Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

HTR solver: An open-source exascale-oriented task-based multi-GPU high-order code for hypersonic aerothermodynamics[☆,☆☆]

Mario Di Renzo[*], Lin Fu, Javier Urzay

Center for Turbulence Research, Stanford University, Stanford, CA 94305, USA

ARTICLE INFO

Article history:
Received 30 September 2019
Received in revised form 19 February 2020
Accepted 28 February 2020
Available online 14 March 2020

Keywords:
Hypersonic aerothermodynamics
GPUs
High-order numerics

ABSTRACT

In this study, the open-source Hypersonics Task-based Research (HTR) solver for hypersonic aerothermodynamics is described. The physical formulation of the code includes thermochemical effects induced by high temperatures (vibrational excitation and chemical dissociation). The HTR solver uses high-order TENO-based spatial discretization on structured grids and efficient time integrators for stiff systems, is highly scalable in GPU-based supercomputers as a result of its implementation in the Regent/Legion stack, and is designed for direct numerical simulations of canonical hypersonic flows at high Reynolds numbers. The performance of the HTR solver is tested with benchmark cases including inviscid vortex advection, low- and high-speed laminar boundary layers, inviscid one-dimensional compressible flows in shock tubes, supersonic turbulent channel flows, and hypersonic transitional boundary layers of both calorically perfect gases and dissociating air.

Program summary
Program Title: Hypersonics Task-based Research solver
Program Files doi: http://dx.doi.org/10.17632/9zsxjtzfr7.1
Licensing provisions: BSD 2-clause
Programming language: Regent
Nature of problem: This code solves the Navier–Stokes equations at hypersonic Mach numbers including finite-rate chemistry for air dissociation along with multicomponent transport. The solver is designed for direct numerical simulations (DNS) of transitional and turbulent hypersonic turbulent flows at high enthalpies, and accounts for thermochemical effects such as vibrational excitation and chemical dissociation.
Solution method: This code uses a low-dissipation sixth-order targeted essentially non-oscillatory (TENO) scheme for the spatial discretization of the conservation equations on Cartesian stretched grids. The time advancement is performed either with an explicit method, when the chemistry is slow and therefore does not introduce additional stiffness in the integration, or with an operator-splitting method that integrates the chemical production rates with an implicit discretization.
Additional comments: The HTR solver builds on the runtime Legion [1] and is written in the programming language Regent [2] developed at Stanford University. Instructions for the installation of the components are provided in the README file enclosed with the HTR solver and in the Legion repository [1].

References:
[1] Legion web page: https://legion.stanford.edu
[1] Regent web page: http://regent-lang.org

1. Introduction

Numerical simulations of hypersonic flows are laborious and require special considerations because of several effects induced by the prevailing high Mach numbers. These complexities are compounded by turbulence in cases of practical engineering interest, which oftentimes involve high Reynolds numbers.

The overall aspect of the flow that the engineer or designer of hypersonic flight systems has to typically deal with is one with

[☆] The review of this paper was arranged by Prof. N.S. Scott.
[☆☆] This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
[*] Corresponding author.
E-mail address: mariodr@stanford.edu (M. Di Renzo).

https://doi.org/10.1016/j.cpc.2020.107262
0010-4655/© 2020 Elsevier B.V. All rights reserved.

shock waves and turbulence, with transitional zones in compressible boundary layers, and with large fluctuations of temperature, velocity, and pressure [1–3]. High Mach numbers involve high kinetic energies relative to the thermal energy of the flow. However, high kinetic energies also lead to the occurrence of high temperatures in regions where the flow slows down, because of the close coupling between the transfer of momentum and thermal energy. The resulting high temperatures may activate thermochemical processes such as vibrational excitation, dissociation, and ionization [4–6].

The intense computational resources required to solve the multiscale dynamics of hypersonic flows often forestall simulation-based discovery and analysis of fundamental phenomena. Common issues are the numerical stiffness induced by the strong interplay between compressibility, turbulence, and thermochemical effects, and the increasing similarities between shock-like and eddy-like structures at high values of Mach and Reynolds numbers, which curtail the effectiveness of numerical schemes.

This study describes an open-source code, the Hypersonics Task-based Research (HTR) solver, for direct numerical simulations (DNS) of hypersonic turbulent flows subject to thermochemical effects in structured grids. The HTR solver is suitable for deployment in GPU-based high-performance supercomputing facilities, and is furnished with novel capabilities in computational science and numerical methods, along with physical models to investigate phenomena induced by high temperatures in hypersonic flows, as summarized below.

The architecture of today's computing facilities is characterized by utilizing a variety of computing chips, including CPUs and GPUs. Such heterogeneity entails endless memory hierarchies that the programmer has to deal with in pursuit of making full use of the supercomputer. In order to achieve this goal, parallel CFD solvers based on traditional domain-decomposition methods require complex adaptations to the particular architecture of the machine in question. In contrast, the HTR solver is based on the task-based environment provided by the versatile library Legion recently developed at Stanford and thought to bear potential for exploitation of future Exascale machines [7–9]. The task-based structure enables efficient partition of the work load at runtime in the available computing chips, while the required memory-management operations are managed by the Legion infrastructure. This makes possible the exploitation of heterogeneous architectures without incurring costly and tailored modifications of the CFD solver for one particular machine. The HTR solver is written in the programming language Regent designed to simplify the user interface with Legion runtime [10]. Using Regent, one can write parallel programs employing sequential semantics, which are automatically translated by the framework into OpenMP and CUDA kernels that can be used by the runtime.

In simulating fundamental multiscale processes in hypersonic flows at high Reynolds numbers using DNS, the numerical discretization of the conservation equations in time should handle the stiffness induced by vastly different time scales with efficiency and stability, whereas the discretization in space should minimize numerical dissipation in order to capture turbulent fluctuations, albeit in a way that also allows for stable capture of shocks, the latter becoming increasingly more indistinguishable from small-scale eddies as the turbulent Mach number increases. The HTR solver employs two different time-advancement schemes, namely a third-order Runge–Kutta method [11] when the chemical reactions are slow, and an operator-splitting method [12] when chemical reactions are fast and the numerical integration becomes correspondingly stiff. For spatial discretization, the HTR solver uses a low-dissipation sixth-order targeted essentially non-oscillatory (TENO) scheme as a compromise solution between low numerical dissipation and stable capture of shocks [13].

The high temperatures prevailing in hypersonic flows require consideration of non-calorically-perfect effects. The HTR solver is based on a multicomponent transport formulation that includes variable specific heat capacities [14] and transport coefficients [15–22], along with a chemical-kinetic description for air dissociation [4]. This formulation enables the investigation of phenomena induced by vibrational excitation of air molecules and their dissociation in hypersonic flows at high Reynolds numbers. However, in configurations where the stagnation temperature is sufficiently cold, or thermochemical effects are ignored, the HTR solver also includes a simplified formulation for calorically perfect gases.

This manuscript describes the first release of the HTR solver, which incorporates the characteristics mentioned above and is aimed at DNS of canonical hypersonic flows. It should however be stressed that the flexibility afforded by the Legion/Regent stack should make possible for the developer a number of extensions of this work if an augmentation of the capabilities of the code is sought. For instance, the numerical framework and the implementation of the code released are organized in such a way that they are completely independent of the chemical mechanism, and of the thermophysical and transport properties employed in the simulations, thereby providing ample room for improvement in physical models.

The remainder of this paper is structured as follows. Section 2 outlines the conservation equations, whereas Section 3 describes the chemical mechanism and the thermophysical and transport properties implemented in the solver. Section 4 focuses on the numerical discretization of the conservation equations. The solver is verified in Section 6 using inviscid vortex advection, low- and high-speed laminar boundary layers, inviscid one-dimensional compressible flows in shock tubes, supersonic turbulent channel flows, and hypersonic transitional boundary layers. While most of these benchmark cases pertain to low-enthalpy conditions, DNS results of a hypersonic transitional boundary layer obtained using the HTR solver at high enthalpies are highlighted in Section 7. Additionally, an assessment of the parallel performance of code in both GPU and CPU environments is performed in Section 8. Lastly, conclusions are provided in Section 9. The first release of the HTR solver, along with the setup files for the benchmark cases analyzed in Section 6, and for the extra case in Section 7, are included in the Supplementary Material.

2. Formulation of the conservation equations in the HTR solver

The HTR solver integrates the dimensional compressible Navier–Stokes conservation equations for a chemically reacting mixture of ideal gases, which can be written in compact form as

$$\frac{\partial \mathbf{C}}{\partial t} + \frac{\partial[\mathbf{F}(\mathbf{C}) + \mathbf{F}_\nu(\mathbf{C})]}{\partial x} + \frac{\partial[\mathbf{G}(\mathbf{C}) + \mathbf{G}_\nu(\mathbf{C})]}{\partial y} + \frac{\partial[\mathbf{H}(\mathbf{C}) + \mathbf{H}_\nu(\mathbf{C})]}{\partial z} = \dot{\mathbf{S}} \tag{1}$$

in time t and Cartesian coordinates $\{x, y, z\}$. In this formulation, $\mathbf{C}$ is a vector of conserved variables defined as

$$\mathbf{C} = \left[\rho_1, \ldots, \rho_{N_s}, \rho u, \rho v, \rho w, \rho e_0\right]^{\mathrm{T}}, \tag{2}$$

where the components correspond to the following quantities: (a) the partial densities $\rho_i = \rho Y_i$ of the N_s species in the mixture, with ρ and Y_i being the mixture density and mass fraction of species i, respectively; (b) the three components of the momentum per unit volume ρu, ρv, and ρw, where $\{u, v, w\}$ are the velocity components in the aforementioned Cartesian coordinate system; and (c) the specific stagnation internal energy $e_0 = e + |\mathbf{u}|^2/2$, with e being the specific internal energy of the mixture defined as

$$e = \sum_{i=1}^{N_s} Y_i h_i - P/\rho, \tag{3}$$

where P is the thermodynamic pressure and h_i is the partial specific enthalpy of species i given by

$$h_i = h_{i,\mathrm{ref}} + \int_{T_\mathrm{ref}}^{T} c_{p,i}(T')dT'. \tag{4}$$

The symbol $h_{i,\mathrm{ref}}$ denotes a reference value taken at the reference temperature T_ref. In Eq. (4), $c_{p,i}$ is the specific heat capacity of species i at constant pressure. These equations are supplemented with the equation of state

$$P/\rho = \sum_{i=1}^{N_s} Y_i \mathrm{R}^0 T/\mathcal{M}_i \tag{5}$$

for an ideal multicomponent gas, where R^0 is the universal gas constant, and $\mathcal{M}_i$ is the molecular weight of species i. Note that Eq. (5) can be generally expressed in functional form as $P = f(e, \rho_1, \ldots \rho_{N_s})$, with the partial derivatives $P_{\rho_i} = (\partial P/\partial \rho_i)_{e,\rho_j|j=1,\ldots,N_s(j\neq i)}$ and $P_e = (\partial P/\partial e)_{\rho_i|i=1,\ldots,N_s}$ being equal to constant values in the particular case of a calorically perfect gas. In Eq. (1), **F**, **G**, and **H** are Euler fluxes given by

$$\mathbf{F}(\mathbf{C}) = \begin{bmatrix} \rho_1 u \\ \vdots \\ \rho_{N_s} u \\ \rho u u + P \\ \rho u v \\ \rho u w \\ \rho u h_0 \end{bmatrix}, \quad \mathbf{G}(\mathbf{C}) = \begin{bmatrix} \rho_1 v \\ \vdots \\ \rho_{N_s} v \\ \rho v u \\ \rho v v + P \\ \rho v w \\ \rho v h_0 \end{bmatrix}, \quad \mathbf{H}(\mathbf{C}) = \begin{bmatrix} \rho_1 w \\ \vdots \\ \rho_{N_s} w \\ \rho w u \\ \rho w v \\ \rho w w + P \\ \rho w h_0 \end{bmatrix}, \tag{6}$$

where $h_0 = e_0 + P/\rho$ is the specific stagnation enthalpy of the mixture. Similarly, $\mathbf{F}_\nu$, $\mathbf{G}_\nu$, and $\mathbf{H}_\nu$ are diffusive fluxes defined as

$$\mathbf{F}_\nu(\mathbf{C}) = \begin{bmatrix} \rho_1 U_1 \\ \vdots \\ \rho_{N_s} U_{N_s} \\ -\tau_{11} \\ -\tau_{21} \\ -\tau_{31} \\ \sum_{i=1}^{N_s} \rho_i U_i h_i - \lambda \dfrac{\partial T}{\partial x} - \tau_{11} u - \tau_{12} v - \tau_{13} w \end{bmatrix},$$

$$\mathbf{G}_\nu(\mathbf{C}) = \begin{bmatrix} \rho_1 V_1 \\ \vdots \\ \rho_{N_s} V_{N_s} \\ -\tau_{12} \\ -\tau_{22} \\ -\tau_{32} \\ \sum_{i=1}^{N_s} \rho_i V_i h_i - \lambda \dfrac{\partial T}{\partial y} - \tau_{21} u - \tau_{22} v - \tau_{23} w \end{bmatrix},$$

and

$$\mathbf{H}_\nu(\mathbf{C}) = \begin{bmatrix} \rho_1 W_1 \\ \vdots \\ \rho_{N_s} W_{N_s} \\ -\tau_{13} \\ -\tau_{23} \\ -\tau_{33} \\ \sum_{i=1}^{N_s} \rho_i W_i h_i - \lambda \dfrac{\partial T}{\partial z} - \tau_{31} u - \tau_{32} v - \tau_{33} w \end{bmatrix}. \tag{7}$$

In these expressions, τ_{ij} $(i, j = 1, 2, 3)$ are the components of the viscous stress tensor

$$\bar{\bar{\boldsymbol{\tau}}} = \mu \left[\nabla \mathbf{u} + \nabla \mathbf{u}^T - 2\left(\nabla \cdot \mathbf{u}\right) \bar{\bar{\mathbf{I}}}/3\right], \tag{8}$$

where $\bar{\bar{\mathbf{I}}}$ is the identity tensor and μ is the dynamic viscosity of the mixture. Additionally, $\{U_i, V_i, W_i\}$ are the components of the diffusion velocity vector

$$\mathbf{V}_i = -D_i \nabla\left(\ln X_i\right) + \sum_{j=1}^{N_s} Y_j D_j \nabla \left(\ln X_j\right) \tag{9}$$

in the Cartesian coordinate system defined above. The right-hand side of Eq. (9) for the diffusion velocity is composed of two terms. The first one is a Fickian term, whereas the second one is a mass corrector [23–25]. In the notation, X_i and D_i are the molar fraction and mass diffusivity of species i, respectively.

