Adaptive non-oscillatory complete flux schemes
F. Wessels

29th June 2015

Supervised by:
dr. ir. J. H. M. ten Thije Boonkkamp
Abstract

The complete flux scheme, a numerical method for solving advection-diffusion-reaction types of equations is considered. This finite volume method calculates the numerical flux using all equation variables and parameters. For time dependent equations the time derivative is optionally used in calculating the numerical flux. When the time derivative is included you get the transient complete flux scheme, if not you get the stationary complete flux scheme. The theta-method is used for time integration. With examples it is shown that for advection dominant equations the stationary complete flux scheme is first order while the transient complete flux scheme is second order accurate but can cause spurious oscillations. Both schemes are combined using a parameter for which bounds are found, both independent as well as dependent on the smoothness of the solution, such that the scheme has the TVD property when applied to the advection equation. Examples are given which show that also for the general equation the derived bounds are applicable. A basic grid refinement method is introduced which is combined with the hybrid scheme. This method is based on recursion and refines only where needed. Results of all schemes and of the grid refinement method are compared.
# Contents

1 Introduction .................................................. 2

2 Finite Volume-Complete Flux scheme ................. 5
   2.1 Finite Volume Method ................................... 5
   2.2 Finite Volume-Complete Flux scheme in one
       dimension ............................................. 5
   2.3 Limiting cases ........................................ 11
   2.4 Time integration ..................................... 12
   2.5 Order of convergence in time and space .......... 13

3 TVD restrictions ............................................... 18
   3.1 Restrictions on combined TCF/SCF-scheme for
       $\varepsilon = 0$ (advection equation) ............... 18
   3.2 TVD restrictions combined TCF/SCF-scheme,
       constant coefficients .............................. 19

4 TVD restrictions depending on the smoothness. .... 26
   4.1 Advection equation, explicit restrictions .......... 27
   4.2 Advection equation, implicit restrictions .......... 29
   4.3 Results .............................................. 32

5 Adaptive grid refinement ................................. 35
   5.1 Basic method ......................................... 35
   5.2 Combined TCF/SCF solver using flux limiter ......... 36
   5.3 Function $r$ .......................................... 38
   5.4 Results .............................................. 41

6 Conclusions and discussion ............................. 46

A Matlab code ................................................ 48
   A.1 Solver ................................................. 48
   A.2 Function $r$ .......................................... 50
Conservation laws appear everywhere in continuum physics, they occur in combustion theory, plasma physics, fluid mechanics etc. We will consider the advection-diffusion-reaction type of conservation law which describes the interplay of three different processes. First there is diffusion/conduction which is the net movement of a substance (e.g., an atom, ion or molecule) from a region of high concentration to a region of low concentration. Then there is advection which is a transport mechanism. The reaction term can describe chemical reactions or recombination/generation. Examples are the conservation equations for reaction flow [6] and the drift-diffusion equations for semiconductor devices [3], [7].

The advection-diffusion-reaction equation is solved with numerical discretisation methods. There are many different methods for space discretisation available, such as finite element, finite difference, finite volume or spectral methods. We have chosen to consider complete flux schemes, which are finite volume methods where an approximation for the flux is derived by solving a local boundary value problem (BVP) for the entire equation, including the source term and for time dependent problems optionally including the time derivative. When the time derivative is included we will refer to it as the transient complete flux scheme (TCF), else the scheme is referred to as the stationary complete flux scheme (SCF). These schemes are described by J.H.M. ten Thije Boonkkamp and M.J.H. Anthonissen in [1]. For examples it is shown that for advection dominant equations the SCF-scheme is first order accurate while the TCF-scheme is second order accurate. The TCF-scheme however produces spurious oscillations while the SCF-scheme does not. For time discretization the $\theta$-method is chosen, which gives second order accuracy in time when $\theta = 1/2$.

Our objective was to find a hybrid scheme, a combination of the TCF- and SCF-
scheme, which is second order and does not create spurious oscillations. By simply using a parameter $\nu \in [0, 1]$ we have constructed this hybrid scheme which consist of $\nu$ times the TCF-scheme and $(1 - \nu)$ times the SCF-scheme. Bounds are found for $\nu$, both independent as well as dependent on the smoothness of the solution, such that the scheme has the TVD property when applied to the 1D advection equation. Examples are given which show that also for the general equation the derived bounds are applicable. The scheme seems to be suitable to discretise the whole class of advection-diffusion-reaction equations. Because of the approximation of the flux using all equation parameters and variables, the scheme seems especially suited for steep interior/boundary layers while having a small discretization stencil.

We have also developed a grid refinement method. In combination with the hybrid scheme this method works very well. We have only tested some examples in 1D, but we see much potential in more than one dimension because of the small coupling of the scheme and the high probability that independent regions needs refinement. Usually grid refinement methods first refine, than assume that in time the refinement area stays the same and occasionally check again if refinement is needed. Our method is continuously scanning all calculated solutions to check whether refinement is needed or not. This could be more expensive, but high resolution is only used where needed. Our method uses a recursive function and a one timestep solver. This makes the method very easy to use and to adjust. Problems that are perfectly suited for this grid refinement method are problems where the regions of high activity where the solution changes very rapidly are only in a small part of the domain. Examples are problems with steep interior or boundary layers.

A typical example of a steep interior layer can be found when calculating the temperature of concentration of fuel in laminar flames. In [5] pp. 99-110, a model is derived of a premixed laminar flame,

$$\frac{d\tau}{dx} - \frac{d}{dx}\left(\frac{\lambda}{\bar{m}c_p} \frac{d\tau}{dx}\right) = S(\tau).$$

(1.1)

The explanation of all parameters can be found in [5], but it is not hard to see the diffusion, advection and reaction terms. The parameter $\tau$ represents a scaled temperature outside a burner. What is typical for these kinds of laminar flames is that on a small part of the domain both temperature as well as the concentration of oxygen/fuel rapidly changes. In order to get a correct solution it is essential that in that specific small section of the domain high resolution is obtained. Also spurious oscillation are not wanted here, because it is impossible to have a negative temperature or negative concentration.

In the case of a laminar flame there is a large but thin peek in the source term. As a teaser, but without detailed explanation, we will show a result of the grid refinement method in combination with the hybrid scheme for an example with such a small peek in the source term. This peek is modelled by a simple Gaussian which can be found in the left image of Figure 1. In the right image of Figure 1 the results can
be found using advection dominant parameters. The black dots are the gridpoints in space and time. The grid refinement is adding more points where the numerical solution changes rapidly. This solution $\varphi$ can represent temperature or concentration etc.

Our report is organized as follows. In Section 2 numerical methods are derived for time independent and time dependent equations. In Section 3 bound are found for $\nu$ such that the hybrid scheme is TVD. In Section 4 these bounds are made dependent on the smoothness of the solution. In Section 5 a grid refinement algorithm is introduced and in Section 6 we present the conclusions and the results are discussed.
2 Finite Volume-Complete Flux scheme

In this section the complete flux scheme will be introduced for both constant and non-
constant parameters of the diffusion-advection-reaction equation. It will be shown
what happens with the scheme when there is no diffusion or no advection. The
$\theta$-method is introduced for time integration and at the end of this section some
examples are given for which the order of convergence is tested. These examples will
be referred to throughout this report.

2.1 Finite Volume Method

Let us first introduce a generic conservation law of advection-diffusion-reaction type,
i.e.,
\[
\frac{\partial \phi}{\partial t} + \nabla \cdot (u \phi - \varepsilon \nabla \phi) = s. \tag{2.1}
\]
which is defined on $\mathbb{R}^d$ ($d \in \{1, 2, 3\}$). Here $u$ is a velocity or an electric field
(flow/drift), $\varepsilon > 0$ a diffusion/conduction coefficient and $s$ a source term. The
unknown $\phi$ can be the concentration of a species in a reacting flow, or the temperature.
The Finite Volume Method (FVM) is applied to (2.1). For the sake of discretisation
we will consider $\varepsilon$, $s$ and $u$ to be functions of the spatial coordinate $x$ and the time$t$. We define the flux vector as
\[
f := u \phi - \varepsilon \nabla \phi. \tag{2.2}
\]
For every fixed domain $\Omega \subset \mathbb{R}^d$ it should hold that
\[
\frac{d}{dt} \int_\Omega \phi dV + \oint_{\Gamma} f \cdot n dS = \int_\Omega s dV, \tag{2.3}
\]
where $n$ is the outward unit normal on the boundary $\Gamma = \partial \Omega$. In the FVM, the
domain is covered with disjunct control volumes $\Omega_j$ and the integral representation
(2.3) is imposed on each of these volumes. Then for each $\Omega_j$,
\[
\frac{d}{dt} \int_{\Omega_j} \phi dV + \sum_{k \in \mathcal{N}(j)} \int_{\Gamma_{j,k}} f \cdot n dS = \int_{\Omega_j} s dV, \tag{2.4}
\]
where $\mathcal{N}(j)$ is the index set of neighbouring grid points of $x_j$ and where $\Gamma_{j,k}$ is the
segment or edge of the boundary $\Gamma_j = \partial \Omega_j$ connecting the adjacent control volumes
$\Omega_j$ and $\Omega_k$. An approximation for the integrals and finding an expression for the
numerical flux usually concludes the FVM.

2.2 Finite Volume-Complete Flux scheme in one
dimension

The following derivation of the Finite Volume-Complete Flux scheme has been done
by J.H.M. ten Thije Boonkamp and M.J.H. Anthonissen as part of their paper [1],
and is partly included in this report for completeness. We consider (2.4) in one
dimension and assume the space time domain to be $\Omega \times [0,T]$ for arbitrary $T \in \mathbb{R}^+$ and $\Omega := [0,1]$. The spatial axis is called the $x$-axis which is discretized into $n_x$ equidistant points. With $j \in \{0,1,\ldots,n_x-1\}$ we define

$$x_j := \frac{j}{n_x-1}. \quad (2.5)$$

For arbitrary function $a(x)$ we define $a_j := a(x_j)$. Equation (2.4) can be written as

$$\frac{d}{dt} \int_{\Omega_j} \varphi \, dx + f(x_{j+1/2}, t) - f(x_{j-1/2}, t) = \int_{\Omega_j} s \, dx, \quad (2.6)$$

where $\Omega_j = (x_{j-1/2}, x_{j+1/2})$. Defining $\varphi_j := \partial \varphi(x_j,t)/\partial t$, Equation (2.6) can be approximated by

$$\varphi_j \Delta x + F_{j+1/2} - F_{j-1/2} = s_j \Delta x, \quad (2.7)$$

using the midpoint rule for the integrals involved, where $\Delta x := x_{j+1/2} - x_{j-1/2}$. This already is a second order approximation. In this one-dimensional case, $F_{j\pm1/2}$ are the numerical fluxes at interface points $x_{j\pm1/2}$. We would like to express these numerical fluxes, for example $F_{j+1/2}$, in terms of $\varphi_j$, $\varphi_{j+1}$, $s_j$ and $s_{j+1}$. In order to do so, we first find an integral representation for the flux. In the interval $(x_j, x_{j+1})$, (2.1) still has to hold. In the one-dimensional case the flux is given by

$$f = u \varphi - \varepsilon \varphi', \quad (2.8)$$

where $\varphi' := \partial \varphi/\partial x$. The idea of the Complete Flux scheme is to use the original conservation law (2.1) in the interval $(x_j, x_{j+1})$ to express the flux at the interface $x_{j+1/2}$ in terms of the equation variables and parameters. This leads to the following BVP

\begin{align*}
(u \varphi - \varepsilon \varphi')' &= \hat{s}, \quad x_j < x < x_{j+1}, \quad (2.9a) \\
\varphi(x_j) &= \varphi_j, \quad \varphi(x_{j+1}) = \varphi_{j+1},
\end{align*}

where $\hat{s}$ is defined either as

\begin{align*}
\hat{s} &= s(x), \quad (2.10a) \\
\hat{s} &= s(x) - \varphi_t. \quad (2.10b)
\end{align*}

Later we will consider both cases. We define the following variables $\lambda$, $P$, $\Lambda$ and $S$ by

$$\lambda := \frac{u}{\varepsilon}, \quad P := \lambda \Delta x, \quad \Lambda := \int_{x_{j+1/2}}^{x} \lambda(y)dy, \quad S(x) := \int_{x_{j+1/2}}^{x} \hat{s}(y)dy, \quad (2.11)$$

with $\Delta x = x_{j+1} - x_j = x_{j+1/2} - x_{j-1/2}$. Here $P$ is the Péclet function. Integrating (2.9a) from $x_{j+1/2}$ to $x$, defining $f_{j+1/2} := f(x_{j+1/2})$, gives the integral balance

$$f(x) - f_{j+1/2} = S(x). \quad (2.12)$$
Using the definition of $\Lambda$ in (2.11), expression (2.8) can be written as

$$f = -\varepsilon (\varphi e^{-\Lambda})' e^\Lambda. \tag{2.13}$$

Substituting (2.13) in (2.12), integrating the resulting equation from $x_j$ to $x_{j+1}$ and applying the boundary conditions (2.9b), we obtain the following expressions for the flux $f_{j+1/2}$:

$$f_{j+1/2} = f_{j+1/2}^h + f_{j+1/2}^i, \tag{2.14a}$$

$$f_{j+1/2}^h = -\left( e^{-\Lambda_{j+1}} \varphi_{j+1} - e^{-\Lambda_j} \varphi_j \right) / \int_{x_j}^{x_{j+1}} \varepsilon^{-1} e^{-\Lambda} dx, \tag{2.14b}$$

$$f_{j+1/2}^i = -\int_{x_j}^{x_{j+1}} \varepsilon^{-1} e^{-\Lambda} S dx / \int_{x_j}^{x_{j+1}} \varepsilon^{-1} e^{-\Lambda} dx. \tag{2.14c}$$

