aerospace

MDPI

Article

A Hybridized Discontinuous Galerkin Solver for High-Speed Compressible Flow

Georg May [1,*,†], Koen Devesse [2], Ajay Rangarajan [3] and Thierry Magin [1]

[1] Aeronautics and Aerospace Department, von Karman Institute for Fluid Dynamics, B-1640 Sint-Genesius-Rode, Belgium; magin@vki.ac.be
[2] Department of Mechanical Engineering, KU Leuven, B-3001 Leuven, Belgium; koen.devesse@kuleuven.be
[3] AICES Graduate School, RWTH Aachen University, 52062 Aachen, Germany; ajay.rangarajan@rwth-aachen.de
* Correspondence: georg.may@vki.ac.be; Tel.: +32-2-486-91-21
† Current address: von Karman Institute for Fluid Dynamics, Waterloosesteenweg 72, B-1640 Sint-Genesius-Rode, Belgium.

Abstract: We present a high-order consistent compressible flow solver, based on a hybridized discontinuous Galerkin (HDG) discretization, for applications covering subsonic to hypersonic flow. In the context of high-order discretization, this broad range of applications presents unique difficulty, especially at the high-Mach number end. For instance, if a high-order discretization is to efficiently resolve shock and shear layers, it is imperative to use adaptive methods. Furthermore, high-Enthalpy flow requires non-trivial physical modeling. The aim of the present paper is to present the key enabling technologies. We discuss efficient discretization methods, including anisotropic metric-based adaptation, as well as the implementation of flexible modeling using object-oriented programming and algorithmic differentiation. We present initial verification and validation test cases focusing on external aerodynamics.

Keywords: Discontinuous Galerkin Methods; high-enthalpy flow; anisotropic adaptation; object-oriented programming; algorithmic differentiation

Citation: May, G.; Devesse, K.; Rangarajan A.; Magin, T. A Hybridized Discontinuous Galerkin Solver for High-Speed Compressible Flow. *Aerospace* **2021**, *8*, 322. https://doi.org/10.3390/aerospace8110322

Academic Editor: Hirotaka Sakaue

Received: 8 September 2021
Accepted: 10 October 2021
Published: 28 October 2021

1. Introduction

Compressible flow simulation has long been dominated by at most second order consistent methods. Prime examples are finite volume schemes and stabilized finite element schemes [1–8]. Higher order consistent methods promise superior resolution at reduced number of degrees-of-freedom (DoF), provided the solution is (locally) smooth enough [9]. In compressible flow, however, flow features such as shock waves can compromise local solution regularity. Moreover, besides shock waves, other strongly anisotropic features are usually present. Examples include boundary and shear layers. To allow efficient resolution of these features, while taking advantage of superior resolution on coarse elements, whenever the flow is smooth, a good high-order methodology ought to be adaptive, preferably using anisotropic adaptation. Furthermore, computations are often carried out primarily with the aim to compute a few solution-dependent functionals very accurately. Examples, such as force or heat transfer coefficients, readily come to mind. This suggests goal-oriented adaptation, which may be implemented via adjoint methods [10].

For a while now, metric-based adaptation methods have been used in compressible flow simulation [11–14]. A convenient formalism for such methods is the continuous mesh framework [15,16]. However, the formulation for high-order discretization, using piecewise polynomial approximation methods, has been relatively recent [17,18]. In previous work, we proposed a high-order method using analytical optimization of the continuous mesh model [18,19], including hp-adaptation [20] and goal-oriented adaptation [21,22]. The main advantage is that the procedure is entirely free of parameters, using only a constraint given

by the overall mesh-complexity. Demonstrating the fidelity of our high-order continuous mesh method with respect to high-Mach number flow is the first aim of the present paper.

Secondly, for the wide range of applications and flow conditions envisioned here, one has to consider a range of physical models. In our implementation we use the library `Mutation++` [23] to provide a flexible interface to thermodynamic closure equations and transport coefficients. We present an efficient implementation framework, based on object-oriented programming and class-based templates.

Finally, the flexible approach to modeling has implications for the overall solution strategy applied to the nonlinear discretized equations. Compressible flow discretization usually leads to stiff problems. This is even more pronounced for high-order discretization methods on very anisotropic grids. Under these circumstances, fully coupled implicit methods are attractive. For instance, Newton–Raphson methods offer asymptotic quadratic convergence, provided a very accurate differentiation of the residual function is available. This is a complication when the goal is to support easy implementation of different physical models, as it requires not only implementation of the physical model, but also its differentiation with respect to the state. We use algorithmic differentiation to facilitate the embedding of different physical models into our solver framework.

The idea of using higher order methods to simulate compressible flow is not new [24,25], and indeed there are publicly available tools [26,27]. Moreover, there is recent work with partially overlapping scope [28–30]. In particular, Nguyen et al. developed a high-order HDG solver and used it to simulate hypersonic flow, using a single-species, perfect gas model [30]. However, at the intersection of (1) high-order methods, (2) high-Mach-number flow, (3) flexible, non-trivial physical modeling, (4) anisotropic adaptation, and (5) fast solver convergence, there is a dearth of existing tools.

In summary, the purpose of this paper is to present a high-order HDG solver for high-Mach-number flows addressing the above aspects. Our solver originates from the work first presented in [31]. The overall implementation strategy is designed to support a wide range of applications and flexible modeling, while keeping the solver components (adaptation, implicit solution methods, etc.) intact without extensive programming efforts. The paper is organized as follows: In Section 2, we briefly review the governing equations. Next, in Sections 3 and 4, we discuss the discretization and adaptation methods used in this work, while Section 5 discusses our implementation strategy. Finally, in Section 6 we present numerical results.

2. Governing Equations

Our computational approach is designed to solve balance laws of the following type:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \sum_{i=1}^{d} \frac{\partial \boldsymbol{f}_i(\boldsymbol{u}, \nabla \boldsymbol{u})}{\partial x_i} = \boldsymbol{S}(\boldsymbol{u}, \nabla \boldsymbol{u}). \tag{1}$$

where $(\boldsymbol{f}_1, \ldots, \boldsymbol{f}_d) =: \boldsymbol{f}$ is the flux vector, and $\boldsymbol{S}$ is a source term. In the present paper, we focus on steady compressible flow problems. We consider models for a homogeneous mixture of ideal gases with frozen chemistry, as well as models for a mixture in chemical non-equilibrium.

For a gas composed of N_s molecular species, the state vector of conserved variables is written as

$$\boldsymbol{u} = \begin{pmatrix} \boldsymbol{\rho} \\ \rho \boldsymbol{v} \\ E \end{pmatrix}, \tag{2}$$

where $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_{N_s})^T$ is the vector of species densities, and $\rho = \sum_{i=1}^{N_s} \rho_i$ is the total density. The bulk fluid velocity vector and total energy are denoted by $\boldsymbol{v} = (v_1, \ldots, v_d)^T$, and E, respectively. In the absence of chemical reactions (frozen flow), we may treat the gas as a homogenous mixture. In this case, we may set $N_s = 1$, and only one mass equation is solved. (The same is true for flows in local thermodynamic equilibrium (LTE), where

chemical reactions are assumed to be fast enough, such that the equilibrium mixture is locally achieved and provided that the elemental composition is frozen. However, we do not consider LTE models in this work.)

We write $\boldsymbol{f} = \boldsymbol{f}_c - \boldsymbol{f}_v$, where $\boldsymbol{f}_c$ and $\boldsymbol{f}_v$, are the convective and diffusive flux, respectively. Together with the source term $\boldsymbol{S}$, they are given by

$$\boldsymbol{f}^c = \begin{pmatrix} \rho \boldsymbol{v}^T \\ \rho \boldsymbol{v}\boldsymbol{v}^T + p\boldsymbol{I}_d \\ (E+p)\boldsymbol{v}^T \end{pmatrix}, \quad \boldsymbol{f}^v = \begin{pmatrix} -\boldsymbol{J} \\ \boldsymbol{\tau} \\ \boldsymbol{v}^T\boldsymbol{\tau} - \boldsymbol{q}^T \end{pmatrix}, \quad \boldsymbol{S} = \begin{pmatrix} \dot{\boldsymbol{\omega}} \\ \boldsymbol{0} \\ 0 \end{pmatrix}. \tag{3}$$

Here, p is the pressure. Moreover, $\boldsymbol{\tau} = (\tau_{ij})_{i,j=1,\ldots,d}$ is the viscous stress tensor, $\boldsymbol{q} = (q_1, \ldots, q_d)^T$ is the heat flux vector, $\boldsymbol{J} = (J_{ij})_{i=1,\ldots N_s, j=1,\ldots,d}$ is the species diffusion flux, and $\dot{\boldsymbol{\omega}} = (\dot{\omega}_1, \ldots, \dot{\omega}_{N_s})^T$ is the vector of reaction rates. Models for these quantities are discussed below. For simple fluid models with frozen chemistry and negligible species diffusion, $N_s = 1$, $\boldsymbol{J} = \boldsymbol{0}$, and $\dot{\boldsymbol{\omega}} = \boldsymbol{0}$. If, furthermore, the flow is considered inviscid, $\boldsymbol{\tau} = \boldsymbol{0}$ and $\boldsymbol{q} = \boldsymbol{0}$.

Closure Equations, Constitutive Models, and Transport Coefficients

We assume that the gas mixture is thermally perfect (although not necessarily calorically perfect). This assumption is often made in high-enthalpy hypersonic applications where the pressure is not too high [32]. Thus, each species satisfies the ideal gas law for the partial pressure p_i, and the mixture satisfies Dalton's law,

$$p = \sum_{i=1}^{N_s} p_i = \sum_{i=1}^{N_s} \rho_i R_i T, \tag{4}$$

where R_i is the specific gas constant of species i. Under the above assumptions, we have $R_i = c_{p,i} - c_{v,i}$, where $c_{v,i} = c_{v,i}(T)$ and $c_{p,i} = c_{p,i}(T)$ are the specific heat capacities at constant volume and pressure, respectively. The thermal energy and enthalpy of each species are solely determined by the temperature [33]:

$$e_i = \int_{T_0}^{T} c_{v,i}(T)dT + e_{0,i}, \qquad h_i = \int_{T_0}^{T} c_{p,i}(T)dT + h_{0,i}. \tag{5}$$

Under the present modeling assumptions, we obtain the bulk thermal energy and enthalpy as weighted by the mixture composition:

$$e = \sum_{i=1}^{N_s} Y_i e_i, \qquad h = \sum_{i=1}^{N_s} Y_i h_i. \tag{6}$$

Here, $Y_i = \rho_i/\rho$ is the mass fraction of species i. For frozen chemistry, we may use the effective gas constant R, given by $\sum_{i=1}^{N_s} Y_i R_i$, where Y_i is assumed constant, to compute

$$p = \rho R T. \tag{7}$$

Simpler models, which may be used in external aerodynamics under subsonic or moderately supersonic free stream conditions, regard the mixture as a both thermally and calorically perfect gas. Under these assumptions we may write the equation of state as

$$p = (\gamma - 1)\rho e = (\gamma - 1)\left(E - \frac{1}{2}\rho \boldsymbol{v} \cdot \boldsymbol{v}\right), \tag{8}$$

where γ is the (constant) ratio of specific heats.

When considering Newtonian viscous flow, the stress tensor is given by

$$\boldsymbol{\tau} = \mu(\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T - \frac{2}{3}(\nabla \cdot \boldsymbol{v})\, \boldsymbol{I}_d),$$

where for simplicity, we set the bulk viscosity to zero. For moderate temperatures, one may use Sutherland's law [34] to compute the viscosity $\mu = \mu(T)$ as

$$\mu = \frac{C_1 T^{3/2}}{C_2 + T}, \tag{9}$$

with $C_1 = 1.458 \cdot 10^{-6} \frac{\text{kg}}{\text{ms}\sqrt{\text{K}}}$ and $C_2 = 110.4\,\text{K}$ for an air mixture. For higher temperatures, the library `Mutation++` may be used to compute the viscosity from the current local state.