The chemical mechanism is formally given by the set of $j = 1, 2, \ldots, M$ elementary steps $\sum_{i=1}^{N_s} \nu'_{ij}\mathcal{R}_i \rightleftharpoons \sum_{i=1}^{N_s} \nu''_{ij}\mathcal{R}_i$, with $\mathcal{R}$ the chemical symbol of species i, ν'_{ij} the stoichiometric coefficient of the reactant i in the step j on the reactants side, and ν''_{ij} the stoichiometric coefficient of the reactant i in the step j on the products side. The source term in Eq. (1) is defined as $\dot{\mathbf{S}} = \left[\dot{w}_1, \ldots, \dot{w}_{N_s}, 0, 0, 0, 0\right]^T$ in terms of the chemical rates of mass production of species i per unit volume,

$$\dot{w}_i = \mathcal{M}_i \sum_{j=1}^{M} (\nu''_{ij} - \nu'_{i}) \sum_{i=1}^{N_s} F_{ij} \left(\frac{\rho Y_i}{\mathcal{M}_i}\right) \times \left[k_{f,j} \prod_{i=1}^{N_s} \left(\frac{\rho Y_i}{\mathcal{M}_i}\right)^{\nu'_{ij}} - k_{b,j} \prod_{i=1}^{N_s} \left(\frac{\rho Y_i}{\mathcal{M}_i}\right)^{\nu''_{ij}}\right], \tag{10}$$

where F_{ij} is the chaperon efficiency of species i participating as collider in the reaction j, and $k_{f,j}$ and $k_{b,j}$ are, respectively, the forward and backward rate constants of the chemical step j.

3. Chemical kinetics and thermophysical and transport properties implemented in the HTR solver

A number of different sets of thermophysical, transport, and kinetic properties can be supplied to the HTR solver depending on the particular application because of the flexible modularity of the Legion/Regent framework. However, in this study, the analysis is narrowed down to two different sets of mixtures for showing the performance of the implementation provided in the open-source version of the HTR solver, as described below.

3.1. *Calorically perfect gas*

For calorically perfect gases, the specific heat capacity c_p is constant and is related to the adiabatic coefficient γ as $c_p = \gamma R_g/(\gamma - 1)$, where $R_g = R^0/\mathcal{M}$ is the gas constant. In addition, since there is no chemical conversion and $N_s = 1$, the diffusion velocities (9) and the chemical production rates (10) are identically zero. The dynamic viscosity of the gas is computed using the power law $\mu = \mu_{\text{ref}}\,(T/T_{\text{ref}})^{\sigma}$ with $\sigma = 0.7$, and where μ_{ref} is a reference value evaluated at the reference temperature T_{ref}. Once c_p and μ are calculated, the thermal conductivity $\lambda = Pr/(\mu c_p)$ is evaluated by assuming a constant value of the Prandtl number Pr.

3.2. Non-equilibrium dissociating air

The dissociating air used in the simulations below is assumed to be in thermodynamic equilibrium and is composed of N_2, O_2, NO, O, and N, which react according to the reversible chemical steps

$$O_2 + M \leftrightharpoons 2O + M, \tag{R1}$$

$$NO + M \leftrightharpoons N + O + M, \tag{R2}$$

$$N_2 + M \leftrightharpoons 2N + M, \tag{R3}$$

$$N_2 + O \leftrightharpoons NO + N, \tag{R4}$$

$$NO + O \leftrightharpoons O_2 + N, \tag{R5}$$

where M represents a collider (i.e., each of the species listed above). The rate constants of the steps (R1)–(R5) can be found in Ref. [4]. The equilibrium constants are evaluated using the traditional expression $K_{p,j} = \exp[-\sum_{i=1}^{N_s}(\nu''_{ij} - \nu'_{ij})\overline{g}_{i,\text{ref}}/(R^0T)]$, where $\overline{g}_{i,\text{ref}}$ are reference molar Gibbs free energies obtained from chemical databases [14].

The constant-pressure specific heat capacities are computed using the 9-coefficient NASA polynomials [14], whereas the dynamic viscosity of the mixture is evaluated using Wilke's rule [16]

$$\mu = \left(\sum_{i=1}^{N_s} Y_i\mu_i\right) \Bigg/ \left(\sum_{j=1}^{N_s} G_{ij}\frac{\mathcal{M}_i}{\mathcal{M}_j}Y_j\right). \tag{11}$$

In Eq. (11), the symbol G_{ij} is given by

$$G_{ij} = \frac{1}{\sqrt{8}}\left(1 + \frac{\mathcal{M}_i}{\mathcal{M}_j}\right)^{-\frac{1}{2}}\left[1 + \left(\frac{\mu_i}{\mu_j}\right)^{-\frac{1}{2}}\left(\frac{\mathcal{M}_j}{\mathcal{M}_i}\right)^{\frac{1}{4}}\right]^2, \tag{12}$$

where μ_i is the dynamic viscosity of species i [19],

$$\mu_i = \frac{5}{16}\frac{\sqrt{\pi k_B T \mathcal{M}_i/N_A}}{\pi\sigma_i^2\Omega_i^{(2,2)}}, \tag{13}$$

with k_B as the Boltzmann constant, N_A as the Avogadro number, and σ_i as the Lennard-Jones collision diameter of the species i.

The diffusivity of species i is computed as a function of the mixture composition and temperature as [17]

$$D_i = (1 - Y_i) \Bigg/ \sum_{j=1, j\neq i}^{N_s}\frac{X_j}{D_{ij}}, \tag{14}$$

where the binary diffusivity D_{ij} is given by [19]

$$D_{ij} = \frac{3}{16}\frac{\sqrt{2\pi N_A k_B^3 T^3/\mathcal{M}_{ij}}}{P\pi\sigma_{ij}^2\Omega_{ij}^{(1,1)}}. \tag{15}$$

In this expression, $\mathcal{M}_{ij} = \mathcal{M}_i\mathcal{M}_j/(\mathcal{M}_i + \mathcal{M}_j)$ is the reduced molecular weight and σ_{ij} is the reduced collision diameter. The symbols $\Omega_{ij}^{(1,1)}$ and $\Omega_i^{(2,2)}$ appearing in Eqs. (13) and (15) are collision integrals computed as polynomial functions of the reduced temperatures $T_i^* = Tk_B/\xi_i$ [for $\Omega_i^{(2,2)}$] and $T_{ij}^* = Tk_B/\xi_{ij}$ [for $\Omega_{ij}^{(1,1)}$], where ξ_i is the depth of the potential well of species i calculated using the Stockmayer potential [18]. The value of σ_{ij} and of ξ_{ij} depends on whether the colliding molecules are polar or nonpolar. If both molecules are either polar or non-polar, then $\sigma_{ij} = (\sigma_i + \sigma_j)/2$ and $\xi_{ij} = \sqrt{\xi_i\xi_j}$. If one of the colliding molecules is polar and the other is non-polar, then $\sigma_{ij} = (\sigma_i + \sigma_j)/(2\xi^{\frac{1}{6}})$ and $\xi_{ij} = \xi^2\sqrt{\xi_i\xi_j}$, where $\xi = 1 + (\alpha_n^*\mu_p^*/4)\sqrt{\xi_p/\xi_n}$, with α_n^* being the reduced polarizability of the non-polar molecule and μ_p^* being the reduced dipole moment of the polar molecule.

The thermal conductivity of the mixture λ is computed as [20]

$$\lambda = \frac{1}{2}\left[\sum_{i=1}^{N_s}X_i\lambda_i + \left(\sum_{i=1}^{N_s}\frac{X_i}{\lambda_i}\right)^{-1}\right], \tag{16}$$

where the thermal conductivity of species i is defined as

$$\lambda_i = \frac{15R^0\mu}{4\mathcal{M}_i} \tag{17}$$

for monoatomic gases. In polyatomic gases, λ_i is expressed as [21]

$$\lambda_i = \frac{\mu}{\mathcal{M}_i}\left(f_{i,\text{trans}}\overline{c}_{v,i,\text{trans}} + f_{i,\text{rot}}\overline{c}_{v,i,\text{rot}} + f_{i,\text{vib}}\overline{c}_{v,i,\text{vib}}\right), \tag{18}$$

where $\overline{c}_{v,i,\text{trans}} = 3R^0/2$, $\overline{c}_{v,i,\text{rot}} = R^0$, and $\overline{c}_{v,i,\text{vib}} = c_{p,i}\mathcal{M}_i - 7R^0/2$ are constant-volume molar heat capacities for translational, rotational, and vibrational degrees of freedom in linear molecules, while $\overline{c}_{v,i,\text{trans}} = \overline{c}_{v,i,\text{rot}} = 3R^0/2$ and $\overline{c}_{v,i,\text{vib}} = c_{p,i}\mathcal{M}_i - 4R^0$ are the corresponding values for non-linear molecules. Similarly,

$$f_{i,\text{trans}} = \frac{5}{2}\left(1 - \frac{2\overline{c}_{v,i,\text{rot}}A_i}{\pi\overline{c}_{v,i,\text{trans}}B_i}\right), \quad f_{i,\text{rot}} = \frac{\rho_i D_{ii}}{\mu_i}\left(1 - \frac{2A_i}{\pi B_i}\right), \quad f_{i,\text{vib}} = \frac{\rho_i D_{ii}}{\mu_i}, \tag{19}$$

are contributions from different molecular degrees of freedom to the thermal conductivity of species i, with $A_i = 5/2 - f_{i,\text{vib}}$ and $B_i = Z_{i,\text{rot}} + (2/\pi)[5\overline{c}_{vi,\text{rot}}/(3R^0) + f_{i,\text{vib}}]$. The rotational-relaxation collision number $Z_{i,\text{rot}}$ is computed as [15]

$$Z_{i,\text{rot}}(T) = Z_{i,\text{rot}}(T = 298\text{ K})f_i(T = 298\text{ K})/f_i(T), \tag{20}$$

where

$$f_i(T) = 1 + \frac{\pi^{3/2}}{2}\left(\frac{\xi_i}{k_BT}\right)^{1/2} + \left(\frac{\pi^2}{4} + 2\right)\left(\frac{\xi_i}{k_BT}\right) + \pi^{3/2}\left(\frac{\xi_i}{k_BT}\right)^{3/2}. \tag{21}$$

In Eq. (20), the parameter $Z_{i,\text{rot}}(T = 298\text{ K})$ is extracted for each species from the existing database [22].

4. Numerical methods employed in the HTR solver

A brief summary of the spatiotemporal discretization algorithms implemented in the HTR solver to numerically integrate the conservation equations is presented in this section. The reader is referred to the original sources in the relevant references cited below for further details on the formulation.

4.1. Spatial discretization

Since the flows of interest addressed in this study are at high Reynolds numbers, the diffusion fluxes (7) are evaluated using a second-order central finite-difference scheme written in conservative form, as in Ref. [26]. In this way, the complexity of the implementation is reduced, and the efficient parallelization properties of the solver are preserved.

The Euler fluxes (6) are computed on a collocated grid using a sixth-order finite difference method, whose reconstruction procedure can be summarized in the following five steps:

1. The right eigenvectors of the Jacobian matrices $\partial\mathbf{F}/\partial\mathbf{C}$, $\partial\mathbf{G}/\partial\mathbf{C}$, and $\partial\mathbf{H}/\partial\mathbf{C}$ are estimated at the cell interfaces by using the Roe average of the variables in the first neighboring cell centers. For instance, the matrix of right eigenvectors of $\partial\mathbf{F}/\partial\mathbf{C}$ can be written as

$$\overline{\overline{K}}_{\mathbf{F}} = \begin{bmatrix} Y_1 & 1 & \dots & 0 & 0 & 0 & Y_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ Y_N & 0 & \dots & 1 & 0 & 0 & Y_N \\ u-a & u & \dots & u & 0 & 0 & u+a \\ v & v & \dots & v & 1 & 0 & v \\ w & w & \dots & w & 0 & 1 & w \\ h_0 - ua & e_0 - \frac{\rho P_{\rho_1}}{P_e} & \dots & e_0 - \frac{\rho P_{\rho_N}}{P_e} & v & w & h_0 + ua \end{bmatrix}, \tag{22}$$

where a is the frozen speed of sound. In calorically non-perfect cases, P_{ρ_i} and P_e computed using the Roe-averaged conditions at cell interfaces are not compatible with the pressure jump conditions. The correction procedure proposed by [27] is applied here, which is based on projecting the point $\{P_{\rho_1}, \dots, P_{\rho_{N_s}}, P_e\}$, obtained using the Roe averages, onto the $N_s + 1$ dimensional hyperplane

$$\Delta P = \widehat{P}_e \Delta e + \sum_{i=1}^{N_s} \widehat{P}_{\rho_i} \Delta\rho_i, \tag{23}$$

where the hat symbol is utilized here to identify the corrected values of the derivatives.