Now let us first assume $\varepsilon$, $u$ and $\hat{s}$ are constants and define the Péclet number $P := u \Delta x / \varepsilon$ which is now constant. In this case the flux can be written in a nice compact expression. The denominator of $f_{j+1/2}^h$ can be written as

$$\int_{x_j}^{x_{j+1}} \varepsilon^{-1} e^{-\Lambda} dx = \varepsilon^{-1} \int_{x_j}^{x_{j+1}} e^{-\lambda (x - x_{j+1/2})} dx \tag{2.15}$$

$$\lambda \neq 0 \quad \lambda^{-1} = -\lambda^{-1} \left( -e^{-\lambda(x_j - x_{j+1/2})} + e^{-\lambda(x_{j+1} - x_{j+1/2})} \right)$$

$$= e^\lambda x_{j+1/2} / u (e^{-\lambda x_j} - e^{-\lambda x_{j+1}}).$$
Thus the homogeneous part can be written as
\[
f_j^{h} = -u\frac{e^{-\lambda_{j+1} \varphi_{j+1} - e^{-\lambda_{j} \varphi_{j}}}}{e^{\lambda_{j+1}/2}(e^{-\lambda_{j}} - e^{-\lambda_{j+1}})}
\]
\[
= \frac{u(-e^{-\lambda(x_{j+1}-x_{j+1/2}) \varphi_{j+1} + e^{-\lambda(x_{j}-x_{j+1/2}) \varphi_{j}}})}{e^{\lambda_{j+1}/2}(e^{-\lambda_{j}} - e^{-\lambda_{j+1}})}
\]
\[
= \frac{ue^{\lambda_{j+1}/2}(-e^{-\lambda x_{j+1}} \varphi_{j+1} + e^{-\lambda x_{j}} \varphi_{j})}{e^{\lambda_{j+1}/2}(e^{-\lambda_{j}} - e^{-\lambda_{j+1}})}
\]
\[
= \frac{u(-e^{-\lambda x_{j+1}} \varphi_{j+1} + e^{-\lambda x_{j}} \varphi_{j})}{e^{-\lambda x_{j}} - e^{-\lambda x_{j+1}}}
\]
\[
= -\frac{ue^{-\lambda x_{j+1}} \varphi_{j+1}}{e^{-\lambda x_{j}} - e^{-\lambda x_{j+1}}} + \frac{ue^{-\lambda x_{j}} \varphi_{j}}{e^{-\lambda x_{j}} - e^{-\lambda x_{j+1}}}
\]
\[
= -\frac{\varepsilon}{\Delta x} \frac{\Delta x}{\varepsilon \Delta x - 1} + \frac{\Delta x}{\varepsilon \Delta x - 1} \varphi_{j+1}
\]
\[
= -\frac{\varepsilon}{\Delta x} \left( \frac{P \varphi_{j+1}}{e^P - 1} + \frac{P \varphi_{j}}{e^P - 1} \right)
\]
\[
= -\frac{\varepsilon}{\Delta x} \left( B(P) \varphi_{j+1} - B(-P) \varphi_{j} \right),
\]
with
\[
B(z) := \frac{z}{e^z - 1}.
\]

If \(\lambda = 0\) the denominator can be written as
\[
\int_{x_j}^{x_{j+1}} \varepsilon^{-1} e^{-\Lambda} dx = \varepsilon^{-1} \int_{x_j}^{x_{j+1}} 1 dx = \frac{\Delta x}{\varepsilon},
\]
thus for \(\lambda = 0\) we get
\[
f_j^{h} = -\frac{\varepsilon}{\Delta x} (\varphi_{j+1} - \varphi_{j}),
\]
which is equivalent with \(-\varepsilon/\Delta x (B(0) \varphi_{j+1} - B(-0) \varphi_{j})\), because
\[
\lim_{z \to 0} B(z) = \lim_{z \to 0} \frac{z}{z + \frac{1}{2}z^2 + O(z^3)} = 1.
\]

A similar derivation can be done for the inhomogeneous part which leads to
\[
f_j^{i} = \left( \frac{1}{2} - W(P) \right) \hat{s} \Delta x,
\]
with
\[ W(z) := \frac{e^{z} - z - 1}{z(e^{z} - 1)}. \] (2.22)

In Figure 2 the functions $B$ and $W$ are plotted. Function $W$ has the weight function properties

\[ 0 \leq W(z) \leq 1, \]
\[ W(-z) + W(z) = 1, \] (2.23)

for arbitrary $z \in \mathbb{R}$. In the general case, when $\varepsilon$, $u$ and $\hat{s}$ are not constant the derivation is more complicated but the results are similar. The derivation can be found in [1], pp. 52-56. Using the arithmetic and weighted average defined as

\[ \bar{a}_{j+1/2} := \frac{1}{2} (a_j + a_{j+1}), \]
\[ \tilde{a}_{j+1/2} := W(-\bar{P}_{j+1/2})a_j + W(\bar{P}_{j+1/2})a_{j+1}, \] (2.24)

of an arbitrary parameter $a$, the numerical flux at the cell interface $x_{j+1/2}$ can be written as

\[ F_{j+1/2} = \alpha_{j+1/2}\varphi_j - \beta_{j+1/2}\varphi_{j+1} + \Delta x(\gamma_{j+1/2}\hat{s}_j + \delta_{j+1/2}\hat{s}_{j+1}), \] (2.25)

where the coefficients are defined as

\[ \alpha_{j+1/2} = \frac{\mathcal{E}_{j+1/2}}{\Delta x} B(-\bar{P}_{j+1/2}), \]
\[ \beta_{j+1/2} = \frac{\mathcal{E}_{j+1/2}}{\Delta x} B(\bar{P}_{j+1/2}), \]
\[ \gamma_{j+1/2} = \max\left(\frac{1}{2} - W(\bar{P}_{j+1/2}), 0\right), \]
\[ \delta_{j+1/2} = \min\left(\frac{1}{2} - W(\bar{P}_{j+1/2}), 0\right), \] (2.26)
with

\[ P_j := \frac{u(x_j) \Delta x}{\varepsilon(x_j)}, \]
\[ \mathcal{E}_{j+1/2} := \frac{\lambda_{j+1/2}}{\lambda_{j+1/2}} \tilde{\varepsilon}_{j+1/2}, \]
\[ \lambda_j := \frac{u(x_j)}{\varepsilon(x_j)}, \]
\[ \varepsilon_j := \varepsilon(x_j). \] (2.27)

Substituting these coefficients in the discrete conservation law (2.7) we get for the stationary case

\[-a_{W,j} \varphi_{j-1} + a_{C,j} \varphi_j - a_{E,j} \varphi_{j+1} = b_{W,j} s_{j-1} + b_{C,j} s_j + b_{E,j} s_{j+1}, \] (2.28)

with the coefficients defined as

\[ a_{W,j} := \alpha_{j-1/2}, \quad a_{E,j} := \beta_{j+1/2}, \quad a_{C,j} := \alpha_{j+1/2} + \beta_{j-1/2}, \]
\[ b_{W,j} := \gamma_{j-1/2} \Delta x, \quad b_{E,j} := -\delta_{j+1/2} \Delta x, \quad b_{C,j} := (1 - \gamma_{j+1/2} + \delta_{j-1/2}) \Delta x. \] (2.29)

When \( \varphi_t \neq 0 \) we get either the stationary complete flux scheme (SCF)

\[ \dot{\varphi}_j \Delta x - a_{W,j} \varphi_{j-1} + a_{C,j} \varphi_j - a_{E,j} \varphi_{j+1} = b_{W,j} s_{j-1} + b_{C,j} s_j + b_{E,j} s_{j+1}, \] (2.30)

using (2.10a), or the transient complete flux scheme (TCF)

\[ b_{W,j} \dot{\varphi}_{j-1} + b_{C,j} \dot{\varphi}_j + b_{E,j} \dot{\varphi}_{j+1} - a_{W,j} \varphi_{j-1} + a_{C,j} \varphi_j - a_{E,j} \varphi_{j+1} = b_{W,j} s_{j-1} + b_{C,j} s_j + b_{E,j} s_{j+1}, \] (2.31)

using (2.10b). Let \( a_W := (a_{W,0}, ..., a_{W,n_x-1}) \) and let \( a_C, a_E \) be defined similarly. Let \( A \) be the matrix with \( a_W \) placed at the first lower diagonal, \( a_C \) on the main diagonal and \( a_E \) on the first upper diagonal. The placement of the first and last elements of the coefficient vectors depends on the boundary conditions of (2.1) which we would like to leave unspecified. Matrix \( B \) is defined similarly. In the stationary case we get the following linear system,

\[ A \varphi = B s + b. \] (2.32)

The SCF-scheme is defined as

\[ \dot{\varphi} \Delta x + A \varphi = B s + b_1, \] (2.33)

and the TCF-scheme as

\[ B \dot{\varphi} + A \varphi = B s + b_2. \] (2.34)

The vectors \( b, b_1 \) and \( b_2 \) depends on the boundary conditions of (2.1) and on \( s \).
2.3 Limiting cases

In this paragraph we will spend some time deriving special cases of the earlier defined schemes. These special cases can be seen as simplifications of the general scheme which will be helpful later on.

Case 1: In the first case the limit for $\varepsilon \to 0$ is taken. So no diffusion takes place and the Péclet number goes to infinity. The coefficient $\alpha_{j+1/2}$, defined in (2.26), can be written as

$$\alpha_{j+1/2} = \frac{1}{\lambda_{j+1/2}\Delta x} \left( W(-\bar{P}_{j+1/2}) \frac{u_j}{\varepsilon_j} + W(\bar{P}_{j+1/2}) \frac{u_{j+1}}{\varepsilon_{j+1}} \right).$$

(2.35)

Assuming $\frac{1}{2}(u_j + u_{j+1}) > 0$, $\varepsilon$ continuous and using that $W(P) \to 0$ and $W(-P) \to 1$ for $P \to \infty$, we get

$$\lim_{\varepsilon \to 0} \alpha_{j+1/2} = \lim_{\varepsilon \to 0} \frac{\bar{P}_{j+1/2}}{\lambda_{j+1/2}\Delta x} \frac{-1}{e^{-\bar{P}_{j+1/2}} - 1} = u_j. \quad (2.36)$$

A similar derivation can be done for the other coefficients defined in (2.26), which results in

$$\alpha_{j+1/2} = u_j,$n
$$\beta_{j+1/2} = 0,$n
$$\gamma_{j+1/2} = \frac{1}{2},$$n
$$\delta_{j+1/2} = 0. \quad (2.37)$$

For $\frac{1}{2}(u_j + u_{j+1}) < 0$, $\alpha_{j+1/2} = 0$, $\beta_{j+1/2} = -u_{j+1}$, $\gamma_{j+1/2} = 0$ and $\delta_{j+1/2} = -\frac{1}{2}$. If for example for all $j$, $u_j > 0$, the stationary scheme (2.32) reduces to

$$u_j \varphi_j - u_{j-1} \varphi_{j-1} = \frac{1}{2}(s_{j-1} + s_j) \Delta x, \quad (2.38)$$

which can be interpreted as a second order cell-vertex FVM [9]. Standard methods omit the inhomogeneous flux, such that (2.38) reduces to

$$u_j \varphi_j - u_{j-1} \varphi_{j-1} = s_j \Delta x, \quad (2.39)$$

which is just the first order upwind scheme for the advection equation.

