The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5-–9 June 2022, Oslo, Norway

AN EXTENDED DISCONTINUOUS GALERKIN METHOD FOR HIGH-ORDER SHOCK TREATMENT

JAKOB SEBASTIAN[1], FLORIAN KUMMER[1,2]

[1] TU Darmstadt, Chair of Fluid Dynamics, Otto-Berndt-Straße 2, 64287 Darmstadt, Germany, sebastian@fdy.tu-darmstadt.de, www.fdy.tu-darmstadt.de

[2] TU Darmstadt, Graduate School Computational Engineering, Dolivostraße 15, 64293 Darmstadt, Germany, www.gsc.tu-darmstadt.de

Key words: compressible flow, (extended) discontinuous Galerkin, level set, shock fitting

Abstract. In this paper, we are going to present a high-order shock fitting approach based on a cut-cell method. We formulate a suitable Constraint Optimization Problem and develop an algorithm aiming to reconstruct the shock front represented by the zero iso-contour of a Level Set function.

1 INTRODUCTION

In Computational Fluid Dynamics, the Discontinuous Galerkin (DG) method is of current scientific interest due to its high-order convergence rates enabling running simulations on coarse grids. In order to work properly, when simulating compressible flows, the method at hand must be able to deal with discontinuities in the flow field such as shock waves. In the BoSSS-framework the so called eXtended Discontinuous Galerkin (XDG) aka. unfitted DG (UDG) method is used for different flows to deal with different types of discontinuities [1]. There, a Level Set function is used to represent the discontinuity surface adding further degrees of freedom (DOF) to the method.

Shock waves are physical phenomena that naturally arise in trans-sonic flows, where the local Mach number exceeds 1 in certain regions [2]. The transition from super- to sub-sonic flow is a very thin layer in the flow field in which scalar properties change drastically and are therefore numerically challenging, especially when using high-order schemes like the DG-method. Because of their thinness they can be modelled as discontinuity surfaces along which scalar quantities jump.

One way of dealing with shock waves is trying to align the computational mesh with the shock, making use of the discontinuous nature of the DG-method, which is called *shock fitting*. The advantage of this approach is the preservation of high order convergence, which for some other methods e.g. those based on artificial viscosity is not given [3]. One difficulty in this approach is, that the position of the shock wave is generally not known and can change or develop in time. Also complex patterns can form due to interactions of different shock waves or reflections, thus shock fitting played a minor role in the simulation of trans-sonic flows. Recently, new shock fitting approaches are being developed and successfully applied to 2D flows in the context of DG methods. There ([4], [5]) implicit methods are proposed which simultaneously

compute the numerical solutions of conservation laws and align the mesh with discontinuities. This is achieved by an optimization algorithm and doesn't require explicit knowledge of the position of the discontinuity.

In our work we use a similar approach in the context of the XDG method, where we represent the discontinuity's position by the zero iso-contour of a Level Set function leaving the background mesh fixed. By doing so we avoid some of the computational difficulties since we do not have to worry about ill-shaped cells. Also for XDG methods there exists a wide range of methods [6] of evolving interfaces in time (e.g. moving shock fronts), therefore the XDG method might have a general potential for moving shocks.

2 Preliminaries

Our goal is to numerically solve hyperbolic PDE's, such as e.g the Euler Equations, in the vicinity of shocks. In order to do so we consider a general system of $m \in \{2, 3, 4, 5\}$ inviscid conservation laws defined on a physical domain $\Omega \subset \mathbb{R}^D$

$$\nabla \cdot \mathbf{f}(U) = 0 \quad \text{in } \Omega \tag{1}$$

subject to appropriate boundary conditions. Here $U : \Omega \to \mathbb{R}^m$ denotes the solution of the system and $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^D$ denotes the physical flux. We define the background grid as