2. The eigenvector matrices computed, for instance, at the cell interface $i + 1/2$, are used to project the conserved variables, along with the flux functions involved in the reconstruction stencil, onto the characteristic space. Continuing with the example of the fluxes in the x-direction, the variables in the characteristic space, which are denoted by tilde symbols, are defined as

$$\widetilde{\mathbf{C}} = \overline{\overline{K}}_{\mathbf{F}}^{-1}\mathbf{C}, \quad \widetilde{\mathbf{F}} = \overline{\overline{K}}_{\mathbf{F}}^{-1}\mathbf{F}. \tag{24}$$

3. The local Lax–Friedrichs flux-splitting method is then used to compute the positive and negative numerical flux functions $\widetilde{f}^{\pm} = \widetilde{\mathbf{F}} \pm \lambda\widetilde{\mathbf{C}}$, where λ is the vector of the maximum eigenvalues of the problem taken across the entire reconstruction stencil (i.e., $\lambda = [|u-a|, |u|, \dots, |u|, |u+a|]$ for the x-direction, where the multiplicity of the eigenvalue u is $N_s + 2$).
4. A sixth-order TENO reconstruction scheme [13] is employed to compute the value of the numerical flux functions at the location $i + 1/2$. The corresponding reconstruction stencil, for the $\widetilde{f}^{+}$ component of the fluxes, is sketched in Fig. 1 and consists of four low-order candidate stencils that span six points symmetrically distributed across the reconstruction location. In smooth regions, the coefficients of the low-order stencils, which are determined depending on the local stretching of the grid, are linearly combined in order to recover a sixth-order central reconstruction. If a low-order stencil crosses a discontinuity, it is excluded from the reconstruction stencil using the procedure described in Section 3.2 of Ref. [28]. The dynamic nonlinear procedure for control of dissipation ensures that sufficient numerical dissipation is generated for capturing shocks while minimum dissipation is produced for turbulence-like high-wavenumber fluctuations.
5. The local numerical flux at the cell interface,

$$\widetilde{f}_{i+\frac{1}{2}} = \left(\widetilde{f}^{+}_{i+\frac{1}{2}} + \widetilde{f}^{-}_{i+\frac{1}{2}}\right)/2, \tag{25}$$

is re-projected onto physical space by using $f_{i+\frac{1}{2}} = \overline{\overline{K}}_{\mathbf{F}}\widetilde{f}_{i+\frac{1}{2}}$.

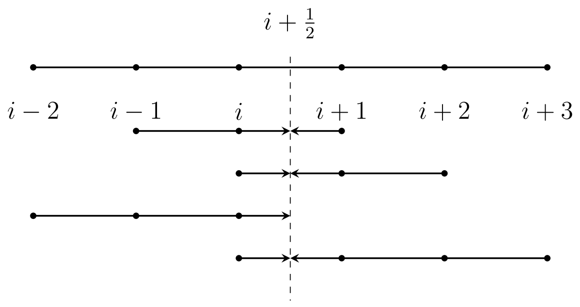


Fig. 1. Schematics of sixth-order TENO reconstruction stencils.

The reconstruction procedure described above is not strictly bound preserving [29–31]. Out-of-bounds behavior for critical primitive variables is not admissible when strict limits are required by physical constraints. The HTR solver incorporates a flux limiter similar to that proposed in Ref. [32] in order to ensure stability when excessive departures from established bounds are detected. In this method, a provisional state $\mathbf{C}^*$ is computed using the fluxes reconstructed with the procedure outlined above. If a mass fraction computed with $\mathbf{C}^*$ becomes negative, the fluxes of the corresponding partial density for the cell interfaces adjacent to where the bound is exceeded are recomputed using a first-order upwind scheme. Similarly, if the temperature computed with $\mathbf{C}^*$ is not larger than a lower bound T_{min} and smaller than an upper bound T_{max}, with $[T_{max}, T_{min}]$ corresponding to the maximum temperature range in which the NASA polynomials [14] are defined, the fluxes of all the transported variables are recomputed using the first-order upwind reconstruction. These fluxes are then considered for updating the conserved variables in the subsequent steps of the calculation. It is worth stressing that the flux limiter was not required in any of the verification tests cases analyzed in Section 6. A violation of admissible boundedness in temperature was however observed in the extra case provided in Section 7, where the flux limiter was necessary.

4.2. Time integration

Two different numerical schemes are used in the HTR solver to perform the time advancement of the conservation equations depending on the stiffness of the finite-rate chemistry considered in the calculation. When the characteristic time scales of the chemical reactions are comparable or larger than those of the flow field, the chemistry does not introduce any additional stiffness to the problem, and therefore time advancement is performed using the third-order strong-stability-preserving Runge–Kutta (SSP-RK3) method in Ref. [11]. On the contrary, in cases where the characteristic time scales of the chemical reactions are much shorter than those of the flow field, the chemistry leads to stiffness in the calculations and makes explicit time integration unfeasible. In these cases, the HTR solver adopts the operator-splitting approach in Ref. [12], which can be briefly summarized in the following two steps:

1. The algorithm begins by advancing in time the system of ordinary differential equations

$$\frac{d\mathbf{C}}{dt} = \dot{\mathbf{S}} - \mathbf{T}(\mathbf{C}^0) \tag{26}$$

from time t^0 to $t^0 + \Delta t$, with $\mathbf{C}^0$ representing the initial condition at time t^0. In Eq. (26), the symbol $\mathbf{T}(\mathbf{C})$ is defined

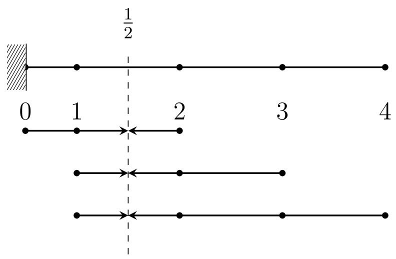


Fig. 2. Schematics of reconstruction scheme for staggered boundary conditions. The dashed line corresponds to the first off-wall reconstructed cell interface.

as

$$\mathbf{T}(\mathbf{C}) = \frac{\partial[\mathbf{F}(\mathbf{C})+\mathbf{F}_\nu(\mathbf{C})]}{\partial x} + \frac{\partial[\mathbf{G}(\mathbf{C})+\mathbf{G}_\nu(\mathbf{C})]}{\partial y} + \frac{\partial[\mathbf{H}(\mathbf{C})+\mathbf{H}_\nu(\mathbf{C})]}{\partial z}. \tag{27}$$

Since the vector of chemical sources $\dot{\mathbf{S}}$ depends only on the local value of the vector of conserved variables $\mathbf{C}$, the integration of Eq. (26) is performed independently at each cell of the computational grid by using an implicit algorithm. In particular, the present formulation uses a fourth-order Rosenbrock solver for this step [33].

2. A supplementary system of equations given by

$$\frac{d\mathbf{C}}{dt} = \mathbf{T}(\mathbf{C}) - \mathbf{T}(\mathbf{C}^0) \tag{28}$$

is integrated from time $t^0+\Delta t/2$ to $t^0+\Delta t$ using $\widehat{\mathbf{C}}$ as initial condition, where $\widehat{\mathbf{C}}$ is solution of Eq. (26) at time $t^0+\Delta t$. The system (28) does not involve chemical source terms and can be therefore integrated using a standard explicit integration scheme. In particular, the present formulation uses the SSP-RK3 method in this step. In this way, the solution of Eq. (28) at time $t^0+\Delta t$ converges to the solution of the original system of Eq. (1) with an accuracy of order $(\Delta t)^2$.

The procedure described above is steady-state preserving and reduces the computational cost of integrating the transport equations with stiff chemistry by allowing a larger time step Δt determined by the stability limits of the Runge–Kutta method employed to integrate Eq. (28).

4.3. Boundary conditions

Wall boundary conditions, including non-slip for velocity, non-catalytic for species, and isothermal or adiabatic for temperature, are enforced by using a staggered approach with the stencil depicted in Fig. 2. This method has the advantage of strictly enforcing the mass flux imposed by the boundary conditions at the boundary location [34]. This property, which is particularly important for the correct prediction of wall-bounded flows, circumvents the computation of fluxes on the boundary faces by the reconstruction algorithm, thus enforcing correct values directly from the boundary condition. For instance, the first node of the grid, denoted by the index 0 in Fig. 2, is staggered at the boundary face of the domain. The first cell interface, where the fluxes need to be evaluated using the reconstruction algorithm, is indicated by the index 1/2 in Fig. 2. There, the reconstruction-order accuracy is decreased to match a fifth-order upwind biased scheme, in which the coefficients of the low-order stencils are re-computed considering local grid spacing and the staggered node. The standard reconstruction scheme for the cell interface given by the index 3/2 is used by recomputing the low-order stencil coefficient such that it becomes compliant with the staggered node.

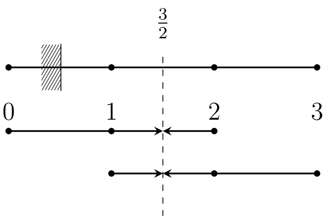


Fig. 3. Schematics of reconstruction scheme for collocated boundary conditions.

Non-reflecting boundary conditions are instead implemented using a collocated approach. A ghost cell is placed on the other side of the boundary face of the domain. Similarly to the staggered case, the order of accuracy of the reconstruction is decreased to enable sufficient points to support the stencil. In particular, the interface between the ghost cell and the first internal cell is reconstructed using first-order upwind discretization. This choice is dictated by the need of preserving shock-capturing capabilities at the boundaries of the domain and by the practical requirement of avoiding multiple rows of ghost cells. Fourth-order accuracy is instead obtained for the reconstruction of the first cell interface away from the boundary using the stencil depicted in Fig. 3.

5. Regent-based implementation of the HTR solver

An innovative aspect of the HTR solver is its implementation in the programming language Regent developed at Stanford University [10,35]. Regent employs the library Legion [7] to produce highly parallel applications addressed to distributed heterogeneous architectures. Specifically, Legion enables a different approach to parallelization compared to traditional Message-Passing-Interface (MPI) implementations. In Legion, the parallel application is organized in tasks, which are functions that sequentially operate on one or more arguments in a collection of data. While programs are represented as dependency graphs of tasks, Legion's runtime manages the placement of tasks in the machine along with the data allocation to ensure that the dependencies are enforced. The only responsibility of the application developer is to define which data fields will be read or modified by the corresponding task. In this way, Legion foresees the dependencies between the different tasks that populate the application, determines the concurrency of the tasks applied to the data, designs the parallel execution of the tasks, evaluates whether the optimization strategies demanded by the application developer are feasible, and performs all memory-management operations, including communications and memory allocations required to perform instructions commanded by the task on the selected machine architecture.

An example illustrating the ease of implementation of a task in Regent is provided in the snippet shown in Fig. 4. The instructions `reads` (line 5) and `writes` (line 6) inform the compiler about the fields of the region `Fluid` that have to be provided and will be modified by the task `UpdateUsingFluxX`. Similarly, the statement `__demand` (lines 1 and 8) is used to command the compiler to optimize the task at compilation time with the following options:

- Option `__parallel` [line 1] demands the compiler to use the algorithm described in Lee et al. [36]. It designs the way to run the task in parallel by assigning to each node one partition of the domain `Fluid` while preserving the correctness of the calculation. In the task presented in Fig. 4,

```
 1 __demand(__parallel, __cuda, __leaf)
 2 task UpdateUsingFluxX(Fluid : region(ispace(int3d), Fluid_columns),
 3                       Fluid_bounds : rect3d)
 4 where
 5    reads(Fluid.{cellWidth, FluxX}),
 6    reads writes atomic(Fluid.Conserved_t)
 7 do
 8    __demand(__openmp)
 9    for c in Fluid do
10       var xCellWidthInv = 1.0/Fluid[c].cellWidth[0]
11       var cm1 = (c+{-1, 0, 0})%Fluid_bounds
12       for i=0, nEq do
13          Fluid[c].Conserved_t[i] += (((Fluid[c  ].FluxX[i] -
14                                        Fluid[cm1].FluxX[i]))*xCellWidthInv)
15       end
16    end
17 end
```

Fig. 4. Snippet of HTR solver showing a single task written in Regent that computes the divergence of the fluxes in the x direction.

this optimization consists of providing the node with the additional points required to support the stencil for the divergence operator.

- Option `__cuda` [line 1] forces the compiler to produce a CUDA kernel for the task and run it on an available GPU.
- Option `__leaf` [line 1] informs the compiler that the task does not call any other subtasks, which allows for exhaustive optimization strategies.
- Option `__openmp` [line 8] commands the compiler to run the loop in lines 9–16 with an OpenMP approach using CPUs with shared memory access.

The unique features of the Regent programming language enable a number of advantages in the development and execution of the HTR solver as follows. Since the sequential semantics are preserved, the source HTR solver can be easily interpreted. Similarly, since the parallel management of the tasks is undertaken by Legion, bugs in parallelization-related developments in the HTR solver are minimized. Additionally, improvements on parallelization strategies can be effortlessly implemented in the HTR solver with new releases of the Legion library. Since the compiler and the runtime component of the Legion library are capable of performing automatic adaptations of the algorithm to the particular computational architecture where the application is compiled, the portability of the HTR solver is enhanced and does not represent an issue for deployment in heterogeneous architectures. Lastly, since both the CPU procedure and the CUDA kernel are generated from the single source code, traditional challenges associated with maintaining different versions of the same code (i.e. one version for CPU, and another version for GPU) are eliminated.

6. Verification of the HTR solver

The HTR solver is verified in this section with a set of benchmark cases involving low-speed two-dimensional laminar flows, one-dimensional inviscid compressible flows, supersonic turbulent channel flows, and hypersonic transitional boundary layers. In analyzing the results, the discussion focuses on the accuracy of the predictions of the code rather than on detailed physical aspects of the configurations, which are available in the corresponding references cited below for each benchmark case.

6.1. Inviscid vortex advection

This benchmark case aims at testing the numerical accuracy of the spatial discretization by considering the inviscid advection of a 2D vortex across a periodic domain. The initial condition is constructed by perturbing a uniform velocity field $\mathbf{u}_0 = [1.0, 1.0, 0.0]^T$ with a vortex centered in a square computational domain of size 10×10 in arbitrary units. The velocity perturbation expressed as a function of radial distance from the center of the vortex $\mathbf{r}$ is

$$\delta\mathbf{u} = \mathbf{r} \times \left[0, 0, -\frac{\beta}{2\pi} e^{(1-\mathbf{r}^2)/2}\right]^T, \tag{29}$$

where $\beta = 5$ is the intensity of the perturbation. Similarly, the temperature field is initialized with a uniform background temperature $T_0 = 1$. The temperature perturbation is given by

$$\delta T = -\frac{\gamma - 1}{\gamma} \frac{\beta^2}{8\pi^2} e^{1-\mathbf{r}^2}, \tag{30}$$

with $\gamma = 1.4$. The pressure field is computed as $P = T^{1/(\gamma-1)}$ to generate a constant entropy field. The flow is advanced for 0.2 units of time with a time-step corresponding to a CFL condition of 10^{-2}. These parameters, which are consistent with similar tests employed in early work [37], have been chosen in order to limit the contamination of the numerical error by the third-order time-advancement scheme.

