrix) is the discrete transport operator of the high order method. The operators must satisfy the property of zero row sum. An operator has zero row sum if the following holds

$$a_{ii} = -\sum_{j\neq i} a_{ij}, \tag{4.50}$$

with $a_{ij}$ entries of the matrix $A$. For every entry of the term $K\underline{c}^n$ (where $K$ is an arbitrary matrix) we have

$$\begin{aligned}
(K\underline{c}^n)_i = \sum_{j=1} k_{ij}c_j^n &= k_{ii}c_i^n + \sum_{j\neq i} k_{ij}c_j^n \\
&= -\sum_{j\neq i} k_{ij}c_i^n + \sum_{j\neq i} k_{ij}c_j^n \qquad \text{by (4.50)}\\
&= \sum_{j\neq i} k_{ij}(c_j^n - c_i^n).
\end{aligned}$$

With the derivation above we can rewrite Equation (4.47) for all $i$ as

$$\begin{aligned}
c_i^{n+1} - \theta\Delta t\sum_j k^L_{ij}c_j^{n+1} &= c_i^n + (1-\theta)\Delta t\sum_j k^L_{ij}c_j^n \\
&+ \Delta t\sum_{j\neq 1}\{\alpha_{ij}(k^H_{ij} - k^L_{ij})[\theta(c_j^{n+1} - c_i^{n+1}) - (1-\theta)(c_j^n - c_i^n)]\} \qquad (4.51)
\end{aligned}$$

This expression is the numerical scheme for the advection equation and has to be solved for $c_i^{n+1}$. Below we proceed with the solution algorithm by Kuzmin.

1. First we determine the low and high order matrix operator. For $K^L$ we use the upwind method given by

$$K^L = \begin{bmatrix} -\frac{u}{\Delta x} & & & & \frac{u}{\Delta x} \\ \frac{u}{\Delta x} & -\frac{u}{\Delta x} & & & \\ & \ddots & \ddots & & \\ & & \frac{u}{\Delta x} & -\frac{u}{\Delta x} & \\ & & & \frac{u}{\Delta x} & -\frac{u}{\Delta x} \end{bmatrix}.$$

For matrix $K^H$ we choose the central method which is given by

$$K^H = \begin{bmatrix} 0 & -\frac{u}{2\Delta x} & & & \frac{u}{2\Delta x} \\ \frac{u}{2\Delta x} & 0 & -\frac{u}{2\Delta x} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{u}{2\Delta x} & 0 & -\frac{u}{2\Delta x} \\ -\frac{u}{2\Delta x} & & & \frac{u}{2\Delta x} & 0 \end{bmatrix}.$$

We can immediately observe that the zero-row-sum property (4.50) is fulfilled.

2. At the beginning of the time-step we determine the low order approximation $\tilde{c}_i^{n+1}$ by

$$\tilde{c}_i^{n+1} = c_i^n + (1-\theta)\Delta t \sum_j k_{ij}^L c_j^n,$$

where $k_{ij}^L$ are the entries of matrix $K^L$. The solution $\tilde{c}_i^{n+1}$ is an intermediate solution computed at the time instant $t^{n+1-\theta}$ by the explicit low-order scheme. The low-order approximation $\tilde{c}_i^{n+1}$ ensures monotonically behavior.

In steps 3-7 we apply the iteration procedure for the current time-step.

3. Next we define the anti-diffusive fluxes. As we already mentioned before, is that according to Kuzmin also the previous anti-diffusive fluxes are taken into account. Since we will add after each iteration a correction to the intermediate solution we are only interested in the difference between the new anti-diffusive fluxes and the previous ones for every iteration. The difference between the anti-diffusive fluxes $f_{ij}^m$ and the net effect of previous flux corrections $g_{ij}^m$ is given by the formula

$$\Delta f_{ij}^m = f_{ij}^m - g_{ij}^m,$$

with

$$f_{ij}^m = -\Delta t(k_{ij}^L - k_{ij}^H)[\theta(c_j^{n+1,m} - c_i^{n+1,m}) + (1-\theta)(c_j^n - c_i^n)], \quad f_{ji}^m = -f_{ij}^m, \ i < j.$$

and

$$g_{ij}^m = g_{ij}^{m-1} + \alpha_{ij}^{m-1}\Delta f_{ij}^{m-1}, \quad g_{ij}^0 = 0. \tag{4.52}$$

For the first iteration (m=0) we have $c_i^{n+1,0} = \tilde{c}_i^0$ and no previous fluxes are available, hence $g_{ij}^0 = 0$. Note that for $\theta \in (0,1)$ the anti-diffusive fluxes $f_{ij}^m$ depends on both the solution at the start of the iteration $c^n$ and on the solution at the intermediate state $c^{n+1,m}$. For $\theta = 0$ and $\theta = 1$ the solution depends only on $c^n$ and $c^{n+1,m}$ respectively.

4. Before we continue with the computation of the correction factors $\alpha$, we first apply a pre-limiting step which is an important component of the FCT limiter. The purpose is to cancel those anti-diffusive fluxes that directed down the gradient of $c^{n+1,m}$. So we prevent the anti-diffusive flux to be diffusive. The test to be performed is

$$\text{Set } \Delta f_{ij}^m = 0, \quad \text{if} \quad \Delta f_{ij}^m(c_i^{n+1,m} - c_j^{n+1,m}) < 0. \tag{4.53}$$

Without using this step the monotonicity property will not be preserved.

5. In this step the correction factors (limiters) are computed. First we determine the maximum and minimum values of the low order solution $c_i^{n+1,m}$ by

$$c_i^{\max} = \max_{j\in J_i} c_j^{n+1,m}, \tag{4.54}$$

$$c_i^{\min} = \min_{j\in J_i} c_j^{n+1,m}, \tag{4.55}$$

where $J_i$ consists of node i and its nearest neighbors.

The reason for determining (4.54) and (4.55) is to cancel completely those anti-diffusive fluxes which try to accentuate a local maximum or minimum.

Next we define the allowed flux increase/decrease by

$$Q_i^+ = c_i^{\max} - c_i^{n+1,m},$$
$$Q_i^- = c_i^{\min} - c_i^{n+1,m}.$$

Further we define the following quantities according to Zalesak's limiter

$$P_i^{\pm} = \sum_{j\neq i} {}^{\max}_{\min}(0, \Delta f_{ij}^m),$$

and

$$R_i^{\pm} = \begin{cases} \min(1, Q_i^{\pm}/P_i^{\pm}), & \text{if } P_i^{\pm} \neq 0, \\ 0, & \text{if } P_i^{\pm} = 0. \end{cases}$$

The values for $R_i^{\pm}$ must lie in the interval [0,1], since the corrected flux must be a fraction of the amount of flux $P_i^{\pm}$ along the boundaries of cell $V_i$. The values $R_i^{\pm}$ represents the fraction of mass that is allowed to flow into cell $V_i$.

The exchange of mass through the interface of the cells $V_i$ and $V_j$ is the mass fraction that is allowed by both adjacent cells, so therefore the correction factors (flux limiters) are defined by

$$\alpha_{ij}^m = \begin{cases} \min(R_i^+, R_j^-), & \text{if } \Delta f_{ij}^m \geq 0, \\ \min(R_j^+, R_i^-), & \text{if } \Delta f_{ij}^m < 0. \end{cases}$$

Furthermore the symmetry property must hold for the correction factors, i.e. $\alpha_{ij}^m = \alpha_{ji}^m$.

For the correction factors we have that $\alpha_{ij}^m \in [0, 1]$. It is important to note that the computation of the correction factors is in accordance with the cancellation of anti-diffusive fluxes. This means that having $Q_i^{\pm} = 0$ implies $\alpha_{ij}^m = 0$.

6. In the final step we update the solution to $c_i^{n+1,m+1}$ according to

$$c_i^{n+1,m+1} - \theta\Delta t \sum_j k_{ij}^L c_j^{n+1,m+1} = c_i^{n+1,m} + \sum_{j\neq i} \alpha_{ij}^m \Delta f_{ij}^m(c^n, c^{n+1,m}), \tag{4.56}$$

where

$$c_i^{n+1,m} = c_i^{n+1,m-1} + \sum_{j\neq i} \alpha_{ij}^{m-1} \Delta f_{ij}^{m-1}, \quad (m \geq 1) \tag{4.57}$$

and with $c^{n+1,0}$ given. Note that the positivity-preserving solution $c_i^{n+1,m}$ is updated during each iteration. The update formula is a linear equation of the form $Ax = b$ which can be solved directly.

Remark: An alternative is the defect correction method applied by Kuzmin et al. 2004 [6]. Instead of solving equation (4.56) directly the following approach is used:

First the defect vector $\underline{r}^m$ is computed by

$$\underline{r}^m = \underline{b}^m - (I - \theta\Delta t K^L)\underline{c}^{n+1,m},$$

where $\underline{b}^m$ is a vector with the $i$-th entry given by the right-hand side of Equation (4.56). The matrix $K^L$ represents the operator for the low order method and $I$ is the identity matrix.

Next we determine the solution increment $\Delta\underline{c}^m$ by solving

$$(I - \theta\Delta t K^L)\Delta\underline{c}^m = \underline{r}^m.$$

Afterwards, the correction is added to the last iterate to get the updated solution $\underline{c}^{n+1,m+1}$

$$\underline{c}^{n+1,m+1} = \underline{c}^{n+1,m} + \Delta\underline{c}^m.$$

7. Together with an iteration process also a stop-criterion should be given. This criterion defines

when the updated solution $c_i^{n+1,m+1}$ is sufficient. Several stop-criteria can be given, in the sense

$$||\underline{c}^{n+1,m+1} - \underline{c}^{n+1,m}|| \leq \epsilon, \quad \epsilon > 0,$$

where $||\cdot||$ is the L1-norm.

If the stop-criterion is not fulfilled, then steps 3 to 6 are repeated until the criterion is satisfied.

After the iteration is terminated the solution at the next time-step is determined in the final step below.

8. If sufficient (by the stop-criterion) flux-correction is added to the low order approximation $\tilde{c}_i^{n+1}$ (step 2), then we set

$$c_i^{n+1} = c_i^{n+1,m+1},$$

where $c_i^{n+1,m+1}$ is the final update in the iteration process.

In the algorithm described above we increase in each iteration step the anti-diffusion for the intermediate solution $\tilde{c}_i^{n+1}$ (step 2). This means that for each iteration the positivity-preserving solution $\tilde{c}_i^{n+1}$ is more corrected and so more accurate, hence higher accuracy can be obtained for the updated solution $c_i^{n+1}$. In the next chapter we treat this method in more detail.

# Chapter 5

# Iterative FCT methods

In this chapter we describe iterative flux-limiter methods for solving the non-linear equation (4.51). In the previous chapter we used a matrix-vector formulation for the numerical fluxes. From this point we will use the formulation according to Section 4.3.4 which is more transparent and therefore no confusion can be made. Besides the method by Kuzmin also other iterative approaches will be discussed for solving the advection equation. In the next chapter we look at the results after applying the concerning methods.

## 5.1 Iterative FCT method with one limiter

### 5.1.1 FCT method without accumulation

The following scheme for the linear advection equation must be solved for each cell $i$

$$c_i^{n+1} = c_i^n - \frac{\Delta t}{\Delta x_i}(F_{i+1/2} - F_{i-1/2}), \tag{5.1}$$

where $F_{i+1/2}$ and $F_{i-1/2}$ is the numerical flux along the right and left boundary of the i-th cell respectively.

The fluxes are defined by

$$\begin{aligned} F_{i-1/2} = (1-\theta_{i-1/2})[F_L(c_{i-1}^n, c_i^n) + l_{i-1/2}(F_H(c_{i-1}^n, c_i^n) - F_L(c_{i-1}^n, c_i^n))] \\ +\theta_{i-1/2}[F_L(c_{i-1}^{n+1}, c_i^{n+1}) + l_{i-1/2}(F_H(c_{i-1}^{n+1}, c_i^{n+1}) - F_L(c_{i-1}^{n+1}, c_i^{n+1}))], \end{aligned} \tag{5.2}$$

where $F_L$ is the first order upwind method, i.e. $F_L(c_{i-1}^n, c_i^n) = u_{i-1/2}c_{i-1}^n$ for $u_{i-1/2} > 0$, and $F_H$ an high order method. Furthermore $l_{i-1/2} \in [0,1]$, which determine a convex combination between $F_L$ and $F_H$ and further $\theta_{i-1/2} \in [0,1]$.

By inserting the definition of the numerical fluxes into general Equation (5.1) we can rewrite the equation for all $i$ as

$$\begin{aligned}
c_i^{n+1} + \frac{\Delta t}{\Delta x_i}[\theta_{i+1/2}F_L(c_i^{n+1}, c_{i+1}^{n+1}) - \theta_{i-1/2}F_L(c_{i-1}^{n+1}, c_i^{n+1})] = \\
c_i^n - \frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})F_L(c_i^n, c_{i+1}^n) - (1-\theta_{i-1/2})F_L(c_{i-1}^n, c_i^n)] \\
-\frac{\Delta t}{\Delta x_i}[l_{i+1/2}\{(1-\theta_{i+1/2})\Delta F_{i+1/2}^n + \theta_{i+1/2}\Delta F_{i+1/2}^{n+1}\} \\
-l_{i-1/2}\{(1-\theta_{i-1/2})\Delta F_{i-1/2}^n + \theta_{i-1/2}\Delta F_{i-1/2}^{n+1}\}],
\end{aligned} \tag{5.3}$$

where $\Delta F_{i\pm1/2}^n = F_H(c_i^n, c_{i\pm1}^n) - F_L(c_i^n, c_{i\pm1}^n)$.

The limiter $l_{i\pm1/2}$ is according to Zalesak and depends non-linearly on $c_{i\pm1/2}^n$ and $c_{i\pm1/2}^{n+1}$. Its description will be given below. This non-linear dependency makes sure that for each volume cell $i$ the non-linear equation (5.3) should be solved iteratively. In general (by taking all cells $i$) we have an equation of the form

$$A\underline{c}^{n+1} = f(\underline{c}^n, \underline{c}^{n+1}), \tag{5.4}$$

where A represents the square matrix for the left-hand side of the concerning non-linear system.

The iterative process starts with an initial estimate $\underline{c}^{n+1,0}$ and continues by

$$A\underline{c}^{n+1,m+1} = f(\underline{c}^n, \underline{c}^{n+1,m}), \tag{5.5}$$

until the following stop-criterion is fulfilled

$$||\underline{c}^{n+1,m+1} - \underline{c}^{n+1,m}||_1 \leq \epsilon, \quad \epsilon > 0. \tag{5.6}$$

Below we present the first method for solving the non-linear Equation (5.3) according to (5.5).

**Method 1A : One limiter and without accumulated fluxes**

Given the solution $\underline{c}^n$ at time-step $t^n$ we compute the solution $\underline{c}^{n+1}$ at time-step $t^{n+1}$. We perform the following steps with $\theta_{i\pm1/2}$ defined by:

$$\theta_{1\pm1/2} = \max\{0, 1 - \frac{\Delta x_i}{|u|\Delta t}, 1 - \frac{\Delta x_{i\pm1}}{|u|\Delta t}\}. \tag{5.7}$$

---

1. Determine for each cell $i$ the intermediate solution $\tilde{c}_i^{n+1}$ by

$$\tilde{c}_i^{n+1} = c_i^n - \frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})F_L(c_i^n, c_{i+1}^n) - (1-\theta_{i-1/2})F_L(c_{i-1}^n, c_i^n)].$$

2. Set $c_i^{n+1,0} = c_i^n$ for all $i$.

   **While** $||\underline{c}^{n+1,m+1} - \underline{c}^{n+1,m}||_1 > \epsilon, \quad \epsilon > 0,\ m = 0,\ 1,\ 2,\ 3,\ ...$ **do**
   (for $m = 0$ take $||\underline{c}^{n+1,1} - \underline{c}^{n+1,0}||_1 > \epsilon$)

   - Determine for each cell $i$ the flux difference along its right and left cell interface by

     $$\begin{aligned} \Delta f_{i,i+1}^m &= -\frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})\Delta F_{i+1/2}^n + \theta_{i+1/2}\Delta F_{i+1/2}^{n+1,m}], \\ \Delta f_{i,i-1}^m &= \frac{\Delta t}{\Delta x_i}[(1-\theta_{i-1/2})\Delta F_{i-1/2}^n + \theta_{i-1/2}\Delta F_{i-1/2}^{n+1,m}], \end{aligned}$$

     with

     $$\Delta F_{i\pm1/2}^{n+1,m} = F_H(c_i^{n+1,m}, c_{i\pm1}^{n+1,m}) - F_L(c_i^{n+1,m}, c_{i\pm1}^{n+1,m}).$$

     Note that $\Delta f_{i+1,i}^m = -\Delta f_{i,i+1}^m$ for every cell interface.

   - Apply limited anti-diffusive fluxes to the intermediate solution $\tilde{c}_i^{n+1}$ by

     $$b_i^{m+1} = \tilde{c}_i^{n+1} + l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m,$$

     where $l_{i\pm1/2}^m$ (with input $\tilde{c}_i^{n+1}$ and $\Delta f_{i,i\pm1}^m$) are the limiters determined according to the method of Zalesak. The limiter is defined below and has to be applied for each iteration $m$.

   - Solve for all cells $i$ the following linear equation for the next approximation of $c_i^{n+1}$ by

     $$c_i^{n+1,m+1} + \frac{\Delta t}{\Delta x_i}[\theta_{i+1/2}F_L(c_i^{n+1,m+1}, c_{i+1}^{n+1,m+1}) - \theta_{i-1/2}F_L(c_{i-1}^{n+1,m+1}, c_i^{n+1,m+1})] = b_i^{m+1}.$$

3. Set $c_i^{n+1} = c_i^{n+1,m+1}$ for each cell $i$.

---

**The limiter according to Zalesak**

The limiter is included in the iterative solution method and its value differs for each iteration step $m = 0, 1, 2, ...$ . To determine the limiters we need as input in this “limiter formula” the intermediate solution $\underline{\tilde{c}}^{n+1}$ and the flux differences $\Delta f_{i,i\pm1}^m$ for all $i$.

- First we apply a so-called pre-limiting step which is also part of this limiter description. Since the term $\Delta f_{ij}^m$ corrects the diffusive first-order flux it should not be diffusive. Hence we set $\Delta f_{ij}^m = 0$ if $\Delta f_{ij}^m(\tilde{c}_i^{n+1} - \tilde{c}_j^{n+1}) \leq 0$, with $j \in J_i = [i-1, i, i+1]$.

- An important property for the solution is monotonicity, hence no new local maximum or minimum must be created or accentuated. In order to satisfy this condition we first determine an upper and lower bound for the intermediate solution $\tilde{c}_i^{n+1}$

$$c_i^{\max} = \max_{j \in J_i} \tilde{c}_j^{n+1},$$
$$c_i^{\min} = \min_{j \in J_i} \tilde{c}_j^{n+1},$$

  where $J_i$ consists of node $i$ and its nearest neighbors.

- Next we define the amount of flux that flows into cell $V_i$

$$P_i^+ = \max(0, \Delta f_{i,i-1}^m) + \max(0, \Delta f_{i,i+1}^m).$$

  The allowed flux increase is

$$Q_i^+ = c_i^{\max} - \tilde{c}_i^{n+1}.$$

  The fraction of flux that is allowed to flow into the cell is given by

$$R_i^+ = \begin{cases} \min(1, \frac{Q_i^+}{P_i^+}), & P_i^+ > 0, \\ 1, & P_i^+ = 0. \end{cases}$$

  For flux decrease we can define in a similar way the following quantities:

$$\begin{aligned} P_i^- &= \min(0, \Delta f_{i,i-1}^m) + \min(0, \Delta f_{i,i+1}^m), \\ Q_i^- &= c_i^{\min} - \tilde{c}_i^{n+1}, \\ R_i^- &= \begin{cases} \min(1, \frac{Q_i^-}{P_i^-}), & P_i^- < 0, \\ 1, & P_i^- = 0. \end{cases} \end{aligned}$$

  The values $R_i^+$ and $R_i^-$ guarantees no overshoot and undershoot in cell $i$ respectively.

- In the next step we determine the limiter which is the flux fraction that is allowed by both adjacent cells

$$l_{i\pm1/2}^m = \begin{cases} \min(R_i^+, R_{i\pm1}^-), & \Delta f_{i,i\pm1}^m \geq 0 \\ \min(R_{i\pm1}^+, R_i^-), & \Delta f_{i,i\pm1}^m < 0. \end{cases}$$

  Note that the symmetry condition $l_{i,i+1}^m = l_{i+1/2}^m = l_{i+1,i}^m$ is fulfilled. This guarantees that the limiter of the right boundary of cell $i$ ($l^m_{i,i+1}$) is equal to the limiter of the left boundary of cell $i$+1 ($l^m_{i+1,i}$).

Below we will show that the numerical solution by the FCT algorithm presented above is indeed non-negative, i.e. $\underline{c}^{n+1} \geq 0$, and monotone.

**Non-negativity and monotonicity**

Below we will show that for each iteration $m$ the numerical solution is non-negative for method 1A.

The equation to be solved is presented by

$$c_i^{n+1,m+1} + \frac{\Delta t}{\Delta x_i}[\theta_{i+1/2}F_L(c_i^{n+1,m+1}, c_{i+1}^{n+1,m+1}) - \theta_{i-1/2}F_L(c_{i-1}^{n+1,m+1}, c_i^{n+1,m+1})] = \tilde{c}_i^{n+1} + l^m_{i+1/2}\Delta f^m_{i,i+1} + l^m_{i-1/2}\Delta f^m_{i,i-1}. \tag{5.8}$$

The corresponding system to be solved is of the form $A\underline{c}^{n+1,m+1} = \underline{b}^{m+1}$. The equation above is positivity-preserving if the right-hand side $\underline{b}^{m+1} \geq 0$ and matrix A satisfies the following three conditions (Kuzmin et al. 2004 [6], Kuzmin 2008 [5])

1. All diagonal coefficients are positive: $a_{ii} > 0$ for all $i$
2. The M-matrix property holds: the non-diagonal elements are non-positive ($a_{ij} \leq 0, j \neq i$) and the eigenvalues have positive real parts
3. Matrix A is strictly diagonally dominant: $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for all $i$

These are sufficient, but not necessary conditions for non-negativity. First we will show that the right-hand side is non-negative, i.e. $\underline{b}^{m+1} \geq 0$ before we proceed with showing that the conditions for matrix $A$ are fulfilled. Only the results for $u > 0$ will be presented below, though in a similar way one can show non-negativity for $u < 0$.

To show that the right-hand side of Equation (5.8) is positive for all $i$, we first show that the intermediate solution $\tilde{c}_i^{n+1}$ is non-negative for $\underline{c}^n \geq 0$. Remember the expression for the intermediate solution which is given by

$$\tilde{c}_i^{n+1} = c_i^n - \frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})F_L(c_i^n, c_{i+1}^n) - (1-\theta_{i-1/2})F_L(c_{i-1}^n, c_i^n)]. \tag{5.9}$$

If we further take $u > 0$ then the equation can be written as

$$\begin{aligned} \tilde{c}_i^{n+1} &= c_i^n - \frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})uc_i^n - (1-\theta_{i-1/2})uc_{i-1}^n] \\ &= (1 - \frac{|u|\Delta t}{\Delta x_i}(1-\theta_{i+1/2}))c_i^n + \frac{|u|\Delta t}{\Delta x_i}(1-\theta_{i-1/2})c_{i-1}^n. \end{aligned} \tag{5.10}$$

Since the terms $c_i^n$ and $c_{i-1}^n$ are non-negative we have positivity for $\tilde{c}_i^{n+1}$ if the following condition is fulfilled

$$\frac{|u|\Delta t}{\Delta x_i}(1-\theta_{i\pm 1/2}) \leq 1. \tag{5.11}$$

In solution method 1A we have chosen $\theta_{i-1/2} = \max\{0, 1-\frac{\Delta x_i}{|u|\Delta t}, 1-\frac{\Delta x_{i-1}}{|u|\Delta t}\}$, so condition (5.11) is fulfilled. Further, condition (5.11) tells us that the intermediate solution $\tilde{c}^{n+1}$ is obtained by shifting the entire solution $c^n$ at most one cell to the right.

Next we continue with the addition of the corrected anti-diffusion terms to the term $\tilde{c}_i^{n+1}$ and show that the RHS of Equation (5.8) is still positive. Therefore we need to rewrite the RHS of Equation (5.8) in the following form

$$(1-v_i)\tilde{c}_i^{n+1} + v_i\tilde{c}_k^{n+1}, \tag{5.12}$$

with $k$ the number of a neighboring node at which a local extreme is attained and where $v_i$ should be such that $0 \leq v_i \leq 1$ in order to have positivity for the expression above.

Now we choose $v_i$ as follows

$$v_i = \begin{cases} \frac{l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m}{\tilde{c}_k^{n+1} - \tilde{c}_i^{n+1}}, & \text{if} \quad l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m > 0, \\ \frac{l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m}{\tilde{c}_k^{n+1} - \tilde{c}_i^{n+1}}, & \text{if} \quad l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m < 0, \\ 0, & \text{if} \quad l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m = 0. \end{cases}$$

It requires now three steps to show that the definition for $v_i$ is correct. First we rewrite Equation (5.12) as $\tilde{c}_i^{n+1} + (\tilde{c}_k^{n+1} - \tilde{c}_i^{n+1})v_i$ and by substitution of $v_i$ for each different case it can be easily checked that the right-hand side of Equation (5.8) is obtained. Furthermore, we have $v_i \geq 0$ for each of these cases.

The third step is to show that $v_i \leq 1$. Therefore we need the definitions given in the "Limiter by Zalesak" in this section. For $l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m > 0$ this is as follows

$$l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m \leq l_{i+1/2}^m \max\{0, \Delta f_{i,i+1}^m\} + l_{i-1/2}^m \max\{0, \Delta f_{i,i-1}^m\}. \tag{5.13}$$

Since $\max\{0, \Delta f_{i,i\pm 1}^m\} \geq 0$ the limiter is given by $l_{i\pm 1/2}^m = \min\{R_i^+, R_{i\pm 1}^-\}$, hence

$$\begin{aligned} l_{i+1/2}^m \max\{0, \Delta f_{i,i+1}^m\} + l_{i-1/2}^m \max\{0, \Delta f_{i,i-1}^m\} &\leq R_i^+ \max\{0, \Delta f_{i,i+1}^m\} + R_i^+ \max\{0, \Delta f_{i,i-1}^m\} \\ &= R_i^+ P_i^+. \end{aligned} \tag{5.14}$$

For $P_i^+ > 0$ we have $\frac{Q_i^+}{P_i^+} \geq \min\{1, \frac{Q_i^+}{P_i^+}\} = R_i^+$. So for $P_i^+ \geq 0$ we have

$$R_i^+ P_i^+ \leq Q_i^+, \tag{5.15}$$

So finally we have from (5.13), (5.14) and (5.15)

$$l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m \leq Q_i^+ = c_i^{\max} - \tilde{c}_i^{n+1}.$$

After dividing both sides by $Q_i^+$ with $Q_i^+ > 0$ we get

$$v_i \leq 1.$$

So the inequality $v_i \leq 1$ is verified for $l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m > 0$. This inequality can also be shown in a similar way for $l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m < 0$. For $l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m = 0$ this is obvious, since $v_i = 0$.