$$\mathcal{K}^h = \{K_1, K_2, \ldots, K_J\} \tag{2}$$

covering the whole domain, with non-overlapping cells e.g. $\bigsqcup_{i=1}^{J} K_i = \Omega$. Furthermore, we denote by n_Γ the normal field on the set of all edges $\Gamma = \bigcup_{j \in J} \partial K_j$. The edges can be divided into the inner edges Γ_{int} and outer edges Γ_{out} and the latter are further subdivided into those where we have Dirichlet Boundary Conditions and those where we have Neumann Boundary Conditions,

$$\Gamma = \Gamma_N \cup \Gamma_D \cup \Gamma_{\text{int}}, \quad \Gamma_{\text{int}} = \Gamma \setminus \partial\Omega. \tag{3}$$

Furthermore we subdivide depending whether we have inflow or outflow Neumann Boundary Conditions:

$$\Gamma_N = \Gamma_{\text{in}} \cup \Gamma_{\text{out}}. \tag{4}$$

Using this one can then define the vector valued DG Space:

$$\mathcal{V}_P^m := [\mathbb{P}_P(\mathcal{K}_h)]^m = \left\{ f \in [L^2(\Omega)]^m; \forall K \in \mathcal{K}_h : f|_K \in [\mathbb{P}_P(K)]^m \right\}. \tag{5}$$

in which we will approximate the solution U of our conservation law. For a field $\psi \in C^0(\Omega \setminus \Gamma) \subset \mathcal{V}_P^1$ we denote by

$$\psi^{\text{in}} := \lim_{\epsilon \to 0^+} \psi(x - \epsilon n_\Gamma), \quad \psi^{\text{out}} := \lim_{\epsilon \to 0^-} \psi(x - \epsilon n_\Gamma), \quad \forall x \in \Gamma \tag{6}$$

the inner and outer values and the jump operator by

$$[\![\psi]\!] := \begin{cases} \psi^{\text{in}} - \psi^{\text{out}} & \text{on } \Gamma_{\text{int}} \\ \psi^{\text{in}} & \text{on } \partial\Omega \end{cases}. \tag{7}$$

For every cell K_j one can define by $\{\phi_{j,n}\}_{1\leq n\leq N_P}$ a basis of $\mathbb{P}_P(K_j)$ and introduce the polynomial basis vector ϕ:

$$\phi_j = (\phi_{j,n})_{1\leq n\leq N_P} \in (\mathcal{V}_P)^{N_P}\ ,\ \phi = (\phi_j)_{1\leq j\leq J}, \quad \Phi = (\underbrace{\phi\ \dots\ \phi}_{m \text{ times}})^T \tag{8}$$

Then the unique coordinate vector of an element $g \in \mathcal{V}_P^m$ is given as :

$$\underline{g} \in \mathbb{R}^{m\cdot J\cdot N_P} \text{ s.t. } g = \underline{g}\cdot\Phi \tag{9}$$

In the course of this work the DG-method will be extended using a continuos Level Set $\varphi \in \mathcal{V}_{P'}(\Omega)$ with polynomial degree P'. Let ϕ_Ω be a polynomial basis vector of $\mathcal{V}_{P'}(\Omega)$ and $\underline{\varphi} \in \mathbb{R}^{N_{P'}}$ such that

$$\varphi = \underline{\varphi}\cdot\phi_\Omega. \tag{10}$$

With this Level Set one can define the (Level Set dependent) sub-domains of our physical domain

$$\begin{aligned} \mathfrak{A} = \mathfrak{A}_\varphi &:= \{x\in\Omega : \varphi(x) < 0\} \\ \mathfrak{I} = \mathfrak{I}_\varphi &:= \{x\in\Omega : \varphi(x) = 0\} \\ \mathfrak{B} = \mathfrak{B}_\varphi &:= \{x\in\Omega : \varphi(x) > 0\}. \end{aligned} \tag{11}$$