The instantaneous L_2 norms of the errors with respect to the reference solution are shown in the left panel in Fig. 5 as a function of the number of grid points N_p. The solution is characterized by two different convergence orders for the range of grid points analyzed in the present work. The convergence rate for all errors is third-order for $N_p < 64$ and sixth-order for $N_p > 64$. This variation in the convergence rate can be understood by noticing that the size of the vortex is comparable to the size of the grid elements when the grid becomes coarse. As a result, the nonlinear adaptation between the different candidate stencils is triggered by the vortex, thereby decreasing the order of accuracy of the numerical approach. Upon refining the grid, its elements become much smaller than the size of the vortex, which exploits the full capabilities of the reconstruction scheme in performing with the linear sixth-order central stencil.

6.2. Viscous Taylor–Green vortex

The accuracy of the discretization of the viscous fluxes is addressed using the benchmark test case of the two-dimensional Taylor–Green vortex. The flow field is initialized in a computational domain of size $2\pi \times 2\pi$ with a constant density ρ_0, a velocity field

$$\mathbf{u} = U_0 \left[\sin x \cos y, -\cos x \sin y, 0\right]^T, \tag{31}$$

and a pressure field

$$P = P_0 - \frac{\rho_0 U_0^2}{4} \left(\cos 2x + \cos 2y\right). \tag{32}$$

The reference values of U_0, P_0, ρ_0 and dynamic viscosity μ_{ref} are chosen such that they lead to a Reynolds number $Re_0 =$

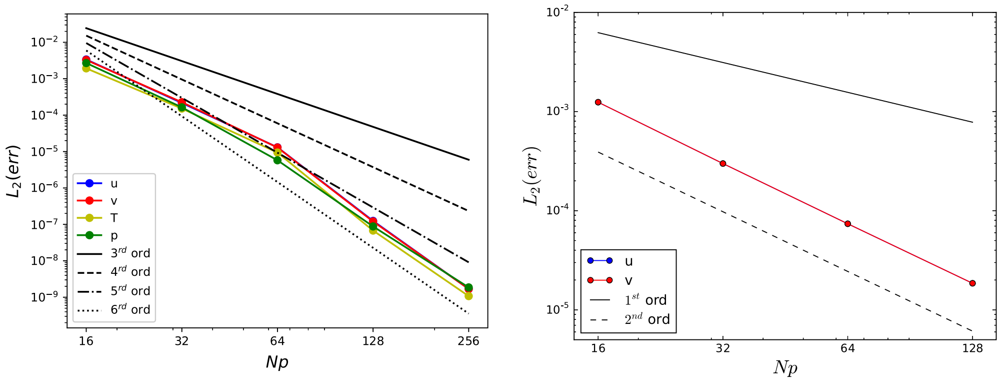


Fig. 5. Error convergence of 2D inviscid vortex advection (left) and of 2D viscous Taylor–Green vortex (right) as a function of the number of grid elements. Colored lines with symbols indicate errors with respect to the refenrence values of the velocity components (u and v), temperature (T), and pressure (P), whereas the dark patterned lines indicate different rates of convergence.

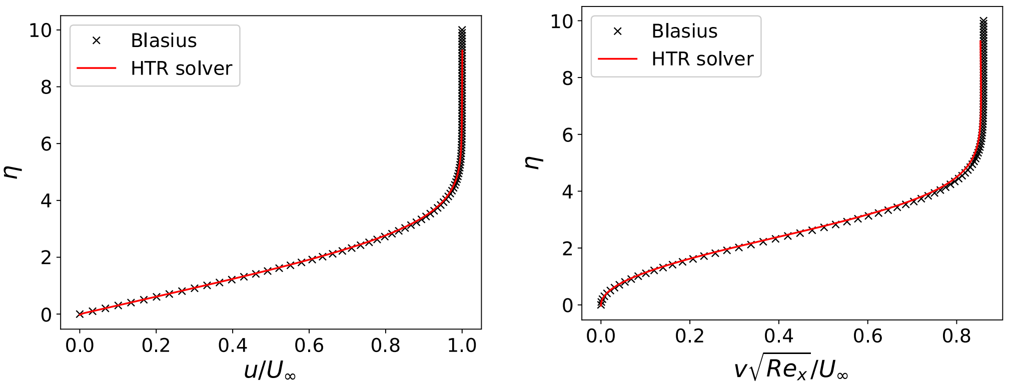


Fig. 6. Streamwise (left) and wall-normal (right) velocity profiles in a Mach-0.03 laminar boundary layer as a function of the self-similar coordinate η obtained from the self-similar theory (symbols) and numerical integrations using HTR solver (lines).

$\rho_0 U_0 \pi/\mu_{ref} = \pi$ and a Mach number $Ma_0 = U_0/\sqrt{\gamma P_0/\rho_0} = 10^{-2}/\sqrt{\gamma}$, with $\gamma = 1.4$. The equations are time-advanced for 0.5 units of time with a time-step corresponding to a CFL condition of 10^{-2}. The right panel in Fig. 5 shows the instantaneous L_2-norms of the errors with respect to the reference solution as a function of the number of grid points N_p. As expected, the rate of convergence to the exact solution is second-order as prescribed by the numerical discretization of the viscous fluxes.

6.3. Laminar boundary layers

The correctness of the implementation of the viscous fluxes and of the boundary conditions is tested below by using the reference solutions for low- and high-speed laminar boundary layers of calorically perfect gases over a flat plate under zero pressure gradient.

The computational domain is rectangular in shape in both incompressible and compressible cases. The boundary conditions are (a) non-slip isothermal wall at temperature T_w equal to the free-stream temperature T_∞; (b) outflow non-reflecting at the top and outlet boundaries [38,39]; and (c) inflow non-reflecting at the inlet $x = x_0$ in order to impose there a self-similar solution computed from the transformed conservation equations [40]. The local Reynolds number at the inflow, $Re_{x=x_0} = \rho_\infty U_\infty x_0/\mu_\infty$, is $Re_{x=x_0} = 500$ for the low-speed case, and $Re_{x=x_0} = 10^5$ for the high-speed case. In this formulation, ρ_∞, U_∞, and μ_∞ are, respectively, the free-stream values of the density, velocity, and dynamic viscosity. The Mach numbers based on the free-stream values of the flow velocity and speed of sound are $Ma_\infty = 0.03$ and $Ma_\infty = 6$ for the low- and high-speed cases, respectively.

Comparisons between the self-similar solution and the numerical solution obtained with the HTR solver are provided in Figs. 6 and 7 at streamwise locations corresponding to $Re_x = 1460$ (low-speed case) and $Re_x = 4.1{\cdot}10^5$ (high-speed case). Good agreement with the self-similar solution is observed in both cases. Exceptions are the slight deviations of order 5% in the wall-normal velocity occurring for $\eta > 4$ in the high-speed case because of the finite wall-normal pressure gradient, which is neglected in the self-similar approximation employed to construct the inflow boundary condition, but is retained in the numerical integration of the full conservation equations.

6.4. Inviscid one-dimensional compressible flows in shock tubes

The shock-capturing capabilities of the HTR solver are tested below in inviscid one-dimensional compressible flows in shock tubes. Three benchmark cases are considered, namely Lax's, Sod's, Shu–Osher's, and Grossmann–Cinnella's configurations.

6.4.1. Sod's shock tube

This benchmark case consists of a shock tube of length 1 in arbitrary units, which is filled with a calorically perfect gas with $R^0/\mathcal{M} = 1$ and $\gamma = 1.4$ [41]. Initially, the gas at $x < 0.5$ has a pressure $P = 1$ and temperature $T = 1$, whereas the gas at $x \geq 0.5$ has a pressure $P = 0.1$ and a temperature $T = 0.8$, both gases being initially at rest. A comparison between reference and numerical solutions obtained at time 0.2 s is provided in Fig. 8. The numerical solution is obtained by using a grid with 100 evenly spaced elements. Good agreement is observed between both solutions except in the prediction of the contact line, where the density and internal-energy discontinuities are slightly broadened by the flux splitting method.

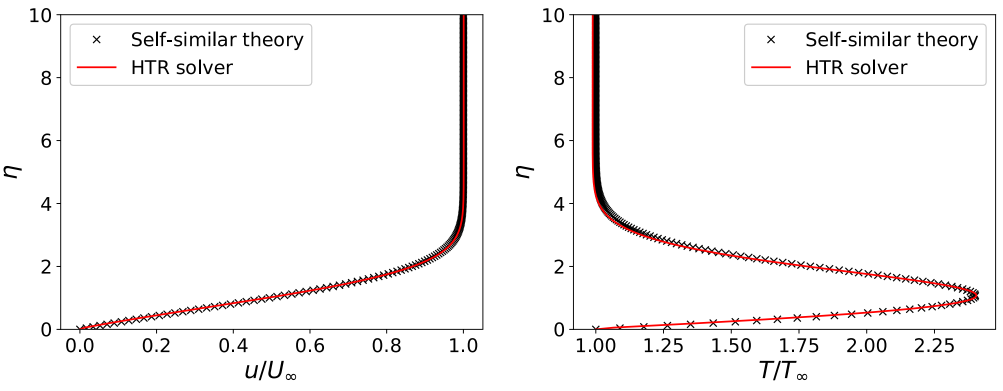


Fig. 7. Streamwise velocity (left) and temperature (right) velocity profiles in a Mach-6 laminar boundary layer as a function of the self-similar coordinate η obtained from the self-similar theory (symbols) and numerical integrations using HTR solver (lines).

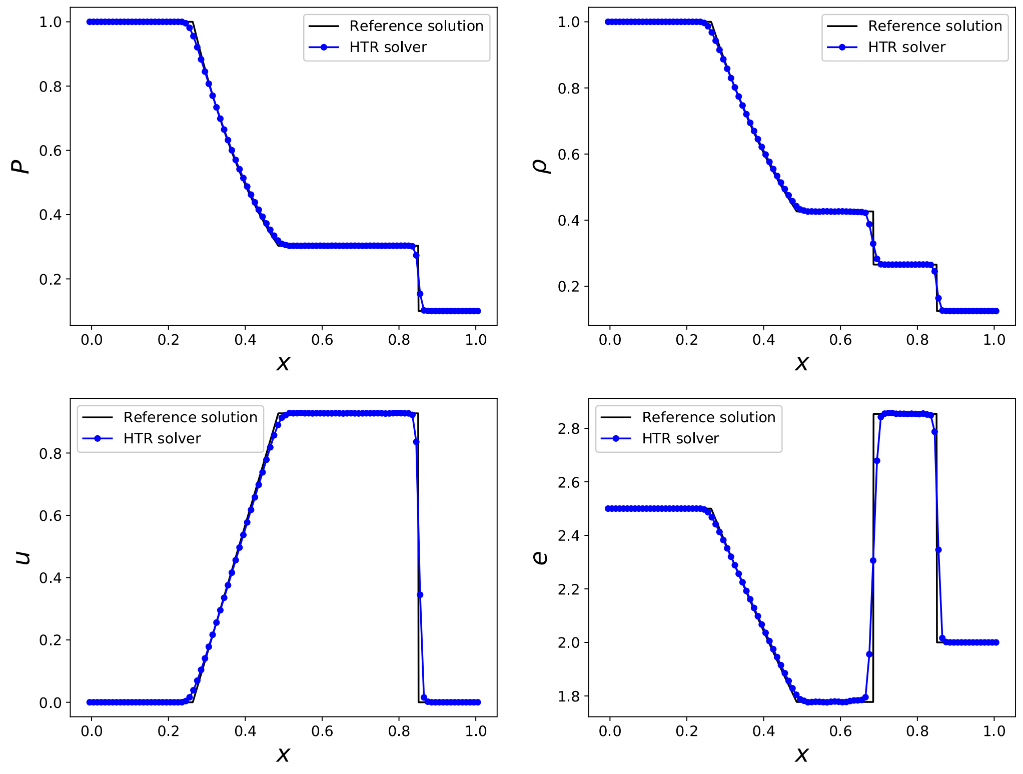


Fig. 8. Normalized values of the pressure, density, velocity, and internal energy resulting from theoretical solutions (dark solid lines) and numerical integrations using HTR solver (symbols) for Sod's shock tube [41].

6.4.2. Lax's shock tube

In Lax's configuration [42], a shock tube of length 1 in arbitrary units is filled with a calorically perfect gas with the same properties as in the Sod's benchmark case. Initially, for $x < 0.5$, the velocity, pressure, and temperature are, respectively, 0.698, 3.528, and 7.928. For $x \geq 0.5$, the corresponding values are 0, 0.571, and 1.142. Fig. 9 shows comparisons between the reference and numerical solutions. The latter is obtained on a grid using 100 evenly spaced elements. Good agreement is observed between both solutions except for (a) slight overshoots of the velocity and undershoots of the pressure, density, and internal energy at $x \approx 0.3$, and (b) small oscillations close to the contact discontinuity at $x \approx 0.7$. Note that similar errors have been observed in the literature that are related to the rather mild numerical diffusion introduced by the flux-splitting method used in the present formulation [13].

