

Conservative and entropy stable solid wall boundary conditions for the compressible Navier–Stokes equations: Adiabatic wall and heat entropy transfer

Lisandro Dalcin^{a,1}, Diego Rojas^{a,2}, Stefano Zampini^{a,1}, David C. Del Rey Fernández^{b,c,3}, Mark H. Carpenter^{c,4}, Matteo Parsani^{a,5,*}

^a*King Abdullah University of Science and Technology (KAUST), Computer Electrical and Mathematical Science and Engineering Division (CEMSE), Extreme Computing Research Center (ECRC), Thuwal, Saudi Arabia*

^b*National Institute of Aerospace, Hampton, Virginia, United States*

^c*Computational AeroSciences Branch, NASA Langley Research Center, Hampton, Virginia, United States*

Abstract

We present a novel technique for the imposition of non-linear entropy conservative and entropy stable solid wall boundary conditions for the compressible Navier–Stokes equations in the presence of an adiabatic wall, or a wall with a prescribed heat entropy flow. The procedure relies on the formalism and mimetic properties of diagonal-norm, summation-by-parts and simultaneous-approximation-term operators, and is a generalization of previous works on discontinuous interface coupling [1] and solid wall boundary conditions [2].

Using the method of lines, a semi-discrete entropy estimate for the entire domain is obtained when the proposed numerical imposition of boundary conditions are coupled with an entropy-conservative or entropy-stable discrete interior operator. The resulting estimate mimics the global entropy estimate obtained at the continuous level. The boundary data at the wall are weakly imposed using a penalty flux approach and a simultaneous-approximation-term technique for both the conservative variables and the gradient of the entropy variables.

Discontinuous spectral collocation operators (mass lumped nodal discontinuous Galerkin operators), on high-order unstructured grids, are used for the purpose of demonstrating the robustness and efficacy of the new procedure for weakly enforcing boundary conditions. Numerical simulations confirm the non-linear stability of the proposed technique, with applications to three-dimensional subsonic and supersonic flows. The procedure described is compatible with any diagonal-norm summation-by-parts spatial operator, including finite element, finite difference, finite volume, discontinuous Galerkin, and flux reconstruction schemes.

Keywords: Compressible Navier–Stokes equations, Solid wall, Entropy conservation, Entropy stability, Summation-by-parts operators, Simultaneous-approximation-terms

1. Introduction

Next-generation numerical algorithms for use in large eddy simulations and direct numerical simulations of computational fluid dynamics will rely on efficient, high-order formulations, that are able to deliver better accuracy per degree of freedom than low-order methods, and that feature much smaller numerical errors both in terms of dispersion and dissipation [3, 4]. While these properties make high-order methods well suited for time-dependent simulations, these techniques are more prone to instability when compared to their lower-order counterparts. This is because numerical instabilities may occur if the flow contains discontinuities or under-resolved physical features. Various stabilization strategies (e.g. filtering [5], artificial viscosity, over-integration, and slope limiting [4] to cite a few) are commonly used to address these issues. However, such stabilization techniques possess several drawbacks since i) they reduce accuracy [4], ii) they usually require tuning parameters for each problem configuration, and iii) they do not guarantee that solvers designed to be high-order accurate in space will not crash.

A very promising and mathematically rigorous alternative consists in focusing on discrete operators that are non-linearly stable⁶ or, as in the case of the compressible Navier–Stokes equations, *entropy stable*. These operators simultaneously conserve mass, momentum, and total energy. In addition, they satisfy a discrete analogue to the conservation or dissipation of entropy which, with positivity assumptions on temperature and density, guarantees an L^2 bound on the conservative variables [6, 7]. We remark that the idea of enforcing entropy stability in numerical methods is old and commonly used for low-order operators, see e.g. [8, 9]. For extensions to high-order accurate operators see [10–14].

Until recently, fully discrete entropy stability was mostly established for implicit time stepping schemes. However, Ranocha and colleagues [15] developed and applied new explicit Runge–Kutta schemes (i.e., relaxation Runge–Kutta schemes) to entropy conservative or entropy dissipative semi-discretizations of any order for the compressible Euler and Navier–Stokes equations. The new time integration schemes can conserve or dissipate any solution properties with respect to any convex functional by the additional of a relaxation parameter that multiplies the Runge–Kutta update at each step. The general technique is not limited to the compressible Euler and Navier–Stokes equations setting but can be applied to many ordinary differential equations, and to both explicit and implicit Runge–Kutta methods.

However, issues remain on the path towards complete entropy stability for the compressible Navier–Stokes equations, e.g. shock capturing and bound-preserving limiter for high-order accurate discretizations. One major obstacle is the need for boundary conditions that preserve the entropy conservation or stability property of the interior operator. Practical experience indicates that numerical instabilities frequently originate at domain boundaries; the interaction of shocks with

*Corresponding author

Email addresses: `dalcinl@gmail.com` (Lisandro Dalcin), `diego.rojasblanco@kaust.edu.sa` (Diego Rojas), `stefano.zampini@kaust.edu.sa` (Stefano Zampini), `dcdelrey@gmail.com` (David C. Del Rey Fernández), `mark.h.carpenter@nasa.gov` (Mark H. Carpenter), `matteo.parsani@kaust.edu.sa` (Matteo Parsani)

¹Research Scientist

²Ph.D. student

³Postdoctoral Fellow

⁴Senior Research Scientist

⁵Assistant Professor

⁶We use the term rigorous because, as we will see in the next sections, these operators can mimic at the discrete level the results of the non-linear stability analysis at the continuous level.

these physical boundaries is particularly challenging for high-order formulations. An important step towards entropy stable wall boundary conditions for the compressible Euler and Navier–Stokes equations appears in [2, 16, 17]. More specifically, non-linearly stable wall boundary conditions for the compressible Navier–Stokes equations are presented in [2]. Therein, it is shown that entropy stability requires two conditions to be satisfied: i) Euler no-penetration, and ii) a prescribed value for the product of temperature and the gradient of the temperature in the normal direction to the wall. An additional term providing a controllable numerical dissipation has to be introduced to impose a zero relative velocity at the wall, i.e. the no-slip condition. Therefore, the solid wall boundary conditions proposed in [2] are entropy stable, but not entropy conservative. Note that in [17] it is shown that demanding a bound on velocity gradients necessitates the use of the full no-slip conditions, i.e. the thermal and the relative velocity boundary conditions.

In this work we present a general procedure for the development of point-wise entropy conservative boundary conditions representing either an adiabatic solid wall or a wall with a prescribed heat entropy flow for the compressible Navier–Stokes equations, discretized by using diagonal-norm, summation-by-parts (SBP) and simultaneous-approximation-term (SAT) operators (i.e. SBP-SAT operators). Entropy conservation is obtained by penalizing, using a SAT penalty, both the entropy variables and their gradients in the normal direction to the wall, as in the local discontinuous Galerkin approach [18]. The overall algorithm closely follows the treatment of the discontinuous interior interfaces coupling presented in [1]; a single implementation, with different inputs, can be used for interface penalization and imposition of boundary conditions. A controllable amount of dissipation can be added to make the boundary conditions entropy stable. The new procedure can be immediately applied to a moving wall, as will be shown in the theoretical and numerical results sections.

The manuscript is organized as follows. A brief review concerning the derivation of continuous entropy inequalities and the entropy analysis of the viscous wall boundary conditions for the compressible Navier–Stokes equations is provided in Section 2. The weak, point-wise, imposition of entropy conservative and entropy stable boundary conditions is carried out in Section 3 for an adiabatic solid wall and for a wall with a prescribed heat entropy transfer. Section 5 presents numerical results which confirm the accuracy and stability of the proposed boundary conditions. Conclusions are drawn in Section 6. Finally, in AppendixA a Python script is provided that symbolically verifies all proofs for curvilinear grids, while in AppendixB a simple and dimension-agnostic implementation of the entropy stable solid wall boundary condition coded in FORTRAN is presented.

2. A brief review of entropy stability theory

In this Section, we review the continuous entropy theory for the compressible Navier–Stokes equations and the solid wall boundary conditions by closely following [2, 19].

2.1. The compressible Navier–Stokes equations

To keep the presentation simple but without loss of generality, we consider the three-dimensional compressible Navier–Stokes equations in Cartesian coordinates (x_1, x_2, x_3) for an ideal gas in a

bounded domain Ω with boundary Γ

$$\begin{aligned} \frac{\partial \mathcal{Q}}{\partial t} + \sum_{m=1}^3 \frac{\partial \mathcal{F}_{x_m}^{(I)}}{\partial x_m} &= \sum_{m=1}^3 \frac{\partial \mathcal{F}_{x_m}^{(V)}}{\partial x_m}, \quad \forall (x_1, x_2, x_3) \in \Omega, \quad t \geq 0, \\ \mathcal{Q}(x_1, x_2, x_3, t) &= \mathcal{G}^{(B)}(x_1, x_2, x_3, t), \quad \forall (x_1, x_2, x_3) \in \Gamma, \quad t \geq 0, \\ \mathcal{Q}(x_1, x_2, x_3, 0) &= \mathcal{G}^{(0)}(x_1, x_2, x_3, 0), \quad \forall (x_1, x_2, x_3) \in \Omega. \end{aligned} \quad (1)$$

The vectors \mathcal{Q} , $\mathcal{F}_{x_m}^{(I)}$, and $\mathcal{F}_{x_m}^{(V)}$ respectively denote the conserved variables, the inviscid (I) fluxes, and the viscous (V) fluxes. The boundary data, $\mathcal{G}^{(B)}$, and the initial condition, $\mathcal{G}^{(0)}$, are assumed to be in $L^2(\Omega)$, with the further assumption that $\mathcal{G}^{(B)}$ will be set to coincide with linear well posed boundary conditions and such that entropy conservation/stability is achieved.

The vector of conserved variables is given as

$$\mathcal{Q} = [\rho, \rho \mathcal{U}_1, \rho \mathcal{U}_2, \rho \mathcal{U}_3, \rho \mathcal{E}]^\top, \quad (2)$$

where ρ denotes the density, $\mathcal{U} = [\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3]^\top$ is the velocity vector, and \mathcal{E} is the specific total energy. The inviscid fluxes are given by

$$\mathcal{F}_{x_m}^{(I)} = [\rho \mathcal{U}_m, \rho \mathcal{U}_m \mathcal{U}_1 + \delta_{m,1} \mathcal{P}, \rho \mathcal{U}_m \mathcal{U}_2 + \delta_{m,2} \mathcal{P}, \rho \mathcal{U}_m \mathcal{U}_3 + \delta_{m,3} \mathcal{P}, \rho \mathcal{U}_m \mathcal{H}]^\top, \quad (3)$$

where \mathcal{P} is the pressure, \mathcal{H} is the specific total enthalpy, and $\delta_{i,j}$ is the Kronecker delta. The viscous flux $\mathcal{F}_{x_m}^{(V)}$ is given as

$$\mathcal{F}_{x_m}^{(V)} = \left[0, \tau_{1,m}, \tau_{2,m}, \tau_{3,m}, \sum_{i=1}^3 \tau_{i,m} \mathcal{U}_i - \kappa \frac{\partial \mathcal{T}}{\partial x_m} \right]^\top, \quad (4)$$

where $\kappa = \kappa(T)$ is thermal conductivity, and the viscous stresses is given by

$$\tau_{i,j} = \mu \left(\frac{\partial \mathcal{U}_i}{\partial x_j} + \frac{\partial \mathcal{U}_j}{\partial x_i} - \delta_{i,j} \frac{2}{3} \sum_{n=1}^3 \frac{\partial \mathcal{U}_n}{\partial x_n} \right), \quad (5)$$

where $\mu = \mu(T)$ is the dynamic viscosity.

The required constitutive relations are

$$\mathcal{H} = c_p \mathcal{T} + \frac{1}{2} \mathcal{U}^\top \mathcal{U}, \quad \mathcal{P} = \rho R \mathcal{T}, \quad R = \frac{R_u}{M_w},$$

where c_p is the specific heat at constant pressure, \mathcal{T} is the temperature, R_u is the universal gas constant, and M_w is the molecular weight of the gas. Finally, the thermodynamic entropy is given as

$$s = \frac{R}{\gamma - 1} \log \left(\frac{\mathcal{T}}{\mathcal{T}_\infty} \right) - R \log \left(\frac{\rho}{\rho_\infty} \right), \quad \gamma = \frac{c_p}{c_p - R},$$

with \mathcal{T}_∞ and ρ_∞ the reference temperature and density, respectively.

It is well known that the compressible Navier–Stokes equations given in (1) possess a convex extension that, when integrated over the physical domain Ω , only depends on the boundary data. Such an extension yields the entropy function

$$\mathcal{S} = -\rho s, \quad (6)$$

which is a useful tool for proving stability in the L^2 norm [6, 7]. We can then define the entropy variables $\mathbf{W} = \partial\mathcal{S}/\partial\mathbf{Q}$, which for the compressible Navier–Stokes equations given in (1) are

$$\mathbf{W} = \left[\frac{\mathcal{H} - 1/2\mathbf{u}^\top\mathbf{u}}{\mathcal{T}} - s - \frac{\mathbf{u}^\top\mathbf{u}}{\mathcal{T}}, \frac{u_1}{\mathcal{T}}, \frac{u_2}{\mathcal{T}}, \frac{u_3}{\mathcal{T}}, -\frac{1}{\mathcal{T}} \right]^\top. \quad (7)$$

We remark that the convexity of \mathcal{S} guarantees the invertibility of the mapping between conservative and entropy variables, provided that the temperature \mathcal{T} and the density ρ are positive. In what follows, we always assume that such positivity is preserved.

The vector of entropy variables simultaneously contracts all of the inviscid spatial fluxes $\mathcal{F}_{x_m}^{(I)}$ as

$$\frac{\partial\mathcal{S}}{\partial\mathbf{Q}} \frac{\partial\mathcal{F}_{x_m}^{(I)}}{\partial x_m} = \frac{\partial\mathcal{S}}{\partial\mathbf{Q}} \frac{\partial\mathcal{F}_{x_m}^{(I)}}{\partial\mathbf{Q}} \frac{\partial\mathbf{Q}}{\partial x_m} = \frac{\partial\mathcal{F}_{x_m}}{\partial\mathbf{Q}} \frac{\partial\mathbf{Q}}{\partial x_m} = \frac{\partial\mathcal{F}_{x_m}}{\partial x_m}, \quad m = 1, 2, 3, \quad (8)$$

where the scalar $\mathcal{F}_{x_m}(\mathbf{Q})$ denotes the entropy flux in the m -th direction. By letting \mathbf{W} take the role as of a new set of independent variables, i.e. $\mathbf{Q} = \mathbf{Q}(\mathbf{W})$, the entropy variables (7) symmetrize the system (1) as [20]

$$\frac{\partial\mathbf{Q}}{\partial\mathbf{W}} \frac{\partial\mathbf{W}}{\partial t} + \sum_{m=1}^3 \frac{\partial\mathcal{F}_{x_m}^{(I)}}{\partial\mathbf{W}} \frac{\partial\mathbf{W}}{\partial x_m} = \sum_{m,j=1}^3 \frac{\partial}{\partial x_m} \left(\mathbf{C}_{m,j} \frac{\partial\mathbf{W}}{\partial x_j} \right), \quad (9)$$

where the viscous fluxes $\mathcal{F}_{x_m}^{(V)}$ have been recast in term of the entropy variables as

$$\mathcal{F}_{x_m}^{(V)} = \sum_{j=1}^3 \mathbf{C}_{m,j} \frac{\partial\mathbf{W}}{\partial x_j}. \quad (10)$$

For the definition of the symmetric and semi-definite $\mathbf{C}_{m,j}$ matrices see [2, 21].