The (Level Set dependent) cut cell grid is then obtained by:

$$\mathcal{K}_h^X(\varphi) = \{K_{1,\mathfrak{A}_\varphi}, K_{1,\mathfrak{B}_\varphi}, \dots, K_{J,\mathfrak{A}_\varphi}, K_{J,\mathfrak{B}_\varphi}\}, \quad K_{j,\mathfrak{s}_\varphi} := K_j \cap \mathfrak{s}_\varphi, \quad \mathfrak{s}_\varphi \in \{\mathfrak{A}_\varphi, \mathfrak{B}_\varphi\}. \tag{12}$$

Using the above, the vector valued XDG space can be defined as:

$$\mathcal{V}_P^{X,m} := [\mathbb{P}_P^X(\mathcal{K}_h)]^m = \left\{ f\in[L^2(\Omega)]^m; \forall K\in\mathcal{K}_h : f|_{K\cap\mathfrak{s}_\varphi} \in [\mathbb{P}_P(K\cap\mathfrak{s}_\varphi)]^m,\ \forall \mathfrak{s}_\varphi \right\}. \tag{13}$$

For the XDG Space $\mathcal{V}_P^X$ one obtains a basis for each sub-domain (and dependent of the level Set φ) by multiplying every basis function by an indicator function:

$$\phi_\mathfrak{A} = \phi\cdot\mathbb{1}_\mathfrak{A}, \text{ and } \phi_\mathfrak{B} = \phi\cdot\mathbb{1}_\mathfrak{B} \tag{14}$$

Then one can write:

$$\phi_X = \begin{pmatrix}\phi_\mathfrak{A}\\ \phi_\mathfrak{B}\end{pmatrix}, \quad \text{and } \Phi_X = (\underbrace{\phi_X\ \dots\ \phi_X}_{m \text{ times}})^T. \tag{15}$$

Similarly, the coordinate vector of an element $g \in \mathcal{V}_P^{X,m}$ is given as:

$$\underline{g}_X \in \mathbb{R}^{2m\cdot J\cdot N_P} \text{ s.t. } g = \underline{g}_X\cdot\Phi_X. \tag{16}$$

3 XDG-Method and Error Function

In the following we will define an Error Function based on the residuals obtained from the XDG method. To obtain a XDG method one needs to first multiply every component of our system (1) by suitable test functions $v \in \mathcal{V}_P^X$ and integrate by parts in order to obtain:

$$\oint_\Gamma f_i(U) n_\Gamma [\![v]\!] \, dS - \int_\Omega f_i(U) \nabla_h v \, dV = 0. \tag{17}$$

Next, a discrete state vector and its coordinate vector can be defined as

$$U \approx U_h := \underline{U}_X \cdot \Phi_X \tag{18}$$

Inserting the discrete state vector into (17), introducing a numerical flux

$$\hat{F}_i(U^+, U^-, n_\Gamma) \approx \mathbf{f}_i(U) \cdot n_\Gamma \tag{19}$$

for every component $1 \leq i \leq m$ and inserting a specific basis function $\phi_{j,n,\mathfrak{s}}$ one obtains the respective residual:

$$r_n^{i,j,\mathfrak{s}}(U_h, \varphi) := \oint_{\partial K_{j,\mathfrak{s}}} \hat{F}_i(U_h^+, U_h^-, \mathbf{n}_\Gamma) [\![\phi_{j,n,\mathfrak{s}}]\!] dS - \int_{K_{j,\mathfrak{s}}} \mathbf{f}_i(U_h) \nabla_h \phi_{j,n,\mathfrak{s}} dV = 0. \tag{20}$$

Collecting all of these local residuals leads to an algebraic system

$$r_P(U_h, \varphi) := \begin{pmatrix} \vdots \\ r_n^{i,j,\mathfrak{s}}(U_h, \varphi) \\ \vdots \end{pmatrix} = 0. \tag{21}$$

Let us fix the P for notational purpose and define the residual in coordinate form. Set $N_U := 2 \cdot m \cdot N_P \cdot J$, then $r : \mathbb{R}^{N_U} \times \mathbb{R}^{N_{P'}} \to \mathbb{R}^{N_U}$,

$$r(\underline{U}_h, \underline{\varphi}) := r_P(\underline{U}_h \cdot \Phi_X, \underline{\varphi} \cdot \phi_\Omega). \tag{22}$$