As mentioned above, we distinguish between simple fluid models with frozen chemistry, characterized by constant mass fractions, and more general gas mixtures. For the former, recall that only one mass equation is solved. For the latter, we model species diffusion using Fick's law with averaged species diffusion coefficients [35,36],

$$\boldsymbol{J}_i = -\rho D_i \nabla Y_i + Y_i \sum_{j=1}^{N_s} \rho D_j \nabla Y_j. \tag{10}$$

Thermal and baro-diffusion effects were neglected here. Please note that the second term ensures zero net diffusion flux, when summed over all species. The averaged species diffusion coefficients D_i are obtained directly from `Mutation++` from the binary diffusion coefficients of the gas mixture. More accurate diffusion models, using multicomponent diffusion coefficients, are available in `Mutation++`, but have not been used here.

We model the heat flux using Fourier's law and energy transfer due to species diffusion (neglecting thermal diffusion induced by species diffusion) [36]:

$$\mathbf{q} = \sum_{i=1}^{N_s} h_i \mathbf{J}_i - k \nabla T. \tag{11}$$

Here, k is the mixture thermal conductivity. For simple fluid models with $Y_i = const.$, only the second term is present. At moderate flow conditions, we may assume a constant Prandtl number, whence

$$k = \frac{\mu c_p}{\text{Pr}}. \tag{12}$$

For example, $\text{Pr} = 0.72$ for air. At higher temperatures, this is no longer a good approximation, and we again use the library `Mutation++` to compute the thermal conductivity k from the local state.

Finally, for flows in chemical non-equilibrium, the species source terms $\dot{\boldsymbol{\omega}}$ must be considered. These represent the creation and destruction of the species through chemical reactions. Upon choosing a desired mixture model with a predefined set of N_r elementary reactions, `Mutation++` models the reaction rates using the Law of Mass Action, and provides the local net production rates for each species. The forward rate coefficients are computed based on Arrhenius' law. The forward and backward rate coefficients are linked through the equilibrium constant, satisfying the second law of thermodynamics [37].

To summarize, Table 1 gives an overview of models used in the present work.

Table 1. Overview of models used in the present paper. Please note that the first two models solve only one mass Equation (with $\boldsymbol{J} = \mathbf{0}$ and $\dot{\boldsymbol{\omega}} = \mathbf{0}$).

	Model 1	**Model 2**	**Model 3**
	Perfect Gas	**High-Enthalpy Frozen**	**High-Enthalpy Non-Equilibrium**
Number of equations	$d+2$	$d+2$	N_s+d+1
Equation of state	Equation (8)	Equation (7)	Equation (4)
Viscosity	Equation (9)	Equation (9) or `Mutation++`	`Mutation++`
Thermal Conductivity	Equation (12)	Equation (12) or `Mutation++`	`Mutation++`
Species Diffusion	-	-	`Mutation++`
Production Rates	-	-	`Mutation++`

3. Numerical Discretization

3.1. Hybridized Discontinuous Galerkin Method

The governing equations presented in Section 2 are discretized using a hybridized discontinuous Galerkin (HDG) method. We only give a brief summary here. A more detailed overview of HDG schemes can be found elsewhere [38–40].

Let Ω be the computational domain. For a given a triangulation $\mathcal{T}_h = \{\kappa\}$, and setting $m = N_s + d + 1$ for the number of equations to be solved, we define the approximation spaces

$$V_h := \{\boldsymbol{w} \in (L^2(\Omega))^m : \boldsymbol{w}|_\kappa \in (P^p(\kappa))^m,\ \forall \kappa \in \mathcal{T}_h\} \tag{13}$$

$$\Sigma_h := \{\boldsymbol{\zeta} \in (L^2(\Omega))^{m\times d} : \boldsymbol{\zeta}|_\kappa \in (P^p(\kappa))^{m\times d},\ \forall \kappa \in \mathcal{T}_h\} \tag{14}$$

$$\Lambda_h := \{\boldsymbol{\nu} \in (L^2(\mathcal{E}_h^0))^m : \boldsymbol{\nu}|_e \in (P^p(e))^m,\ \forall e \in \mathcal{E}_h^0\}. \tag{15}$$

Here $\mathcal{E}_h^0$ denotes the set of interior edges, and P^p denotes the space of polynomials with total degree at most p. Upon defining $\mathbb{X}_h := \Sigma_h \times V_h \times \Lambda_h$, the HDG discretization of the steady-state version of the balance law (1) is given as the variational problem

$$\mathbf{x}_h \in \mathbb{X}_h : \qquad N_h(\mathbf{x}_h, \mathbf{y}) = 0, \qquad \forall \mathbf{y} \in \mathbb{X}_h. \tag{16}$$

Defining elementwise integration,

$$\int_{\mathcal{T}_h} \cdot\ dx := \sum_{\kappa\in\mathcal{T}_h} \int_\kappa \cdot\, dx, \qquad\qquad \int_{\partial\mathcal{T}_h^0} \cdot\ ds := \sum_{\kappa\in\mathcal{T}_h} \int_{\partial\kappa\setminus\mathcal{E}_h^\partial} \cdot\, ds,$$

where $\mathcal{E}_h^\partial$ are the boundary edges, and setting $\mathbf{x}_h = (\boldsymbol{\sigma}_h, \boldsymbol{u}_h, \boldsymbol{\lambda}_h)$, $\mathbf{y} = (\boldsymbol{\zeta}, \boldsymbol{w}, \boldsymbol{\nu})$, we can write

$$\begin{aligned} N_h(\mathbf{x}_h, \mathbf{y}) = \int_{\mathcal{T}_h} \boldsymbol{\sigma}_h : \boldsymbol{\zeta}\, dx + \int_{\mathcal{T}_h} (\boldsymbol{u}_h \otimes \nabla) : \boldsymbol{\zeta}\, dx - \int_{\partial\mathcal{T}_h^0} \boldsymbol{\lambda}_h \otimes \boldsymbol{n} : \boldsymbol{\zeta}\, ds - \int_{\mathcal{E}_h^\partial} \hat{\boldsymbol{u}}^b \otimes \boldsymbol{n} : \boldsymbol{\zeta}\, ds \\ - \int_{\mathcal{T}_h} \boldsymbol{f}(\boldsymbol{u}_h, \boldsymbol{\sigma}_h) : \nabla \boldsymbol{w}\, dx + \int_{\partial\mathcal{T}_h^0} \hat{\boldsymbol{f}}(\boldsymbol{\sigma}_h, \boldsymbol{u}_h, \boldsymbol{\lambda}_h; \boldsymbol{n}) \cdot \boldsymbol{w}\, ds + \int_{\mathcal{E}_h^\partial} \hat{\boldsymbol{f}}^b(\boldsymbol{\sigma}_h, \boldsymbol{u}_h; \boldsymbol{n}) \cdot \boldsymbol{w}\, ds \\ - \int_{\mathcal{T}_h} \boldsymbol{S}(\boldsymbol{u}_h, \boldsymbol{\sigma}_h) \cdot \boldsymbol{w}\, dx + \int_{\mathcal{E}_h^0} [\![\hat{\boldsymbol{f}}(\boldsymbol{\sigma}_h, \boldsymbol{u}_h, \boldsymbol{\lambda}_h; \boldsymbol{n})]\!] \cdot \boldsymbol{\nu}\, ds. \end{aligned} \tag{17}$$

The numerical flux function is written as $\hat{\boldsymbol{f}} = \hat{\boldsymbol{f}}_c - \hat{\boldsymbol{f}}_v$, where $\hat{\boldsymbol{f}}_c$ and $\hat{\boldsymbol{f}}_v$ are standard convective and viscous numerical fluxes, respectively. In the present work we use the Lax-Friedichs flux for the former, and the LDG flux for the latter. More precisely, let elements κ^+ and κ^- be separated by edge $e \in \mathcal{E}_h^0$, Then we define the numerical fluxes such that

$$\hat{\boldsymbol{f}}_c \equiv \hat{\boldsymbol{f}}_c^\pm := \boldsymbol{f}(\boldsymbol{\lambda}_h) \cdot \boldsymbol{n}^\pm - \alpha_c(\boldsymbol{\lambda}_h - \boldsymbol{u}_h^\pm) \qquad \text{on } e \tag{18}$$

$$\hat{\boldsymbol{f}}_v \equiv \hat{\boldsymbol{f}}_v^\pm := \boldsymbol{f}(\boldsymbol{\lambda}_h, \boldsymbol{\sigma}_h^\pm) \cdot \boldsymbol{n}^\pm - \alpha_v(\boldsymbol{\lambda}_h - \boldsymbol{u}_h^\pm) \qquad \text{on } e \tag{19}$$

Here $\boldsymbol{n}^{\pm}$ denotes the outward pointing normal on $\partial\kappa^{\pm}$, and $\boldsymbol{u}_h^{\pm}$, $\boldsymbol{\sigma}_h^{\pm}$ are the limits of the respective functions approaching the iterface from $\kappa^{\pm}$. (Please note that $\boldsymbol{\lambda}_h$ is single-valued by definition.) The jump of the total flux on e then reads $[\![\hat{\boldsymbol{f}}]\!] = \hat{\boldsymbol{f}}^{+}\boldsymbol{n}^{+} + \hat{\boldsymbol{f}}^{-}\boldsymbol{n}^{-}$. Again, it ought to be noted that the numerical flux functions are not by definition conservative. Instead, the last term in (17) enforces conservation as a constraint [39–41]. Finally, the boundary fluxes $\widehat{\boldsymbol{u}}^{b}$ and $\hat{\boldsymbol{f}}^{b}(\boldsymbol{u}_h, \boldsymbol{\sigma}_h; \boldsymbol{n})$ are assumed to be consistent with the boundary conditions [42].

To avoid oscillations near discontinuities, the discretization can be augmented with a shock-capturing term. At present, we use a classic method based on artificial viscosity [43,44]. We define

$$N_h^{SC}(\mathbf{x}_h, \mathbf{y}) := \int_{\mathcal{T}_h} \epsilon(\boldsymbol{u}_h)\boldsymbol{\sigma}_h : \nabla \boldsymbol{w}\, dx\,. \tag{20}$$

The artificial diffusion coefficient is computed as a function of the solution $\boldsymbol{u}_h$, such that

$$\epsilon(\boldsymbol{u}_h)|_{\kappa} := \frac{\epsilon_0 (h_\kappa / p_\kappa)^{2-\beta}}{|\kappa|} \int_{\kappa} \sum_{i=1}^{m} |\nabla \cdot \boldsymbol{f}_c(\boldsymbol{u}_h)|\, dx, \qquad \forall \kappa \in \mathcal{T}_h. \tag{21}$$

Here h_κ is the diameter of κ, and p_κ is the polynomial degree of approximation. The parameters ϵ_0 and β can be used to tune the shock-capturing term. In the present work, we set $\beta = 0$. The term (20) can be added to the semilinear form (17) to provide nonlinear stabilization. This approach is somewhat generic in the sense that it is not especially tuned to the physical model under consideration. It has proven successful for transonic and supersonic external aerodynamics. Below we show that the range of application extends to moderately hypersonic flow conditions, not exceeding $M_\infty \approx 8$. For higher Mach numbers more carefully tuned shock capturing may become necessary [30].

