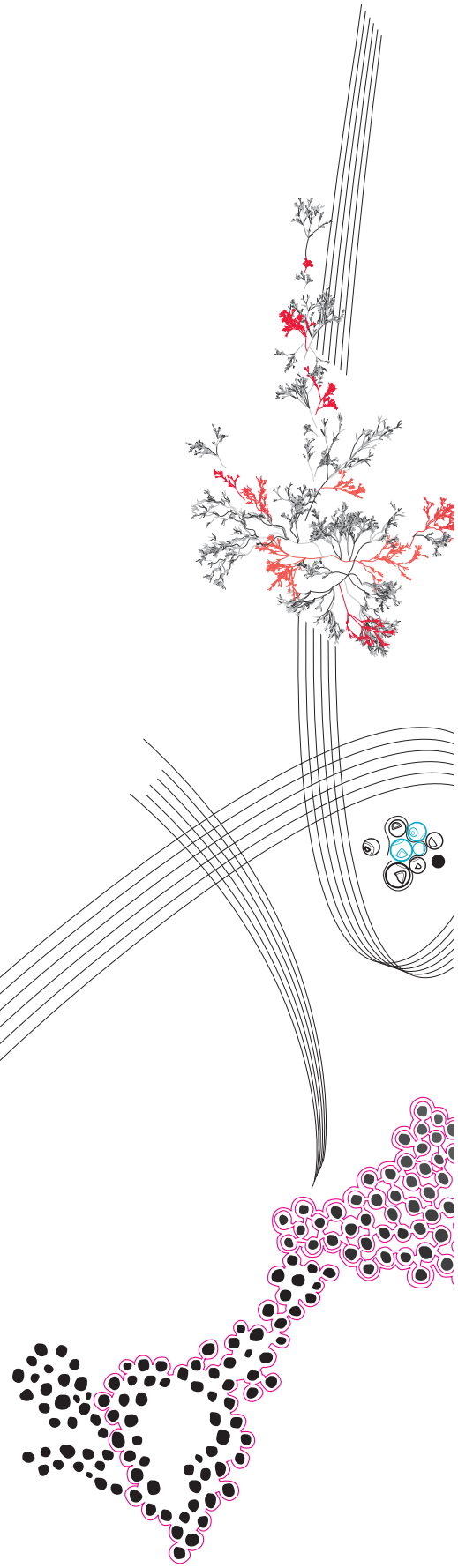


BSc Thesis Applied Mathematics

Bound-preserving limiting methods for Hybrid-DG discretization of linear advection equations

Bram J. D. Kanger

Supervisor: P. L. Lederer

July, 2024

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Bound-preserving limiting methods for Hybrid-DG discretization of linear advection equations

Bram. J. D. Kanger*

July, 2024

Abstract

In many fields of physics, using numerical methods to solve systems described by partial differential equations play a crucial role. These solutions must follow certain laws of physics in order to have meaning. A solution u to a scalar conservation law is bounded when $u_* \leq u \leq u^*$ for some global bounds u_* and u^*. Yet, a numerical solution might violate these bounds. We can enforce these bounds for element averages by properly defining numerical fluxes between elements and limiting them. In this work, we formulate several methods for calculating numerical fluxes and how to limit them to ensure mass and bound preservation for HDG solutions. We extend the results of the work "Bound-preserving flux limiting schemes for DG discretizations of conservation laws with applications to the Cahn–Hilliard equation", by Florian Frank, Andreas Rupp and Dmitri Kuzmin (CMAME, 2020), by modifying their fractional step limiter procedure. We also define two separate slope limiters to ensure a point-wise bound preserved solution. This fractional step limiter (FSL) uses an implicit time scheme to calculate a higher and lower order solution, which is then used to calculate the limiting factors. Both the FSL and the slope limiters work for multidimensional problems.

1 Introduction

Solving partial differential equations play an incredibly important role in many fields of engineering. These partial differential equations can often not be solved analytically. Thus, we must resort to numerical methods if we wish to find the solution. The solution for practical uses often represents a physical quantity, such as a concentration or a density. These quantities must follow the laws of physics that often take the form of a lower, or upper bound (bound preserving). Often, the solution must also "conserve mass" i.e., the total solution over the entire space must remain constant throughout time. In some cases, the numerical solution violates these so-called conservation laws.

In this work, we expand on the flux limiting schemes [3] by adapting the flux limiting scheme to be applicable on the hybrid discontinuous Galerkin method. We enforce the solution to bound-preserving by limiting the mass that flows in, or out of, otherwise bound violating, elements. We introduce a "fractional step" limiter (FSL) which ensures bound preservation of local means. The FSL uses non-limited higher and lower order solutions in order to make an approximation of the limiting factors. These limiting factors decrease the numerical flux in or out of "troubled" elements.

*Email: b.j.d.kanger@student.utwente.nl

1.1 Discontinuous Galerkin method

The discontinuous Galerkin (DG) method is a method that uses polynomials to approximate a solution on the closure of elements of a given triangulation. This means that a solution resulting from the DG method is not uniquely defined on intersection of elements. The DG method uses numerical fluxes to transport mass across elements. Implementing conservation laws becomes much less complicated as we can directly control these numerical fluxes.

1.2 Hybrid discontinuous Galerkin formulation

The hybrid discontinuous Galerkin (HDG) method is quite similar to the standard DG method (see 1.1). One big difference between these two methods is that in contrary to the DG method, the HDG method also approximates the solution on the vertices. The HDG method still uses numerical fluxes, however these fluxes are used to transport mass between an element and its vertices. A downside to the HDG method is that one can not use explicit Euler to calculate the solution to a time dependent problem. Hence, we must resort to an implicit Euler scheme. This however makes the implementing of some conservation laws more complicated. In the rest of this paper we use Legendre polynomials for both the DG and HDG method, as these Legendre polynomials have the useful property that they form a L^2-orthonormal basis.

2 Formalization of the problem

2.1 General formulation

We consider a flux F and the corresponding conservation law

$$\frac{\partial u}{\partial t} + \operatorname{div}(F(u)) = 0,$$

equipped with some (possibly periodic) boundary conditions and an initial condition. Now let $b(x) \in \mathbb{R}^d$ be given *wind/velocity* and let Ω be a d-dimensional bounded domain. In this work we define $F(u) = bu$, thus we solve the following problem:

$$\begin{aligned} \partial_t u(x,t) + \operatorname{div}(b\,u(x,t)) &= 0, && \text{for } (x,t) \in \Omega \times \mathbb{R}^+, \\ u(x,t) &= u_0, && \text{for } (x,t) \in \Omega \times \{0\}, \end{aligned} \tag{1}$$

for an unknown $u : \Omega \times \mathbb{R}^+ \to \mathbb{R}$, an initial condition $u_0 : \Omega \to \mathbb{R}$. If the solution $u(x,t)$ is bounded, we require

$$u_* \leq u(x,t) \leq u^* \qquad \forall (x,t) \in \Omega \times \mathbb{R}^+, \tag{2}$$

for some upper bound $u^* \in \mathbb{R}$ and lower bound $u_* \in \mathbb{R}$. We now formulate a DG and HDG scheme for general differential equations.