In an XDG method one typically wants to find a discrete solution where the residual vanishes. As shown in [5] the residual of an DG method can vanish without having fit the interface why we won't use it as our primary objective function but as a constraint to our minimization problem. As our objective function we use the so called Enriched Residual of higher degree $P* > P$ which is also used in [4] and has more sensitivity to non-aligned interfaces. So let us consider $U_h \in \mathcal{V}_P^{X,m}$ and the canonical injection $i : \mathcal{V}_P^{X,m} \to \mathcal{V}_{P*}^{X,m}$. Then one can define the Enriched Residual $R : \mathbb{R}^{N_U} \times \mathbb{R}^{N_\varphi} \to \mathbb{R}^{2 \cdot m \cdot N_{P^*} \cdot J}$ in coordinate form using the residual of degree P^*

$$R(\underline{U}_h, \underline{\varphi}) = r_{P^*}(i(\underline{U}_h \cdot \Phi_X), \underline{\varphi} \cdot \phi_\Omega). \tag{23}$$

Let us from now on start omitting the underlines and define the non-linear error function which will serve as our objective function $f : \mathbb{R}^{N_U} \times \mathbb{R}^{N_\varphi} \to \mathbb{R}$

$$f(U, \varphi) := \frac{1}{2} R(U, \varphi)^T R(U, \varphi) = \frac{1}{2} \|R(U, \varphi)\|^2. \tag{24}$$

Using this one can formulate the following non-linear constrained optimization problem. Find (U, φ) such that:

$$\begin{aligned} f(U, \varphi) &\to \min! \\ \text{subject to } r(U, \varphi) &= 0. \end{aligned} \tag{25}$$

4 Solving the Optimization Problem

Next, a method, which aims to solve the optimization problem (25) will be presented. Utilizing the theory of non-linear constraint optimization and using Lagrange multipliers one obtains the functional $\mathcal{L} : \mathbb{R}^{N_U} \times \mathbb{R}^{N_\varphi} \times \mathbb{R}^{N_U} \to \mathbb{R}$ which writes

$$\mathcal{L}(U, \varphi, \lambda) = f(U, \varphi) - \lambda^\top r(U, \varphi). \tag{26}$$

Then U^*, φ^* is a first order solution if there exists a $\lambda^* \in \mathbb{R}^{N_U}$ such that

$$\nabla \mathcal{L}\left(U^*, \varphi^*, \lambda^*\right) = 0 \tag{27}$$

which in more detail writes as:

$$\begin{aligned} \nabla_U f\left(U^*, \varphi^*\right) - \frac{\partial \mathbf{r}}{\partial U}\left(U^*, \varphi^*\right)^T \lambda^* = 0, \\ \nabla_\varphi f\left(U^*, \varphi^*\right) - \frac{\partial \mathbf{r}}{\partial \varphi}\left(U^*, \varphi^*\right)^T \lambda^* = 0, \\ \mathbf{r}\left(U^*, \varphi^*\right) = 0 \end{aligned} \tag{28}$$

For notational purpose we write

$$z := \begin{pmatrix} U \\ \varphi \end{pmatrix} \in \mathbb{R}^{N_z}, \quad N_z = N_U + N_\varphi. \tag{29}$$