Equation (16) is solved using a Newton–Raphson method. Writing $\mathbf{x}_h^{n+1} = \mathbf{x}_h^n + \delta\mathbf{x}_h^n$, the linearized global system is given in block-matrix form as

$$\begin{bmatrix} A & B & R \\ C & D & S \\ L & M & N \end{bmatrix} \begin{bmatrix} \delta Q \\ \delta W \\ \delta \Lambda \end{bmatrix} = \begin{bmatrix} F \\ G \\ H \end{bmatrix}, \tag{22}$$

where the vector $[\delta Q, \delta W, \delta\Lambda]^T$ contains the expansion coefficients of $\delta\mathbf{x}_h^n = (\delta\boldsymbol{\sigma}_h, \delta\boldsymbol{u}_h, \delta\boldsymbol{\lambda}_h)$ with respect to the chosen basis. The matrix on the left represents the Jacobian matrix of $N_h(\mathbf{x}_h, \mathbf{y})$ as a function of these coefficients. In particular, the nonlinear contributions from flux functions, numerical flux functions, and boundary fluxes are often cumbersome to compute. Moreover, these elements change upon implementation of a new physical model. It is here that the algorithmic differentiation mentioned in the introduction is very useful. We come back to this issue in Section 5 below.

Taking the Schur complement, one obtains the hybridized system

$$\left(N - \begin{bmatrix} L & M \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} \begin{bmatrix} R \\ S \end{bmatrix} \right) \delta\Lambda = H - \begin{bmatrix} L & M \end{bmatrix} \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} \begin{bmatrix} F \\ G \end{bmatrix}. \tag{23}$$

System (23) is solved at each Newton iteration using an iterative linear solver. This system is clearly algebraically equivalent to (22). Moreover, the structure of the HDG discretization makes its assembly economically feasible, as the sub-matrix to be inverted is block-diagonal. The system is globally coupled only in the degrees-of-freedom corresponding to $\boldsymbol{\lambda}_h$. This remains true when the shock-capturing term (20) is added. In contrast, standard DG schemes, written in primal form (possibly upon locally eliminating the degrees-of-freedom Q, which can be done for all standard DG schmes [45]), lead to larger linear systems. This is increasingly so for higher polynomial degree p [44]. Indeed, HDG schemes were found to be competitive in terms of efficiency with continuous Galerkin schemes using static condensation [46].

3.2. Adjoint Equations

Let $J(\boldsymbol{u}, \nabla \boldsymbol{u})$ be a solution-dependent functional, and let $J_h(\mathbf{x}_h)$ be a consistent approximation, such that for $\mathbf{x} = (\boldsymbol{u}, \nabla \boldsymbol{u}, \boldsymbol{u}|_{\mathcal{E}_h^0})$, $J_h(\mathbf{x}) = J(\boldsymbol{u}, \nabla \boldsymbol{u})$. Adjoint methods can be used to provide a map to relate variations in the functional to the discrete residual. An approximation is given by the discrete adjoint method, where

$$J_h(\mathbf{x}_h) - J(\boldsymbol{u}, \nabla \boldsymbol{u}) \approx N_h(\mathbf{x}_h, \mathbf{z}_h), \tag{24}$$

provided that $\mathbf{z}_h \in \hat{\mathbb{X}}_h \supset \mathbb{X}_h$ satisfies, for a fixed solution $\mathbf{x}_h \in \mathbb{X}_h$ of the primal problem,

$$N_h'[\mathbf{x}_h](\mathbf{y}, \mathbf{z}_h) = J'[\mathbf{x}_h](\mathbf{y}), \qquad \forall \mathbf{y} \in \hat{\mathbb{X}}_h. \tag{25}$$

Please note that taking the adjoint solution from a superset $\hat{\mathbb{X}}_h \supset \mathbb{X}_h$ is necessary to avoid an identically zero right-hand side in (24). The adjoint-based error estimate can be used to drive a goal-oriented adaptation process [47].

We proved in a previous publication that, with appropriate treatment of boundary conditions, the HDG discretization (17) is adjoint consistent [42,48]. It remains asymptotically adjoint consistent when the shock capturing term (20) is added. In other words, the discrete adjoint equations automatically give a consistent discretization of the adjoint PDE with appropriate boundary conditions. This is true for admissible functionals of the form

$$J(\boldsymbol{u}, \nabla \boldsymbol{u}) = \int_{\partial\Omega} \boldsymbol{\beta} \otimes \boldsymbol{n} : \boldsymbol{f}(\boldsymbol{u}, \nabla \boldsymbol{u})\, ds, \tag{26}$$

where $\boldsymbol{\beta}$ is any L^2 function. For instance, at an adiabatic wall one has, by the definition of the fluxes and the boundary conditions,

$$\boldsymbol{f}(\boldsymbol{u}, \nabla \boldsymbol{u}) \cdot \boldsymbol{n} = (\boldsymbol{f}_c(\boldsymbol{u}) - \boldsymbol{f}_v(\boldsymbol{u}, \nabla \boldsymbol{u})) \cdot \boldsymbol{n} = \begin{pmatrix} 0 \\ p\boldsymbol{n} \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ \boldsymbol{\tau}\boldsymbol{n} \\ 0 \end{pmatrix}. \tag{27}$$

Thus, compatible functionals include force coefficients. For non-adiabatic walls, heat transfer coefficients may also be used [42].

4. Adaptation

Metric-based methods have emerged as an interesting paradigm for mesh adaptation. The idea is to encode a desirable mesh in a tensor-valued mapping, yielding a one-parameter family of s.p.d. matrices, $(\mathcal{M}(x))_{x\in\Omega}$, where $\Omega \subset \mathbb{R}^d$ is some computational domain. A metric-conforming mesh is a triangulation whose elements are (nearly) equilateral under the Riemannian metric induced by $\mathcal{M}$. This mapping is the continuous mesh [15]. In this section, we summarize our continuous mesh optimization, which we used in the computational studies below. More details can be found in [18–22].

4.1. Encoding a Mesh via Metric Tensors

Let $\mathcal{T}_h := \{\kappa\}$ be a triangulation of Ω. For any non-degenerate simplex element, there is a unique symmetric positive definite matrix

$$\mathbb{M} = \begin{pmatrix} m_{11} & m_{12} \\ m_{12} & m_{22} \end{pmatrix} \tag{28}$$

such that,

$$\|\boldsymbol{e}_i\|_{\mathbb{M}}^2 = \boldsymbol{e}_i^T \mathbb{M} \boldsymbol{e}_i = C, \qquad i = 1, 2, 3, \tag{29}$$

where $\boldsymbol{e}_i$ is an edge of the triangle, given as a vector of arbitrary orientation. In other words, the mesh element is equilateral in the norm $\|\cdot\|_{\mathbb{M}}$ induced by the metric $\mathbb{M}$, i.e.,

$$\|\boldsymbol{x}\|_{\mathbb{M}} := \sqrt{\boldsymbol{x}^T \mathbb{M} \boldsymbol{x}}.$$

We say the element is *unit* with respect to $\mathbb{M}$. A straightforward geometric interpretation is shown in Figure 1: The mesh element is inscribed into the unit circle $\|\boldsymbol{x}\|_{\mathbb{M}} = 1$ (i.e., an ellipse).

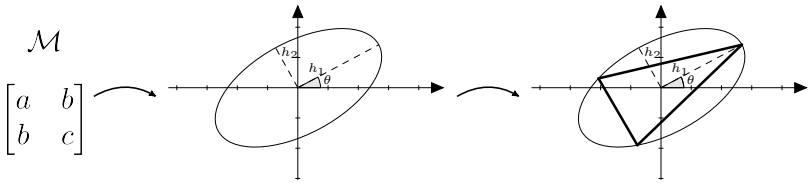

Figure 1. The unit circle of $\mathbb{M}$ defines an ellipse of semi-axes $h_1^2 = \lambda_1^{-1}$, and $h_2^2 = \lambda_2^{-1}$, where $0 < \lambda_1 \leq \lambda_2$ are the eigenvalues of $\mathbb{M}$. The geometry of a unit triangle κ is fixed by (29). The configuration shown corresponds to $C = 3$: The unit element is inscribed into the unit circle. Changing C will change the area $|\kappa|$, but not aspect ratio or orientation.

It will prove useful to characterize the metric by its eigenstructure, i.e., $\mathbb{M} = Q^T \Lambda Q$, where

$$Q = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \qquad \Lambda = \operatorname{diag}(\lambda_1, \lambda_2) \tag{30}$$

Thus, $\mathbb{M}$ may be represented by the tuple (d, ζ, θ), where the angle θ defines the orientation of the eigenvectors in (30), and

$$\zeta := \sqrt{\frac{\lambda_1}{\lambda_2}}, \qquad d := \sqrt{\lambda_1 \lambda_2} \tag{31}$$

are the aspect ratio and mesh density, respectively. The latter definition is motivated by the relation

$$d = \frac{\alpha}{|\kappa|} \tag{32}$$

where the constant α depends only on the constant C in (29).

Given an element-wise metric, one may, using a suitable interpolation, generate a metric field $(\mathcal{M}(x))_{x\in\Omega}$, defined for any $x \in \Omega$. Metric-based mesh generators can generate a discrete mesh, composed of elements which are unit with respect to a metric field, usually interpreted in a least-squares sense, such that one approximates the solution to the minimization problem

$$\mathcal{T}_h^\star = \arg\min_{\mathcal{T}_h} \sum_{\boldsymbol{e}\in\mathcal{T}_h} \left| ||\boldsymbol{e}||_{\mathcal{M}}^2 - C \right|.$$

The length of a segment $\boldsymbol{e} := \boldsymbol{x}_2 - \boldsymbol{x}_1$ under the metric $(\mathcal{M}(\boldsymbol{x}))_{x\in\Omega}$ is now interpreted in a Riemannian sense, using the parametrization

$$||\boldsymbol{e}||_{\mathcal{M}} = \int_0^1 \sqrt{\boldsymbol{e}^T \mathcal{M}(\boldsymbol{x}_1 + s\boldsymbol{e})\boldsymbol{e}}\, \mathrm{ds}, \tag{33}$$

and a suitable quadrature rule. The mesh so produced can be viewed as a discrete approximation of a continuous mesh $(\mathcal{M}(\boldsymbol{x}))_{x\in\Omega}$. Of course, the idea is to generate a new mesh not from the metric field corresponding to the current mesh (in this case not much would change), but from an *optimized* metric $\mathcal{M}^\star$, which is obtained using the solution estimate on the current mesh.

4.2. Optimizing the Continuous Mesh

The optimization is defined as the minimization of an interpolation error into the piecewise polynomial approximation space (13). The local error model is essentially that of a Taylor series,

$$|e_{\mathbf{x},p}(\mathbf{y})| = \frac{1}{(p+1)!}|u^{(p+1)}(\boldsymbol{x};\boldsymbol{\xi})||\boldsymbol{y}-\boldsymbol{x}|^{p+1}, \qquad \forall \boldsymbol{y} \in \Omega, \quad \boldsymbol{\xi} = \frac{\boldsymbol{y}-\boldsymbol{x}}{|\boldsymbol{y}-\boldsymbol{x}|},$$

where the directional derivative is defined, using the usual multiindex notation for $\boldsymbol{\alpha} = (a_1, \ldots, a_d)$,

$$u^{(k)}(\boldsymbol{x};\boldsymbol{\xi}) := \sum_{|\alpha|=k} D^{\alpha}u(\boldsymbol{x})\boldsymbol{\xi}^{\alpha}, \qquad \boldsymbol{\xi} \in \mathbb{R}^2, \ |\boldsymbol{\xi}| = 1.$$

The optimal metric is obtained by a two-step formal optimization procedure. First, following [17], one obtains local values for the optimal aspect ratio and orientation, $(\zeta^{\star}, \theta^{\star})$ leading to a bound on the interpolation error, valid for a triangle centered at $x = x_{\kappa}$, and having locally optimal anisotropy $(\zeta^{\star}(x_{\kappa}), \theta^{\star}(x_{\kappa}))$:

$$\|e^{\text{int}}_{x_K,p}\|^q_{L^q(\kappa)} \leq e_d(x_{\kappa})^q d^{-1}(x_{\kappa}), \tag{34}$$

where

$$e_d(x) := \left(\frac{2\pi}{q(p+1)+2}\right)^{\frac{1}{q}} \bar{A}(x) d(x)^{-\frac{p+1}{2}}, \tag{35}$$

and $\bar{A} = \sqrt{A_{\max}A^{\perp}}$. Here $A_{\max}$ is the maximum directional derivative

$$A_{\max} = \frac{1}{(p+1)!}\max_{|\xi|=1}\left|u^{(p+1)}(\boldsymbol{x};\xi)\right|$$

while $A^{\perp}$ is the derivative in direction orthogonal to that defining $A_{\max}$.