Now it remains to show that the left-hand side of Equation (5.8) satisfies the three conditions. Since $F_L$ is the first-order upwind method the left-hand side of Equation (5.8) can be written for $u > 0$ as

$$(1 + \frac{|u|\Delta t}{\Delta x_i} \theta_{i+1/2}) c_i^{n+1,m+1} - \frac{|u|\Delta t}{\Delta x_i} \theta_{i-1/2} c_{i-1}^{n+1,m+1}. \tag{5.16}$$

The coefficients in the expression above can for all $i \in [1, ..., N]$ be given in matrix form by

$$A = \begin{bmatrix} 1 + \frac{|u|\Delta t}{\Delta x_1}\theta_{1+1/2} & & & & -\frac{|u|\Delta t}{\Delta x_1}\theta_{1-1/2} \\ -\frac{|u|\Delta t}{\Delta x_2}\theta_{2-1/2} & 1 + \frac{|u|\Delta t}{\Delta x_2}\theta_{2+1/2} & & & \\ & \ddots & \ddots & & \\ & & -\frac{|u|\Delta t}{\Delta x_{N-1}}\theta_{(N-1)-1/2} & 1 + \frac{|u|\Delta t}{\Delta x_{N-1}}\theta_{(N-1)+1/2} & \\ & & & -\frac{|u|\Delta t}{\Delta x_N}\theta_{N-1/2} & 1 + \frac{|u|\Delta t}{\Delta x_N}\theta_{N+1/2} \end{bmatrix}.$$

From the matrix we immediately notice that condition 1 is fulfilled. The same holds for the first part of condition 2. It remains to show for the second part of condition 2 that the eigenvalues of matrix $A$ have positive real part. Therefore we use the Theorem of Gershgorin which gives precisely in which discs of the complex plane the eigenvalues can be found (Vuik et al. 2006, [12]).

**Theorem of Gershgorin**
The eigenvalues $\lambda$ of a matrix $M$ are in the union of circles if

$$|z - m_{ii}| \leq \sum_{j=1, j \neq i}^{N} |m_{ij}|, \quad \text{with } z \in \mathbb{C}.$$

If we apply the Theorem to the matrix given above we get for all $i$ the following result

$$|\lambda_i - (1 + \frac{|u|\Delta t}{\Delta x_i}\theta_{i+1/2})| \leq \frac{|u|\Delta t}{\Delta x_i}\theta_{i-1/2}. \tag{5.17}$$

Hence, each eigenvalue $\lambda_i$ is in the circle with center $(1 + \frac{|u|\Delta t}{\Delta x_i}\theta_{i+1/2}, 0)$ and radius $\frac{|u|\Delta t}{\Delta x_i}\theta_{i-1/2}$. This means for the real part that

$$Re(\lambda_i) \in [1 + \frac{|u|\Delta t}{\Delta x_i}\theta_{i+1/2} - \frac{|u|\Delta t}{\Delta x_i}\theta_{i-1/2}, 1 + \frac{|u|\Delta t}{\Delta x_i}\theta_{i+1/2} + \frac{|u|\Delta t}{\Delta x_i}\theta_{i-1/2}]. \tag{5.18}$$

For a uniform grid we have that $\theta_{i-1/2} = \theta_{i+1/2}$, so $Re(\lambda_i) \in [1, 1 + 2\frac{|u|\Delta t}{\Delta x_i}\theta_{i+1/2}] \subset [1, \infty)$. Hence, for a uniform grid Condition 2 is automatically satisfied. For a non-uniform grid this condition is not obvious and therefore will be explained below. From above we know that

$$\min(\mathrm{Re}(\lambda_i)) = 1 + \frac{|u|\Delta t}{\Delta x_i}\theta_{i+1/2} - \frac{|u|\Delta t}{\Delta x_i}\theta_{i-1/2}. \tag{5.19}$$

Further, we can derive from the positivity condition (5.11) that

$$\frac{|u|\Delta t}{\Delta x_i}\theta_{i+1/2} \geq \frac{|u|\Delta t}{\Delta x_i} - 1. \tag{5.20}$$

Combining Equation (5.19) and Equation (5.20) we get

$$\begin{aligned}
\min(Re(\lambda_i)) &= 1 + \frac{|u|\Delta t}{\Delta x_i}\theta_{i+1/2} - \frac{|u|\Delta t}{\Delta x_i}\theta_{i-1/2} \\
&\geq 1 + \frac{|u|\Delta t}{\Delta x_i} - 1 - \frac{|u|\Delta t}{\Delta x_i}\theta_{i-1/2} \\
&= \frac{|u|\Delta t}{\Delta x_i} - \frac{|u|\Delta t}{\Delta x_i}\theta_{i-1/2} \\
&> 0. 
\end{aligned} \tag{5.21}$$

The last inequality is true since we haven chosen in the solution algorithm (see also method 1A) $\theta_{i-1/2} = \max\{0, 1 - \frac{\Delta x_i}{|u|\Delta t}, 1 - \frac{\Delta x_{i-1}}{|u|\Delta t}\} < 1$. So Condition 2 is also fulfilled for a non-uniform grid.

According to Condition 3 we must have strictly diagonally dominance for matrix A. This is automatically satisfied for a uniform grid, since $\theta_{i-1/2} = \theta_{i+1/2}$. For a non-uniform case the third condition must be checked. For each row $i$ of matrix A we have after summation of the corresponding elements

$$1 + \frac{|u|\Delta t}{\Delta x_i}\theta_{i+1/2} - \frac{|u|\Delta t}{\Delta x_i}\theta_{i-1/2}. \tag{5.22}$$

From Equation (5.21) we see that this expression is positive, hence Condition 3 is also fulfilled.

We have shown above that the right-hand side of Equation (5.8) is non-negative. Together with the fact that matrix A satisfies the three conditions we have for every iteration $m$ positivity for $\underline{c}^{n+1,m+1}$, i.e. $\underline{c}^{n+1,m+1} \geq 0$.

Starting with an initial condition (IC) $c(x,0) = \underline{c}^0 \geq 0$ we are guaranteed by the analysis above that a positive numerical solution is obtained for the numerical solution at the next time step, i.e. $c^1 \geq 0$. Continuing the limiter procedure for every time step we obviously get a positive solution $\underline{c}^n$ for all time steps.

Given that $c_i^n$ is monotone we want this property also holds for the numerical solution at the next time step. Remember that the intermediate solution $\underline{\tilde{c}}^{n+1}$ is computed by first-order upwind, so we automatically have monotonicity for $\underline{\tilde{c}}^{n+1}$. Monotonicity must also hold after addition of the anti-diffusive fluxes. Above we showed that positivity is guaranteed for the right-hand side after correcting the intermediate solution. Hence, the right-hand side must be monotone. Since matrix $A$ in the LHS of (5.8) is the first-order implicit part we have positivity for $c_i^{n+1}$, hence $c_i^{n+1}$ is monotone.

Since we start with a monotonously initial condition $c(x,0)$, we have for every iteration $m$ at each time step a monotonous solution by the FCT method.

### Accuracy at extrema

For solutions with discontinuities the FCT method is a good approximation method. The method will be first-order accurate at the discontinuity and high-order accurate elsewhere. For smooth functions we expect the FCT method to be high-order accurate everywhere, but this is not the case. At extremal points the accuracy is decreased by the flux limiter. This behavior for smooth functions will be discussed below.

For this explanation we return to the description of the limiter by Zalesak, which is presented in the section above. To determine the limiter value one starts with the computation of the maximum and minimum of the intermediate solution $\tilde{c}_i^{n+1}$. For points other then extremal points these maxima and minima are different from the value in the corresponding cell, that is

$$\begin{aligned} c_i^{\max} &\neq \tilde{c}_i^{n+1}, \\ c_i^{\min} &\neq \tilde{c}_i^{n+1}. \end{aligned}$$

Though, for extremal points at least one of $c_i^{\max}$ or $c_i^{\min}$ is equal to value $\tilde{c}_i^{n+1}$ in the cell. This automatically corresponds with $Q_i^+ = 0$ for maximum points and $Q_i^- = 0$ for minimum points. If next to it the amount of flux $P_i^\pm$ that flows into or out of cell $i$ is equal to zero (this happens only at constant parts in the the intermediate solution $\tilde{c}_i^{n+1}$) then by the definition of $R_i^\pm$ this quantity is equal to 1 and then the limiter $l_{i+1/2}^m$ is according to its definition determined by the value $R_{i+1}^\pm$ for cell $i$+1. If however the amount of flux $P_i^\pm$ is not equal to zero then the value $R_i^\pm$ is equal to zero caused by $Q_i^\pm$ which implies a limiter value zero. In summary:

$$Q_i^\pm = 0, P_i^\pm \neq 0 \quad => \quad R_i^\pm = 0 \quad => \quad l_{i+1/2}^m = 0.$$

A limiter value of zero corresponds with a low-order method (in our case first order). This proves that the FCT solution cannot be high-order accurate for smooth solutions at extremal points, in other words the accuracy is locally decreased from $O(h^2)$ to $O(h)$. Though, for the numerical solution at the end it is not clear yet if this problem has a large effect on the global accuracy, since also other aspects are responsible for not being second-order accurate, which will be shown in the next chapter. On the other hand, for solutions with discontinuities this effect (having limiter value 0) is desired in order to remove the wiggles.

### 5.1.2 FCT method with accumulation

We want to obtain more improvement in the numerical solution by the FCT method. An approach is by introducing accumulation of fluxes into the FCT method. Below we present a slightly different algorithm for solving general Equation (5.3). In the previous section no anti-diffusive fluxes of the previous iterations are taken in the computation, whereas in this case the anti-diffusive fluxes of the previous iterations are indeed taken into account. This latter method is according to the approach of Kuzmin et al. 2004 [6]. See also the previous chapter.

In this case we have the following iterative procedure

$$\underline{c}^{n+1,m+1} = \underline{c}^{n+1,m} + \Delta\underline{c}^m. \tag{5.23}$$

This is a cumulative process with $\Delta\underline{c}^m$ a correction applied on the previous solution $\underline{c}^{n+1,m}$. Below we first present the numerical method with two limiters and shortly after we will go into more detail of this approach.

**Method 1B : One limiter and with accumulated fluxes**

Given the solution $\underline{c}^n$ at time-step $t^n$ we compute the solution $\underline{c}^{n+1}$ at time-step $t^{n+1}$. We perform the following steps with $\theta_{i\pm1/2}$ defined in (5.7):

1. Determine for each cell $i$ the intermediate solution $\tilde{c}_i^{n+1}$ by

$$\tilde{c}_i^{n+1} = c_i^n - \frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})F_L(c_i^n, c_{i+1}^n) - (1-\theta_{i-1/2})F_L(c_{i-1}^n, c_i^n)].$$

2. For all $i$ set $c_i^{n+1,0} = c_i^n$, $g_{i,i\pm1}^0 = 0$ and $b_i^0 = \tilde{c}_i^{n+1}$.

   **While** $||\underline{c}^{n+1,m+1} - \underline{c}^{n+1,m}||_1 > \epsilon, \quad \epsilon > 0,\ m = 0,\ 1,\ 2,\ 3,\ ...$ **do**
   (for $m = 0$ take $||\underline{c}^{n+1,1} - \underline{c}^{n+1,0}||_1 > \epsilon$)

   - Determine for each cell $i$ the flux difference along its right and left cell interface by

$$\begin{aligned}\Delta f_{i,i+1}^m &= -\frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})\Delta F_{i+1/2}^n + \theta_{i+1/2}\Delta F_{i+1/2}^{n+1,m}] - g_{i,i+1}^m,\\ \Delta f_{i,i-1}^m &= \frac{\Delta t}{\Delta x_i}[(1-\theta_{i-1/2})\Delta F_{i-1/2}^n + \theta_{i-1/2}\Delta F_{i-1/2}^{n+1,m}] - g_{i,i-1}^m,\end{aligned}$$

with

$$\Delta F_{i\pm1/2}^{n+1,m} = F_H(c_i^{n+1,m}, c_{i\pm1}^{n+1,m}) - F_L(c_i^{n+1,m}, c_{i\pm1}^{n+1,m}).$$

The terms $g_{i,i\pm1}^m$ represents the anti-diffusive fluxes of the previous iterations $m-1,...,0$.

- Apply limited anti-diffusive fluxes to the intermediate solution $b_i^m$ by

$$b_i^{m+1} = b_i^m + l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m,$$

where $l_{i\pm1/2}^m$ (with input $b_i^m$ and $\Delta f_{i,i\pm1}^m$) are the limiters determined according to the method of Zalesak given in Section 5.1.1. Replace in the formula for the limiter the term $\tilde{c}_i^{n+1}$ by $b_i^m$.

- Solve for all cells $i$ the following linear equation for the next approximation of $c_i^{n+1}$ by

$$c_i^{n+1,m+1} + \frac{\Delta t}{\Delta x_i}[\theta_{i+1/2}F_L(c_i^{n+1,m+1}, c_{i+1}^{n+1,m+1}) - \theta_{i-1/2}F_L(c_{i-1}^{n+1,m+1}, c_i^{n+1,m+1})] = b_i^{m+1}.$$

- Update $g_{i,i\pm1}^m$ for each cell $i$ by

$$g_{i,i\pm1}^{m+1} = g_{i,i\pm1}^m + l_{i\pm1/2}^m \Delta f_{i,i\pm1}^m.$$

3. Set $c_i^{n+1} = c_i^{n+1,m+1}$ for each cell $i$.

---

Next we will show that the algorithm above corresponds with Equation (5.23), i.e. a cumulative procedure.

For the $m$-th iteration we have the following equation to be solved (third part of step 2)

$$\begin{aligned}A\underline{c}^{n+1,m+1} &= \underline{b}^{m+1}\\ &= \underline{b}^m + \underline{l}^m \cdot \Delta \underline{f}^m.\end{aligned}$$

where A represents the matrix for the left-hand side of the corresponding non-linear system and $\cdot$ is the inner product.

Multiplying both left- and right-hand side with $A^{-1}$ we get

$$\underline{c}^{n+1,m+1} = A^{-1}\underline{b}^m + A^{-1}\underline{l}^m \cdot \Delta\underline{f}^m.$$

Since also $A\underline{c}^{n+1,m} = b^m$ the last expression can be written as

$$\underline{c}^{n+1,m+1} = \underline{c}^{n+1,m} + A^{-1}\underline{l}^m \cdot \Delta\underline{f}^m, \tag{5.24}$$

which shows that the iteration procedure is indeed cumulative.

In method 1B we correct the intermediate solution $\tilde{c}^{n+1}$ by adding a flux correction in each iteration step. The corrections are accumulated, i.e. the corrections of previous iterations are included in $\tilde{c}^{n+1}$. At a certain moment sufficient correction is added due to the stop-criterion. From the stop-criterion and Equation (5.24) we derive the following

$$\begin{aligned} ||\underline{c}^{n+1,m+1} - \underline{c}^{n+1,m}|| &= ||A^{-1}\underline{l}^m \cdot \Delta\underline{f}^m|| \\ &\leq ||A^{-1}|| \cdot ||\underline{l}^m \Delta\underline{f}^m|| \\ &\leq \epsilon. \end{aligned} \tag{5.25}$$

This implies that

$$||\underline{l}^m \Delta\underline{f}^m|| \leq C^{-1}\epsilon, \tag{5.26}$$

where $C$ is some constant. This equation tells us that the iteration is stopped when the limited flux correction is sufficiently small in the norm. This means that the limited corrections $l^m_{i\pm1/2}\Delta f^m_{i,i\pm1}$ must decrease as $m$ get larger in order to fulfill the equation above.

Furthermore we can also show that if $l^m_{i\pm1/2} = 1$ for all $i$ and all $m$, then the equation to solve for method 1A and method 1B are equal. For method 1B we have

$$\begin{aligned} b_i^{m+1} &= b_i^m + \Delta f^m_{i,i+1} + \Delta f^m_{i,i-1}, \\ &= b_i^n + \sum_{k=0}^{m} \Delta f^k_{i,i+1} + \sum_{k=0}^{m} \Delta f^k_{i,i-1} \\ &= b_i^n + g^m_{i,i+1} + \Delta f^m_{i,i+1} + g^m_{i,i-1} + \Delta f^m_{i,i-1} \\ &= b_i^n + f^m_{i,i+1} + f^m_{i,i-1}, \end{aligned} \tag{5.27}$$

with $b_i^n = \tilde{c}_i^{n+1}$.

The third and final equality follow respectively from the definition of $g_{i,i\pm1}^{m+1}$ and the definition of the flux difference $\Delta f_{i,i\pm1}^{m}$ given in method 1B above. One can easily see that the final equation holds for method 1A, since $l_{i\pm1/2}^{m} = 1$ for all $i$ and $m$.

Method 1B differs mainly from method 1A in the formulation of the fluxes $\Delta f_{i,i\pm1}^{m}$, so non-negativity and monotonicity can be shown in a similar way as with method 1A. So method 1B is a valid method, in the sense that no negative values and/or wiggles can occur in the numerical solution.

## 5.2 Iterative FCT method with two limiters

In the previous section we presented the numerical FCT scheme for the advection equation (Equation (5.3)) and discussed two methods for solving the corresponding system of equations. The limiter for those methods depends both on the numerical solution at the old and new time-level. This has as result that the fluxes at the old and new time-level are limited to the same order of magnitude. For one or both of these numerical fluxes this limiting can be less accurate, since the limiter has to take into account the information of both levels. One might question if the numerical solution can be improved by using two different limiters with one for the old time-level and the other for the new time-level. With this approach one might expect the numerical fluxes to be limited properly. We will discuss this new approach to obtain the numerical solution.

### 5.2.1 FCT method without accumulation

Before we go into more detail we first give the general numerical scheme for the homogeneous advection equation, where now we have a splitting of the fluxes between the old and new time-level.

$$c_i^{n+1} = c_i^n - \frac{\Delta t}{\Delta x_i}[((1-\theta_{i+1/2})F_{i+1/2}^n - (1-\theta_{i-1/2})F_{i-1/2}^n) + (\theta_{i+1/2}F_{i+1/2}^{n+1} - \theta_{i-1/2}F_{i-1/2}^{n+1})], \quad (5.28)$$

with the numerical fluxes defined in the following form

$$F_{i\pm 1/2}^n = F_L^n + l_{i\pm 1/2}^n(F_H^n - F_L^n), \quad (5.29)$$

$$F_{i\pm 1/2}^{n+1} = F_L^{n+1} + l_{i\pm 1/2}^{n+1}(F_H^{n+1} - F_L^{n+1}), \quad (5.30)$$

where $F_L$ is the first-order upwind method and $F_H$ an higher order method, e.g. central spatial-discretization. Notice that we are dealing with a limiter at the old level ($l_{i\pm 1/2}^n$) and a limiter at the new level ($l_{i\pm 1/2}^{n+1}$), which are not necessarily equal. The values are still contained in [0,1]. Furthermore we have $\theta_{i\pm 1/2} \in [0, 1]$. Note that in the previous section we discussed a special case of the numerical scheme above where $l_{i\pm 1/2}^n = l_{i\pm 1/2}^{n+1}$.

By inserting the definition of the numerical fluxes into general Equation (5.28) we can rewrite the equation for all $i$ as

$$\begin{aligned} c_i^{n+1} + \frac{\Delta t}{\Delta x_i}[\theta_{i+1/2}F_L(c_i^{n+1}, c_{i+1}^{n+1}) - \theta_{i-1/2}F_L(c_{i-1}^{n+1}, c_i^{n+1})] = \\ c_i^n - \frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})\{F_L(c_i^n, c_{i+1}^n) + l_{i+1/2}^n \Delta F_{i+1/2}^n\} - \\ (1-\theta_{i-1/2})\{F_L(c_{i-1}^n, c_i^n) + l_{i-1/2}^n \Delta F_{i-1/2}^n\}] \\ -\frac{\Delta t}{\Delta x_i}[\theta_{i+1/2} l_{i+1/2}^{n+1} \Delta F_{i+1/2}^{n+1} - \theta_{i-1/2} l_{i-1/2}^{n+1} \Delta F_{i-1/2}^{n+1}\}], \end{aligned} \quad (5.31)$$

where $\Delta F_{i\pm 1/2}^n = F_H(c_i^n, c_{i\pm 1}^n) - F_L(c_i^n, c_{i\pm 1}^n)$.

The limiter $l_{i\pm1/2}^{n+1}$ is according to Zalesak and depends non-linear on $c_{i\pm1/2}^{n}$ and $c_{i\pm1/2}^{n+1}$. For each volume cell $i$ the non-linear equation above should be solved iteratively. Though, the limiter $l_{ij}^{n}$ depends solely on $c_{i\pm1/2}^{n}$. In general (by taking all cells $i$) we have an equation of the form

$$A\underline{c}^{n+1} = g(\underline{c}^{n}, \underline{c}^{n+1}). \tag{5.32}$$

The iterative process starts with an initial estimate $\underline{c}^{n+1,0}$ and continue by

$$A\underline{c}^{n+1,m+1} = g(\underline{c}^{n}, \underline{c}^{n+1,m}), \tag{5.33}$$

where A represents the square matrix for the left-hand side of the concerning non-linear system.

Together with an iteration process also a stop-criterion should be given. This criterion defines when the updated solution $c_i^{n+1,m+1}$ is sufficient. This condition is defined by

$$||\underline{c}^{n+1,m+1} - \underline{c}^{n+1,m}||_1 \leq \epsilon, \quad \epsilon > 0.$$

Below we present the first method for solving Equation (5.31) according to the iterative process (5.33).

**Method 2A : two limiters and without accumulated fluxes**

Given the solution $\underline{c}^{n}$ at time-step $t^n$ we compute the solution $\underline{c}^{n+1}$ at time-step $t^{n+1}$. We perform the following steps with $\theta_{i\pm1/2}$ defined in (5.7):

---

1. Determine the intermediate solution $\tilde{c}_i^{n+1}$ for each cell $i$ by

$$\begin{aligned}\tilde{c}_i^{n+1} = c_i^n - \frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})\{F_L(c_i^n, c_{i+1}^n) + l_{i+1/2}^n(F_H(c_i^n, c_{i+1}^n) - F_L(c_i^n, c_{i+1}^n))\}\\ -(1-\theta_{i-1/2})\{F_L(c_{i-1}^n, c_i^n) + l_{i-1/2}^n(F_H(c_{i-1}^n, c_i^n) - F_L(c_{i-1}^n, c_i^n))\}],\end{aligned}$$

   where $l_{i\pm1/2}^n$ are the limiters determined according to the method of Zalesak given in Section 5.1.1. Further, we must replace in the formula for the limiter the term $\tilde{c}_i^{n+1}$ by $c_i^n$.

2. Set $c_i^{n+1,0} = c_i^n$ for all $i$.

   **While** $||\underline{c}^{n+1,m+1} - \underline{c}^{n+1,m}||_1 > \epsilon, \quad \epsilon > 0, \ m = 0, 1, 2, 3, ...$ **do**
   (for $m = 0$ take $||\underline{c}^{n+1,1} - \underline{c}^{n+1,0}||_1 > \epsilon$)

   - Determine for each cell $i$ the flux difference along its right and left cell interface by

$$\Delta f^m_{i,i+1} = -\frac{\Delta t}{\Delta x_i}\theta_{i+1/2}\Delta F^{n+1,m}_{i+1/2},$$
$$\Delta f^m_{i,i-1} = \frac{\Delta t}{\Delta x_i}\theta_{i-1/2}\Delta F^{n+1,m}_{i-1/2},$$

with

$$\Delta F^{n+1,m}_{i\pm1/2} = F_H(c_i^{n+1,m}, c_{i\pm1}^{n+1,m}) - F_L(c_i^{n+1,m}, c_{i\pm1}^{n+1,m}).$$

- Apply limited anti-diffusive fluxes to the intermediate solution $\tilde{c}_i^{n+1}$ by

$$b_i^{m+1} = \tilde{c}_i^{n+1} + l^m_{i+1/2}\Delta f^m_{i,i+1} + l^m_{i-1/2}\Delta f^m_{i,i-1},$$

  where $l^m_{i\pm1/2}$ (with input $\tilde{c}_i^{n+1}$ and $\Delta f^m_{i,i\pm1}$) are the limiters determined according to the method of Zalesak defined in Section 5.1.1.

- Solve for all cells $i$ the linear equation for the next approximation of $c_i^{n+1}$ by

$$c_i^{n+1,m+1} + \frac{\Delta t}{\Delta x_i}[\theta_{i+1/2}F_L(c_i^{n+1,m+1}, c_{i+1}^{n+1,m+1}) - \theta_{i-1/2}F_L(c_{i-1}^{n+1,m+1}, c_i^{n+1,m+1})] = b_i^{m+1}.$$

3. Set $c_i^{n+1} = c_i^{n+1,m+1}$ for each cell $i$.

---

### 5.2.2 FCT method with accumulation

Below we present a slightly different method for solving Equation (5.31). Above no anti-diffusive fluxes of the previous iterations are taken in the computation, whereas in this case the anti-diffusive fluxes of the previous iterations are indeed taken into account. This latter method is an adapted version of the approach of Kuzmin et al. 2004 [6].

In this case we have the following iterative procedure

$$\underline{c}^{n+1,m+1} = \underline{c}^{n+1,m} + \Delta\underline{c}^m. \tag{5.34}$$

This is a cumulative process with $\Delta\underline{c}^m$ a correction applied on the previous solution $\underline{c}^{n+1,m}$. Similar as before (Method 1B) one can show that the method below is indeed of the form (5.34).

**Method 2B : two limiters and with accumulated fluxes**

Given the solution $\underline{c}^n$ at time-step $t^n$ we compute the solution $\underline{c}^{n+1}$ at time-step $t^{n+1}$. We perform the following steps with $\theta_{i\pm1/2}$ defined in (5.7):

---

1. Determine the intermediate solution $\tilde{c}_i^{n+1}$ for each cell $i$ by

$$\begin{aligned}\tilde{c}_i^{n+1} = c_i^n - \frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})\{F_L(c_i^n, c_{i+1}^n) + l_{i+1/2}^n(F_H(c_i^n, c_{i+1}^n) - F_L(c_i^n, c_{i+1}^n))\}\\ -(1-\theta_{i-1/2})\{F_L(c_{i-1}^n, c_i^n) + l_{i-1/2}^n(F_H(c_{i-1}^n, c_i^n) - F_L(c_{i-1}^n, c_i^n))\}],\end{aligned}$$

where $l_{i\pm1/2}^n$ are the limiters determined according to the method of Zalesak given in Section 5.1.1. Further, we must replace in the formula for the limiter the term $\tilde{c}_i^{n+1}$ by $c_i^n - \frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})F_L(c_i^n, c_{i+1}^n) - (1-\theta_{i-1/2})F_L(c_{i-1}^n, c_i^n)]$.