In order to solve (27) one can apply Newtons's method, for which the Hessian of $\mathcal{L}$ is needed. The Hessian writes as

$$H_{\mathcal{L}} = \begin{pmatrix} A(z, \lambda) & J_r^T(z) \\ J_r(z) & 0 \end{pmatrix}, \text{ where } A(z, \lambda) := H_f(z) - \sum_{k+1}^{N_Z} \lambda_k H_{r_k}(z) \tag{30}$$

and H_f, H_{r_k} denote Hessian matrices and J_r denote Jacobian matrices of the respective functions. Computing those Hessians is a very challenging task involving second order derivatives that are hard to approximate. In order to circumvent this issue we follow [4] and use the Levenberg-Marquardt method [7], replacing the Hessian with an regularized approximation:

$$A(z, \lambda) \approx \frac{\partial R}{\partial z}(z)^T \frac{\partial R}{\partial z}(z)^T + \gamma I \tag{31}$$

where $\gamma \in \mathbb{R}^+$ is an regularization parameter. Furthermore, we only regularize the components corresponding to Level Set DOFs, so that the approximation of the Hessian becomes

$$B(z) := \begin{pmatrix} B_{UU}(z) & B_{U\varphi}(z) \\ B_{U\varphi}(z)^T & B_{\varphi\varphi}(z) \end{pmatrix} \tag{32}$$

where

$$B_{UU}(z) = \frac{\partial R}{\partial U}(z)^T \frac{\partial R}{\partial U}(z) \tag{33}$$

$$B_{U\varphi}(z) = \frac{\partial R}{\partial U}(z)^T \frac{\partial R}{\partial \varphi}(z) \tag{34}$$

$$B_{\varphi\varphi}(z) = \frac{\partial R}{\partial \varphi}(z)^T \frac{\partial R}{\partial \varphi}(z) + \gamma I. \tag{35}$$

Using this approach one obtains the linear system

$$\begin{pmatrix} B(z_k) & J_r^T(z_k) \\ J_r(z_k) & 0 \end{pmatrix} \begin{pmatrix} s_k \\ \delta_k \end{pmatrix} = - \begin{pmatrix} \nabla f(z_k)^T + J_r^T(z_k)\lambda_k \\ r(z_k) \end{pmatrix} \tag{36}$$

where the current iterate is updated via

$$\begin{pmatrix} z_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} z_k \\ \lambda_k \end{pmatrix} + \alpha_{k+1} \begin{pmatrix} s_k \\ \delta_k \end{pmatrix} \tag{37}$$

and where $\alpha \in [0, 1)$ is a step length parameter which is computed via an inexact line search algorithm.

Line search globalization As commonly done in non linear solvers we implement an inexact line search algorithm to find a suitable step size α_k. That is, we iteratively search for the maximal

$$\alpha_k \in \{1 = \tau^0, \tau^{-1}, \dots, \alpha_{min}\}$$

(for some $\tau \in (0, 1)$) that satisfies the condition of sufficient decrease

$$\theta_k(\alpha_k) \leq \theta_k(0) + \alpha_k \beta \theta_k'(0). \tag{38}$$

We use the merit function

$$\theta_k(\alpha) := \frac{1}{2}\|R(z_k + \alpha s_k)\|^2 + \mu(z_k)\frac{1}{2}\|r(z_k + \alpha s_k))\|^2 = f(z_k + \alpha s_k) + \mu\|r(z_k + \alpha s_k))\|^2. \tag{39}$$

Here the standard $\beta = 10^{-4}$ is chosen and $\mu(z_k) := 2\|\frac{\partial r}{\partial U}^{-T}(z_k)\frac{\partial f}{\partial U}^{T}(z_k)\|_\infty$ which controls how much we want to weight the constraint.

Adaptive choice of regularisation parameter The regularization parameter γ in turn controls how much we want to regularize our system and therefore it should be adaptive to the change in the interface position. In other words, if we consider the last computed step as "too big" ($\|s_k\| > \sigma_2 L$) we want to have more regularization, if it we consider it "too small" we want less regularization. This can be achieved by the following approach

$$\gamma_{k+1} = \min\{\max\{\hat{\gamma}_{k+1}, \gamma_{min}\}, \gamma_{max}\}, \quad \hat{\gamma}_{k+1} = \begin{cases} \kappa^{-1}\gamma_k & \text{if } \|\Delta\varphi_k\|_2 < \sigma_1 L \\ \kappa\gamma_k & \text{if } \|\Delta\varphi_k\|_2 > \sigma_2 L \\ \gamma_k & \text{else} \end{cases}. \tag{40}$$

Here σ_1, σ_2, L need to be chosen depending on the length scales of our problem and on the basis we use for φ_h. The parameter $\kappa > 1$ in turn controls how aggressive the adaptation is.