Please note that the mesh density d is still unspecified. Imagine now a triangulation composed of elements with arbitrary size distribution, but each having locally optimal anisotropy. Then, starting from (34), one has asymptotically,

$$\|e^{\text{int}}_{x_K,p}\|^q_{L^q(\Omega)} \leq \sum_{\kappa\in\mathcal{T}_h} e_d(x_K)^q d^{-1}(x_K) \propto \sum_{\kappa\in\mathcal{T}_h} e_d(x_K)^q|\kappa| \to \int_{\Omega} e_d(x)^q\,dx, \qquad (h \to 0). \tag{36}$$

The function $x \mapsto e_d(x)$ is termed the continuous interpolation error. This leads to a constraint minimization problem

$$\int_{\Omega} e_d^q(x)\,dx \to \min \qquad \text{s.th.} \qquad \int_{\Omega} d(x)\,dx = \mathcal{N}. \tag{37}$$

Here $\mathcal{N}$ is the mesh complexity, which is related to the number of mesh elements via the constant α in (32). One may apply straightforward calculus of variations to solve for the density distribution $d^{\star}$. Finally, one obtains

$$d^{\star}(x) = C\bar{A}(x)^{\frac{2q}{(p+1)q+2}}, \tag{38}$$

where

$$C = \frac{\mathcal{N}}{\displaystyle\int_{\Omega} \bar{A}^{\frac{2q}{(p+1)q+2}}\,dx}. \tag{39}$$

The optimization presented here is thus with respect to the error model, measured in any L^q norm. The approach may be extended to goal-oriented adaptation. In the most straightforward formulation, the resulting methodology is very similar to L^1-optimization,

using the formulation presented here, but with additional weights coming from an adjoint solution. We omit discussion here, and refer to our earlier work [22]. See also related work in [21,49].

5. Implementation and Code Structure

Our solver framework has a modular code structure (Figure 2). We give a brief overview only insofar as it facilitates discussing the implementation of new physical models. A more detailed documentation is given in [31].

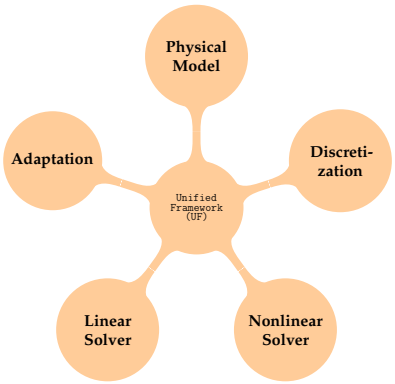

Figure 2. Graphical Illustration of the solver structure: Each Module is made available as a class or external library.

The code is written in C++, using the Finite Element library Netgen/NGsolve [50,51]. Each module shown in Figure 2 is made available as a class, or as an external library. An example of the latter is the linear solver, which is made available through the `PETSc` library [52]. The most important example of the former is the physical modeling. Each model is required to implement certain functions modeling Equation (1) (convective flux, diffusive flux, boundary conditions, etc.). Listing 1 shows the class definition. The template parameters D and $COMP$ refer to the physical dimension and number of state vector components, respectively.

Listing 1. Definition of a Model.

```
template <int D, int COMP>
class MyModel {
...
};
```

For the present work, this data structure has been enlarged to allow more flexible physical modeling, by decoupling the definition of the fluxes in the model and the closure equations used to evaluate them. We follow the Strategy pattern as defined by Gamma et al. [53] to set up a uniform interface between the general models describing the conservation laws, and families of different algorithms that implement the closure equations. A specific implementation is then linked to a model through a template parameter $CLOS$, as shown in Listing 2.

Listing 2. Model definition including new interface.

```
template <int D, int COMP, class CLOS>
class MyModel {
...
};.
```

Physical models, such as the compressible Euler equations, can thus use different thermodynamic and transport models through well-defined interfaces. For example, the

perfect gas model, as well as the model for thermally perfect gases at higher temperatures discussed in Section 2, use the same model class, but have different closure classes as template parameters. The latter includes an interface to the library `Mutation++`.

As an example, consider the convective flux. Convective flux evaluation is a required public function within the model class. For the Euler or Navier–Stokes equations, the implementation using the new thermodynamic interface given in Listing 3.

Listing 3. Function to evaluate the convective flux for the compressible Euler or Navier–Stokes equations.

```
template <typename SCAL>
static void EvalConvFlux(Vec<COMP, SCAL> & state,
    SpatialParams<D> & sparam,
  Mat<COMP, D, SCAL> & res) {

  SCAL p;
  Vec<D, SCAL> U;
  CLOS::EvalPrim(state, U, p);

  res = state * Trans(U);
  res.Rows(1, D+1) += p * Id<D>();
  res.Row(D+1) += p * U;
}
```

Please note that the thermodynamics is contained in the evaluation of the primitive variables, which is thus relegated to the new module. The structure `sparam` contains several mesh variables, such as the node location of the current flux evaluation. This allows implementation of fluxes that depend on the physical coordinates, although it is not needed for the Euler equations. Please note that the template parameter `SCAL` allows function evaluation using standard data types, as well as custom made algorithmic differentiation data types. We come back to this in Section 5.

All model-dependent functions are made available through such public functions. From the solver, which is itself a class, these functions can be queried. Listing 4 illustrates this for evaluation of convective and diffusive fluxes. These calls are used in the volume quadrature when assembling the residual or the Jacobian of the discretization (17).

Listing 4. Calling Model Functions from the Solver Kernel.

```
if   (Model::Convection)
       Model::EvalConvFlux(w, sparam, fc);
if   (Model::Diffusion)
       Model::EvalDiffFlux(w, q, sparam, fv);
f = fc -fv;
```

The model itself is a template parameter to the main solver class (Listing 5). The code can thus be compiled to solve different physical models using the same solver kernel. Implementing new models usually requires implementation of a new (or modified) model class.

Notice that the solver also takes the same template parameters D and $COMP$ as the model. The discretization is written for arbitrary values of these parameters. For instance, the compressible Navier–Stokes equations are implemented only once. Two- or three-dimensional flow can be selected by compiling with $D = 2$, or $D = 3$, respectively. Moreover, models with arbitrary number of components are easily implemented.

Listing 5. The solver class.

```
template <int D, int COMP, class Model>
class UnifyingFramework : public NumProc {
...
};
```

Recall that our code is built on the library `Netgen / NGSolve`. In Listing 5, the inherited class `NumProc` is indeed a class defined in `NGSolve` , and may be used to register our solver as an object inside the NGSolve framework. The code is then compiled as a dynamic library which may be loaded by the `NGSolve` executable at runtime.

Algorithmic Differentiation

The solver is intended to function in a research environment, where modeling of different physical phenomena is frequent. The overall code structure is designed to facilitate code maintenance, as well as the implementation of new models and numerical methods. The modular structure described in the previous section follows this rationale. However, as we argued in the introduction, flexibility must extend to differentiation of physical models. An accurate differentiation of the residual function not only enables fast convergence of a Newton solver (Section 3.1), but also allows direct use of the system Jacobian in a discrete adjoint context. We therefore opted for an approach using algorithmic differentiation.

Algorithmic differentiation is implemented via a class-based data structure. The `AutoDiff` class, made available by the `NGSolve` library, carries both values and derivatives. Within the class definition, methods are defined to overload standard operations and functions. Figure 3 illustrates the concept, using multiplication as an example.

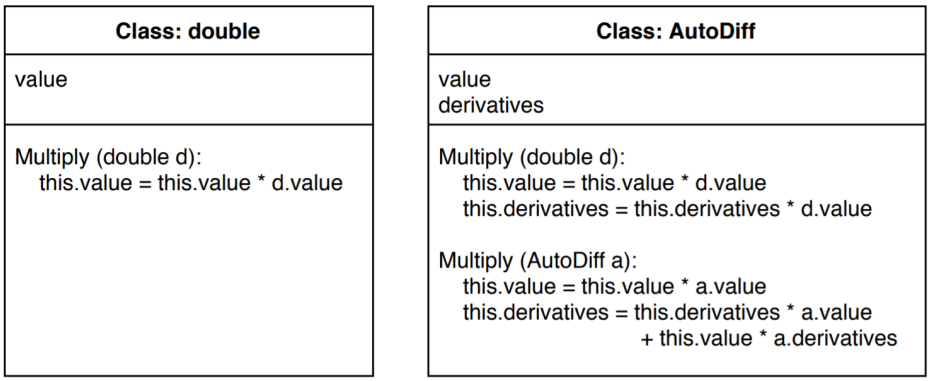

Figure 3. Conceptual example of operator overloading in the AutoDiff class. Multiplication is redefined to update the derivatives as well as the value. This is done for all operations used by the models.

Individual routines contributing to the computation of the residual, such as the flux functions shown in Listing 3, are templated and can thus take `AutoDiff` arguments.

However, interfaces to undifferentiated external libraries are problematic. The most straightforward solution, if manual differentiation is not deemed economically feasible, is to use (local) finite-difference approximation. On the other hand, alternatives may be available. In particular, for some quantities computed by the library `Mutation++` , thermodynamic calculus may be used to fill the data structure locally with exact derivatives. As an example, consider the computation of temperature. Using the templated flux interface in Listing 3, the elements of the state vector are passed as `AutoDiff` types. The temperature is obtained

(as a double) directly from `Mutation++` as a function of the local state. The starting point for computing the derivatives is the relation for the internal energy,

$$\rho e = \sum_i^{N_s} \rho_i e_i(T) = E - \frac{\rho \boldsymbol{v} \cdot \boldsymbol{v}}{2}, \tag{40}$$

and noting $de_i = c_{v,i} dT$. A straightforward computation yields

$$\frac{\partial T}{\partial \rho_j} = \frac{(\boldsymbol{v} \cdot \boldsymbol{v})/2 - e_j(T)}{\sum_i^{N_s} \rho_i c_{v,i}}, \qquad \frac{\partial T}{\partial \rho \boldsymbol{v}} = \frac{-\boldsymbol{v}}{\sum_i^{N_s} \rho_i c_{v,i}}, \qquad \frac{\partial T}{\partial E} = \frac{1}{\sum_i^{N_s} \rho_i c_{v,i}}. \tag{41}$$

One may thus instantiate the temperature as an `AutoDiff` by directly storing $\nabla_{\boldsymbol{u}} T$ computed in this way. For frozen flow, the computation simplifies somewhat, as derivatives with respect to the individual species densities are not needed. Instead, temperature derivatives (with respect to the state) can be assigned easily using

$$e = \frac{E}{\rho} - \frac{1}{2} \boldsymbol{v} \cdot \boldsymbol{v}, \tag{42}$$

which gives the mixture internal energy as an `AutoDiff` (i.e., carries the derivatives), and using the mixture specific heat via $de = c_v dT$. The mixture specific heat can be obtained directly from `Mutation++`.