6.4.3. Shu–Osher's shock tube

Fig. 10 shows the results obtained for the benchmark case taken from [43], where a shock wave interacts with a sinusoidal density field in a calorically perfect gas. Initially, for $x < 1$ the velocity, pressure, and temperature are, respectively, 2.629, 10.333, and 2.679. For $x \geq 1$, the corresponding values are 0, 1, and $[1 + 0.2\sin(5x - 25)]^{-1}$. The profiles shown in Fig. 10 correspond to the time instant $t = 1.8$. The numerical solution is obtained on a grid with 200 evenly spaced cells. The reference solution is obtained using a fifth-order WENO-JS scheme on a grid composed of 2000 elements. Good agreement between the reference solution and the present work is observed except for small oscillations in the shock train, which are due to the low numerical diffusivity of the flux-splitting method used in the present formulation. Note however that the low dissipation property of the present formulation is beneficial for obtaining a

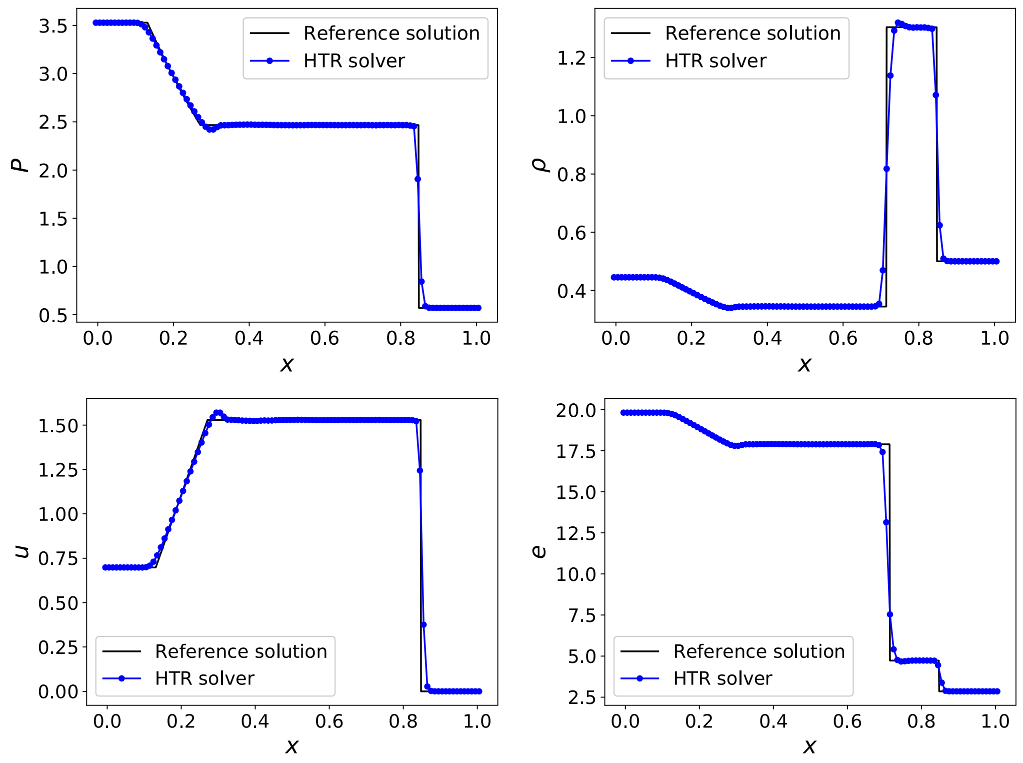


Fig. 9. Normalized values of the pressure, density, velocity, and internal energy in the reference solutions (dark solid lines) and numerical integrations using HTR solver (symbols) for Lax's shock tube [42].

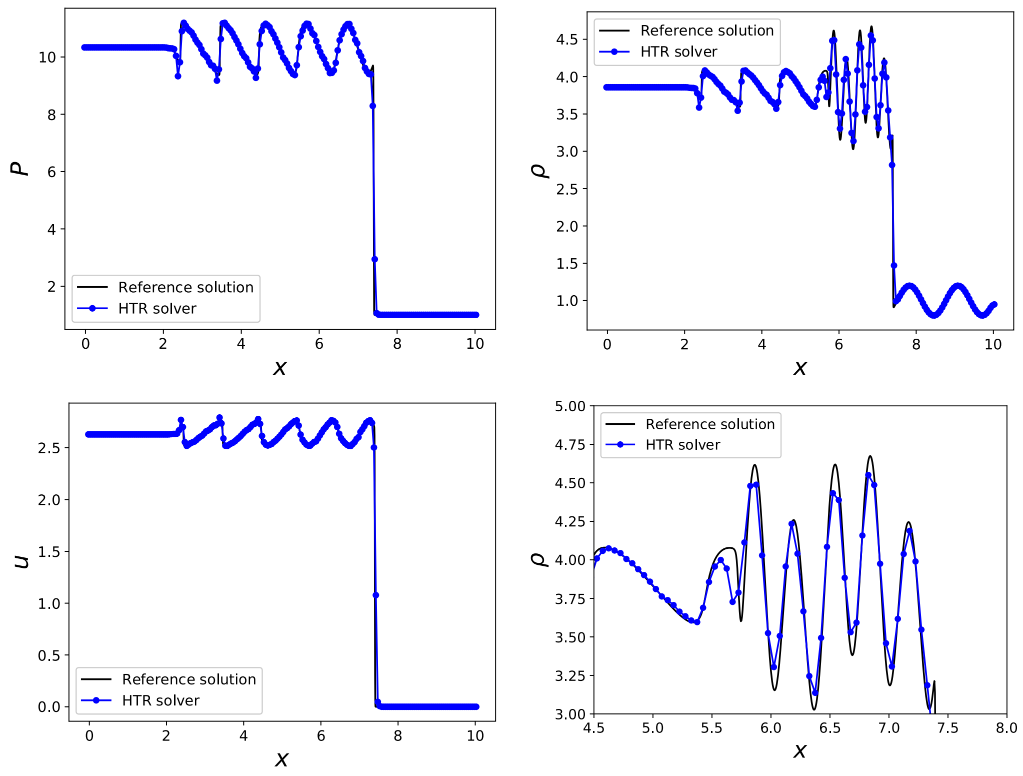


Fig. 10. Normalized values of the pressure, density, velocity, and internal energy in the reference solutions (dark solid lines) and numerical integrations using HTR solver (symbols) for Shu–Osher's shock tube [43].

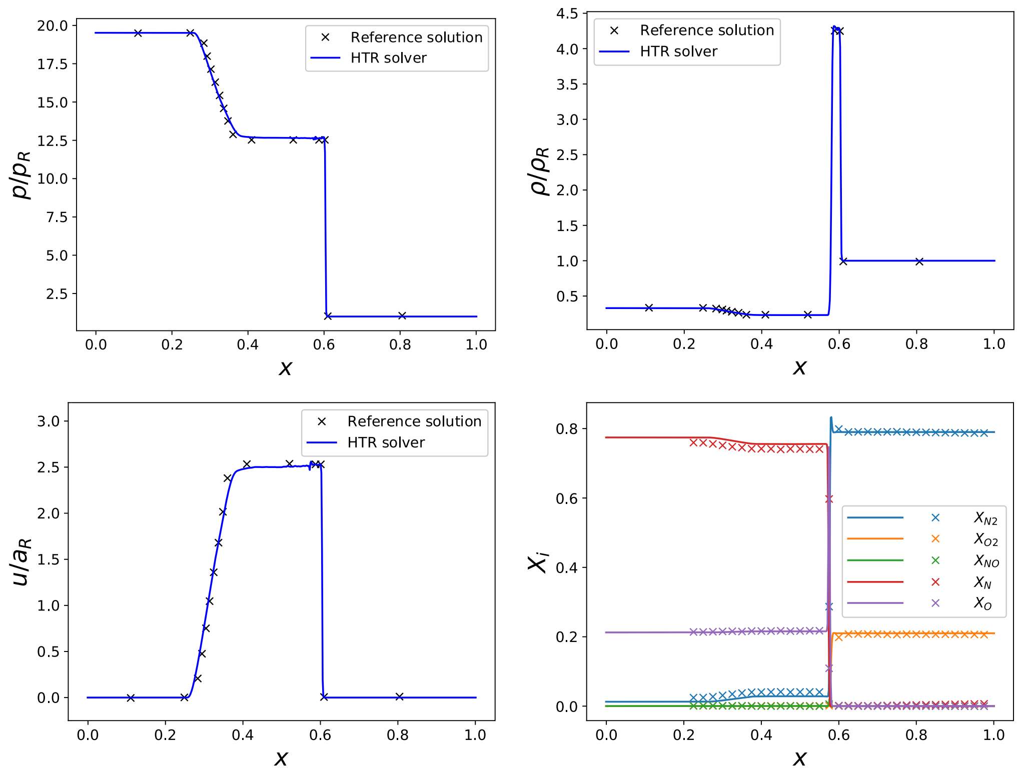


Fig. 11. Pressure, density, and velocity profiles normalized, respectively, with the pressure (P_R), density (ρ_R) and speed of sound (a_R) of the gas initially located at $x < 0.5$ m for Grossmann–Cinnella's shock tube [44], including reference solutions (symbols) and numerical integrations using the HTR solver (solid lines). The bottom right panel shows profiles of molar mass fractions of the species in the shock tube for the same case, including reference solutions (symbols) by Ref. [44] and numerical integrations using the HTR solver.

good solution for the high-frequency oscillations of the density at $5.7\,\text{m} \leq x \geq 7.4\,\text{m}$.

6.4.4. Grossman–Cinnella's shock tube

The Grossmann–Cinella benchmark case consists of a 1 m long shock-tube where the gas is subject to thermochemical effects [44]. Initially, for $x < 0.5$ m, the shock tube contains equilibrium dissociated air at pressure $P = 1.95$ bar and temperature $T = 9000$ K. For $x \geq 0.5$ m, the air in the shock tube is at pressure $P = 0.1$ bar and $T = 300$ K. The system then evolves until the shock wave reaches the position $x = 0.61$ m. In this particular case, the operator splitting method described in Section 4.2 has been used in order to perform the chemistry integration more efficiently. The results presented in Fig. 11 show that the present formulation is able to accurately reproduce the shock, the contact discontinuity, and the rarefaction wave. An additional panel is provided in Fig. 11 that shows the performance of HTR solver in correctly predicting the chemical composition of the gases in the shock tube.

6.5. Transitional and turbulent compressible flows

The capabilities of the HTR solver for predicting wall-bounded high-speed flows are tested in this section. The benchmark cases include supersonic turbulent channel flows and transitional hypersonic boundary layers, both for calorically perfect gases.

6.5.1. Supersonic turbulent channel flows

Bi-periodic channel flows have been a characteristic configuration in early work on compressible wall bounded turbulence at moderate Mach numbers [34,45–50]. The two homogeneous directions greatly facilitate the analysis and interpretation of results. In this study, the benchmark cases in Coleman et al. [45] and Sciacovelli et al. [50] are considered. They consist of supersonic turbulent flows of calorically perfect gases in bi-periodic channels driven by an imposed pressure gradient. The latter is chosen such that it keeps the flow at a constant bulk Reynolds number $Re_b = \dot{m}_b'' h/\mu_w$, where $\dot{m}_b''$ is the mass flow rate through the channel per unit cross sectional area, h half of the channel height, and μ_w is the dynamic viscosity at the walls of the channel, which are kept at uniform temperature.

In both benchmark cases, the Prandtl number is $Pr = 0.7$ and the adiabatic coefficient is $\gamma = 1.4$. However, the two cases differ in two parameters: (a) the bulk Reynolds number Re_b (i.e., $Re_b = 3000$ for Coleman's case, and $Re_b = 7000$ for Sciacovelli's case); and (b) the bulk Mach number $Ma_b = U_b/a_w$ based on the bulk velocity $U_b = \dot{m}_b''/\rho_b$ and on the speed of sound at the wall a_w, with ρ_b the bulk mean density (i.e., $Ma_b = 1.5$ for Coleman's case, and $Ma_b = 3.0$ for Sciacovelli's case). In both cases, the calculation is initialized with constant pressure and temperature, along with an axial velocity profile that is a function of the fourth power of the wall-normal distance and guarantees the expected mass flux $\dot{m}_b''$. Counter-rotating vertices whose axis is aligned with the stream-wise direction are superposed on this initial condition. The flow evolves until the bulk temperature and pressure in the domain reach a statistically steady state. Time averages are then collected over a time interval of order $3000\nu_w/u_\tau^2$ based on the friction velocity u_τ and on the kinematic viscosity at the wall ν_w.

For Coleman's case, the dimensions of the computational domain are $4\pi h \times 2h \times 2\pi h$ in the streamwise, wall-normal, and spanwise directions, respectively. The number of elements is $256 \times 128 \times 128$. The grid cells are evenly distributed in the periodic directions, whereas their distribution in the wall-normal direction follows a hyperbolic-tangent stretching function in order to achieve a wall-normal size of the first cell $\Delta y^+ = 0.8$. In the

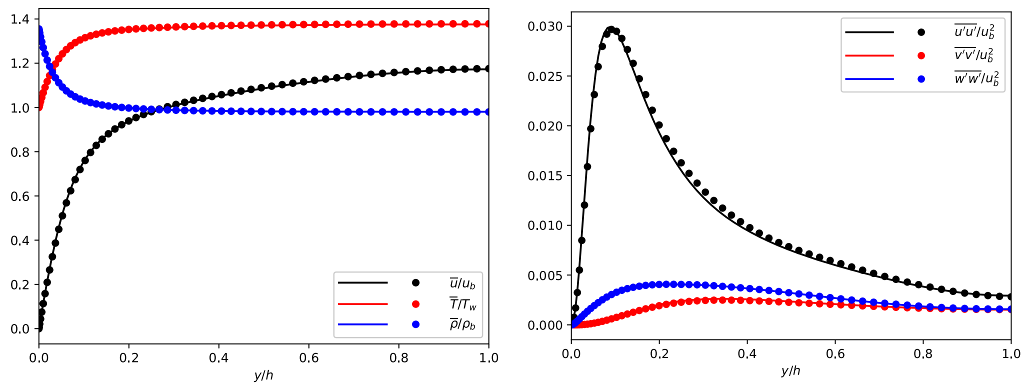


Fig. 12. Normalized Reynolds-averaged values of velocity, temperature, and density (left), along with the diagonal components of the Reynolds-stress tensor (right), including Coleman's reference solution (symbols) [45] and the numerical solution computed with HTR solver (solid lines).

