Journal of Computational Physics 505 (2024) 112914

Contents lists available at ScienceDirect

Journal of Computational Physics

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

Efficient implementation of complex equations of state in a high-order framework

E. Mantecca [a,*], A. Colombo [b], A. Ghidoni [a], G. Noventa [a]

[a] University of Brescia, Department of Mechanical and Industrial Engineering, Via Branze 36, Brescia, 25123, Italy
[b] University of Bergamo, Department of Engineering and Applied Sciences, Viale Marconi 5, Dalmine, 24044, Italy

ARTICLE INFO

Keywords:
Discontinuous Galerkin
Non-ideal EoSs
RANS equations
NICFD

ABSTRACT

In the last decades, the interest in non-ideal compressible fluid dynamics (NICFD) flows and high-order accurate numerical methods, such as discontinuous Galerkin (dG), has quickly grown. In fact, advanced simulation capabilities are of paramount importance to develop new sustainable technologies with higher efficiency and low environmental impact, and to decrease the use of expensive experimental facilities. Nowadays however, the coupling of accurate Computational Fluid Dynamics (CFD) solvers with sophisticated thermodynamic models has been investigated mainly in the finite volume (FV) framework. Moreover, the solution of complex equations of state (EoS) has a too high computational cost for real-life use in industry, which is often overcome with the look-up table (LuT) interpolation approach. LuTs seem to be more suitable for FV solvers rather than high-order or finite element ones, since the interpolation error introduced may deteriorate the higher accuracy provided by these numerical methods. The novelty of this work lies in the assessment of an efficient implementation for real gas simulations in a high-order solver, by resorting to structured LuTs and automatic differentiation (AD). The proposed implementation is assessed with the computation of the inviscid and turbulent flow around a NACA0012. This approach guarantees an overhead of the computational cost around 17% with respect to an ideal gas solver with manually derived jacobian matrix.

1. Introduction

In the last two decades, the interest in non-ideal compressible fluid dynamics (NICFD) flows has quickly grown, due to the engineering applications related to the conversion of renewable and waste energy sources, e.g., organic Rankine cycles (ORCs), supercritical carbon dioxide cycle power plants, combustors operating with supercritical fluids, gas liquefaction or carbon capture and storage (CCS) implants, and heat-pumps.

In this context, advanced CFD capabilities are of paramount importance to develop new sustainable technologies with higher efficiency and low environmental impact, and to decrease the use of expensive experimental facilities. The main problems that a CFD solver has to deal with in dense working conditions are the large compressibility effects that characterise a non-ideal gas, but also the possible presence of non-classical gas dynamic phenomena such as expansion shocks or compression fans.

These observations are pushing a continuous development in numerical models and schemes. The dense gas thermodynamic features strongly differ from those of an ideal gas and numerics based on an ideal gas behaviour may lose their consistency if strong

* Corresponding author.
E-mail address: e.mantecca@unibs.it (E. Mantecca).

https://doi.org/10.1016/j.jcp.2024.112914
Received 6 July 2023; Received in revised form 27 February 2024; Accepted 3 March 2024

Available online 8 March 2024

Nomenclature

AD	Automatic Differentiation
CCS	Carbon capture and storage
CFD	Computational fluid dynamics
CFL	Courant–Friedrichs–Lewy
dG	discontinuous Galerkin
EoSs	Equations of State
FD	Finite differences
FEM	Finite element methods
IG	Ideal gas
LuT	Look-up-Table
MD	Manual differentiation
NICFD	Non-ideal compressible fluid dynamics
NNPS	Nearest neighbour point search
ORCs	Organic Rankine cycles
PR	Peng-Robinson
RANS	Reynolds-Averaged Navier-Stokes equations
SW	Span-Wagner

variations of the thermodynamic properties are present. This is the case, for example, of boundary conditions [1] or convective [2] flux, or in general, of every numerical procedure that explicitly depends on the thermodynamic model used by the solver. For all these reasons a generalisation is required to work in such conditions, and the only effective solution is the use of a non-ideal gas model. Different thermodynamic models can be adopted, characterised by different levels of accuracy, but also different computational costs. Nowadays, the use of Equations of State (EoSs) that are cubic in the density, such as the Peng-Robinson [3] (PR) model, can represent a good compromise between the level of approximation of the non-ideal gas behaviour, and the computational cost. However, more accurate and costly models, e.g., multiparameter EoS, such as the Span-Wagner [4] (SW) model, can be mandatory for some applications.