2.2 DG formulation

In the following section, we are going to derive a semi discrete DG formulation. For this, let $\mathcal{T}$ be a triangulation of the domain Ω into non overlapping elements $\{T\}$ such that

$$\overline{\Omega} = \bigcup_{T \in \mathcal{T}} T.$$

On each element $T \in \mathcal{T}$ we define the space of polynomials of order k by $\mathbb{P}^k(T)$. We extend this notation to via

$$\mathbb{P}^k(\mathcal{T}) := \{p_h \in L^2(\Omega) : p_h|_T \in \mathbb{P}^k(T)\ \forall T \in \mathcal{T}\}.$$

Now we choose $V_h = \mathbb{P}^1(\mathcal{T})$ and $\bar{V}_h = \mathbb{P}^0(\mathcal{T})$. The DG formulation then reads as: find $u_h \in V_h$, such that

$$\sum_T \int_T \partial_t u_h v_h \,\mathrm{d}x - \int_T F(u_h) \cdot \nabla v_h \,\mathrm{d}x + \int_{\partial T} \hat{F}(u_h) v_h \,\mathrm{d}s = 0, \quad \forall v_h \in V_h,$$

where $\hat{F}(u_h)$ is a numerical flux that it single valued at element interfaces. Here we need to use a numerical flux since the DG solution u_h has (generally) no unique value at the facets. Consider an upwinded numerical flux, thus we define

$$\hat{F}(u_h) = b \cdot n\, u_h^{up},$$

where u_h^{up} on $E \subset \partial T$ is defined as

$$u_h^{up} = \begin{cases} u_h|_T & \text{if } b \cdot n > 0, \\ u_h|_{T'} & \text{else,} \end{cases}$$

where T' is the neighbouring element of T such that $E = T \cap T'$. This then gives the problem: find $u_h \in V_h$, such that

$$\sum_T \int_T \partial_t u_h v_h \,\mathrm{d}x - \int_T (b u_h) \cdot \nabla v_h \,\mathrm{d}x + \int_{\partial T} b \cdot n\, u_h^{up} v_h \,\mathrm{d}s = 0, \quad \forall v_h \in V_h.$$

2.3 HDG formulation

We introduce a space on the skeleton of the mesh denoted by $\mathcal{F}$. Note that for $d = 1$ this gives the set of vertices shared by the element of $\mathcal{T}$. We define the space

$$\hat{V}_h = \{\hat{p}_h \in L^2(\mathcal{F}) : \hat{p}_h|_E \in P^1(E) \forall E \in \mathcal{F}\}.$$

Although defined here for the general case, for $d = 1$ this simplifies to scalars (and not! linear polynomials) defined at each vertex, thus we could equivalently define

$$\hat{V}_h := \mathbb{R}^{|\mathcal{F}|}.$$

For a $(u_h, \hat{u}_h) \in V_h \times \hat{V}_h$ we define the HDG numerical flux $\hat{F}(u_h, \hat{u}_h) = b \cdot n\,(u_h, \hat{u}_h)^{up}$ where $(u_h, \hat{u}_h)^{up}$ on $E \subset \partial T$ is defined by

$$(u_h, \hat{u}_h)^{up} = \begin{cases} \hat{u}_h & \text{if } b \cdot n \leq 0, \\ u_h & \text{else.} \end{cases}$$

For simplicity we define for a $T \in \mathcal{T}$ the outflow and inflow boundary via

$$\partial T_{in} := \{x \in \partial T : b(x) \cdot n \leq 0\}, \quad \text{and} \quad \partial T_{out} = \partial T \setminus \partial T_{in}.$$

Then the HDG formulation reads as: Find $(u_h, \hat{u}_h) \in V_h \times \hat{V}_h$ such that

$$\sum_T \int_T \partial_t u_h v_h \,\mathrm{d}x - \int_T (b u_h) \cdot \nabla v_h \,\mathrm{d}x + \int_{\partial T_{in}} b \cdot n\, \hat{u}_h v_h \,\mathrm{d}s + \int_{\partial T_{out}} b \cdot n\, u_h v_h \,\mathrm{d}s = 0, \quad \forall v_h \in V_h, \tag{3}$$

$$\sum_T \int_{\partial T_{out}} b \cdot n\,(u_h - \hat{u}_h) \hat{v}_h \,\mathrm{d}s = 0, \quad \forall \hat{v}_h \in \hat{V}_h. \tag{4}$$

2.4 Time discretization

For the time discretization of Equation (3) we use either an implicit or explicit Euler method to calculate the solution on an element for the next time step. Let $t_i + \Delta t = t_{i+1}$ be distinct time steps for $i = 0, 1, \ldots, N$ with $t_0 = 0$ and a sufficiently small time step Δt. Equation (3) reads as: Find $(u_h^{n+1}, \hat{u}_h^{n+1}) \in V_h \times \hat{V}_h$ such that

$$\begin{aligned}\sum_T \int_T (u_h^{n+1} - u_h^n) v_h \,\mathrm{d}x - \Delta t \int_T (b u_h^l) \cdot \nabla v_h \,\mathrm{d}x \\ + \Delta t \int_{\partial T_{in}} b \cdot n \, \hat{u}_h^{n+1} v_h \,\mathrm{d}s + \Delta t \int_{\partial T_{out}} b \cdot n \, u_h^l v_h \,\mathrm{d}s = 0, \quad \forall v_h \in V_h, \\ \sum_T \int_{\partial T_{out}} b \cdot n \, (u_h^l - \hat{u}_h^{n+1}) \hat{v}_h \,\mathrm{d}s = 0, \quad \forall \hat{v}_h \in \hat{V}_h.\end{aligned} \tag{5}$$

For a time implicit scheme we use $l = n + 1$ and for a time explicit scheme we use $l = n$. Using this scheme, the total mass of the solution remains the same.

Theorem 2.1 (Mass conservation)**.** *For the advection Equation* (1)*, given periodic boundary conditions, the approximation found by using Equation* (5) *satisfies*

$$\sum_T \int_T u_h^n \,\mathrm{d}x = \sum_T \int_T u_0 \,\mathrm{d}x \quad \forall n = 0, 1, \ldots, N.$$

Proof. Let $v_h = 1$ on Ω, then the first line of Equation (5) gives

$$\sum_T \int_T (u_h^{n+1} - u_h^n) \,\mathrm{d}x + \Delta t \int_{\partial T_{in}} b \cdot n \, \hat{u}_h^l \,\mathrm{d}s + \Delta t \int_{\partial T_{out}} b \cdot n \, u_h^l \,\mathrm{d}s = 0. \tag{6}$$

Next, let $\hat{v}_h = 1$ on an arbitrary interface $E = T \cap T'$. Without loss of generality let $\partial T_{out} = E$ and thus $\partial T'_{in} = E$. By the second Equation in (5) we see that on E we have $\hat{u}_h = u_h|_{T \cap E}$, and thus we have

$$\int_{T_{out}} b \cdot n \, u_h = \int_{T_{out}} b \cdot n \, \hat{u}_h = -\int_{T'_{in}} b \cdot n \, \hat{u}_h,$$

thus, the numerical flux over some outflow boundary is minus the numerical flux over that inflow boundary of the adjacent element. By that, Equation (6) simplifies to

$$\begin{aligned}&\sum_T \int_T (u_h^{n+1} - u_h^n) \,\mathrm{d}x \\ &+ \Delta t \int_{\partial T_{in}} b \cdot n \, \hat{u}_h^l \,\mathrm{d}x + \Delta t \sum_T \int_{\partial T_{in}} -b \cdot n \, \hat{u}_h^l \,\mathrm{d}x = 0\end{aligned} \tag{7}$$

Which results in

$$\sum_T \int_T u_h^{n+1} \,\mathrm{d}x - \int_T u_h^n \,\mathrm{d}x = 0.$$

This holds for all $n = 0, 1, \ldots, N$.

$$\sum_T \int_T u_h^n \,\mathrm{d}x = \sum_T \int_T u_h^0 \,\mathrm{d}x = \int_\Omega u_0 \,\mathrm{d}x.$$