5 Results

In this section we show results for two conservation laws with a discontinuity in the solution.

5.1 Scalar Advection

First we will showcase our method for the Space-Time formulation of the linear advection equation which writes

$$\frac{\partial u}{\partial t} + a(t)\frac{\partial u}{\partial x} = 0 \quad \text{in } \Omega = [0,1] \times [0,1], \tag{41}$$

where $a : [0,1] \to \mathbb{R}$ is the time dependent velocity and

$$\mathbf{f}(c) = \begin{pmatrix} a(t)c \\ c \end{pmatrix} \tag{42}$$

the corresponding flux. We augment this with a discontinuity in the initial value

$$c(x,0) = H(0.25 - x), \tag{43}$$

where denotes H the Heaviside function, and we obtain a problem in which a discontinuity is advected in time. Further, we prescribe Dirichlet boundary conditions using the exact solution

$$c_{Ex}(x,t) = H(s(t) - x) \tag{44}$$

where the function s is given as

$$s(t) = 0.25 + \int_0^t a(t')dt'. \tag{45}$$

We will discretize this equation by using a Cartesian 10x10 background mesh. As a numerical flux we choose the upwind flux

$$\hat{F}(c^+, c^-, n_\Gamma) = \begin{cases} c^-(a(t),1)^T \cdot n_\Gamma & \text{if } (a(t),1)^T \cdot n_\Gamma < 0 \\ c^+(a(t),1)^T \cdot n_\Gamma & \text{if } (a(t),1)^T \cdot n_\Gamma \geq 0 \end{cases}. \tag{46}$$

In the following we will show results for a case where $a(t)$ is chosen as

$$a(t) = 3t^2 - 3t + 0.5. \tag{47}$$

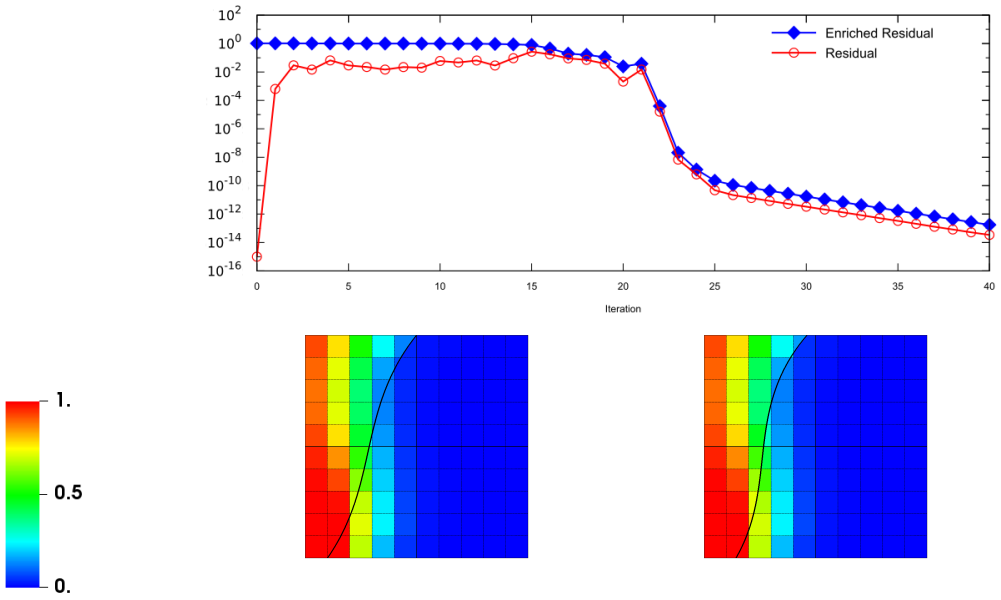
Thus, the exact solution features a shock. In a XDG space of order $P = 0$ it can be represented exactly using the Level Set