In either case, other variables using the temperature analytically are now automatically differentiated. In other words, they can be directly computed as an `AutoDiff` type without further manual input. Examples are the pressure using (4), or the viscosity using Sutherlands law (if applicable). In fact, if additionally the thermal conductivity is computed using a constant Prandtl number approximation, the frozen flow model has exact flux derivatives. If instead the transport coefficients are obtained directly from `Mutation++`, local difference approximations are currently used to produce derivatives for the `AutoDiff` type. `Mutation++` can compute the derivatives of the species production rates with respect to species densities. This is used in the present work. At present, the derivatives of the species production rates with respect to the temperature are computed using finite difference approximations.

6. Results

We present numerical results using the modeling approach, discretization, and adaptation discussed in preceding Sections. Focus is on verification of the implementation, the interplay between the physical modeling, shock capturing, and the convergence of the Newton solver. The interface to the `PETSc` library makes is possible to use a variety of linear solvers. However, a study of linear solvers or preconditioning is not our aim. We therefore computed all results shown here with the same rather general setup, using a GMRES(m) method to solve the linear systems (where $m = 120$), along with an ILU(k) preconditioner and levels of fill $3 \leq k \leq 15$. Using the same setting for all cases allows a direct comparison between different modeling approaches, and to assess influence of the modeling and adaptive procedure on solver performance.

6.1. Verification of Thermodynamics Implementation

Initial verification of the new thermodynamics interface may be provided by considering moderate flow conditions, where the previously implemented perfect gas model ought to be valid. The new models should then yield very similar results. Subsequently, we consider a case at higher temperature, where the caloriacly perface gas assumption ceases to be valid, and finite rate chemistry becomes important.

We shall frequently compare Models 1–3 in Table 1. In the case of the frozen flow high-enthalpy model (Model 2 in Table 1), we generally use `Mutation++` to compute the transport

coefficients. For all results shown below the NASA 9-coefficient polynomial database has been used with `Mutation++` (see [23] and references therein for database options).

6.1.1. Laminar Flow Around a Cylinder

We choose the Reynolds number small enough so that a steady flow develops. Free stream conditions are given by atmospheric pressure ($p_\infty = 1$ atm) and moderate temperature ($T_\infty = 298.15$) K. For the cylinder, adiabatic no-slip wall boundary conditions are used. In this setup, all models should yield very similar results. We used a low free stream Mach number $M_\infty = 0.1$ to stay near the incompressible regime. As a secondary effect, using low Mach numbers puts additional strain on the solver, as it increases stiffness. (At present, no preconditioning for low Mach numbers is used.) Figure 4 shows the results in terms of the drag coefficient.

Adapted grids, using the method of Section 4, were created for each of these Reynolds numbers using the perfect gas model. All results for a given Reynolds number were then computed on the same grid. Solver results are compared to an empirical correlation for incompressible flow [54]. It can be seen that the new thermodynamics modules reproduce the results of the previously implemented model.

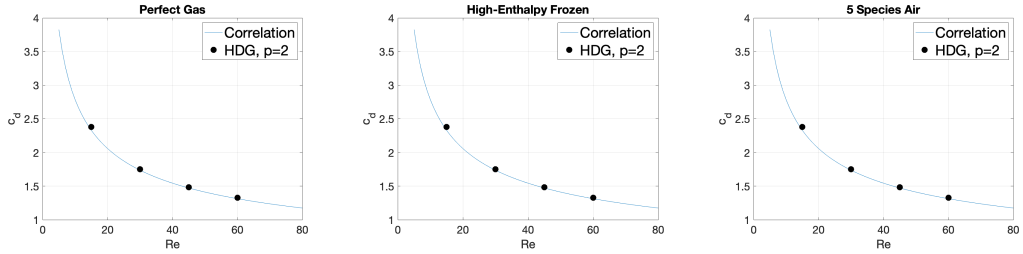

Figure 4. Drag as a function of Reynolds number for laminar flow around a cylinder. Results are computed on triangular adapted grids with ca. 1200 elements. From left to right, Models 1–3 in Table 1 are shown. Model 2 uses `Mutation++` to compute the transport coefficients. Model 3 uses a 5-species air model.

The non-equilibrium multispecies model can be used with a variety of mixture models. In Figure 4, we show results using a five-species air model. For the given flow conditions, only two species (O_2 and N_2) are significant. When computing the same case with a corresponding 2-species model, the solver yields identical results, up to the decimal accuracy used for output (Table 2). The solver is able to keep the non-significant species at machine-zero without problems during the flow evolution process. (We initialize the flow field with free-stream conditions.)

Table 2. Cylinder Test case. Results for the drag coefficient using 2-Species and 5-Species air models.

Re	2-Species Air	5-Species Air
30	1.75088641	1.75088641
45	1.48528380	1.48528380
60	1.32895637	1.32895637

Furthermore, of interest is the convergence of the Newton solver. Figure 5 shows the convergence curves for three representative cases and different models.

Generally, we measure the residual of the globally coupled degrees of freedom by taking the l^2 norm of the right-hand-side of Equation (23). It can be seen that the convergence is indeed very similar for all models. The low Reynolds numbers of this test case emphasize the viscous terms. These contain the transport coefficients which are potentially approximated by finite-differences. (The differentiation of Sutherland's law is analytic.)

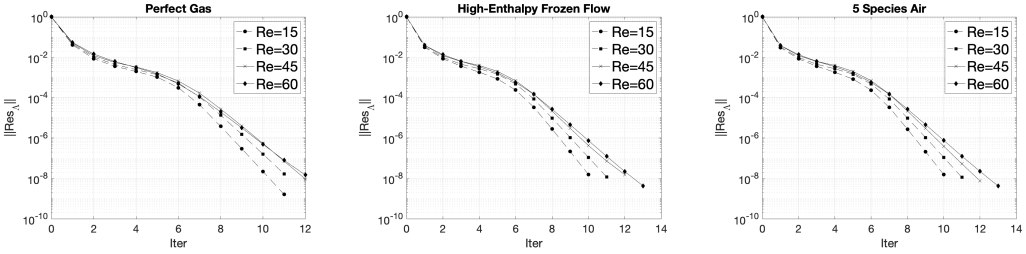

Figure 5. Convergence in terms of Newton iterations for laminar flow around a cylinder.

6.1.2. Post-Shock Relaxation

We consider a simulation of flow relaxation to chemical equilibrium behind a strong normal shock. Table 3 shows the pre- and post-shock conditions of the flow.

Across the shock, the chemistry is assumed to be frozen. Thus, the immediate post-shock species composition is identical to the pre-shock composition. This composition, together with the post-shock flow conditions, are assumed as inflow conditions for our computation. Not that this inflow state is far from equilibrium. The flow relaxes toward chemical equilibrium, as it proceeds downstream the shock. Similar cases have previously been used by several researchers [36,55–57]. We use a five-component, single-temperature inviscid air model with `Mutation++` (Model 3 in Table 1). We compare our results to an independently verified 1D code, produced at the von Karman Institut for Fluid Dynamics, which also uses `Mutation++` [58].

Table 3. Pre- and post-shock conditions for the shock relaxation test case.

	T [K]	p [Pa]	u [m/s]
Pre-shock	300	100	4000
Post-shock	6539	16,101	542

To simulate this one-dimensional case, a thin slab of 1.5 m $\times$ 0.002 m with a graded mesh of approximately 1500 triangles was created, concentrating elements at the domain inlet. At the left and right boundaries, inlet and outlet conditions are used, respectively, imposing the mass fractions, temperature and velocity, respectively, the pressure. Slip-wall boundary conditions in transverse direction were used.

Figures 6 and 7 show the results of the simulation.

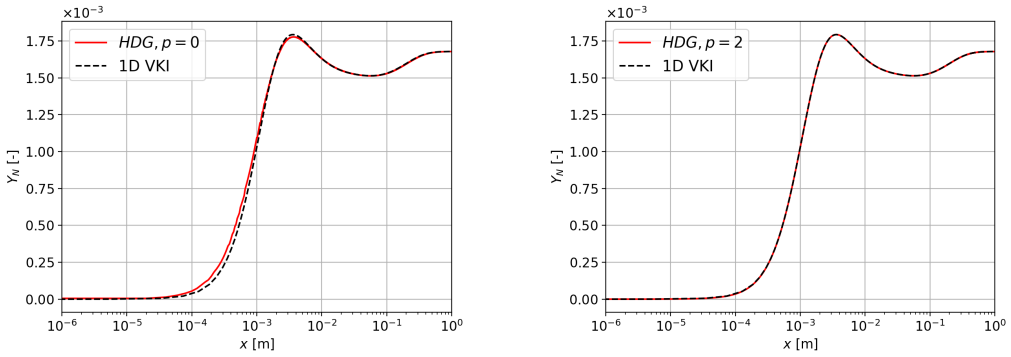

Figure 6. *Cont.*

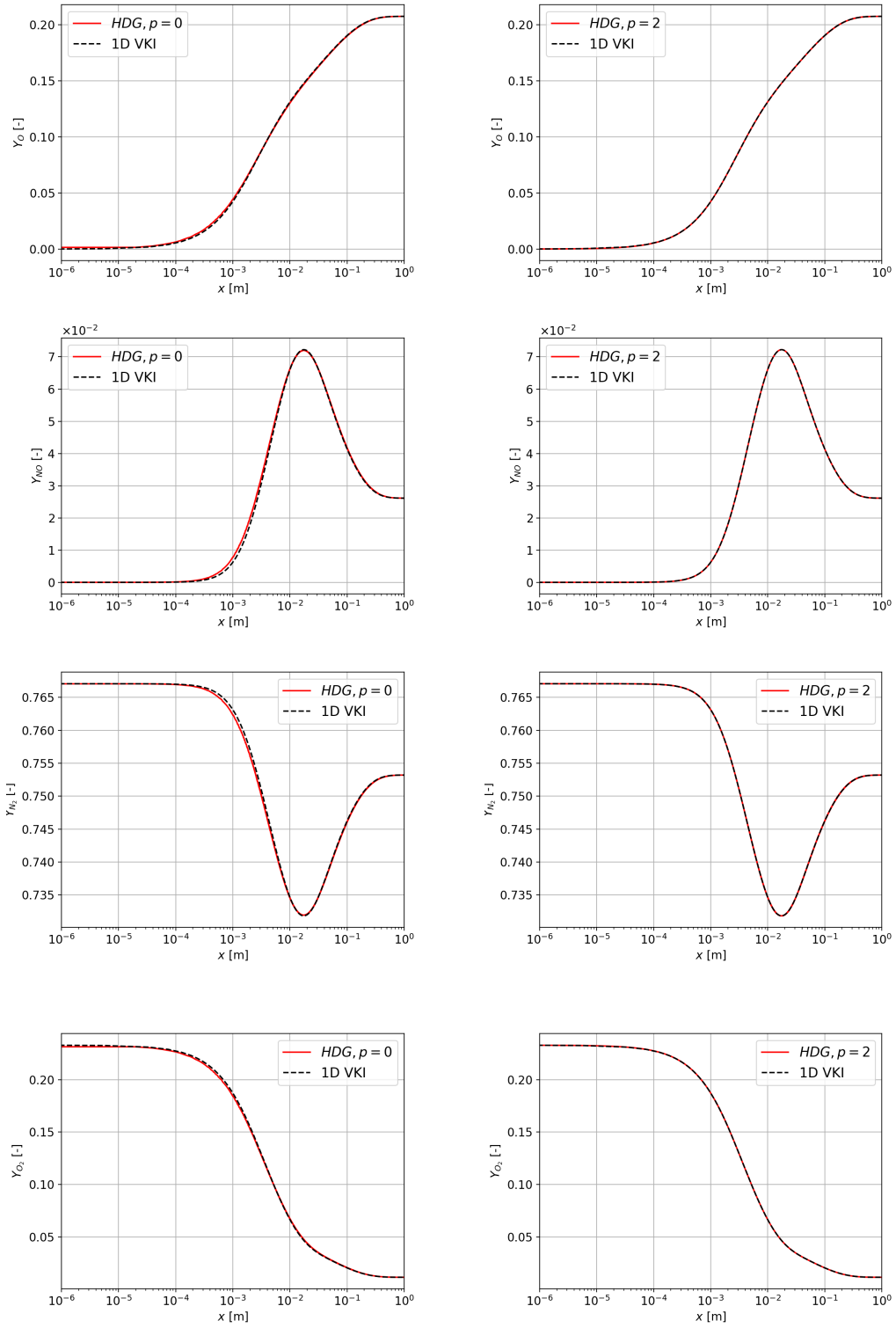

Figure 6. Post-shock relaxation case, species fractions for different polynomial degrees p. (**Left Column**): $p = 0$. (**Right Column**): $p = 2$.