Due to the symmetric nature of $\partial\mathbf{Q}/\partial\mathbf{W}$ and $\partial\mathcal{F}_{x_m}^{(I)}/\partial\mathbf{W}$, there exist scalar functions in entropy variables whose Jacobians represent the conservative variables \mathbf{Q} and the inviscid fluxes $\mathcal{F}_{x_m}^{(I)}$ as

$$\mathbf{Q}^\top = \frac{\partial\Phi}{\partial\mathbf{W}}, \quad \left(\mathcal{F}_{x_m}^{(I)} \right)^\top = \frac{\partial\Psi_{x_m}}{\partial\mathbf{W}}. \quad (11)$$

Φ is the *potential*, whereas the Ψ_{x_m} functions are the *potential fluxes* in the x_m direction, with (Φ, Ψ_{x_m}) the *potential-potential flux* pair [22]. A close relation between entropy and the potential-potential flux pair is summarized in the following Theorem [23], which is due to Godunov (see also [24]).

Theorem 2.1. *If a system of conservation laws can be symmetrized by introducing new variables \mathbf{W} , and \mathbf{Q} is a convex function of Φ , then an entropy function $\mathcal{S} = \mathcal{S}(\mathbf{Q})$ is given by*

$$\Phi = \mathbf{W}^\top \mathbf{Q} - \mathcal{S}, \quad (12)$$

and the entropy fluxes $\mathcal{F}_{x_m}(\mathbf{Q})$ satisfy

$$\Psi_{x_m} = \mathbf{W}^\top \mathcal{F}_{x_m}^{(I)} - \mathcal{F}_{x_m}. \quad (13)$$

By contracting the system of equations (1) with the entropy variables,

$$\frac{\partial \mathcal{S}}{\partial \mathcal{Q}} \frac{\partial \mathcal{Q}}{\partial t} + \sum_{m=1}^3 \frac{\partial \mathcal{S}}{\partial \mathcal{Q}} \frac{\partial \mathcal{F}_{x_m}^{(I)}}{\partial x_m} = \sum_{m=1}^3 \frac{\partial \mathcal{S}}{\partial \mathcal{Q}} \frac{\partial \mathcal{F}_{x_m}^{(V)}}{\partial x_m}, \quad (14)$$

and applying the relations given in (8), (9), and (10), we arrive at the differential form of the (scalar) entropy equation

$$\begin{aligned} \sum_{m=1}^3 \left[\frac{\partial \mathcal{S}}{\partial t} + \frac{\partial \mathcal{F}_{x_m}}{\partial x_m} \right] &= \sum_{m=1}^3 \left(\frac{\partial}{\partial x_m} \left(\mathcal{W}^\top \mathcal{F}_{x_m}^{(V)} \right) - \left(\frac{\partial \mathcal{W}}{\partial x_m} \right)^\top \mathcal{F}_{x_m}^{(V)} \right) \\ &= \sum_{m=1}^3 \left(\frac{\partial}{\partial x_m} \left(\mathcal{W}^\top \mathcal{F}_{x_m}^{(V)} \right) - \sum_{j=1}^3 \left(\frac{\partial \mathcal{W}}{\partial x_m} \right)^\top c_{m,j} \frac{\partial \mathcal{W}}{\partial x_j} \right). \end{aligned} \quad (15)$$

To obtain a global conservation statement for the entropy function \mathcal{S} , we then integrate equation (15) over the domain Ω

$$\begin{aligned} \frac{d}{dt} \int_{\Omega} \mathcal{S} d\Omega &\leq \sum_{m=1}^3 \int_{\Gamma} \left(\mathcal{W}^\top \mathcal{F}_{x_m}^{(V)} - \mathcal{F}_{x_m} \right) n_{x_m} d\Gamma - \sum_{m,j=1}^3 \int_{\Omega} \left(\frac{\partial \mathcal{W}}{\partial x_m} \right)^\top c_{m,j} \frac{\partial \mathcal{W}}{\partial x_j} d\Omega \\ &= \sum_{m=1}^3 \int_{\Gamma} \left(\mathcal{W}^\top \mathcal{F}_{x_m}^{(V)} - \mathcal{F}_{x_m} \right) n_{x_m} d\Gamma - DT, \end{aligned} \quad (16)$$

where n_{x_m} is the m -th component of the outward facing unit normal and

$$DT = \sum_{m,j=1}^3 \int_{\Omega} \left(\frac{\partial \mathcal{W}}{\partial x_m} \right)^\top c_{m,j} \frac{\partial \mathcal{W}}{\partial x_j} d\Omega.$$

We remark that viscous dissipation always introduces a negative rate of change in entropy, since the $-DT$ term in (16) is negative semi-definite. An increase in entropy within the domain can only result from data that convects or diffuses through the boundaries Γ . For smooth flows, we finally note that the inequality sign in (16) becomes an equality.

2.2. No-slip wall boundary conditions

For simplicity, we let the domain of interest be $\Omega = [0, 1]^3$ and we only consider entropy conservation (i.e., the equality relation in (16)). Thus, expanding the notation in equation (16)

yields

$$\begin{aligned}
& \frac{d}{dt} \int_{\Omega} \mathcal{S} dx_1 dx_2 dx_3 = -DT \\
& + \int_{x_1=0} \left[+\mathcal{F}_{x_1} - \boldsymbol{\mathcal{W}}^\top \left(C_{1,1} \frac{\partial \mathcal{W}}{\partial x_1} + C_{1,2} \frac{\partial \mathcal{W}}{\partial x_2} + C_{1,3} \frac{\partial \mathcal{W}}{\partial x_3} \right) \right] dx_2 dx_3 \\
& + \int_{x_1=1} \left[-\mathcal{F}_{x_1} + \boldsymbol{\mathcal{W}}^\top \left(C_{1,1} \frac{\partial \mathcal{W}}{\partial x_1} + C_{1,2} \frac{\partial \mathcal{W}}{\partial x_2} + C_{1,3} \frac{\partial \mathcal{W}}{\partial x_3} \right) \right] dx_2 dx_3 \\
& + \int_{x_2=0} \left[+\mathcal{F}_{x_2} - \boldsymbol{\mathcal{W}}^\top \left(C_{2,1} \frac{\partial \mathcal{W}}{\partial x_1} + C_{2,2} \frac{\partial \mathcal{W}}{\partial x_2} + C_{2,3} \frac{\partial \mathcal{W}}{\partial x_3} \right) \right] dx_1 dx_3 \\
& + \int_{x_2=1} \left[-\mathcal{F}_{x_2} + \boldsymbol{\mathcal{W}}^\top \left(C_{2,1} \frac{\partial \mathcal{W}}{\partial x_1} + C_{2,2} \frac{\partial \mathcal{W}}{\partial x_2} + C_{2,3} \frac{\partial \mathcal{W}}{\partial x_3} \right) \right] dx_1 dx_3 \\
& + \int_{x_3=0} \left[+\mathcal{F}_{x_3} - \boldsymbol{\mathcal{W}}^\top \left(C_{3,1} \frac{\partial \mathcal{W}}{\partial x_1} + C_{3,2} \frac{\partial \mathcal{W}}{\partial x_2} + C_{3,3} \frac{\partial \mathcal{W}}{\partial x_3} \right) \right] dx_1 dx_2 \\
& + \int_{x_3=1} \left[-\mathcal{F}_{x_3} + \boldsymbol{\mathcal{W}}^\top \left(C_{3,1} \frac{\partial \mathcal{W}}{\partial x_1} + C_{3,2} \frac{\partial \mathcal{W}}{\partial x_2} + C_{3,3} \frac{\partial \mathcal{W}}{\partial x_3} \right) \right] dx_1 dx_2.
\end{aligned} \tag{17}$$

Note that the plus and minus signs within the integrand terms of (17) account for the direction of the outward facing normals on the six faces of the unit cube Ω .

Furthermore, without loss of generality, we consider the case of a wall placed at $x_1 = 0$ such that the normal vector is $\mathbf{n} = (-1, 0, 0)^\top$, and we assume that all the other boundaries terms are entropy conservative, which allows us to neglect their contributions. Then, estimate (17) reduces to

$$\begin{aligned}
& \frac{d}{dt} \int_{\Omega} S dx_1 dx_2 dx_3 = -DT \\
& + \int_{x_1=0} \left[\mathcal{F}_{x_1} - \boldsymbol{\mathcal{W}}^\top \left(C_{1,1} \frac{\partial \mathcal{W}}{\partial x_1} + C_{1,2} \frac{\partial \mathcal{W}}{\partial x_2} + C_{1,3} \frac{\partial \mathcal{W}}{\partial x_3} \right) \right] dx_2 dx_3.
\end{aligned} \tag{18}$$

Within the context of linear analysis for a solid viscous wall, the wall behaves like a subsonic outflow [25], and four independent boundary conditions must be imposed to prove energy (linear) stability [26–28] (see also [2] and the references therein). The first three correspond to the no-slip boundary conditions $\mathcal{U}_1 = \mathcal{U}_2 = \mathcal{U}_3 = 0$ that impose a zero relative velocity with respect to the wall. The fourth condition can be either imposed on the gradient of the temperature normal to the wall $(\partial \mathcal{T} / \partial n)_{wall}$ (Neumann boundary condition, e.g. the adiabatic wall), or to the temperature at the wall \mathcal{T}_{wall} , (the Dirichlet or isothermal wall boundary condition), or a mixture of these two (the Robin boundary condition) [27, 28].

In the non-linear case, entropy conservation and entropy stability in the adiabatic solid wall case or a wall with a prescribed heat entropy flow are attained by means of the next two theorems. These theorems provide the conditions that result in a bound on the time rate of change of the entropy function in (18), and are point-wise valid [2]. The first theorem is a generalization of Theorem 3.1 presented in [2] to a moving wall.

Theorem 2.2. *The no-slip boundary conditions $\mathcal{U}_1 = 0$ and $\mathcal{U}_m = \mathcal{U}_m^{wall}$, $m = 2, 3$ bound the inviscid contribution to the time derivative of the entropy in equation (18).*

Proof. Equation (13) provides the following relation for the entropy flux

$$\mathcal{F}_{x_1} = \boldsymbol{\mathcal{W}}^\top \mathcal{F}_{x_1}^{(I)} - \Psi_{x_1} = -\rho s \mathcal{U}_1 R, \quad \Psi_{x_1} = -\rho \mathcal{U}_1 R. \tag{19}$$

Substituting the no-slip conditions, into the definition of the inviscid flux, $\mathcal{F}_{x_1}^{(I)}$, (equation (3)) and the condition $\mathcal{U}_1 = 0$ into the definition of Ψ_{x_1} , yields the desired result $\mathcal{F}_{x_1} = 0$. \square

Remark 2.1. *In a general setting, the no-slip boundary conditions read as $\mathbf{U} = \mathbf{U}^{wall}$ with $\mathbf{U}^{wall} \cdot \mathbf{n} = 0$, where \mathbf{U} , \mathbf{U}^{wall} and \mathbf{n} denote the velocity vector of the fluid, the velocity vector of the wall and the unit normal vector, respectively.*

Theorem 2.3. *The boundary condition*

$$\mathbf{g}(t) = \kappa \frac{\partial \mathcal{T}}{\partial n} \frac{1}{\mathcal{T}}, \quad (20)$$

where $\partial \mathcal{T} / \partial n$ denotes the normal derivative of \mathcal{T} , bounds the viscous contribution to the time derivative of the entropy (18).

Proof. See Theorem 3.2 in [2]. \square

Remark 2.2. *The scalar value,*

$$\kappa \left(\frac{\partial \mathcal{T}}{\partial x_1} \frac{1}{\mathcal{T}} \right) = \mathcal{W}^\top \left(c_{1,1} \frac{\partial \mathcal{W}}{\partial x_1} + c_{1,2} \frac{\partial \mathcal{W}}{\partial x_2} + c_{1,3} \frac{\partial \mathcal{W}}{\partial x_3} \right),$$

accounts for the change in entropy due to the wall heat flux at $x_1 = 0$ [2] and is often denoted as heat entropy transfer or heat entropy flow [29].

3. Entropy conservative and entropy stable solid wall boundary conditions for the semi-discrete system

To discretize in space, we partition the physical domain Ω into non-overlapping hexahedral elements and we semi-discretize the system (1) using a multi-dimensional SBP operator, constructed from a one-dimensional SBP operator by way of tensor products. The nodal distribution within each element is based on N^3 Legendre-Gauss-Lobatto (LGL) points [2, 12, 30, 31], where N is the number of LGL point in one direction.

Here, we summarize the relevant SBP operators used to discretize (1), and to derive the new

procedure to impose the solid wall boundary conditions.

$$\begin{aligned}
\mathbf{D}_{x_1} &= (\mathbf{D}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_5), \quad \cdots \quad \mathbf{D}_{x_3} = (\mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{D}_N \otimes \mathbf{I}_5), \\
\mathbf{Q}_{x_1} &= (\mathbf{Q}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_5), \quad \cdots \quad \mathbf{Q}_{x_3} = (\mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{Q}_N \otimes \mathbf{I}_5), \\
\mathbf{B}_{x_1} &= (\mathbf{B}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_5), \quad \cdots \quad \mathbf{B}_{x_3} = (\mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{B}_N \otimes \mathbf{I}_5), \\
\Delta_{x_1} &= (\Delta_N \otimes \mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_5), \quad \cdots \quad \Delta_{x_3} = (\mathbf{I}_N \otimes \mathbf{I}_N \otimes \Delta_N \otimes \mathbf{I}_5), \\
\mathbf{P}_{x_1} &= (\mathbf{P}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_5), \quad \cdots \quad \mathbf{P}_{x_3} = (\mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{P}_N \otimes \mathbf{I}_5), \\
\mathbf{P}_{x_1, x_2} &= (\mathbf{P}_N \otimes \mathbf{P}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_5), \quad \cdots \quad \mathbf{P}_{x_2, x_3} = (\mathbf{I}_N \otimes \mathbf{P}_N \otimes \mathbf{P}_N \otimes \mathbf{I}_5), \\
\mathbf{P} &= \mathbf{P}_{x_1, x_2, x_3} = (\mathbf{P}_N \otimes \mathbf{P}_N \otimes \mathbf{P}_N \otimes \mathbf{I}_5),
\end{aligned} \tag{21}$$

\mathbf{D}_N , \mathbf{Q}_N , \mathbf{B}_N , Δ_N and \mathbf{P}_N are the one-dimensional SBP operators, and \mathbf{I}_N is the identity matrix of dimension N . The matrices $\mathbf{P}_{(\cdot)}$ may be thought of as mass matrices in the context of the discontinuous Galerkin finite element method. Herein, the focus is exclusively on diagonal-norm SBP operators, based on fixed element-based polynomials of order p ($p = N - 1$). The matrices $\mathbf{D}_{(\cdot)}$ are used to approximate the first derivatives and are defined as $\mathbf{P}_{(\cdot)}^{-1} \mathbf{Q}_{(\cdot)}$. The nearly skew-symmetric matrices $\mathbf{Q}_{(\cdot)}$ are undivided differencing operators where all rows sum to zero, and the first and last column sum to -1 and 1 respectively. The matrices $\mathbf{B}_{(\cdot)}$ pick off the interface terms in the respective directions. For the spectral element discretization considered in this paper, the $\mathbf{B}_{(\cdot)}$ matrices take on a particularly simple form; as an example, consider \mathbf{B}_{x_1} , which is given as

$$\mathbf{B}_{x_1} = \mathbf{B}_{x_1}^+ - \mathbf{B}_{x_1}^-, \quad \mathbf{B}_{x_1}^- = \text{diag}(1, 0, \dots, 0) \otimes \mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_5, \quad \mathbf{B}_{x_1}^+ = \text{diag}(0, \dots, 0, 1) \otimes \mathbf{I}_N \otimes \mathbf{I}_N \otimes \mathbf{I}_5.$$

For a high-order accurate scheme on a tensor product cell, they pick off the values of whatever vector they act on (typically the solution or the flux) at the nodes of the two opposite faces multiplied by the orthogonal component of the unit normal.