□

3 Flux limiting

The next section focuses on the concept of flux limiters and how they can be applied. Flux limiting, as the name suggests, limits the amount of mass that is transported from one element to the next (flux). In a time explicit or implicit scheme we do this by multiplying our numerical flux $\hat{F}$ by a factor $\alpha_{T,T'}$ which is defined on every interface. We have that $\alpha_{T,T'} \in [0,1]$. The idea of flux limiting is only applied to a **low order HDG approximation**. Equation (5) will then read as: Given $u_0 : \Omega \to [u_*, u^*]$ and for $n = 0, 1, \ldots, N$, find $(\bar{u}_h^{n+1}, \hat{u}_h^{n+1}) \in \bar{V}_h \times \hat{V}_h$ such that

$$\begin{aligned} &\sum_T \int_T (\bar{u}_h^{n+1} - \bar{u}_h^n) v_h \, \mathrm{d}x \\ +\alpha_{T,T'} \Delta t \Big(&\int_{\partial T_{in}} b \cdot n \, \hat{u}_h^{n+1} v_h \, \mathrm{d}s + \int_{\partial T_{out}} b \cdot n \, \bar{u}_h^l v_h \, \mathrm{d}s \Big) = 0, \quad \forall v_h \in \bar{V}_h, \\ &\sum_T \int_{\partial T_{out}} b \cdot n \, (\bar{u}_h^l - \hat{u}_h^{n+1}) \hat{v}_h \, \mathrm{d}s = 0, \quad \forall \hat{v}_h \in \hat{V}_h. \end{aligned} \tag{8}$$

Theorem 2.1 also holds for the solution of Equation (8). The proof for this is quite similar. The fluxes will be multiplied by $\alpha_{T,T'}$ but the fluxes will still cancel out in the end.

3.1 Fully explicit FSL for DG schemes

We will not discuss the implementation of the fractional step limiter for the DG method in great detail as this can be found in [3]. Note that explicit FSL method do not work for the HDG method as mentioned in section 1.2.

The idea behind this FSL is that we first calculate a non-limited higher order solution, which is used as an initial guess. We then calculate a lower order solution that is bound preserving. Finally, we compare these two solutions to find how much flux, from one element to the next, we can add to the low order approximation without violating the bounds.

3.2 FSL for the HDG scheme

We now formulate an algorithm that ensures bound preservation for local means for the HDG scheme. In the following section, we will extend on this idea and modify the algorithm given in the paper [3] to work on the HDG scheme. We will use an implicit time scheme in our algorithm. This however creates a problem when trying to find the limiting factors α_T. When using an implicit scheme, the α_T's depend on the numerical fluxes of the solution in the next time step, which we do not yet know. There are two ways to circumvent this problem. The first way is; we calculate the α's iteratively in order to approximate the true value of α. The second option is using a combination of implicit and explicit time schemes.

3.2.1 Fully implicit approach

We first formulate the fully implicit method. A very important difference compared to the algorithm presented in [3], is that in step 4 we make a step from $\bar{u}_{h,T}^{(0)}$ instead of $\bar{u}_{h,T}^{(m-1)}$. In [3] we keep adding a small amount of flux on top of $\bar{u}_{h,T}^{(m-1)}$ until the next step would cause a violation of the boundedness. In Algorithm 1, we try to find a value for $\alpha_T^{(m)}$ such that we can make this step in one go. We need to find these values of $\alpha_T^{(m)}$ iteratively as

Algorithm 1 Fully implicit fractional time step limiter for bounding local means:

Given a time step $n \in \{1, 2, \dots N\}$ and u_h^{n-1}.

Step 1: Solve higher order solution: Solve Equation (5) to get a higher order solution $(u_h^{high}, \hat{u}_h^{high}) \in V_h \times \hat{V}_h$, using $l = n + 1$.
Step 2: Approximate low order solution: Solve Equation (8) with $\alpha_{T,T'} = 1$ to get a low order solution $(u_h^{low}, \hat{u}_h^{low}) \in \bar{V}_h \times \hat{V}_h$, using $l = n + 1$, that is bound preserving (see Theorem 3.2 below).
Step 3: Initialize iteration: Choose $0 < \epsilon_1, \epsilon_2 \ll 1$ and let

$$\hat{F}^{high} = \hat{F}(u_h^{high}, \hat{u}_h^{high}) \quad \text{and} \quad \hat{F}^{low} = \hat{F}(u_h^{low}, \hat{u}_h^{low}).$$

Then we set

$$m = 1 \quad \text{and} \quad \hat{F}^{(m-1)} = \hat{F}^{high} - \hat{F}^{low}.$$

Step 4: Finding the limiting factors: Use Algorithm 2 to find $\alpha_{T,T'}$ such that:

$$\begin{aligned} &\sum_T \int_T (\bar{u}_h^{(m)} - \bar{u}_h^0) v_h \, \mathrm{d}x \\ +\alpha_{T,T'} \Delta t \Big(&\int_{\partial T_{in}} b \cdot n \, \hat{u}_h^{(m)} v_h \, \mathrm{d}s + \int_{\partial T_{out}} b \cdot n \, \bar{u}_h^{(m)} v_h \, \mathrm{d}s \Big) = 0, \quad \forall v_h \in \bar{V}_h, \\ &\sum_T \int_{\partial T_{out}} b \cdot n \, (\bar{u}_h^{(m)} - \hat{u}_h^{(m)}) \hat{v}_h \, \mathrm{d}s = 0, \quad \forall \hat{v}_h \in \hat{V}_h. \end{aligned}$$

satisfies $u_* \le \bar{u}_h^{(m)} \le u^*$ on each element T. Use this equation to calculate $\bar{u}_h^{(m)}$ and $\hat{u}_h^{(m)}$.
Step 5: Calculate Fluxes:

$$\hat{F}^{(m)} = \hat{F}(\bar{u}_h^{(m)}, \hat{u}_h^{(m)})$$

Step 6: Check whether to terminate: If

$$\max_{T,T'} |\hat{F}^{high} - \hat{F}^{(m)}| < \epsilon_1 \qquad \text{or} \qquad \max_{T,T'} |\hat{F}^{(m)} - \hat{F}^{(m-1)}| < \epsilon_2 \text{ for } m > 1 \text{ only},$$

then go to Step 7, otherwise set $m \leftarrow m + 1$ go back to step 4.
Step 7: Construct final solution: for every element T, set

$$u_{h,T}^n = u_{h,T}^{high} - \bar{u}_{h,T}^{high} + \bar{u}_{h,T}^{(m)}, \quad \text{with} \quad \bar{u}_{h,T}^{high} = \frac{1}{|T|} \int_T u_{h,T}^{high},$$

thus in the final solution u_h^n we have replaced the (possibly) non bound preserving mean values $\bar{u}_{h,T}^{high}$ by the corrected values $\bar{u}_{h,T}^{(m)}$ (see Theorem 3.1 below).

the values of $\alpha_T^{(m)}$ depend on the solution $\bar{u}_h^{(m)}$, which in turn depend on $\alpha_T^{(m)}$. By making a step from $\bar{u}_h^{(0)}$ we can guarantee an convergence of $\hat{F}^{(m)}$.
Note that in step 6, we could also state a condition:

$$\max_{T,T'} |\alpha_{T,T'}^{(m)} - \alpha_{T,T'}^{(m-1)}| < \epsilon_3 \text{ for } m > 1 \text{ only,}$$

for some $0 < \epsilon_3 << 1$. However, the convergence of $\alpha_{T,T'}^{(m)}$ can not be guaranteed, so having this condition as your only terminating condition could cause an infinite loop.

Algorithm 2 Zalesak's fully implicit FCT Limiter.