2. For all $i$ set $c_i^{n+1,0} = c_i^n$, $g_{i,i\pm1}^0 = 0$ and $b_i^0 = \tilde{c}_i^{n+1}$.

**While** $||\underline{c}^{n+1,m+1} - \underline{c}^{n+1,m}||_1 > \epsilon, \quad \epsilon > 0,\ m = 0,\ 1,\ 2,\ 3,\ ...$ **do**
(for $m = 0$ take $||\underline{c}^{n+1,1} - \underline{c}^{n+1,0}||_1 > \epsilon$)

- Determine for each cell $i$ the flux difference along its right and left cell interface by

$$\begin{aligned}\Delta f_{i,i+1}^m &= -\frac{\Delta t}{\Delta x_i}\theta_{i+1/2}\Delta F_{i+1/2}^{n+1,m} - g_{i,i+1}^m,\\ \Delta f_{i,i-1}^m &= \frac{\Delta t}{\Delta x_i}\theta_{i-1/2}\Delta F_{i-1/2}^{n+1,m} - g_{i,i-1}^m,\end{aligned}$$

with

$$\Delta F_{i\pm1/2}^{n+1,m} = F_H(c_i^{n+1,m}, c_{i\pm1}^{n+1,m}) - F_L(c_i^{n+1,m}, c_{i\pm1}^{n+1,m}).$$

The term $g_{i,i\pm1}^m$ represents the anti-diffusive fluxes of the previous iterations $m-1, ..., 0$.

- Apply limited anti-diffusive fluxes to the intermediate solution $b_i^m$ by

$$b_i^{m+1} = b_i^m + l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m,$$

where $l_{i\pm1/2}^m$ (with input $b_i^m$ and $\Delta f_{i,i\pm1}^m$) are the limiters determined according to the method of Zalesak defined in Section 5.1.1. Replace in the formula for the limiter the term $\tilde{c}_i^{n+1}$ by $b_i^m$.

- Solve for all cells $i$ the linear equation for the next approximation of $c_i^{n+1}$ by

$$c_i^{n+1,m+1} + \frac{\Delta t}{\Delta x_i}[\theta_{i+1/2}F_L(c_i^{n+1,m+1}, c_{i+1}^{n+1,m+1}) - \theta_{i-1/2}F_L(c_{i-1}^{n+1,m+1}, c_i^{n+1,m+1})] = b_i^{m+1}.$$

- Update $g_{i,i\pm1}^m$ for each cell $i$ by

$$g_{i,i\pm1}^{m+1} = g_{i,i\pm1}^m + l_{i\pm1/2}^m \Delta f_{i,i\pm1}^m.$$

3. Set $c_i^{n+1} = c_i^{n+1,m+1}$ for each cell $i$.

---

In a similar way as in Section 5.1.1 we can explain for method 2A and 2B that for each iteration $m$ the numerical solution remains non-negative and monotone by the limiter. Though, the main difference is that we deal in this case with two limiters. Remember the equation for method 2 to be solved is given by

$$\begin{aligned} c_i^{n+1,m+1} + \frac{\Delta t}{\Delta x_i}[\theta_{i+1/2}F_L(c_i^{n+1,m+1}, c_{i+1}^{n+1,m+1}) - \theta_{i-1/2}F_L(c_{i-1}^{n+1,m+1}, c_i^{n+1,m+1})] = \\ c_i^n - \frac{\Delta t}{\Delta x_i}[(1-\theta_{i+1/2})F_L(c_i^n, c_{i+1}^n) - (1-\theta_{i-1/2})F_L(c_{i-1}^n, c_i^n)] + \\ l_{i+1/2}^n \Delta f_{i,i+1}^n + l_{i-1/2}^n \Delta f_{i,i-1}^n + l_{i+1/2}^m \Delta f_{i,i+1}^m + l_{i-1/2}^m \Delta f_{i,i-1}^m, \end{aligned} \tag{5.35}$$

where $F_L$ is the first-order upwind operator, $\Delta f_{i,i\pm1}^m$ is the flux difference defined in method 2A/2B and $\Delta f_{i,i\pm1}^n$ is given by

$$\Delta f_{i,i\pm1}^n = \mp\frac{\Delta t}{\Delta x_i}(1-\theta_{i\pm1/2})(F_H(c_i^n, c_{i+1}^n) - F_L(c_{i-1}^n, c_i^n)), \tag{5.36}$$

with $F_H$ the high-order operator. If $l_{i\pm1/2}^n = l_{i\pm1/2}^m$ we obtain the one-limiter approach.

Equation (5.35) is of the form $A\underline{c}^{n+1,m+1} = \underline{b}^{m+1}$ and since matrix $A$ is equal as in method 1A and 1B we have that this matrix satisfies the three conditions for non-negativity (see Section 5.1.1). Next it remains to show that the right-hand side is non-negative, i.e. $\underline{b}^{m+1} \geq 0$.

In Section 5.1.1 we showed that the sum of the first two terms in the right-hand side of (5.35) is positive, since theta is defined by $\theta_{i\pm1/2} = \max\{0, 1 - \frac{\Delta x_i}{|u|\Delta t}, 1 - \frac{\Delta x_{i\pm1}}{|u|\Delta t}\}$. The addition of corrected anti-diffusive fluxes is for the two-limiter approach done twice (last line in (5.35)). By applying the steps on pages 48 and 49 first for $\Delta f_{i,i\pm1}^n$ and next for $\Delta f_{i,i\pm1}^m$ we can show in a similar way that positivity is preserved for $\underline{b}^{m+1}$, and so for $\underline{c}^{n+1}$. Note that for the former case the expression $\tilde{c}_i^{n+1}$ is defined by the first two terms of the RHS of Equation (5.35) and for the latter case the expression $\tilde{c}_i^{n+1}$ is defined by the first four terms of the RHS of (5.35).

Further, due to the positivity-preserving property of $\underline{c}^n$ we have that also monotonicity is maintained for the two-limiter approach.

# Chapter 6

# Numerical results

In this chapter we present the numerical results for the advection equation by method 1A/B and method 2A/B presented in the previous chapter. Note that the number 1 stands for the one-limiter approach whereas the number 2 stands for the two-limiter approach. Also included in this chapter are some important insights into the numerical methods to understand the solution procedure.

## 6.1 One-limiter FCT approach

In this first section we apply the numerical iteration methods to the one-dimensional water quality model. A simplified version is the advection equation without any source terms. The one-dimensional problem reads

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} = 0, \quad x \in [0, 10],\ t \geq 0, \tag{6.1}$$

where $u > 0$ (substances flow from left to right) is constant and with periodic boundary conditions

$$c(0, t) = c(10, t), \quad t \geq 0 \tag{6.2}$$

and with initial condition $c(x, 0) = c_0$. To see the effect of the iterative FCT method we will use both a smooth initial condition as a non-smooth initial condition.

### 6.1.1 Method 1A

First of all we take a closer look at the results by the one-limiter approach without flux accumulation, i.e. method 1A. We apply the mentioned method on an uniform grid and with central discretization in space for the high order flux $F_H$ and first order upwind for the low order flux $F_L$. Choosing also central discretization in time corresponds with $\theta_{i+1/2} = 1/2$ at each cell interface. Furthermore, we profit from the fact that the numerical diffusion caused by the central approach in time is equal to zero (see the corresponding modified equation in Section 4.3.4).

Below we present the numerical equation from method 1A that has to be solved

$$c_i^{n+1,m+1} + \frac{u\Delta t}{2\Delta x_i}(c_i^{n+1,m+1} - c_{i-1}^{n+1,m+1}) = c_i^n - \frac{u\Delta t}{2\Delta x_i}(c_i^n - c_{i-1}^n) +$$

$$l_{i+1/2}^m[\frac{u\Delta t}{2\Delta x_i}(\frac{c_i^n - c_{i+1}^n}{2} + \frac{c_i^{n+1,m} - c_{i+1}^{n+1,m}}{2})] + l_{i-1/2}^m[\frac{u\Delta t}{2\Delta x_i}(\frac{c_i^n - c_{i-1}^n}{2} + \frac{c_i^{n+1,m} - c_{i-1}^{n+1,m}}{2})]. \quad (6.3)$$

In order to have a positive, stable and non-oscillatory solution the following condition must be fulfilled

$$\theta \geq 1 - \frac{\Delta x_i}{u\Delta t}. \quad (6.4)$$

Since we are dealing with central discretization in time ($\theta = 0.5$), this means that the CFL number is equal to 2, i.e. $\frac{u\Delta t}{\Delta x} = 2$. This leads to the following equation

$$\begin{aligned} 2c_i^{n+1,m+1} - c_{i-1}^{n+1,m+1} &= c_{i-1}^n + l_{i+1/2}^m[\frac{c_i^n - c_{i+1}^n}{2} + \frac{c_i^{n+1,m} - c_{i+1}^{n+1,m}}{2}] \\ &+ l_{i-1/2}^m[\frac{c_i^n - c_{i-1}^n}{2} + \frac{c_i^{n+1,m} - c_{i-1}^{n+1,m}}{2}]. \end{aligned} \quad (6.5)$$

The number of time steps after $p$ periods can be computed by $N = \frac{10p}{u\Delta t}$.

For the following results we use a sinusoidal function for the initial condition $c_0$. In the first experiments we use N = 75 and $\epsilon = 0.001$. Since we are mainly interested in the limiter for the equation above and need to understand the computation of it, we present the quantities $P^{\pm}, Q^{\pm}$ and $R^{\pm}$ described in Section 5.1.1 for the first time-step. As start point for the iteration method we used $\underline{c}^{n+1,0} = \underline{c}^n$. After 1 iteration we have the following figures for the intermediate solution $\tilde{c}^{n+1}$ and the quantities $P^{\pm}, Q^{\pm}$ and $R^{\pm}$.

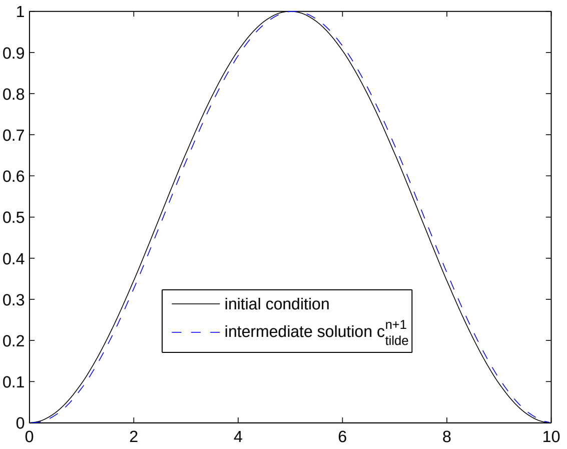

Figure 6.1: The initial condition and the intermediate solution at $t = 0.5\Delta t$

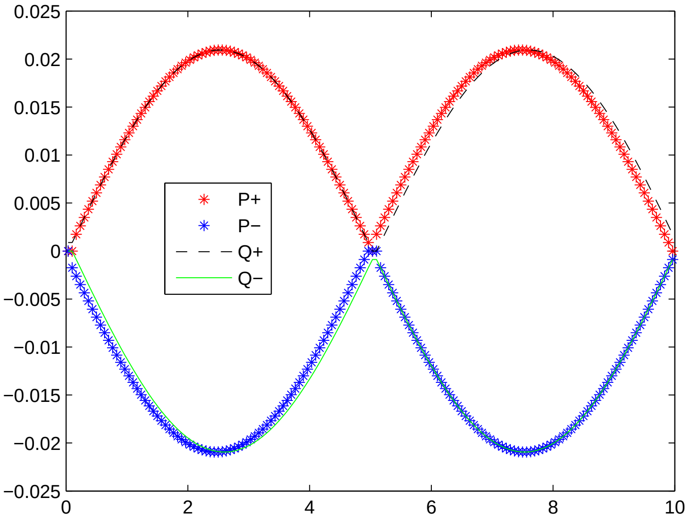

Figure 6.2: $P^{\pm}$ and $Q^{\pm}$ for the first iteration

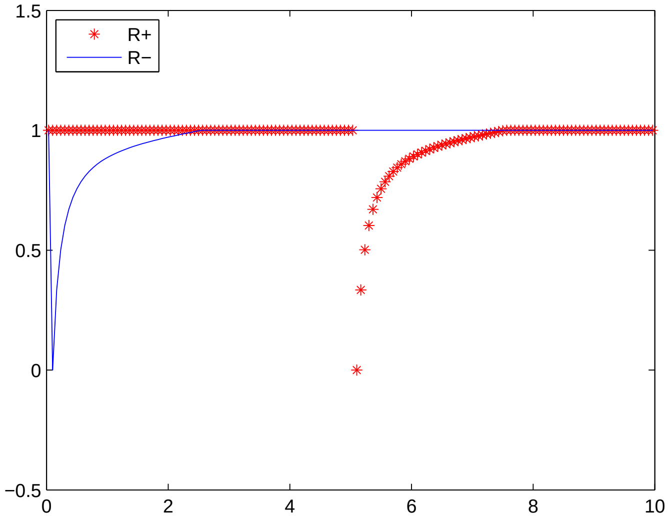

Figure 6.3: $R^{\pm}$ for the first iteration

Below we will show that the figures above are correct for $c_i^{n+1,0} = c_i^n$. Keep in mind that the results are necessary for a better understanding of the FCT method.

The intermediate solution $\tilde{c}_i^{n+1}$ is defined by the first two terms of the RHS of Equation (6.3). For CFL $= 2$ we get

$$\begin{aligned} \tilde{c}_i^{n+1} &= c_i^n - \frac{u\Delta t}{2\Delta x_i}(c_i^n - c_{i-1}^n) \\ &= c_{i-1}^n \end{aligned} \tag{6.6}$$

With a CFL $= 2$ this means that the intermediate solution is just a shift of one cell to the right which agrees with Fig 6.1.

The definitions for $P_i^+, Q_i^+$ and $R_i^+$ are for $\theta = 0.5$ defined by

$$\begin{aligned}
P_i^+ &= \max(0, \frac{c_i^n - c_{i-1}^n}{2} + \frac{c_i^{n+1,m} - c_{i-1}^{n+1,m}}{2}) + \max(0, \frac{c_i^n - c_{i+1}^n}{2} + \frac{c_i^{n+1,m} - c_{i+1}^{n+1,m}}{2}) \\
Q_i^+ &= c_i^{\max} - \tilde{c}_i^{n+1} \qquad (6.7) \\
R_i^+ &= \begin{cases} \min(1, \frac{Q_i^+}{P_i^+}), & P_i^+ \neq 0, \\ 1, & P_i^+ = 0. \end{cases}
\end{aligned}$$

For the "minus" quantities we must replace the term max by min in $P_i^+$ and $Q_i^+$. With these definitions we can verify Fig 6.2 and Fig 6.3.

First we take a closer look at the interval where the intermediate solution $\tilde{c}^{n+1}$ is increasing. For this situation with $m = 0$ we have

$$\frac{Q_i^+}{P_i^+} = \frac{c_i^n - c_{i-1}^n}{c_i^n - c_{i-1}^n} = 1. \qquad (6.8)$$

So $Q_i^+ = P_i^+$, hence $R_i^+ = 1$ for the increasing part of $\tilde{c}^{n+1}$.

For the increasingly declining part of the intermediate solution we have

$$\frac{Q_i^+}{P_i^+} = \frac{c_{i-1}^n - c_i^n}{c_i^n - c_{i+1}^n} = \frac{(c_i^n - c_{i-1}^n)/\Delta x}{(c_{i+1}^n - c_i^n)/\Delta x} < 1, \qquad (6.9)$$

where $\Delta x$ is the distance between two neighboring grid points. So $Q_i^+ < P_i^+$, hence $R_i^+ < 1$ for the increasing declining part of $\tilde{c}^{n+1}$.

For the decreasingly declining part of $\tilde{c}^{n+1}$ we have

$$\frac{Q_i^+}{P_i^+} = \frac{c_{i-1}^n - c_i^n}{c_i^n - c_{i+1}^n} = \frac{(c_i^n - c_{i-1}^n)/\Delta x}{(c_{i+1}^n - c_i^n)/\Delta x} > 1. \qquad (6.10)$$

And so $Q_i^+ > P_i^+$, hence $R_i^+ = 1$ for the decreased declining part of $\tilde{c}^{n+1}$.

For the maximum holds $P_i^+ > 0$ and $Q_i^+ = 0$, so $R_i^+ = 0$.
For the minimum holds $P_i^+ = 0$ , so $R_i^+ = 1$.

The results above agree with Figures 6.2 and 6.3 presented above. On a similar way we can define the values for $Q^-, P^-$ and $R^-$. This will be done below.

For the increasingly rising part of the intermediate solution $\tilde{c}^{n+1}$ we have

$$\frac{Q_i^-}{P_i^-} = \frac{c_{i-2}^n - c_{i-1}^n}{c_i^n - c_{i+1}^n} = \frac{(c_{i-1}^n - c_{i-2}^n)/\Delta x}{(c_{i+1}^n - c_i^n)/\Delta x} < 1. \qquad (6.11)$$

So $Q_i^- > P_i^-$, hence $R_i^- < 1$.

For the decreasingly rising part of $\tilde{c}^{n+1}$ we have

$$\frac{Q_i^-}{P_i^-} = \frac{c_{i-2}^n - c_{i-1}^n}{c_i^n - c_{i+1}^n} = \frac{(c_{i-1}^n - c_{i-2}^n)/\Delta x}{(c_{i+1}^n - c_i^n)/\Delta x} > 1. \tag{6.12}$$

So $Q_i^- < P_i^-$, hence $R_i^- = 1$.

For the decreasing part of $\tilde{c}^{n+1}$ we have

$$\frac{Q_i^-}{P_i^-} = \frac{c_i^n - c_{i-1}^n}{c_i^n - c_{i-1}^n} = 1. \tag{6.13}$$

Thus $Q_i^- = P_i^-$, hence $R_i^- = 1$.

For the maximum holds $P^- = 0$, so $R_i^- = 1$.
For the minimum holds $P^- < 0$ and $Q_i^- = 0$, so $R_i^- = 0$.

Also these results agree with Figures 6.2 and 6.3 presented above.

Since for the limiter itself holds that the minimum is taken between $R^+$ and $R^-$, we get obviously the following result out of Figure 6.3

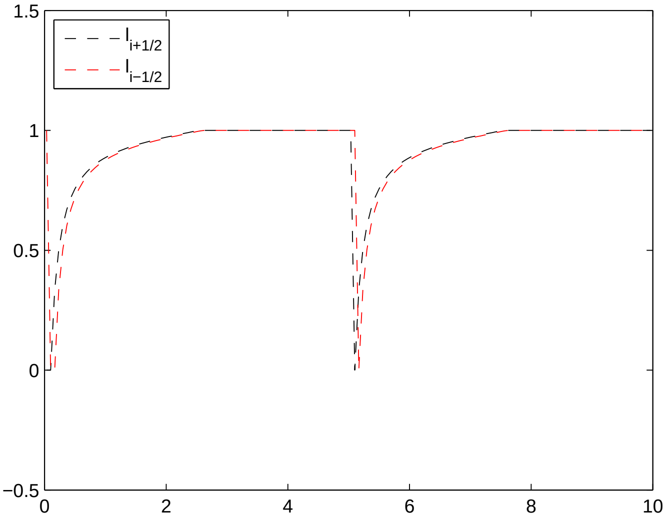

Figure 6.4: The limiter for the first iteration

For the second iteration (final iteration due to $\epsilon = 0.001$) we have the following results

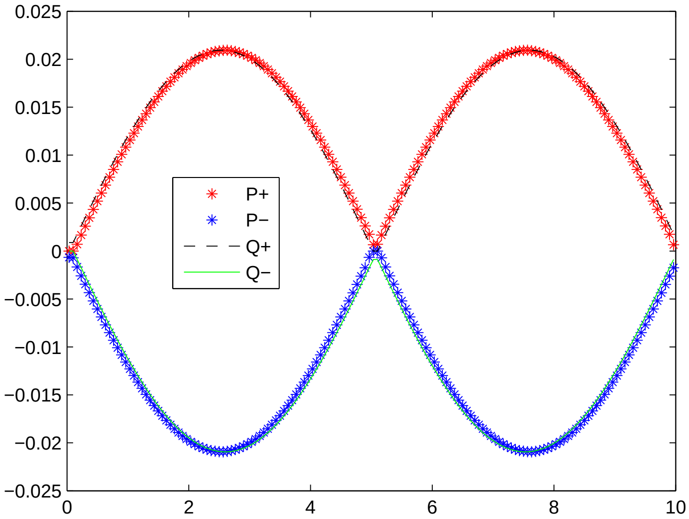

Figure 6.5: $P^{\pm}$ and $Q^{\pm}$ for the second iteration

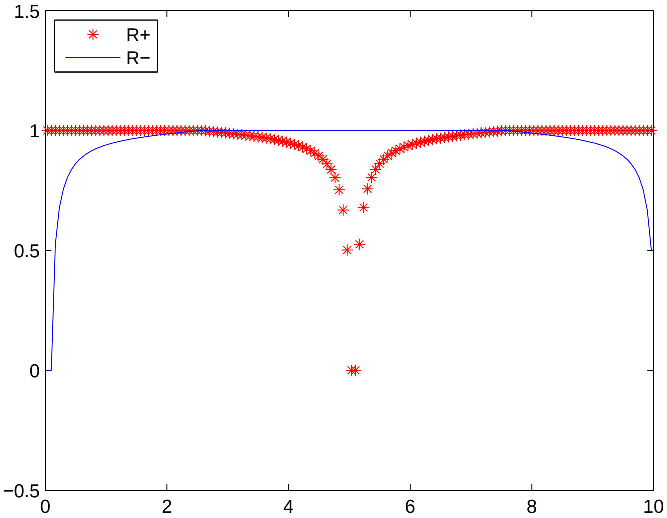

Figure 6.6: $R^{\pm}$ for the second iteration

For the verification of these figures we also need the following figure

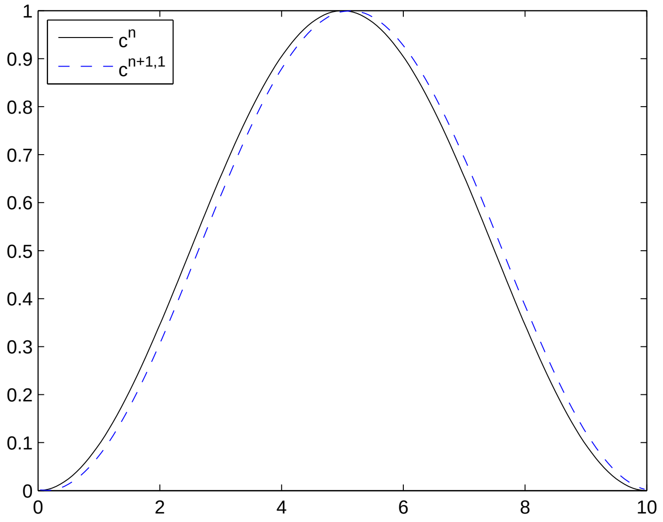

Figure 6.7: The initial condition and the solution after 1 iteration

On the interval where the intermediate solution $\tilde{c}^{n+1}$ is increasingly rising we have

$$\begin{aligned}\frac{Q_i^+}{P_i^+} &= \frac{c_i^n - c_{i-1}^n}{(c_i^n - c_{i-1}^n)/2 + (c_i^{n+1,1} - c_{i-1}^{n+1,1})/2} \\ &= \frac{2(c_i^n - c_{i-1}^n)/\Delta x}{(c_i^n - c_{i-1}^n)/\Delta x + (c_i^{n+1,1} - c_{i-1}^{n+1,1})/\Delta x} \\ &> \frac{2(c_i^n - c_{i-1}^n)/\Delta x}{2(c_i^n - c_{i-1}^n)/\Delta x} = 1. \end{aligned} \tag{6.14}$$

So $Q_i^+ > P_i^+$, hence $R_i^+ = 1$.

On the interval where $\tilde{c}^{n+1}$ is decreasingly rising we have

$$\begin{aligned}\frac{Q_i^+}{P_i^+} &= \frac{c_i^n - c_{i-1}^n}{(c_i^n - c_{i-1}^n)/2 + (c_i^{n+1,1} - c_{i-1}^{n+1,1})/2} \\ &= \frac{2(c_i^n - c_{i-1}^n)/\Delta x}{(c_i^n - c_{i-1}^n)/\Delta x + (c_i^{n+1,1} - c_{i-1}^{n+1,1})/\Delta x} \\ &< \frac{2(c_i^n - c_{i-1}^n)/\Delta x}{2(c_i^n - c_{i-1}^n)/\Delta x} = 1. \end{aligned} \tag{6.15}$$

So $Q_i^+ < P_i^+$, hence $R_i^+ < 1$.

On the interval where $\tilde{c}^{n+1}$ is increasingly declining we have

$$\begin{aligned}\frac{Q_i^+}{P_i^+} &= \frac{c_{i-2}^n - c_{i-1}^n}{(c_i^n - c_{i+1}^n)/2 + (c_i^{n+1,1} - c_{i+1}^{n+1,1})/2} \\ &= \frac{2(c_{i-1}^n - c_{i-2}^n)/\Delta x}{(c_{i+1}^n - c_i^n)/\Delta x + (c_{i+1}^{n+1,1} - c_i^{n+1,1})/\Delta x} \\ &< \frac{2(c_{i-1}^n - c_{i-2}^n)/\Delta x}{2(c_{i+1}^{n+1,1} - c_i^{n+1,1})/\Delta x} < 1. \qquad (6.16)\end{aligned}$$

So $Q_i^+ < P_i^+$, hence $R_i^+ < 1$.

On the interval where $\tilde{c}^{n+1}$ is decreasingly declining we have

$$\begin{aligned}\frac{Q_i^+}{P_i^+} &= \frac{c_{i-2}^n - c_{i-1}^n}{(c_i^n - c_{i+1}^n)/2 + (c_i^{n+1,1} - c_{i+1}^{n+1,1})/2} \\ &= \frac{2(c_{i-1}^n - c_{i-2}^n)/\Delta x}{(c_{i+1}^n - c_i^n)/\Delta x + (c_{i+1}^{n+1,1} - c_i^{n+1,1})/\Delta x} \\ &> \frac{2(c_{i-1}^n - c_{i-2}^n)/\Delta x}{2(c_{i+1}^{n+1,1} - c_i^{n+1,1})/\Delta x} > 1. \qquad (6.17)\end{aligned}$$

So $Q_i^+ > P_i^+$, hence $R_i^+ > 1$.

For the maximum of $\tilde{c}^{n+1}$ holds $P^+ > 0$ and $Q_i^+ = 0$, so $R_i^+ = 0$.
For the minimum holds $P^+ = 0$, so $R_i^+ = 1$.

The values for $Q^+, P^+$ and $R^+$ indeed agree with Figures 6.5 and 6.6 above. In a similar way the values for the "minus" quantities can be verified. Also for these cases we have agreement with these figures.