$$\varphi(x,t) = x - s(t) = x - t^3 + \frac{3}{2}t^2 - \frac{1}{2}t - \frac{1}{4}, \tag{48}$$

which is curved.
As the solution is constant we set the polynomial degree to $P = 0$ and use an ansatz for the Level Set with 5 DOFs $(x,\ t^3,\ t^2, t\ , 1)$. For the initial guess we use the Level Set

$$\varphi(x,t) = x - s(t) = x - 0.7t^3 + t^2 - 0.7t - 0.1, \tag{49}$$

and compute a $P = 0$ solution as a starting guess which is plotted in Figure 1 (a). We run 30 iterations and observe convergence of our method. Note that we start from a zero residual while the level Set is clearly not aligned. This is only achieved by the simultaneous minimization of the Enriched Residual.



(a) Iteration 0 (b) Iteration 5

(c) Iteration 10 (d) Iteration 15 (e) Iteration 20

Figure 1: Selected iterations for the scalar advection equation and opt. history.

5.2 Inviscid Burgers equation

Lastly we will apply out method onto the non-linear Space-Time formulation of the inviscid Burgers equation which writes

$$\frac{\partial c}{\partial t} + c\frac{\partial c}{\partial x} = 0 \quad \text{in } \Omega = [0,1] \times [0,1], \tag{50}$$

where

$$\mathbf{f}(c) = \begin{pmatrix} \frac{1}{2}c^2 \\ c \end{pmatrix} \tag{51}$$

is the corresponding flux. Like before we augment this with a discontinuity in the initial value

$$c(x,0) = 0.75H(0.25 - x) + 0.25H(x - 0.75). \tag{52}$$

From the Rankine Hugoniot Conditions the shock speed immediately follows to be constantly 0.5 which leads to a linear shock profile and a known Exact solution

$$c_{Ex}(x,t) = 0.75H(0.25 + 0.5t - x) + 0.25H(x - (0.25 + 0.5t). \tag{53}$$

Using this, we prescribe Dirichlet boundary conditions. The solution can be represented in a XDG space of order $P = 0$ using the Level Set

$$\varphi(x,t) = x - s(t) = x - 0.25 - 0.5t. \tag{54}$$

Again, we we use a 10x10 background mesh and choose the upwind flux

$$\hat{F}(c^+, c^-, n_\Gamma) = \begin{cases} c^-(\frac{c^+ + c^-}{2}, 1)^T \cdot n_\Gamma & \text{if } (\frac{c^+ + c^-}{2}, 1)^T \cdot n_\Gamma < 0 \\ c^+(\frac{c^+ + c^-}{2}, 1)^T \cdot n_\Gamma) & \text{if } (\frac{c^+ + c^-}{2}, 1)^T \cdot n_\Gamma \geq 0 \end{cases} . \tag{55}$$

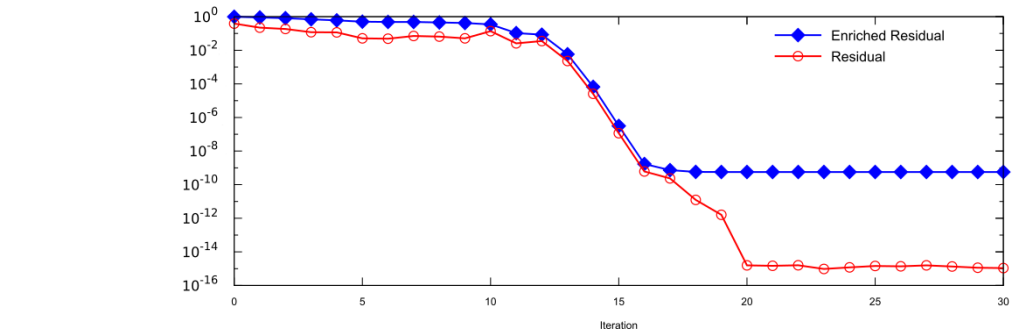
We use an ansatz for the Level Set with 4 DOFs $(x,\ t^2, t\ , 1)$ in order to show that that method is capable of finding the straight shock even though we start from a curved first guess. Thus for the initial guess we use the curved Level Set