SBP operators can be recast in telescoping flux form [21]. For example,

$$\mathbf{D}_{x_1} \mathbf{f}_{x_1}^{(I)} = \mathbf{P}_{x_1}^{-1} \mathbf{Q}_{x_1} \mathbf{f}_{x_1}^{(I)} = \mathbf{P}_{x_1}^{-1} \Delta_{x_1} \mathbf{l}_{StoF}^{x_1} \mathbf{f}_{x_1}^{(I)} = \mathbf{P}_{x_1}^{-1} \Delta_{x_1} \bar{\mathbf{f}}_{x_1}^{(I)},$$

where $\bar{\mathbf{f}}_{x_1}^{(I)} = \mathbf{l}_{StoF}^{x_1} \mathbf{f}_{x_1}^{(I)}$ and the one-dimensional telescoping operator, Δ_N , is defined as

$$\Delta_N = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}.$$

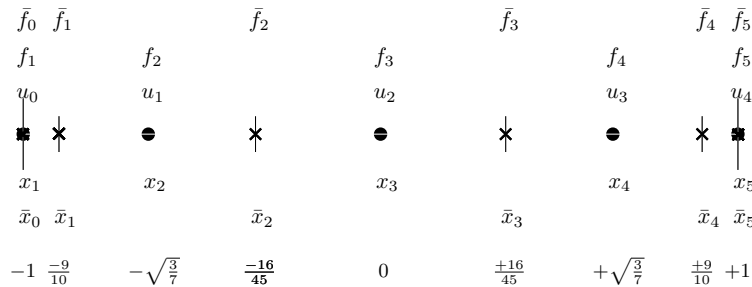


Figure 1: The one-dimensional discretization for $p = 4$ Legendre collocation. Solution LGL points are denoted by \bullet and flux points are denoted by \times .

The operator $\mathcal{I}_{\text{StoF}}^{x_1}$ interpolates the flux at the solution nodes on to a set of flux nodes (see Figure 1).

When applying any of these operators to the scalar entropy equation in space, a hat will be used to differentiate the scalar operator from the full vector operator, e.g.

$$\widehat{\mathbf{P}} = (\mathbf{P}_N \otimes \mathbf{P}_N \otimes \mathbf{P}_N).$$

We finally note that in the present work, the quadrature nodes and solution nodes are collocated.

Using an SBP operator and its equivalent telescoping form, the semi-discrete form of the three-dimensional compressible Navier–Stokes equations (1) for Cartesian grids in each hexahedral element reads (see, e.g. [2, 30])

$$\frac{\partial \mathbf{q}}{\partial t} + \sum_{m=1}^3 \left(\mathbf{P}_{x_m}^{-1} \Delta_{x_m} \bar{\mathbf{f}}_{x_m}^{(I)} - \mathbf{D}_{x_m} \mathbf{f}_{x_m}^{(V)} \right) = \sum_{m=1}^3 \mathbf{P}_{x_m}^{-1} \left(\mathbf{g}_{x_m}^{(B)} + \mathbf{g}_{x_m}^{(In)} \right). \quad (22)$$

We remark that we have omitted any term involved with the geometrical mapping of the elements in order to simplify the discussion. The extension of the following analysis to curvilinear grids is straightforward and all the theorems presented herein remain valid. The interested reader is referred to the supplementary python script (see Appendix A) which symbolically verifies all the proofs for the general case of curvilinear grids. Furthermore, a simple, dimension-agnostic implementation of the entropy stable solid wall boundary conditions for the general case is provided in Appendix B.

The vectors $\mathbf{g}_{x_m}^{(B)}$ in (22) enforce the boundary conditions, while $\mathbf{g}_{x_m}^{(In)}$ patches interfaces together using a SAT approach [1]. The derivatives appearing in the viscous fluxes $\mathbf{f}_{x_m}^{(V)}$ are also computed using the operators \mathbf{D}_{x_m} defined in (21).

The discrete no-slip wall boundary conditions constructed herein follows a local discontinuous Galerkin-type approach [18], where also the gradient of the entropy variables is penalized. The penalization of the gradient represents one of the main novelties in this work within the context of SBP-SAT discretizations for the imposition of boundary conditions. In fact, this is a key difference with respect to previous work [2] which introduced for the first time an entropy stable approach to impose solid wall boundary conditions.

Following the same procedure based on local discontinuous Galerkin (LDG) and interior penalty approach (IP) described in [1, 32], equation (22) can be recast as

$$\frac{\partial \mathbf{q}}{\partial t} + \sum_{m=1}^3 \left(\mathbf{P}_{x_m}^{-1} \Delta_{x_m} \bar{\mathbf{f}}_{x_m}^{(I)} - \sum_{j=1}^3 \mathbf{D}_{x_m} [\mathbf{C}_{m,j}] \Theta_{x_j} \right) = \sum_{m=1}^3 \mathbf{P}_{x_m}^{-1} \left(\mathbf{g}_{x_m}^{(B),q} + \mathbf{g}_{x_m}^{(In),q} \right) \quad (23a)$$

$$\Theta_{x_m} - D_{x_m} \mathbf{w} = \mathbf{P}_{x_m}^{-1} \left(\mathbf{g}_{x_m}^{(B),\Theta} + \mathbf{g}_{x_m}^{(In),\Theta} \right), \quad m = 1, 2, 3, \quad (23b)$$

where Θ_{x_m} is the gradient of the entropy variables in the m -th direction, whereas $\mathbf{g}_{x_m}^{(B),q}$, $\mathbf{g}_{x_m}^{(B),\Theta}$ and $\mathbf{g}_{x_m}^{(In),q}$, $\mathbf{g}_{x_m}^{(In),\Theta}$ are the SAT penalty boundary (B) and interface (I) terms on the conservative variable \mathbf{q} , and the gradient of the entropy variable Θ , respectively [32]. The contributions of the interface penalty terms are non-zero only in the normal direction to the interface. The matrices $[\mathbf{C}_{m,j}]$ are block diagonal matrices with blocks of size 5, corresponding to the viscous coefficients at each solution point.

Remark 3.1. *In order to build a high-order accurate entropy-conservative/stable spatial discretization, the linear interpolation operation, $\mathcal{I}_{StoF}^{x_m}$ is replaced with a non-linear interpolation operator (or equivalently, the linear SBP operator is replaced with a non-linear SBP operator) [2, 21, 22, 30]. Critically, the resulting non-linear operator, $\mathbf{P}_{x_m}^{-1} \Delta_{x_m} \bar{\mathbf{f}}_{x_m,sc}^{(I)}$ or $\mathbf{P}_{x_m}^{-1} \Delta_{x_m} \bar{\mathbf{f}}_{x_m,ssr}^{(I)}$ (entropy conservative or entropy stable, respectively), has the property that when contracted with the entropy variables and discretely integrated over the domain, the result is a discrete surface integral with respect to the entropy flux; that is, for the entropy-conservative formulation, the operator telescopes in the entropy flux. For example*

$$\mathbf{w}^\top \mathbf{P} \mathbf{P}_{x_1}^{-1} \Delta_{x_1} \bar{\mathbf{f}}_{x_1,sc}^{(I)} = \mathbf{1}^\top \left(\hat{\mathbf{P}}_{x_2,x_3} \hat{\mathbf{B}}_{x_1} \mathbf{F}_{x_1} \right) \approx \int_{x_1=1} \mathcal{F}_{x_1} dx_2 dx_3 - \int_{x_1=0} \mathcal{F}_{x_1} dx_2 dx_3,$$

where $\bar{\mathbf{f}}_{x_1,sc}^{(I)}$ is the vector that results from the entropy-conservative non-linear interpolation of the flux (this flux is replaced with $\bar{\mathbf{f}}_{x_1,ssr}^{(I)}$ for the entropy-stable version), $\mathbf{1}$ is a vector of ones of appropriate size, and \mathbf{F}_{x_1} is a vector of the entropy flux in the x_1 direction evaluated on the mesh nodes.

To obtain an equation for the entropy of the system, we follow the entropy stability analysis presented in [30, 32]. Therefore, multiplying the two discrete equations by $\mathbf{w}^\top \mathbf{P}$ and $([\mathbf{C}_{m,j}] \Theta_{x_j})^\top \mathbf{P}$, respectively, the expression for the time derivative of the entropy function in each element is

$$\begin{aligned} & \frac{d}{dt} \mathbf{1}^\top \hat{\mathbf{P}} \mathbf{S} + \mathbf{DT} + \mathbf{1}^\top \left(\hat{\mathbf{P}}_{x_2,x_3} \hat{\mathbf{B}}_{x_1} \mathbf{F}_{x_1} + \hat{\mathbf{P}}_{x_1,x_3} \hat{\mathbf{B}}_{x_2} \mathbf{F}_{x_2} + \hat{\mathbf{P}}_{x_1,x_2} \hat{\mathbf{B}}_{x_3} \mathbf{F}_{x_3} \right) \\ &= \mathbf{w}^\top \left(\mathbf{P}_{x_2,x_3} \mathbf{B}_{x_1} [\mathbf{C}_{1,j}] \Theta_{x_j} + \mathbf{P}_{x_1,x_3} \mathbf{B}_{x_2} [\mathbf{C}_{2,j}] \Theta_{x_j} + \mathbf{P}_{x_1,x_2} \mathbf{B}_{x_3} [\mathbf{C}_{3,j}] \Theta_{x_j} \right) \\ &+ \mathbf{w}^\top \left(\mathbf{P}_{x_2,x_3} \left(\mathbf{g}_{x_1}^{(B),q} + \mathbf{g}_{x_1}^{(In),q} \right) + \mathbf{P}_{x_1,x_3} \left(\mathbf{g}_{x_2}^{(B),q} + \mathbf{g}_{x_2}^{(In),q} \right) + \mathbf{P}_{x_1,x_2} \left(\mathbf{g}_{x_3}^{(B),q} + \mathbf{g}_{x_3}^{(In),q} \right) \right) \quad (24) \\ &+ ([\mathbf{C}_{1,j}] \Theta_{x_j})^\top \mathbf{P}_{x_2,x_3} \left(\mathbf{g}_{x_1}^{(B),\Theta} + \mathbf{g}_{x_1}^{(In),\Theta} \right) + ([\mathbf{C}_{2,j}] \Theta_{x_j})^\top \mathbf{P}_{x_1,x_3} \left(\mathbf{g}_{x_2}^{(B),\Theta} + \mathbf{g}_{x_2}^{(In),\Theta} \right) \\ &+ ([\mathbf{C}_{3,j}] \Theta_{x_j})^\top \mathbf{P}_{x_1,x_2} \left(\mathbf{g}_{x_3}^{(B),\Theta} + \mathbf{g}_{x_3}^{(In),\Theta} \right), \end{aligned}$$

where

$$\mathbf{DT} = \left\| \sqrt{[\mathbf{C}_{m,j}]} \Theta_j \right\|_{\mathbf{P}}^2,$$

is a positive quadratic term in the approximation of the first derivative of the solution [2], $\mathbf{1}$ is the unit vector of appropriate size, and \mathbf{F}_{x_m} the vector of the entropy flux in the m -th direction.

Equation (24) can be conveniently rewritten as

$$\frac{d}{dt} \mathbf{1}^\top \hat{\mathbf{P}} \mathbf{S} + \mathbf{DT} = \mathbf{\Xi}. \quad (25)$$

This compact notation will be useful in Section 5.

As in the continuous analysis, we assume that we have a hexahedral element with edge length equal to one, and consider only the plane $(0, x_2, x_3)$ as wall boundary face. Thus, we have $\mathbf{g}_{x_m}^{(In),q} = \mathbf{g}_{x_m}^{(In),\Theta} = 0, \forall m$. We also assume that all the points that lie on the other faces of the cube are treated in an entropy stable fashion such that their contribution can be neglected. With these assumptions, equation (24) reduces to

$$\begin{aligned} \frac{d}{dt} \mathbf{1}^\top \widehat{\mathbf{P}} \mathcal{S} + \mathbf{D}\mathbf{T} + \mathbf{1}^\top \widehat{\mathbf{P}}_{x_2, x_3} \widehat{\mathbf{B}}_{x_1}^- \mathbf{F}_{x_1} &= \mathbf{w}^\top \mathbf{P}_{x_2, x_3} \mathbf{B}_{x_1}^- [\mathbf{C}_{1,j}] \Theta_{x_j} \\ &+ \mathbf{w}^\top \mathbf{P}_{x_2, x_3} \mathbf{g}_{x_1}^{(B),q} + ([\mathbf{C}_{1,j}] \Theta_{x_j})^\top \mathbf{P}_{x_2, x_3} \mathbf{g}_{x_1}^{(B),\Theta}. \end{aligned} \quad (26)$$

The penalty source term $\mathbf{g}_{x_1}^{(B),q}$ is composed of three design-order terms plus a source boundary term

$$\mathbf{g}_{x_1}^{(B),q} = \mathbf{g}_{x_1}^{(B,I),q} + \mathbf{g}_{x_1}^{(B,V),q} + \mathcal{M}^{(B,V)} + \mathcal{L}^{(B,V)}, \quad (27)$$

where

$$\mathbf{g}_{x_1}^{(B,I),q} = -\mathbf{B}_{x_1}^- \left(\mathbf{f}_{x_1} - \mathbf{f}^{sc}(\mathbf{v}, \mathbf{v}^{(B,I)}) \right), \quad (28)$$

$\mathbf{f}^{sc}(\mathbf{v}, \mathbf{v}^{(B,I)})$ is the entropy conservative flux vector, which is a function of the two arguments \mathbf{v} and $\mathbf{v}^{(B,I)}$, and

$$\mathbf{g}_{x_1}^{(B,V),q} = \frac{1}{2} \mathbf{B}_{x_1}^- \left([\mathbf{C}_{1,j}] \Theta_{x_j} - \mathbf{f}_{x_1}^{(B,V)} \right). \quad (29)$$

For clarity of presentation, the expressions of $\mathcal{M}^{(B,V)}$, which is used to add provably dissipation, and of $\mathcal{L}^{(B,V)}$ will be given and analyzed at the end of the Section.

The penalty $\mathbf{g}_{x_1}^{(B),\Theta}$ contains a single design-order term

$$\mathbf{g}_{x_1}^{(B),\Theta} = \frac{1}{2} \mathbf{B}_{x_1}^- \left(\mathbf{w} - \mathbf{w}^{(B,V)} \right). \quad (30)$$

In each of the contributions, the first component (the numerical state) is constructed from the numerical solution, while the second component (the boundary state) is constructed from a combination of the numerical solution and four independent components of physical boundary data.