This algorithm finds $\alpha_{T,T'}$ such that:

$$\begin{aligned}
&\sum_T \int_T (\bar{u}_h^{(m)} - \bar{u}_h^{(0)}) v_h \, \mathrm{d}x \\
+\alpha_{T,T'} \Delta t \Big(&\int_{\partial T_{in}} b \cdot n \, \hat{u}_h^{(m)} v_h \, \mathrm{d}s + \int_{\partial T_{out}} b \cdot n \, \bar{u}_h^{(m)} v_h \, \mathrm{d}s \Big) = 0, \quad \forall v_h \in \bar{V}_h, \\
&\sum_T \int_{\partial T_{out}} b \cdot n \, (\bar{u}_h^{(m)} - \hat{u}_h^{(m)}) \hat{v}_h \, \mathrm{d}s = 0, \quad \forall \hat{v}_h \in \hat{V}_h.
\end{aligned}$$

satisfies $u_* \leq \bar{u}_h^{(m)} \leq u^*$ on each element T.
Step 1: Sum up fluxes:

$$P_T^+ = \Delta t \sum_{T' \neq T} \max\{0, -\hat{F}_{T,T'}^{(m)}\}, \qquad P_T^- = \Delta t \sum_{T' \neq T} \min\{0, -\hat{F}_{T,T'}^{(m)}\}.$$

Step 2: Admissible bounds:

$$Q_T^+ = |T|(u^* - \bar{u}_h^{(m)}), \qquad Q_E^- = |T|(u_* - \bar{u}_h^{(m)}).$$

Step 3: Element limiting factors:

$$[0,1] \ni \alpha_T^{\pm} = \min\{1, \frac{Q_T^{\pm}}{P_T^{\pm}}\}.$$

Step 4: Flux correction factors: For $T' \neq T$,

$$\alpha_{T,T'}^{(m)} = \begin{cases} \min\{\alpha_{T'}^-, \alpha_T^+\} & \text{if } \hat{F}^{(m)} < 0, \\ \min\{\alpha_{T'}^+, \alpha_T^-\} & \text{if } \hat{F}^{(m)} > 0. \end{cases}$$

Theorem 3.1. *All $\bar{u}_{h,T}^{(m)}$ generated by Algorithm* (1) *and Algorithm* (2) *are BP, i.e., $\forall m \in \mathbb{N}, \forall T, u_* \leq \bar{u}_{h,T}^{(m)} \leq u^*$.*

Proof. We assume $u_* \leq \bar{u}_{h,T}^{(0)} \leq u^*$. We show that $\bar{u}_{h,T}^{(m)} \leq u^*$

$$\begin{aligned}
\bar{u}_{h,T}^{(m)} =& \bar{u}_{h,T}^{(0)} - \frac{\Delta t}{|T|} \sum_T \alpha_{T,T'}^{(m)} \hat{F}^{(m)}, \\
\leq& \bar{u}_{h,T}^{(0)} - \frac{\Delta t}{|T|} \sum_T \min\{\alpha_{T'}^-, \alpha_T^+\} \max\{0, -\hat{F}^{(m)}\}, \\
\leq& \bar{u}_{h,T}^{(0)} - \frac{\Delta t}{|T|} \sum_T \min\{1, \frac{Q_T^-}{P_T^-}, \frac{Q_T^+}{P_T^+}\} \max\{0, -\hat{F}^{(m)}\}.
\end{aligned}$$

We can now use the fact that $\sum_i \min\{x_i, y_i\} \leq \min\{\sum_i x_i, \sum_i y_i\}$, which gives that

$$\begin{aligned}
\bar{u}_{h,T}^{(m)} \leq \bar{u}_{h,T}^{(0)} - \min\Big\{ & \frac{\Delta t}{|T|} \sum_T \max\{0, -\hat{F}^{(m)}\}, \\
& \frac{\Delta t}{|T|} \frac{Q_T^+}{P_T^+} \sum_T \max\{0, -\hat{F}^{(m)}\}, \\
& \frac{\Delta t}{|T|} \frac{Q_T^-}{P_T^-} \sum_T \max\{0, -\hat{F}^{(m)}\} \Big\}.
\end{aligned}$$

It holds that

$$\begin{aligned}
\frac{\Delta t}{|T|} \frac{Q_T^+}{P_T^+} \sum_T \max\{0, -\hat{F}^{(m)}\} =& \frac{\Delta t |T| (u^* - \bar{u}_{h,T}^{(0)}) \sum_{T' \neq T} \max\{0, -\hat{F}^{(m)}\}}{\Delta t |T| \sum_{T' \neq T} \max\{0, -\hat{F}^{(m)}\}}, \\
=& u^* - \bar{u}_{h,T}^{(0)}.
\end{aligned}$$

This gives our final inequality:

$$\bar{u}_{h,T}^{(m)} \leq \bar{u}_{h,T}^{(0)} - u^* - \bar{u}_{h,T}^{(0)} = u^*.$$

Which gives that $\bar{u}_{h,T}^{(m)}$ is upper bounded by u^* for all m. The proof for showing that $\bar{u}_{h,T}^{(m)}$ is bounded below by u_* is very similar to this proof, where the difference is that you decompose $\hat{F}^{(m)}$ into the outflows instead of the inflows.

□

In step 2 of Algorithm 1, and correspondingly needed for the assumption of Theorem 3.1, we require the low order solution u_h^{low} to be bound preserving. We can show that when we use an implicit time scheme and given an bounded solution that the solution in our next time step is also bounded by these same bounds.

Theorem 3.2 (Preservation of non-negativity and boundness)**.** *Given that $u_* \leq \bar{u}_h^n \leq u^*$, let $\bar{u}_h^{n+1}$ be the solution of Equation 8 with $\alpha_{T,T'} = 1$. There holds:*

$$u_* \leq \bar{u}_h^{n+1} \leq u^*. \tag{9}$$

Proof. In a first step, we proceed similarly as in the proof of Theorem 2.1 to see that the hybrid variable $\hat{u}_h$ corresponds to the upwinded value. Thus, we can reformulate the problem given in Equation 8 in a DG setting. The proof then follows the same lines as in the proof from [2] but is given in the following for completeness. Further, for the sake

of simplicity, we use the same symbols for functions in $\bar{V}_h$ and their corresponding (finite element) coefficient vectors.

From the implicit method, we know:

$$M(u_h^{n+1} - u_h^n) + \Delta t C u_h^{n+1} = 0,$$

where M is the positive diagonal mass matrix, implying that all non-zero entries are on the diagonal and are positive. We rewrite this as:

$$(M + \Delta t C) u_h^{n+1} = M u_h^n.$$

As the entries of $\Delta t C$ are sufficiently small, the matrix $M + \Delta t C$ is a diagonal dominant matrix whose diagonal entries are positive. Hence, $M + \Delta t C$ is an M-matrix, which has the property that the inverse of an M-matrix has all positive entries [2]. We write our equation as:

$$u_h^{n+1} = (M + \Delta t C)^{-1} M u_h^n.$$

Where all entries in the matrix $(M + \Delta t C)^{-1}$, the matrix M and the column vector u_h^n are greater or equal to zero. Hence,

$$u_h^{n+1} = (M + \Delta t C)^{-1} M u_h^n \geq u_*.$$

Now to proof $u_h^{n+1} \leq u^*$, we know $u_h^n \leq u^*$

$$M u_h^{n+1} + \Delta t C u_h^{n+1} = M u_h^n \leq M u^*.$$

The matrix C itself is also a M-matrix, which has the property that the sum of each row is equal to 0.

$$M u_h^{n+1} \leq M u^*.$$

Which obviously gives us our final inequality

$$u_h^{n+1} \leq u^*.$$

□

The proof which shows that an explicit time scheme returns bounded solution, given a small enough time step, can be found in [4] .

3.3 Semi implicit approach

Although it is true that we can not solve the HDG scheme using an explicit time scheme, this does not mean that we can not update the mean values using an explicit time scheme. We slightly alter Algorithm 1 and Algorithm 2. We change step 4 and 5 of Algorithm 3 to instead use an explicit time scheme and a different method recycling fluxes.