Case 2: In this second case the coefficients in (2.26) are evaluated for $u \to 0$. Using $W(0) = 1/2$ and (2.20) we get for $\alpha_{j+1/2}$:

$$\lim_{u \to 0} \alpha_{j+1/2} = \lim_{u \to 0} \frac{\lambda}{\Delta x} \frac{W(-\bar{P}_{j+1/2}) \varepsilon_j + W(\bar{P}_{j+1/2}) \varepsilon_{j+1}}{e^{-\bar{P}_{j+1/2}} - 1} = \frac{\varepsilon_{j+1/2}}{\Delta x}. \quad (2.40)$$
A similar derivation can be done for the other coefficients which results in
\[
\alpha_{j+1/2} = \frac{\bar{\epsilon}_{j+1/2}}{\Delta x}, \\
\beta_{j+1/2} = \frac{\bar{\epsilon}_{j+1/2}}{\Delta x}, \\
\gamma_{j+1/2} = 0, \\
\delta_{j+1/2} = 0.
\]
(2.41)

With these coefficients using the stationary scheme (2.32) we get the second order central difference scheme,
\[
-\frac{1}{\Delta x} (\bar{\epsilon}_{j+1/2}(\varphi_{j+1} - \varphi_j) - \bar{\epsilon}_{j-1/2}(\varphi_j - \varphi_{j-1})) = s_j \Delta x.
\]
(2.42)

### 2.4 Time integration

There are many different methods to discretise the time derivative. We have chosen to use the \(\theta\)-method. The time axes \([0, T]\) is divided into \(n_t\) points. We define
\[
\Delta t := \frac{T}{n_t - 1},
\]
(2.43)
and define \(t_i := i \Delta t, i \in \{0, 1, ..., n_t - 1\}\). Going forward in time, the solution at the new time level \(t_{i+1}\) is calculated using the solution at time \(t_i\). This process starts with a given initial solution at \(t_0\). Applying the \(\theta\)-method to an arbitrary ODE \(\dot{\varphi} = G(\varphi, t)\) results in
\[
\frac{\varphi^{i+1} - \varphi^i}{\Delta t} = \theta G(\varphi^{i+1}, t_{i+1}) + (1 - \theta) G(\varphi^i, t_i),
\]
(2.44)
where \(\varphi^{i+1}\) is the numerical approximation of \(\varphi(x, t_{i+1})\) and \(x = (x_0, x_1, ...)\). This notation with upper indices will also be used for other parameters that are depending on the time \(t\), such as \(s(x, t)\) and vectors \(b_1(t), b_2(t)\) from (2.33) and (2.34). In this report we will assume \(u = u(x)\) and \(\varepsilon = \varepsilon(x)\) such that both matrices \(A\) and \(B\) with coefficients defined in (2.29) are time independent. The only reason for this is that it speeds up the calculations, because all coefficients of \(A\) and \(B\) have to be calculated only once. In practice \(u\) does often depend on time. Using the \(\theta\)-method on the SCF-scheme (2.33) results in
\[
\left(\frac{\Delta x}{\Delta t} I + \theta A\right) \varphi^{i+1} = \left(\frac{\Delta x}{\Delta t} I - (1 - \theta) A\right) \varphi^i + \theta (Bs^{i+1} + b_1^{i+1}) + (1 - \theta) (Bs^i + b_1^i).
\]
(2.45)

The TCF scheme (2.34) becomes
\[
\left(\frac{B}{\Delta t} + \theta A\right) \varphi^{i+1} = \left(\frac{1}{\Delta t} B - (1 - \theta) A\right) \varphi^i + \theta (Bs^{i+1} + b_2^{i+1}) + (1 - \theta) (Bs^i + b_2^i).
\]
(2.46)
In general, the $\theta$-method is stable for $\theta \in [1/2, 1]$, but even for lower values of $\theta$ the method can be stable, [10] pp. 6, 7. Later on it will be explained that it is sometimes best to combine the two schemes. This paragraph is concluded with introducing the combined TCF/SCF scheme with parameter $\nu \in [0, 1]$ using the $\theta$-method for time integration:

$$\left( \frac{1}{\Delta t} ((1 - \nu) \Delta x I + \nu B) + \theta A \right) \varphi^{i+1} = \left( \frac{1}{\Delta t} ((1 - \nu) \Delta x I + \nu B) - (1 - \theta) A \right) \varphi^i + \theta (Bs^{i+1} + b_3^{i+1}) + (1 - \theta) (Bs^i + b_3^i).$$

(2.47)

Here $b_3 = \nu b_2 + (1 - \nu) b_1$. The vector $b_1$ is not necessarily equal to $b_2$ when for example Neumann boundary conditions are used. Choosing $\nu = 0$ gives the SCF-scheme and choosing $\nu = 1$ gives the TCF-scheme.

### 2.5 Order of convergence in time and space

In this paragraph three examples are given. First a low and second a high Péclet number is taken. In the last example a discontinuous initial condition is given. All examples are one dimensional and examples of the following IBVP:

$$\frac{\partial \varphi}{\partial t} + \frac{\partial}{\partial x} \left( u \varphi - \varepsilon \frac{\partial \varphi}{\partial x} \right) = s, \quad 0 < x < 1, \quad t > 0,$$

$$\varphi(x, 0) = \varphi_{in}, \quad \varphi(0, t) = \varphi_l, \quad \varphi(1, t) = \varphi_r.$$

(2.48)

All examples will be simulated using both the TCF-scheme and the SCF-scheme with $\theta = 1/2$. In the first two examples Richardson extrapolation is used to determine the order of convergence. This is done by using data $h_k$ and $a_k$ obtained from 8 simulations. In simulation $k \in \{0, 1, 2, \ldots\}$ a grid is used of size $\Delta x = \Delta t = h_k := 1/10 \cdot (1/2)^k$ to calculate $a_k$ which will be an approximation of $\varphi(1/2, 1)$. We assume the error $e_k$ can be written as

$$e_k = \varphi(1/2, 1) - a_k = \sum_{l=p}^{\infty} c_l h_k^l,$$

(2.49)

where $c_l \in \mathbb{R}$ is a coefficient and $p \in \mathbb{N}^+$. This way we can approximate $\varphi(1/2, 1)$ with

$$\varphi(1/2, 1) \approx a_k + c_p h_k^p,$$

$$\varphi(1/2, 1) \approx a_{k+1} + c_p h_{k+1}^p,$$

$$\varphi(1/2, 1) \approx a_{k+2} + c_p h_{k+2}^p,$$

(2.50)

which implies that

$$\frac{a_{k+1} - a_k}{a_{k+2} - a_{k+1}} \approx \frac{c_p}{c_p} \frac{h_{k+1} - h_k}{h_{k+2} - h_{k+1}} = \frac{1/10}{1/10} \frac{1/2}{1/2} \frac{1/2}{1/2} (1 - 1/2^p) = 2^p.$$

(2.51)
Figure 3: Numerical solution of example 2.1 plotted with $\Delta x = \Delta t = 1/320$.

The order $p$ can be determined by

$$p \approx \log \left( \frac{|a_{k+2} - a_{k+1}|}{|a_{k+1} - a_k|} \right) / \log \left( \frac{1}{2} \right). \quad (2.52)$$

**Example 2.1. Dominant diffusion:** A diffusion dominant example is taken with the following parameters:

$$s = 0, \quad u = \frac{1}{200}, \quad \varepsilon = \frac{1}{20},$$

$$\varphi_{\text{in}} = \begin{cases} 1/2 - 1/2 \cos(4\pi x) & x \in [0, 1/2) \\ 0 & \text{elsewhere} \end{cases}, \quad \varphi_{\text{l}} = \varphi_{\text{r}} = 0. \quad (2.53)$$

The numerical solution is given in Figure 3. In Table 1 the results of Richardson extrapolation are given using the TCF and SCF-scheme using the $\theta$-method for time integration. The parameter $p$ is the order of convergence assuming both schemes actually converge. Both schemes seem to be of order $p = 2$.

**Example 2.2. Dominant advection:** Our second example will be almost an advection-reaction type of equation. It will have an oscillating source term in time
Table 1: Richardson extrapolation applied to the numerical results of the diffusion dominant example with $\Delta x = \Delta t = h$, using the TCF/SCF-scheme with the $\theta$-method.

<table>
<thead>
<tr>
<th>$h_k^{-1}$</th>
<th>TCF</th>
<th>SCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.20686085</td>
<td>0.20678829</td>
</tr>
<tr>
<td>20</td>
<td>0.20672998</td>
<td>0.20671184</td>
</tr>
<tr>
<td>40</td>
<td>0.20669926</td>
<td>0.20669472</td>
</tr>
<tr>
<td>80</td>
<td>0.20669174</td>
<td>2.029</td>
</tr>
<tr>
<td>160</td>
<td>0.20668986</td>
<td>2.008</td>
</tr>
<tr>
<td>320</td>
<td>0.20668940</td>
<td>2.002</td>
</tr>
<tr>
<td>640</td>
<td>0.20668928</td>
<td>2.001</td>
</tr>
<tr>
<td>1280</td>
<td>0.20668925</td>
<td>1.997</td>
</tr>
</tbody>
</table>

Table 1: Richardson extrapolation applied to the numerical results of the diffusion dominant example with $\Delta x = \Delta t = h$, using the TCF/SCF-scheme with the $\theta$-method.

and a positive constant-in-time velocity field. The parameters are given by

\[
\begin{align*}
 s &= -\frac{1}{10}(a \omega \cos(\omega t)) + au + 2\varepsilon a^2 \tanh(a(x - (1/2 + 1/10 \sin(\omega t)))) \cosh(a(x - (1/2 + 1/10 \sin(\omega t))))) , \\
 u &= 1/2, \quad \varepsilon = 10^{-8} , \\
 \phi_{\text{in}} &= \tanh(a(x - 1/2)) + 1, \quad \phi_{\text{f}} = \tanh(a(-1/2 + 1/10 \sin(\omega t))) + 1 , \\
 \phi_{\text{r}} &= \tanh(a(1 - (1/2 + 1/10 \sin(\omega t))) + 1, \\
 a &= 100, \quad \omega = 4\pi .
\end{align*}
\]

(2.54)

The source term seems complicated, but it is calculated using a given solution. Thus in this example the solution is known, it is a simple tanh-function and is defined as

\[
\phi_s := \tanh(a(x - (1/2 + 1/10 \sin(\omega t)))) + 1,
\]

(2.55)

where $a$ is a parameter for how steep the solution will be and $\omega$ is a parameter for the frequency. In Figure 5 this solution is plotted. In Table 2 the results of Richardson extrapolation are given using the numerical results of this advection dominant example solved with the TCF and SCF-scheme. It is notable that it takes some time before $p$, obtained from Richardson extrapolation, actually converges to a stable order. This could be caused by the very large second derivative of the solution because $a = 100$. In Figure 4 two plots are given of $\phi(x, 1)$. The red line (TCF) is a little bit sharper than the blue one (SCF). Both images are showing some small oscillatory behavior of the TCF-scheme and some large smooth errors of the SCF-scheme.

**Example 2.3. Discontinuous initial condition:** In Example 2.2 some oscillations occurred using the TCF-scheme. In this example it is shown that oscillations can be
Table 2: Richardson extrapolation applied to the numerical results of an advection dominated flow with $\Delta x = \Delta t = h$, using the TCF/SCF-scheme with the $\theta$-method.

<table>
<thead>
<tr>
<th>$h_k^{-1}$</th>
<th>TCF</th>
<th>SCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_k$</td>
<td>$p_k$</td>
<td>$a_k$</td>
</tr>
<tr>
<td>10</td>
<td>-2.00473573889</td>
<td>-1.34207122222</td>
</tr>
<tr>
<td>20</td>
<td>0.95233595738</td>
<td>1.02810273604</td>
</tr>
<tr>
<td>40</td>
<td>0.53786933072</td>
<td>3.3768268559</td>
</tr>
<tr>
<td>80</td>
<td>0.103343272056</td>
<td>-0.258</td>
</tr>
<tr>
<td>160</td>
<td>1.01227520380</td>
<td>4.550</td>
</tr>
<tr>
<td>320</td>
<td>1.00024428762</td>
<td>0.814</td>
</tr>
<tr>
<td>640</td>
<td>1.00011727899</td>
<td>6.566</td>
</tr>
<tr>
<td>1280</td>
<td>1.00002942799</td>
<td>0.532</td>
</tr>
<tr>
<td>2560</td>
<td>1.00000736333</td>
<td>1.993</td>
</tr>
<tr>
<td>5120</td>
<td>1.00000184123</td>
<td>1.998</td>
</tr>
</tbody>
</table>

Figure 4: Example 2.2 at $t = 1$. In red TCF-scheme, in blue SCF-scheme, in black analytical solution, left $\Delta x = \Delta t = 1/80$, right $\Delta x = \Delta t = 1/160$.

Figure 5: Solution of Example 2.2 (left) and Example 2.3 (right)
Figure 6: Numerical solution of Example 2.3 at $t = 1$. In red using TCF-scheme, in blue using SCF-scheme, left $\Delta x = \Delta t = 1/40$, right $\Delta x = \Delta t = 1/640$.

a problem indeed. The following parameters are being used:

$$s = 0, \quad u = \frac{1}{2}, \quad \varepsilon = 0,$$

$$\varphi_{\text{in}} = \begin{cases} 1 & x \in [1, 2) \\ 0 & \text{else} \end{cases}, \quad \varphi_l = \varphi_r = 0. \quad (2.56)$$

In Figure 5 the solution is plotted. In Figure 6 two plots of $\varphi(x, 1)$ are given for two different resolutions. The TCF-scheme clearly caused oscillations, even with a very high resolution, while the SCF-scheme seems prone to much more numerical diffusion. Both schemes seem to be unsuited for discontinuous initial conditions with the advection equation.

The conclusion so far is that the TCF-scheme seems to be of order 2 in both advection dominated as well as diffusion dominated flows. Therefore, one could prefer the TCF-scheme to the SCF-scheme. However, the TCF-scheme seems to sometimes create oscillations. In the next section we will try to find the exact cause of this and by combining the two schemes create a scheme that is always second order and does not create oscillations.
3 TVD restrictions

In this section restrictions are derived for the parameter $\nu$ such that the scheme applied to the advection-diffusion equation is TVD using constant coefficients. First we start with an intuitive example without diffusion and source which is so simple that the solution can be expressed in terms of the equations parameters. When demanding that the solution stays positive, restrictions for $\nu$ are found. These restrictions turn out to be the exact same as the restrictions found when demanding the TVD property for the general case. At the end it is shown that there are examples where the bounds are sharp.