In what follows, we analyze each of these contributions from (26) separately. We further restrict our analysis to a single solution point lying on the wall boundary face. Therefore, we will no longer make use of the bold notation, which will be replaced by italics to denote the vector of five components at the collocated point.

The term given in (28) enforces the Euler no-penetration wall condition through the inviscid flux of the compressible Euler equations. The boundary state is formed by constructing an entropy conservative flux based on the numerical state in primitive variables at the face point, v , and a manufactured boundary state given by the vector of the primitive variables

$$v^{(B,I)} = \text{diag}([1, -1, 1, 1, 1]), \quad v = (\rho, -\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3, \mathcal{T})^\top. \quad (31)$$

The term defined in (29), together with the penalty (30), allow a weak imposition of the no-slip condition

$$\mathcal{U}_m = \mathcal{U}_m^{wall}, \quad \mathcal{U}_1^{wall} = 0, \quad m = 2, 3,$$

in an entropy conservative way, where U_m^{wall} is the m -th component of the wall velocity. Furthermore, entropy flow is enforced if $\kappa \left(\frac{\partial \mathcal{T}}{\partial x_1} \frac{1}{\mathcal{T}} \right) \neq 0$.

Using the following primitive variables

$$v^{(B,V)} = \left(\rho, -\mathcal{U}_1, -\mathcal{U}_2 + 2\mathcal{U}_2^{wall}, -\mathcal{U}_3 + 2\mathcal{U}_3^{wall}, \mathcal{T} \right)^\top, \quad (32)$$

we define the point-wise boundary viscous flux $\mathcal{F}_{x_1}^{(B,V)}$, the term $\mathcal{M}^{(B,V)}$, and source term $\mathcal{L}^{(B,V)}$ as

$$\mathcal{F}_{x_1}^{(B,V)} = \mathbf{C}_{1,j}^{(B,V)} \Theta_{x_j}^{(B,V)}, \quad (33)$$

$$\mathcal{M}^{(B,V)} = L \left(w - w^{(B,V)} \right), \quad (34)$$

$$\mathcal{L}^{(B,V)} = -(0, 0, 0, 0, 1)^\top \mathcal{T} \mathbf{g}(t), \quad (35)$$

where $\mathbf{g}(t) = \kappa \frac{\partial \mathcal{T}}{\partial x_1} \frac{1}{\mathcal{T}}$ is a given L^2 function. Thus, in case of a boundary condition with imposed non-zero heat flux, $\mathbf{g}(t)$ is non-zero. We remark that the vector $v^{(B,V)}$ is used to evaluate the matrix of the viscous coefficients $\mathbf{C}_{1,j}^{(B,V)}$, as well as to compute the penalty term in (30).

The manufactured gradient of the entropy variables at the boundary $\Theta_{x_j}^{(B,V)}$ is constructed using the following procedure:

- Rotate the gradient of the entropy variables Θ_{x_j} to the gradient of the primitive variables, v ,

$$\Pi_{x_j} = \frac{\partial V}{\partial W} \Theta_{x_j}, \quad j = 1, 2, 3, \quad (36)$$

where $\frac{\partial V}{\partial W}$ is the Jacobian of the transformation in primitive variables with respect to the entropy variables evaluated at the face point.

- Construct the gradient of the primitive variables at the wall boundary point

$$\Pi_{x_j}^{(B,V)} = \text{diag}([-1, 1, 1, 1, -1]) \Pi_{x_j}, \quad j = 1, 2, 3. \quad (37)$$

This choice follows the imposition of Neumann boundary conditions in the context of the nodal discontinuous Galerkin method [5].

- Rotate the gradient of the primitive variables $\Pi_{x_j}^{(B,V)}$ to the gradient of the entropy variables

$$\Theta_{x_j}^{(B,V)} = \frac{\partial W}{\partial V} \Big|_{(B)} \Pi_{x_j}^{(B,V)}, \quad j = 1, 2, 3, \quad (38)$$

where $\frac{\partial W}{\partial V} \Big|_{(B)}$ is the Jacobian of the entropy variables with respect to the primitive variables evaluated using the state $v^{(B,V)}$ defined in (32).

Finally, the entropy variables $w^{(B,V)}$ needed in (30) are computed from the primitive variables defined in expression (32) by using the relations in (7).

The matrix L in (34) is a negative semi-definite 5x5 matrix which is defined as

$$L = -\beta \frac{\mathbf{C}_{1,1} + \mathbf{C}_{1,1}^{(B,V)}}{2}, \quad (39)$$

where $\mathbf{C}_{1,1}$ and $\mathbf{C}_{1,1}^{(B,V)}$ are the positive semi-definite viscous coefficient matrices in the normal direction ($m = j = 1$), respectively evaluated using the states v and $v^{(B,V)}$, and β is a positive coefficient that modulates the strength of the penalty term. This coefficient has to scale as the inverse of the typical element length⁷.

Summarizing, the penalty at the face point for the conservative variables \mathbf{q} is the sum of two terms:

- the difference between inviscid and entropy conservative fluxes in the normal direction,
- the difference between internal viscous and boundary viscous fluxes in the normal direction.

The penalty on the gradient of the entropy variables Θ is instead given by the difference between the solution at the node and the data imposed at the boundary expressed in terms of entropy variables.

The entropy conservation and stability of the penalty source terms (27) and (30) is demonstrated in the following three theorems. The first theorem, which ensures entropy conservation for the inviscid SAT penalty in (27), and which enforces the no penetration condition, is Theorem 5.1 in [2]. The second, is a new theorem and it ensures entropy conservation or stability for the viscous SAT penalty. The third theorem is also new and ensures that the term $\mathcal{M}^{(B,V)}$ is a dissipative entropy contribution.

Theorem 3.1. *The penalty inviscid flux contribution in equation (27) is entropy conservative if the vector $v^{(B,I)}$ is defined as in (31).*

Proof. See Theorem 5.1 in [2]. □

Theorem 3.2. *The penalty terms for the viscous flux on the conserved variables (27), together with the viscous penalty on the gradient of the entropy variables (30), are*

- entropy conservative if the wall is adiabatic, i.e. $\mathbf{g}(t) = 0$,
- entropy stable in the presence of a heat flux, i.e. $\mathbf{g}(t) \neq 0$, where $\mathbf{g}(t)$ is a given L^2 function.

Proof. By substituting into (26) the expressions for $\mathbf{g}_{x_1}^{(B),q}$ with $\mathcal{M}^{(B,V)} = 0$ (i.e. no dissipation) and $\mathbf{g}_{x_1}^{(B),\Theta}$ given in (27) and (30), respectively, yields

$$\frac{d}{dt} \mathbf{1}^\top \widehat{\mathbf{P}} \mathbf{S} + \mathbf{D}\mathbf{T} = \mathbf{1}^\top \widehat{\mathbf{P}}_{x_2, x_3} \mathbf{g}(t).$$

For an adiabatic wall $\mathbf{g}(t) = 0$ and therefore the proposed boundary conditions are entropy conservative. For $\mathbf{g}(t) \neq 0$ the boundary conditions are entropy stable because the contribution to the time rate of change of the entropy function is only a function of the data $\mathbf{g}(t)$. □

Theorem 3.3. *The interior penalty term*

$$\mathcal{M}^{(B,V)} = [L] \left(\mathbf{w} - \mathbf{w}^{(B,V)} \right) \tag{40}$$

added to the SAT (27) is entropy dissipative.

⁷This is done such that the scaling remains dimensionally consistent.

Proof. By expanding the contraction $\mathbf{w}^\top \mathbf{P}_{x_2, x_3} \mathbf{g}_{x_1}^{(B), q}$ in (26), and by focusing on the dissipation term only, we arrive at the following point-wise contribution to the time-rate of change of the entropy function

$$w^\top L \left(w - w^{(B, V)} \right). \quad (41)$$

Note that we have omitted an extra positive scaling factor corresponding to the entry of the matrix \mathbf{P}_{x_2, x_3} associated with a wall boundary point. Plugging in the definitions of the matrix L (39), w and $w^{(B, V)}$ we obtain

$$w^\top L \left(w - w^{(B, V)} \right) = -\frac{2\beta\mu}{3\mathcal{T}} \left(4\mathcal{U}_1^2 + 3 \left(\mathcal{U}_2^2 - \mathcal{U}_2^{wall} \right)^2 + 3 \left(\mathcal{U}_3^2 - \mathcal{U}_3^{wall} \right)^2 \right),$$

which completes the proof. \square

Remark 3.2. *In order to construct an entropy stable solid wall boundary condition for the Euler equations, it is necessary to add provable entropy dissipation. One solution consists of replacing the entropy conservative flux \mathbf{f}^{sc} in (27) with an entropy stable flux \mathbf{f}^{ssr} in (27) as described in [2].*

4. A common SAT procedure for the imposition of wall boundary conditions and interior interface coupling

The proposed approach for imposing the solid wall boundary conditions allows for a SAT implementation which is identical to the interface treatment shown in [1]. We can use a single subroutine with different inputs corresponding to the imposition of the interior interface couplings, or of the adiabatic solid wall or of the wall with a prescribed heat entropy flow. In fact, the interior interface coupling can be written as (see equations (16a-16d) in [1])

$$\frac{\partial \mathbf{q}_l}{\partial t} + \sum_{m=1}^3 \left(\mathbf{P}_{x_{m,l}}^{-1} \Delta_{x_{m,l}} \bar{\mathbf{f}}_{x_{m,l}}^{(I)} - \sum_{j=1}^3 \mathbf{D}_{x_{m,l}} [\mathbf{C}_{m,j,l}] \Theta_{x_j,l} \right) = \sum_{m=1}^3 \mathbf{P}_{x_{m,l}}^{-1} \mathbf{g}_{x_{m,l}}^{(In),q}, \quad (42a)$$

$$\Theta_{x_{m,l}} - \mathbf{D}_{x_{m,l}} \mathbf{w} = \mathbf{P}_{x_{m,l}}^{-1} \mathbf{g}_{x_{m,l}}^{(In),\Theta}, \quad m = 1, 2, 3, \quad (42b)$$

$$\frac{\partial \mathbf{q}_r}{\partial t} + \sum_{m=1}^3 \left(\mathbf{P}_{x_{m,r}}^{-1} \Delta_{x_{m,r}} \bar{\mathbf{f}}_{x_{m,r}}^{(I)} - \sum_{j=1}^3 \mathbf{D}_{x_{m,r}} [\mathbf{C}_{m,j,r}] \Theta_{x_j,r} \right) = \sum_{m=1}^3 \mathbf{P}_{x_{m,r}}^{-1} \mathbf{g}_{x_{m,r}}^{(In),q}, \quad (42c)$$

$$\Theta_{x_{m,r}} - \mathbf{D}_{x_{m,r}} \mathbf{w} = \mathbf{P}_{x_{m,r}}^{-1} \mathbf{g}_{x_{m,r}}^{(In),\Theta}, \quad m = 1, 2, 3, \quad (42d)$$

which have exactly the same structure as LDG-IP approach used for the imposition of the solid wall boundary conditions except for the boundary penalty interface terms, $\mathbf{g}_{x_{m,r}}^{(B),\cdot}$ in equation (23), which are replaced by the interior penalty interface coupling terms, $\mathbf{g}_{x_{m,r}}^{(In),\cdot}$ in equations (42).

5. Numerical results

In this section we present four three-dimensional test cases which demonstrate the robustness of the new wall boundary conditions coupled with the family of high-order accurate entropy-stable interior SBP-SAT algorithms developed in [1, 2, 12, 30]. The systems of ordinary differential equations arising from the spatial discretizations are integrated using the fourth-order accurate Dormand–Prince method [33] endowed with an adaptive time stepping technique based on digital signal processing [34, 35]. We note that small enough tolerances are always used to make the temporal error negligible.

The unstructured grid solver used herein has been developed at the Extreme Computing Research Center (ECRC) at KAUST on top of the Portable and Extensible Toolkit for Scientific computing (PETSc) [36], its mesh topology abstraction (DMPLEX) [37] and scalable ordinary differential equation (ODE)/differential algebraic equations (DAE) solver library [38], and the Message Passing Interface (MPI). Additionally, the numerical solver is based on the algorithms proposed in [1, 2, 12, 30]. It uses a transformation from computational to physical space that satisfies both the entropy conservation and the geometric conservation law at the semi-discrete level [21]. Unless otherwise stated, the meshes used in this work have been generated using the GMSH package [39].

We present the numerical results for five test cases:

- Laminar flow in a pipe with annular section to verify the accuracy of procedure for the imposition of the solid wall boundary conditions;
- Laminar flow in a lid-driven cavity to validate the entropy conservation and entropy stability properties of the interior discretization operator coupled with the boundary procedure;
- Laminar flow past a three-dimensional cylinder and a sphere to demonstrate the engineering capabilities of the interior discretization operator coupled with the boundary procedure;
- Supersonic turbulent flow past a three-dimensional rod with square section to demonstrate the robustness of the solver and solid wall boundary conditions (“standard” SBP-SAT operators and procedure for imposing solid wall boundary conditions based on linear analysis crash).

5.1. Flow in pipe with annular cross-section

In this section we investigate the accuracy of the solid wall boundary conditions. The proposed entropy stable no-slip wall boundary conditions do not force the numerical solution to exactly fulfill the boundary conditions. Instead the effect can be described as a rubber-band pulling the solution towards the boundary conditions. The computed boundary value (or numerical state) typically deviates slightly from the prescribed value but the deviation is reduced as the grid is refined. To verify the accuracy of the new procedure, we perform a grid convergence study for the flow in a full three-dimensional pipe with annular cross-section. For an incompressible flow, this test case has an analytical solution [40] for both the velocity distribution and the volume flux through the annular pipe. The expression for the axial velocity distribution, \mathcal{U}_1 , as a function of the radial coordinate, r , is

$$\mathcal{U}_1(r) = \frac{G}{4\mu} \left[(R_i^2 - r^2) + (R_o^2 - R_i^2) \frac{\ln(r/R_i)}{\ln(R_o/R_i)} \right], \quad (43)$$

where G is the pressure gradient forcing term and R_i and R_o are the inner and outer radii of the pipe, respectively. Herein, we set $R_o = 0.5$, $R_o/R_i = 4$ and $G/\mu = 1$. We highlight that we have chosen this test problem because i) it has an analytical solution which cannot be represented exactly by the polynomial space of the numerical solution, and ii) it requires the use of curved boundary element faces to capture accurately the geometry of the annular section of the pipe.

The code that is used is a compressible code and in order to obtain results that are very close to those found for the incompressible equations, a Mach number of $M = 1.0e - 05$ is considered. Periodic boundary condition are used in the axial direction.

We run a grid convergence study for $p = 2, 3, 4$ with a sequence of nested grids generated using rational Bezier basis functions such that the geometrical description of the pipe is preserved exactly for each refinement level.

The error in the axial velocity profile are computed using discrete norms as follows:

$$\begin{aligned} \text{Discrete } L^1 : \|\mathbf{u}\|_{L^1} &= \sum_{\kappa=1}^K \mathbf{1}_{N_\kappa}^\top \mathbf{P}_\kappa \mathbf{J}_\kappa \text{abs}(\mathbf{u}_\kappa), \\ \text{Discrete } L^2 : \|\mathbf{u}\|_{L^2}^2 &= \sum_{\kappa=1}^K \mathbf{u}_\kappa^\top \mathbf{P}_\kappa \mathbf{J}_\kappa \mathbf{u}_\kappa, \\ \text{Discrete } L^\infty : \|\mathbf{u}\|_{L^\infty} &= \max_{\kappa=1\dots K} \text{abs}(\mathbf{u}_\kappa), \end{aligned}$$

where \mathbf{J}_κ is the metric Jacobian of the curvilinear transformation from physical space to computational space of the k -th hexahedral element and K is the total number of non-overlapping hexahedral elements in the mesh.