Also the method to calculate $\alpha_{T,T'}^{(m-1)}$ can be changed to calculate the α's explicitly. This new version can be seen as a combination of Algorithm 1 and the algorithm presented in [3].

Algorithm 3 Semi implicit fractional time step limiter for bounding local means:

Given a time step $n \in \{1, 2, \dots N\}$ and u_h^{n-1}.

Step 1: Solve higher order solution: Solve Equation (5) to get a higher order solution $(u_h^{high}, \hat{u}_h^{high}) \in V_h \times \hat{V}_h$, using $l = n + 1$.
Step 2: Approximate low order solution: Solve Equation (8) with $\alpha_{T,T'} = 1$ to get a low order solution $(u_h^{low}, \hat{u}_h^{low}) \in \bar{V}_h \times \hat{V}_h$, using $l = n + 1$, that is bound preserving (see Theorem 3.2 below).
Step 3: Initialize iteration: Choose $0 < \epsilon_1, \epsilon_2 \ll 1$ and let

$$\hat{F}^{high} = \hat{F}(u_h^{high}, \hat{u}_h^{high}) \quad \text{and} \quad \hat{F}^{low} = \hat{F}(u_h^{low}, \hat{u}_h^{low}).$$

Then we set

$$m = 1 \quad \text{and} \quad \hat{F}^{(m-1)} = \hat{F}^{high} - \hat{F}^{low}.$$

Step 4: Finding the limiting factors: Use Algorithm 2 to find $\alpha_{T,T'}$ such that:

$$\begin{aligned} &\sum_T \int_T (\bar{u}_h^{(m)} - \bar{u}_h^{(m-1)}) v_h \, \mathrm{d}x \\ +\alpha_{T,T'} \Delta t \Big(&\int_{\partial T_{in}} b \cdot n \, \hat{u}_h^{(m)} v_h \, \mathrm{d}s + \int_{\partial T_{out}} b \cdot n \, \bar{u}_h^{(m-1)} v_h \, \mathrm{d}s \Big) = 0, \quad \forall v_h \in \bar{V}_h, \\ &\sum_T \int_{\partial T_{out}} b \cdot n \, (\bar{u}_h^{(m-1)} - \hat{u}_h^{(m)}) \hat{v}_h \, \mathrm{d}s = 0, \quad \forall \hat{v}_h \in \hat{V}_h. \end{aligned}$$

satisfies $u_* \le \bar{u}_h^{(m)} \le u^*$ on each element T. Use this equation to calculate $\bar{u}_h^{(m)}$ and $\hat{u}_h^{(m)}$.
Step 5: Calculate Fluxes:

$$\hat{F}^{(m)} = \hat{F}(\bar{u}_h^{(m)}, \hat{u}_h^{(m)})$$

Step 6: Check whether to terminate: If

$$\max_{T,T'} |\hat{F}^{high} - \hat{F}^{(m)}| < \epsilon_1 \qquad \text{or} \qquad \max_{T,T'} |\hat{F}^{(m)} - \hat{F}^{(m-1)}| < \epsilon_2 \text{ for } m > 1 \text{ only},$$

then go to Step 7, otherwise set $m \leftarrow m + 1$ go back to step 4.
Step 7: Construct final solution: for every element T, set

$$u_{h,T}^n = u_{h,T}^{high} - \bar{u}_{h,T}^{high} + \bar{u}_{h,T}^{(m)}, \quad \text{with} \quad \bar{u}_{h,T}^{high} = \frac{1}{|T|} \int_T u_{h,T}^{high},$$

thus in the final solution u_h^n we have replaced the (possibly) non bound preserving mean values $\bar{u}_{h,T}^{high}$ by the corrected values $\bar{u}_{h,T}^{(m)}$

Algorithm 4 Zalesak's semi-implicit FCT Limiter.

This algorithm finds $\alpha_{T,T'}$ such that:

$$\begin{aligned}
&\sum_T \int_T (\bar{u}_h^{(m)} - \bar{u}_h^{(m-1)}) v_h \,\mathrm{d}x \\
+\alpha_{T,T'}\Delta t\Big(&\int_{\partial T_{in}} b\cdot n\, \hat{u}_h^{(m)} v_h \,\mathrm{d}s + \int_{\partial T_{out}} b\cdot n\, \bar{u}_h^{(m-1)} v_h \,\mathrm{d}s\Big) = 0, \quad \forall v_h \in \bar{V}_h, \\
&\sum_T \int_{\partial T_{out}} b\cdot n\, (\bar{u}_h^{(m-1)} - \hat{u}_h^{(m)}) \hat{v}_h \,\mathrm{d}s = 0, \quad \forall \hat{v}_h \in \hat{V}_h.
\end{aligned}$$

satisfies $u_* \le \bar{u}_h^{(m)} \le u^*$ on each element T.

Step 1: Sum up fluxes:

$$P_T^+ = \Delta t \sum_{T'\neq T} \max\{0, -\hat{F}_{T,T'}^{(m-1)}\}, \qquad P_T^- = \Delta t \sum_{T'\neq T} \min\{0, -\hat{F}_{T,T'}^{(m-1)}\}.$$

Step 2: Admissible bounds:

$$Q_T^+ = |T|(u^* - \bar{u}_h^{(m-1)}), \qquad Q_E^- = |T|(u_* - \bar{u}_h^{(m-1)}).$$

Step 3: Element limiting factors:

$$[0,1] \ni \alpha_T^{\pm} = \min\{1, \frac{Q_T^{\pm}}{P_T^{\pm}}\}.$$

Step 4: Flux correction factors: For $T' \neq T$,

$$\alpha_{T,T'}^{(m)} = \begin{cases} \min\{\alpha_{T'}^-, \alpha_T^+\} & \text{if } \hat{F}^{(m-1)} < 0, \\ \min\{\alpha_{T'}^+, \alpha_T^-\} & \text{if } \hat{F}^{(m-1)} > 0. \end{cases}$$

4 Slope limiting

Slope limiting is a post-processing method that constrains the higher order parts of the numerical solution. A crucial part of the slope limiting process is that we can show that changing higher order parts of the solution does not change the mean value of our solution.

Theorem 4.1. *Let $u_h \in \mathbb{P}^M(\mathcal{T})$ arbitrary. The higher order parts of the numerical solution u_h do not affect the mean value of u_h.*

Proof. Let $T \in \mathcal{T}$ be arbitrary. Since the polynomial $u^n_{h,T}$ is a summation of different orders of Legendre polynomials $\mathbb{L}_i$ with their coefficients, say a_i, we can write $u_{T,h}$ as:

$$u_{h,T}(x) = \sum_{i=0}^{M} a_i \mathbb{L}_i(x). \tag{10}$$

To get the mean value of $u_{T,h}$, we integrate over the element T:

$$\bar{u}_{h,T} = \frac{1}{|T|} \int_T u_{T,h}(x)\, \mathrm{d}x = \frac{1}{|T|} \int_T \sum_{i=0}^{N} a_i \mathbb{L}_i(x)\, \mathrm{d}x. \tag{11}$$

Legendre polynomials have the property that the mean value of all Legendre polynomials of order higher than zero is equal to 0, and the mean value of the 0-th order Legendre polynomial is 1

$$\bar{u}_{h,T} = \frac{1}{|T|} \int_T u_{T,h}(x)\, \mathrm{d}x = \frac{1}{|T|} \sum_{i=0}^{N} \int_T a_i \mathbb{L}_i(x)\, \mathrm{d}x = \frac{1}{|T|} a_0. \tag{12}$$

Thus, we can freely change the values of a_i for $i > 0$ without changing the mean value of $u_{h,T}$.