There is excellent agreement between the output of the reacting Euler model implemented in the HDG framework, and that of the 1D VKI code.

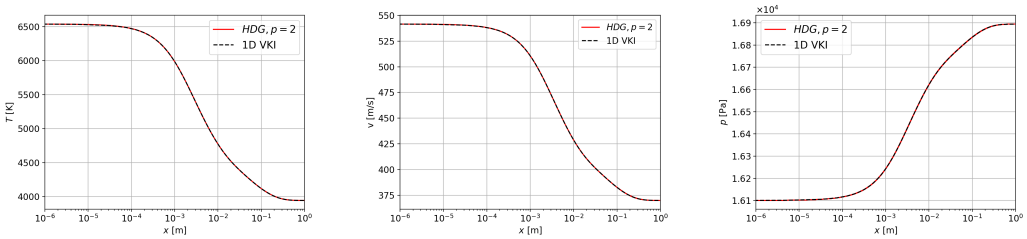

Figure 7. Post-shock relaxation. Various bulk flow quantities ($p = 2$).

6.2. High–Mach–Number Flow

We are particularly interested in the interplay between the metric-based adaptation, shock capturing, and modeling. We shall consider flows in the supersonic to moderately hypersonic regime. Shock capturing terms are available for the frozen flow models, i.e., Models 1 and 2 in Table 1, which will be used in this section. (As we stated in Section 3.1, our shock-capturing method is able to reliably resolve shocks up to around $M_\infty \approx 8$.) We consider external aerodynamics, and to avoid additional complications from the ensuing high Reynolds numbers, inviscid flow.

6.2.1. Diamond Profile

In this section, we solve inviscid supersonic to hypersonic flow around a diamond-shaped profile. The geometry of the profile is defined by a wedge angle of 10°, and a chord length of $c = 1$. Free-stream Mach numbers ranging from $M_\infty = 2$ to $M_\infty = 8$ were used, along with adiabatic slip-wall boundary conditions at the airfoil. Figure 8 shows two representative Mach number contour plots.

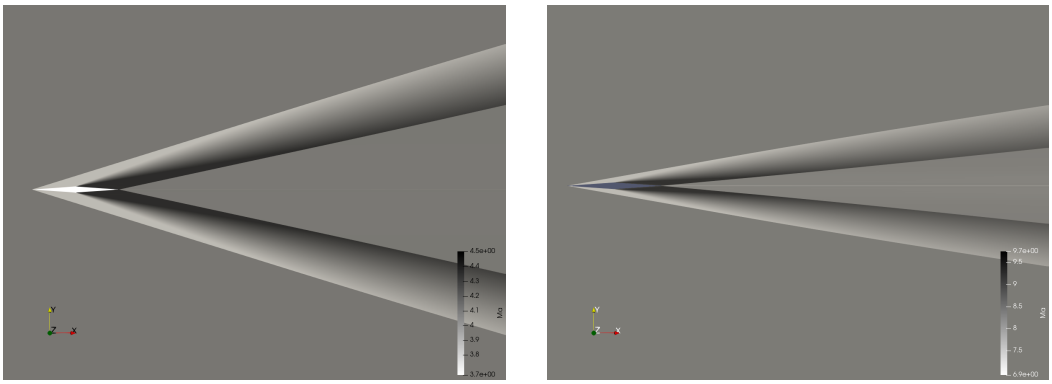

Figure 8. Diamond Airfoil Test Case. Contour plots of Mach number using the perfect gas model. (**Left**): $M_\infty = 4$, adapted mesh with 43,751 triangles ($p = 2$). (**Right**): $M_\infty = 8$, adapted mesh with 46,366 triangles ($p = 2$).

Results are computed on adapted meshes. Figure 9 shows the initial mesh used for all cases in this section, as well as a representative mesh produced by the adaptation sequence. The complexity constraint in Equation (37) is increased by 20% in each adaptation cycle. This produces successively finer meshes, each of which with optimized distribution of degrees-of-freedom (cf. Section 4).

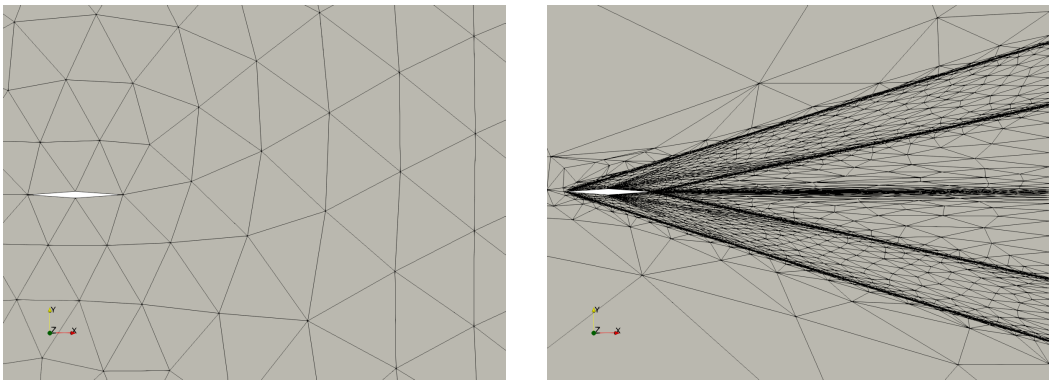

Figure 9. Diamond Airfoil Test Case. (**Left**): Initial mesh with 268 elements. (**Right**): Mesh adapted on L^2-norm, perfect gas model (12,272 elements, $M_\infty = 4$).

For the perfect gas model, the exact solution for the drag coefficient can be computed from the shock-jump relations and Prandtl-Meyer expansion around the tip of the airfoil. The left plot in Figure 10 shows a comparison between the exact values and the solution computed on adapted grids.

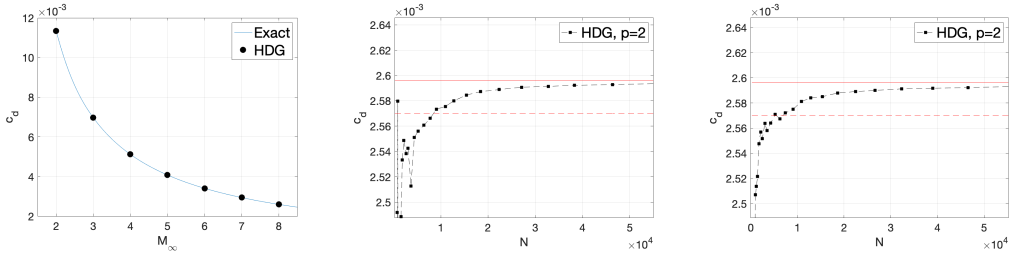

Figure 10. Diamond Airfoil, computation of drag coefficient. (**Left**): Comparison with exact values, perfect gas model. Solution on adapted grids with ca. 20,000 elements. (**Middle**/**Right**): perfect gas/high-enthalpy frozen flow model. Convergence, $M_\infty = 8$ using L^2-based Mesh optimization. The solid red line is the reference value. The dashed red line indicates values within 1% of the reference value.

The high-temperature models admit calorically imperfect gases. For this model the perfect-gas solution for the drag is not exact. Nevertheless, deviations are relatively small, and the perfect gas solution has been used as a reference value for all models. Figure 10 shows the convergence of the drag coefficient as a function of the number of mesh elements. Both the perfect gas model and the high-temperature model produce a solution within 1% of the drag value on relatively coarse meshes. Empirically, it has been found that the convergence as a function of DoF is fairly insensitive to the amount by which they are increased. The regeneration of the mesh, based on the metric, results not only in increase of DoF, but also in redistribution. As a consequence, a higher fraction of the added DoF (and even the existing DoF) can be placed in critical regions, and thus be leveraged more effectively.

As we discussed in Section 3.1, we use a generic shock-capturing term. Up to the Mach numbers discussed here, shocks are captured cleanly. Moreover, the adaptation process, which is able to isolate shocks on anisotropic elements, is an important ingredient in the overall shock-capturing approach. In fact, up to moderate Mach numbers, the adaptation and the diffusion injected by the numerical fluxes sufficiently stabilize the solution, and converged solutions can be obtained without nonlinear stabilization provided by the shock capturing term. A representative result is shown in Figure 11.

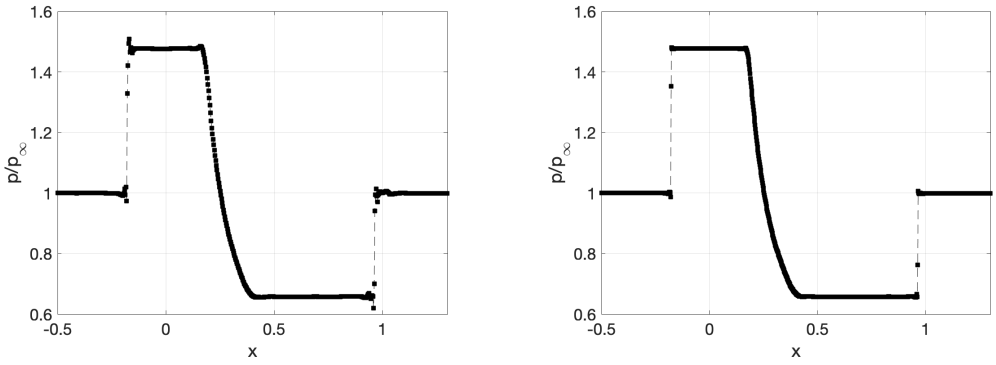

Figure 11. Diamond airfoil ($M_\infty = 4$), perfect gas model. Line plot of non-dimensional pressure at $y = 0.1$. (**Left**): No shock capturing. (**Right**): Shock capturing using Equation (20).

For higher Mach number, the oscillations produced at discontinuities become significant enough to impede the convergence process, and the shock capturing term must be used to obtain fully converged steady-state solutions.

Convergence of force coefficients can be dramatically improved with goal-oriented adaptation. For this purpose, we used the methods of Section 4 with the adjoint-based modification discussed in [22]. Figure 12 shows a representative mesh produced when adapting on the drag coefficient.

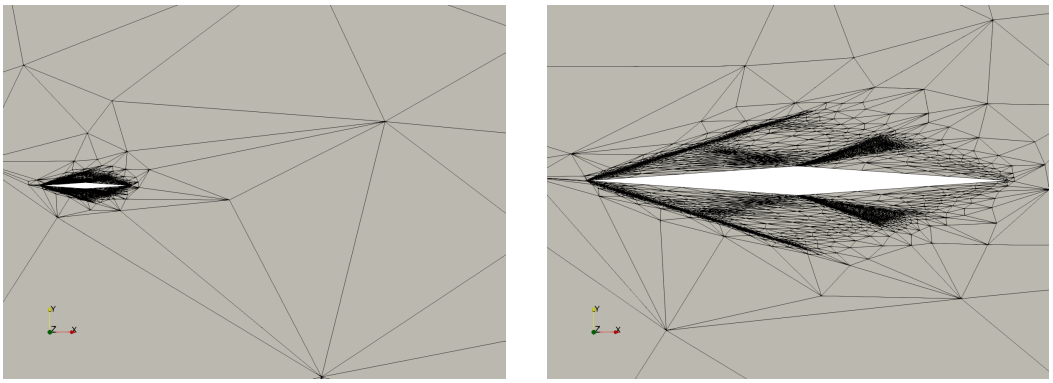

Figure 12. (**Left**): Diamond airfoil $M_\infty = 4$, $p = 2$, perfect gas model. Drag-adapted mesh with ca. 6187 triangles. Compare to Figure 9. (**Right**): Magnified view.