3.1 Restrictions on combined TCF/SCF-scheme for $\varepsilon = 0$ (advection equation)

To test if varying $\theta$ and $\nu$ can influence or even prevent oscillations we start with a very basic one dimensional example. In this example we choose the space time domain to be $[0, 1] \times [0, 1]$ and choose $n_x = n_t = 7$ such that the $x$-axis as well as the time axis are covered with 7 equidistant points. As initial condition $\varphi_0(x) = (0, 0, 0, 1, 0, 0, 0)$ is used and $\varphi_l = \varphi_r = 0$ are the boundary conditions. With $\varepsilon = 0$, no source term and an arbitrary constant $u(x, t) = u \in \mathbb{R}$ ($P \to \infty$) one timestep is calculated using Mathematica [11]. The goal is to find restrictions for $\theta$ and $\nu$ such that $\varphi^1 \geq 0$. These restrictions might also apply to other examples with other initial and boundary conditions, and might help to find restrictions for examples with a low Péclet number. Using the combined SCF/TCF-scheme (2.47) the following system of equations has to be solved.

$$
\left( \frac{\Delta x I + \nu (B - \Delta x I)}{\Delta t} + \theta A \right) \varphi^1 = \left( \frac{\Delta x I + \nu (B - \Delta x I)}{\Delta t} - (1 - \theta) A \right) \varphi^0.
$$

(3.1)

For notational purposes we introduce the Courant number $c$ which is defined as

$$
c := u \frac{\Delta t}{\Delta x}.
$$

(3.2)

With $\varepsilon = 0$ and with $u > 0$, the matrices $A$ and $B$ reduce to

$$
A = u \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & -1 & 1
\end{pmatrix}, \quad B = \frac{1}{2} \Delta x \begin{pmatrix}
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1
\end{pmatrix}.
$$

(3.3)

Solving (3.1) gives the solution for $u > 0$

$$
\varphi^1 = \begin{pmatrix}
0, 0, \frac{1 + c(\theta - 1) - \frac{1}{2} \nu}{1 + c\theta - \frac{1}{2} \nu}, \frac{c}{(1 + c\theta - \frac{1}{2} \nu)^2}, \frac{c(c\theta - \frac{1}{2} \nu)}{(1 + c\theta - \frac{1}{2} \nu)^3}
\end{pmatrix}.
$$

(3.4)
If \( u < 0 \) we get a slightly different solution but with exact same conditions when demanding \( \phi^1 \geq 0 \). Matrices \( A \) and \( B \) are mirrored over the diagonal, we find
\[
\phi^1 = \left( \frac{c(c\theta + \frac{1}{2} \nu)}{(1-c\theta - \frac{1}{2} \nu)^2}, \frac{-c}{(1-c\theta - \frac{1}{2} \nu)^2}, \frac{1 - c(\theta - 1) - \frac{1}{2} \nu}{1 - c\theta - \frac{1}{2} \nu}, 0, 0 \right). \tag{3.5}
\]
The denominators of all fractions are always positive since \( \theta, \nu \in [0,1] \). From the numerators it can be obtained that \( \phi^1 \geq 0 \) if and only if
\[
\frac{2-\nu}{\nu - 2\theta} \geq |c| \geq \frac{\nu}{2\theta} \quad \theta \in (0,1),
\]
\[
|c| \geq \frac{\nu}{2} \quad \theta = 1.
\tag{3.6}
\]
\[
\nu = 0, |c| < 1 \quad \theta = 0.
\]
For \( \theta = 1/2 \) using the TCF-scheme, \( |c| \) has to be equal to 1, while the SCF-scheme allows \( |c| \) to be in \((0,2)\). For \( \nu = 0 \), \( \phi^1 \geq 0 \) if and only if
\[
|c| (1 - \theta) \leq 1. \tag{3.7}
\]
If \( \theta = \nu = 0 \) we have the first order upwind scheme with stability condition that \( |c| \leq 1 \) which is exactly the same as (3.7).

### 3.2 TVD restrictions combined TCF/SCF-scheme, constant coefficients

In the next example an arbitrary \( \varepsilon > 0 \) is chosen. The coefficients of \( A \) and \( B \) are now more complicated, but \( A \) is a tridiagonal matrix and \( B \) is a bidiagonal matrix with an extra diagonal above or below the main diagonal depending on the sign of the Péclet number, thus depending on the sign of \( u \). Because of the diffusion it may not be enough to demand \( \phi^{i+1} > 0 \) if \( \phi^i > 0 \). In this case we demand
\[
TV(\phi^{i+1}) := \sum_j |\phi_{j+1}^i - \phi_{j+1}^{i+1}| \leq TV(\phi^i) := \sum_j |\phi_{j}^i - \phi_{j-1}^i|, \tag{3.8}
\]
such that oscillations are not allowed to arise. Total Variation (TV) is defined as the sum of the absolute value of the increments. When oscillations arise, the total variation will increase. Property (3.8) of a scheme is called Total Variation Diminishing (TVD), [4] p. 392. Let \( K \) and \( L \) be two diagonally dominant tridiagonal matrices with constant coefficients on the diagonals. Let \( k_W \) denote the value at the first lower diagonal, \( k_C \) the main diagonal and \( k_E \) the first upper diagonal of matrix \( K \). Let \( l_W, l_C \) and \( l_E \) denote the value at the three diagonals of matrix \( L \) alike the diagonals of matrix \( K \). Consider the equation
\[
K \phi^{i+1} = L \phi^i. \tag{3.9}
\]
For arbitrary \( j \) (not adjacent to the boundary) we may write
\[
k_W \phi_{j-1}^{i+1} + k_C \phi_{j}^{i+1} + k_E \phi_{j+1}^{i+1} = l_W \phi_{j-1}^{i} + l_C \phi_{j}^{i} + l_E \phi_{j+1}^{i}
\tag{3.10}
\]
Equation (3.10) for \( j - 1 \) is subtracted from (3.10) and terms are brought to the right-hand side to get an equation in incremental form, i.e.,

\[
k_C (\varphi_{j+1}^{i+1} - \varphi_{j}^{i+1}) = -k_W (\varphi_{j-1}^{i+1} - \varphi_{j}^{i+1}) - k_E (\varphi_{j+1}^{i+1} - \varphi_{j}^{i+1}) \\
+ l_W (\varphi_{j-1}^{i} - \varphi_{j}^{i-1}) + l_C (\varphi_{j}^{i} - \varphi_{j-1}^{i}) + l_E (\varphi_{j+1}^{i} - \varphi_{j}^{i}) .
\] (3.11)

For notational simplicity we introduce the diffusion number which is defined as

\[
d := \frac{\varepsilon \Delta t}{(\Delta x)^2}.
\] (3.12)

Using the combined TCF/SCF-scheme, multiplying (2.47) with \( \Delta t/\Delta x \), the entries of the matrices \( K \) and \( L \) are defined as

\[
k_W = \nu \max \left( \frac{1}{2} - W(P), 0 \right) - d\theta B(-P), \\
k_E = -\nu \min \left( \frac{1}{2} - W(P), 0 \right) - d\theta B(P), \\
k_C = 1 - (k_W + k_E), \\
l_W = \nu \max \left( \frac{1}{2} - W(P), 0 \right) + d(1 - \theta) B(-P), \\
l_E = -\nu \min \left( \frac{1}{2} - W(P), 0 \right) + d(1 - \theta) B(P), \\
l_C = 1 - (l_W + l_E).
\] (3.13)

Since \( B(P) > 0 \) for all \( P \in \mathbb{R} \) and \( \theta \in [1/2, 1] \) it is clear that \( l_W \geq 0 \) and \( l_E \geq 0 \). For the diagonal element it holds that

\[
l_C \geq 0 \iff d(1 - \theta)(B(P) + B(-P)) + \nu \left| \frac{1}{2} - W(P) \right| \leq 1.
\] (3.14)

Since \( B(P) > 0, W(P) \in (0, 1) \) for all \( P \in \mathbb{R} \) and \( \theta \in [1/2, 1], \nu \in [0, 1] \), we have that \( k_C \geq 0 \). When demanding that \( k_W \leq 0 \) and \( k_E \leq 0 \) we need to have the conditions

\[
d\theta B(-P) \geq \nu \max \left( \frac{1}{2} - W(P), 0 \right), \\
d\theta B(P) \geq -\nu \min \left( \frac{1}{2} - W(P), 0 \right).
\] (3.15)

Depending on the sign of \( P \) just one condition is not trivial, thus (3.15) can be combined to one condition. It holds that

\[
(k_W \leq 0 \land k_E \leq 0) \iff d\theta B(-|P|) \geq \nu \left| \frac{1}{2} - W(P) \right|.
\] (3.16)
For a diffusion number \( d \geq 1 \) inequality (3.16) is always true. Since \( \theta \in [1/2, 1] \) and \( \nu \in [0, 1] \) we get

\[
d\theta B(-|P|) \geq \frac{1}{2} B(-|P|) \geq \frac{1}{2} \geq \nu \left| \frac{1}{2} - W(P) \right|. \tag{3.17}
\]

When (3.14) and (3.16) hold and thus all increment coefficients in (3.11) are positive, the absolute value left and right and the sum over \( j \) can be taken in (3.11) to get

\[
kC \sum_j |\varphi_j^{i+1} - \varphi_{j-1}^{i+1}| \leq -kW \sum_j |\varphi_{j-1}^{i+1} - \varphi_{j-2}^{i+1}| - kE \sum_j |\varphi_{j+1}^{i+1} - \varphi_j^{i+1}| + lC \sum_j |\varphi_j^i - \varphi_{j-1}^i| + lw \sum_j |\varphi_{j-1}^i - \varphi_{j-2}^i| + lE \sum_j |\varphi_{j+1}^i - \varphi_j^i|. \tag{3.18}
\]

With shifting the index \( j \) in several sums it is now possible to write

\[
(kC + kW + kE) \sum_j |\varphi_j^{i+1} - \varphi_{j-1}^{i+1}| \leq (lC + lw + lE) \sum_j |\varphi_j^i - \varphi_{j-1}^i|. \tag{3.19}
\]

Both coefficients sum up to 1 which gives the desired result:

\[
\sum_j |\varphi_j^{i+1} - \varphi_{j-1}^{i+1}| \leq \sum_j |\varphi_j^i - \varphi_{j-1}^i|. \tag{3.20}
\]

Thus when restrictions (3.14) and (3.16) hold, the scheme has the TVD property. The derivation of these restrictions is similar to [2] and also the restrictions correspond to the restrictions found in [2] p. 7. In Figure 7 these two restrictions rewritten as upperbound for \( \nu \) are plotted as functions of \( d \) ((3.14) in red and (3.16) in blue) for different values of the Courant number. In the last image, we have also changed \( \theta \), which results in a larger TVD area. For large \( d \) TVD can not be guaranteed which seems strange because for large \( \varepsilon \) it is expected that the numerical solution smoothes out and will be TVD. However, these restrictions hold for any given initial condition and for large \( \varepsilon \) and discontinuous initial condition there could be some stability issues. This not investigated any further. For small diffusion number, \( d \leq 1/2 \), these bounds are applicable. The two restrictions can also be expressed in terms of the Péclet number and the Courant number. Restriction (3.14) and (3.16) can be rewritten as respectively

\[
|c| \coth \left( \frac{|P|}{2} \right) \leq \frac{1 - \nu}{1 - \theta} \frac{\left| \frac{1}{2} - W(P) \right|}{1 - \theta}, \tag{3.21}
\]

and

\[
|c| \frac{1}{1 - e^{-|P|}} \nu \frac{\left| \frac{1}{2} - W(P) \right|}{\theta}. \tag{3.22}
\]

It is now easy to see that these upper bounds turn out to be exactly the same as (3.6) for \( \varepsilon \to 0 \). In Figure 8 the bounds are plotted as restrictions for \( \nu \) for \( \theta = 1/2 \).
For large $P$ (small $\varepsilon$) the bounds converge to (3.6).

Next we will see that for a given $|c| < 2$ only one of the two restrictions is necessary. If $1 \leq |c| \leq 2$ and $\theta \in [1/2, 1]$ we may conclude

$$\frac{d\theta B(-|P|)}{|1/2 - W(P)|} \geq \frac{d|P|}{1/2} \geq 2\theta|c| \geq 1,$$

(3.23)

where we have used that $B(-|P|) \geq |P|$. From (3.16) we can now see that this restriction is not necessary anymore because $\nu$ can not be larger than one. If $0 \leq |c| \leq 1, \theta \in [1/2, 1]$ and $d \leq 1/2$ we can do something similar to restriction (3.14)/(3.21) only it will be more work. We will use that

$$1/2 - W(P) = 1/2 - |P| + \frac{1}{|e^{|P|} - 1|} = 1/2 \coth \left(\frac{|P|}{2}\right) - \frac{1}{|P|}.$$  

(3.24)

Next we will rewrite inequality (3.21), assuming $\nu = 1, \theta = 1/2$ (worst-case scenario), to show that this restriction is not always necessary.

$$|c|/2 \coth \left(\frac{|P|}{2}\right) + |1/2 - W(P)| \leq 1 \iff$$

$$|c|/2 \coth \left(\frac{|P|}{2}\right) + \frac{1}{2} \coth \left(\frac{|P|}{2}\right) - \frac{1}{|P|} \leq 1 \iff$$

$$1/2 \coth \left(\frac{|c|}{2d}\right) (|c| + 1) - \frac{d}{|c|} - \frac{d|c|}{|c|} + \frac{d|c|}{|c|} \leq 1 \iff$$

$$\left(\frac{1}{2} \coth \left(\frac{|c|}{2d}\right) - \frac{d}{|c|}\right) (|c| + 1) + \frac{d|c|}{|c|} \leq 1 \iff$$

$$\left(\frac{1}{2} - W\left(\frac{|c|}{d}\right)\right) (|c| + 1) + d \leq 1.$$  

(3.25)