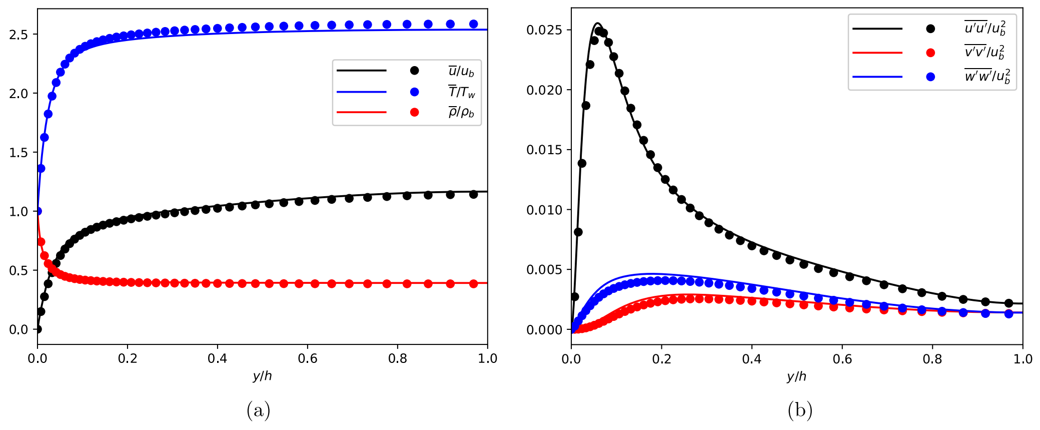


Fig. 13. Normalized Reynolds-averaged values of velocity, temperature, and density (left), along with the diagonal components of the Reynolds-stress tensor (right), including Sciacovelli's reference solution (symbols) [50] and the numerical solution computed with HTR solver (solid lines).

notation, the superscript + is used to express the normalization with the viscous unit ν_w/u_τ.

A larger grid relative to that used in Coleman's case is necessary in Sciacovelli's case because of the additional space required to warrant the decay of the two-point correlation functions, whose tails become longer as the Mach number increases. As a result, for Sciacovelli's case, the domain dimensions are $8\pi h \times 2h \times 2\pi h$, whereas the number of grid elements is $1024 \times 512 \times 762$. The elements in the wall-normal direction are distributed using a hyperbolic tangent stretching function while enforcing $\Delta y^+ = 0.8$ for the first cell.

Comparisons between the reference solutions and the present work are provided in Figs. 12–13, which show profiles of the normalized time-averaged values of the velocity, temperature, density, along with the diagonal components of the Reynolds stress tensor. Good agreement with the reference results is observed except for small discrepancies in the mean temperature and velocity in the channel core with respect to Sciacovelli's reference solution (e.g., see left panel in Fig. 13). This can be understood by noticing that the present simulations lead to a 5% larger friction Reynolds number than the value reported in Sciacovelli et al. [50], and consequently, to higher turbulent intensities near the wall, as shown by comparing the mean Reynolds stresses. As a result, the heat flux in the present calculation is 4.6% larger than that reported by [50], and therefore leads to comparatively low temperatures in the channel core. Note that the simulations in Sciacovelli et al. [50] are performed using an optimized fourth-order finite-difference scheme on an eleven-point stencil, along with an optimized selective sixth-order filter to avoid aliasing errors. That numerical method has dissipation and dispersion properties different from the ones emerging from the present formulation, thereby leading inevitably to differences between the two simulations.

6.5.2. Hypersonic transitional boundary layer of a calorically perfect gas

High-speed boundary layers over flat plates have been analyzed in early work over a wide interval of Mach numbers ranging from supersonic [51–54] to hypersonic [55–60]. In this study, the simulations in Franko and Lele [55], corresponding to a transitional hypersonic boundary layer of a calorically perfect gas over a flat plate under zero pressure gradient, are taken as a benchmark case. In this configuration, an undisturbed Mach-6 laminar boundary layer enters the domain and transitions to turbulence by means of blowing and suction through a thin strip on the wall. A three-dimensional snapshot of the DNS solution of this case computed with the HTR solver is provided in Fig. 14.

The computational domain is a cuboid of dimensions $1000\delta_0^* \times 75\delta_0^* \times 20\pi\delta_0^*$ in the streamwise (x), wall-normal (y), and spanwise (z) directions, respectively, where δ_0^* is the displacement thickness of the boundary layer at the inflow boundary. The origin of the streamwise coordinate $x = 0$ corresponds to the leading edge of the plate, which is not considered in these calculations. Instead, an inflow boundary condition located at $x = x_0$ is employed that consists of an undisturbed compressible laminar boundary layer whose velocity and temperature profiles are obtained by integrating the self-similar conservation equations [40]. The inflow Reynolds number is $Re_{\delta_0^*} = \rho_\infty U_\infty \delta_0^*/\mu_\infty = 3000$ based on the free-stream values of the velocity U_∞, density ρ_∞,

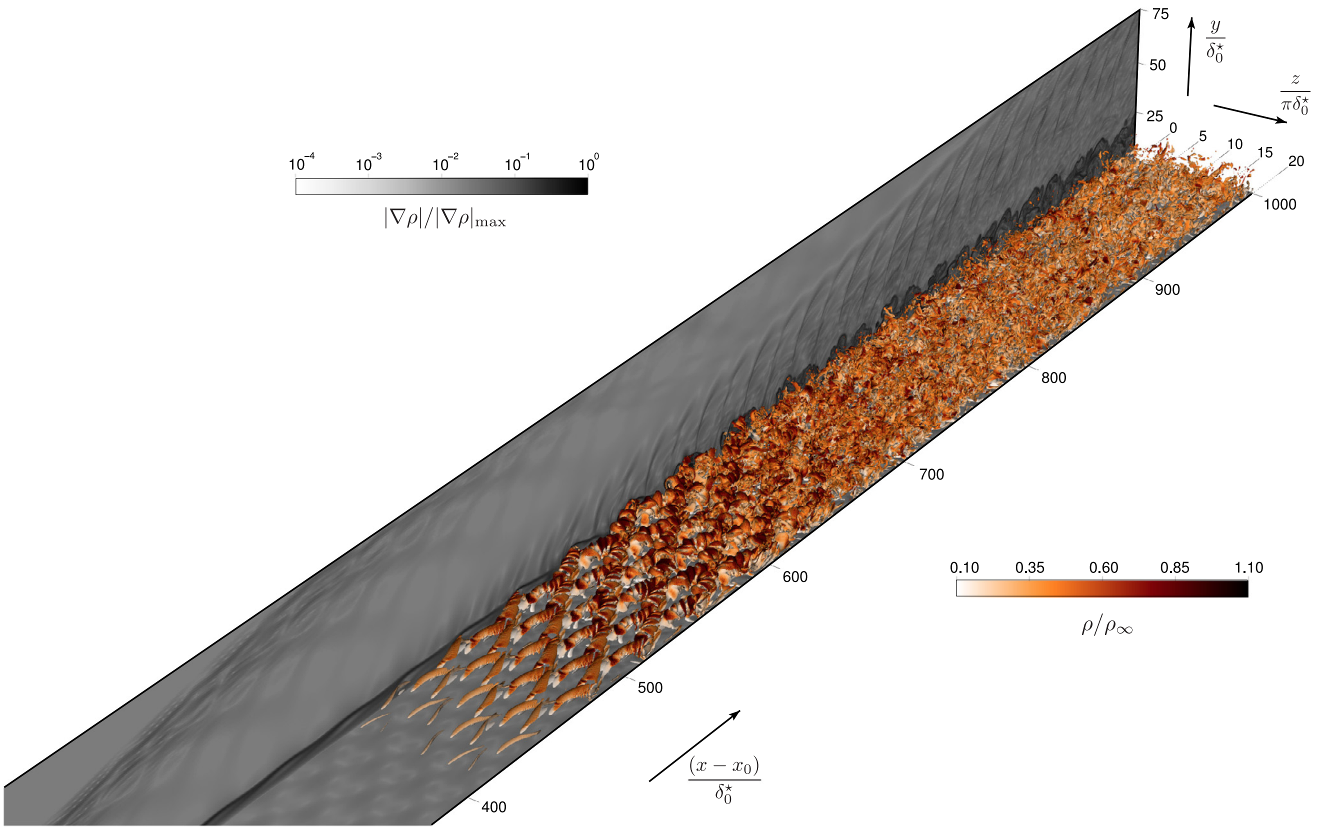


Fig. 14. Mach-6 hypersonic transitional boundary layer computed with the HTR solver. The figure shows solid contours of the normalized density gradient, along with the isosurfaces of the Q invariant of the velocity-gradient tensor colored by the normalized density.

and viscosity μ_∞. Periodic boundary conditions are used in the spanwise direction, whereas characteristic boundary conditions are used on the top and outflow boundaries. The wall is kept at a constant cold temperature $T_w = 6.5T_\infty$, or equivalently, at 79.2% of the stagnation temperature.

The non-slip boundary condition is imposed at the wall except for a thin surface strip that occupies the entire width of the plate and is located within the interval $15 \le (x - x_0)/\delta_0^* \le 20$, where wall-normal suction and blowing velocities are imposed in order to induce transition, as explained in Franko and Lele [55]. In particular, a wall-normal velocity $v_{wall} = f(x)g(z)\sum_i^2 A_i \sin(\omega_i t - \beta_i z)$ is imposed on the aforementioned strip, where the function $f(x) = \exp[-(x - x_s)^2/(2\sigma^2)]$ forces a Gaussian-like profile of the disturbance in the streamwise direction, with $x_s = 17.5\delta_0^*$ and $\sigma = 0.75\delta_0^*$. In addition, the function

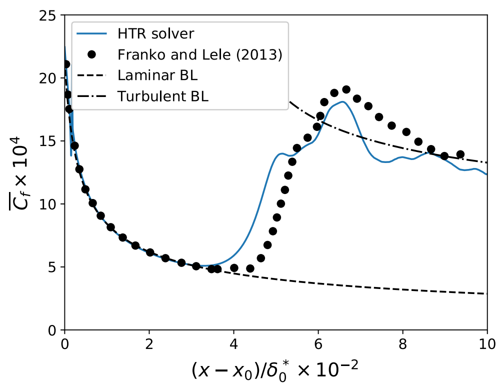
$$g(z) = 1.0 + 0.1\left\{e^{-[(z-z_c-z_w)/z_w]^2} + e^{-[(z-z_c+z_w)/z_w]^2}\right\}, \tag{33}$$

where $z_c = 10\pi\delta_0^*$ and $z_w = 2\pi\delta_0^*$, is utilized to break the symmetry in the boundary layer. The first oblique breakdown mode is induced by using two opposite modes parametrized by $A = [0.05U_\infty, 0.05U_\infty]^T$, $\omega = [0.9\delta_0^*/a_\infty, 0.9\delta_0^*/a_\infty]^T$, and $\beta = [0.3/\delta_0^*, -0.3/\delta_0^*]^T$, where a_∞ is the speed of sound in the free stream.

The computational domain is discretized using $4096 \times 250 \times 288$ cells in the streamwise, wall-normal, and spanwise directions, respectively. The grid is uniform in the streamwise and spanwise directions, whereas a hyperbolic-tangent stretching is used in the wall-normal direction in order to cluster cells near the wall. The stretching parameter of the distribution is determined by enforcing that the wall-normal size of the first grid element close to the wall, normalized in viscous units measured at the exit domain boundary, is $\Delta y^+ = 0.3$. The corresponding values in the streamwise and spanwise directions are $\Delta x^+ = 2.0$ and $\Delta z^+ = 1.8$, respectively.

Time- and spanwise-averaged distributions of the skin-friction coefficient $\overline{C_f}$ and Stanton number $\overline{St}$, along the normalized streamwise distance over the wall, are shown in Fig. 15. The averaging in $\overline{C_f}$ and $\overline{St}$ is made in a Reynolds sense for 6 periods of the forcing induced by the suction and blowing. Good qualitative agreement between the present work and the results in Franko and Lele [55] is observed, including the kink present along the ramp in $\overline{C_f}$ and the double peak present in $\overline{St}$ in the region $500 < (x - x_0)/\delta_0^* < 600$. However, quantitative differences are observed in the location of the ramp in both $\overline{C_f}$ and $\overline{St}$, and in the posterior evolution of these quantities in the turbulent zone. In particular, the HTR solver predicts 10% earlier transition and 10% smaller peak values of both $\overline{C_f}$ and $\overline{St}$. In addition, both $\overline{C_f}$ and $\overline{St}$ obtained with the HTR solver appear to comply more tightly with van Driest's turbulent correlation [61] at the exit domain boundary.

While the numerical method in Franko and Lele [55] is of comparable order but uses a high-wavenumber filter to control numerical instabilities, the present simulations utilize a numerical method with similar nominal accuracy but without any filter. In addition, for this benchmark case, the present simulations did not require the flux limiter introduced above in Section 4.1. As a result, the present simulations must necessarily employ a higher resolution across the boundary layer than that used in Franko and Lele [55] in order to avoid numerical instabilities. Specifically, in comparison with the grid used in Franko and Lele [55] the grid used in the present simulations has a larger number of elements by factors of $2 \times 5/4 \times 3/2$ in the streamwise, wall-normal, and spanwise directions, respectively. The finer grid resolution used in the present simulations, along with the comparatively



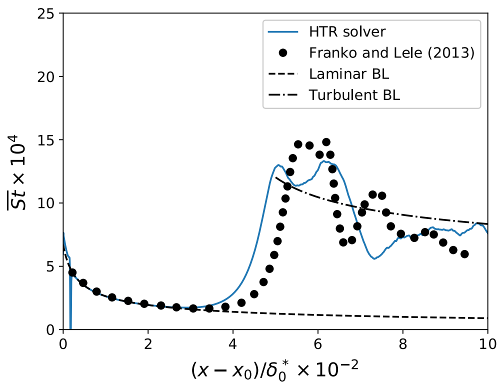


Fig. 15. Time- and spanwise-averaged skin-friction coefficient (left) and Stanton number (right) as function of the dimensionless streamwise coordinate, including Franko and Lele's reference solution (symbols) [55], the numerical solution computed with HTR solver (solid lines), and the laminar and turbulent correlations of van Driest [61].

lower numerical dissipation attained here by ruling out the utilization of filters for controlling numerical instabilities, may play an important role in the differences observed in Fig. 15. Note however that boundary-layer transition is a challenging case for code verification, in that it is known to be a strongly non-linear process sensitive to the computational setup, grid resolution, and numerical methods employed.

7. Simulation of a hypersonic transitional boundary layer with thermochemical effects using the HTR solver

The hypersonic boundary layers analyzed above in Sections 6.3 and 6.5.2 correspond to conditions where the stagnation temperature is assumed to be sufficiently small to render negligible thermochemical effects, and therefore the gas can be treated as calorically perfect using the model described in Section 3.1. Vibrational excitation and dissociation in hypersonic laminar boundary layers were incorporated in early work by Lees [62], and by Fay and Riddell [63]. More recently, numerical simulations have addressed these effects on temporally-developing turbulent boundary layers [59]. Counterpart studies in spatially-developing hypersonic boundary layers have been mostly focused on linear stability theory [64,65], parabolized stability equations [66, 67], and numerical simulations precluded to the initial stages of transition [68–71].