□

4.1 The idea

As mentioned in section 3 the FSL from Algorithm 1 and Algorithm 3 only ensures that the mean value of each element is bound preserving. The higher order parts can still cause the solution to exceed these bounds on certain points. In order to ensure point-wise bound preservation, we implement a vertex based slope limiter (V-SL). After every iteration of the FSL we multiply the higher order linear part of the solution with a variable $\beta \in [0, 1]$ such that:

$$u_*^{\partial T} \leq \bar{u}^n_{h,T} + \beta \nabla u^n_{h,T} \cdot (C_T - x) \leq u^*_{\partial T}. \tag{13}$$

Here $u_*^{\partial T}$ and $u^*_{\partial T}$ denote a lower bound and higher bound on the vertices respectively. Further, C_T is the position of the centre of the element. The term $(C_T - x)$ should thus be interpreted as the distance between the centre of the element and the considered interface.

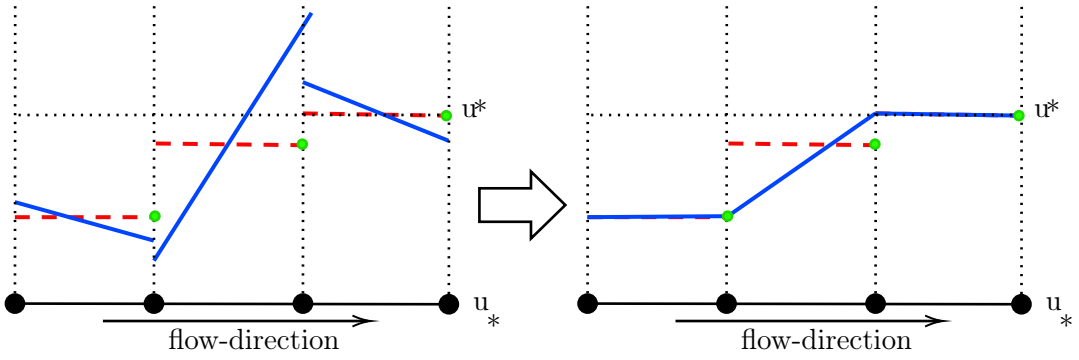


FIGURE 1: A visualisation of slope limiting on an example situation to demonstrate the point-wise bound preservation. The red dotted lines represent the local means on the element, the blue lines represent the higher order solution and the green dots the approximation on the facets found by the HDG scheme. u_* and u^* represent the lower bound and upper bound respectively.

4.2 Vertex based slope limiter

We implement the linear Kuzmin slope limiter [6]. This V-SL constraints the slope on an element based on the mean values on the neighbouring elements. Let $\xi(T)$ denote the set of facets of an element T We define our vertex wise lower- and upper bound as follows:

$$u_*^{\partial T} = \min_{\{T' \in \mathcal{T} : \partial T' \in \xi(T)\}} \bar{u}_{h,T'} \qquad \text{and} \qquad u^*_{\partial T} = \max_{\{T' \in \mathcal{T} : \partial T' \in \xi(T)\}} \bar{u}_{h,T'}. \tag{14}$$

To force the numerical solution to be bounded at every value of x we define

$$\beta_T = \min_{\partial T \in \xi(T)} \begin{cases} \frac{u^*_{\partial T} - \bar{u}_{h,T}}{u_h(\partial T) - \bar{u}_{h,T}} & \text{if } u_h(\partial T) > u^*_{\partial T}, \\ 1 & \text{if } u_*^{\partial T} < u_h(\partial T) < u^*_{\partial T}, \\ \frac{u_*^{\partial T} - \bar{u}_{h,T}}{u_h(\partial T) - \bar{u}_{h,T}} & \text{if } u_h(\partial T) < u_*^{\partial T}. \end{cases}$$

We use $u_h(\partial T)$ to denote the value of the numerical solution on the boundary ∂T, looked at from the solution on T.

Thus, by element wise changing the higher order part of the numerical solution, we can ensure that the entirety of our solution is bound preserving. We can do this without changing the total mass of the entire solution.

4.3 Global Bound based slope limiter

The goal of the slope limiter is to make sure that $u_* \leq u^n_{h,T} \leq u^*$ for all values of x in the element T. The downside of the vertex based slope limiter is, that we sometimes limit the slope even tough Equation (2) is not violated. To prevent this we propose a new slope limiter, a so-called bound based slope limiter (B-SL). The idea is the same as for the V-SL of Section 4.2. Instead of limiting the slope based on neighbouring elements, we limit the slope based on pre-defined global bounds. We define

$$\beta_T = \min_{\partial T \in \xi(T)} \begin{cases} \frac{u^* - \bar{u}_{h,T}}{u_h(\partial T) - \bar{u}_{h,T}} & \text{if } u_h(\partial T) > u^*, \\ 1 & \text{if } u_* < u_h(\partial T) < u^*, \\ \frac{u_* - \bar{u}_{h,T}}{u_h(\partial T) - \bar{u}_{h,T}} & \text{if } u_h(\partial T) < u_*. \end{cases} \tag{15}$$

Using this method we only limit the slope if the solution were to violate the boundaries at any point. There is of course also a downside to using a B-SL compared to using a V-SL. Using a B-SL allows the solution to be more discontinuous. The jump of the solution from one element to the next can be much larger. This is in most cases not a problem, however when large discontinuities are undesired using a V-SL can be preferable. As we try to develop a method that allows for a more physically viable solution, we will not analyse numerical results using B-SL in great detail.

5 Numerical results

In the following sections, we look at the implementation of the FSL and the SL on a certain scenario, after which we discuss the results. All simulations are done using Python and the NGSolve framework [1] and (`www.ngoslve.org`). The code for the simulation can be found on GitLab [5]. All simulation are made using zeroth and first order Legendre polynomials in an 1-dimensional space. The mesh is defined on the domain $(0, 1)$, with periodic boundary. The mesh is partitioned in 50 equally sized elements.

5.1 Linear advection equation

The situation we analyse is the linear advection of a step-function. We use this situation, as the discontinuity in the step function causes a non-limited solution to break conservation laws. Thus, this situation is a nice test to see if the FSL and the SL prevent the breaking of these laws. Consider Equation (1), define $b = 1$ and let

$$u_0(x) = \begin{cases} 1 & \text{if } x \geq \frac{1}{2}, \\ 0 & \text{if } x < \frac{1}{2}, \end{cases} \tag{16}$$

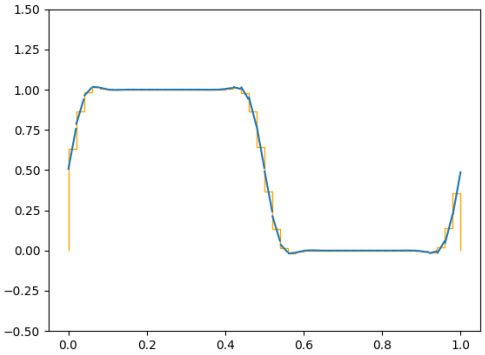
with upper bound $u^* = 1$ and lower bound $u_* = 0$. We look at the time interval $\mathbb{T} := (0, 0.5]$ with $\Delta t = 10^{-3}$. This gives us a total of 500 time steps. We see the numerical solution at $t = 0.5$ of a non-limited solution, see Figure 2a, the DG explicit method with FSL and V-SL, see Figure 2b, the HDG fully implicit methods with FSL and V-SL, see Figure 2c, and finally the HDG semi-implicit methods with FSL and V-SL, see Figure 2d.