The results of the grid convergence study are shown in Tables 1, 2 and 3 where the numbering the first column indicates the number of elements in the radial, angular and axial coordinates. It can be seen that the computed order of accuracy is very close to the formal value of $\sim (p + 1)$.

Grid	L^1	Rate	L^2	Rate	L^∞	Rate
4	8.77e-02	-	1.79e-01	-	5.69e-01	-
8	1.28e-02	-2.77	3.45e-02	-2.38	1.50e-01	-1.92
16	2.03e-03	-2.66	5.47e-03	-2.66	3.12e-02	-2.27
32	2.68e-04	-2.92	7.51e-04	-2.86	5.07e-03	-2.62
64	3.33e-05	-3.01	9.66e-05	-2.96	7.23e-04	-2.81

Table 1: Convergence study for the flow in a pipe with annular section; $p = 2$; error in the axial velocity.

5.2. Lid-driven cavity

Next, we validate the algorithm on the simple problem of the three-dimensional lid-driven cavity with adiabatic solid walls. The domain is a cube with sides of length l discretized using a Cartesian grid composed of eight elements in each direction. A velocity field is imposed on one of the walls, corresponding to a rigid body rotation about the center of the wall at a speed ω (see figure 2). Based on the rotation velocity and the length of the cavity, this example is

Grid	L^1	Rate	L^2	Rate	L^∞	Rate
4	2.04e-02	-	3.87e-02	-	1.54e-01	-
8	2.71e-03	-2.91	5.68e-03	-2.77	3.17e-02	-2.28
16	2.15e-04	-3.65	5.69e-04	-3.32	4.43e-03	-2.84
32	1.25e-05	-4.10	4.40e-05	-3.69	4.63e-04	-3.26
64	6.65e-07	-4.23	2.87e-06	-3.94	4.05e-05	-3.51

Table 2: Convergence study for the flow in a pipe with annular section; $p = 3$; error in the axial velocity.

Grid	L^1	Rate	L^2	Rate	L^∞	Rate
4	4.54e-03	-	9.55e-03	-	4.70e-02	-
8	4.48e-04	-3.34	9.58e-04	-3.32	6.31e-03	-2.90
16	1.89e-05	-4.57	5.68e-05	-4.08	5.14e-04	-3.66
32	5.98e-07	-4.98	2.34e-06	-4.60	2.76e-05	-4.22
64	2.22e-08	-4.75	7.65e-08	-4.95	1.13e-06	-4.61

Table 3: Convergence study for the flow in a pipe with annular section; $p = 4$; error in the axial velocity.

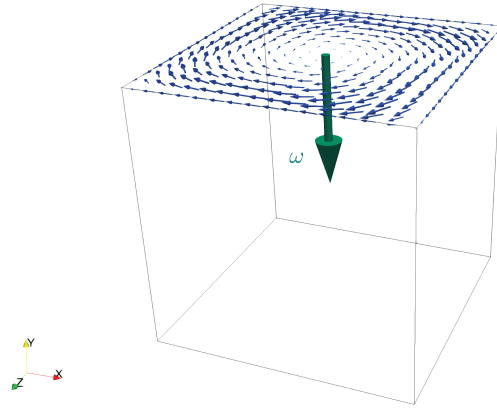


Figure 2: Driven cavity with rigid body rotation ω in one of its faces.

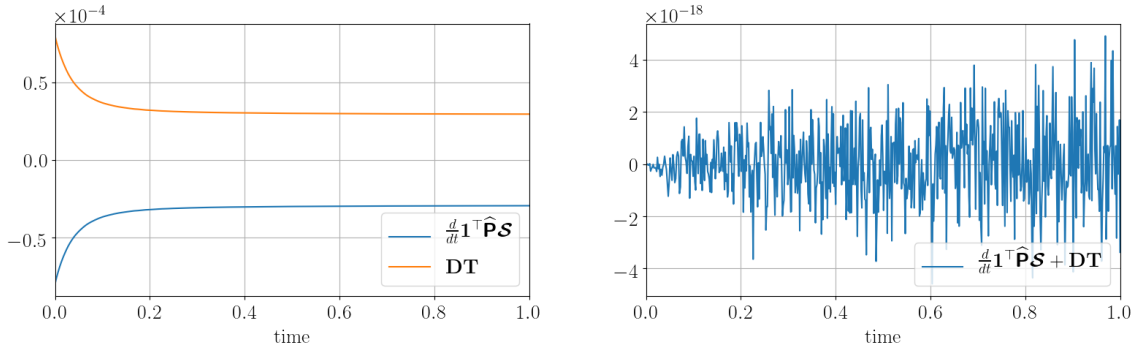


Figure 3: Lid-driven cavity. Left: discrete integral of the time rate of change of the entropy function, $\frac{d}{dt} \mathbf{1}^\top \widehat{\mathbf{P}} \mathbf{S}$, and discrete dissipation term, **DT** (see equation (25)). Right: instantaneous entropy balance (see equation (25)).

characterized by a Reynolds number $Re = l^2 \omega / \nu = 100$ and a Mach number $M = l \omega / c = 0.05$. All the dissipation terms used for the interface coupling [1] and the imposition of the boundary conditions are turned off, including upwind and interior-penalty SAT terms. The two terms on the left-hand side of equation (25), $\frac{d}{dt} \mathbf{1}^\top \widehat{\mathbf{P}} \mathbf{S} + \mathbf{DT}$, are monitored at every time step. Note that because we do not include any dissipation terms for the interface couplings and adiabatic wall boundary conditions are used, Ξ in equation (25) is zero. The monitored values are reported in Figure 3, left panel, together with the error committed in entropy conservation, right panel, which is below machine (double) precision. Therefore, we have confirmed numerically that the newly developed wall boundary conditions together with the interior discretization operator are entropy conservative when all the entropy dissipative terms are turned off.

We also present the results of a modified version of the lid-driven cavity example by considering non-adiabatic boundaries, and by imposing a nonzero entropy flux in a face adjacent to the rotating face, given as $\mathbf{g}(t) = 10^{-4} \sin(4\pi t)$ (see figure 4). Figure 5 shows the terms on the entropy balance for this example, including the boundary contribution (left panel), and the error in the entropy conservation (right), which is below machine (double) precision.

5.3. Subsonic flow past a cylinder

We further explore the engineering capabilities of the entropy-stable SBP-SAT operators and the numerical procedure for the weak imposition of the solid wall boundary conditions by simulating the flow around a cylinder, a canonical example of external flows with important applications such as particle transport, fluid-structure interaction and bluff body aerodynamics, that has been extensively studied both numerically [41–44] and experimentally [45–49].

We described the flow in a Cartesian coordinate system (x_1, x_2, x_3) , with the free-stream velocity aligned in the x_1 direction. A circle of diameter d is centered at the origin, with the domain of interest delimited by a rectangular box that respectively extends $20d$ and $60d$ upstream and downstream of the flow direction, and $30d$ in the x_2 direction. Such a 2D domain is then extruded a distance d in the x_3 direction. We prescribe an adiabatic no-slip wall boundary condition on the surface of the cylinder; periodic boundaries are applied in the x_3 direction. The remaining faces of the box are treated as a far field.

The domain is discretized as follows: we first mesh the (x_1, x_2) plane with second-order quadrilateral elements, and include a boundary layer around the cylinder wall as pictured in Figure 6.

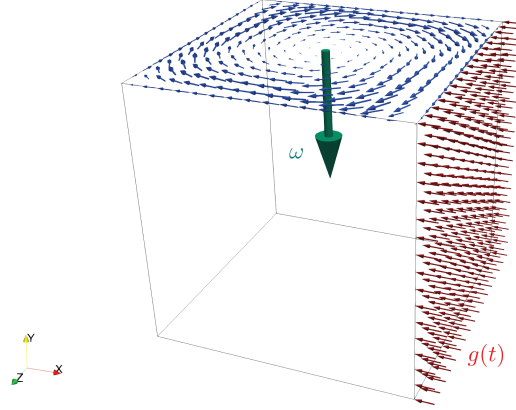


Figure 4: Driven cavity with rigid body rotation ω in one of its faces and a time-dependent entropy flux $\mathbf{g}(t)$ on and adjacent face.

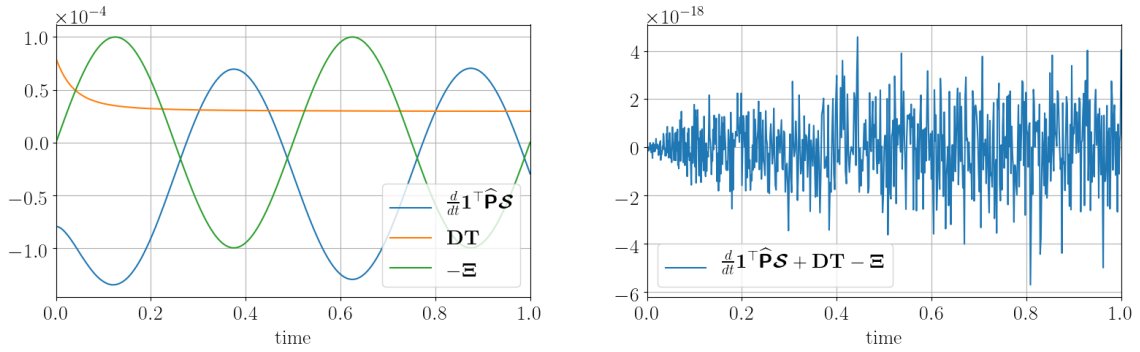


Figure 5: Lid-driven cavity with nonzero entropy flux. Left: discrete integral of the time rate of change of the entropy function, $\frac{d}{dt} \mathbf{1}^\top \widehat{\mathbf{P}} \mathcal{S}$, discrete dissipation term, \mathbf{DT} , and discrete integral of the boundary data contribution, $\Xi = \widehat{\mathbf{P}}_{x_2, x_3} \mathbf{g}(t)$ (see equation (25)). Right: instantaneous entropy balance (see equation (25)).

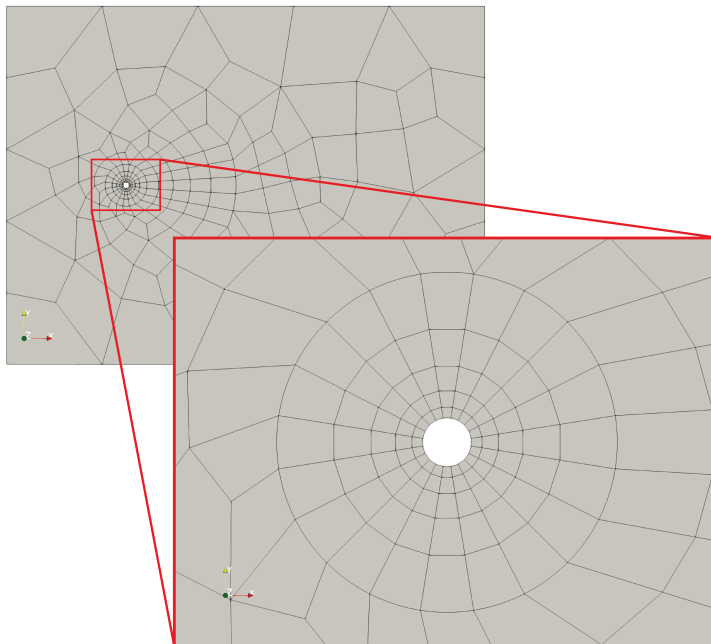


Figure 6: Subsonic flow past a cylinder: mesh cut on the x_3 plane, with a zoom of the boundary layer area.

We then extrude this mesh using three layers of elements in the x_3 direction over the span of the cylinder.

Considering the free-stream velocity u_∞ and the diameter of the cylinder d , the free-stream flow is characterized by a Mach number $M_\infty = 0.1$ and a Reynolds number $Re_\infty = 300$. Under these conditions, the flow developed behind the cylinder is three-dimensional and results in vortex shedding at a constant frequency.

In Table 4 we report the time average drag coefficient, \bar{c}_d , and the Strouhal number, $St = fd/u_\infty$ based on the frequency of the vortex shedding, resulting from uniform p -refinements (with polynomial orders for the solution space ranging from $p = 2$ to $p = 5$) and three levels of uniform h -refinement of the initial mesh described below, resulting in 714 (denoted by $rl = 0$ in Table 4), 5,712 ($rl = 1$), and 45,969 ($rl = 2$) hexahedral elements respectively. Comparisons with results reported in the literature for these aerodynamic coefficients [50] are also provided. From these tables, it can be seen that in all cases the accuracy of the results improve by increasing the order of accuracy of the scheme and the grid resolution. The fifth- ($p = 4$) and sixth-order ($p = 5$) accurate entropy-stable schemes perform very well on the last level of h -refinement considered ($rl = 2$), which is coarser compared to the typical grids used with second-order finite volume and finite differences schemes.

5.4. Subsonic flow past a sphere

We then test our implementation within a more complicated setting represented by the flow around a sphere. In this case, a sphere of diameter d is centered at the origin, and a box is respectively extended $20d$ and $60d$ upstream and downstream of the flow direction; the box size is $30d$ in both the x_2 and x_3 directions. As boundary conditions, we consider adiabatic solid walls at the surface of the sphere and far field on all faces of the box.

Poly. order	$rl = 0$		$rl = 1$		$rl = 2$	
	\bar{c}_d	St	\bar{c}_d	St	\bar{c}_d	St
$p = 2$	1.579	0.1995	1.541	0.1998	1.443	0.1999
$p = 3$	1.439	0.1997	1.439	0.1999	1.383	0.1999
$p = 4$	1.458	0.1998	1.386	0.1999	1.384	0.1999
$p = 5$	1.414	0.1999	1.382	0.1999	1.386	0.1999
[50]	1.381	0.2000	1.381	0.2000	1.381	0.2000

Table 4: Subsonic flow past a cylinder: mean drag coefficient and Strouhal number for different polynomial orders p and uniform refinements rl , $Re_\infty = 300$, $M_\infty = 0.1$. Reference values are also provided.