The coupling of an accurate Computational Fluid Dynamics (CFD) solver with sophisticated thermodynamic models has been investigated, mainly in the finite volume framework [1,5]. However, the increasing computational power and the higher accuracy expected by the design offices worldwide, motivate the recent interest in higher-order accurate solvers, such as discontinuous Galerkin (dG) methods, which are particularly attractive for their geometrical flexibility, simple implementation of h/p adaptive algorithms [6,7], and compact stencil. Their drawback with respect to standard FV solvers is the higher computation cost, which prevents a widespread application. For this reason, the computational efficiency of a non-ideal gas models should be maximised in a high-order framework, obviously without spoiling the higher accuracy.

The present work aims to investigate the main aspects to perform efficient non-ideal gas simulations with an implicit dG solver [8,9], whose prediction capabilities have been recently extended to non-ideal gas [10,11]. In general, the non-ideal extension of a CFD code introduces an overhead in terms of computing time caused by the computation of thermodynamic properties/derivatives and thermodynamic dependent contributions to the jacobian matrix of the residual. A classical approach to reduce the computational cost associated to the EoS evaluation is the use of the look-up tables (LuTs) [12–16]. In a LuT method thermodynamic properties/derivatives are tabulated on a discrete set of values, which are used to interpolate values on the complete thermodynamic space. Normally, different tables are built for properties and derivatives, or properties tables are used to compute also derivative with a finite difference method. The former approach stores a huge number of tables that can increase excessively the memory consumption, while the latter can create convergence problem when complex transonic flow field is computed. Moreover, structured or unstructured tables can be used: the former allows a fast interpolation, while the latter allows a better representation of the saturation line and the possibility to exploit local adaptation algorithms. As the main goal is the non ideal simulation in the vapour zone maximising the computational efficiency, structured LuTs are adopted, where only properties tables are generated and derivatives are computed with an ad-hoc interpolant on the same properties tables. The computing time reduction with respect to EoS evaluation and the influence of the LuT interpolation error on the dG accuracy are investigated for inviscid and turbulent test cases. Moreover, two approaches are investigated for the efficient calculation of all the thermodynamic dependent contributions to the jacobian matrix of the residual, i.e., the Automatic Differentiation (AD) [17] and the Finite Differences (FD). Their computational efficiency are compared also with implicit computations based on a manually derived jacobian for the ideal gas. The main aspects investigated in this work are summarized in Fig. 1.

The paper is organized as follows: the main features of the solver are outlined in Sec. 2, where governing equations, thermodynamic models, spatial and temporal discretization methods are described. In Sec. 3.1 the use of LuTs to approximate thermodynamic modelling is discussed and demonstrated. Section 3.2 describes possible solutions to compute the jacobian matrix, and an efficient choice based on AD is proposed, while Sec. 4 shows the performance of the proposed approach for the simulation of the inviscid and turbulent flow around a NACA0012 airfoil, using both the PR and SW models. Finally, concluding remarks are reported in Sec. 5. In Appendix A, examples of AD code generated by TAPENADE are reported in their original and modified (to improve the computational efficiency) version.

2. Implicit non-ideal gas dG solver

This section outlines the main features of the solver, with particular emphasis on the governing equations (Sec. 2.1), i.e., the RANS equations, the thermodynamic models (Sec. 2.2) and the spatial and temporal discretization methods (Sec. 2.3). For sake of briefness the Euler equations are here not reported.