Since for the limiter itself holds that the minimum is taken between $R^+$ and $R^-$, we get obviously the following result out of Figure 6.6

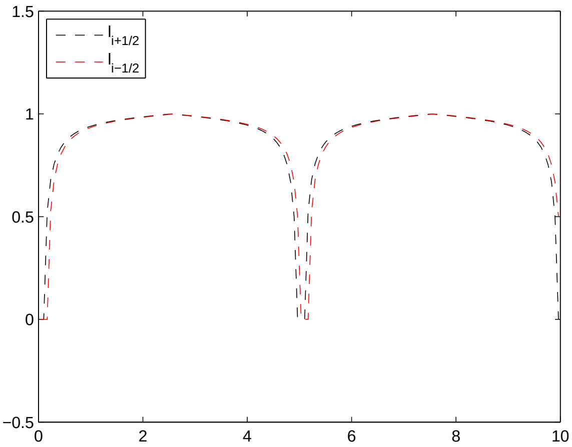

Figure 6.8: The limiter for the second iteration

The analysis above shows that due to the behavior in the derivative of the intermediate solution $\tilde{c}^{n+1}$ the limiter is not everywhere equal to 1. This makes the numerical solution less accurate than desired. Below we will show the effect of this disadvantage on the numerical result after 1 period.

In the analysis above we only deal with two iterations in the first time step due to the stop-criterion. If a much smaller value is used for $\epsilon$ then we expect more iterations. In the figure below we present the error $||\underline{c}^{n+1,m+1} - \underline{c}^{n+1,m}||_1$ as function of the number of iterations $m$. The figure shows that the error is a decreasing function of the number of iterations. Hence a smaller value for $\epsilon$ corresponds with a larger number of iterations. We used for all the time steps $\epsilon = 0.001$, which corresponds according to the figure below with 2 or 3 iterations per time step.

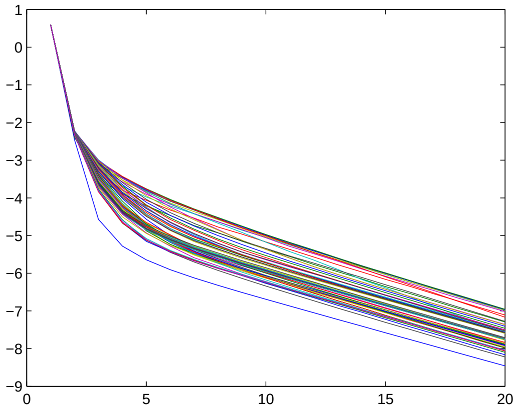

Figure 6.9: The log-error as function of the number of iterations per time step. Figure shows the errors for all time steps (N = 75). The vertical axis is on logarithmic scale.

After 1 period we have the following result for the FCT solution compared to the exact and the first order upwind solution. Remember that the numerical methods are implicit, since $\theta = 0.5$. The upwind solution can be obtained by numerical scheme (6.5), but with limiter $l_{i+1/2} = 0$ at each cell interface. This upwind method can also be seen as the least achievable result by the FCT method.

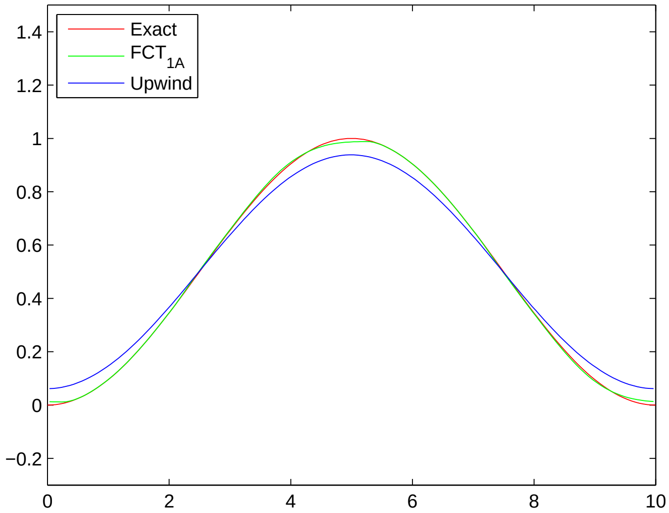

Figure 6.10: The numerical solution after 1 period

We are also interested in the numerical results after grid refinement. The results for the accuracy are presented in the table below. As measure for the accuracy of the numerical solution the Root Mean Squared Error (RMSE) value is used, which is defined by

$$RMSE = \sqrt{}(\frac{\sum_i (q_i - c_i)^2}{nc}), \tag{6.18}$$

where $q_i$ and $c_i$ is the exact solution and numerical solution respectively for $x_i$. Further is $nc$ the total number of cells.

Table 6.1: The global error of the numerical solutions

| *Total grid cells* | $RMSE_{upwind}$ | $RMSE_{FCT}$ | $RMSE_{FCT}$ *with* $l = 1$ |
|---|---|---|---|
| 150 | 0.0435 | 0.0054 | 0.0019 |
| 300 | 0.0225 | 0.0017 | 0.00049 |
| 600 | 0.0114 | 0.00052 | 0.00012 |

In the final column of the table we find the errors of the FCT method if the limiter is set equal to 1 at each cell interface. This can also be seen as the best achievable result by the FCT method. Due to the gradient behavior of the sinusoid this accuracy is not attained as discussed in the section above. The results in the table are presented in the figure below as the log-RMSE against the log-$\Delta x$.

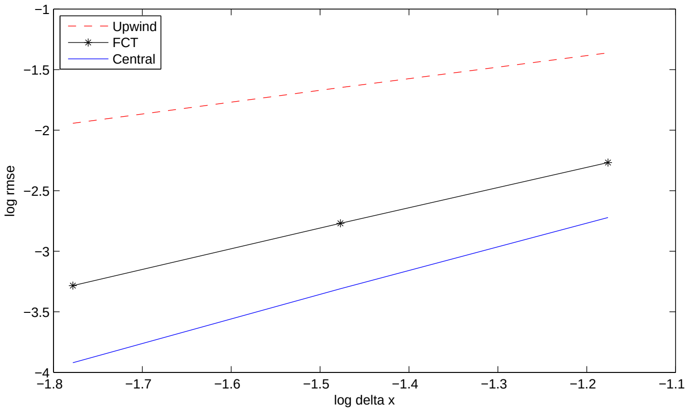

Figure 6.11: The error against the gridsize both on logarithmic scale

The slope of each plot indicates the order of accuracy of the numerical methods. For the upwind implicit method we have a slope of 0.97, i.e. $O(h)$. For the central implicit method we have a slope equal to 1.99, i.e. $O(h^2)$. Though, for the method were we are most interested in has a slope of 1.7. This implies that the FCT method has no second order accuracy for smooth solutions as desired and therefore no full potential of the limiter is attained. This decrease in accuracy is caused by the gradient of the sinusoid which follows from the analysis in the beginning of this section.

Conclusions for smooth solutions

The reason for using a sinusoidal function is that we deal with different types of behavior in the numerical solution. From these results we can draw accurate conclusions about the FCT method and furthermore these conclusions can be extended to general smooth solutions. From the results above we can conclude the following:

- At extremal points the limiter is equal to zero, which has a negative effect on the accuracy. Even if grid-refinement is applied this problem will be maintained.
- For some parts of the sinusoid the limiter is not "optimal", i.e. $l_{i+1/2} = 1$, hence no second order accuracy. It seems that in the neighborhood of a maximum and minimum this drawback is more likely to appear. On the other hand, where the absolute gradient is maximal the limiter is fully attained ($l_{i+1/2} = 1$). An explanation can be found in the rate $Q_i^\pm / P_i^\pm$. A larger absolute gradient corresponds with a larger absolute value for $Q_i^\pm$, which implies a higher value for its limiter.
- More improvement in the FCT method must be possible for smooth solutions, since unnecessarily smaller values for the limiter are used. The computation of the limiter is mainly based on the gradient behavior of the intermediate solution $\tilde{c}^{n+1}$. A possibility is to make the value for $Q_i^\pm$ less sensitive to the gradient of $\tilde{c}^{n+1}$.

We will also present briefly the results for a block-shaped solution. For this case we use again numerical scheme (6.5) with 150 grid cells(so $\theta = 0.5$ and $CFL = 2$). We have the following results

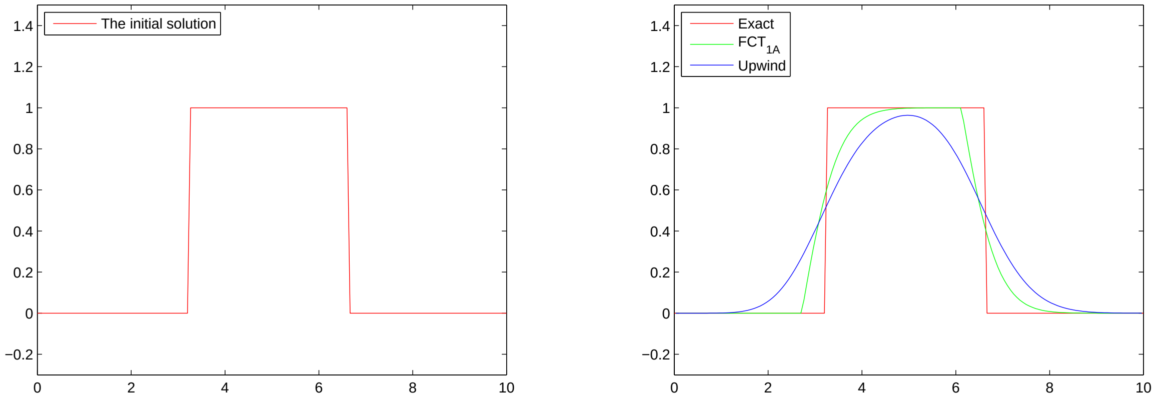

Figure 6.12: Left: The initial solution. Right: The numerical solution after 1 period for 150 grid cells.

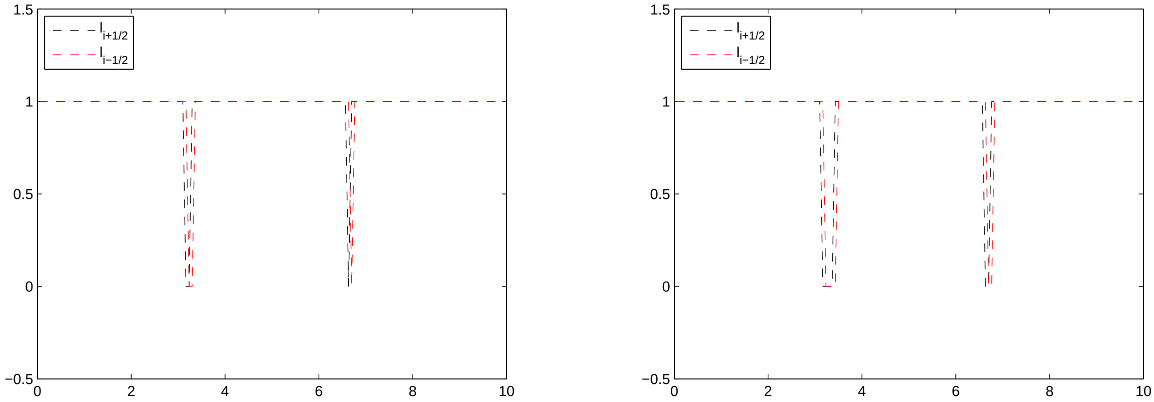

Figure 6.13: The limiters for iteration step 1 and 2 for the first time-step.

In the figures presenting the value for the limiter we notice that the limiter is set equal to 1, except at the locations where a discontinuity appears in the numerical solution. For these points the limiter is zero. If a higher value for the limiter would appear at the discontinuity location, then this would cause wiggles in the numerical solution. Therefore, the first order method must be applied to prevent non-physically behavior, which is the case since the limiter is zero.

Below we have a table showing the errors of the FCT method and the upwind method, which can be seen as the least achievable results by the FCT method. The FCT method is approximately a factor 1.5 better then the upwind method. Though, this statement which is based on Table 6.2 is only valid for grids from 150 cells to 600 cells.

Table 6.2: The global error of the numerical solutions

| *Total grid cells* | $RMSE_{upwind}$ | $RMSE_{FCT}$ |
|---|---|---|
| 150 | 0.1872 | 0.1317 |
| 300 | 0.1592 | 0.1065 |
| 600 | 0.1351 | 0.0857 |

Conclusion for block-shaped solutions

Finally we can conclude that for block-shaped solutions the FCT method is performing as expected with limiter equal to zero in the neighborhood of discontinuities. This means that around these points the solution is approximated by only first order upwind in order to ensure positivity and monotonicity.

### 6.1.2 Method 1B

Below we present the numerical results by method 1B, which uses accumulation. Again the same parameters are used as in method 1A, i.e. uniform grid, $\theta = 0.5$ and CFL $= 2$. The numerical scheme reads

$$\begin{aligned} 2c_i^{n+1,m+1} - c_{i-1}^{n+1,m+1} = b_i^m + l_{i+1/2}^m[\frac{c_i^n - c_{i+1}^n}{2} + \frac{c_i^{n+1,m} - c_{i+1}^{n+1,m}}{2} - g_{i,i+1}^m] + \\ l_{i-1/2}^m[\frac{c_i^n - c_{i-1}^n}{2} + \frac{c_i^{n+1,m} - c_{i-1}^{n+1,m}}{2} - g_{i,i-1}^m], \end{aligned} \tag{6.19}$$

with

$$\begin{aligned} b_i^m = b_i^{m-1} + l_{i+1/2}^{m-1}[\frac{c_i^n - c_{i+1}^n}{2} + \frac{c_i^{n+1,m-1} - c_{i+1}^{n+1,m-1}}{2} - g_{i,i+1}^{m-1}] + \\ l_{i-1/2}^{m-1}[\frac{c_i^n - c_{i-1}^n}{2} + \frac{c_i^{n+1,m-1} - c_{i-1}^{n+1,m-1}}{2} - g_{i,i-1}^{m-1}], \quad \text{with} \quad b_i^0 = \tilde{c}_i^{n+1} \end{aligned} \tag{6.20}$$

and

$$g_{i,i\pm1}^m = g_{i,i\pm1}^{m-1} + l_{i\pm1/2}^{m-1}[\frac{c_i^n - c_{i\pm1}^n}{2} + \frac{c_i^{n+1,m-1} - c_{i\pm1}^{n+1,m-1}}{2} - g_{i,i\pm1}^{m-1}], \qquad g_{i,i\pm1}^0 = 0. \tag{6.21}$$

For the following results we use the same sinusoidal function as before for the initial condition $c_0$. The results are compared with the exact solution and with the implicit upwind scheme, i.e. $l_{i\pm1/2} = 0$ for every cell interface.

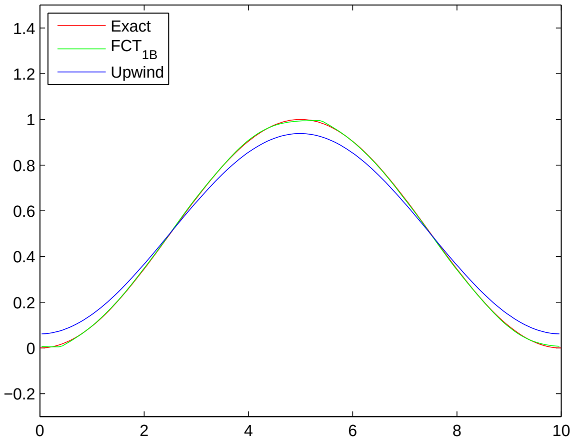

Figure 6.14: The numerical solution after 1 period for 150 grid cells

Below we have the corresponding limiters at the first time step

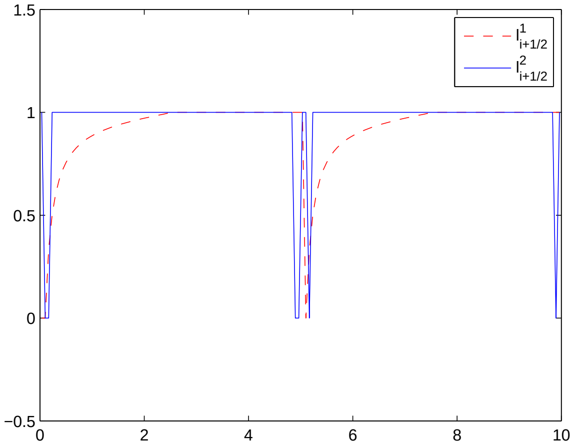

Figure 6.15: The limiter for the first and second iteration for the first time-step

We notice from the figure above that the limiter is improved after 2 iterations. Though, the limiters correspond in this case to the flux differences $\Delta f^m_{i+1/2}$ instead of the usual fluxes $f^m_{i+1/2}$. According to the figure above we have for the final iteration (m=2 due to stop criterion) no limiting of the flux difference, except at extremal points. An explanation that for later iterations (in this case only the second iteration) the flux corrections $\Delta f^m_{i\pm1/2}$ are almost fully allowed, is because the corrections become smaller in magnitude for each iteration step.

In Table 6.3 we find the results for method 1B and other numerical methods after refining the grid.

Table 6.3: The global error of the numerical solutions after 1 period

| *Total grid cells* | $RMSE_{upwind}$ | $RMSE_{FCT}$ method 1B | $RMSE_{FCT}$ method 1A |
|---|---|---|---|
| 150 | 0.0435 | 0.0035 | 0.0054 |
| 300 | 0.0225 | 0.0011 | 0.0017 |
| 600 | 0.0114 | 0.00034 | 0.00052 |

Based on the previous table we present the log-error against the log-grid size

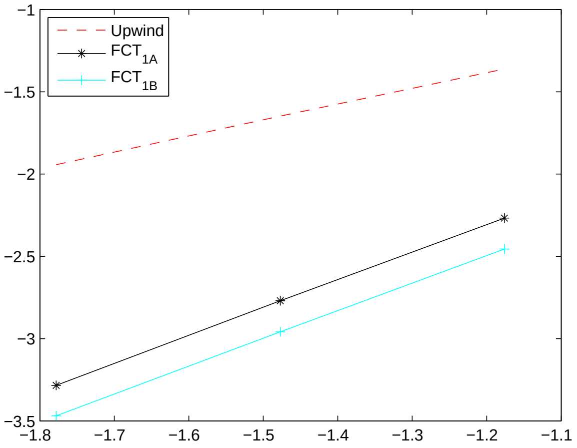

Figure 6.16: The error against the grid-size both on logarithmic scale

By the figure above we can compare the results on basis of their order of accuracy. The order of accuracy is equal to the slope of each plot. The slopes are equal to 0.97, 1.7 and 1.7 for upwind, method 1A and method 1B respectively. Method 1A and 1B have the same order of accuracy, but according to Table 6.3 the errors are for the latter method approximately a factor 1.5 lower than the former method.

Next we present the results for the block-shaped solution by method 1B compared to 1A. Together with the exact solution also the implicit upwind solution is shown. Further we have also given the limiters for the first time step.

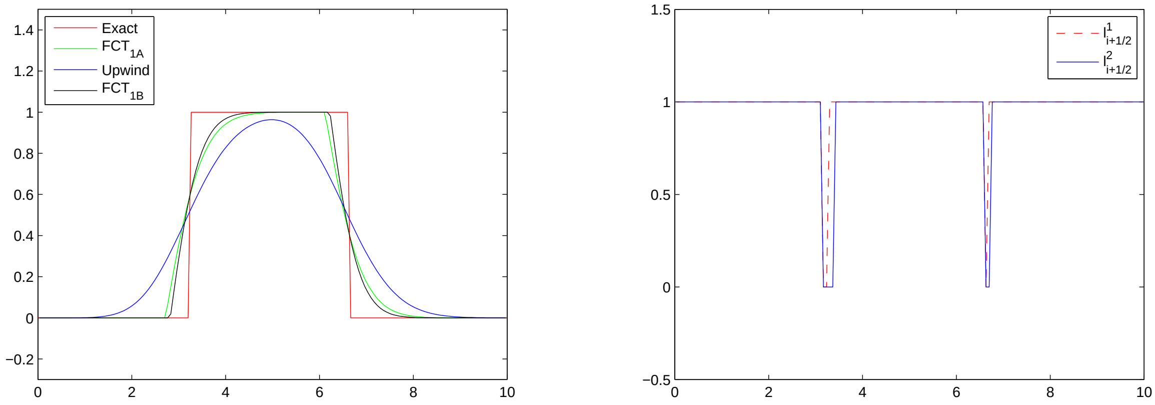

Figure 6.17: Left: The numerical solution after 1 period for 150 grid cells. Right: The limiters for iteration step 1 and 2 for the first time-step.

In the figure to the left we see that the numerical solution by method 1B is slightly better than method 1A. The limiter is as expected, with only zero values at the discontinuities. In Table 6.4 we present the errors for method 1B compared to other numerical methods after refining the grid. The errors confirm that method 1B is more accurate.

Table 6.4: The global error of the numerical solutions after 1 period

| *Total grid cells* | $RMSE_{upwind}$ | $RMSE_{FCT}$ method 1B | $RMSE_{FCT}$ method 1A |
|---|---|---|---|
| 150 | 0.1872 | 0.1150 | 0.1317 |
| 300 | 0.1592 | 0.0933 | 0.1065 |
| 600 | 0.1351 | 0.0754 | 0.0857 |

Conclusions

- For both the smooth and non-smooth solution we get a higher accuracy for the FCT solution by method 1B compared with the results for method 1A. For the sinusoid the errors by method 1B are approximately a factor 1.5 lower than method 1A and for the block-shaped situation the errors are approximately 1.2 lower. Though, the methods have for smooth solutions the same order of accuracy.
- An important aspect of method 1B for smooth solutions is that as the number of iterations increases the flux differences become less limited, so that more correction is allowed. This is a consequence of the decrease in magnitude of the correction terms for each iteration step.
- Also with method 1B we have zero limiter at extremal points for smooth solutions.

### 6.1.3 Improving method 1A

In the conclusion for method 1A applied on a sinusoidal function we mentioned that the limiter is highly dependent on the gradient behavior of the intermediate solution $\tilde{c}^{n+1}$. A possible remedy for this problem is to make the values for $Q_i^{\pm}$ less sensitive to the gradient of $\tilde{c}^{n+1}$. In the definition we have that $Q_i^{\pm}$ depends only on $\tilde{c}_i^{n+1}$, so one can also include other solutions in the computation for $Q_i^{\pm}$, which is the allowed flux increase/decrease. Since this is an approximation for the actual allowed flux increase/decrease (which is unknown) this new approach seems an acceptable alternative. In this new approach we let the computation for $Q_i^{\pm}$ depends also on $c_i^n$, so $Q_i^{\pm}$ will be less sensitive to the gradient behavior of the intermediate solution. This new approach is defined by

New approach:

$$\begin{aligned} c_i^{\max} &= \max_{i-1,i,i+1}[\max(c_j^n, \tilde{c}_j^{n+1})], \qquad (6.22)\\ c_i^{\min} &= \min_{i-1,i,i+1}[\min(c_j^n, \tilde{c}_j^{n+1})]. \qquad (6.23)\end{aligned}$$

The old approach is:

$$\begin{aligned} c_i^{\max} &= \max_{i-1,i,i+1} \tilde{c}_j^{n+1}, \qquad (6.24)\\ c_i^{\min} &= \min_{i-1,i,i+1} \tilde{c}_j^{n+1}. \qquad (6.25)\end{aligned}$$

Below we show the effect of the new approach, implemented in method 1A, on the limiter. For comparison the old approach is given.

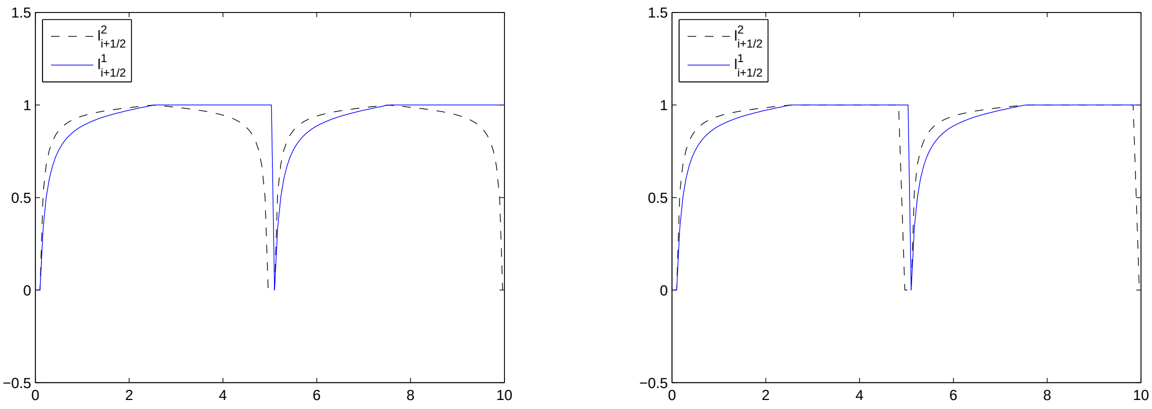

Figure 6.18: The limiters for iteration step 1 and 2 for the first time-step. Left: old approach, Right: new approach

From the figures above we clearly notice that for the new approach the limiter is improved after two iterations. Compared to the old approach we have a better limiter after two iterations. In the table below we give the errors of this new approach compared to the other numerical methods.

Table 6.5: The global error of the numerical solutions after 1 period

| *Total grid cells* | $RMSE_{FCT}$<br>1A: old approach | $RMSE_{FCT}$<br>1A: new approach | $RMSE_{FCT}$<br>method 1B | $RMSE_{FCT}$<br>1A: with $l = 1$ |
|---|---|---|---|---|
| 150 | 0.0054 | 0.0032 | 0.0035 | 0.0019 |
| 300 | 0.0017 | 0.00097 | 0.0011 | 0.00049 |
| 600 | 0.00052 | 0.00030 | 0.00034 | 0.00012 |

The new approach leads to a significantly better solution than the old approach. The errors are even lower than method 1B. The final column represents the lowest achievable error by the FCT method. In the figure below we plot the numerical solution for this new approach.

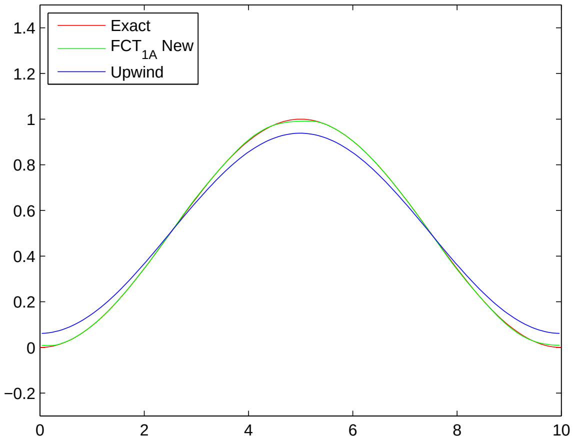

Figure 6.19: Numerical solutions after 1 period for 150 grid cells

Based on the table above we are able to plot the error against the grid-size on logarithmic scale in order to determine the order of accuracy for each method in Table 6.5

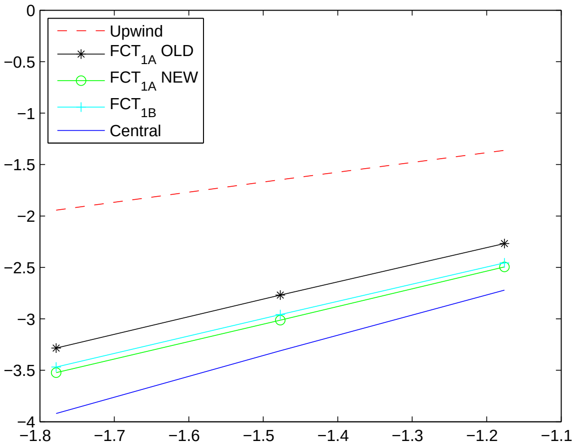

Figure 6.20: The error against the grid-size both on logarithmic scale

In the picture above we can compare the numerical methods against each other. The results by method 1A and 1B (”central” excluded) seems to have similar order of accuracy, which can be observed from the slope of each plot. The slopes for these methods are indeed equal. They have a slope of 1.7, hence no second order accuracy. The upwind and central method (1A with l = 1) have a slope of 0.97 and 1.99 respectively. Based on these results we prefer method 1A with the new approach above method 1B.