Unfortunately we have to analyse function $W$. Inequality (3.25) holds if and only if

$$-W\left(\frac{|c|}{d}\right) \leq \frac{1 - d}{|c| + 1} - \frac{1}{2} \iff$$

$$-\frac{d}{|c|} + \frac{1}{e^{|c|/d} - 1} \leq \frac{1 - d - 1/2|c| - 1/2}{|c| + 1} \iff$$

$$\frac{1}{e^{|c|/d} - 1} \leq \frac{1 - d - 1/2|c| - 1/2 + d + |c|}{|c| + 1} \iff$$

$$|c| + 1 \leq \left(1 - d - 1/2|c| - 1/2 + d + \frac{d}{|c|}\right) \left(e^{|c|/d} - 1\right) \iff$$

(3.26)

Assuming $d \leq 1/2$ we have that

$$\frac{1}{2} |c| \leq \frac{1}{4} d, \quad \frac{1}{2} \frac{|c|^2}{d} \leq \frac{1}{4} \frac{|c|^2}{d^2},$$

(3.27)

and assuming $|c| \leq 1$ we have that

$$\frac{1}{4} \frac{|c|^3}{d} \leq \frac{1}{4} \frac{|c|}{d}.$$  

(3.28)
From (3.27) in combination with (3.28) follows
\[
\frac{1}{2} |c| + \frac{1}{2} |c|^2 + \frac{1}{4} |d|^2 \leq \frac{1}{4} |c| + \frac{1}{4} |c|^2 + \frac{1}{4} |d|^2 \quad \Rightarrow
\]
\[
|c| + 1 \leq \frac{1}{2} |c| + \frac{1}{4} |c|^2 + \frac{1}{2} |c| - \frac{1}{2} \frac{|c|^2}{d} - \frac{1}{4} \frac{|c|^3}{d} \quad \Rightarrow
\]
\[
|c| + 1 \leq \left( \frac{1}{2} - \frac{1}{2} |c| + \frac{d}{|c|} \right) \left( \frac{|c|}{d} + \frac{1}{2} \frac{|c|^2}{d^2} \right) \quad \Rightarrow
\]
\[
(3.29)
\]
\[
|c| + 1 \leq \left( 1 - \frac{1}{2} |c| - \frac{1}{2} + \frac{d}{|c|} \right) \left( \frac{|c|}{d} + \frac{1}{2} \frac{|c|^2}{d^2} \right) \quad \Rightarrow
\]
\[
|c| + 1 \leq \left( 1 - d - \frac{1}{2} |c| - \frac{1}{2} + d + \frac{d}{|c|} \right) \left( e^{|c|/d} - 1 \right).
\]

From (3.29) follows (3.26) and (3.25). Thus if \( d \leq 1/2 \) and \(|c| < 1\) we can now see that restriction (3.14) is not necessary anymore because \( \nu \) can not be larger than 1. Thus for a constant \(|c| < 2\), \( \nu \) is bounded by 1 and by exactly one of the two restriction (3.14), (3.16), depending on \(|c|\). Since the TCF-scheme is of higher order than the SCF-scheme we would like to choose \( \nu \) as close to 1 as possible (but not larger) while demanding that the TVD property still holds for the scheme. This can be achieved by rewriting (3.14), (3.16), choosing \( \nu \) to be
\[
\nu = \begin{cases} 
\min \left( \frac{1-d(1-\theta)(B(P)+B(-P))}{\frac{1}{2}-W(P)},1 \right), & 1 \leq |c| \leq 2 \\
\min \left( \frac{d \theta B(-|P|)}{\frac{1}{2}-W(P)},1 \right), & 0 \leq |c| \leq 1 
\end{cases}
\]
\[
(3.30)
\]
To test the bounds (3.14) and (3.16) in practice we have tried to find an initial condition where the bounds we find in our simulation match the theoretical bounds. An example that came close is Example 2.3. We only changed the initial condition to
\[
\varphi_{in} = \begin{cases} 
1 & x \in [0.3,.7] \\
0 & \text{else}
\end{cases},
\]
and left \( u \) and \( \varepsilon \) variable. In the simulation we choose \( \Delta x = \Delta t = 1/40 \) and calculated the total variation after the first timestep. The total variation of the initial condition is 2. Then parameter \( \nu \) is being raised until the TVD property does not hold (until the total variation is larger than 2). The results can be found in Table 3. In this table \( \nu_{Ex} \) is the last number such that \( TV(\varphi^1) \leq 2 \). We have increased the last digit displayed. Thus for \( c = 3/2, d = 1/2 \) and \( \nu_{Ex} = 960 \), \( TV(\varphi^1) > 2 \).

What can be seen from Table 3 is that for small diffusion number the theoretical TVD bounds appear to be exact. We do see a difference for \( c = 3/2 \). We do not think the TVD bounds are too strict, but that we did not find a good enough example where these bounds are needed. We also have discovered that when started with a
Figure 7: Restrictions (3.14) (red), (3.16) (blue) applied to $\nu$, expressed in terms of $d$, for different values of the Courant number and theta, which results in the TVD area (grey).

Figure 8: Upper bounds for $\nu$ as function of $c$, for different values of the Péclet number, which results in the TVD area (grey).
Table 3: Theoretical TVD restriction versus TVD restriction of an example.

<table>
<thead>
<tr>
<th></th>
<th>( \nu )</th>
<th>( \nu_{\text{Ex}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c = 1/2 )</td>
<td>( d = 1/8 )</td>
<td>0.947915</td>
</tr>
<tr>
<td></td>
<td>( d = 1/16 )</td>
<td>0.666294</td>
</tr>
<tr>
<td></td>
<td>( d = 1/32 )</td>
<td>0.571428</td>
</tr>
<tr>
<td>( c = 3/2 )</td>
<td>( d = 1/2 )</td>
<td>0.923077</td>
</tr>
<tr>
<td></td>
<td>( d = 1/4 )</td>
<td>0.733351</td>
</tr>
<tr>
<td></td>
<td>( d = 1/8 )</td>
<td>0.599969</td>
</tr>
</tbody>
</table>

smooth initial condition, the theoretical TVD bounds are much too strict. In the next section we will find TVD bounds which will depend on the smoothness of the solution.
4 TVD restrictions depending on the smoothness.

We have seen in the previous section that there are two different kinds of restrictions, one coming from the implicit part of the scheme and one coming from the explicit part. In this section we will try to make both bounds less strict by letting $\nu$ be dependent on the smoothness of the solution. We will focus on the advection equation where the restrictions for $\nu$ are more simple. A second important reason for this is that we expect that with diffusion the bounds also have to hold, because diffusion makes the solution more smooth. Proving the TVD property will be labor-intensive but the results at the end of this section are worth it.

For $\varepsilon = 0$ assuming $c > 0$, the implicit restriction demands $(1 - \theta)c \leq 1 - \nu/2$ while the explicit restriction demands $2\theta c \geq \nu$. With these restrictions TVD is proved when $c$ is constant, see (3.30). When using the second order TCF-scheme ($\nu = 1$) with $\theta = 1/2$, there is only 1 solution for which TVD is guaranteed, $c = 1$, while the first order SCF-scheme allows the Courant number to be between 0 and 2. Because for Courant numbers higher than 2 TVD is not guaranteed, we will assume $c \leq 2$. The goal now is to be able to use the second order TCF-scheme more often by making both types of restrictions dependent on the smoothness of the solution. This can be done by looking at the restrictions locally at each timestep. First we define $r_{j+1/2}$ as

$$r_{j+1/2} := \frac{\varphi_{j+1}^i - \varphi_j^{i-1}}{\varphi_{j+1}^i - \varphi_j^i}.$$

(4.1)

The parameter $r_{j+1/2}$ can be seen as a parameter for the smoothness of the solution. In this section we will assume $c > 0$. Because of symmetry the outcome when $c < 0$ is almost the same. The only difference is that $r_{j+1/2}$ will be defined as a right increment divided by a left increment. Next we introduce the parameter $\nu_j = \nu(r_{j-1/2}, r_{j+1/2})$, such that the parameter for combining the two schemes can be adjusted locally. This parameter will be used as a sort of flux limiter, only it is a parameter for combining the TCF-scheme with the SCF-scheme, not a flux limiter combining the TCF flux with the SCF flux. There is a small difference between both definitions. With this new definition (2.47) becomes

$$\left(\frac{1}{\Delta t} \left((I - V)\Delta x + VB\right) + \theta A\right)\varphi^{i+1} = \left(\frac{1}{\Delta t} \left((I - V)\Delta x + VB\right) - (1 - \theta)A\right)\varphi^i + \theta (Bs^{i+1} + b_s^{i+1}) + (1 - \theta) (Bs^i + b_s^i),$$

(4.2)

where $V$ is defined as a diag($\nu_j$). For simplicity we define

$$\tau_j := \frac{1}{2} \nu_j + (1 - \theta)c,$$

(4.3)

$$\zeta_j := \frac{1}{2} \nu_j - \theta c.$$
For $\varepsilon = 0$ the scheme can be written as
\[
\varphi_{j}^{i+1} - (\varphi_{j}^{i+1} - \varphi_{j-1}^{i+1})\zeta_j = \varphi_{j}^{i} - (\varphi_{j}^{i} - \varphi_{j-1}^{i})\tau_j. \tag{4.4}
\]

We will follow similar steps as previously in Section 3.2 when the TVD restrictions were derived. First (4.4) for $j - 1$ is substracted from (4.4) in order to get
\[
(\varphi_{j}^{i+1} - \varphi_{j-1}^{i+1})(1 - \zeta_j) + (\varphi_{j-1}^{i+1} - \varphi_{j-2}^{i+1})\zeta_{j-1} = \\
(\varphi_{j}^{i} - \varphi_{j-1}^{i})(1 - \tau_j) + (\varphi_{j-1}^{i} - \varphi_{j-2}^{i})\tau_{j-1}. \tag{4.5}
\]

### 4.1 Advection equation, explicit restrictions

First we focus on the explicit restrictions. For $\nu_j \leq 2 - 2(1 - \theta)c$ the TVD property is already proven. The goal is to set $2 - 2(1 - \theta)c \leq \nu_j$, preferable $\nu_j = 1$, without creating oscillations. We will assume $\theta \in [1/2, 1]$ and $2 - 2(1 - \theta)c \leq 1$. If $2 - 2(1 - \theta)c > 1$ the TVD property is already proven for $\nu_j = 1$ with the disadvantage that for $\theta > 1/2$ the scheme is of lower order in time. When $2 - 2(1 - \theta)c \leq \nu_j \leq 1$ given $1 \leq c \leq 2$ it is important to notice that $\zeta_j \leq 0$ and $\tau_j \geq 1$. Before taking the absolute value, we rewrite (4.5) as
\[
(\varphi_{j}^{i+1} - \varphi_{j-1}^{i+1})(1 - \zeta_j) + (\varphi_{j-1}^{i+1} - \varphi_{j-2}^{i+1})\zeta_{j-1} = \\
(\varphi_{j-1}^{i} - \varphi_{j-2}^{i})\left(\frac{1}{r_{j-1/2}}(1 - \tau_j) + \tau_{j-1}\right). \tag{4.6}
\]

Next, the absolute value and the sum is taken over $j$ similar to the standard TVD proof. We get the inequality
\[
\sum_j |\varphi_{j}^{i+1} - \varphi_{j-1}^{i+1}|(|1 - \zeta_j| - |\zeta_j|) \leq \\
\sum_j |\varphi_{j-1}^{i} - \varphi_{j-2}^{i}| \left(\frac{1}{r_{j-1/2}}(1 - \tau_j) + |\tau_{j-1} - 1|\right). \tag{4.7}
\]

Since $\zeta_j \leq 0$, $|1 - \zeta_j| - |\zeta_j| = 1$. We may split the right hand side of (4.7) into
\[
\sum_j |\varphi_{j}^{i+1} - \varphi_{j-1}^{i+1}| \leq \sum_j |\varphi_{j-1}^{i} - \varphi_{j-2}^{i}| \left(\frac{1}{r_{j-1/2}}(1 - \tau_j) + 1 + |\tau_{j-1} - 1|\right). \tag{4.8}
\]