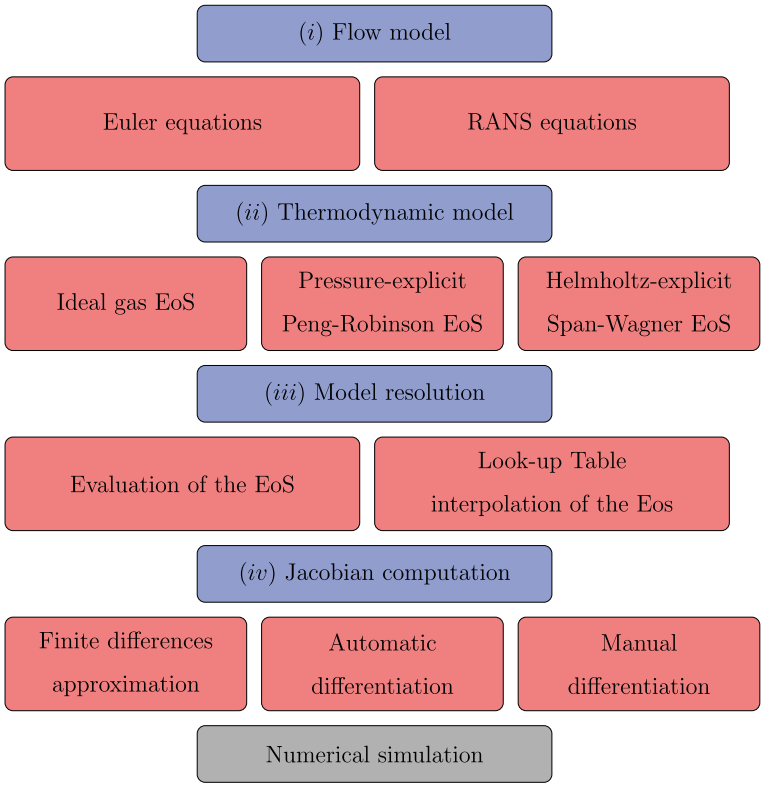

Fig. 1. Schematic representation of the building blocks investigated in this work to efficiently perform non-ideal gas simulations. The different blocks implemented in the solver concern: (*i*) the flow model (Euler and RANS equations), (*ii*) the thermodynamic model (ideal gas, PR, and SW EoS), (*iii*) the resolution of the EoSs (direct evaluation of the EoS and LuT), and (*iv*) the jacobian computation (finite differences, automatic and manual differentiation). Each block can be used independently with the exception of the manually derived jacobian, which is available only for the ideal gas model.

2.1. Governing equations

The compressible Reynolds-Averaged Navier-Stokes (RANS) equations supplemented by the k-$\widetilde{\omega}$ turbulence model [8,9,18] can be written as

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_j) = 0, \tag{1}$$

$$\frac{\partial}{\partial t}\left(\rho u_i\right) + \frac{\partial}{\partial x_j}(\rho u_j u_i) = -\frac{\partial p}{\partial x_i} + \frac{\partial \widehat{\tau}_{ji}}{\partial x_j}, \tag{2}$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}(\rho u_j H) = \frac{\partial}{\partial x_j}\left[u_i \widehat{\tau}_{ij} - \widehat{q}_j\right] - \tau_{ij}\frac{\partial u_i}{\partial x_j} + \beta^* \rho \overline{k} e^{\widetilde{\omega}_r}, \tag{3}$$

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_j}(\rho u_j k) = \frac{\partial}{\partial x_j}\left[(\mu + \sigma^* \overline{\mu}_t)\frac{\partial k}{\partial x_j}\right] + \tau_{ij}\frac{\partial u_i}{\partial x_j} - \beta^* \rho \overline{k} e^{\widetilde{\omega}_r}, \tag{4}$$

$$\frac{\partial}{\partial t}(\rho \widetilde{\omega}) + \frac{\partial}{\partial x_j}(\rho u_j \widetilde{\omega}) = \frac{\partial}{\partial x_j}\left[(\mu + \sigma \overline{\mu}_t)\frac{\partial \widetilde{\omega}}{\partial x_j}\right] + \frac{\alpha}{\overline{k}}\tau_{ij}\frac{\partial u_i}{\partial x_j} - \beta \rho e^{\widetilde{\omega}_r} + \left(\mu + \sigma \overline{\mu}_t\right)\frac{\partial \widetilde{\omega}}{\partial x_k}\frac{\partial \widetilde{\omega}}{\partial x_k}, \tag{5}$$