Clearly, the shock is not resolved beyond the near field. In contrast, as we showed in Figure 9, norm-based optimization will resolve the entire shock into the farfield, irrespective of whether or not this is necessary for accurate computation of the drag. The goal-oriented approach can thus potentially save a large number of degrees of freedom. Indeed, Figure 13 demonstrates that adjoint-based adaptation leads to much better convergence in the target quantity.

Finally, Figure 14 shows the convergence of the nonlinear solver on adapted grids.

These curves are representative for the convergence on all the adapted meshes. Exceptions to the pattern of reliable convergence begin to emerge for $M_\infty = 8$, where a few runs on intermediate adapted meshes did not fully converge. (For $M_\infty = 4$ all runs converged to machine zero.) The rightmost plot in Figure 14 shows an example, where the residual convergence stalls after a reduction of about three orders of magnitude. For even higher Mach numbers, as we mentioned in Section 3.1, the currently implemented generic shock-capturing procedure is not expected to be the best choice.

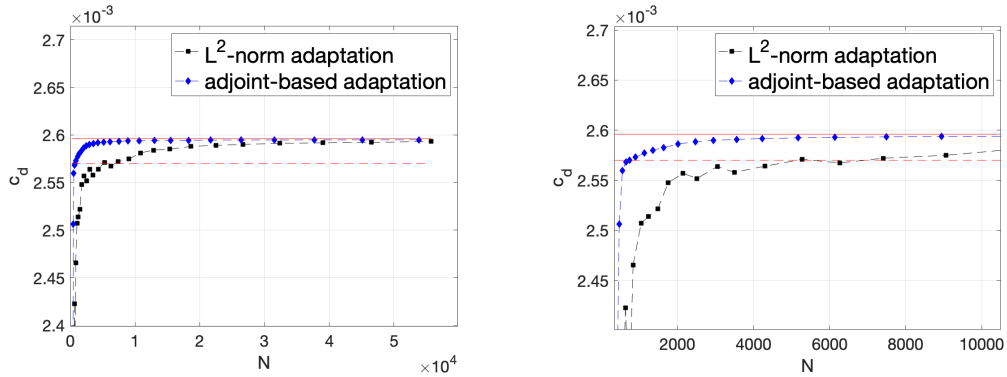

Figure 13. Diamond Airfoil, drag convergence. (High-temperature model, $M_\infty = 8$, $p = 2$). The right figure shows a magnified view. The solid red line is the reference value. The dashed red line indicates values within 1% of the reference value.

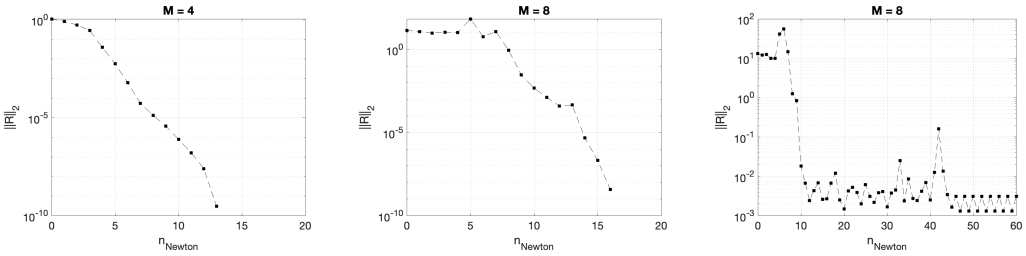

Figure 14. Solver Convergence on adapted mesh for Diamond Wing Test Case, goal-oriented adaptation on drag. (**Left**): $M = 4$. (**Middle**) $M = 8$. (**Right**): $M = 8$. Example of stalled convergence.

6.2.2. NACA 0012

Consider supersonic to hypersonic inviscid flow around the NACA0012 airfoil. This case is somewhat more challenging, as it includes a stronger normal shock along the stagnation line, and a curved shock, bending into the farfield. Again, we use goal-oriented adaptation, based on the drag coefficient. Figures 15 and 16 show adapted meshes.

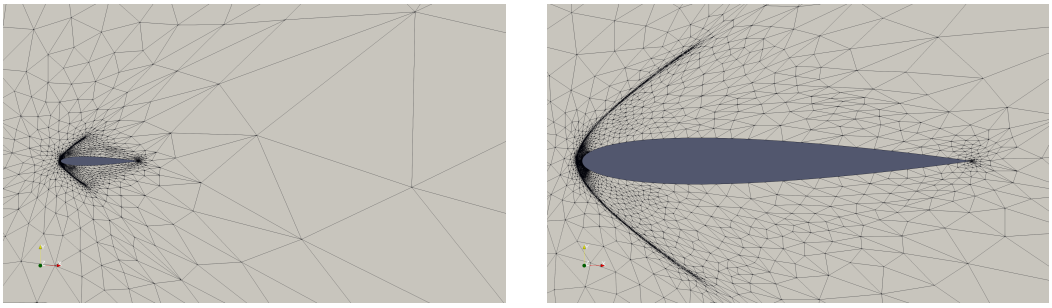

Figure 15. Drag-adapted mesh for NACA0012 Airfoil test case (6756 elements, $M_\infty = 3$, $p = 2$). The right figure shows a magnified view.

Shock are again resolved only as far as necessary for the computation of the drag coefficient. For free stream conditions $M_\infty = 5$, Figure 17 shows a plot of Mach number and pressure along the stagnation line, using the solution obtained from the high-enthalpy frozen flow model (Model 2 in Table 1). It is clearly visible that the shock is well-captured.

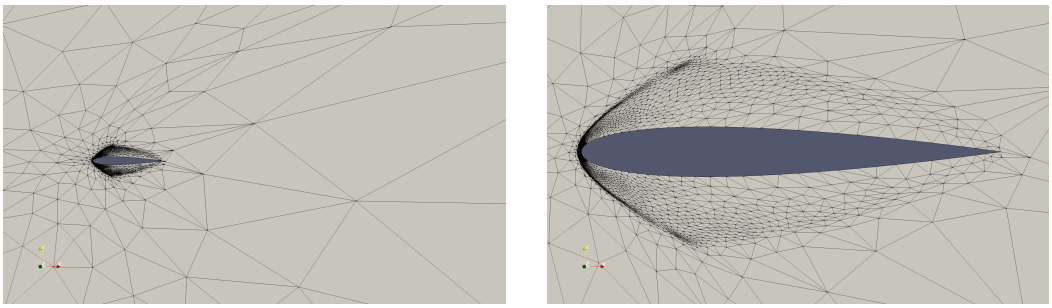

Figure 16. Drag-adapted mesh for NACA0012 Airfoil test case (6711 elements, $M_\infty = 5$, $p = 2$.). The right figure shows a magnified view.

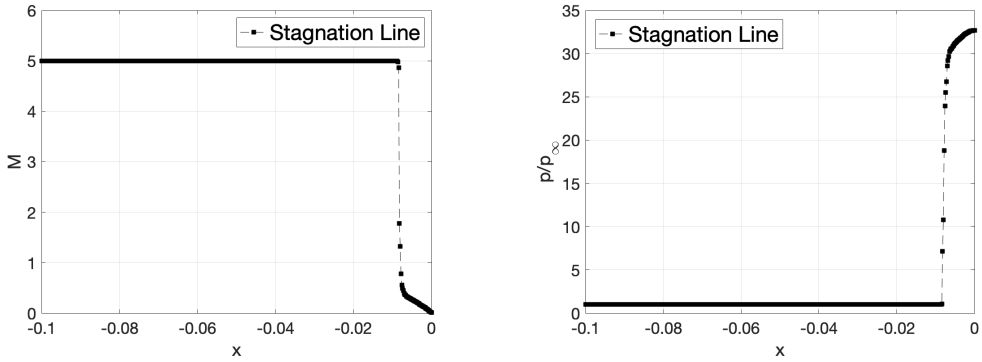

Figure 17. HDG, $p = 2$. Mach number (**left**) and pressure p/p_∞ (**right**) along the stagnation line.

7. Conclusions and Outlook

We presented design and initial verification of a high-order, HDG-based solver for high-speed flows, and, more broadly, flows requiring non-trivial physical modeling. We used object-oriented programming, and algorithmic differentiation (AD), to encapsulate different aspects of the modeling. Fluxes, source terms, and boundary conditions are implemented inside a model class, which is made available to the solver as a template parameter. These functions are differentiated automatically using the AD data structure. Closure equations and transport coefficients are implemented in a separate class, made available to the model, again as a template parameter. This hierarchical approach allows fairly flexible, non-intrusive implementation of physical models, and will thus serve as a stepping stone for future endeavors. For example, work is underway to implement modeling for flows in local thermodynamic equilibrium (LTE), as well as plasma flows. In addition, the shock capturing mechanism needs to be extended to multispecies non-equilibrium models and higher Mach numbers.

Moreover, future work will focus on more extensive validation campaigns regarding species diffusion and, in particular, heat flux into surfaces. In this context, extensions are planned to deal with gas-surface interaction on catalytic surfaces. This requires more complex boundary conditions than those considered in the present paper. Nevertheless, the same data structure can be used.

Efficient simulation of compressible flows using high-order methods requires adaptation. The metric-based approach was shown to be a very promising approach for high-speed flow. Moreover, the AD data structure in conjunction with adjoint-consistent discretization enables one to use a discrete adjoint approach for new physical models without implementation overhead. The adjoint-based approach leads to superior results, provided a suitable target of the computation can be identified. In the present paper, drag has been used. However, heat flux, in conjunction with non-adiabatic walls, is also a viable target, and will be considered in future work.

Author Contributions: Conceptualization, G.M.; Software, G.M., K.D., A.R. and T.M.; Supervision, G.M. and T.M.; Validation, G.M., K.D. and A.R.; Visualization, G.M., K.D. and A.R.; Writing—original draft, G.M. All authors have read and agreed to the published version of the manuscript.