Next we apply the new approach to the block-shaped solution. The results are given below.

Table 6.6: The global error of the numerical solutions after 1 period

| *Total grid cells* | $RMSE_{FCT}$<br>1A: old approach | $RMSE_{FCT}$<br>1A: new approach | $RMSE_{FCT}$<br>method 1B |
|---|---|---|---|
| 150 | 0.1317 | 0.1156 | 0.1150 |
| 300 | 0.1065 | 0.0936 | 0.0933 |
| 600 | 0.0857 | 0.0755 | 0.0754 |

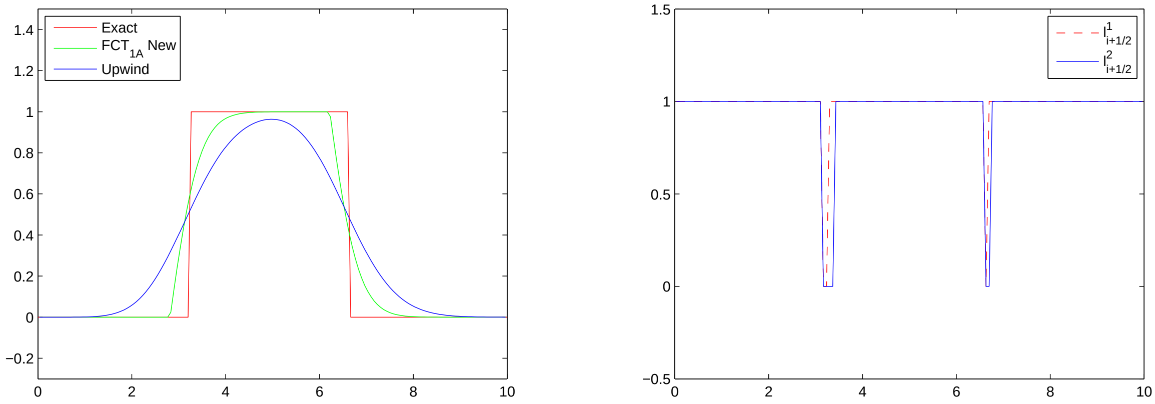

Figure 6.21: Left: The numerical solution by the new approach for 150 grid cells. Right: The limiters for iteration step 1 and 2 for the first time step.

Also for the block-shaped solution we get a higher accuracy by the new approach compared to the old approach, but similar to method 1B. Based on these results the methods in the final two columns of Table 6.6 can be chosen. The same holds for the sinusoidal case. For the limiter based on the new approach (Figure 6.21) we see no difference in comparison with the limiter based on the old approach (Figure 6.13). The limiter gives values equal to 1 except at the discontinuities where the limiter is zero.

Conclusions

- Method 1A (both approaches) has the same order of accuracy as method 1B.
- Method 1A with the new approach for $c_i^{\max}$ and $c_i^{\min}$ gives similar results (errors) as method 1B.

### 6.1.4 Variable and different theta

The results we have seen so far are valid for $\theta = 0.5$ on the whole domain for both the first-order method as the high-order method. According to the analysis in Section 4.3.4 where we derived the modified equation, we know that for this choice of theta the theta scheme with central discretization has zero numerical diffusion. The only numerical diffusion for the FCT scheme is caused by the upwind method.

Remember the numerical fluxes which where defined according to Section 5.1.1 by

$$\begin{aligned} F_{i-1/2} &= (1-\theta_{i-1/2})[F_L(c^n_{i-1/2}) + l_{i-1/2}(F_H(c^n_{i-1/2}) - F_L(c^n_{i-1/2}))] \\ &+ \theta_{i-1/2}[F_L(c^{n+1}_{i-1/2}) + l_{i-1/2}(F_H(c^{n+1}_{i-1/2}) - F_L(c^{n+1}_{i-1/2}))] \\ &= (1-\theta_{i-1/2})[(1-l_{i-1/2})F_L(c^n_{i-1/2}) + l_{i-1/2}F_H(c^n_{i-1/2})] \\ &+ \theta_{i-1/2}[(1-l_{i-1/2})F_L(c^{n+1}_{i-1/2}) + l_{i-1/2}F_H(c^{n+1}_{i-1/2})] \qquad (6.26) \end{aligned}$$

This formulation for the fluxes shows that a convex combination is used between the first-order and high-order method. Hence, this will also holds for the numerical diffusion terms of both methods. These artificial diffusion terms are given in the modified equations given in Section 4.3.4. For the theta-scheme with first-order fluxes we have a numerical diffusion given by

$$\frac{u\Delta x}{2}(1-(1-2\theta)\frac{u\Delta t}{\Delta x}). \qquad (6.27)$$

For the theta-scheme with central fluxes we have

$$(\theta - \frac{1}{2})u^2\Delta t. \qquad (6.28)$$

Using a convex combination of both numerical diffusion terms (6.27) and (6.28) we get the numerical diffusion for the implicit FCT scheme with central fluxes for $F_H$

$$\begin{aligned} (1-l)[\frac{u\Delta x}{2}(1-(1-2\theta)\frac{u\Delta t}{\Delta x})] + l(\theta-\frac{1}{2})u^2\Delta t = \\ (1-l)\frac{u\Delta x}{2} + (\theta - \frac{1}{2})u^2\Delta t. \qquad (6.29) \end{aligned}$$

From the RHS in Expression (6.29) we see that for $\theta = 0.5$ and $l = 1$ the artificial diffusion is zero. Since it is hard to get maximal limiter on the whole domain by the FCT method, i.e. $l_{i-1/2} = 1$, we have that the numerical diffusion caused by the upwind method is always present, unless $\theta = 0$ with CFL = 1. Though, the amount of numerical diffusion by upwind can be reduced by the numerical diffusion term of the central method for $\theta < 0.5$. If the reduction by the latter term is too large then the summed numerical diffusion becomes negative, which is in this case called anti-diffusion. For $\theta > 0.5$ there will occur only smearing of the numerical solution, because the numerical diffusion is in this case always positive.

In the figures below we present the numerical diffusions by Equations (6.27)-(6.29) as function of theta and CFL respectively. The results in the figure below are presented under the condition $\theta = \max\{1 - \frac{1}{CFL}, 0\}$ (see also (5.7)).

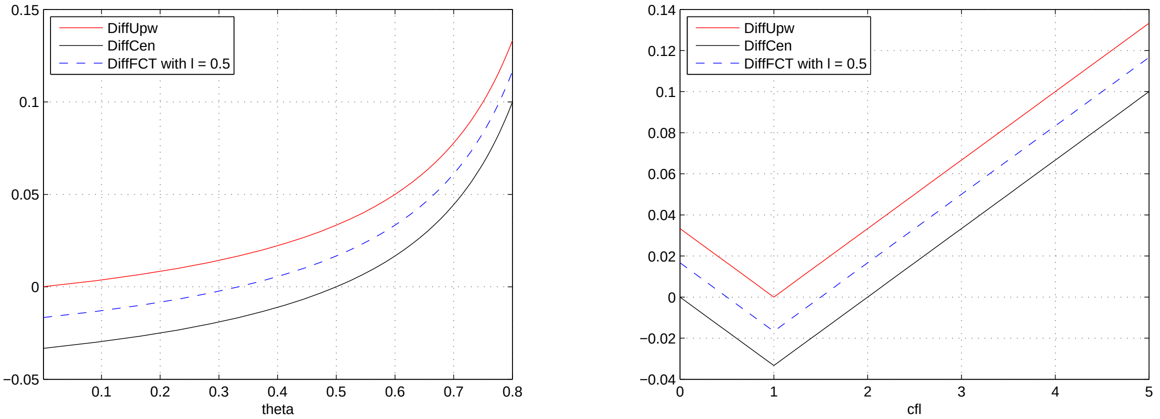

Figure 6.22: Left: The numerical diffusion as function of theta and CFL respectively

In the left figure we see that the numerical diffusion is an increasing function of theta. In the right figure we present the numerical diffusion as function of the CFL number. For both figures hold that for small values of the limiter the numerical diffusion graph of the FCT method is closer or equal ($l = 0$) to the graph of the upwind method, while for larger values it is closer or equal ($l = 1$) to the graph of the central approach. Further, from each plot we can see for which values of theta or CFL the numerical diffusion is zero or positive.

Below we see the results for the numerical solution for different values of theta. For method 1A we use the new approach given in the previous section. As initial condition we use a combination of a sinusoid and a block-shaped solution.

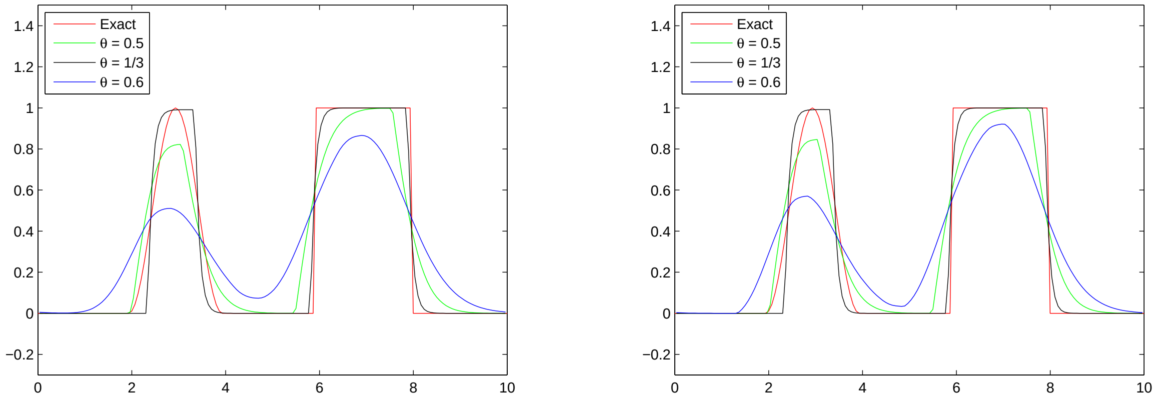

Figure 6.23: Numerical solution after 1 period by method 1A and 1B respectively. Total cells used: 150. $\theta = 1/3$ and $\theta = 0.6$ corresponds respectively with CFL = 1.5 and CFL = 2.5.

We indeed have for $\theta < 0.5$ smearing of the numerical solution, while for $\theta > 0.5$ we deal with anti-diffusion. This anti-diffusion is causing a block-shaped numerical solution which is only in favor of the block-part. Despite of this negative effect to the sinusoidal part we obtain nevertheless the lowest RMSE-error for $\theta = 1.3$. This holds for both methods.

Actually, the present of anti-diffusion is not desired, especially for initial conditions containing a smooth part. Therefore we introduce another approach by using different theta values for the upwind and the central method. For the central method we take $\theta = 0.5$ fixed so that the numerical diffusion caused by this method is zero, while the theta value for the upwind method is defined by Equation (5.7). In this case we deal only with non-negative numerical diffusion for the FCT scheme, hence no anti-diffusion. According to this approach we have that the numerical diffusion term for the FCT scheme is now defined by

$$(1-l)\frac{u\Delta x}{2}(1-(1-2\theta)\frac{u\Delta t}{\Delta x}). \tag{6.30}$$

In the figure below we compare the numerical diffusion of (6.30) with the numerical diffusion of (6.29). Again under the condition $\theta = \max\{1 - \frac{1}{CFL}, 0\}$.

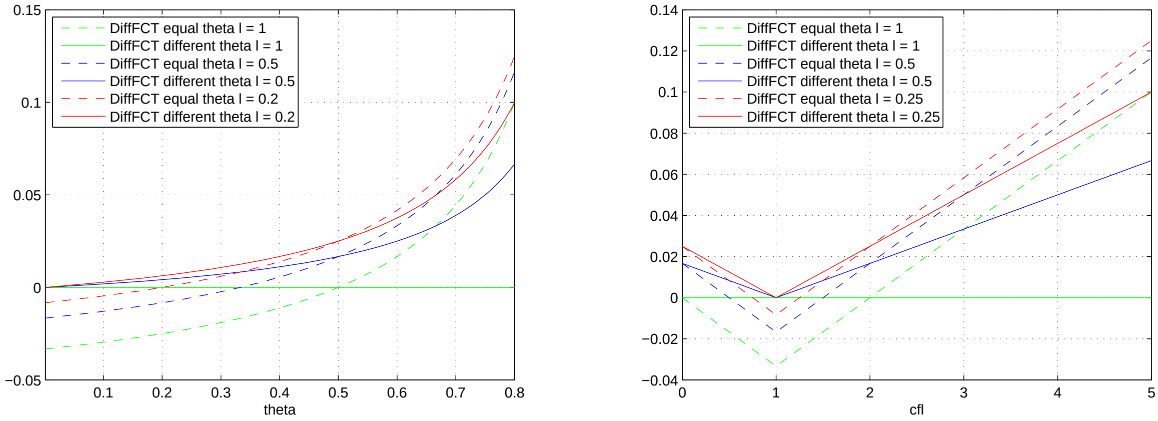

Figure 6.24: Left: The numerical diffusion as function of theta and CFL respectively

In the figures above we present the results for different values for the limiter. We see that (6.30) and (6.29) are equal for $\theta = 0.5$ and CFL $= 2$. For larger values than these “intersection points” the different theta approach leads to a lower amount of numerical diffusion than with equal theta for $F_L$ and $F_H$.

The fluxes differences $\Delta f^m_{i,i\pm1}$ in method 1A are now defined by

$$\begin{aligned}\Delta f^m_{i,i\pm1} &= \mp\frac{\Delta t}{\Delta x_i}[\frac{1}{2}F_H(c^n_i, c^n_{i\pm1}) - (1-\theta_{i\pm1/2})F_L(c^n_i, c^n_{i\pm1}) \\ &+ \frac{1}{2}F_H(c^{n+1,m}_i, c^{n+1,m}_{i\pm1}) - \theta_{i\pm1/2}F_L(c^{n+1,m}_i, c^{n+1,m}_{i\pm1})].\end{aligned}$$

For method 1B holds a similar formulation for the flux differences which we will not mention here. With this new formulation for $\Delta f^m_{i,i\pm1}$ we obtain other results as Figure 6.25 below demonstrates.

The numerical solutions in Figure 6.23 and Figure 6.25 are equal for $\theta = 0.5$ (CFL $= 2$) as Figure 6.24 indicates. For $\theta = 1/3$ we have no block-shaped numerical solution since no anti-diffusion is available. Instead we have smearing of the numerical solution. For $\theta = 0.6$ we expect more accurate results with different theta for $F_H$ and $F_L$, since the presence of numerical diffusion is now only caused by the upwind method (see also Figure 6.24). At first sight both results by method 1A seems equal, but the RMSE-error is for Figure 6.25 slightly lower (0.223495 against 0.223503). According to Figure 6.24 the different theta approach must be more accurate. For method 1B we get also more accurate results for $\theta = 0.6$ compared with the situation in Figure 6.23, but the differences are for method 1B much larger.

Another important result is that more accurate results are obtained with method 1B than with method 1A. So with the different theta approach for $F_L$ and $F_H$ the former method is preferred.

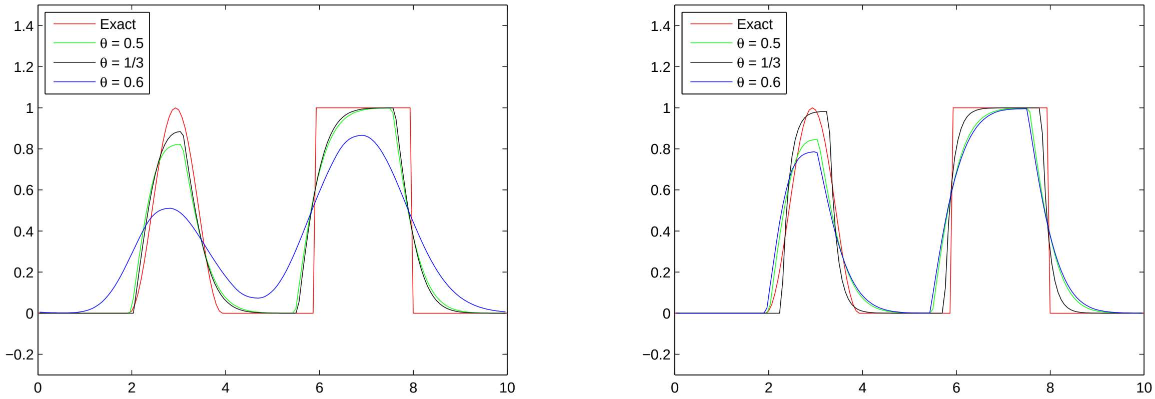

Figure 6.25: Numerical solution after 1 period by method 1A and 1B respectively. Total cells used: 150. $\theta = 1/3$ and $\theta = 0.6$ corresponds respectively with CFL = 1.5 and CFL = 2.5.

### Non-uniform grid

In Chapter 5 we used in the definition of the flux difference $\Delta f^m_{i,i\pm1}$ a local theta value per flux, which means that we can have a variable theta over the domain of interest. Variable theta only occurs for non-uniform grids. For uniform grids we know that the theta values are constant due to the cell size which is equal everywhere. Below we present the numerical results with central fluxes for $F_H$ on a non-uniform grid. Further, we use the different theta approach for $F_L$ and $F_H$ discussed above such that anti-diffusion is avoided.

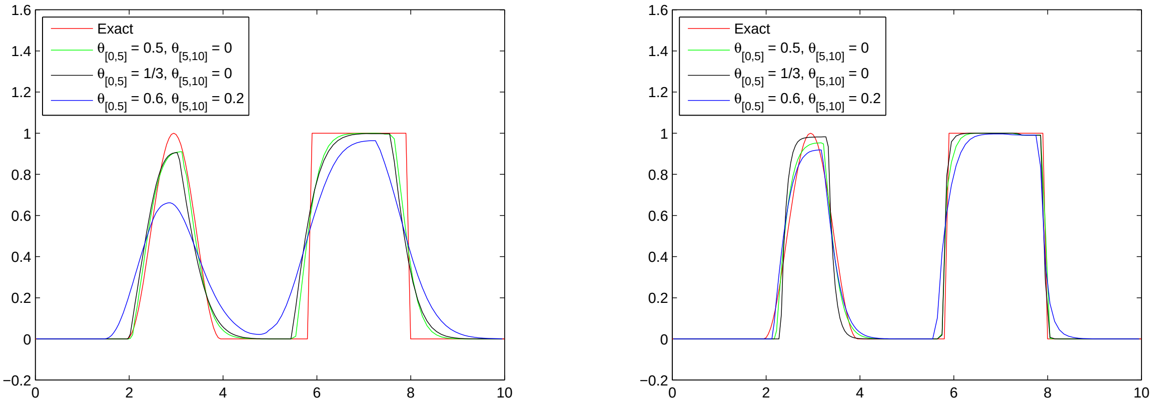

Figure 6.26: Numerical solutions after 1 period by method 1A and 1B respectively. Non-uniform grid with 100 cells on [0,5] and 50 cells on [5,10]

In the figure above we present the results for different CFL values. On the domain [0,5] we have (following the legend in the figures) respectively CFL = 2, CFL = 1.5 and CFL = 2.5, whereas on [5,10] the CFL values are CFL = 1, CFL = 0.75 and CFL = 1.25 respectively. For the non-uniform case we obtain more accurate results than the uniform case given in Figure 6.25, especially for method 1B (right figures). For method 1A (left figures) this is confirmed by the RMSE-error: 0.1287, 0.1140 and 0.2235 in Figure 6.25 against 0.737, 0.0984 and 0.1594 in Figure 6.26. Note that the values for theta belong to $F_L$, since for $F_H$ the theta value is equal to 0.5.

Again we noticed that method 1B leads to higher accuracy compared with method 1A, which we already observed for the uniform case (Figure 6.22).

An important aspect for the non-uniform case with different theta for $F_L$ and $F_H$ is that for theta equal to zero the implicit part of the numerical scheme is constantly present due to theta for $F_H$ which is equal to 0.5 (see also the flux formulation given in this section). So the numerical solution at the new time-level is always included in the computation. This means for the figure above that for those locations where theta is equal to zero an implicit scheme is used, whereas normally for $\theta = 0$ we have an explicit time scheme. So, for different theta for $F_L$ and $F_H$ applied on a non-uniform grid we always deal with an implicit numerical method.

For the non-uniform case we can apply instead of central fluxes also another high-order flux function $F_H$ in method 1A and 1B. This is done by using Lax-Wendroff flux correction for $F_H$ when $\theta = 0$ and central correction for $F_H$ when $\theta > 0$. This approach is chosen since for $\theta = 0$ the numerical diffusion for Lax-Wendroff fluxes is equal to zero (see also Section 4.3.4). Lax-

Wendroff fluxes for $F_H$ are inserted in (6.26) and are defined by

$$F_H(c_i^n, c_{i-1}^n) = uc_{i-1}^n + \frac{1}{2}u(c_i^n - c_{i-1}^n)(1 - u\frac{\Delta t}{\Delta x_i}). \tag{6.31}$$

This procedure is only possible with equal theta for $F_L$ and $F_H$. We know from above that with the different theta approach the numerical scheme is always implicit and because of that a Lax-Wendroff flux correction is not suitable. Moreover, this approach would lead to a numerical scheme with Lax-Wendroff flux correction at the old time level and central discretization at the new time-level, which might be mathematically not a valid numerical method.

From here we get the following results

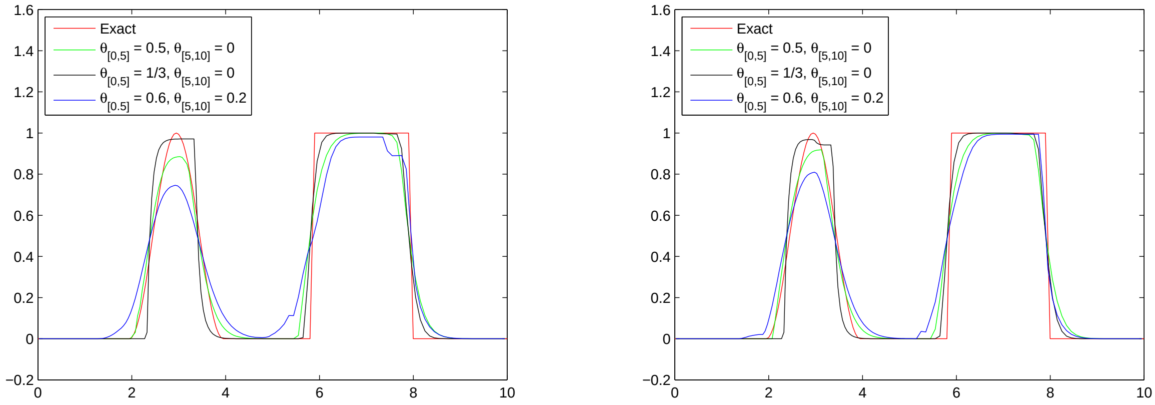

Figure 6.27: Numerical solutions after 1 period by method 1A and 1B respectively. Non-uniform grid with 100 cells on [0,5] and 50 cells on [5,10].

In the figure above we present the results for different CFL values. On the domain [0,5] we have (following the legend in the figures) respectively CFL = 2, CFL = 1.5 and CFL = 2.5, whereas on [5,10] the CFL values are CFL = 1, CFL = 0.75 and CFL = 1.25 respectively. For the numerical solution with $\theta = 0.5$ and $\theta = 0$ we get a smooth solution, since we only have positive numerical diffusion caused by the first-order upwind method. The numerical diffusion caused by $F_H$ is on both domains equal to zero due to the corresponding theta value. For the other two cases in the figure above the smoothness is destroyed by the presence of anti-diffusion (see also (6.29)).

The approach with Lax-Wendroff flux correction for $\theta = 0$ and central flux correction for $\theta > 0$ is only suitable for cell boundaries with $\theta = 0$ or cell boundaries with $\theta \geq 0.5$ since then only numerical diffusion appears in the solution. So we can try the following strategy. For cells with $\theta = 0$ we use Lax-Wendroff fluxes for $F_H$. For cells with $\theta_{i\pm1/2} > 0$ we use central fluxes for $F_H$ and increase the theta value to a minimum of 0.5, i.e.

$$\theta_{i\pm1/2} := \max\{0.5, \theta_{i\pm1/2}\}. \tag{6.32}$$

In this case we only have smearing of the numerical solution. Below we present the results according to this strategy for the situations given in Figure 6.27.

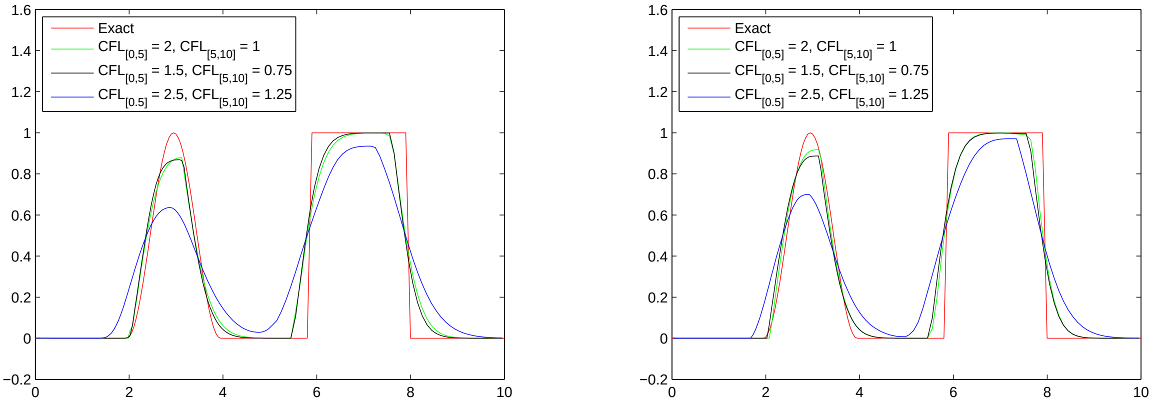

Figure 6.28: Numerical solutions after 1 period by method 1A and 1B respectively. Non-uniform grid with 100 cells on [0,5] and 50 cells on [5,10].