From Figure 2a we see at around $x = 0.55$ and $x = 0.45$ that we get an under and overshoot of the solution. These under and overshoots violate the restrictions of $0 = u_* \leq u_{h,T} \leq u^* = 1$. In Figure 2b we see that the solution is between the 0 and 1. This shows that, indeed, the FSL and the V-SL prevent the violation of the conservation laws. In Figure 2c and 2d we see the solution found using both an implicit and semi-implicit FSL for the HDG method. For the fully implicit HDG approach, we observe a difference (Figure 2c) at $x = 0.1$ and $x = 0.5$. At these values there seems to be a small dent and bump respectively, which we do not see when using an explicit FSL in Figure 2d. A reason for these dents or bumps could be that when using an implicit FSL we do not fully converge to our desired α_T, which then limits our flux too much. However, this would need some further research to be certain of this.

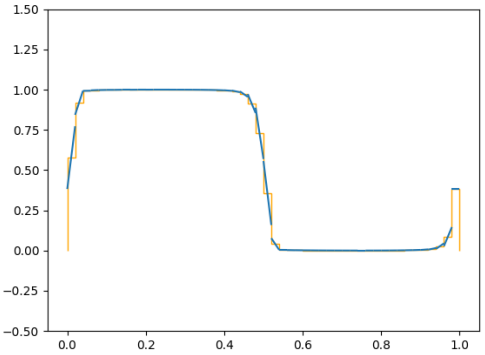
5.2 Second scenario: Smooth function

We now look at a smooth initial condition that lies well between the upper- and lower bound. Consider Equation (1), define $b = 1$ and let

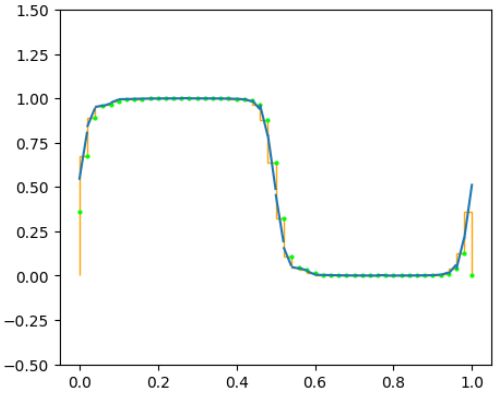
$$u_0(x) = 0.5\sin(2\pi x) + 0.5 \tag{17}$$



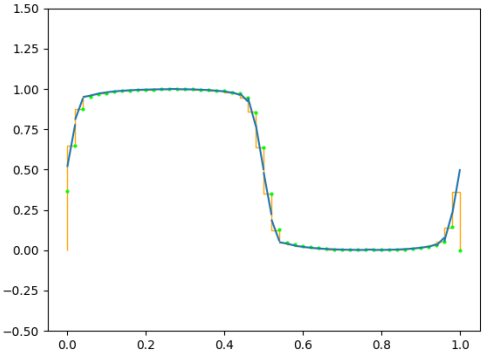
(A) Explicit DG solution without using limiters.



(B) Explicit DG solution using FSL and V-SL.

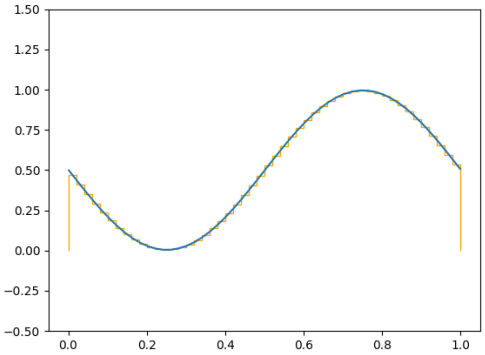


(C) Implicit HDG solution using FSL and V-SL.

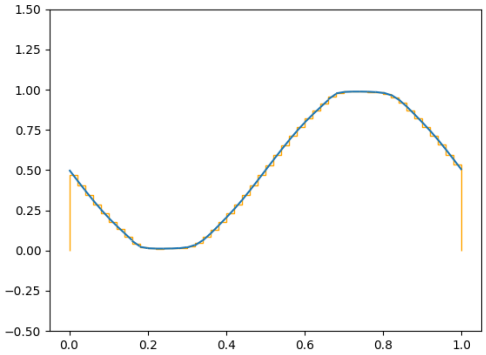


(D) Implicit HDG solution using explicit FSL and V-SL.

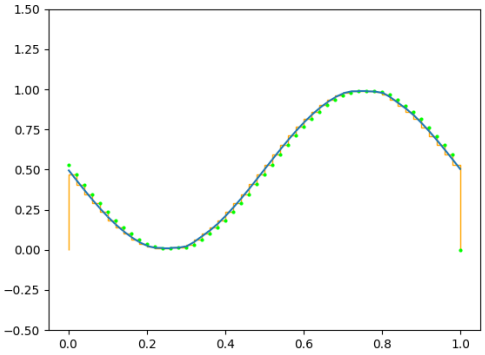
FIGURE 2: Numerical solution of Equation (1) at time $t = 0.5$, using 50 elements, $\Delta t = 10^{-3}$,$b = 1$ and u_0 as defined in 16. The line segments represent the high order solution, the orange lines represent the mean values of the elements, and the green dots are the solution on the vertices for the HDG method.



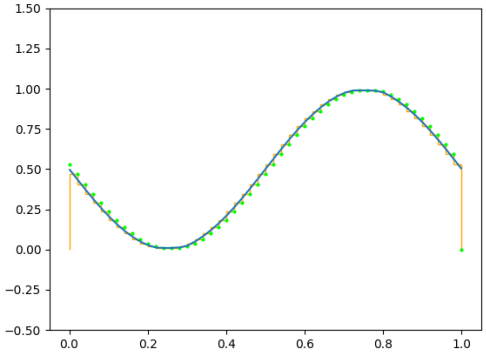
(A) Explicit DG solution of a smooth initial condition without using limiters.



(B) Explicit DG solution of a smooth initial condition using FSL and V-SL.



(C) Implicit HDG solution of a smooth initial condition using FSL and V-SL.



(D) Implicit HDG solution of a smooth initial condition using explicit FSL and V-SL.

FIGURE 3: Numerical solution of Equation (1) at time $t = 0.5$, using 50 elements, $\Delta t = 10^{-3}$,$b = 1$ and u_0 as defined in 17. The line segments represent the high order solution, the orange lines represent the mean values of the elements, and the green dots are the solution on the vertices for the HDG method.

with upper bound $u^* = 1.5$ and lower bound $u_* = -0.5$. We look at the time interval $\mathbb{T} := (0, 0.5]$ with $\Delta t = 10^{-3}$.

From Figure 3 we see that when we have a system that does not require any flux limiting or slope limiting, Algorithm 1 and 3 give an almost identical solution as the non-limited method.

5.3 Error analysis

By merely looking at the numerical solution of these various methods at a given time, it is quite hard to determine how "well" the method works. Hence, in this section, we look at the total error between the numerical solution and the theoretical solution over time. If we wish to calculate this error, we must first define how to calculate this error. Let

$$\hat{L}_T(t_n) = ||u_h(t_n) - u(t_n)||_{L^2(\Omega)} = \sqrt{\sum_T \int_T (u_{h,T}^{(n)}(x) - u(x, t_n))^2 \, \mathrm{d}x}, \tag{18}$$

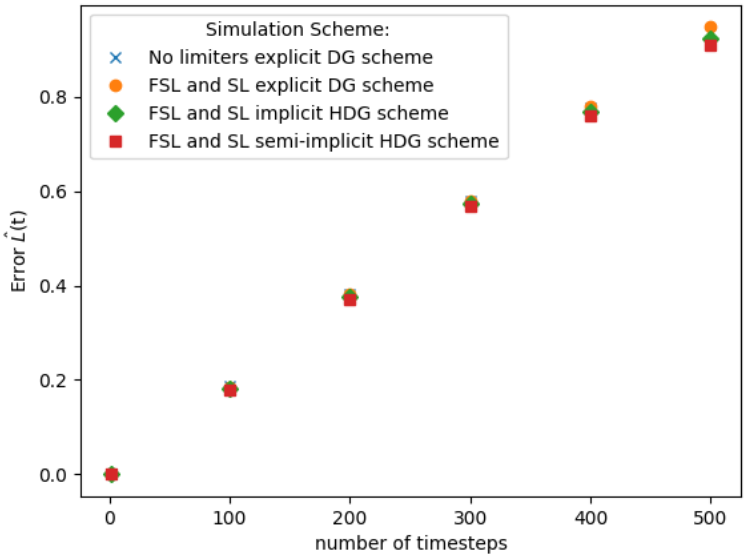


FIGURE 4: The error for the linear advection problem mentioned in section 5.1 of various simulation schemes as a function of the number of time steps, where one time step $(\Delta t) = 10^{-3}$. The error is calculated at the time step-number 1 , 100 , 200 , 300 , 400 and 500.

thus $\hat{L}_T(t_n)$ denotes the error at time t_n measured in the L^2-norm. Here $u(x,t)$ is the exact solution.