where u_i is the flow velocity, p the pressure, ρ and μ the density and dynamic viscosity, E and H the total mass-specific internal energy and enthalpy, τ_{ij} and $\widehat{\tau}_{ij}$ the turbulent and overall shear stress tensors, $\widehat{q}_j$ the overall Fourier's conductive heat flux, μ_t the turbulent dynamic viscosity, $\overline{k}$ the limited mass-specific turbulent kinetic energy, and $\widetilde{\omega} = \log(\omega)$ is the logarithm of the specific dissipation rate ω. The different terms are

$$E = \hat{e} + u_k u_k/2, \quad H = h + u_k u_k/2,$$

$$\tau_{ij} = 2\overline{\mu}_t\left(S_{ij} - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\rho\overline{k}\delta_{ij}, \quad \widehat{\tau}_{ij} = 2\mu\left(S_{ij} - \frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) + \tau_{ij},$$

$$\widehat{q}_j = -\left(\lambda + \lambda_t\right)\frac{\partial T}{\partial x_j}, \quad \overline{\mu}_t = \alpha^* \rho \overline{k} e^{-\widetilde{\omega}_r}, \quad \overline{k} = \max(0, k).$$

α, β, β^*, σ, σ^* are the closure parameters of the turbulence model [19], while λ is the laminar thermal conductivity and λ_t the turbulent one (calculated using a constant turbulent Prandtl number). The value $\widetilde{\omega}_r$ satisfies the realizability condition for the turbulent stresses [9].

The compact form of Eqs. (1)-(5) is

$$\mathbf{P}(\mathbf{w})\frac{\partial \mathbf{w}}{\partial t} + \nabla \cdot \mathbf{F}_c(\mathbf{w}) + \nabla \cdot \mathbf{F}_v(\mathbf{w}, \nabla \mathbf{w}) + \mathbf{s}(\mathbf{w}, \nabla \mathbf{w}) = \mathbf{0}, \tag{6}$$

where $\mathbf{w}$ is the vector of the unknown variables, $\mathbf{F}_c$ and $\mathbf{F}_v$ are the convective and viscous fluxes, while $\mathbf{s}$ is the source terms vector. The matrix $\mathbf{P}(\mathbf{w})$ allows the change of variables from the standard conservative set to the primitive one $\mathbf{w} = \left[\widetilde{p}, u_1, u_2, u_3, \widetilde{T}, k, \widetilde{\omega}\right]^T$, where $\widetilde{p} = \log(p)$ and $\widetilde{T} = \log(T)$ are used to enhance the solver's robustness [9].

2.2. Thermodynamic models

The gas models used in this work are the cubic pressure-explicit PR EoS and the multiparameter Helmholtz-explicit SW EoS. The first one reads

$$p(\rho, T) = \frac{\rho R^* T}{(1 - \rho B)} - \frac{a\rho^2 \alpha^2(T)}{(1 + 2\rho b - \rho^2 b^2)}, \tag{7}$$

where the coefficients $a = 0.45724(R^* T_{cr})^2 / p_{cr}$ and $b = 0.0778 R^* T_{cr} / p_{cr}$ are functions of the critical pressure and temperature of the fluid, p_{cr} and T_{cr}, but also of the specific gas constant $R^* = R/m_M$. $R = 8314.463 J/(kmol K)$ is the universal gas constant and m_M the molecular weight of the fluid. The function $\alpha(T) = 1 + h\left(1 - \sqrt{T/T_{cr}}\right)$ is instead a function of $h = 0.37464 + 1.54226\theta - 0.26992\theta^2$, which is in turn a function of the acentric factor of the fluid θ. Thanks to an approximation, usually a polynomial, of the IG contribution to the isochoric specific heat $c_v^0(T)$, the IG internal energy and entropy are defined as

$$e^0(T) = e_0 + \int_{T_0}^{T} c_v^0(\eta) d\eta, \quad s^0(\rho, T) = s_0 + \int_{T_0}^{T} \frac{c_v^0}{\eta} d\eta - R^* \log\left(\frac{\rho}{\rho_0}\right), \tag{8}$$

where (ρ_0, T_0, e_0, s_0) represents an arbitrary reference state. Then, by using Eq. (7), the non-ideal gas internal energy and entropy are calculated as

$$e(\rho, T) = e^0(T) + \int_0^{\rho} \frac{1}{\xi^2}\left[p - T\left(\frac{\partial p}{\partial T}\right)_{\xi}\right] d\xi, \tag{9}$$

$$s(\rho, T) = s^0(\rho, T) + \int_0^{\rho} \frac{1}{\xi^2}\left[\xi R^* - \left(\frac{\partial p}{\partial T}\right)_{\xi}\right] d\xi. \tag{10}$$