We indeed obtain according to (6.32) smooth solutions. By raising the value of theta to a minimum of 0.5 we get the anti-diffusion replaced by numerical diffusion. For the numerical solution with CFL = 2 and CFL = 1 ($\theta = 0.5$ and $\theta = 0$) we maintain the same result as in Figure 6.27.

The same strategy (6.32) can also be applied with central fluxes for $\theta_{i\pm1/2} = 0$ instead of Lax-Wendroff fluxes. Remember from this section above that we applied central fluxes with different theta for $F_L$ and $F_H$ in order to prevent the presence of anti-diffusion (see also Fig. 6.26). In this new situation we use an equal theta for $F_L$ and $F_H$ and by using the strategy above for cells with $\theta_{i\pm1/2} \in [0, 0.5)$ we have another approach to avoid anti-diffusion. From here we obtain the following results

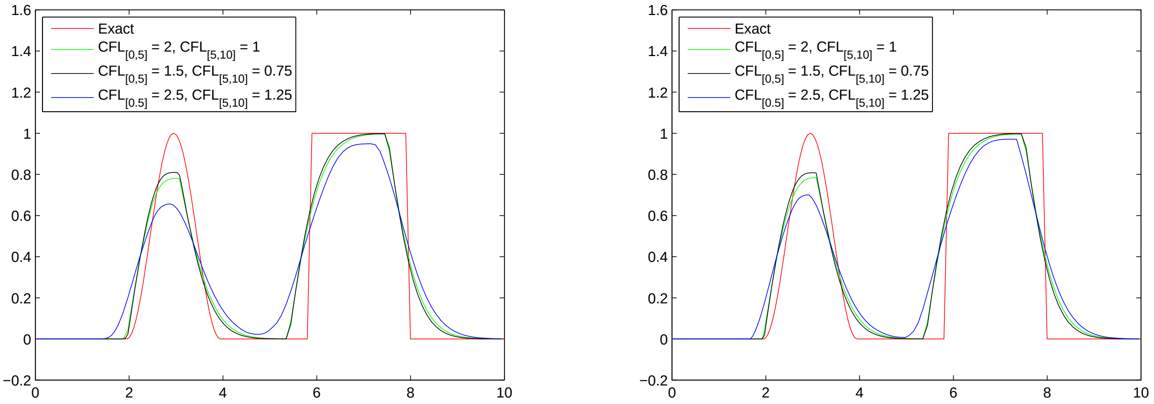

Figure 6.29: Numerical solutions after 1 period by method 1A and 1B respectively. Non-uniform grid with 100 cells on [0,5] and 50 cells on [5,10].

The numerical solutions we obtain are of similar accuracy as the results in Figure 6.28. But the different theta approach leads to higher accuracy as Figure 6.26 demonstrates.

Conclusions

- By using a different theta for $F_L$ and $F_H$ we can avoid anti-diffusion in the numerical solution, but note that the numerical scheme is always implicit, even for $\theta = 0$.
- The different theta approach for the FCT scheme is only useful for $\theta > 0.5$ or CFL $> 2$, since in this case the numerical diffusion is smaller than the numerical diffusion by the approach with equal theta.
- The one-limiter FCT methods can also be applied on non-uniform grids in order to obtain more accuracy.
- By using Lax-Wendroff fluxes instead of central fluxes for $F_H$ if $\theta_{i\pm1/2} = 0$ and raising the theta value to a minimum of 0.5 for cells with $\theta > 0$, we have another approach to obtain accurate results. This strategy of raising the value for theta can also be applied for central fluxes only. Both approaches give similar results.

## 6.2 Two-limiter FCT approach

Below we present the results according to method 2A and method 2B. Remember that the number 2 stands for the two-limiter approach. The difference with the previous section is that flux-limiting is done separately at the old and at the new time level. The results are based on the 1D water quality model (6.1) with periodic boundary conditions given in the beginning of this chapter.

### 6.2.1 Constant theta

For the first results we use a sinusoidal function as initial condition. For the low-order fluxes $F_L$ we take first-order upwind and for the high-order fluxes $F_H$ we take central discretization. Furthermore, we use in the mentioned methods CFL = 2 ($\theta = 0.5$) on a uniform grid. We know that for this CFL number the numerical diffusion caused by central discretization is zero. We get the following results

Table 6.7: The global error of the numerical solutions after 1 period

| *Total grid cells* | $RMSE_{FCT}$<br>1A: old approach | $RMSE_{FCT}$<br>1A: new approach | $RMSE_{FCT}$<br>method 1B | $RMSE_{FCT}$<br>method 2A | $RMSE_{FCT}$<br>method 2B |
|---|---|---|---|---|---|
| 150 | 0.0054 | 0.0032 | 0.0035 | 0.0035 | 0.0049 |
| 300 | 0.0017 | 0.00097 | 0.0011 | 0.0011 | 0.0016 |
| 600 | 0.00052 | 0.00030 | 0.00034 | 0.00033 | 0.00050 |

Besides the two-limiter approaches also the results by the one-limiter methods are given for comparison purposes. Method 2A give results that are similar as method 1A (new approach). For method 2A the alternative approach for $c_i^{\max}$ and $c_i^{\min}$ is used (see also Section 6.1.3). Method 2B seems to be even of similar accuracy as method 1A (old approach), which is the lowest accurate method.

Below we present the numerical results according to method 2 and their corresponding limiters

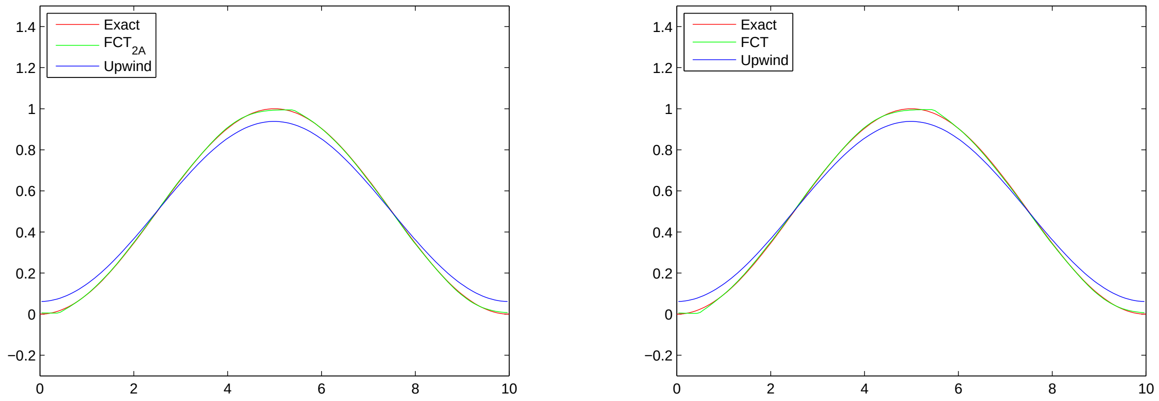

Figure 6.30: Numerical solution after 1 period. Total cells used 150. Left: Method 2A. Right: Method 2B

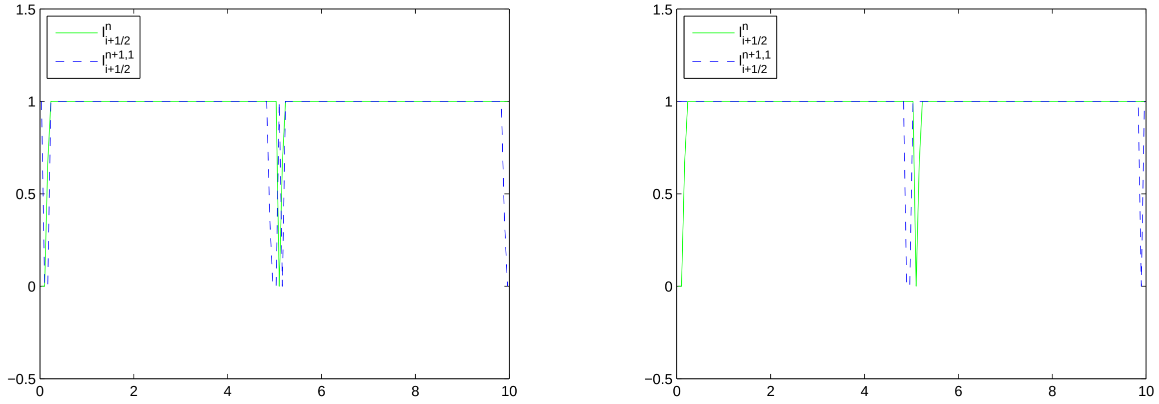

Figure 6.31: The limiter for the first time step. Left: Method 2A. Right: Method 2B

The figures show accurate results with the two-limiter approach. For the limiters at the first time step we notice that its value is equal to 1 except in the neighborhood of extremal points of the solution.

For a block-shaped initial condition under the conditions described above (CFL = 2, etc.) we have the following global errors

Table 6.8: The global error of the numerical solutions after 1 period

| *Total grid cells* | $RMSE_{FCT}$<br>1A: old approach | $RMSE_{FCT}$<br>1A: new approach | $RMSE_{FCT}$<br>method 1B | $RMSE_{FCT}$<br>method 2A | $RMSE_{FCT}$<br>method 2B |
|---|---|---|---|---|---|
| 150 | 0.1317 | 0.1156 | 0.1150 | 0.1153 | 0.1150 |
| 300 | 0.1065 | 0.0936 | 0.0933 | 0.0935 | 0.933 |
| 600 | 0.0857 | 0.0755 | 0.0754 | 0.0754 | 0.0754 |

For this situation we have by method 2A similar results as method 1A (new approach), which we also have seen with the sinusoid. For method 2B we get similar accuracy as method 1B, which is different in comparison with the sinusoidal case. Again the new approach for $c_i^{\max}$ and $c_i^{\min}$ (see Section 6.1.3) is applied to method 2A.

Below we have the numerical results according to method 2 and their corresponding limiters

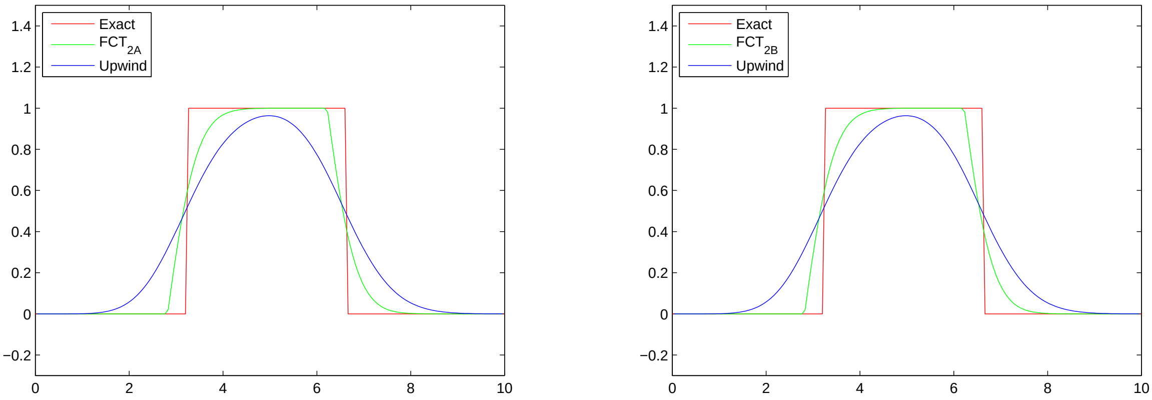

Figure 6.32: Numerical solution after 1 period. Total cells used 150. Left: Method 2A. Right: Method 2B

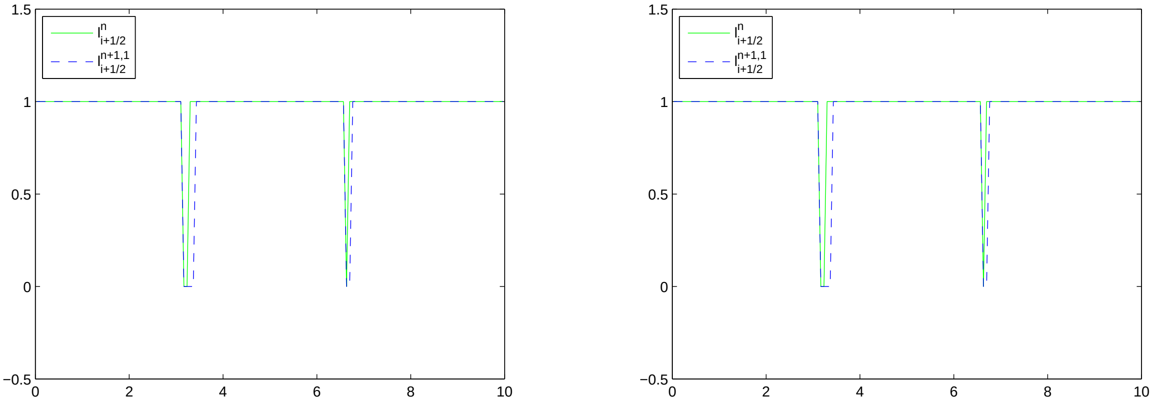

Figure 6.33: The limiter at the first time step. Left: Method 2A. Right: Method 2B

Also for this case we get accurate results with the two-limiter approach. Furthermore the limiter is almost everywhere equal to 1 as expected. Only in the neighborhood of discontinuities we get zero limiter.

Conclusion

The results according to the two-limiter approach are not improved compared with the results by the one-limiter approach. Method 2A is of similar accuracy as method 1A (both with the new approach), whereas method 2B gives at most equal results as method 1B.

In the rest of this section we will apply the methods 2A and 2B for different (but constant) values for theta. The results will be presented for an initial condition containing a sinusoid and a block shape as we have seen in Section 6.1.4. For the high-order flux function $F_H$ we will use central discretization. Again the new approach for $c_i^{\max}$ and $c_i^{\min}$ (see Section 6.1.3) is applied to method 2A. We have the following numerical results

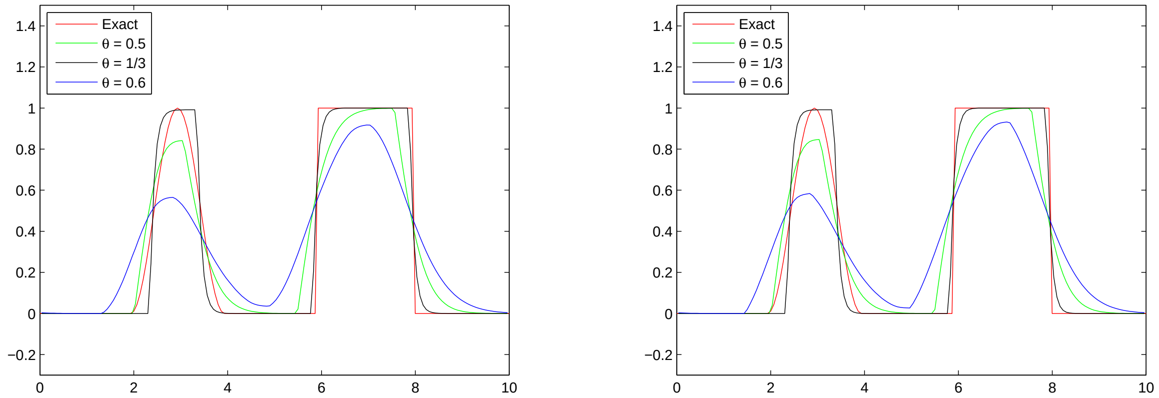

Figure 6.34: Numerical solution after 1 period. Total cells used 150. Left: Method 2A. Right: Method 2B

From the figures above we note that method 2A and 2B give similar results. Furthermore, the numerical solutions above are similar as the solutions by the one-limiter approach (see also Figure 6.23).

### 6.2.2 Variable theta

In this section we continue with the previous example and apply the two-limiter approach discussed in Section 5.2 on a non-uniform grid. On the domain [0,5] we take 100 cells and on the domain [5,10] we take 50 cells. The methods are applied only for $\theta_{i\pm1/2} \geq 0.5$, so that only smearing of the solution takes place, i.e. positive numerical diffusion. This means that for cell boundaries with $\theta_{i\pm1/2} \in [0, 0.5)$ we increase the theta value to a minimum of 0.5 by

$$\theta_{i\pm1/2} := \max\{0.5, \theta_{i\pm1/2}\}. \tag{6.33}$$

If this strategy is not applied, then for the lower values of theta ($\theta_{i\pm1/2} \in [0, 0.5)$) we deal with both numerical diffusion and anti-diffusion which can reduce the accuracy of the numerical solution. With central fluxes for $F_H$ we get the following results

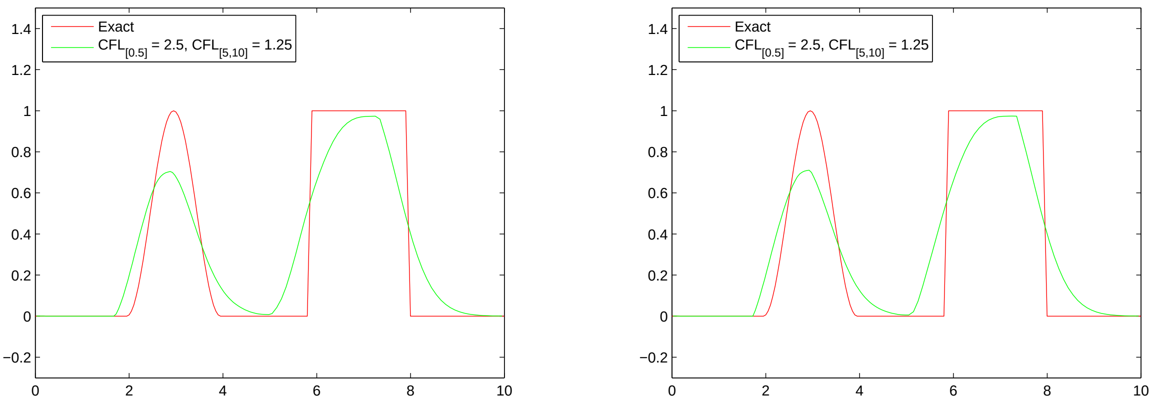

Figure 6.35: Numerical solution after 1 period. Non-uniform grid with 150 cells. Left: Method 2A. Right: Method 2B

From the results above we see that the strategy approach (6.33) leads to reasonable results. An explanation is that a large amount of numerical diffusion is introduced with (6.33). The two-limiter approach gives unfortunately only with CFL $> 2$ for the smallest cell(s) a reasonable accuracy. With lower values for the smallest cell(s) inaccurate results are obtained.

Conclusions

- On a uniform grid we get accurate results by the two-limiter approach. The results are similar as the one-limiter approach.
- On a non-uniform grid inaccurate results are obtained with the two-limiter approach. Only if a CFL $> 2$ is used for the smallest cell(s) then we get a reasonable accuracy.

# Chapter 7

# Conclusions

In this thesis we discussed the water quality model and treated some important numerical methods for solving the corresponding equations, the advection-diffusion-reaction equations. The objective for this thesis was to obtain a robust (accurate and stable) and fast method for solving these equations, where the focus is on the advection part. This because the advection part of the partial differential equation is causing most difficulties for numerical methods.

The Water Quality Model (WQM) is discretized by the Finite Volume Method (FVM) to obtain a discrete model. One particular technique we used for the spatial discretization was the Flux Corrected Transport (FCT) method which is a flux limiter approach. The FCT method is able to reduce sufficiently well the non-physically behavior (wiggles) of the numerical solution near discontinuities caused by steep gradients in the concentration. Furthermore, the WQM model is solved by an implicit time scheme to allow larger time steps in the computation of the solution, especially for the smaller gridcells. In combination with the FCT method one can obtain in an efficient way an accurate numerical solution. Two important properties of the FCT method are that positivity and monotonicity of the solution are preserved. Below we discuss the results that we obtained with an implicit FCT approach applied to the advection equation.

We used a local theta-scheme as time discretization in combination with the FTC method for the spatial discretization. From here a non-linear equation is obtained which is solved by an iterative procedure. We presented two different main approaches, the one-limiter approach and the two-limiter approach. In the former method the limiter is based on the fluxes at the old and new time level together, whereas the latter one applies the limiting of fluxes on both time levels separately. Since the two-limiter approach leads only for a uniform grid to satisfactory results for the numerical solution, the one-limiter approach is used since this one is more accurate. Besides, non-uniform grids are also very important and for this case only the one-limiter approach is appropriate for. Below we discuss the results for only the one-limiter method.

For the computation of the limiter values the FCT method looks at the gradient behavior in the concentration profile in order to apply the proper spatial discretization, i.e. high order method for smooth parts and first order method for parts with steep gradients. From the test results we observed that the ordinary FCT method (method 1A) can not deal in an optimal way with the gradient behavior of the solution, especially at extremal points. Therefore we looked at another possibility to tackle this problem. By including the numerical solution $\underline{c}^n$ at the old time level in the computation of the limiter we noticed that the accuracy is improved compared with the former approach.

Further, we also looked at an approach which uses accumulation (method 1B) in the iteration procedure by adding for each iteration step a flux correction to the intermediate solution $\underline{\tilde{c}}^{n+1}$. Also for this case we obtain higher accuracy than the ordinary FCT approach, since the numerical solution is for each time step corrected as much as possible. This method with accumulation is based on the algorithm of Kuzmin et al. 2004 [6] and is another good alternative to obtain an improved numerical solution.

On a uniform grid with constant theta the accumulated method is similar in accuracy as the improved FCT method mentioned earlier. On a uniform grid with different theta for the high-order flux $F_H$ and for the low-order flux $F_L$ we obtained higher accuracy with the accumulated method than method 1A. The different theta approach for $F_H$ and $F_L$ is introduced in order to reduce the amount of numerical diffusion for $\theta > 0.5$. On a non-uniform grid with the different theta approach we also get higher accuracy with the accumulated FCT method. Using for non-uniform grids equal theta for $F_H$ and $F_L$ is only useful if theta is raised to a minimum of 0.5 since then anti-diffusion is avoided. Though, for this case we get similar accuracy for the accumulated method and the ordinary method. Method 1B and the improved version of method 1A lead both under several circumstances to sufficient accuracy for the numerical solution and are therefore recommended, but method 1B is for non of the cases less than method 1A (improved version).

For the efficiency of the solution methods on the other hand, we know that by raising the value for the CFL larger time steps can be used. But we have seen that the numerical diffusion of the numerical schemes is increasing with the CFL value, hence a decrease of the accuracy. So a faster solution method is only appropriate to a certain extent. An approach for this problem is not to use a relatively large amount of very small cells in the grid-domain, since for these small cells a large CFL value is required for higher efficiency.

# Chapter 8

# Recommendations

In this section we will discuss several cases that can be taken for further research. First we mention one recommendation for the methods used in this thesis and next we present some complex problems for which the FCT method can be applied to.

First of all, the “improved” and “accumulated” FCT methods (one-limiter approach) can be optimized for (especially) smooth functions. In the neighborhood of extremal points the limiter values are unnecessary low, so obtaining more accuracy must be possible.

Further, the implicit FCT scheme must be applied to the two-dimensional advection equation to observe the effect on the numerical solution. Keeping the practical situations in mind, it is important to extend both the "improved" and "accumulated" FCT method to multi-dimensional problems.

Finally, one can include source terms and/or diffusion terms to the advection equation. As mentioned in the second chapter, the WQM includes water quality processes, so it is important to include these source terms in the numerical model. The sources can be either slow or fast processes, but this should not be a problem if the local theta scheme is used.