For all $j$, $|\tau_{j-1} - 1|$ can now be moved to one increment lower such that (4.8) can be written as
\[
\sum_j |\varphi_{j}^{i+1} - \varphi_{j-1}^{i+1}| \leq \sum_j |\varphi_{j-1}^{i} - \varphi_{j-2}^{i}| \left(\frac{1}{r_{j-1/2}}(1 - \tau_j) + 1 + \frac{1}{r_{j-1/2}}(\tau_j - 1)\right). \tag{4.9}
\]
which proves TVD if for every $j$ it holds that
\[
\left| \frac{1}{r^i_{j-1/2}} (1 - \tau_j) + 1 \right| + \left| \frac{1}{r^i_{j-1/2}} (\tau_j - 1) \right| \leq 1, \tag{4.10}
\]
which is true if and only if
\[
\frac{1}{r^i_{j-1/2}} (1 - \tau_j) + 1 \geq 0. \tag{4.11}
\]
From (4.11) it can be obtained that for given $c$ and $r^i_{j-1/2} > 0$ it should hold that
\[
\nu_j \leq 2(r^i_{j-1/2} + 1 - (1 - \theta)c). \tag{4.12}
\]
To find a simple restriction for negative $r^i_{j-1/2}$ we start again from Equation (4.7). Both terms of the coefficient of an increment in the right hand side of the inequality are now positive. Instead of moving $\tau_j - 1$ we now move $\tau_j - 1$ to one increment lower which gives us
\[
\sum_j |\varphi^{i+1}_j - \varphi^{i}_j| \leq \sum_j |\varphi^{i}_j - \varphi^{i-1}_j| \left( \left| \frac{1}{r^i_{j-1/2}} (1 - \tau_j) \right| + \left| \frac{1}{r^i_{j-1/2}} (\tau_j - 1) \right| \right), \tag{4.13}
\]
which proves TVD if for every $j$ it holds that
\[
\left| \frac{1}{r^i_{j-1/2}} (1 - \tau_j) \right| + \left| \frac{1}{r^i_{j-1/2}} (\tau_j - 1) \right| \leq 1. \tag{4.14}
\]
From this can be obtained that for given $c$ and $r^i_{j-1/2} < 0$ it should hold that
\[
\nu_j \leq -r^i_{j-1/2} + 1 - 2(1 - \theta)c. \tag{4.15}
\]
Two bounds have been found, one for positive $r^i_{j-1/2}$ and one for negative $r^i_{j-1/2}$. There are more options to construct bounds, but choosing these bounds has one big advantage. When $r^i_{j-1/2} < 0$ and $r^i_{j+1/2} > 0$ or visa versa no problems will occur. When $r^i_{j-1/2} < -1$ (4.13) will be used, otherwise (4.9). This way, for small $|r^i_{j-1/2}|$, $\tau_j$ can always be chosen to be 1 ($\nu_j = 2 - 2(1 - \theta)c$) such that the coefficient of each increment will always be smaller or equal than one and the TVD property is guaranteed. Thus $\nu_j$ is limited from below by $2 - 2(1 - \theta)c$ and $\nu_j$ can not be larger than 1. With these observations and the bounds (4.12) and (4.15) in mind it is now possible to construct an ideal $\nu_j$ which should be as high as possible. Thus for constant $1 \leq c \leq 2$, defining
\[
\nu_j := \begin{cases} 
1 & r^i_{j-1/2} < -2(1 - \theta)c \\
-r^i_{j-1/2} + 1 - 2(1 - \theta)c & -2(1 - \theta)c \leq r^i_{j-1/2} \leq -1 \\
-1 < r^i_{j-1/2} < 0 & -1 < r^i_{j-1/2} < 0 \\
2(r^i_{j-1/2} + 1 - (1 - \theta)c) & 0 \leq r^i_{j-1/2} \leq (1 - \theta)c - \frac{1}{2} \\
1 & r^i_{j-1/2} > (1 - \theta)c - \frac{1}{2} 
\end{cases}, \tag{4.16}
\]
Figure 9: Flux limiter $\nu_j$ plotted against $r_{j-1/2}^i$ for different values of $c$ and with $\theta = 1/2$.

the scheme is TVD. In Figure 9, $\nu_j$ is given for different values of $c$. For $c \in [-2, -1]$ similar steps as above can be done to obtain the exact same $\nu_j$. Note that it is possible to let $\nu_j$ depend on more parameters than only $r_{j-1/2}^i$ which could improve the results even more.

4.2 Advection equation, implicit restrictions

Next, we focus on the implicit restrictions, which hold when $0 < c < 1$. TVD is already proven for constant $c$ choosing $\nu_j \leq 2\theta c$. We would like to be able to prove the TVD property for $2\theta c < \nu_j \leq 1$. Again we assume $\theta \in [1/2, 1]$ and $2\theta c \leq 1$. If $2\theta c > 1$ we have $\theta > 1/2$. Thus we then already have the TVD property for $\nu_j = 1$ with the disadvantage that the scheme is of lower order in time. It is important to notice that $0 \leq \zeta_j \leq 1/2$ and $0 < \tau_j \leq 1$. This time a different strategy for the TVD proof is chosen. Equation (4.5), using $j + 1$, can be written as

$$ (\varphi_{j+1}^{i+1} - \varphi_j^i)(1 - \zeta_{j+1} + r_{j+1/2}^{i+1}\zeta_j) = (\varphi_{j+1}^i - \varphi_j^i)(1 - \tau_{j+1}) + (\varphi_j^i - \varphi_{j-1}^i)\tau_j, \quad (4.17) $$

where $r_{j+1/2}^{i+1}$ is defined as

$$ r_{j+1/2}^{i+1} := \frac{\varphi_{j+1}^{i+1} - \varphi_j^{i+1}}{\varphi_{j+1}^i - \varphi_j^i}. \quad (4.18) $$

Next the absolute value and the sum is taken over $j$. We get the inequality

$$ \sum_j |\varphi_{j+1}^{i+1} - \varphi_j^i| |1 - \zeta_{j+1} + r_{j+1/2}^{i+1}\zeta_j| \leq $$

$$ \sum_j |\varphi_{j+1}^i - \varphi_j^i||1 - \tau_{j+1}| + |\tau_{j+1}|. \quad (4.19) $$
When \( r_{j+1/2}^{i+1} \geq 0 \) it is possible to split \( |1 - \zeta_{j+1} + r_{j+1/2}^{i+1} \zeta_j| \) into \( |1 - \zeta_{j+1}| + |r_{j+1/2}^{i+1} \zeta_j| \).

Assuming \( r_{j+1/2}^{i+1} \geq 0 \) from which it follows that \( r_{j+1/2}^{i+1}/|r_{j+1/2}^{i+1}| = 1 \) we can rewrite the inequality from (4.19) into

\[
\sum_j |\varphi_{j+1}^{i+1} - \varphi_j^{i+1}| (|1 - \zeta_{j+1}| + |\zeta_{j+1}|) \leq \\
\sum_j |\varphi_{j+1}^i - \varphi_j^i| (|1 - \tau_{j+1}| + |\tau_{j+1}|). \tag{4.20}
\]

Now both the implicit as well as the explicit coefficient of the increments are equal to 1. Thus if \( r_{j+1/2}^{i+1} \geq 0 \) the TVD property is proven. Unfortunately proving \( r_{j+1/2}^{i+1} \geq 0 \) is not simple. We will use (4.5) and (4.17) for the proof. To start, three assumptions are needed. We need \( r_{j-1/2}^i \geq 0 \), \( r_{j+1/2}^i \geq 0 \) and \( \zeta_{j-1}(\varphi_{j-1}^{i+1} - \varphi_{j-2}^{i+1}) \) should have the same sign as \( \zeta_j(\varphi_j^{i+1} - \varphi_{j-1}^{i+1}) \). This last requirement can always be met by adjusting \( \nu_{j-1} \), such that \( \zeta_{j-1} = 0 \). Observe that using (4.5) and applying the first two assumptions that now all terms in (4.5) and (4.17) must have the same sign, except for the term \((\varphi_{j+1}^{i+1} - \varphi_j^{i+1})(1 - \zeta_{j+1})\). If we are able to prove that this last term also has the same sign we have that \( r_{j+1/2}^{i+1} \geq 0 \) and we are done. Let

\[
r_{j-1/2}^i \leq \frac{c}{\left(\frac{1}{2} + (1 - \theta)c\right) \zeta_j} = \frac{(1 - \zeta_j)\tau_j - \zeta_j(1 - \tau_j)}{\zeta_j(\frac{1}{2} + (1 - \theta)c)} \tag{4.21}
\]

Using \( \tau_{j-1} \leq \frac{1}{2} + (1 - \theta)c \) and assuming \( \varphi_j^i - \varphi_{j-1}^i > 0 \) we conclude from inequality (4.21) that

\[
\frac{(1 - \tau_j)(\varphi_j^i - \varphi_{j-1}^i) + \tau_{j-1}(\varphi_{j-1}^i - \varphi_{j-2}^i)}{1 - \zeta_j} \leq \frac{\tau_j(\varphi_j^i - \varphi_{j-1}^i)}{\zeta_j}. \tag{4.22}
\]

By assumption \( \zeta_{j-1}(\varphi_{j-1}^{i+1} - \varphi_{j-2}^{i+1}) \geq 0 \), thus from (4.22) it follows that

\[
\frac{(1 - \tau_j)(\varphi_j^i - \varphi_{j-1}^i) + \tau_{j-1}(\varphi_{j-1}^i - \varphi_{j-2}^i) - \zeta_{j-1}(\varphi_{j-1}^{i+1} - \varphi_{j-2}^{i+1})}{1 - \zeta_j} \leq \frac{\tau_j(\varphi_j^i - \varphi_{j-1}^i)}{\zeta_j}. \tag{4.23}
\]

Equation (4.5) is used to conclude from (4.23) that

\[
(\varphi_j^{i+1} - \varphi_{j-1}^{i+1}) \zeta_j \leq (\varphi_j^i - \varphi_{j-1}^i) \tau_j. \tag{4.24}
\]

By assumption \( \varphi_j^i - \varphi_j^i > 0 \), thus

\[
(\varphi_j^{i+1} - \varphi_{j-1}^{i+1}) \zeta_j \leq (\varphi_j^i - \varphi_{j-1}^i) \tau_j + (\varphi_j^{i+1} - \varphi_{j-1}^{i+1}) (1 - \tau_{j+1}). \tag{4.25}
\]

This implies, using (4.17), that

\[
(\varphi_j^{i+1} - \varphi_{j+1}^{i+1})(1 - \zeta_{j+1}) > 0, \tag{4.26}
\]
and thus
\[
\frac{\varphi_j^{i+1} - \varphi_j^{i+1}}{\varphi_j^{i+1} - \varphi_j^{i+1}} = r_{j+1/2}^i > 0.
\] (4.27)

Observe that the inequality \( r_{j+3/2}^i > 0 \)

are sufficient to prove that \( r_{j+3/2}^i > 0 \).

If \( \varphi_j^i - \varphi_{j-1}^i < 0 \) we can follow exactly the

same steps. Using \( \tau_{j-1} \leq \frac{1}{2} + (1 - \theta)c \) we conclude from (4.21) that

\[
\frac{(1 - \tau_j)(\varphi_j^i - \varphi_{j-1}^i) + \tau_{j-1}(\varphi_{j-1}^i - \varphi_{j-2}^i)}{1 - \zeta_j} \geq \frac{\tau_j(\varphi_j^i - \varphi_{j-1}^i)}{\zeta_j}.
\] (4.29)

By assumption \( \zeta_{j-1}(\varphi_{j-1}^{i+1} - \varphi_{j-2}^{i+1}) \leq 0 \), thus from (4.29) it follows that

\[
\frac{(1 - \tau_j)(\varphi_j^i - \varphi_{j-1}^i) + \tau_{j-1}(\varphi_{j-1}^i - \varphi_{j-2}^i) - \zeta_{j-1}(\varphi_{j-1}^{i+1} - \varphi_{j-2}^{i+1})}{1 - \zeta_j} \geq \frac{\tau_j(\varphi_j^i - \varphi_{j-1}^i)}{\zeta_j}.
\] (4.30)

Equation (4.5) is used to conclude from (4.30) that

\[
(\varphi_j^{i+1} - \varphi_{j-1}^{i+1}) \zeta_j \geq (\varphi_j^i - \varphi_{j-1}^i) \tau_j.
\] (4.31)

By assumption \( \varphi_{j+1}^i - \varphi_j^i < 0 \), thus

\[
(\varphi_j^{i+1} - \varphi_{j-1}^{i+1}) \zeta_j \geq (\varphi_j^i - \varphi_{j-1}^i) \tau_j + (\varphi_{j+1}^i - \varphi_j^i) (1 - \tau_{j+1}).
\] (4.32)

This implies, using (4.17), that

\[
(\varphi_j^{i+1} - \varphi_{j-1}^{i+1}) (1 - \zeta_{j+1}) < 0,
\] (4.33)

and thus

\[
\frac{\varphi_j^{i+1} - \varphi_{j-1}^{i+1}}{\varphi_{j+1}^{i+1} - \varphi_j^{i+1}} = r_{j+1/2}^i > 0.
\] (4.34)

Inequality (4.21) can be rewritten such that \( \nu_j \) is expressed in terms of \( r_{j-1/2}^i \). We already know \( \nu_j \) is bounded from below by \( 2\theta c \) and \( \nu_j \) can not be higher than 1. Thus for constant \( 0 \leq c \leq 1 \), defining

\[
\nu_j := \begin{cases} 
2 \left( \frac{c}{r_{j-1/2}^i (\frac{1}{2} + (1 - \theta)c)} + \theta c \right) & r_{j-1/2}^i \geq (\frac{1}{2} + (1 - \theta)c)(\frac{1}{2} - \theta c), \\
2\theta c & \text{otherwise}
\end{cases}
\] (4.35)

the scheme is TVD. In Figure 10, \( \nu_j \) is plotted. For \( c \in [-1, 0] \) similar steps as above can be done to obtain the exact same \( \nu_j \).
Figure 10: Flux limiter $\nu_j$ plotted against $r_{j-1/2}^i$ for different values of $c$ and with $\theta = 1/2$.

Figure 11: Example 2.2 at $t = 1$. In red using the flux limiter, in black analytical solution, left $\Delta x = \Delta t = 1/80$, right $\Delta x = \Delta t = 1/160$.

4.3 Results

First we solved Example 2.2 numerically using the flux limiter described in the previous paragraphs. In Figure 11 this result (red) is plotted along with the actual solution (black) using $\theta = 1/2$. We can compare this result with the results in Figure 4. We can conclude that the small oscillations of the TCF-scheme are gone but there is still some error. When using the flux limiter the results seem much better than both TCF and SCF-scheme. The error still made can be explained by looking at the source function. In Figure 12 a top view of the source function is plotted, left with a low resolution grid and at the right-hand side with a high resolution grid. It is clear to see that because of the low grid size, details of the source function are lost.

Example 2.3 is redone, now by using the flux limiter described in the previous paragraphs. The results can be found in Figure 13. When using a flux limiter, there are no spurious oscillations. Using the flux limiter dependent on $r_{j-1/2}^i$, the result is
Figure 12: Top view of the source function $s$, left $\Delta x = \Delta t = 1/80$, right $\Delta x = \Delta t = 1/160$.