This section illustrates the capabilities of the HTR solver in simulating a spatially-developing Mach-6 hypersonic transitional boundary layer subject to the thermochemical effects of vibrational excitation and air dissociation over a non-catalytic wall, including late nonlinear stages and the onset of turbulence. No verification of the results is attempted by comparing with existing literature because of the limited range of early work, as explained above. The results in this section should therefore be interpreted as a glance at the potentiality of the HTR solver after having built confidence on the code based on the benchmark cases addressed in Section 6. Elaborated physical analyses of the results are the subject of future work.

The computational setup, including Δy^+, number of grid cells, domain size, blowing and suction parameters, and boundary conditions, are the same as those described in Section 6.5.2 with exception of the following aspects. The non-equilibrium dissociating-air model outlined in Section 3.2 is employed here in place of the calorically perfect gas model used in Section 6.5.2. Specifically, air in chemical equilibrium at temperature $T_\infty = 450\,\text{K}$ and pressure $P_\infty = 1$ atm flows over the flat plate. The chemical equilibrium in the free stream is largely displaced toward the reactants and leads to a negligible dissociation degree

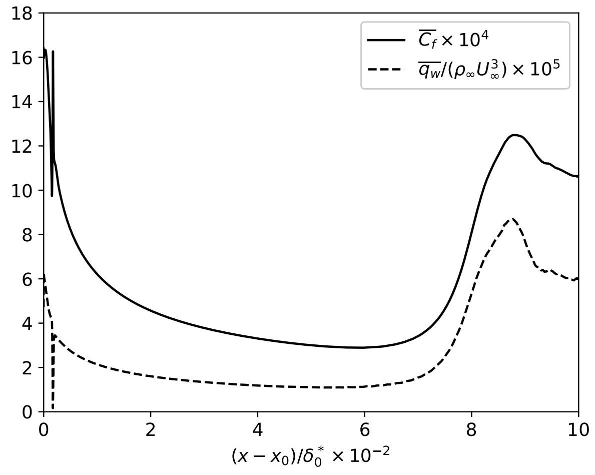
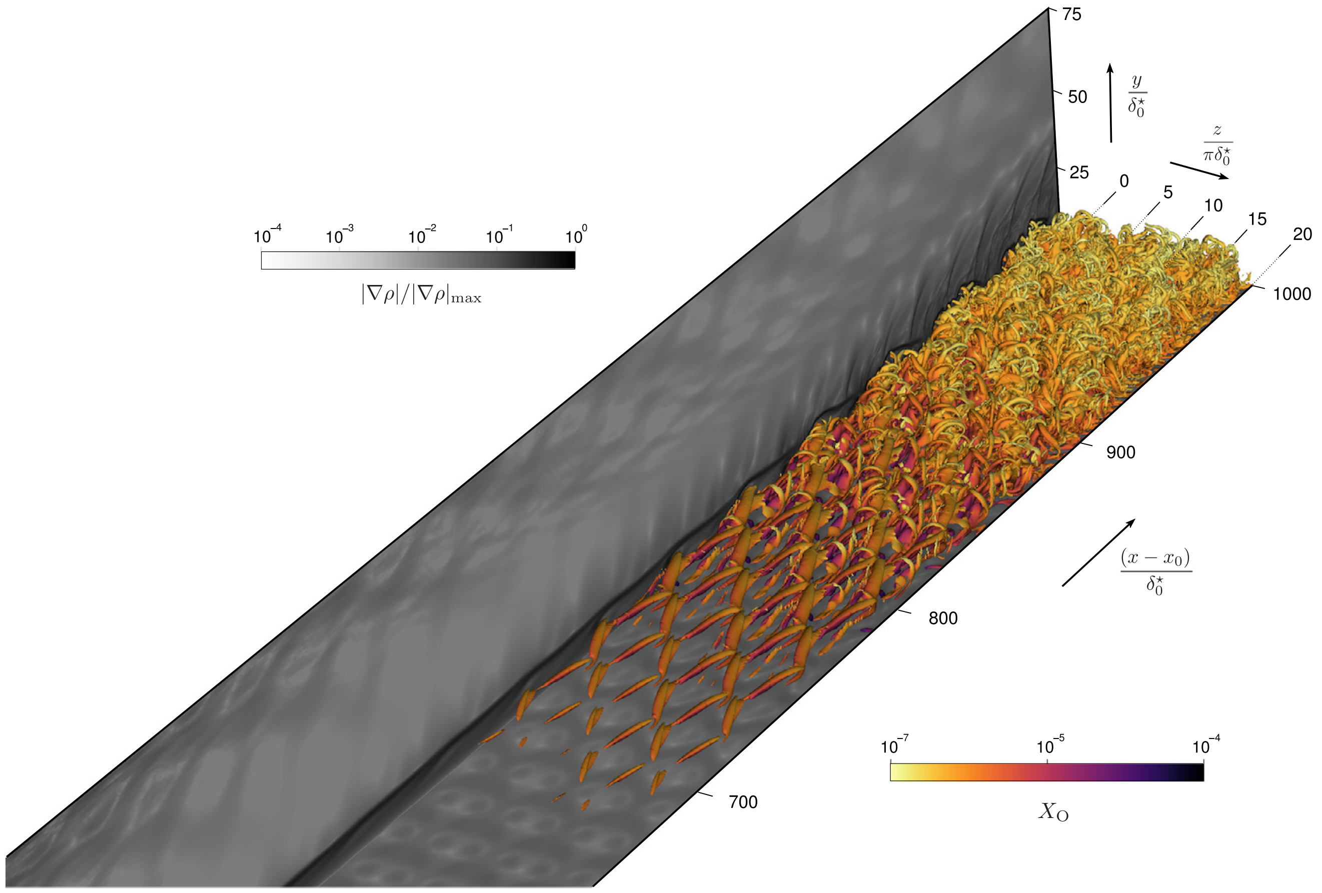


Fig. 16. Time and spanwise-averaged skin-friction coefficient and normalized heat-flux at the wall as function of the streamwise coordinate.

there. However, higher temperatures attained within the boundary layer, which are induced by both aerodynamic heating and the relatively high wall temperature $T_w = 6.5T_\infty$, increase the dissociation degree to approximately 0.01% based on the molar fraction of atomic oxygen X_O, whereas the remaining dissociation products are much smaller. Zero wall-normal gradients are imposed for all species mass fractions at the wall. The inflow boundary conditions are obtained by solving the laminar, locally self-similar boundary-layer equations including species transport and chemical reactions [62,63,72]. The higher value of $Re_{\delta_0^*}$ utilized here translates into minimum streamwise and spanwise sizes of the grid elements on the wall in viscous units measured at the exit domain boundary $\Delta x^+ = 2.7$ and $\Delta z^+ = 2.5$, which are comparatively larger than those in Section 6.5.2.

In addition, the inflow Reynolds number $Re_{\delta_0^*} = 4000$ is used in place of the lower value set in Section 6.5.2 in order to avoid an excessive delay of transition as a result of the energy drain caused mainly by vibrational excitation, with dissociation playing a secondary role in the dynamics.

The time- and spanwise-averaged distribution of the skin-friction coefficient $\overline{C_f}$ along the normalized streamwise distance over the wall is shown in Fig. 16. The qualitative aspect of the curve resembles that of the calorically perfect gas reported in Fig. 15, including a laminar decay, a ramp-up caused by transition, and a subsequent slower decay under impending turbulence. However, the magnitude of $\overline{C_f}$ is generally smaller here despite the higher Reynolds number employed.



75
50
25
$\frac{y}{\delta_0^\star}$
$\frac{z}{\pi\delta_0^\star}$
0
5
10
15
20
1000
900
800
700
$\frac{(x-x_0)}{\delta_0^\star}$
10^{-4} 10^{-3} 10^{-2} 10^{-1} 10^{0}
$|\nabla\rho|/|\nabla\rho|_{\max}$
10^{-7} 10^{-5} 10^{-4}
X_O

Fig. 17. Mach-6 hypersonic transitional boundary layer of dissociating air computed with the HTR solver. The figure shows solid contours of the normalized density gradient, along with the isosurfaces of the Q invariant of the velocity-gradient tensor colored by the molar fraction of atomic oxygen X_O.

It is worth mentioning that the standard definition of the Stanton number proves cumbersome in the presence of thermochemical effects since the recovery factor for the adiabatic wall enthalpy is unknown, and the wall enthalpy is not necessarily uniform as a result of variations of composition along the wall. Instead, the time- and spanwise-averaged wall heat flux $\overline{q_w}$ (with $\overline{q_w} > 0$ indicating heat entering the wall) is normalized here with the free-stream flux of kinetic energy $\rho_\infty U_\infty^3$. The corresponding distribution is shown in Fig. 16 and bears a resemblance to the curve of $\overline{C_f}$. Comparisons between these simulation results and the turbulent correlation of van Driest [61] are not attempted here, since the latter assumes calorically perfect behavior across the boundary layer.

Three-dimensional isosurfaces of the second invariant of the velocity-gradient tensor colored by X_O, along with the dimensionless modulus of the gradient of density, are provided in Fig. 17. Although the Reynolds number of this case is larger compared to the one employed in Section 6.5.2, the growth of the disturbances in the boundary layer is much slower. These findings are in qualitative agreement with linear stability analysis [64]. In particular, six streaky structures are generated early along the width of the plate that interact non-linearly downstream until a breakdown begins at $(x-x_0)/\delta_0^* \approx 800$. The flow loses symmetry thereafter while the skin-friction coefficient and the wall-heat transfer increase rapidly with distance downstream. Broadband distributions of all flow variables, including the concentration of atomic oxygen, are observed near the domain exit. Those, in conjunction with the slow decay of the wall shear stress and wall heat flux with distance downstream, are characteristic signatures of the onset of wall-bounded turbulence.

8. Parallel performance of the HTR solver in GPU and CPU environments

The parallel performance of the HTR solver is assessed here by weak scaling tests of the simulation case involving the supersonic turbulent channel flow described in Section 6.5.1. The weak scaling tests were performed on the supercomputing facilities Lassen and Quartz at the Lawrence Livermore National Laboratory (LLNL). In particular, each node of Lassen consists of two IBM POWER9 CPUs and four NVIDIA V100 GPUs, with approximately 4% of the computational power being provided by the CPUs while the remaining 96% rests on the GPUs. On the other hand, Quartz is a CPU-based machine where each node is equipped with two Intel Xeon 18-core E5-2695 v4 CPUs.

Exactly the same HTR solver with the same source files is compiled and run on both Lassen and Quartz. Each calculation on Quartz is setup such that each node is assigned with approximately 786×10^3 grid cells, whereas each node on Lassen is assigned with 19×10^6 grid cells, or equivalently, 4.7×10^6 grid cells per GPU. With these setups, the wall-clock time required to advance one time step in the numerical integration is 2.0 s in Quartz and 0.6 s in Lassen. The averaged throughput generated by the HTR solver is approximately 32×10^6 points per second per node on Lassen and 391×10^3 points per second per node on Quartz.

The efficiency of the HTR solver in the weak scaling tests is shown in Fig. 18. The code scales satisfactory up to 128 nodes (i.e., 512 GPUs) with an efficiency of 96.6% on Lassen. This case involved 2.4×10^9 grid cells with five degrees of freedom per cell and a total number of unknowns equal to 12×10^9. Generally

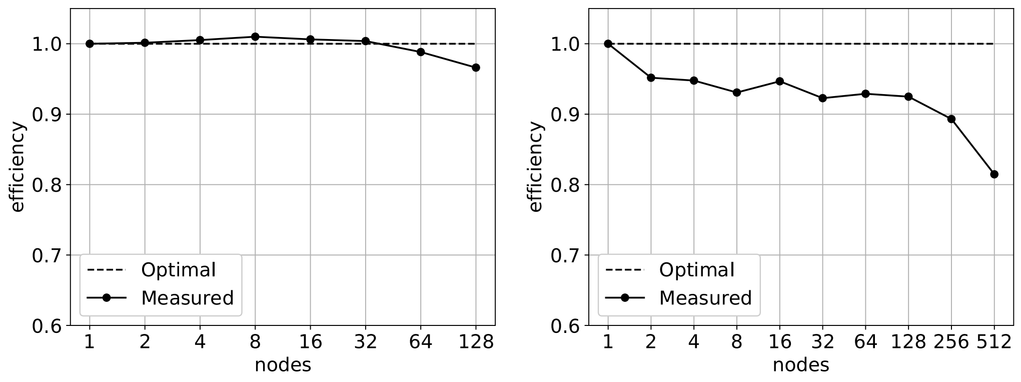


Fig. 18. Parallel efficiency of the solver for weak scaling tests on LLNL supercomputers Lassen (left) and Quartz (right). The configuration consists of the supersonic turbulent channel flow analyzed in Section 6.5.1.

lower efficiencies are observed on Quartz, although it should be stressed that these runs were performed without any tuning of the runtime in order to exploit more efficiently the computational resources of the machine.

Note that the scalability of the HTR solver reported here is likely to be superseded by improvements in subsequent releases of the Legion library without requiring any major modifications of the solver algorithm, since both are largely decoupled, as described in Section 5. For instance, it is known that the present releases of both the solver and Legion do not perform optimally in strong-scaling tests. Upcoming releases of the solver will lead to improved strong scalability by including the “tracing” feature of the Legion runtime, which optimizes the execution of the dependency analysis of the runtime for repeating tasks and is still under development [73].

9. Conclusions

An open-source DNS code for hypersonic aerothermodynamics, the Hypersonics Task-based Research (HTR) solver, is described in this work. The solver includes thermochemical effects (vibrational excitation and chemical dissociation), is written based on an innovative task-based parallelization technique for easier and more efficient deployment in heterogeneous supercomputing facilities, and operates with high-order numerical methods. The solver is highly portable between CPU- and GPU-based supercomputers and has been tested using a number of benchmark cases involving laminar boundary layers, one-dimensional sock-tubes, inviscid vortex advection, supersonic turbulent channel flows, and hypersonic transitional boundary layers.