# Appendix A

# Matlab codes

## A.1 Limiter by Zalesak

```
function [l_rechtsNew, l_linksNew, DeltaF_rechts,DeltaF_links] = ...\\
 limiter(DeltaF_rechts,DeltaF_links,C,Ctilde)
%l_rechtsNew      = limiter at rigth boundary
%l_linksNew       = limiter at left boundary
%DeltaF_rechts    = flux difference at rigth boundary
%DeltaF_links     = flux difference at left boundary
%Ctilde           = intermediate solution
%C                = solution at previous timestep or intermediate solution

%prelimiting
for i = 1:length(C)-1
    if (DeltaF_rechts(i)*(Ctilde(i)-Ctilde(i+1))<=0)
       DeltaF_rechts(i) = 0;
    end
end
if (DeltaF_rechts(length(C))*(Ctilde(length(C))-Ctilde(1))<=0)
    DeltaF_rechts(length(C)) = 0;
end
if (DeltaF_links(1)*(Ctilde(1)-Ctilde(length(C)))<=0)
    DeltaF_links(1) = 0;
end
for i = 2:length(C)
    if (DeltaF_links(i)*(Ctilde(i)-Ctilde(i-1))<=0)
        DeltaF_links(i) = 0;
    end
end

cmax = max(C',Ctilde');
cmin = min(C',Ctilde');
cmaxNew(1) = max([cmax(length(C)),cmax(1),cmax(2)]);
cminNew(1) = min([cmin(length(C)),cmin(1),cmin(2)]);
for i = 2:length(C)-1
    cmaxNew(i) = max([cmax(i-1),cmax(i),cmax(i+1)]);
    cminNew(i) = min([cmin(i-1),cmin(i),cmin(i+1)]);
end
cmaxNew(length(C)) = max([cmax(length(C)-1),cmax(length(C)),cmax(1)]);
cminNew(length(C)) = min([cmin(length(C)-1),cmin(length(C)),cmin(1)]);

PplusNew = max(0,DeltaF_links)+max(0,DeltaF_rechts);
PminNew = min(0,DeltaF_rechts)+min(0,DeltaF_links);

QplusNew = cmaxNew-Ctilde;
QminNew = cminNew-Ctilde;

RplusNew = zeros(1,length(C));
RminNew = zeros(1,length(C));
for i = 1:length(C)
    if PplusNew(i) > 0
        RplusNew(i) = min(1,(QplusNew(i)/(PplusNew(i))));
    else
        RplusNew(i) = 1;
    end
    if PminNew(i) < 0
        RminNew(i) = min(1,(QminNew(i)/(PminNew(i))));
    else
        RminNew(i) = 1;
    end
end

l_rechtsNew(1) = (DeltaF_rechts(1)>=0)*min(RplusNew(1),RminNew(2))+ ... \\
(DeltaF_rechts(1)<0)*min(RplusNew(2),RminNew(1));
for i = 2:length(C)-1
    l_rechtsNew(i) = (DeltaF_rechts(i)>=0)*min(RplusNew(i),RminNew(i+1))+ ...\\
(DeltaF_rechts(i)<0)*min(RplusNew(i+1),RminNew(i));
end
l_rechtsNew(length(C)) = (DeltaF_rechts(length(C))>=0)*min(RplusNew(length(C)), ... \\
RminNew(1))+(DeltaF_rechts(length(C))<0)*min(RplusNew(1),RminNew(length(C)));
l_linksNew = circshift(l_rechtsNew,[1 1]);

% disp(l_rechtsNew)
end
```

## A.2 Method 1A

```
% Method 1A for the 1D homogeneous advection equation

clc; clear all; close all

%parameters
np = 151; %number of gridpoints
p = 1; %number of periods later
u = 1; %velocity
nc = np-1; %number of volume cells
nc1 = 100; %number of cells of first half of domain
nc2 = nc-nc1; %number of cells of second half of domain
xeind = 10;
deltax = [0.5*xeind/nc1*ones(nc1,1); 0.5*xeind/nc2*ones(nc2,1)];
dx = zeros(1,np);
for k = 1:np
    dx(k) = sum(deltax(1:k-1));
end
factor = 2.5; %determines cfl-number, for implicit schemes factor >= 1
dt = factor*min(deltax)/u;
cfl = u*dt./deltax;
N_eind = round(p*xeind/(u*dt));
NumIt = 20; %number of iterations per timestep
epsilon = 1e-3;
max_iter = 10;
NORM = 1; %L1-norm
constTheta = 1; % 1 = equal theta, 0 = different theta

%Choice of high-order numerical fluxes
% cen_old = 2, LW_old = 3, cenLW_old = 4, cen_new = 5
fluxes_old = 2;
fluxes_new = 5;

%Determing theta
thetaR = 0*ones(1,nc);
thetaL = 0*ones(1,nc);
thetaR(1) = max([1-deltax(1)/(u*dt),1-deltax(2)/(u*dt),0]);
thetaL(1) = max([1-deltax(1)/(u*dt),1-deltax(nc)/(u*dt),0]);
for i = 2:nc-1
    thetaR(i) = max([1-deltax(i)/(u*dt),1-deltax(i+1)/(u*dt),0]);
    thetaL(i) = max([1-deltax(i)/(u*dt),1-deltax(i-1)/(u*dt),0]);
end
thetaR(end) = max([1-deltax(nc)/(u*dt),1-deltax(1)/(u*dt),0]);
thetaL(end) = max([1-deltax(nc)/(u*dt),1-deltax(nc-1)/(u*dt),0]);

%for Lax-Wendroff fluxes
% for i = 1:nc
%    if thetaR(i) > 0
%        thetaR(i) = max(0.5,thetaR(i));
%    else
%        thetaR(i) = thetaR(i);
%    end
%
%    if thetaL(i) > 0
%        thetaL(i) = max(0.5,thetaL(i));
%    else
%        thetaL(i) = thetaL(i);
%    end
% end

%for central fluxes
for i = 1:nc
   if thetaR(i) >= 0
       thetaR(i) = max(0.5,thetaR(i));
   else
       thetaR(i) = thetaR(i);
   end

   if thetaL(i) >= 0
       thetaL(i) = max(0.5,thetaL(i));
   else
       thetaL(i) = thetaL(i);
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%exact solution
y = zeros(1,np);
for x = (2/5*nc1):(4/5*nc1)
    y(x) = 0.5*(1-cos(pi*x*0.5*xeind/nc1));
end
for x = (nc1+1/5*nc2):(nc1+3/5*nc2)
    y(x) = 1;
end
% for x = 1:np
%    y(x) = 0.5*(1-cos(0.2*pi*(x-1)*xeind/nc));
% end
% for x = 1/3*nc:2/3*nc
%     y(x) = 1;
% end
% plot(dx,y,'red')
% hold on
```

```
%Trapezium integration rule; average concentration
Q = zeros(1,nc);
for x=1:nc
    Q(x)=(y(x)+y(x+1))/2;
end

%upwind discretisation
A1 = diag(-thetaR'.*u./deltax.*ones(nc,1));
A2 = diag(thetaL(2:end)'.*u./deltax(2:end).*ones(nc-1,1),-1);
A2(1,end) = thetaL(1)*u./deltax(1);
K_L = A1+A2;

A3 = diag(-(1-thetaR)'.*u./deltax.*ones(nc,1));
A4 = diag((1-thetaL(2:end)').*u./deltax(2:end).*ones(nc-1,1),-1);
A4(1,end) = (1-thetaL(1))*u./deltax(1);
K_L_2 = A3+A4;


%FCT by Zalesak
C = zeros(N_eind+1,nc);
FL_rechts = C;
FL_links = C;
FH_rechts = C;
FH_links = C;
DeltaF_rechts = C;
DeltaF_links = C;
us_old = C;
NumIter = zeros(1,N_eind);


C(1,:) = Q; %IC
for n = 1:N_eind

    %upwind
    FL_rechts(n,1) = u*C(n,1);
    FL_links(n,1) = u*C(n,end);
    for i = 2:nc
        FL_rechts(n,i) = u*C(n,i); %i-th volume cell
        FL_links(n,i) = u*C(n,i-1); %(i-1)-th volume cell
    end
    us_old(n+1,:) = C(n,:) - dt./deltax'.*((1-thetaR).*FL_rechts(n,:)- ... \\
(1-thetaL).*FL_links(n,:));

    %central discretization
    if fluxes_old == 2
        FH_rechts(n,1) = 0.5*u*(C(n,1)+C(n,2));
        FH_links(n,1) = 0.5*u*(C(n,end)+C(n,1));
        for i = 2:nc-1
            FH_rechts(n,i) = 0.5*u*(C(n,i)+C(n,i+1));
            FH_links(n,i) = 0.5*u*(C(n,i-1)+C(n,i));
        end
        FH_rechts(n,end) = 0.5*u*(C(n,end)+C(n,1));
        FH_links(n,end) = 0.5*u*(C(n,end-1)+C(n,end));

    %explicit Lax-Wendroff
    elseif fluxes_old == 3
        FH_rechts(n,:) = u*C(n,:)+0.5*u*(C(n,i+1)-C(n,i))*(1-cfl(i)); %i-th cell
        FH_links(n,:) = u*C(n,i-1)+0.5*u*(C(n,i)-C(n,i-1))*(1-cfl(i)); %i-th cell


        FH_rechts(n,1) = u*C(n,1)+0.5*u*(C(n,2)-C(n,1))*(1-cfl(1));
        FH_links(n,1) = u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*(1-cfl(1));
        for i = 2:nc-1
            FH_rechts(n,i) = u*C(n,i)+0.5*u*(C(n,i+1)-C(n,i))*(1-cfl(i)); %i-th cell
            FH_links(n,i) = u*C(n,i-1)+0.5*u*(C(n,i)-C(n,i-1))*(1-cfl(i)); %i-th cell
        end
        FH_rechts(n,end) = u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*(1-cfl(end));
        FH_links(n,end) = u*C(n,end-1)+0.5*u*(C(n,end)-C(n,end-1))*(1-cfl(end));

    %LW or Central
    elseif fluxes_old == 4
        FH_rechts(n,1) = (thetaR(1)==0)*(u*C(n,1)+0.5*u*(C(n,2)-C(n,1))*(1-cfl(1)))+ ... \\
(thetaR(1)~=0)*(0.5*u*(C(n,1)+C(n,2)));
        FH_links(n,1) = (thetaL(1)==0)*(u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*(1-cfl(1)))+ ...\\
(thetaL(1)~=0)*(0.5*u*(C(n,end)+C(n,1)));
        for i = 2:nc-1
            FH_rechts(n,i) = (thetaR(i)==0)*(u*C(n,i)+0.5*u*(C(n,i+1)-C(n,i))* ...\\
(1-cfl(i)))+(thetaR(i)~=0)*(0.5*u*(C(n,i)+C(n,i+1))); %i-th cell
            FH_links(n,i) = (thetaL(i)==0)*(u*C(n,i-1)+0.5*u*(C(n,i)-C(n,i-1))* ...\\
(1-cfl(i)))+(thetaL(i)~=0)*(0.5*u*(C(n,i-1)+C(n,i))); %i-th cell
        end
        FH_rechts(n,end) = (thetaR(end)==0)*(u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*...\\
(1-cfl(end)))+(thetaR(end)~=0)*(0.5*u*(C(n,end)+C(n,1)));
        FH_links(n,end) = (thetaL(end)==0)*(u*C(n,end-1)+0.5*u*(C(n,end)-C(n,end-1))*...\\
(1-cfl(end)))+(thetaL(end)~=0)*0.5*u*(C(n,end-1)+C(n,end));

    end


    DeltaF_rechts(n,:) = FH_rechts(n,:) - FL_rechts(n,:);
    DeltaF_links(n,:) = FH_links(n,:) - FL_links(n,:);

    C_aprox(n+1,:) = (eye(nc)-dt.*K_L)\us_old(n+1,:)’;


%%%%%%%%%%%%%%%%%%%%%%%Iteration procedure%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

m = 1;
```

```
CH = zeros(max_iter+1,nc);
CH(1,:) = C(n,:); %startoplossing iteratieproces
us(1,:) = us_old(n+1,:); %circshift(Q,[1 2*(n-1)+1]); %RHS
dc(1,:) = ones(1,nc);
while (norm(dc(m,:),NORM) > epsilon) && (m<=max_iter) %m<=k

    %upwind
    FL_rechtsNew(m,1) = u*CH(m,1);
    FL_linksNew(m,1) = u*CH(m,end);
    for i = 2:nc
        FL_rechtsNew(m,i) = u*CH(m,i); %i-th volume cell
        FL_linksNew(m,i) = u*CH(m,i-1); %(i-1)-th volume cell
    end

    if fluxes_new == 5
        %central discretization
        FH_rechtsNew(m,1) = 0.5*u*(CH(m,1)+CH(m,2));
        FH_linksNew(m,1) = 0.5*u*(CH(m,nc)+CH(m,1));
        for i = 2:nc-1
            FH_rechtsNew(m,i) = 0.5*u*(CH(m,i)+CH(m,i+1));
            FH_linksNew(m,i) = 0.5*u*(CH(m,i-1)+CH(m,i));
        end
        FH_rechtsNew(m,nc) = 0.5*u*(CH(m,nc)+CH(m,1));
        FH_linksNew(m,nc) = 0.5*u*(CH(m,nc-1)+CH(m,nc));
    end


    if constTheta == 1
        %Equal theta
        DeltaF_rechtsNew(m,:) = -dt./deltax'.*(thetaR.*(FH_rechtsNew(m,:) - ...\\
 FL_rechtsNew(m,:))+(1-thetaR).*DeltaF_rechts(n,:));
        DeltaF_linksNew(m,:) = dt./deltax'.*(thetaL.*(FH_linksNew(m,:) - ...\\
 FL_linksNew(m,:))+(1-thetaL).*DeltaF_links(n,:));
    else
        %Different theta
        DeltaF_rechtsNew(m,:) = -dt./deltax'.*((0.5*FH_rechtsNew(m,:) - ...\\
 thetaR.*FL_rechtsNew(m,:))+0.5*FH_rechts(n,:) - (1-thetaR).*FL_rechts(n,:));
        DeltaF_linksNew(m,:) = dt./deltax'.*((0.5*FH_linksNew(m,:) - ... \\
thetaL.*FL_linksNew(m,:))+(0.5*FH_links(n,:) - (1-thetaL).*FL_links(n,:)));
    end

    deltaf_rechts(m,:) = DeltaF_rechtsNew(m,:);
    deltaf_links(m,:) = DeltaF_linksNew(m,:);


    %FCT fluxlimiter
    [l_rechtsNew(m,:), l_linksNew(m,:), deltaf_rechts(m,:),deltaf_links(m,:)] = ...\\
limiter(deltaf_rechts(m,:),deltaf_links(m,:),C(n,:),us(m,:));
```

```
    limF_rechtsNew(m,:) = l_rechtsNew(m,:).*deltaf_rechts(m,:);
    limF_linksNew(m,:) = l_linksNew(m,:).*deltaf_links(m,:);


    us(m+1,:) = us(m,:);
    if m == 1
        CH(2,:) = (eye(nc)-dt.*K_L)\(us(1,:)+limF_rechtsNew(1,:)+limF_linksNew(1,:))';
    else
        if constTheta == 1
            CH(m+1,:) = (max([1-deltax./(u*dt), zeros(nc,1)],[],2) > 0)'.* ...\\
((eye(nc)-dt.*K_L)\(us(m,:)+limF_rechtsNew(m,:)+limF_linksNew(m,:))')' + ...\\
 (max([1-deltax./(u*dt), zeros(nc,1)],[],2) == 0)'.*CH(2,:);
        else
            CH(m+1,:) = (eye(nc)-dt.*K_L)\(us(m,:)+limF_rechtsNew(m,:)+limF_linksNew(m,:))';
        end
    end
    dc(m+1,:) = CH(m+1,:)-CH(m,:);


    %Iteration error
    It_err(m,n) = norm(dc(m+1,:),NORM);



    m = m+1;


    if sum(thetaL+thetaR)== 0
        break
    end

end

C(n+1,:) = CH(m,:);

%number of iterations per time step
NumIter(n) = m-1;

end

plot(dx(2:end)'-0.5*deltax,C(n+1,:),'- b'); %numerical solution
hold on
axis([0 10 -0.2 1.6])
```

## A.3 Method 1B

```
% Method 1B for the 1D homogeneous advection equation

clc; clear all; close all

%parameters
np = 151; %number of gridpoints
p = 1; %number of periods later
u = 1; %velocity
nc = np-1; %number of volume cells
nc1 = 75; %number of cells of first half of domain
nc2 = nc-nc1; %number of cells of second half of domain
xeind = 10;
deltax = [0.5*xeind/nc1*ones(nc1,1); 0.5*xeind/nc2*ones(nc2,1)];
dx = zeros(1,np);
for k = 1:np
    dx(k) = sum(deltax(1:k-1));
end
factor = 2; %determines cfl-number, for implicit schemes factor >= 1
dt = factor*min(deltax)/u;
cfl = u*dt./deltax; %cfl(i)-number <= 1
N_eind = round(p*xeind/(u*dt));
NumIt = 20; %number of iterations per timestep
epsilon = 1e-3;
max_iter = 10;
NORM = 1; %L1-norm
constTheta = 1; % 1 = equal theta, 0 = different theta

%Choice of high-order numerical fluxes
% cen_old = 2, LW_old = 3, cenLW_old = 4, cen_new = 5
fluxes_old = 2;
fluxes_new = 5;

%Determing theta
thetaR = 0*ones(1,nc);
thetaL = 0*ones(1,nc);
thetaR(1) = max([1-deltax(1)/(u*dt),1-deltax(2)/(u*dt),0]);
thetaL(1) = max([1-deltax(1)/(u*dt),1-deltax(nc)/(u*dt),0]);
for i = 2:nc-1
    thetaR(i) = max([1-deltax(i)/(u*dt),1-deltax(i+1)/(u*dt),0]);
    thetaL(i) = max([1-deltax(i)/(u*dt),1-deltax(i-1)/(u*dt),0]);
end
thetaR(end) = max([1-deltax(nc)/(u*dt),1-deltax(1)/(u*dt),0]);
thetaL(end) = max([1-deltax(nc)/(u*dt),1-deltax(nc-1)/(u*dt),0]);

%for Lax-Wendroff fluxes
% for i = 1:nc
%    if thetaR(i) > 0
%        thetaR(i) = max(0.5,thetaR(i));
%    else
%        thetaR(i) = thetaR(i);
%    end
%
%    if thetaL(i) > 0
%        thetaL(i) = max(0.5,thetaL(i));
%    else
%        thetaL(i) = thetaL(i);
%    end
% end

% %for central fluxes
% for i = 1:nc
%    if thetaR(i) >= 0
%        thetaR(i) = max(0.5,thetaR(i));
%    else
%        thetaR(i) = thetaR(i);
%    end
%
%    if thetaL(i) >= 0
%        thetaL(i) = max(0.5,thetaL(i));
%    else
%        thetaL(i) = thetaL(i);
%    end
% end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%exact solution
y = zeros(1,np);
% for x = (2/5*nc1):(4/5*nc1)
%     y(x) = 0.5*(1-cos(pi*x*0.5*xeind/nc1));
% end
% for x = (nc1+1/5*nc2):(nc1+3/5*nc2)
%     y(x) = 1;
% end
for x = 1:np
   y(x) = 0.5*(1-cos(0.2*pi*(x-1)*xeind/nc));
end
% for x = 1/3*nc:2/3*nc
%     y(x) = 1;
% end
plot(dx,y,'red')
hold on
```

```
%Trapezium integration rule; average concentration
Q = zeros(1,nc);
for x=1:nc
    Q(x)=(y(x)+y(x+1))/2;
end


%upwind discretisation
A1 = diag(-thetaR'.*u./deltax.*ones(nc,1));
A2 = diag(thetaL(2:end)'.*u./deltax(2:end).*ones(nc-1,1),-1);
A2(1,end) = thetaL(1)*u./deltax(1);
K_L = A1+A2;

A3 = diag(-(1-thetaR)'.*u./deltax.*ones(nc,1));
A4 = diag((1-thetaL(2:end)').*u./deltax(2:end).*ones(nc-1,1),-1);
A4(1,end) = (1-thetaL(1))*u./deltax(1);
K_L_2 = A3+A4;


%FCT by Zalesak
C = zeros(N_eind+1,nc);
FL_rechts = C;
FL_links = C;
FH_rechts = C;
FH_links = C;
DeltaF_rechts = C;
DeltaF_links = C;
us_old = C;


C(1,:) = Q; %IC
for n = 1:N_eind

    %upwind
    FL_rechts(n,1) = u*C(n,1);
    FL_links(n,1) = u*C(n,end);
    for i = 2:nc
        FL_rechts(n,i) = u*C(n,i); %i-th volume cell
        FL_links(n,i) = u*C(n,i-1); %(i-1)-th volume cell
    end
    us_old(n+1,:) = C(n,:) - dt./deltax'.*((1-thetaR).*FL_rechts(n,:)- ...\\
(1-thetaL).*FL_links(n,:));

    %central discretization
    if fluxes_old == 2
        FH_rechts(n,1) = 0.5*u*(C(n,1)+C(n,2));
        FH_links(n,1) = 0.5*u*(C(n,end)+C(n,1));
        for i = 2:nc-1
```

```
            FH_rechts(n,i) = 0.5*u*(C(n,i)+C(n,i+1));
            FH_links(n,i) = 0.5*u*(C(n,i-1)+C(n,i));
        end
        FH_rechts(n,end) = 0.5*u*(C(n,end)+C(n,1));
        FH_links(n,end) = 0.5*u*(C(n,end-1)+C(n,end));

    %explicit Lax-Wendroff
    elseif fluxes_old == 3
        FH_rechts(n,1) = u*C(n,1)+0.5*u*(C(n,2)-C(n,1))*(1-cfl(1));
        FH_links(n,1) = u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*(1-cfl(1));
        for i = 2:nc-1
            FH_rechts(n,i) = u*C(n,i)+0.5*u*(C(n,i+1)-C(n,i))*(1-cfl(i)); %i-th cell
            FH_links(n,i) = u*C(n,i-1)+0.5*u*(C(n,i)-C(n,i-1))*(1-cfl(i)); %i-th cell
        end
        FH_rechts(n,end) = u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*(1-cfl(end));
        FH_links(n,end) = u*C(n,end-1)+0.5*u*(C(n,end)-C(n,end-1))*(1-cfl(end));

    %LW or central
    elseif fluxes_old == 4
        FH_rechts(n,1) = (thetaR(1)==0)*(u*C(n,1)+0.5*u*(C(n,2)-C(n,1))*(1-cfl(1)))+ ...\\
(thetaR(1)~=0)*(0.5*u*(C(n,1)+C(n,2)));
        FH_links(n,1) = (thetaL(1)==0)*(u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*(1-cfl(1)))+ ...\\
(thetaL(1)~=0)*(0.5*u*(C(n,end)+C(n,1)));
        for i = 2:nc-1
           FH_rechts(n,i) = (thetaR(i)==0)*(u*C(n,i)+0.5*u*(C(n,i+1)-C(n,i))* ...\\
(1-cfl(i)))+(thetaR(i)~=0)*(0.5*u*(C(n,i)+C(n,i+1))); %i-th cell
           FH_links(n,i) = (thetaL(i)==0)*(u*C(n,i-1)+0.5*u*(C(n,i)-C(n,i-1))*...\\
(1-cfl(i)))+(thetaL(i)~=0)*(0.5*u*(C(n,i-1)+C(n,i))); %i-th cell
        end
        FH_rechts(n,end) = (thetaR(end)==0)*(u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*... \\
(1-cfl(end)))+(thetaR(end)~=0)*(0.5*u*(C(n,end)+C(n,1)));
        FH_links(n,end) = (thetaL(end)==0)*(u*C(n,end-1)+0.5*u*(C(n,end)-C(n,end-1))*...\\
(1-cfl(end)))+(thetaL(end)~=0)*0.5*u*(C(n,end-1)+C(n,end));
    end


    DeltaF_rechts(n,:) = FH_rechts(n,:) - FL_rechts(n,:);
    DeltaF_links(n,:) = FH_links(n,:) - FL_links(n,:);

    C_aprox(n+1,:) = (eye(nc)-dt.*K_L)\us_old(n+1,:)';



%%%%%%%%%%%%%%%%%%%%%%%%Iteration procedure%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
g_rechts(1,:) = zeros(1,nc);
g_links(1,:) = zeros(1,nc);


m = 1;
```

```
CH = zeros(max_iter+1,nc);
CH(1,:) = C(n,:);
us(1,:) = us_old(n+1,:); %RHS
dc(1,:) = ones(1,nc);
while (norm(dc(m,:),NORM) > epsilon) && (m<=max_iter) %m<=k

    %upwind
    FL_rechtsNew(m,1) = u*CH(m,1);
    FL_linksNew(m,1) = u*CH(m,end);
    for i = 2:nc
        FL_rechtsNew(m,i) = u*CH(m,i); %i-th volume cell
        FL_linksNew(m,i) = u*CH(m,i-1); %(i-1)-th volume cell
    end

    %central discretization
    if fluxes_new == 5
        FH_rechtsNew(m,1) = 0.5*u*(CH(m,1)+CH(m,2));
        FH_linksNew(m,1) = 0.5*u*(CH(m,nc)+CH(m,1));
        for i = 2:nc-1
            FH_rechtsNew(m,i) = 0.5*u*(CH(m,i)+CH(m,i+1));
            FH_linksNew(m,i) = 0.5*u*(CH(m,i-1)+CH(m,i));
        end
        FH_rechtsNew(m,nc) = 0.5*u*(CH(m,nc)+CH(m,1));
        FH_linksNew(m,nc) = 0.5*u*(CH(m,nc-1)+CH(m,nc));
    end

    if constTheta == 1
        %Equal theta
        DeltaF_rechtsNew(m,:) = -dt./deltax'.*(thetaR.*(FH_rechtsNew(m,:) - ...\\
 FL_rechtsNew(m,:))+(1-thetaR).*DeltaF_rechts(n,:));
        DeltaF_linksNew(m,:) = dt./deltax'.*(thetaL.*(FH_linksNew(m,:) - ...\\
 FL_linksNew(m,:))+(1-thetaL).*DeltaF_links(n,:));
    else
        %Different theta
        DeltaF_rechtsNew(m,:) = -dt./deltax'.*((0.5*FH_rechtsNew(m,:) - ...\\
 thetaR.*FL_rechtsNew(m,:))+(0.5*FH_rechts(n,:) - (1-thetaR).*FL_rechts(n,:)));
        DeltaF_linksNew(m,:) = dt./deltax'.*((0.5*FH_linksNew(m,:) - ...\\
 thetaL.*FL_linksNew(m,:))+(0.5*FH_links(n,:) - (1-thetaL).*FL_links(n,:)));
    end

    deltaf_rechts(m,:) = DeltaF_rechtsNew(m,:)-g_rechts(m,:);
    deltaf_links(m,:) = DeltaF_linksNew(m,:)-g_links(m,:);


    %FCT fluxlimiter
    [l_rechtsNew(m,:), l_linksNew(m,:),deltaf_rechts(m,:),deltaf_links(m,:)] =...\\
limiter(deltaf_rechts(m,:),deltaf_links(m,:),us(m,:),us(m,:));
```

```
    limF_rechtsNew(m,:) = l_rechtsNew(m,:).*deltaf_rechts(m,:);
    limF_linksNew(m,:) = l_linksNew(m,:).*deltaf_links(m,:);


    us(m+1,:) = us(m,:)+limF_rechtsNew(m,:)+limF_linksNew(m,:);
    if m == 1
        CH(2,:) = (eye(nc)-dt.*K_L)\us(2,:)';
    else
        if constTheta == 1
            CH(m+1,:) = (max([1-deltax./(u*dt), zeros(nc,1)],[],2) > 0)'.* ...\\
((eye(nc)-dt.*K_L)\us(m+1,:)')'+(max([1-deltax./(u*dt), zeros(nc,1)],[],2) == 0)'.*CH(2,:);
        else
            CH(m+1,:) = (eye(nc)-dt.*K_L)\us(m+1,:)';
        end
    end
    dc(m+1,:) = CH(m+1,:)-CH(m,:);


    g_rechts(m+1,:) = g_rechts(m,:) + limF_rechtsNew(m,:);
    g_links(m+1,:) = g_links(m,:) + limF_linksNew(m,:);


    %Iteration error
    It_err(m,n) = norm(dc(m+1,:),NORM);



    m = m+1;


    if sum(thetaL+thetaR)== 0
        break
    end

end

C(n+1,:) = CH(m,:);

%number of iterations per time step
NumIter(n) = m-1;

end

plot(dx(2:end)'-0.5*deltax,C(n+1,:),'- b'); %numerical solution
hold on
axis([0 10 -0.2 1.6])
```