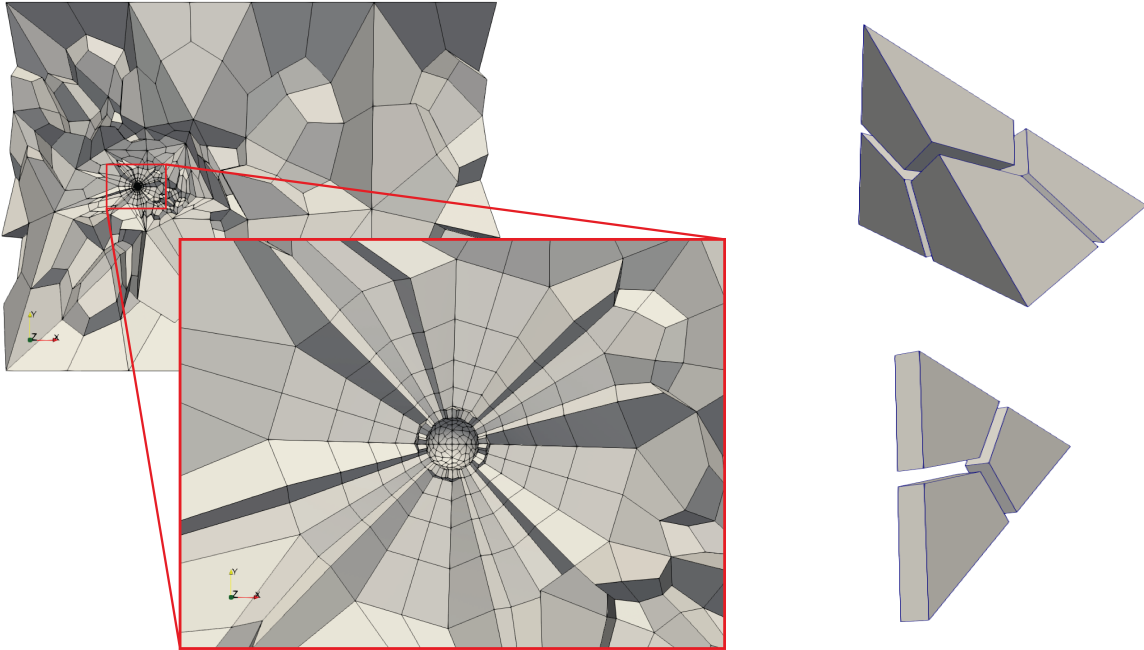


Figure 7: Subsonic flow past a sphere. Left: mesh cut on the x_3 plane, with a zoom of the boundary layer area. Right panel, top: tetrahedral to hexahedral refinement. Right panel, bottom: prismatic to hexahedral refinement.

The surface of the sphere is first triangulated using second-order simplices and a boundary layer composed of triangular prisms is extruded from the sphere surface for a total length of $3d$. The rest of the domain is meshed with an unstructured tetrahedral mesh. We then obtain an unstructured conforming hexahedral mesh by uniformly splitting each tetrahedron in to four hexahedral elements, and each prism in to three hexahedral elements, resulting in a total of 4,328 hexahedral elements. A cut of the final mesh is illustrated in Figure 7, together with a representative splitting of a tetrahedral and a prismatic cell.

The free-stream flow is characterized by a Mach number $M_\infty = 0.1$ and a Reynolds number $Re_\infty = 300$. Under these conditions, the flow developed behind the sphere is non-axisymmetric, with hairpin vortices shedding from the wake at a constant rate [51], inducing a total non-zero lift force on the sphere. Figure 8 depicts these hairpin vortices with isocontours of the Q-criterion colored by the vorticity magnitude at a given time instant.

As in the previous test case, we analyze the convergence of some quantities of interest under

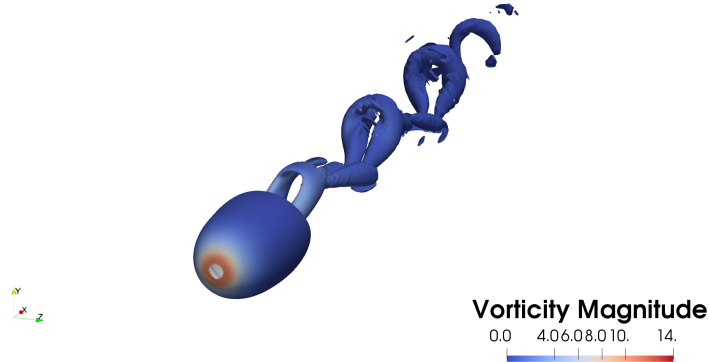


Figure 8: Subsonic flow past a sphere: isocontour 0.02 of the Q-criterion colored by vorticity magnitude at $t = 150$, $Re_\infty = 300$, $M_\infty = 0.1$. Solution with $rl = 2$ and $p = 5$. Notice the hairpin vortices with a nearly constant orientation.

Poly. order	$rl = 0$				$rl = 1$				$rl = 2$			
	\bar{c}_d	\bar{c}_l	θ	St	\bar{c}_d	\bar{c}_l	θ	St	\bar{c}_d	\bar{c}_l	θ	St
$p = 2$	1.125	0.095	-0.34	0.060	0.843	0.076	-0.10	0.140	0.691	0.069	0.45	0.140
$p = 3$	0.815	0.074	-0.18	0.120	0.671	0.068	0.24	0.140	0.659	0.068	1.71	0.140
$p = 4$	0.688	0.069	0.14	0.140	0.656	0.067	1.80	0.140	0.659	0.068	1.70	0.140
$p = 5$	0.650	0.066	0.39	0.140	0.659	0.067	1.68	0.140	0.659	0.068	1.70	0.140
[52]	0.656	0.069	-	0.137	0.656	0.069	-	0.137	0.656	0.069	-	0.137

Table 5: Subsonic flow past a sphere: mean drag coefficient \bar{c}_d , mean lift coefficient \bar{c}_l and its orientation in the $x_1 = 0$ plane θ , and Strouhal number St for different polynomial orders p and uniform refinements l , $Re_\infty = 300$, $M_\infty = 0.1$. Reference values are also provided.

p - and h -refinement. The number of elements in the sequence of nested grids considered are 4,328, 34,624 and 276,992 respectively; the polynomial order of the solution ranges from $p = 2$ to $p = 5$. For this test, we monitor the time average drag coefficient of the sphere, \bar{c}_d , the average lift coefficient, \bar{c}_l (with its orientation θ in the $x_3 - x_2$ plane), and the Strouhal number St based on the frequency of the vortex shedding. We remark that \bar{c}_l is nonzero because of the asymmetric flow features [51]. The results provided in Table 5 are in good agreement with those reported in the literature [52] for sufficiently refined meshes (i.e. $rl = 1$ and $rl = 2$), and polynomial orders greater than or equal to 3.

Figure 8 shows the 0.02 isocontour of the Q-criterion at $t = 150$ for the solution of the flow around a sphere with $rl = 2$, and $p = 5$. At this time, the flow has reached a constant periodic state. From this figure, it can be seen the asymmetric pattern of the flow, consistent with a nonzero \bar{c}_l . Notice the heads of the hairpin vortices are almost aligned with the x_2 axis, in agreement with our reported value of θ for this solution.

5.5. Supersonic flow past a square cylinder

We finally provide further evidence of the robustness of the algorithm in the context of supersonic flow around a square cylinder with $Re_\infty = 10^4$ and $M_\infty = 1.5$, which features shocks, expansion regions and three-dimensional vortical structures [2]. We start with a square of side s placed in the $x_1 - x_2$ plane, and construct an unstructured mesh around it, manually refined

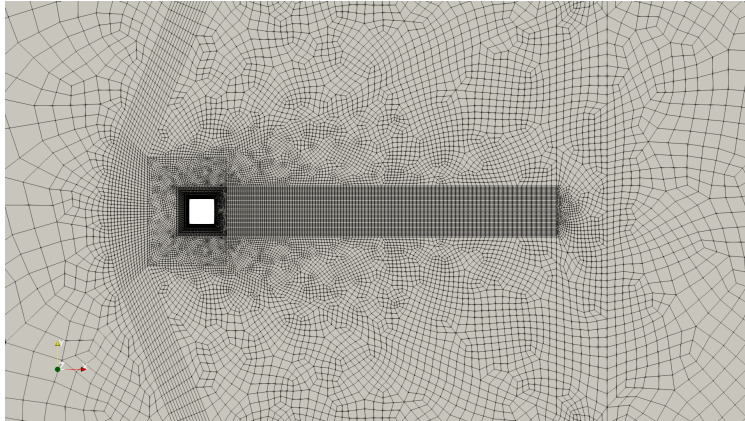


Figure 9: Mesh used for the solution of the supersonic flow past a square cylinder.

in order to capture the main features of the flow (see Figure 9). We then extrude the mesh for a total size s in the x_3 direction using four elements; the final three-dimensional mesh used in the study consists of 87,872 hexahedral elements. The boundary conditions imposed are adiabatic solid wall on the square cylinder surfaces, periodic boundary conditions in the x_3 direction, and far field at the remaining boundaries. The problem is solved using a fourth-order accurate ($p = 3$) discretization.

Figure 10 show the results for the supersonic square cylinder at $t = 100$. At this point in time, the flow is fully unsteady and the shock in front of the cylinder has reached its final position. The flow is characterized by the shock in front of the square cylinder and those in the near wake region. There is also an unsteady wake populated by three-dimensional vortices shedding from the body.

We finally remark that the small oscillations near the shock region are caused by discontinuities in the solution and are expected for this scheme. In fact, we are not using any shock capturing method or reducing the order of scheme at the discontinuity. Nevertheless, the simulation remains stable at all time, and the oscillations are always confined to small regions near the discontinuities. This is a feat unattainable with several alternative approaches to wall boundary conditions based on linear analysis which for this test problem lead to numerical instabilities and an almost immediate crash of the solver.

6. Conclusions

We have used entropy stability and the summation-by-parts framework to derive entropy stable wall boundary conditions for the three-dimensional compressible Navier–Stokes equations in the presence of an adiabatic wall, or a wall with a prescribed heat entropy flow.

A point-wise entropy-stable numerical procedure has been presented for weakly enforcing these solid wall boundary conditions at the semi-discrete level combining a penalty flux approach with a simultaneous-approximation-term technique for the conservative variables and the variables representing the gradient of entropy. The resulting semi-discrete operator mimics exactly the behavior at the continuous level, and the proposed non-linear boundary treatment provides a mechanism for ensuring non-linear stability in the L^2 norm of the continuous and semi-discretized compressible Navier–Stokes equations.

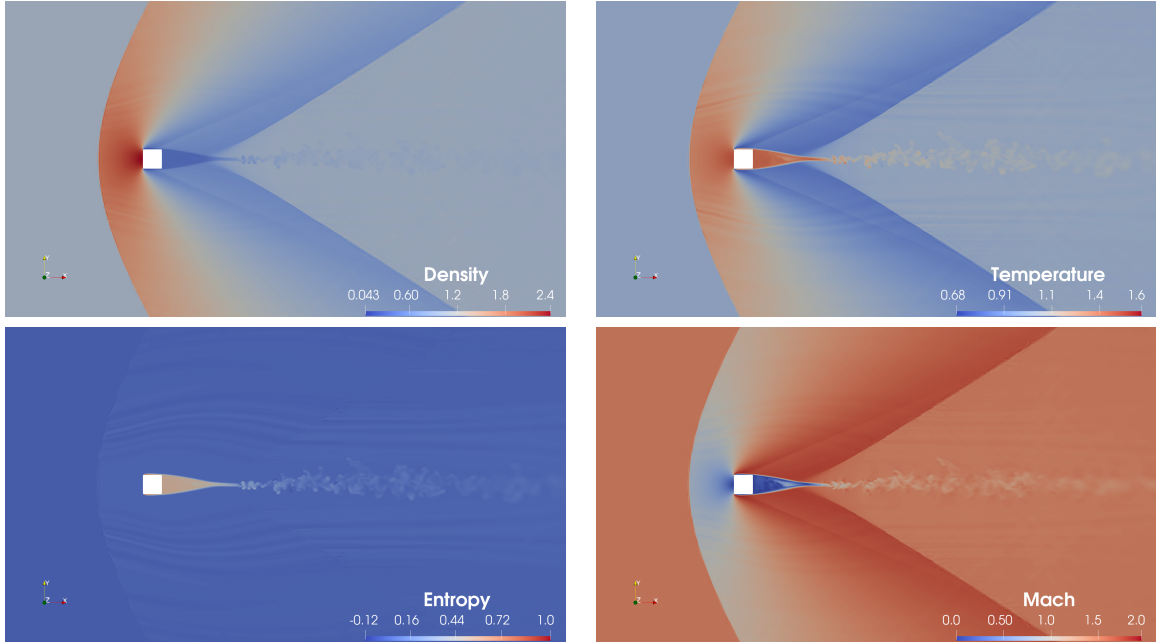


Figure 10: Density, temperature, entropy and local Mach number contours for the supersonic flow around a square cylinder with $Re_\infty = 10^4$ and $M_\infty = 1.5$ at $t = 100$.

The design order properties of the scheme are validated in the context of laminar flow in a pipe with an annular section. Detailed viscous numerical computations in a three-dimensional subsonic lid-driven cavity flow have been presented to assess the accuracy of the proposed numerical techniques. The error in the entropy function balance showed an excellent agreement with the theory with or without heat entropy flux.

Unsteady laminar flow past a cylinder and a sphere have been presented to highlight the efficacy in computing aerodynamic forces; numerical simulations considering both p - and h -refinements showed very good agreement with results available from the literature.

The robustness of the complete semi-discrete operator (i.e. the entropy-stable interior operator coupled with the new boundary treatment) was demonstrated for the supersonic flow past a three-dimensional square cylinder at $Re_\infty = 10^4$ and $M_\infty = 1.5$, as proposed in [2]. This test has been successfully computed with a fourth-order accurate method without the need of introducing artificial dissipation, limiting techniques, or filtering, for the purpose of stabilizing the computations, a feat unattainable with several alternative approaches based on linear analysis only.

Although the robustness and efficacy of the techniques presented in this work have been validated using discontinuous spectral collocation operators on unstructured grids, the new boundary conditions can be applied to a very broad class of spatial discretizations and they are compatible with any diagonal-norm summation-by-parts spatial operator, including finite element, finite difference, finite volume, discontinuous Galerkin, and flux reconstruction schemes.

Acknowledgments

The research reported in this paper was funded by King Abdullah University of Science and Technology. We are thankful for the computing resources of the Supercomputing Laboratory and

the Extreme Computing Research Center at King Abdullah University of Science and Technology.