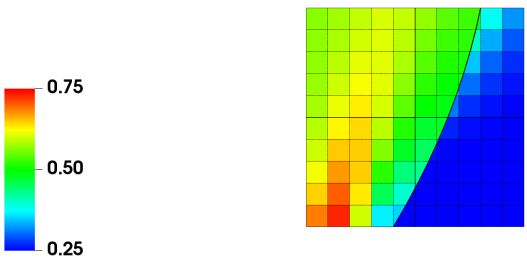
$$\varphi(x,t) = x - s(t) = x + 0.2t^2 - 0.6t - 0.4, \tag{56}$$

and compute a $P = 0$ solution as a starting guess which is plotted in Figure 2 (a). We run 30 iterations and observe convergence of our method.

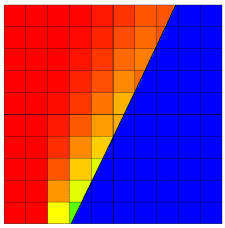
6 Conclusion and Outlook

Concluding, we presented an optimization algorithm which aligns a Level Set function with the shock while simultaneously computing the solution to the problem at hand.
The algorithm was implemented for two different Space-Time problems and for each a test run was presented. The preliminary results look promising as convergence to the desired solution was achieved from an initial guess which was not even sub-cell accurate. We demonstrated that the algorithm is able to detect the surface of unknown discontinuities. In future work we will extend this method to the steady Euler Equations in 2D.

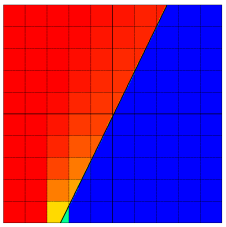




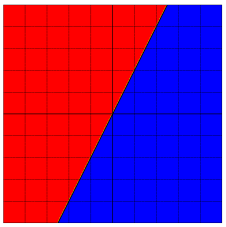
(a) Iteration 0



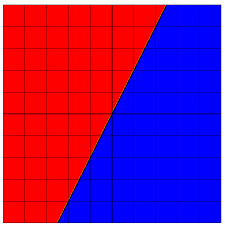
(b) Iteration 5



(c) Iteration 10



(d) Iteration 15



(e) Iteration 20

Figure 2: Selected iterations for the inviscid burgers equation and opt. history.

REFERENCES

[1] Kummer, Smuda, Weber. BoSSS: a package for multigrid extended discontinuous Galerkin methods. *Computers and Mathematics with Applications* (2021) **81**: 237–257.

[2] Anderson. *Modern Compressible Flow 3th Edition*. McGraw Hill, Vol. 3 (2003).

[3] Wang, Fidkowski, Abgrall, Bassi, Caraeni, Cary, Deconinck, Hartmann, Hillewaert, Huynh, Kroll, May, Persson, van Leer, Visbal. High-order CFD methods: current status and perspective *Int. J. Num. Meth. Fluids* (2013) **72**: 811—845.

[4] Huang, Zahr. A robust, high-order implicit shock tracking method for simulation of complex, high-speed flows *Journal of Computational Physics* (2022) **454**: 110981.

[5] Corrigan, Kercher, Kessler. A moving discontinuous Galerkin finite element method for flows with interfaces *Int. J. Numer. Meth. Fluids* (2019) **89**: 362–406.

[6] Kummer, Müller, Utz. Time Integration for extended discontinuous Galerkin methods with moving domains *Int. J. Numer. Meth. Eng.* (2018) **113**: 767-–788.

[7] Dennis, Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations* Prentice-Hall Series in Computational Mathematics, Englewood Cliffs (1983).