Funding: The development of the Mutation++ library was sponsored by the European Space Agency (GSTP and TRP programs), the European Research Council (Starting Grant #259354) and Air Force Office of Scientific Research (grant FA9550-18-1-0209). The development of the solver was funded in part by the Deutsche Forschungsgemeinschaft (German Research Association) through grant GSC 111.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hughes, T.J.R. Recent Progress in the Development and understanding of SUPG Methods with Special Reference to the Compressible Euler and Navier–Stokes Equations. *Int. J. Numer. Meth. Fluids* **1987**, *7*, 1261–1275. [CrossRef]
2. Jameson, A. Computational transonics. *Commun. Pure Appl. Math.* **1988**, *41*, 507–549. [CrossRef]
3. Hughes, T.J.R.; Franca, L.P.; Hulbert, G.M. A new finite element formulation for computational fluid dynamics: VIII. The galerkin/least-squares method for advective-diffusive equations. *Comput. Methods Appl. Mech. Eng.* **1989**, *73*, 173–189. [CrossRef]
4. Jameson, A. Aerodynamics. In *Encyclopedia of Computational Mechanics*; Stein, E., De Borst, R., Hughes, T.J.R., Eds.; Wiley: Hoboken, NJ, USA, 2004; Volume 3, Section 11.
5. Candler, G.V.; Mavriplis, D.J.; Trevino, L. *Current Status and Future Prospects for the Numerical Simulation of Hypersonic Flows*; AIAA Paper 09-0153; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2009. [CrossRef]
6. Gnoffo, P.A.; Johnston, C.O.; Kleb, B. *Challenges to Computational Aerothermodynamic Simulation and Validation for Planetary Entry Vehicle Analysis*; Report RTO-EN-AVT-186; NATO Research and Technology Organization, 2010. Available online: https://core.ac.uk/display/10573004 (accessed on 26 October 2021).
7. Economon, T.D.; Palacios, F.; Copeland, S.R.; Lukaczyk, T.W.; Alonso, J.J. SU2: An Open-Source Suite for Multiphysics Simulation and Design. *AIAA J.* **2015**, *54*, 828–846. [CrossRef]
8. Anderson, W.K.; Newman, J.C.; Karman, S.L. Stabilized Finite Elements in FUN3D. *J. Aircr.* **2017**, *55*, 696–714. [CrossRef]
9. Wang, Z.J.; Fidkowski, K.; Abgrall, R.; Bassi, F.; Caraeni, D.; Cary, A.; Deconinck, H.; Hartmann, R.; Hillewaert, K.; Huynh, H.T.; et al. High-order CFD methods: Current status and perspective. *Int. J. Numer. Methods Fluids* **2013**, *72*, 811–845. [CrossRef]
10. Becker, R.; Rannacher, R., An Optimal Control Approach to a Posteriori Error Estimation in Finite Element Methods. In *Acta Numerica*; Iserles, A., Ed.; Cambridge University Press: Cambridge, UK, 2001; Volume 10, pp. 1–102.
11. Frey, P.J.; Alauzet, F. Anisotropic mesh adaptation for CFD computations. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 5068–5082. [CrossRef]
12. Coupez, T. Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing. *J. Comput. Phys.* **2011**, *230*, 2391–2405. [CrossRef]
13. Yano, M.; Darmofal, D.L. An optimization-based framework for anisotropic simplex mesh adaptation. *J. Comput. Phys.* **2012**, *231*, 7626–7649. [CrossRef]
14. Alauzet, F.; Loseille, A. A decade of progress on anisotropic mesh adaptation for computational fluid dynamics. *Comput.-Aided Des.* **2016**, *72*, 13–39. [CrossRef]
15. Loseille, A.; Alauzet, F. Continuous Mesh Framework Part I: Well-Posed Continuous Interpolation Error. *SIAM J. Numer. Anal.* **2011**, *49*, 38–60. [CrossRef]
16. Loseille, A.; Alauzet, F. Continuous Mesh Framework Part II: Validations and Applications. *SIAM J. Numer. Anal.* **2011**, *49*, 61–86. [CrossRef]
17. Dolejší, V. Anisotropic hp-adaptive method based on interpolation error estimates in the Lq-norm. *Appl. Numer. Math.* **2014**, *82*, 80–114. [CrossRef]
18. Rangarajan, A.; Balan, A.; May, G. Mesh Optimization for Discontinuous Galerkin Methods Using a Continuous Mesh Model. *AIAA J.* **2018**, *56*, 4060–4073. [CrossRef]
19. Rangarajan, A.M.; Chakraborty, A.; May, G.; Dolejsi, V. *A Continuous-Mesh Optimization Technique for Piecewise Polynomial Approximation on Tetrahedral Grids*; AIAA Paper; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2018. [CrossRef]
20. Dolejší, V.; May, G.; Rangarajan, A. A continuous hp-mesh model for adaptive discontinuous Galerkin schemes. *Appl. Numer. Math.* **2018**, *124*, 1–21. [CrossRef]
21. Bartoš, O.; Dolejší, V.; May, G.; Rangarajan, A.; Roskovec, F. A goal-oriented anisotropic hp-mesh adaptation method for linear convection–diffusion–reaction problems. *Comput. Math. Appl.* **2019**, *78*, 2973–2993.
22. Rangarajan, A.; May, G.; Dolejsi, V. Adjoint-based anisotropic hp-adaptation for discontinuous Galerkin methods using a continuous mesh model. *J. Comput. Phys.* **2020**, *409*, 109321. [CrossRef]
23. Scoggins, J.B.; Leroy, V.; Bellas-Chatzigeorgis, G.; Dias, B.; Magin, T.E. Mutation++: MUlticomponent Thermodynamic and Transport properties for IONized gases in C++. *SoftwareX* **2020**, *12*, 100575. [CrossRef]

24. Bassi, F.; Rebay, S. A High-Order Accurate Discontinuous Finite-Element Method for the Numerical Solution of the Compressible Navier–Stokes Equations. *J. Comp. Phys.* **1997**, *131*, 267–279. [CrossRef]
25. Lomtev, I.; Karniadakis, G.E. A Discontinuous Galerkin Method for the Navier–Stokes Equations. *Int. J. Numer. Meth. Fluids* **1999**, *29*, 587–603. [CrossRef]
26. Witherden, F.D.; Farrington, A.M.; Vincent, P.E. PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach. *Comput. Phys. Commun.* **2014**, *185*, 3028–3040. [CrossRef]
27. Hindenlang, F.; Gassner, G.J.; Altmann, C.; Beck, A.; Staudenmaier, M.; Munz, C.D. Explicit discontinuous Galerkin methods for unsteady problems. *Comput. Fluids* **2012**, *61*, 86–93. [CrossRef]
28. Ching, E.J.; Lv, Y.; Gnoffo, P.; Barnhardt, M.; Ihme, M. Shock capturing for discontinuous Galerkin methods with application to predicting heat transfer in hypersonic flows. *J. Comput. Phys.* **2019**, *376*, 54–75. [CrossRef]
29. Lv, Y.; Ihme, M. Discontinuous Galerkin method for multicomponent chemically reacting flows and combustion. *J. Comput. Phys.* **2014**, *270*, 105–137. [CrossRef]
30. Terrana, S.; Hoskin, D.; Eichstädt, J.; Nguyen, N.C.; Peraire, J. *GPU-Accelerated Large Eddy Simulation of Hypersonic Flows*; AIAA Paper 20-1062; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2020. [CrossRef]
31. Woopen, M.; Balan, A.; May, G. *A Unifying Computational Framework for Adaptive High-Order Finite Element Methods*; AIAA Paper 15-2601; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2015. [CrossRef]
32. Anderson, J.D. *Hypersonic and High Temperature Gas Dynamcis*; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2006.
33. Gnoffo, P.A.; Gupta, R.N.; Shinn, J.L. *Conservation Equations and Physical Models for Hypersonic Air Flows in Thermal and Chemical Nonequilibrium*; NASA Technical Paper 2867; National Aeronautics and Space Administration: Washington, DC, USA, 1989.
34. Sutherland, W. The viscosity of gases and molecular force. *Philos. Mag.* **1893**, *5*, 507–531. [CrossRef]
35. Sutton, K.; Gnoffo, P.A. *Multi-Component Diffusion with Application To Computational Aerothermodynamics*; AIAA Paper 98-2575; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 1998.
36. Schrooyen, P.; Hillewaert, K.; Magin, T.E.; Chatelain, P. Fully implicit Discontinuous Galerkin solver to study surface and volume ablation competition in atmospheric entry flows. *Int. J. Heat Mass Transf.* **2016**, *103*, 108–124. [CrossRef]
37. Giovangigli, V. *Multicomponent Flow Modeling*; Birkhäuser: Boston, MA, USA, 1999.
38. Cockburn, B.; Gopalakrishnan, J.; Lazarov, R. Unified Hybridization of Discontinuous Galerkin, Mixed, and Continuous Galerkin Methods for Second Order Elliptic Problems. *SIAM J. Numer. Anal.* **2009**, *47*, 1319–1365. [CrossRef]
39. Nguyen, N.C.; Peraire, J.; Cockburn, B. An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations. *J. Comput. Phys.* **2009**, *228*, 3232–3254. [CrossRef]
40. Nguyen, N.C.; Peraire, J.; Cockburn, B. An implicit high-order hybridizable discontinuous Galerkin method for nonlinear convection-diffusion equations. *J. Comput. Phys.* **2009**, *228*, 8841–8855. [CrossRef]
41. Schütz, J.; May, G. A hybrid mixed method for the compressible Navier–Stokes equations. *J. Comput. Phys.* **2013**, *240*, 58–75. [CrossRef]
42. May, G. *Hybridized Discontinuous Galerkin Methods: Formulation and Discrete Adjoint*; VKI 38th lecture series on advanced computational fluid dynamics; von Karman Institute for Fluid Dynamics: Sint-Genesius-Rode, Belgium, 2015.
43. Hartmann, R.; Houston, P. Adaptive Discontinuous Galerkin Finite Element Methods for the Compressible Euler Equations. *J. Comp. Phys.* **2002**, *183*, 508–532. [CrossRef]
44. Woopen, M.; Balan, A.; May, G.; Schütz, J. A comparison of hybridized and standard DG methods for target-based hp-adaptive simulation of compressible flow. *Comput. Fluids* **2014**, *98*, 3–16. [CrossRef]
45. Arnold, D.N.; Brezzi, F.; Cockburn, B.; Marini, L.D. Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems. *SIAM J. Num. Anal.* **2002**, *39*, 1749–1779. [CrossRef]
46. Kirby, R.M.; Sherwin, S.J.; Cockburn, B. To CG or to HDG: A Comparative Study. *J. Sci. Comput.* **2012**, *51*, 183–212. [CrossRef]
47. May, G. *Output-Based Error Estimation and Mesh Adaptation for HDG Methods*; VKI 38th lecture series on advanced computational fluid dynamics; von Karman Institute for Fluid Dynamics: Sint-Genesius-Rode, Belgium, 2015.
48. Schütz, J.; May, G. An adjoint consistency analysis for a class of hybrid mixed methods. *IMA J. Numer. Anal.* **2014**, *34*, 1222–1239. [CrossRef]
49. Loseille, A.; Dervieux, A.; Alauzet, F. Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations. *J. Comput. Phys.* **2010**, *229*, 2866–2897. [CrossRef]
50. Schöberl, J. *C++11 Implementation of Finite Elements in NGSolve*; ASC Report ASC Report No. 30/2014; Institute for Analysis and Scientific Computing-Vienna University of Technology: Vienna, Austria, 2014.
51. Schöberl, J. NETGEN An advancing front 2D/3D-mesh generator based on abstract rules. *Comput. Vis. Sci.* **1997**, *1*, 41–52. [CrossRef]
52. Balay, S.; Buschelman, K.; Eijkhout, V.; Gropp, W.D.; Kaushik, D.; Knepley, M.G.; McInnes, L.C.; Smith, B.F.; Zhang, H. *PETSc Users Manual*; Report ANL-95/11-Revision 3.15; Argonne National Laboratory: Lemont, IL, USA, 2004.
53. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*; Addison-Wesley: Boston, MA, USA, 1995.
54. Henderson, R.D. Details of the Drag Curve Near the Onset of Vortex Shedding. *Phys. Fluids* **1995**, *7*, 2102–2104. [CrossRef]

55. Klomfaß, A. Hyperschallströmungen im Thermodynamischen Nichtgleichgewicht. Ph.D. Thesis, RWTH Aachen, Aachen, Germany, 1995.
56. Kumar, S. Numerical Simulation of Chemically Reactive Hypersonic Flows. Ph.D. Thesis, RWTH Aachen, Aachen, Germany, 2006.
57. Windisch, C. Efficient Simulation of Thermochemical Nonequilibrium Flows Using Highly-Resolved H-Adapted Grids. Ph.D. Thesis, RWTH Aachen, Aachen, Germany, 2014.
58. Magin, T.E.; Caillault, L.; Bourdon, A.; Laux, C.O. Nonequilibrium radiative heat flux modeling for the Huygens entry probe. *J. Geophys. Res. Planets* **2006**, *111*, E07S12. [CrossRef]