We should note that the value of this error does not have any physical meaning and is merely used as a way to compare different numerical methods. The error we might be interested in highly depends on the practical scenario that we wish to solve, and many more variables.

We calculate the error for all the simulation schemes, including the explicit DG scheme without any limiters. We use the setting of the first case mentioned in Section 5.1, with the same parameters. We do this at several given time step: $t_n = 1$, 100, 200, 300, 400 and 500. We do not include the point $t_n = 0$ as this would result in an error of 0 for all schemes. The chosen time step numbers are such that the step made by the step function happens exactly at the interface of an element, causing the theoretical solution to be constant in an element.

In Figure 4 we see the error of various simulation schemes as a function of the number of time steps. We see that the total error of these different schemes barely differ over time. Even with those undesired dents and bumps in our solution in the fully implicit HDG scheme with FSL and V-SL.

We now calculate the error of our solution again using

$$L(t_n) = \sum_T \int_T u_{h,T}^{(n)}(x) - u(x,t_n)\,\mathrm{d}x.$$

This gives us an "error bias", giving us insight on whether our solution is consistently too large or to small. By Theorem 2.1 we expect $L(t) = 0$ as the total mass of $u_h(t)$ must be equal to the mass of $g(t)$ for all t. For all schemes these error values are around 10^{-15} which corresponds to the machine precision, i.e. represent a "numerical zero".

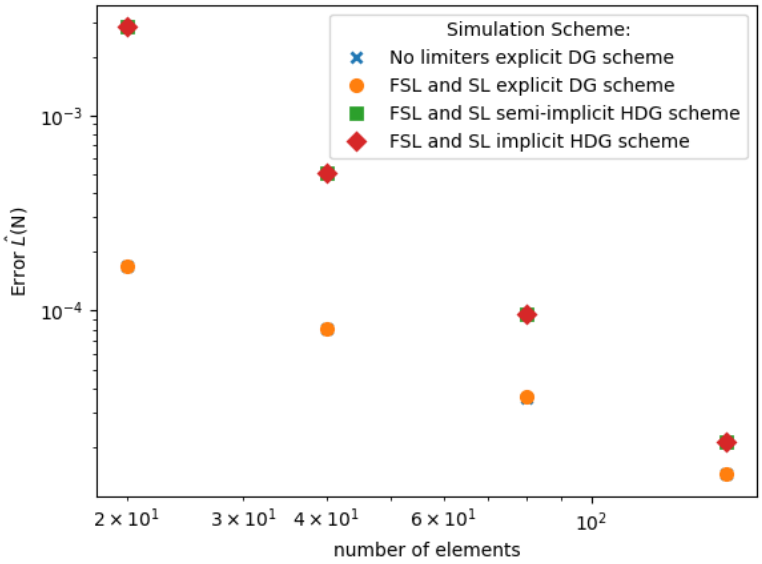


FIGURE 5: The error, as calculated in (18), of various simulation schemes as a function of the number elements $|\mathcal{T}|$. The error is evaluated after 100 time steps for $|\mathcal{T}| = 20,\ 40,\ 80,\ 160$, with $\Delta t = 10^{-5}$.

5.4 Error in terms of grid size

We now look into the error as calculated in Equation (18) as a function of the number of elements in our mesh. Consider the scenario as described in section 5.2, again with $b = 1$, $\Delta t = 10^{-3}$ and u_0 as defined in (17). We calculate the error after 100 time steps, this way our time-error is much smaller than the spatial error. The time error becomes negligible when $|\mathcal{T}| \ll r\Delta t$, where r is the number of time steps.

In Figure (5) we see that both the non-limited and limited explicit method have the same order of convergence. Implementing the limiters does not decrease our rate of convergence. The same holds for both implicit schemes.

6 Conclusions

In this work we formulate a flux limiting scheme for HDG schemes based on the fractional step limiter of [3]. This limiter ensures upper and lower boundedness for local means. We develop two methods of implementing the FSL scheme for HDG schemes. The first method is a fully implicit version. At every time step we implicitly find an initial guess, after which we find the flux limiting factors $\alpha_{T,T'}$ using implicit time steps. The second method again uses an implicit time step to find an initial guess, but then uses an explicit time step to find the flux limiting factors. Secondly, we define a vertex-based slope limiting algorithm from [6] to ensure that point-wise maximum principality holds for first order polynomials. We also look at how these limiters affect the accuracy of the solution as a function of the number of time steps. We see that the application of the FSL and the SL does not decrease the accuracy.

Even though the FSL does significantly increase the computational time, in combination with the SL it is still a very effective method of enforcing conservation laws.

7 Discussion

Even though the methods presented in this work allow us to achieve our goal of the preservation of conservation laws, the methods have their downsides and their restrictions. The biggest restriction of the slope limiter is that when we want to use a second, or higher order approximation, the slope limiter fails. The slope limiter could perhaps be modified to also work for higher order solutions. For example, we could use a hierarchical iterative manner to modify the higher order part. This would however would require some further study. In this work, we only discuss the application of the FSL and SL for a 1 dimensional linear advection equation. Looking into multidimensional problems or non-linear equations, such as Burgers' equation or the shallow water problem would be very interesting.

Acknowledgments

The past few months, while working on my thesis, Philip Lederer has taught me many things about partial differential equations, finite element methods, numerical mathematics and how to properly write a mathematical paper. For this I want to thank Philip Lederer. I also want to thank my brothers, Lou and Hidde, for proof reading the thesis and giving insightful feedback on both the content and writing style.

References

[1] Netgen/NGSolve.

[2] M. Akbas, T. Gallouët, A. Gassmann, A. Linke, and C. Merdon. A gradient-robust well-balanced scheme for the compressible isothermal Stokes problem. *Computer Methods in Applied Mechanics and Engineering*, 367:113069, August 2020.

[3] Florian Frank, Andreas Rupp, and Dmitri Kuzmin. Bound-preserving flux limiting schemes for DG discretizations of conservation laws with applications to the Cahn–Hilliard equation. *Computer Methods in Applied Mechanics and Engineering*, 359:112665, February 2020.

[4] Shou-Jun Huang and Li-Fan Wu. Stability of singular solutions to the b-family of equations. *Monatsh Math*, 204(1):63–79, May 2024.

[5] Bram .J.D.Kanger. Kanger, B.J.D. (Bram, Student B-AM) / Bachelor Thesis · GitLab, May 2024.

[6] Dmitri Kuzmin. A vertex-based hierarchical slope limiter for p-adaptive discontinuous Galerkin methods. *Journal of Computational and Applied Mathematics*, 233(12):3077–3085, April 2010.