<table>
<thead>
<tr>
<th>$h^{-1}$</th>
<th>#gridpoints</th>
<th>$|e^{n-1}|_{1}/n_x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>100</td>
<td>0.1394</td>
</tr>
<tr>
<td>20</td>
<td>400</td>
<td>0.1049</td>
</tr>
<tr>
<td>40</td>
<td>1600</td>
<td>0.0763</td>
</tr>
<tr>
<td>80</td>
<td>6400</td>
<td>0.0431</td>
</tr>
<tr>
<td>160</td>
<td>25600</td>
<td>0.0264</td>
</tr>
<tr>
<td>320</td>
<td>102400</td>
<td>0.0168</td>
</tr>
<tr>
<td>640</td>
<td>409600</td>
<td>0.0108</td>
</tr>
<tr>
<td>1280</td>
<td>1638400</td>
<td>0.0069</td>
</tr>
</tbody>
</table>

Table 4: The error against the number of gridpoints using the flux limiter.

closest to the actual solution, but is nonsymmetric. In Table 4 the error is plotted for various gridsizes. Here

$$e^{n-1} := \varphi^{n-1} - \varphi(x, 1).$$

Also this error seems to go to zero. Later we will compare this table with the results from using grid refinement.
Figure 13: Solution (blue) plotted with numerical solution using SCF-scheme (left-top), TCF-scheme (right-top) and combined SCF/TCF-scheme using (4.16) (left-under) and (4.35) (right-under), all with $\Delta x = \Delta t = 1/160$. 
5 Adaptive grid refinement

In this section we will introduce a grid refinement method. First we introduce the idea of the method in arbitrary space dimensions. We have programmed the method in one dimension. The method required two main functions, a function that is able to solve the advection-diffusion-reaction equation for one timestep and a recursive function. For both functions the pseudocode is given. The results can be found at the end of this section.

5.1 Basic method

We assume that there is an equation of type (2.1) that has to be solved in $d$ spatial dimensions and time, with given bounded functions $s := s(x, t)$, $u := u(x, t)$ and $\varepsilon := \varepsilon(x, t)$. We also assume the domain of (2.1) to be $\Omega \times [0, T]$ for $T \in \mathbb{R}^+$, where $\Omega := [0, 1]^d$ and assume either $u(x, t) \geq 0$ or $u(x, t) \leq 0$ for all $(x, t) \in \Omega \times [0, T]$. The domain can be divided into equally large cells. This is done by first dividing each axis of the spatial domain into $n_x$ equidistant points. The distance between two adjacent points in space is defined as $\Delta x := 1/(n_x - 1)$. The time axis is divided into $n_t$ points. The distance between two adjacent points in time is defined as $\Delta t := \bar{T}/(n_t - 1)$ for some $\bar{T} \in \mathbb{R}^+$. A numerical scheme will be used in this adaptive method that requires

$$|c| = |u| \frac{\Delta t}{\Delta x} \in [0, 2],$$

and performs best if $|c|$ is close to 1. This can be achieved by setting

$$n_t = \text{Round} \left( \frac{T}{\Delta x \left( \min_{\Omega \times [0,T]} u + \max_{\Omega \times [0,T]} u \right)} \right) + 1,$$

where Round() rounds a real value to the nearest integer. If a solution is required at the exact time $T$, $\Delta t$ can be adjusted such that $T$ is divisible by $\Delta t$. This small difference is negligible. Now let $j_k \in \{0, ..., n_x - 1\}$ and $i \in \{0, ..., n_t - 1\}$, then each gridpoint in space and time can be written as

$$(x_j, t_i) := (j_1 \Delta x, ..., j_d \Delta x, i \Delta t).$$

A cell is defined as the interior of a region enclosed by gridpoints from the set ${}(x_{j+k}, t_i), (x_{j+k}, t_{i+1}) \mid k \in \{0, 1\}^d {}}}$. We assume there is already a function that can solve the discretized version of (2.1) for one timestep using an implicit and/or explicit scheme which only uses values from the surrounding gridpoints. Thus $\varphi_j^{i+1}$, defined as the numerical solution at $x_j^{i+1}$, is calculated using values from a maximum of $2 \cdot 3^d - 1$ other gridpoints, $3^d - 1$ at the old time level and $3^d$ at the new time level. See Figure 14 for a schematic overview. The advantage of such a small coupling is that solving the linear system is much faster, because the matrices have a smaller bandwidth.
The grid refinement method has two main parameters, \( N \) and \( M \). First the axes from the spatial domain are each divided into \( n_x = N \) equidistant points. The parameter \( N \) should be high enough to make the global structure of the solution visible, but low enough to be calculated quickly. With given \( T \) and \( n_x \), \( n_t \) is fixed. There are now \( n_t - 1 \) timesteps that have to be calculated. For each timestep the method calls a recursive function \( r \). This function first calculates the solution after one timestep using the given solver. For each calculated cell (calculated solutions at the points surrounding the cell at the new time level) the function \( r \) checks if the solution is smooth enough. This can be done in several ways which will be described later in more detail. If the solution around a cell is not smooth enough, this cell is divided into \((M-1)^d+1\) subcells. For each group of connected cells that is not smooth enough, and for each timestep, the function \( r \) is called to calculate a more detailed solution. This process repeats itself until all cells are smooth enough, or a maximum depth has been reached. In Figure 15 an explanation is given of the first few steps of the grid refinement algorithm in 1D.

### 5.2 Combined TCF/SCF solver using flux limiter

First the implementation of a solver that can solve Equation (2.1) in 1D will be described. The solver requires initial conditions \( \varphi^{\text{old}} \) and two boundary conditions \( \varphi^{\text{new}}_1 \) and \( \varphi^{\text{new}}_n \) (Dirichlet). The parameter \( n \) is the number of spatial gridpoints from which two are part of the boundary. Thus eventually this function ends up with a system of linear equations of size \( n - 2 \). The parameter \( x_l \) and \( x_r \) are the \( x \)-coordinates of the left and right boundary respectively. The parameter \( t_{\text{old}} \) is the time-coordinate of the initial conditions and \( t_{\text{new}} \) is the time-coordinate of the to be calculated solution set. We assume \( t_{\text{new}} > t_{\text{old}} \), \( x_r > x_l \) and \( n > 2 \).

**Algorithm 5.1.**

```plaintext
input: TCFSCF(\( x_l, x_r, t_{\text{old}}, t_{\text{new}}, n, \varphi^{\text{new}}_1, \varphi^{\text{new}}_n, \varphi^{\text{old}} \))
```
A recursive function is called to calculate a (detailed) numerical solution in each timeinterval \([i, i + 1]\).

Boundary conditions and initial conditions are being used when available, or are being calculated using simple linear interpolation.

The solution at the new timelevel is being calculated.

For each cell it is checked whether the solution around that cell is smooth enough.

Groups are being formed of ‘bad’ cells (red) that are connected.

They will be divided into \((M - 1)^{d+1}\) smaller cells.

Figure 15: First steps of the grid refinement algorithm in one dimension explained with \(N = 9\), \(T = 1\), \(M = 4\), \(c = 1\) and maximum depth of 2.
Construct matrices $A$ and $B$, see Section 2.2

Use Dirichlet boundary conditions:

- $b_{\text{old}} := (0, 0, ..., 0)$ of length $n - 2$
- $b_{\text{new}} := (0, 0, ..., 0)$ of length $n - 2$
- $b_{\text{old}}(1) = h\gamma_1/2s_1^{\text{old}} + \alpha_1/2\varphi_1^{\text{old}}$
- $b_{\text{old}}(n - 2) = -h\delta_{n-1}/2s_n^{\text{old}} + \beta_{n-1}/2\varphi_n^{\text{old}}$
- $b_{\text{new}}(1) = h\gamma_1/2s_1^{\text{new}} + \alpha_1/2\varphi_1^{\text{new}}$
- $b_{\text{new}}(n - 2) = -h\delta_{n-1}/2s_n^{\text{new}} + \beta_{n-1}/2\varphi_n^{\text{new}}$

Construct flux limiter $\nu$ like defined in Section 4

Solve (4.2)

output: Solution $\varphi_{\text{new}}$ between $x_l$ and $x_r$.

The matrices $A$ and $B$ are constructed as described in Section 2.2. The vector $b$ is described for the first time in detail, because this time the boundary conditions are specified. The flux limiter is constructed as specified in Section 4 and the system is solved using (4.2).

When solving Equation (2.1) in one dimension numerically without the adaptive gridsize method, this algorithm can simply be called $n_t - 1$ times. However, the method can be much faster when matrices $A$ and $B$ are precalculated. When using the grid refinement method, this is not an option, because the matrices change. However, it could be faster to precalculate the coefficients of the matrices $A$ and $B$, already for the maximum depth of refinement, still assuming $u$ and $\varepsilon$ not depending on $t$. This is not implemented. We have made the solver in a program called Matlab, the code can be found in Appendix A. Note that in contrast to Algorithm 5.1, in this code, $u$ and $\varepsilon$ may depend on $t$. As a consequence both matrices $A$ and $B$ have to be calculated at $t_{\text{old}}$ and $t_{\text{new}}$.

### 5.3 Function r

In this paragraph the one dimension version of the recursive function $r$ will be described in full detail. In pseudocode $r$ is described in Algorithm 5.2. The most important input of this function is the solution set sol. This set contains all numerical solutions needed with their corresponding $x$ and $t$ values. The parameter $n$ is the number of spatial gridpoints of the row of $n - 1$ cells being looked at. The area $[x_l, x_r] \times [t_{\text{old}}, t_{\text{new}}]$ contains the $n - 1$ cells and the output of $r$ contains all new numerical values in this area. The parameter $md$ is simple a positive integer which
Figure 16: The row of grey cells is being updated. The blue dots are the known solutions, the red dots are unknown boundary conditions. Linear interpolation is used to calculate them.

represents the depth of the recursion.

First, a solution for \( t_{\text{new}} \) will be calculated, see Figure 16 for an example. Before doing so, boundary conditions and initial condition are needed. Often they do not exist in the solution set. Since there is always a solution for \(( t_{\text{old}}, x_l), ( t_{\text{old}}, x_r)\) in the set, initial conditions can easily be calculated using linear interpolation. Since there is also always a solution for \(( t, x_l), ( t, x_r)\) in the set, for \( t > t_{\text{new}} \), also the two boundary conditions can be calculated using linear interpolation if needed. In Figure 16 the two red dots are unknown, and will be calculated using the two blue dots above and below the red dots. The initial condition is in this case completely known, because they have been calculated already in the cells below.

**Algorithm 5.2.**

**input:** \( r(\text{sol}, M, n, x_l, x_r, t_{\text{old}}, t_{\text{new}}, \text{md}) \)

Get boundary conditions \( \varphi_{1,\text{new}}^t, \varphi_{n,\text{new}}^t \) from sol

Use linear interpolation if boundary conditions do not exists and add to solution set

Get initial conditions \( \varphi^t_{\text{old}} \) from sol

Use linear interpolation if initial conditions do not exists and add to solution set

Get solutions at time \( t_{\text{new}} \) using Algorithm 5.1 with input:

\( x_l, x_r, t_{\text{old}}, t_{\text{new}}, (M-1)(n-1)+1, \varphi_{1,\text{new}}^t, \varphi_{n,\text{new}}^t, \varphi_{\text{old}}^t \)

**if** \( \text{md} \) is smaller than a maximum depth

Use function to check which cells need refinement

Form groups of connected cells which need refinement

for each group \( j \) define:

\( n_j \) is cell length+1 of group \( j \)

\([x_{lj}, x_{rj}]\) is the interval of group \( j \)
Figure 17: The 8 sets of three points from which the second derivative is calculated which is used to determine whether the highlighted cell C needs refinement.

\[ \Delta t_{md} := (t_{new} - t_{old})/(M - 1) \]

Refine each group of cells and add new solutions:

for \( j = 1 \) to amount of cell groups

for \( i = 1 \) to \( M - 1 \)

update sol with \( r(sol, M, n_j, x_{ij}, x_{ij}, t_{old} + (i - 1)\Delta t_{md}, t_{old} + i\Delta t_{md}, md + 1) \)

end

end

output: sol updated with solutions in \([x_l, x_r] \times [t_{old}, t_{new}]\)

After the new numerical solutions have been calculated and if the maximum depth has not been reached, each cell will be checked for refinement. There are some possibilities for a function to decide whether to refine a cell or not. We have chosen to make a function dependent on a set of second derivatives approximations calculated from already calculated points around the cell. When available four sets of horizontal, two sets of diagonal and two sets of vertical points are taken, see Figure 17. So when available the set contains time derivatives, space derivatives and mixed derivatives. When they are not available they will not be used. This is not a problem because if they are not available, we are at the boundary of the refinement area. Thus for each cell \( j \) a maximum of 8 sets of three numerical solutions is obtained, saved in a 3 by
maximum 8 matrix which will be called $C_j$. We define
\[ C(j) := \|C_j(-1, 2, -1)^T\|_\infty. \] (5.4)

Let $\eta \in [0, 1]$. Cell $j$ will be marked to refine if
\[ \max\{C(j - 1), C(j), C(j + 1)\} > \eta \max_k C(k). \] (5.5)

Then each group of connected marked cells will be divided into smaller cells and for each timelevel, function $r$ is used for these groups. If $\eta = 1$ only a few cells will be refined, while with $\eta = 0$ all cells will be refined. Throughout many simulations with numerous examples we have discovered that $\eta \approx 1/5$ seems overall to give the best results.