## A.4 Method 2A

```
% Method 2A for the 1D homogeneous advection equation

clc; clear all; close all


%parameters
np = 151; %number of gridpoints
p = 1; %number of periods later
u = 1; %velocity
nc = np-1; %number of volume cells
nc1 = 100; %number of cells of first half of domain
nc2 = nc-nc1; %number of cells of second half of domain
xeind = 10;
deltax = [0.5*xeind/nc1*ones(nc1,1); 0.5*xeind/nc2*ones(nc2,1)];
dx = zeros(1,np);
for k = 1:np
    dx(k) = sum(deltax(1:k-1));
end
factor = 2.1; %determines cfl-number, for implicit schemes factor >= 1
dt = factor*min(deltax)/u;
cfl = u*dt./deltax; %cfl(i)-number <= 1
N_eind = round(p*xeind/(u*dt));
NumIt = 20; %number of iterations per timestep
epsilon = 0.001;
max_iter = 10;
NORM = 1; %L1-norm
constTheta = 1; % 1 = equal theta, 0 = different theta

%Choice of high-order numerical fluxes
% cen_old = 2, LW_old = 3, cenLW_old + equal theta = 4, cen_new = 5
fluxes_old = 2;
fluxes_new = 5;

%Determing theta
thetaR = 0*ones(1,nc);
thetaL = 0*ones(1,nc);
thetaR(1) = max([1-deltax(1)/(u*dt),1-deltax(2)/(u*dt),0]);
thetaL(1) = max([1-deltax(1)/(u*dt),1-deltax(nc)/(u*dt),0]);
for i = 2:nc-1
    thetaR(i) = max([1-deltax(i)/(u*dt),1-deltax(i+1)/(u*dt),0]);
    thetaL(i) = max([1-deltax(i)/(u*dt),1-deltax(i-1)/(u*dt),0]);
end
thetaR(end) = max([1-deltax(nc)/(u*dt),1-deltax(1)/(u*dt),0]);
thetaL(end) = max([1-deltax(nc)/(u*dt),1-deltax(nc-1)/(u*dt),0]);

% % for Lax-Wendroff fluxes
% for i = 1:nc
%    if thetaR(i) > 0
%        thetaR(i) = max(0.5,thetaR(i));
%    else
%        thetaR(i) = thetaR(i);
%    end
%
%    if thetaL(i) > 0
%        thetaL(i) = max(0.5,thetaL(i));
%    else
%        thetaL(i) = thetaL(i);
%    end
% end

%for central fluxes
for i = 1:nc
   if thetaR(i) >= 0
       thetaR(i) = max(0.5,thetaR(i));
   else
       thetaR(i) = thetaR(i);
   end

   if thetaL(i) >= 0
       thetaL(i) = max(0.5,thetaL(i));
   else
       thetaL(i) = thetaL(i);
   end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%exact solution
y = zeros(1,np);
for x = (2/5*nc1):(4/5*nc1)
    y(x) = 0.5*(1-cos(pi*x*0.5*xeind/nc1));
end
for x = (nc1+1/5*nc2):(nc1+3/5*nc2)
    y(x) = 1;
end
% for x = 1:np
%    y(x) = 0.5*(1-cos(0.2*pi*(x-1)*xeind/nc));
% end
% for x = 1/3*nc:2/3*nc
%     y(x) = 1;
% end
plot(dx,y,'red')
```

```
hold on


%Trapezium integration rule; average concentration
Q = zeros(1,nc);
for x=1:nc
    Q(x)=(y(x)+y(x+1))/2;
end


%upwind discretisation
A1 = diag(-thetaR'.*u./deltax.*ones(nc,1));
A2 = diag(thetaL(2:end)'.*u./deltax(2:end).*ones(nc-1,1),-1);
A2(1,end) = thetaL(1)*u./deltax(1);
K_L = A1+A2;
l_eig= real(eig((eye(nc)-dt.*K_L)));


A3 = diag(-(1-thetaR)'.*u./deltax.*ones(nc,1));
A4 = diag((1-thetaL(2:end)').*u./deltax(2:end).*ones(nc-1,1),-1);
A4(1,end) = (1-thetaL(1))*u./deltax(1);
K_L_2 = A3+A4;



%FCT by Zalesak
C = zeros(N_eind+1,nc);
FL_rechts = zeros(1,nc);
FL_links = zeros(1,nc);
FH_rechts = zeros(1,nc);
FH_links = zeros(1,nc);
us = zeros(N_eind,nc);
us_old = zeros(N_eind,nc);
FL_rechtsNew = zeros(1,nc);
FL_linksNew = zeros(1,nc);
FH_rechtsNew = zeros(1,nc);
FH_linksNew = zeros(1,nc);


tic
C(1,:) = Q; %IC
% for k = 1:NumIt
for n = 1:N_eind

    %upwind
    FL_rechts(1) = u*C(n,1);
    FL_links(1) = u*C(n,end);
    for i = 2:nc
        FL_rechts(i) = u*C(n,i); %i-th volume cell
        FL_links(i) = u*C(n,i-1); %(i-1)-th volume cell
    end
    us(n+1,:) = C(n,:) - dt./deltax'.*((1-thetaR).*FL_rechts-(1-thetaL).*FL_links);

    %central discretization
    if fluxes_old == 2
        FH_rechts(1) = 0.5*u*(C(n,1)+C(n,2));
        FH_links(1) = 0.5*u*(C(n,end)+C(n,1));
        for i = 2:nc-1
            FH_rechts(i) = 0.5*u*(C(n,i)+C(n,i+1));
            FH_links(i) = 0.5*u*(C(n,i-1)+C(n,i));
        end
        FH_rechts(end) = 0.5*u*(C(n,end)+C(n,1));
        FH_links(end) = 0.5*u*(C(n,end-1)+C(n,end));

    %explicit Lax-Wendroff
    elseif fluxes_old == 3
        FH_rechts(1) = u*C(n,1)+0.5*u*(C(n,2)-C(n,1))*(1-cfl(1));
        FH_links(1) = u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*(1-cfl(1));
        for i = 2:nc-1
            FH_rechts(i) = u*C(n,i)+0.5*u*(C(n,i+1)-C(n,i))*(1-cfl(i)); %i-th cell
            FH_links(i) = u*C(n,i-1)+0.5*u*(C(n,i)-C(n,i-1))*(1-cfl(i)); %i-th cell
        end
        FH_rechts(end) = u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*(1-cfl(end));
        FH_links(end) = u*C(n,end-1)+0.5*u*(C(n,end)-C(n,end-1))*(1-cfl(end));

    %LW or central
    elseif fluxes_old == 4
        FH_rechts(1) = (thetaR(1)==0)*(u*C(n,1)+0.5*u*(C(n,2)-C(n,1))*(1-cfl(1)))+ ...\\
(thetaR(1)~=0)*(0.5*u*(C(n,1)+C(n,2)));
        FH_links(1) = (thetaL(1)==0)*(u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*(1-cfl(1)))+ ...\\
(thetaL(1)~=0)*(0.5*u*(C(n,end)+C(n,1)));
        for i = 2:nc-1
            FH_rechts(i) = (thetaR(i)==0)*(u*C(n,i)+0.5*u*(C(n,i+1)-C(n,i))* ...\\
(1-cfl(i)))+ (thetaR(i)~=0)*(0.5*u*(C(n,i)+C(n,i+1))); %i-th cell
            FH_links(i) = (thetaL(i)==0)*(u*C(n,i-1)+0.5*u*(C(n,i)-C(n,i-1))* ... \\
(1-cfl(i)))+ (thetaL(i)~=0)*(0.5*u*(C(n,i-1)+C(n,i))); %i-th cell
        end
        FH_rechts(end) = (thetaR(end)==0)*(u*C(n,end)+0.5*u*(C(n,1)-C(n,end))* ...\\
(1-cfl(end)))+(thetaR(end)~=0)*(0.5*u*(C(n,end)+C(n,1)));
        FH_links(end) = (thetaL(end)==0)*(u*C(n,end-1)+0.5*u*(C(n,end)-C(n,end-1))*...\\
(1-cfl(end)))+(thetaL(end)~=0)*0.5*u*(C(n,end-1)+C(n,end));
    end

    if constTheta == 1
        %Equal theta
        DeltaF_rechts = - dt./deltax'.*(1-thetaR).*(FH_rechts - FL_rechts);
        DeltaF_links = dt./deltax'.*(1-thetaL).*(FH_links - FL_links);
    else
```

```
        %Different theta
        DeltaF_rechts = - dt./deltax'.*(0.5*FH_rechts - (1-thetaR).*FL_rechts);
        DeltaF_links = dt./deltax'.*(0.5*FH_links - (1-thetaL).*FL_links);
    end


    %FCT fluxlimiter
    [l_rechts, l_links,DeltaF_rechts,DeltaF_links] = ...\\
 limiter(DeltaF_rechts,DeltaF_links,C(n,:),us(n+1,:));


    limF_rechts = l_rechts.*DeltaF_rechts;
    limF_links = l_links.*DeltaF_links;


    us_old(n+1,:) = us(n+1,:)+limF_rechts+limF_links;



%%%%%%%%%%%%%%%%%%%%%%%%Iteration procedure%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

m = 1;
CH(1,:) = C(n,:);
usNew(1,:) = us_old(n+1,:); %RHS
if sum(thetaR+thetaL) == 0
    dc(1,:) = zeros(1,nc);
else
    dc(1,:) = (max([1-deltax./(u*dt), zeros(nc,1)],[],2) > 0)'.*ones(1,nc);
end

while (norm(dc(m,:),NORM) > epsilon) && (m<=max_iter) %m<=k

    %upwind
    FL_rechtsNew(1) = u*CH(m,1);
    FL_linksNew(1) = u*CH(m,end);
    for i = 2:nc
        FL_rechtsNew(i) = u*CH(m,i); %i-th volume cell
        FL_linksNew(i) = u*CH(m,i-1); %(i-1)-th volume cell
    end

    if fluxes_new == 5
        %central discretization
        FH_rechtsNew(1) = 0.5*u*(CH(m,1)+CH(m,2));
        FH_linksNew(1) = 0.5*u*(CH(m,nc)+CH(m,1));
        for i = 2:nc-1
            FH_rechtsNew(i) = 0.5*u*(CH(m,i)+CH(m,i+1));
            FH_linksNew(i) = 0.5*u*(CH(m,i-1)+CH(m,i));
        end
        FH_rechtsNew(nc) = 0.5*u*(CH(m,nc)+CH(m,1));
        FH_linksNew(nc) = 0.5*u*(CH(m,nc-1)+CH(m,nc));
    end

    if constTheta == 1
        %Equal theta
        DeltaF_rechtsNew = -dt./deltax'.*(thetaR.*(FH_rechtsNew - FL_rechtsNew));
        DeltaF_linksNew = dt./deltax'.*(thetaL.*(FH_linksNew - FL_linksNew));
    else
        %Different theta
        DeltaF_rechtsNew = -dt./deltax'.*(0.5*FH_rechtsNew - thetaR.*FL_rechtsNew);
        DeltaF_linksNew = dt./deltax'.*(0.5*FH_linksNew - thetaL.*FL_linksNew);
    end

    deltaf_rechts = DeltaF_rechtsNew;
    deltaf_links = DeltaF_linksNew;


    %FCT fluxlimiter
    [l_rechtsNew, l_linksNew,deltaf_rechts,deltaf_links] = ...\\
 limiter(deltaf_rechts,deltaf_links,C(n,:),usNew(m,:));


    limF_rechtsNew = l_rechtsNew.*deltaf_rechts;
    limF_linksNew = l_linksNew.*deltaf_links;


    usNew(m+1,:) = usNew(m,:);
    CH(m+1,:) = (max([1-deltax./(u*dt), zeros(nc,1)],[],2) > 0)'.* ...\\
((eye(nc)-dt.*K_L)\(usNew(m,:)+limF_rechtsNew+limF_linksNew)')' + ...\\
 (max([1-deltax./(u*dt), zeros(nc,1)],[],2) == 0)'.*us_old(n+1,:);
    dc(m+1,:) = CH(m+1,:)-CH(m,:);


    %Iteration error
    It_err(m,n) = norm(dc(m+1,:),NORM);


    m = m+1;
end

C(n+1,:) = (sum(thetaL+thetaR)~=0)*CH(m,:)+(sum(thetaL+thetaR)==0)*us_old(n+1,:);

%number of iterations per time step
NumIter(n) = m-1;

end

plot(dx(2:end)'-0.5*deltax,C(n+1,:),'- g'); %numerical solution
axis([0 10 -0.3 1.5])
```

## A.5 Method 2B

```
% Method 2B for the 1D homogeneous advection equation

clc; clear all; close all


%parameters
np = 151; %number of gridpoints
p = 1; %number of periods later
u = 1; %velocity
nc = np-1; %number of volume cells
nc1 = 100; %number of cells of first half of domain
nc2 = nc-nc1; %number of cells of second half of domain
xeind = 10;
deltax = [0.5*xeind/nc1*ones(nc1,1); 0.5*xeind/nc2*ones(nc2,1)];
dx = zeros(1,np);
for k = 1:np
    dx(k) = sum(deltax(1:k-1));
end
factor = 2.1; %determines cfl-number, for implicit schemes factor >= 1
dt = factor*min(deltax)/u;
cfl = u*dt./deltax; %cfl(i)-number <= 1
N_eind = round(p*xeind/(u*dt));
NumIt = 20; %number of iterations per timestep
epsilon = 0.001;
max_iter = 10;
NORM = 1; %L1-norm
constTheta = 1; % 1 = equal theta, 0 = different theta

%Choice of high-order numerical fluxes
% cen_old = 2, LW_old = 3, cenLW_old + equal theta = 4, cen_new = 5
fluxes_old = 2;
fluxes_new = 5;


%Determing theta
thetaR = 0*ones(1,nc);
thetaL = 0*ones(1,nc);
thetaR(1) = max([1-deltax(1)/(u*dt),1-deltax(2)/(u*dt),0]);
thetaL(1) = max([1-deltax(1)/(u*dt),1-deltax(nc)/(u*dt),0]);
for i = 2:nc-1
    thetaR(i) = max([1-deltax(i)/(u*dt),1-deltax(i+1)/(u*dt),0]);
    thetaL(i) = max([1-deltax(i)/(u*dt),1-deltax(i-1)/(u*dt),0]);
end
thetaR(end) = max([1-deltax(nc)/(u*dt),1-deltax(1)/(u*dt),0]);
```

```
thetaL(end) = max([1-deltax(nc)/(u*dt),1-deltax(nc-1)/(u*dt),0]);

% for Lax-Wendroff fluxes
% for i = 1:nc
%    if thetaR(i) > 0
%        thetaR(i) = max(0.5,thetaR(i));
%    else
%        thetaR(i) = thetaR(i);
%    end
%
%    if thetaL(i) > 0
%        thetaL(i) = max(0.5,thetaL(i));
%    else
%        thetaL(i) = thetaL(i);
%    end
% end

%for central fluxes
for i = 1:nc
   if thetaR(i) >= 0
       thetaR(i) = max(0.5,thetaR(i));
   else
       thetaR(i) = thetaR(i);
   end

   if thetaL(i) >= 0
       thetaL(i) = max(0.5,thetaL(i));
   else
       thetaL(i) = thetaL(i);
   end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%exact solution
y = zeros(1,np);
for x = (2/5*nc1):(4/5*nc1)
    y(x) = 0.5*(1-cos(pi*x*0.5*xeind/nc1));
end
for x = (nc1+1/5*nc2):(nc1+3/5*nc2)
    y(x) = 1;
end
% for x = 1:np
%    y(x) = 0.5*(1-cos(0.2*pi*(x-1)*xeind/nc));
% end
% for x = 1/3*nc:2/3*nc
%     y(x) = 1;
% end
plot(dx,y,'red')
hold on


%Trapezium integration rule; average concentration
Q = zeros(1,nc);
for x=1:nc
    Q(x)=(y(x)+y(x+1))/2;
end


%upwind discretisation
A1 = diag(-thetaR'.*u./deltax.*ones(nc,1));
A2 = diag(thetaL(2:end)'.*u./deltax(2:end).*ones(nc-1,1),-1);
A2(1,end) = thetaL(1)*u./deltax(1);
K_L = A1+A2;
l_eig= real(eig((eye(nc)-dt.*K_L)));


A3 = diag(-(1-thetaR)'.*u./deltax.*ones(nc,1));
A4 = diag((1-thetaL(2:end)').*u./deltax(2:end).*ones(nc-1,1),-1);
A4(1,end) = (1-thetaL(1))*u./deltax(1);
K_L_2 = A3+A4;



%FCT by Zalesak
C = zeros(N_eind+1,nc);
FL_rechts = zeros(1,nc);
FL_links = zeros(1,nc);
FH_rechts = zeros(1,nc);
FH_links = zeros(1,nc);
us = zeros(N_eind,nc);
us_old = zeros(N_eind,nc);
FL_rechtsNew = zeros(1,nc);
FL_linksNew = zeros(1,nc);
FH_rechtsNew = zeros(1,nc);
FH_linksNew = zeros(1,nc);


tic
C(1,:) = Q; %IC
% for k = 1:NumIt
for n = 1:N_eind

    %upwind
    FL_rechts(1) = u*C(n,1);
    FL_links(1) = u*C(n,end);
    for i = 2:nc
        FL_rechts(i) = u*C(n,i); %i-th volume cell
        FL_links(i) = u*C(n,i-1); %(i-1)-th volume cell
    end
    us(n+1,:) = C(n,:) - dt./deltax'.*((1-thetaR).*FL_rechts-(1-thetaL).*FL_links);

    %central discretization
    if fluxes_old == 2
        FH_rechts(1) = 0.5*u*(C(n,1)+C(n,2));
        FH_links(1) = 0.5*u*(C(n,end)+C(n,1));
        for i = 2:nc-1
            FH_rechts(i) = 0.5*u*(C(n,i)+C(n,i+1));
            FH_links(i) = 0.5*u*(C(n,i-1)+C(n,i));
        end
        FH_rechts(end) = 0.5*u*(C(n,end)+C(n,1));
        FH_links(end) = 0.5*u*(C(n,end-1)+C(n,end));

    %explicit Lax-Wendroff
    elseif fluxes_old == 3
        FH_rechts(1) = u*C(n,1)+0.5*u*(C(n,2)-C(n,1))*(1-cfl(1));
        FH_links(1) = u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*(1-cfl(1));
        for i = 2:nc-1
            FH_rechts(i) = u*C(n,i)+0.5*u*(C(n,i+1)-C(n,i))*(1-cfl(i)); %i-th cell
            FH_links(i) = u*C(n,i-1)+0.5*u*(C(n,i)-C(n,i-1))*(1-cfl(i)); %i-th cell
        end
        FH_rechts(end) = u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*(1-cfl(end));
        FH_links(end) = u*C(n,end-1)+0.5*u*(C(n,end)-C(n,end-1))*(1-cfl(end));

    %LW or central
    elseif fluxes_old == 4
        FH_rechts(1) = (thetaR(1)==0)*(u*C(n,1)+0.5*u*(C(n,2)-C(n,1))*(1-cfl(1)))+ ...\\
(thetaR(1)~=0)*(0.5*u*(C(n,1)+C(n,2)));
        FH_links(1) = (thetaL(1)==0)*(u*C(n,end)+0.5*u*(C(n,1)-C(n,end))* ...\\
(1-cfl(1)))+(thetaL(1)~=0)*(0.5*u*(C(n,end)+C(n,1)));
        for i = 2:nc-1
            FH_rechts(i) = (thetaR(i)==0)*(u*C(n,i)+0.5*u*(C(n,i+1)-C(n,i))*...\\
(1-cfl(i)))+(thetaR(i)~=0)*(0.5*u*(C(n,i)+C(n,i+1))); %i-th cell
            FH_links(i) = (thetaL(i)==0)*(u*C(n,i-1)+0.5*u*(C(n,i)-C(n,i-1))*...\\
(1-cfl(i)))+(thetaL(i)~=0)*(0.5*u*(C(n,i-1)+C(n,i))); %i-th cell
        end
        FH_rechts(end) = (thetaR(end)==0)*(u*C(n,end)+0.5*u*(C(n,1)-C(n,end))*...\\
(1-cfl(end)))+(thetaR(end)~=0)*(0.5*u*(C(n,end)+C(n,1)));
        FH_links(end) = (thetaL(end)==0)*(u*C(n,end-1)+0.5*u*(C(n,end)-C(n,end-1))*...\\
(1-cfl(end)))+ (thetaL(end)~=0)*0.5*u*(C(n,end-1)+C(n,end));
    end


    if constTheta == 1
        %Equal theta
        DeltaF_rechts = - dt./deltax'.*(1-thetaR).*(FH_rechts - FL_rechts);
        DeltaF_links = dt./deltax'.*(1-thetaL).*(FH_links - FL_links);
    else
        %Different theta
        DeltaF_rechts = - dt./deltax'.*(0.5*FH_rechts - (1-thetaR).*FL_rechts);
        DeltaF_links = dt./deltax'.*(0.5*FH_links - (1-thetaL).*FL_links);
    end


    %FCT fluxlimiter
    [l_rechts, l_links,DeltaF_rechts,DeltaF_links] = ...\\
 limiter(DeltaF_rechts,DeltaF_links,us(n+1,:),us(n+1,:));


    limF_rechts = l_rechts.*DeltaF_rechts;
    limF_links = l_links.*DeltaF_links;

    us_old(n+1,:) = us(n+1,:)+limF_rechts+limF_links;



%%%%%%%%%%%%%%%%%%%%%%%%Iteration procedure%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

m = 1;
CH(1,:) = C(n,:);
usNew(1,:) = us_old(n+1,:); %RHS
if sum(thetaR+thetaL) == 0
    dc(1,:) = zeros(1,nc);
else
    dc(1,:) = (max([1-deltax./(u*dt), zeros(nc,1)],[],2) > 0)'.*ones(1,nc);
end
g_rechts = zeros(1,nc);
g_links = zeros(1,nc);

while (norm(dc(m,:),NORM) > epsilon) && (m<=max_iter) %m<=k

    %upwind
    FL_rechtsNew(1) = u*CH(m,1);
    FL_linksNew(1) = u*CH(m,end);
    for i = 2:nc
        FL_rechtsNew(i) = u*CH(m,i); %i-th volume cell
        FL_linksNew(i) = u*CH(m,i-1); %(i-1)-th volume cell
    end

    %central discretization
    if fluxes_new == 5
        FH_rechtsNew(1) = 0.5*u*(CH(m,1)+CH(m,2));
        FH_linksNew(1) = 0.5*u*(CH(m,nc)+CH(m,1));
        for i = 2:nc-1
```

```
            FH_rechtsNew(i) = 0.5*u*(CH(m,i)+CH(m,i+1));
            FH_linksNew(i) = 0.5*u*(CH(m,i-1)+CH(m,i));
        end
        FH_rechtsNew(nc) = 0.5*u*(CH(m,nc)+CH(m,1));
        FH_linksNew(nc) = 0.5*u*(CH(m,nc-1)+CH(m,nc));
    end


    if constTheta == 1
        %Equal theta
        DeltaF_rechtsNew = -dt./deltax'.*(thetaR.*(FH_rechtsNew - FL_rechtsNew));
        DeltaF_linksNew = dt./deltax'.*(thetaL.*(FH_linksNew - FL_linksNew));
    else
        %Different theta
        DeltaF_rechtsNew = -dt./deltax'.*(0.5*FH_rechtsNew - thetaR.*FL_rechtsNew);
        DeltaF_linksNew = dt./deltax'.*(0.5*FH_linksNew - thetaL.*FL_linksNew);
    end

    deltaf_rechts = DeltaF_rechtsNew-g_rechts;
    deltaf_links = DeltaF_linksNew-g_links;


    %FCT fluxlimiter
    [l_rechtsNew, l_linksNew,deltaf_rechts,deltaf_links] = ... \\
 limiter(deltaf_rechts,deltaf_links,usNew(m,:),usNew(m,:));


    limF_rechtsNew = l_rechtsNew.*deltaf_rechts;
    limF_linksNew = l_linksNew.*deltaf_links;


    usNew(m+1,:) = usNew(m,:)+limF_rechtsNew+limF_linksNew;
    CH(m+1,:) = (max([1-deltax./(u*dt), zeros(nc,1)],[],2) > 0)'.* ...\\
((eye(nc)-dt.*K_L)\usNew(m+1,:)')' + ...\\
 (max([1-deltax./(u*dt), zeros(nc,1)],[],2) == 0)'.*us_old(n+1,:);
    dc(m+1,:) = CH(m+1,:)-CH(m,:);


    g_rechts = g_rechts + limF_rechtsNew;
    g_links = g_links + limF_linksNew;


    %Iteration error
    It_err(m,n) = norm(dc(m+1,:),NORM);


    m = m+1;
end
```

```
C(n+1,:) = (sum(thetaL+thetaR)~=0)*CH(m,:)+(sum(thetaL+thetaR)==0)*us_old(n+1,:);

%number of iterations per time step
NumIter(n) = m-1;

end

plot(dx(2:end)'-0.5*deltax,C(n+1,:),'- g'); %numerical solution
axis([0 10 -0.3 1.5])
```

# Appendix A

# Current numerical schemes in Delft3D-WAQ

At present, 23 different numerical schemes can be used in Delft3D-WAQ. The most used schemes are briefly presented below.

**Scheme 1** The explicit first order upwind scheme.

**Scheme 2** Like scheme 1, except that it uses the predictor corrector method for time integration.

**Scheme 3** The explicit Lax-Wendroff scheme.

**Scheme 4** An Alternation Direction Implicit (ADI) method. It can only be applied in two dimensions on a structured grid. This scheme uses the theta scheme for $\theta = \frac{1}{2}$.

**Scheme 5** An explicit FCT scheme a la Boris and Book with Lax-Wendroff flux correction.

**Scheme 10** Theta upwind scheme with $\theta = 1$.

**Scheme 11** The horizontal and vertical direction are treated separately. In horizontal direction the explicit upwind scheme. In vertical direction the theta scheme with $\theta = \frac{1}{2}$ and central fluxes.

**Scheme 12** Like scheme 11, except that it uses an explicit FCT scheme (scheme 5) in the horizontal direction.

**Scheme 13** Like scheme 11, except that it uses the theta upwind scheme with $\theta = 1$ in the vertical direction.

**Scheme 14** Like scheme 12, except that it uses the theta upwind scheme with $\theta = 1$ in the vertical direction.

**Scheme 15** Like scheme 10, except that in horizontal direction the linear systems are solved by means of GMRES with a symmetric GS preconditioner. In the vertical direction a direct method is used.

**Scheme 16** Like scheme 15, except that it uses the theta scheme with $\theta = \frac{1}{2}$ and central discretization in the in the vertical direction.

**Scheme 19** The horizontal and vertical direction are treated separately. In the horizontal direction an ADI method is used. In the vertical direction central fluxes are used.

**Scheme 20** Like scheme 19, except that it uses first order upwind discretization in the vertical direction.

**Scheme 21 – 22** The local theta scheme combined with the FCT scheme a la Boris and Book.

**Scheme 23** DHI Quickest solver.

# Bibliography

[1] K. Alhumaizi, *Flux limiting solution techniques for simulation of reaction − diffusion− convection system*, Communications in Nonlinear Science and Numerical Simulation, volume 12, page 953-965, 2007

[2] J.P. Boris and D.L. Book, *Flux Corrected Transport, I. SHASTA, A fluid transport algorithm that works*, Journal of Computational Physics volume 11, page 38-69, 1973

[3] Delft3D-WAQ user manual, *Versatile water quality modelling in 1D, 2D or 3D systems including physical, (bio)chemical and biological processes*, Deltares, Delft, 2009

[4] R. Eymard, T. Gallouet and R. Herbin, *Finite Volume Methods*, page 4-11, 2006

[5] D. Kuzmin, *Explicit and implicit FEM − FCT algorithms with flux linearization*, Journal of Computational Physics 228, page 2517-2534, 2009

[6] D. Kuzmin, M. Möller, S. Turek, *High − resolution FEM − FCT schemes for multidimensional conservation laws*, Technical Report 2011, Institute of Applied Mathematics, University of Dortmund, 2004.

[7] D. Kuzmin, S. Turek, *Flux correction tools for finite elements*, Journal of Computational Physics 175, page 525-558, 2002.

[8] R.J. Leveque, *Finite Volume Methods for hyperbolic problems*, Cambridge University Press, New York, 2002.

[9] L. Postma, *Water quality of surface waters*, Technical report, Deltares, Delft, 2011

[10] P. van Slingerland, *An accurate and robust finite volume method for the advection diffusion equation*, Delft Institute of Applied Mathematics, June 2007.

[11] S. van Veldhuizen, *Efficient numerical methods for the instationary solution of laminar reacting gas flow problems*, Delft Centre for Computational Science and Engineering, 2009.

[12] C. Vuik, P. van Beek, F. Vermolen, J. van Kan, *Numerieke methoden voor differentiaal vergelijkingen*, VSSD, Delft, 2006

[13] P. Wesseling, *Elements of computational fluid dynamics*, Delft University of Technology, 2001

[14] S.T. Zalesak, *Fully multidimensional flux corrected transport algorithm for fluids*, Journal of Computational Physics volume 31, page 335-362, 1979