References

- [1] Parsani, M., Carpenter, M. H., and Nielsen, E. J., “Entropy stable discontinuous interfaces coupling for the three-dimensional compressible Navier–Stokes equations,” *Journal Computational Physics*, Vol. 290, 2015, pp. 132–138.
- [2] Parsani, M., Carpenter, M. H., and Nielsen, E. J., “Entropy stable wall boundary conditions for the three-dimensional compressible Navier–Stokes equations,” *Journal of Computational Physics*, Vol. 292, 2015, pp. 88–113.
- [3] Hesthaven, J. S., *Numerical methods for conservation laws: From analysis to algorithms*, Computational Science and Engineering. 18, SIAM Publishing, Philadelphia, 2017.
- [4] Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., Leer, B., and Visbal, M., “High-order CFD methods: current status and perspective,” *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 8, pp. 811–845.
- [5] Hesthaven, J. S. and Warburton, T., *Nodal discontinuous Galerkin methods: Algorithms, analysis, and applications*, Texts in Applied Mathematics, Springer, 2008.
- [6] Dafermos, C. M., *Hyperbolic conservation laws in continuum physics*, Springer-Verlag, Berlin, 2010.
- [7] Svärd, M., “Weak solutions and convergent numerical schemes of modified compressible Navier–Stokes equations,” *Journal Computational Physics*, Vol. 288, 2015, pp. 19–51.
- [8] Hughes, T. J. R., Franca, L. P., and Mallet, M., “A new finite element formulation for computational fluid dynamics: K. Symmetric forms of the compressible Euler and Navier–Stokes equations and the second law of thermodynamics,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 54, 1986, pp. 223 – 234.
- [9] Tadmor, E., “The numerical viscosity of entropy stable schemes for systems of conservation laws. I,” *Mathematics of Computation*, Vol. 49, 1987, pp. 91–103.
- [10] Fisher, T. C. and Carpenter, M. H., “High-order entropy stable finite difference schemes for nonlinear conservation laws: finite domains,” *Journal Computational Physics*, Vol. 252, 2013, pp. 518–557.
- [11] Carpenter, M. H. and Fisher, T. C., “High-order entropy stable formulations for computational fluid dynamics,” *21st AIAA Computational Fluid Dynamics Conference*, AIAA 2013-2868, American Institute of Aeronautics and Astronautics (AIAA), 2013.
- [12] Carpenter, M. H., Parsani, M., Nielsen, E. J., and Fisher, T. C., “Towards an entropy stable spectral element framework for computational fluid dynamics,” *54th AIAA Aerospace Sciences Meeting*, AIAA 2016-1058, American Institute of Aeronautics and Astronautics (AIAA), 2016.
- [13] Friedrich, L., Winters, A. R., Del Rey Fernández, D. C., Gassner, G. J., Parsani, M., and Carpenter, M. H., “An entropy stable h/p non-conforming discontinuous Galerkin method with the summation-by-parts property,” *Journal of Scientific Computing*, 2018.
- [14] Chan, J., “On discretely entropy conservative and entropy stable discontinuous Galerkin methods,” *Journal of Computational Physics*, Vol. 362, 2018, pp. 346 – 374.
- [15] Ranocha, H., Sayyari, M., Dalcin, L., Parsani, M., and Ketcheson, D. I., “Relaxation Runge–Kutta Methods: Fully-Discrete Explicit Entropy-Stable Schemes for the Euler and Navier–Stokes Equations,” 05 2019, Submitted to SIAM Journal on Scientific Computing.
- [16] Svärd, M. and Özcan, H., “Entropy-stable schemes for the Euler equations with far-field and wall boundary conditions,” *Journal of Scientific Computing*, Vol. 58, No. 1, 2014, pp. 61–89.
- [17] Svärd, M., Carpenter, M. H., and Parsani, M., “Entropy stability and the no-slip wall boundary condition,” *SIAM Journal on Numerical Analysis*, Vol. 56, No. 1, 2018, pp. 256–273.
- [18] Cockburn, B. and Shu, C.-W., “The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems,” *SIAM Journal on Numerical Analysis*, Vol. 35, No. 6, 1998, pp. 2440–2463.
- [19] Carpenter, M. H., Parsani, M., Fisher, T. C., and Nielsen, E. J., “Entropy stable staggered grid spectral collocation for the Burgers’ and the compressible Navier–Stokes equations,” *NASA TM-2015-218990*, 2015.
- [20] Dutt, P., “Stable boundary conditions and difference schemes for Navier–Stokes equations,” *SIAM Journal on Numerical Analysis*, Vol. 25, No. 2, 1988, pp. 245–267.
- [21] Fisher, T. C., *High-order L^2 stable multi-domain finite difference method for compressible flows*, Ph.D. thesis, Purdue University, 2012.
- [22] Tadmor, E., “Entropy stability theory for difference approximations of nonlinear conservation laws and related time-dependent problems,” *Acta Numerica*, Vol. 12, 2003, pp. 451–512.

- [23] Godunov, S. K., “An interesting class of quasilinear systems,” *Dokl. Akad. Nauk SSSR*, Vol. 139, No. 3, 1961, pp. 521–523.
- [24] Harten, A., “On the symmetric form of systems of conservation laws with entropy,” *Journal of Computational Physics*, Vol. 49, No. 1, 1983, pp. 151–164.
- [25] Nordström, J. and Svärd, M., “Well-posed boundary conditions for the Navier–Stokes equations,” *SIAM Journal on Numerical Analysis*, Vol. 43, No. 3, 2005, pp. 1231–1255.
- [26] Kreiss, H.-O. and Lorenz, J., *Initial boundary value problems and the Navier–Stokes equations*, Academic Press, New York, 1989.
- [27] Svärd, M. and Nordström, J., “A stable high-order finite difference scheme for the compressible Navier–Stokes equations: No-slip wall boundary conditions,” *Journal of Computational Physics*, Vol. 227, No. 10, 2008, pp. 4805–4824.
- [28] Berg, J. and Nordström, J., “Stable Robin solid wall boundary conditions for the Navier–Stokes equations,” *Journal of Computational Physics*, Vol. 230, No. 19, 2011, pp. 7519–7532.
- [29] Bejan, A., *Entropy generation minimization*, CRC, Boca Raton, New York, 1st ed., 1996.
- [30] Carpenter, M., Fisher, T., Nielsen, E., and Frankel, S., “Entropy Stable Spectral Collocation Schemes for the Navier–Stokes Equations: Discontinuous Interfaces,” *SIAM Journal on Scientific Computing*, Vol. 36, No. 5, 2014, pp. B835–B867.
- [31] Del Rey Fernández, D. C., Carpenter, M. H., Fredrich, L., Winters, A. R., Gassner, G. J., Dalcin, L., and Parsani, M., “Entropy stable non-conforming discretizations with the summation-by-parts property for curvilinear coordinates,” *NASA TM-2018-*, 2018.
- [32] Parsani, M., Carpenter, M. H., and Nielsen, E. J., “Entropy stable wall boundary conditions for the compressible Navier–Stokes equations,” *NASA TM 218282*, 2014.
- [33] Dormand, J. R. and Prince, P. J., “A family of embedded Runge–Kutta formulae,” *Journal of Computational and Applied Mathematics*, Vol. 6, No. 1, 1980, pp. 19 – 26.
- [34] Söderlind, G., “Digital Filters in Adaptive Time-stepping,” *ACM Transactions on Mathematical Software*, Vol. 29, No. 1, 2003, pp. 1–26.
- [35] Söderlind, G. and Wang, L., “Adaptive time-stepping and computational stability,” *Journal of Computational and Applied Mathematics*, Vol. 185, No. 2, 2006, pp. 225–243.
- [36] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Dener, A., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., May, D. A., McInnes, L. C., Mills, R. T., Munson, T., Rupp, K., Sanan, P., Smith, B. F., Zampini, S., Zhang, H., and Zhang, H., “PETSc Users Manual,” Tech. Rep. ANL-95/11 - Revision 3.10, Argonne National Laboratory, 2018.
- [37] Knepley, M. G. and Karpeev, D. A., “Mesh Algorithms for PDE with Sieve I: Mesh Distribution,” *Scientific Programming*, Vol. 17, No. 3, 2009, pp. 215–230.
- [38] Abhyankar, S., Brown, J., Constantinescu, E. M., Ghosh, D., Smith, B. F., and Zhang, H., “PETSc/TS: A Modern Scalable ODE/DAE Solver Library,” *arXiv preprint arXiv:1806.01437*, 2018.
- [39] Geuzaine, C. and Remacle, J.-F., “Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities,” *International journal for numerical methods in engineering*, Vol. 79, No. 11, 2009, pp. 1309–1331.
- [40] Rosenhead, L., *Laminar boundary layers; an account of the development, structure, and stability of laminar boundary layers in incompressible fluids, together with a description of the associated experimental techniques.*, Oxford [England] Clarendon Press, 1st ed., 1963.
- [41] Karniadakis, G. E. and Triantafyllou, G. S., “Three-dimensional dynamics and transition to turbulence in the wake of bluff objects,” *Journal of Fluid Mechanics*, Vol. 238, 1992, pp. 1–30.
- [42] Barkley, D. and Henderson, R. D., “Three-dimensional Floquet stability analysis of the wake of a circular cylinder,” *Journal of Fluid Mechanics*, Vol. 322, 1996, pp. 215–241.
- [43] Henderson, R. D., “Nonlinear dynamics and pattern formation in turbulent wake transition,” *Journal of Fluid Mechanics*, Vol. 352, 1997, pp. 65–112.
- [44] Park, J., Kwon, K., and Choi, H., “Numerical solutions of flow past a circular cylinder at Reynolds numbers up to 160,” *KSME International Journal*, Vol. 12, No. 6, Nov 1998, pp. 1200–1205.
- [45] Williamson, C. H. K., “Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers,” *Journal of Fluid Mechanics*, Vol. 206, 1989, pp. 579–627.
- [46] Norberg, C., “An experimental investigation of the flow around a circular cylinder: influence of aspect ratio,” *Journal of Fluid Mechanics*, Vol. 258, 1994, pp. 287–316.
- [47] Zhang, H., Fey, U., Noack, B. R., Knig, M., and Eckelmann, H., “On the transition of the cylinder wake,” *Physics of Fluids*, Vol. 7, No. 4, 1995, pp. 779–794.

- [48] Williamson, C. H. K., “Three-dimensional wake transition,” *Journal of Fluid Mechanics*, Vol. 328, 1996, pp. 345–407.
- [49] Prasad, A. and Williamson, C. H. K., “The instability of the shear layer separating from a bluff body,” *Journal of Fluid Mechanics*, Vol. 333, 1997, pp. 375–402.
- [50] Henderson, R. D., “Details of the drag curve near the onset of vortex shedding,” *Physics of Fluids*, Vol. 7, No. 9, 1995, pp. 2102–2104.
- [51] Mittal, R., Dong, H., Bozkurttas, M., Najjar, F., Vargas, A., and von Loebbecke, A., “A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries,” *Journal of Computational Physics*, Vol. 227, No. 10, 2008, pp. 4825 – 4852.
- [52] JOHNSON, T. A. and PATEL, V. C., “Flow past a sphere up to a Reynolds number of 300,” *Journal of Fluid Mechanics*, Vol. 378, 1999, pp. 1970.

AppendixA. Python script for the verification of the proofs in three dimension using curvilinear grids

The following python script can be used to verify all the theorems and corresponding proofs used to construct the entropy-conservative and entropy-stable solid wall boundary conditions proposed herein. The script takes into account the general case of curvilinear grids.

```

1 import numpy as np
2 from sympy import *
3 init_printing()
4
5
6 # Define few utility routines
7
8 def zeros(*shape):
9     return np.zeros(shape, dtype=object)
10
11 def dot(a, b):
12     a = np.array(a)
13     b = np.array(b)
14     return np.dot(a, b)
15
16 def tensordot(a, b, axes=2):
17     a = np.array(a)
18     b = np.array(b)
19     return np.tensordot(a, b, axes)
20
21 def matmul(a, b):
22     a = np.array(a)
23     b = np.array(b)
24     return np.dot(a, b)
25
26 def outer(a, b):
27     a = np.array(a)
28     b = np.array(b)
29     return np.outer(a, b)
30
31
32 # Viscosity and Conductivity are functions of temperature
33 Mu = symbols('mu', real=True, positive=True, cls=Function)
34 Kappa = symbols('kappa', real=True, positive=True, cls=Function)
35
36 # All these are constants for the analysis
37 R = symbols('R', real=True, positive=True)
38 gamma = symbols('gamma', real=True, positive=True)
39 cp = R*gamma/(gamma-1)
40 cv = R/(gamma-1)
41 rho_inf = symbols('rho_inf', real=True, positive=True)

```

```

42 T_inf = symbols('T_inf', real=True, positive=True)
43
44 def pressure(rho, T):
45     return rho*R*T
46
47 def specific_energy(T):
48     return cv*T
49
50 def specific_enthalpy(T):
51     return cp*T
52
53 def specific_entropy(rho, T):
54     s = cv*log(T/T_inf) - R*log(rho/rho_inf)
55     return s
56
57 def kinetic_energy(u):
58     return dot(u,u)/2
59
60 def EntropyVars(V):
61     rho = V[0] # density
62     u = V[1:4] # velocity
63     T = V[4] # temperature
64     h = specific_enthalpy(T)
65     s = specific_entropy(rho, T)
66     k = kinetic_energy(u)
67     W = zeros(5)
68     W[0] = h/T - k/T - s
69     W[1:4] = u/T
70     W[4] = -1/T
71     return W
72
73 def PrimitiveVars(W):
74     T = -1/W[4]
75     u = W[1:4]*T
76     h = specific_enthalpy(T)
77     k = kinetic_energy(u)
78     s = h/T - k/T - W[0]
79     rho = rho_inf*exp((cv*log(T/T_inf) - s)/R)
80     V = zeros(5)
81     V[0] = rho
82     V[1:4] = u
83     V[4] = T
84     return V
85
86 def dWdV(Vs):
87     V = zeros(5)
88     V[0] = symbols('rho', real=True, positive=True)
89     V[1:4] = symbols('u0:3', real=True)
90     V[4] = symbols('T', real=True, positive=True)
91     W = EntropyVars(V)
92     mat = zeros(5, 5)
93     for a in range(5):
94         for b in range(5):
95             mat[a,b] = W[a].diff(V[b]).subs(zip(V, Vs))
96     return mat
97
98 def dVdW(Vs):
99     mat = Matrix(dWdV(Vs)).inv()
100     return np.array(mat.tolist())
101
102 def InviscidFlux(n, V):
103     rho = V[0] # density
104     u = V[1:4] # velocity
105     T = V[4] # temperature
106     p = pressure(rho, T)

```

```

107     h = specific_enthalpy(T)
108     k = kinetic_energy(u)
109     H = h + k # total enthalpy
110     fI = zeros(5)
111     fI[0] = rho*dot(n, u)
112     fI[1:4] = fI[0]*u + p*n
113     fI[4] = fI[0]*H
114     return fI
115
116 def LogAverage(a, b):
117     if a == b: return a
118     return (b-a)/log(b/a)
119
120 def EntropyConsistentFlux(n, Vl, Vr):
121     # Ismail & Roe, doi:10.1016/j.jcp.2009.04.021
122     gp1og = (gamma+1)/gamma
123     gm1og = (gamma-1)/gamma
124
125     sqrtT_l = sqrt(Vl[4])
126     sqrtT_r = sqrt(Vr[4])
127
128     z1_l = 1/sqrtT_l
129     z1_r = 1/sqrtT_r
130     z1_avg = z1_l + z1_r
131     z1_avg_inv = 1/z1_avg
132     z1_log = LogAverage(z1_l, z1_r)
133
134     z3_l = sqrtT_l * Vl[0]
135     z3_r = sqrtT_r * Vr[0]
136     z3_log = LogAverage(z3_l, z3_r)
137
138     rho_hat = z1_avg * z3_log / 2
139     p_hat = R * z1_avg_inv * (z3_l + z3_r)
140     u_hat = z1_avg_inv * (z1_l * Vl[1:4] + z1_r * Vr[1:4])
141     T_hat = 1/rho_hat * (gp1og * z3_log/z1_log + gm1og/R * p_hat)/2
142
143     h_hat = specific_enthalpy(T_hat)
144     k_hat = kinetic_energy(u_hat)
145     H_hat = h_hat + k_hat # total enthalpy
146
147     fI = zeros(5)
148     fI[0] = rho_hat * dot(n, u_hat)
149     fI[1:4] = fI[0] * u_hat + p_hat * n
150     fI[4] = fI[0] * H_hat
151     return fI
152
153 def ViscousFlux(n, V, Theta):
154     u = V[1:4] # velocity
155     T = V[4] # temperature
156     mu, kappa = Mu(T), Kappa(T) # mu and kappa depend on temperature
157     Pi = matmul(dVdW(V), Theta) # rotate gradient to primitive vars
158     grad_u = Pi[1:4,:] # gradient of velocity
159     grad_T = Pi[4,:] # # gradient of temperature
160     div_u = np.trace(grad_u) # divergence of velocity
161     I = eye(3) # identity tensor
162     epsilon = zeros(3,3) # rate-of-strain tensor
163     for i in range(3):
164         for j in range(3):
165             epsilon[i,j] += (grad_u[i,j] + grad_u[j,i])/2
166     tau = mu * (2*epsilon - Rational(2,3)*div_u*I) # Cauchy stress tensor
167     fV = zeros(5) # normal viscous flux
168     fV[0] = sympify(0)
169     fV[1:4] = dot(n, tau)
170     fV[4] = dot(n, dot(u, tau)) + kappa * dot(n, grad_T)
171     return fV