We have chosen to let the maximum depth and $\eta$ be fixed. There could be room for improvement by for example making $\eta$ dependent on the maximum depth, or make the maximum depth dependent on the ratio cells that need refinement against the total number of cells. In higher dimensions the function $r$ can be similar. More second derivatives have to be taken to check a cell. It will probably be more work to find the groups of connected cells.

### 5.4 Results

We will give the results of three examples, all with $\eta = 1/5$, to show that the method works independent of the initial condition. Also in all examples $\Delta t$ is not adjusted such that $c$ is close to $1$. The reason for this is that especially for constant $u$ this would lead to trivial examples. If $c = 1$ the numerical solution at time level $i+1$ is simply copied from the numerical solution at time level $i$. We hope the three examples give a complete picture of the capabilities of this method. All simulations are done with the program Matlab [11], and the code can be found in Appendix A.

First the adaptive gridsize method is tested with Example 2.1. In Figure 18 the result is shown using $N = 11$, $M = 3$ and $\eta = 1/5$. In the image on the right-hand side the full numerical solution is displayed. In the left image a top view is displayed of the space time domain. The black dots are representing the 20,279 gridpoints. There are four levels of refinement, $\Delta x_{md} = (1/10)2^{-md}$, where $\Delta x_{md}$ is the distance between two gridpoints in space at refinement level $md$. This is a very smooth solution and one can see that almost everywhere full refinement is applied. In the middle to the top of the left image in Figure 18, less points are used. This is because the inflection point is located there, and thus the second derivative is low. Grid refinement does not seem to be necessary in this example, because the solution is very smooth.

Next, Example 2.3 is solved numerically using the adaptive gridsize method. In Figure 19 a result can be found using parameters $N = 11$, $m = 3$, $\eta = 1/5$, a maximum depth of 5 and using 15,581 gridpoints. To compare this results with the earlier results without the grid refinement we define
\[ e^i_j := \varphi(x_j, t_i) - \varphi^i_j, \] (5.6)
Figure 18: Numerical solution of Example 2.1 using the adaptive gridsize method with \( N = 11, M = 3, \eta = 1/5 \), a maximum depth of 4 and using 20,279 gridpoints.

Figure 19: Numerical solution of Example 2.3 using the adaptive gridsize method with \( N = 11, M = 3, \eta = 1/5 \), a maximum depth of 5 and using 15,581 gridpoints.
where $\varphi_s$ is the analytical solution. Since the numerical solution of the adaptive gridsize method does not have a constant $\Delta x$ we have used linear interpolation on the numerical solution to get a set of solutions on equidistant gridpoints in space. In Table 5 the results are given for different maximum depths. This table is compared with Table 4 in Figure 20. Here it is clear that the adaptive gridsize method can get the same error with a fraction of the gridpoints used when not using the grid refinement. Let $\Delta x_{md}$ be the distant between two adjacent points at maximum depth. The error is almost exactly the same when comparing the results where the maximum depth with grid refinement equals $h$ in Table 4. This means grid refinement is only applied where needed.

In Figure 21 some plots are given of the distribution of the gridpoints in space and time for different maximum depths. This gives a nice overview of how the distribution evolves when more depth is allowed. Note that comparing gridpoints

<table>
<thead>
<tr>
<th>md</th>
<th>#gridpoints</th>
<th>$|e^n|_1/n_X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>0.1394</td>
</tr>
<tr>
<td>1</td>
<td>369</td>
<td>0.1090</td>
</tr>
<tr>
<td>2</td>
<td>950</td>
<td>0.0787</td>
</tr>
<tr>
<td>3</td>
<td>2400</td>
<td>0.0441</td>
</tr>
<tr>
<td>4</td>
<td>6341</td>
<td>0.0266</td>
</tr>
<tr>
<td>5</td>
<td>15581</td>
<td>0.0168</td>
</tr>
<tr>
<td>6</td>
<td>34894</td>
<td>0.0107</td>
</tr>
<tr>
<td>7</td>
<td>77305</td>
<td>0.0067</td>
</tr>
<tr>
<td>8</td>
<td>172805</td>
<td>0.0040</td>
</tr>
<tr>
<td>9</td>
<td>392123</td>
<td>0.0022</td>
</tr>
</tbody>
</table>

Table 5: The error against the number of gridpoints.

Figure 20: Table 4 compared with Table 5
is not completely fair. When for example 15,581 gridpoints where distributed, the method calculated about 20,000 numerical solutions. This is still better, and the adaptive method calls the solver approximately 1300 times with an average size of approximately 15 while normally the solver is used \(n_t\) times with an length of \(n_x\), where in our examples \(n_t = n_x\). We assume solving linear systems twice as big costs at least twice the time, which means this is an extra advantage of the adaptive gridsize method. Especially in higher dimensions where the matrix of the linear system has a broader band.

The last example will be Example 2.2. The results are displayed in Figure 22 where the adaptive gridsize method is used with \(N = 11, M = 3, \eta = 1/5\), a maximum depth of 4 and using 25,914 gridpoints. In the image on the right-hand side of Figure 22 the error is plotted, since the real solution is known. Despite the use of the flux limiter, the numerical solution still becomes negative which was already mentioned in Section 4.3 and is probably caused by the steep function \(s\). The adaptive grid refinement method does not change this, but does improve the results a lot.
Figure 22: Adaptive gridsize method applied to Example 2.2 with \( N = 11, M = 3, \eta = 1/5 \) using 25,914 gridpoints, the numerical solution (left) and the error (right).
6 Conclusions and discussion

We have tested two variants of the complete flux scheme on various examples. We have discovered a difference between the TCF-scheme and the SCF-scheme. With advection dominated flows, the SCF-scheme is first order while the TCF-scheme probably is second order. For diffusion dominated flow both schemes are probably second order. Still the TCF-scheme is not necessarily better. In some examples spurious oscillations were generated by this scheme. We have strongly the idea that these oscillations are caused by fact that linear numerical schemes for solving partial differential equations, having the property of not generating new extrema, can be at most first-order accurate (Godunov’s order barrier theorem), [8] p. 341.

We have created a combination of the TCF/SCF-scheme with parameter $\nu$ and found bounds for $\nu$ such that for the one dimensional advection equation with constant $u$ the scheme has the TVD property. Even stricter bounds have been found by letting $\nu$ be dependent on the smoothness of the function. We have the idea, but this is not yet proven, that these bounds are also applicable for the one dimensional advection-diffusion equation. For the complete equation with nonconstant parameters oscillations can always occur depending on $s$, $\varepsilon$ and $u$. Supported by Example 2.2 we think that the restrictions derived are also applicable to the most general one dimensional equation. The flux limiter seems to be able to distinguish the spurious oscillations from the ones that are actually part of the solution and remove them. Since TVD can never be proven for the whole equation, other methods have to be found to prove this. Also for higher dimensions there is more work to be done.

A basic adaptive refinement method was developed and tested on numerous one dimensional examples. The method works well and uses recursion which makes it a very simple method. The next step would be to program the method for higher dimensions. We think that in two and especially in three dimensions the method would really distinguish itself because in higher dimensions there is a much higher probability that there exists non-connected areas where more resolution is needed. Also the advantage of solving many small linear systems instead of a few large linear systems is there because the matrices have a larger bandwidth in three dimensions and they are already much larger by itself.
References


A Matlab code

A.1 Solver

```matlab
function [sol] = TCFSCFsolver(n, x0, x1, t0, t1, sol, bc)
%Function for solving 1 timestep with two dirichlet ...
%boundary conditions

%Parameters
nu=1; %Option for when not using a Flux Limiter
theta=.5; %Theta method for time integration
FL=1; %Using Flux Limiter 1: yes 0:No

%n>2 (n including boundary points)
h=(x1-x0)/(n-1);
ht=t1-t0;
x=x0:h:x1;

%Construct the matrices A and B and vector b
%At time t0
A0=zeros(n-2, n-2)+diag(Calpha(x(2:n-1),x(3:n), t0)+Cbeta...(x(1:n-2),x(2:n-1), t0))--diag(Calpha(x(2:n-2),x(3:n...-1), t0), -1)--diag(Cbeta(x(3:n-1),x(4:n), t0), 1);
B0=zeros(n-2, n-2)+diag(-h*Cgamma(x(2:n-1),x(3:n), t0)+h*...Calpha(x(1:n-2),x(2:n-1), t0)+h*ones(n-2, 1))+h*diag(...Cgamma(x(2:n-2),x(3:n-1), t0), -1)+h*diag(Calpha(x(3:n-1)...,x(4:n), t0), 1);
b0=zeros(n-2, 1);

%At time t1
A1=zeros(n-2, n-2)+diag(Calpha(x(2:n-1),x(3:n), t1)+Cbeta...(x(1:n-2),x(2:n-1), t1))--diag(Calpha(x(2:n-2),x(3:n...-1), t1), -1)--diag(Cbeta(x(3:n-1),x(4:n), t1), 1);
B1=zeros(n-2, n-2)+diag(-h*Cgamma(x(2:n-1),x(3:n), t1)+h*...Calpha(x(1:n-2),x(2:n-1), t1)+h*ones(n-2, 1))+h*diag(...Cgamma(x(2:n-2),x(3:n-1), t1), -1)+h*diag(Calpha(x(3:n-1)...,x(4:n), t1), 1);
b1=zeros(n-2, 1);
```

```
%Construct flux limiter
if (FL==1);

%Smoothness parameter
r=[0, 0, ((sol(2:n-1)-sol(1:n-2))/(sol(3:n)-sol(2:n-1)))'];

%Fluxlimiter basic
nu=min(2*theta*abs(Fu(x, t0)), 2-2*(1-theta)*abs(Fu(x... , t0)));

%Fluxlimiter dependent on r
for j=3:n;
    if (Fu(x(j), t0)>1);
        if (r(j)>0);
            Con=2*(r(j)+1-(1-theta)*Fu(x(j), t0));
            nu(j)=min(max(Con, 2-2*(1-theta)*abs(Fu(x... (j), t0))), 1);
        end
    else
        if (r(j)≥0);
            Con=2*(((Fu(x(j), t0))/(r(j)*(.5+(1-theta)... *Fu(x(j), t0))))+theta*Fu(x(j), t0));
            nu(j)=min(max(Con, 2*theta*abs(Fu(x(j), ... t0))), 1);
        end
    end
else
    nu=nu*ones(length(x), 1);
end
nu=nu(2:n-1);

%Construct form Ax=b
I=h*diag(diag(ones(n-2)));
C0=(I+bsxfun(@times,nu,(B0-I)));
C1=(I+bsxfun(@times,nu,(B1-I)));
Apart=(1/ht).*C0+theta.*A1;
bpart=(1/ht).*C0*sol(2:n-1)+theta.*(C1*Fs(x(2:n-1),t1)+b1... )+(1-theta).*(-A0*sol(2:n-1)+C0*Fs(x(2:n-1),t0)+b0);

%Solving
sol=(Apart\bpart)';
end
A.2 Function $	ext{r}$

```matlab
function [sol] = rec(n, x0, x1, t0, t1, sol, m, depth)

%parameters
MaximumDepth=5;

h=(x1-x0)/(m*(n-1));
x=x0:h:x1;

%Setting up parameters for calculation
%First the boundary conditions
if (x0==0);
    bc=FBCL(x0, t1);
else
    bc=getPhit(sol, t1, x0);
end
if (x1==1);
    bc=[bc, FBCR(x1, t1)];
else
    bc=[bc, getPhit(sol, t1, x1)];
end

%Add boundary to solution if it does not exists, is ... 
% needed in later stage
sol=M(sol, [t1, t1; x0, x1; bc]);

%Calculate initial conditions
sol_d=zeros(m*(n-1)+1, 1);
for i=1:m*(n-1)+1;
    if (t0==0);
        %Calculate average solution
        sol_d(i)=FIC(x(i), t0);
        %Add this to the solution set
        sol=M(sol, [0; x0+(i-1)*h; sol_d(i)]);
    else
        sol_d(i)=getPhix(sol, t0, x0+(i-1)*h);
        %If new, add to solutions
        sol=M(sol, [t0; x0+(i-1)*h; sol_d(i)]);
    end
end

%Calculate solution
sol=M(sol, [t1*ones(1, m*(n-1)-1); x(2:m*(n-1)) ;... 
            TCFSCFsolver(m*(n-1)+1, x0, x1, t0, t1, sol_d, bc)]);```
if(depth<MaximumDepth);

%Smooth Check (returns for example [0, 0, 1, 1, 0, 1,...
  1, 1])
checkedcells=smoothcheck(sol, x0, x1, t0, t1, h, m, n...
, depth);

%Refine 'bad' cells per connected group
start=0;
for j=1:m*(n-1);
  if(checkedcells(j)==0 && start\neq j-1);
    for i=1:n-1;
      sol=rec(n, x0+(start)*h, x0+(j-1)*h, t0+(...
       i-1)*(t1-t0)/(n-1), t0+(i)*(t1-t0)/(n...
       -1), sol, j-start-1, depth+1);
    end
    start=j;
  elseif(checkedcells(j)==0 && start==j-1);
    start=j;
  elseif(j==m*(n-1));
    for i=1:n-1;
      sol=rec(n, x0+(start)*h, x1, t0+(i-1)*(t1...
       -t0)/(n-1), t0+(i)*(t1-t0)/(n-1), sol...
       , j-start, depth+1);
    end
  end
end