Technical aspects worth developing in the code in future work involve: (a) reformulating the reconstruction procedure for curvilinear coordinates in order to enable the simulation of hypersonic flows over cones, spheres, wedges, or other non-planar geometries of interest for engineering applications; (b) the incorporation of thermodynamic non-equilibrium effects using either multi-temperature [4] or state-to-state models [74,75]; and (c) the coupling of the aerothermochemical flow field with wall-surface processes including conjugate heat transfer, catalysis, erosion, and ablation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This investigation is funded by the U.S. Air Force Office of Scientific Research (AFOSR), grant # 1194592-1-TAAHO, and by the Advanced Simulation and Computing (ASC) program of the U.S. Department of Energy’s National Nuclear Security Administration (NNSA) via the PSAAP-II Center at Stanford, grant # DE-NA0002373. The authors are grateful to Dr. Luca Sciacovelli for useful discussions, and to the Stanford Legion team (Dr. Michael Bauer, Dr. Sean Treichler, Dr. Wonchan Lee, and Mr. Manolis Papadakis) for their support with the implementation of the solver.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cpc.2020.107262. The source files of the computer program are available at http://dx.doi.org/10.17632/9zsxjtzfr7.1 and at https://github.com/stanfordhpccenter/HTR-solver.

References

[1] J.J. Bertin, R.M. Cummings, Annu. Rev. Fluid Mech. 38 (1) (2006) 129–157, http://dx.doi.org/10.1146/annurev.fluid.38.050304.092041.
[2] I.A. Leyva, Phys. Today 70 (11) (2017) 30–36, http://dx.doi.org/10.1063/PT.3.3762.
[3] J. Urzay, Annu. Rev. Fluid Mech. 50 (1) (2018) 593–627, http://dx.doi.org/10.1146/annurev-fluid-122316-045217.
[4] C. Park, Nonequilibrium Hypersonic Aerothermodynamics, Wiley, 1989.
[5] J.D.J. Anderson, Hypersonic and High-Temperature Gas Dynamics, Second Edition, second ed., American Institute of Aeronautics and Astronautics, 2006.
[6] G.V. Candler, Annu. Rev. Fluid Mech. 51 (1) (2019) 379–402, http://dx.doi.org/10.1146/annurev-fluid-010518-040258.
[7] M. Bauer, S. Treichler, E. Slaughter, A. Aiken, International Conference for High Performance Computing, Networking, Storage and Analysis, SC, IEEE, 2012, pp. 1–11.
[8] M. Bauer, S. Treichler, E. Slaughter, A. Aiken, SC ’14: International Conference for High Performance Computing, Networking, Storage and Analysis, Vol. 2015-Janua 2015-Janua, IEEE, New Orleans, 2014, pp. 845–856.
[9] S. Treichler, M. Bauer, A. Bhagatwala, G. Borghesi, R. Sankaran, H. Kolla, P.S. McCormick, E. Slaughter, W. Lee, A. Aiken, J. Chen, Exascale Scientific Applications, Chapman and Hall/CRC, 2017, pp. 257–278.
[10] E. Slaughter, W. Lee, S. Treichler, M. Bauer, A. Aiken, Regent: A high-productivity programming language for HPC with logical regions, in: SC ’15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2015, pp. 1–12, http://dx.doi.org/10.1145/2807591.2807629.
[11] S. Gottlieb, C.-W. Shu, E. Tadmor, SIAM Rev. 43 (1) (2001) 89–112.
[12] H. Wu, P.C. Ma, M. Ihme, Comput. Phys. Comm. 243 (2019) 81–96, http://dx.doi.org/10.1016/j.cpc.2019.04.016.
[13] L. Fu, X.Y. Hu, N.A. Adams, J. Comput. Phys. 305 (2016) 333–359, http://dx.doi.org/10.1016/j.jcp.2015.10.037.

[14] B.J. McBride, M.J. Zehe, S. Gordon, NASA Glenn Coefficients for Calculating Thermodynamic Properties of Individual Species, Technical Report NASA/TP-2002-211556, NASA, 2002.
[15] J.G. Parker, Phys. Fluids 2 (4) (1959) 449, http://dx.doi.org/10.1063/1.1724417.
[16] C.R. Wilke, J. Chem. Phys. 18 (4) (1950) 517–519, http://dx.doi.org/10.1063/1.1747673.
[17] R.B. Bird, W.E. Stewart, E.N. Lightfoot, Transport phenomena, John Wiley & Sons, Inc., New York, 1960, p. 780.
[18] L. Monchick, E.A. Mason, J. Chem. Phys. 35 (5) (1961) 1676–1697, http://dx.doi.org/10.1063/1.1732130.
[19] J.O. Hirschfelder, C.F. Curtiss, R.B. Bird, Molecular Theory of Gases and Liquids, John Wiley & Sons, 1964.
[20] S. Mathur, P. Tondon, S. Saxena, Mol. Phys. 12 (6) (1967) 569–579, http://dx.doi.org/10.1080/00268976700100731.
[21] N. Peters, J. Warnatz, Numerical Methods in Laminar Flame Propagation : A GAMM-Workshop, Vieweg, 1982, p. 202.
[22] R.J. Kee, G. Dixon-Lewis, J. Warnatz, M. Coltrin, J.A. Miller, Sandia Rep. SAND86-824 (December) (1986) 3–39.
[23] C.F. Curtiss, J.O. Hirschfelder, J. Chem. Phys. 17 (6) (1949) 550–555, http://dx.doi.org/10.1063/1.1747319.
[24] T.P. Coffee, J.M. Heimerl, Combust. Flame 43 (C) (1981) 273–289, http://dx.doi.org/10.1016/0010-2180(81)90027-4.
[25] A. Ern, V. Giovangigli, Multicomponent Transport Algorithms, Lecture Notes in Physics Monographs, vol. 24, Springer Berlin Heidelberg, Berlin, Heidelberg, 1994,
[26] G.A. Gerolymos, D. Sénéchal, I. Vallet, Internat. J. Numer. Methods Fluids 64 (2010) 769–810, http://dx.doi.org/10.1002/fld.2096.
[27] J.S. Shuen, M.S. Liou, B. van Leer, J. Comput. Phys. 90 (2) (1990) 371–395, http://dx.doi.org/10.1016/0021-9991(90)90172-W.
[28] L. Fu, X.Y. Hu, N.A. Adams, Commun. Comput. Phys. 26 (2019) 311–345, http://dx.doi.org/10.4208/cicp.OA-2018-0145.
[29] X. Zhang, C.W. Shu, J. Comput. Phys. 229 (9) (2010) 3091–3120, http://dx.doi.org/10.1016/j.jcp.2009.12.030.
[30] X. Zhang, C.-W. Shu, J. Comput. Phys. 231 (5) (2012) 2245–2258, http://dx.doi.org/10.1016/j.jcp.2011.11.020.
[31] L. Fu, Comput. Phys. Comm. 244 (2019) 117–131, http://dx.doi.org/10.1016/j.cpc.2019.06.013.
[32] M. Herrmann, G. Blanquart, V. Raman, AIAA J. 44 (12) (2006) 2879–2886, http://dx.doi.org/10.2514/1.18235.
[33] W.T. Vetterling, W.H. Press, S.A. Teukolsky, B.P. Flannery, Numerical Recipes Example Book (C++): The Art of Scientific Computing, Cambridge University Press, New York, 2002.
[34] D. Modesti, S. Pirozzoli, Int. J. Heat Fluid Flow 59 (2016) 33–49, http://dx.doi.org/10.1016/j.ijheatfluidflow.2016.01.007.
[35] Regent web page, 2019, URL http://regent-lang.org.
[36] W. Lee, E. Slaughter, M. Papadakis, A. Aiken, The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '19), ACM, Denver, 2019.
[37] R. Zhang, M. Zhang, C.W. Shu, Commun. Comput. Phys. 9 (3) (2011) 807–827, http://dx.doi.org/10.4208/cicp.291109.080410s.
[38] T.J. Poinsot, S.K. Lele, J. Comput. Phys. 101 (1) (1992) 104–129, http://dx.doi.org/10.1016/0021-9991(92)90046-2.
[39] N. Okong'o, J. Bellan, J. Comput. Phys. 176 (2) (2002) 330–344, http://dx.doi.org/10.1006/jcph.2002.6990.
[40] F.M. White, Viscous Fluid Flow, second ed., McGraw-Hill, Inc., 1992.
[41] G.A. Sod, A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws, 1978.
[42] P.D. Lax, Comm. Pure Appl. Math. 7 (1) (1954) 159–193, http://dx.doi.org/10.1002/cpa.3160070112.
[43] C.-W. Shu, S. Osher, J. Comput. Phys. 83 (1) (1989) 32–78, http://dx.doi.org/10.1016/0021-9991(89)90222-2.
[44] B. Grossman, P. Cinnella, J. Comput. Phys. 88 (1) (1990) 131–168, http://dx.doi.org/10.1016/0021-9991(90)90245-V.
[45] G.N. Coleman, J. Kim, R.D. Moser, J. Fluid Mech. 305 (1995) 159–183.
[46] P.G. Huang, G.N. Coleman, P. Bradshaw, J. Fluid Mech. 305 (1995) 185–218, http://dx.doi.org/10.1017/S0022112095004599.
[47] Y. Morinishi, S. Tamano, K. Nakabayashi, J. Fluid Mech. 502 (2004) 273–308, http://dx.doi.org/10.1017/S0022112003007705.
[48] A. Trettel, J. Larsson, Phys. Fluids 28 (2) (2016) 026102, http://dx.doi.org/10.1063/1.4942022.
[49] W. Li, Y. Fan, D. Modesti, C. Cheng, J. Fluid Mech. 875 (2019) 101–123, http://dx.doi.org/10.1017/jfm.2019.499.
[50] L. Sciacovelli, P. Cinnella, X. Gloerfelt, J. Fluid Mech. 821 (2017) 153–199, http://dx.doi.org/10.1017/jfm.2017.237.
[51] S.E. Guarini, R.D. Moser, K. Shariff, A. Wray, J. Fluid Mech. 414 (2000) 1–33, http://dx.doi.org/10.1088/0256-307x/23/6/045.
[52] T.B. Gatski, G. Erlebacher, Numerical Evolving Boundary Simulation Supersonic Layer of a Spatially Turbulent, Technical Report NASA/TM-2002-211934, NASA, 2002.
[53] S. Pirozzoli, F. Grasso, T.B. Gatski, Phys. Fluids 16 (3) (2004) 530–545, http://dx.doi.org/10.1063/1.1637604.
[54] S. Pirozzoli, F. Grasso, Phys. Fluids 18 (6) (2006) http://dx.doi.org/10.1063/1.2216989.
[55] K.J. Franko, S.K. Lele, J. Fluid Mech. 730 (2013) 491–532, http://dx.doi.org/10.1017/jfm.2013.350.
[56] C. Zhang, L. Duan, M.M. Choudhari, AIAA J. 56 (11) (2018) 4297–4311, http://dx.doi.org/10.2514/1.J057296.
[57] M.P. Martin, J. Fluid Mech. 570 (2007) 347–364, http://dx.doi.org/10.1017/S0022112006003107.
[58] L. Duan, I. Beekman, M.P. Martin, J. Fluid Mech. 655 (2010) 419–445, http://dx.doi.org/10.1017/S0022112010000959.
[59] L. Duan, M.P. Martin, J. Fluid Mech. 684 (2011) 25–59, http://dx.doi.org/10.1017/jfm.2011.252.
[60] L. Duan, I. Beekman, M.P. Martin, J. Fluid Mech. 672 (2011) 245–267, http://dx.doi.org/10.1017/S0022112010005902.
[61] E. van Driest, Aeronaut. Eng. Rev. 15 (1956) 26–41.
[62] L. Lees, J. Jet Propuls. 26 (4) (1956) 259–269, http://dx.doi.org/10.2514/8.6977.
[63] J.A. Fay, F.R. Riddell, J. Aerosp. Sci. 25 (2) (1958) 73–85, http://dx.doi.org/10.2514/8.7517.
[64] M.R. Malik, E.C. Anderson, Phys. Fluids A 3 (5) (1991) 803–821, http://dx.doi.org/10.1063/1.858012.
[65] K.J. Franko, R. MacCormack, S.K. Lele, 40th Fluid Dynamics Conference and Exhibit, 2010, p. 4601.
[66] C.-L. Chang, H. Vinh, M. Malik, 28th Fluid Dynamics Conference, 1997.
[67] H. Johnson, G.V. Candler, 35th AIAA Fluid Dynamics Conference and Exhibit, American Institute of Aeronautics and Astronautics, Toronto, 2005.
[68] C.P. Knisely, X. Zhong, AIAA SciTech Forum, 2019.
[69] O. Marxen, T.E. Magin, G. Iaccarino, E.S.G. Shaqfeh, Phys. Fluids 23 (8) (2011) http://dx.doi.org/10.1063/1.3614526.
[70] O. Marxen, T.E. Magin, E.S.G. Shaqfeh, G. Iaccarino, J. Comput. Phys. 255 (2013) 572–589, http://dx.doi.org/10.1016/j.jcp.2013.07.029.
[71] O. Marxen, G. Iaccarino, T.E. Magin, J. Fluid Mech. 755 (2014) 35–49, http://dx.doi.org/10.1017/jfm.2014.344.
[72] A. Liñán, I. Da Riva, Chemical nonequilibrium effects in hypersonic aerodynamics, Technical Report DTIC Report AD0294638, DTIC, 1962.
[73] W. Lee, E. Slaughter, M. Bauer, S. Treichler, T. Warszawski, M. Garland, A. Aiken, Dynamic tracing: Memoization of task graphs for dynamic task-based runtimes, in: Proceedings - International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2018, 2019, pp. 441–453, http://dx.doi.org/10.1109/SC.2018.00037.
[74] G. Colonna, M. Tuttafesta, M. Capitelli, D. Giordano, J. Thermophys. Heat Transfer 13 (3) (1999) 372–375, http://dx.doi.org/10.2514/2.6448.
[75] G. Colonna, F. Bonelli, G. Pascazio, Phys. Rev. Fluids 4 (3) (2019) 1–19, http://dx.doi.org/10.1103/PhysRevFluids.4.033404.