```

```

172
173 # From now on, we use the following conventions:
174 # * (-) and '_m' refers to internal state at a point within a cell boundary face
175 # * (w) and '_w' refers to given boundary values, superscript (wall) in the paper
176 # * (+) and '_p' refers to manufactured state, superscript (B) in the paper
177
178 # Unit face normal
179 nrm = symbols('n0:3', real=True)
180 nrm = np.array(nrm)
181 nrm = nrm / sqrt(dot(nrm, nrm))
182
183 # Index 'k' identifying the boundary face.
184 # If k is None, use an arbitrary face normal.
185 # Otherwise, set k from 1 to 6 to assume a regular hexahedron [0,1]^3
186 # (unit volume, unit face areas). In the paper we pick k = 1 for the sake
187 # of simplicity, in here we default to k = None to proof the general case.
188 k = None
189 if k == 1: nrm[:] = sympify([-1,0,0])
190 if k == 2: nrm[:] = sympify([+1,0,0])
191 if k == 3: nrm[:] = sympify([0,-1,0])
192 if k == 4: nrm[:] = sympify([0,+1,0])
193 if k == 5: nrm[:] = sympify([0,0,-1])
194 if k == 6: nrm[:] = sympify([0,0,+1])
195
196 # Wall velocity, satisfies dot(nrm, u_w) == 0, Remark 2.1
197 u_w = symbols('u0:3^(wall)', real=True)
198 u_w = np.array(u_w)
199 u_w = u_w - dot(u_w, nrm) * nrm
200
201 # Wall entropy flux function, Equation 20
202 g = symbols('g', real=True, cls=Function)
203 t = symbols('t', real=True) # time
204
205 # State (-) at a point in the face
206 V_m = zeros(5)
207 V_m[0] = symbols('rho', real=True, positive=True)
208 V_m[1:4] = symbols('u0:3', real=True)
209 V_m[4] = symbols('T', real=True, positive=True)
210
211 # Gradient (-) at a point in the face
212 Theta_m = Matrix(MatrixSymbol('Theta', 5, 3))
213
214 # -----
215 # --- Proof for Inviscid Terms ---
216 # -----
217
218 # Manufactured (+) state at a point in the face
219 V_p = zeros(5)
220 u_m = V_m[1:4] # velocity at (-)
221 wn = dot(u_w, nrm) # normal wall velocity, wn == 0, Remark 2.1
222 un = dot(u_m, nrm) # normal flow velocity
223 V_p[0] = V_m[0] # \
224 V_p[1] = V_m[1] + 2 * (wn - un) * nrm[0] # |
225 V_p[2] = V_m[2] + 2 * (wn - un) * nrm[1] # | Equation 31
226 V_p[3] = V_m[3] + 2 * (wn - un) * nrm[2] # |
227 V_p[4] = V_m[4] # /
228
229 W_m = EntropyVars(V_m) # entropy vars at (-)
230
231 fI_m = InviscidFlux(nrm, V_m)
232 fI_s = EntropyConsistentFlux(nrm, V_m, V_p)
233 sat_I = fI_m - fI_s # inviscid penalty term, Equation 28
234
235 rho_m, u_m = V_m[0], V_m[1:4] # density and temperature at (-)
236 Psi_m = R * rho_m * dot(nrm, u_m)

```

```

237 F_m = dot(W_m, fI_m) - Psi_m # entropy flux in LHS of Equation 26
238
239 # Inviscid contributions to the RHS of Equation 26
240 RHS_I = (
241     - F_m
242     + dot(W_m, sat_I)
243 )
244
245 RHS_I = ratsimp(RHS_I) # simplify expressions
246 RHS_I = expand(RHS_I) # trigger cancellations
247 assert RHS_I == 0      # Entropy-conservative BC! Q.E.D.
248
249
250 # -----
251 # --- Proof for Viscous Terms ---
252 # -----
253
254 # Manufactured (+) state at a point in the face
255 # Note that this way  $u^-(w) = 1/2*(u^- + u^+)$ 
256 V_p = zeros(5)
257 V_p[0] = +V_m[0] # \
258 V_p[1] = -V_m[1] + 2 * u_w[0] # |
259 V_p[2] = -V_m[2] + 2 * u_w[1] # | Equation 32
260 V_p[3] = -V_m[3] + 2 * u_w[2] # |
261 V_p[4] = +V_m[4] # /
262
263 # Manufactured (+) gradient at a point in the face
264 Pi_p = zeros(5, 3)
265 Pi_m = matmul(dVdW(V_m), Theta_m) # Equation 36
266 Pi_p[0,:] = -Pi_m[0,:] # \
267 Pi_p[1,:] = +Pi_m[1,:] # |
268 Pi_p[2,:] = +Pi_m[2,:] # | Equation 37
269 Pi_p[3,:] = +Pi_m[3,:] # |
270 Pi_p[4,:] = -Pi_m[4,:] # /
271 Theta_p = matmul(dWdV(V_p), Pi_p) # Equation 38
272
273 W_m = EntropyVars(V_m) # entropy vars at (-)
274 W_p = EntropyVars(V_p) # entropy vars at (+)
275
276 fV_m = ViscousFlux(nrm, V_m, Theta_m)
277 fV_p = ViscousFlux(nrm, V_p, Theta_p) # Equation 33
278 sat_V = -(fV_m-fV_p)/2 # viscous penalty term, Equation 29
279
280 # Source heat flux corresponding to g(t)
281 src_L = zeros(5)
282 T_m = V_m[4] # temperature at (-)
283 src_L[4] = -T_m * g(t) # Equation 35
284
285 # Volume viscous flux to contract with gradient Equation 23b
286 fV = zeros(5,3)
287 fV[:,0] = ViscousFlux([1,0,0], V_m, Theta_m)
288 fV[:,1] = ViscousFlux([0,1,0], V_m, Theta_m)
289 fV[:,2] = ViscousFlux([0,0,1], V_m, Theta_m)
290 # Gradient penalty term
291 sat_Theta = -Rational(1,2) * outer(W_m - W_p, nrm) # Equation 30
292
293 # Viscous contributions to the RHS of Equation 26
294 RHS_V = (
295     + dot(W_m, fV_m)
296     + dot(W_m, sat_V)
297     + dot(W_m, src_L)
298     + tensordot(fV, sat_Theta)
299 )
300
301 RHS_V = ratsimp(RHS_V) # simplify expressions

```



```

302 RHS_V = expand(RHS_V) # trigger cancellations
303 assert RHS_V == g(t) # Entropy-conservative BC! Q.E.D.
304
305
306 # -----
307 # --- Proof for IP dissipation ---
308 # -----
309
310 # This constant is the IP strength factor
311 beta = symbols('beta', real=True, positive=True)
312
313 # Proof using viscous flux evaluations and normal entropy jumps,
314 # this is the practical way to implement the IP dissipation.
315 # This proof holds for any normal vector and wall velocity. Note
316 # that in the paper, and for the sake of simplicity, we express the
317 # IP dissipation in terms of the # viscous C_jj, j=1,2,3 and we
318 # assume normal vectors aligned with the Cartesian coordinate directions.
319
320 dWn = outer(W_m - W_p, nrm)
321 fm = ViscousFlux(nrm, V_m, dWn)
322 fp = ViscousFlux(nrm, V_p, dWn)
323 M = -beta * (fm + fp)/2 # Equation 40
324 RHS_IP = dot(W_m, M) # Equation 41
325
326 RHS_IP = expand(RHS_IP)
327 RHS_IP = simplify(RHS_IP)
328
329 # Now we check that the IP term is non-positive
330 u_m = V_m[1:4]
331 T_m = V_m[4]
332 du = u_m - u_w
333 N = outer(nrm, nrm) # rank 1, symmetric, semi-PD matrix
334 RHS_IP_explicit = (
335     -2*beta*Mu(T_m)/(3*T_m) * ( # this factor is negative
336         + dot(du, matmul(N, du)) # this term is non-negative
337         + 3*dot(du,du)*dot(nrm,nrm) # this term is non-negative
338     ) # then this is non-positive and thus dissipative
339 )
340 RHS_IP_explicit = expand(RHS_IP_explicit)
341 RHS_IP_explicit = simplify(RHS_IP_explicit)
342 assert expand(RHS_IP - RHS_IP_explicit) == 0 # Q.E.D
343
344 # Proof using the C_jj, j=1,2,3 viscous matrices,
345 # this is the usual form we use in proofs.
346 # This proof DOES NOT HOLD for any normal vector and wall velocity,
347 # it assumes that the normal vector is aligned with the Cartesian
348 # coordinate directions and dot(nrm, u_w) == 0.
349
350 def C_11(V):
351     rho, u1, u2, u3, T = V
352     mu, kappa = Mu(T), Kappa(T)
353     C = zeros(5,5)
354     C[1,1] = Rational(4,3)*T*mu
355     C[1,4] = Rational(4,3)*T*mu*u1
356     C[2,2] = T*mu
357     C[2,4] = T*mu*u2
358     C[3,3] = T*mu
359     C[3,4] = T*mu*u3
360     C[4,1] = C[1,4]
361     C[4,2] = C[2,4]
362     C[4,3] = C[3,4]
363     C[4,4] = T**2*kappa+Rational(1,3)*T*mu*(4*u1**2+3*u2**2+3*u3**2)
364     return C
365
366 def C_22(V):

```

```

367     rho,u1,u2,u3,T = V
368     mu, kappa = Mu(T), Kappa(T)
369     C = np.zeros([5,5], dtype=object)
370     C[1,1] = T*mu
371     C[1,4] = T*mu*u1
372     C[2,2] = Rational(4,3)*T*mu
373     C[2,4] = Rational(4,3)*T*mu*u2
374     C[3,3] = T*mu
375     C[3,4] = T*mu*u3
376     C[4,1] = C[1,4]
377     C[4,2] = C[2,4]
378     C[4,3] = C[3,4]
379     C[4,4] = T**2*kappa+Rational(1,3)*T*mu*(3*u1**2+4*u2**2+3*u3**2)
380     return C
381
382 def C_33(V):
383     rho,u1,u2,u3,T = V
384     mu, kappa = Mu(T), Kappa(T)
385     C = zeros(5, 5)
386     C[1,1] = T*mu
387     C[1,4] = T*mu*u1
388     C[2,2] = T*mu
389     C[2,4] = T*mu*u2
390     C[3,3] = Rational(4,3)*T*mu
391     C[3,4] = Rational(4,3)*T*mu*u3
392     C[4,1] = C[1,4]
393     C[4,2] = C[2,4]
394     C[4,3] = C[3,4]
395     C[4,4] = T**2*kappa+Rational(1,3)*T*mu*(3*u1**2+3*u2**2+4*u3**2)
396     return C
397
398 if k is not None:
399
400     if k in (1, 2):
401         C_m = C_11(V_m)
402         C_p = C_11(V_p)
403     if k in (3, 4):
404         C_m = C_22(V_m)
405         C_p = C_22(V_p)
406     if k in (5, 6):
407         C_m = C_33(V_m)
408         C_p = C_33(V_p)
409
410     L = -beta * (C_m + C_p)/2 # Equation 39
411     M = dot(L, W_m - W_p) # Equation 40
412     RHS_IP_v2 = dot(W_m, M) # Equation 41
413     RHS_IP_v2 = expand(RHS_IP_v2)
414     RHS_IP_v2 = simplify(RHS_IP_v2)
415
416     assert RHS_IP == RHS_IP_v2

```

AppendixB. FORTRAN code for the implementation of the boundary conditions on curvilinear grids

In this appendix we provide a simple but yet general FORTRAN implementation of the proposed entropy stable wall boundary conditions on curvilinear grids. The following piece of code receives as input the primitive variables, V , and outputs the primitive variables of the ghost state. It is supposed to be called for each collocated point lying on the wall boundary face.

```

1 module WallBC
2

```

```

3  integer, parameter :: d      = 3           ! dimension in {2,3}
4  integer, parameter :: c      = d+2        ! components
5  integer, parameter :: I1     = 1           ! index of density
6  integer, parameter :: I2(d)  = (/ (i,i=1,d) /)+1 ! index of velocity
7  integer, parameter :: I3     = c           ! index of temperature
8
9  integer, parameter :: dp = selected_real_kind(15, 307)
10
11 contains
12
13 pure subroutine WallBCInviscid(Snrm,u_wall,V_m,V_p)
14   real(kind=dp), intent(in) :: Snrm(d) ! normal vector
15   real(kind=dp), intent(in) :: u_wall(d) ! wall velocity vector
16   real(kind=dp), intent(in) :: V_m(c) ! primitive vars
17   real(kind=dp), intent(out) :: V_p(c) ! primitive vars
18   real(kind=dp) :: nrm(d) ! unit normal
19   real(kind=dp) :: wn ! normal velocity of wall
20   real(kind=dp) :: un ! normal velocity of fluid
21
22   nrm = Snrm/sqrt(dot_product(Snrm,Snrm))
23   wn = dot_product(nrm,u_wall)
24   un = dot_product(nrm,V_m(I2))
25
26   V_p(I1) = V_m(I1)
27   V_p(I2) = V_m(I2) + 2 * (wn - un) * nrm
28   V_p(I3) = V_m(I3)
29
30 end subroutine WallBCInviscid
31
32 pure subroutine WallBCAdiabatic(u_wall,V_m,dWdX_m,V_p,dVdX_p)
33   real(kind=dp), intent(in) :: u_wall(d) ! wall velocity vector
34   real(kind=dp), intent(in) :: V_m(c) ! primitive vars
35   real(kind=dp), intent(in) :: dWdX_m(c,d) ! gradient of entropy vars
36   real(kind=dp), intent(out) :: V_p(c) ! primitive vars
37   real(kind=dp), intent(out) :: dWdX_p(c,d) ! gradient of entropy vars
38   real(kind=dp) :: dVdX_m(c,d) ! gradient of primitive vars
39   real(kind=dp) :: dVdX_p(c,d) ! gradient of primitive vars
40
41   V_p(I1) = +V_m(I1)
42   V_p(I2) = -V_m(I2) + 2 * u_wall
43   V_p(I3) = +V_m(I3)
44
45   dVdX_m = matmul(dVdW(V_m),dWdX_m)
46   dVdX_p(I1,:) = -dVdX_m(I1,:)
47   dVdX_p(I2,:) = +dVdX_m(I2,:)
48   dVdX_p(I3,:) = -dVdX_m(I3,:)
49   dWdX_p = matmul(dWdV(V_p),dVdX_p)
50
51 end subroutine WallBCAdiabatic
52
53 pure function dWdV(V)
54   real(kind=dp), intent(in) :: V(c)
55   real(kind=dp) :: dWdV(c,c)
56   dWdV = ... ! Fill-in with proper values: exercise left to the reader
57 end function dWdV
58
59 pure function dVdW(V)
60   real(kind=dp), intent(in) :: V(c)
61   real(kind=dp) :: dVdW(c,c)
62   dVdW = ... ! Fill-in with proper values: exercise left to the reader
63 end function dVdW
64
65 end module WallBC

```