The SW EoS is instead given in the form

$$\frac{a(\rho, T)}{R^* T} = \frac{e(\rho, T)}{R^* T} - \frac{s(\rho, T)}{R^*} = \psi(\delta, \tau) = \psi^0(\delta, \tau) + \psi^r(\delta, \tau), \tag{11}$$

where $\psi(\delta, \tau)$ is the non-dimensional free Helmholtz energy of the fluid, expressed as the sum of an ideal, $\psi^0(\delta, \tau)$, and of a non-ideal, $\psi^r(\delta, \tau)$, gas contribution. They are both functions of a non-dimensional density and temperature $\delta = \rho/\rho_{cr}$ and $\tau = T_{cr}/T$, where ρ_{cr} is the critical density of the fluid. The dimensional IG contribution can be calculated as $a^0(\rho, T) = e^0 - T s^0$ with Eq. (8), and an approximation of the IG contribution to the isochoric specific heat is

$$c_v^0(T) = \sum_{i=1}^{n_{pol}} \left(c_{1,i} T^{c_{2,i}}\right) + \sum_{i=1}^{n_{exp}} \left[c_{1,i} \frac{(c_{2,i}/T)^2 e^{-c_{2,i}/T}}{\left(1 - e^{-c_{2,i}/T}\right)^2}\right] + \sum_{i=1}^{n_{hyc}} \left[\frac{c_{1,i}/T^2}{(\cosh(c_{2,i}/T^2))^2}\right] + \sum_{i=1}^{n_{hys}} \left[\frac{c_{1,i}/T^2}{(\sinh(c_{2,i}/T^2))^2}\right] - R^*, \tag{12}$$

where the coefficients $(c_{1,i}, c_{2,i})$ are parameters of the fluid. While the non-dimensional non-ideal gas contribution of Eq. (11) is

$$\begin{aligned}\psi^r(\delta, \tau) = &\sum_{i=1}^{n_{pol}} \left(c_{1,i} \delta^{c_{2,i}} \tau^{c_{3,i}}\right) + \sum_{i=1}^{n_{exp}} \left(c_{1,i} \delta^{c_{2,i}} \tau^{c_{3,i}} e^{-\delta^{c_{4,i}}}\right) \\ &+ \sum_{i=1}^{n_{ga1}} \left\{c_{1,i} \delta^{c_{2,i}} \tau^{c_{3,i}} e^{\left[-c_{4,i}(\delta - c_{5,i})^2 - c_{6,i}(\tau - c_{7,i})^2\right]}\right\} \\ &+ \sum_{i=1}^{n_{ga2}} \left\{c_{1,i} \delta \Delta_i^{c_{2,i}} e^{\left[-c_{7,i}(\delta - 1)^2 - c_{8,i}(\tau - 1)^2\right]}\right\},\end{aligned} \tag{13}$$

with $\Delta_i = \left\{(1-\tau) + c_{3,i}\left[(\delta-1)^2\right]^{1/(2c_{4,i})}\right\}^2 + c_{5,i}\left[(\delta-1)^2\right]^{c_{6,i}}$. The non-ideal gas pressure and entropy can be computed with the Maxwell relations as

$$p(\rho,T) = \rho^2\left(\frac{\partial a}{\partial \rho}\right)_T, \quad s(\rho,T) = -\left(\frac{\partial a}{\partial T}\right)_\rho. \tag{14}$$

For every model, the enthalpy is calculated as $h(\rho,T) = e + p/\rho$, while the non-ideal gas isochoric and isobaric specific heats as $c_v(\rho,T) = \partial e/\partial T$ and $c_p(\rho,T) = c_v + (T/\rho^2)[(\partial p/\partial T)^2/(\partial p/\partial \rho)]$. The speed of sound is instead $c(\rho,T) = \sqrt{(c_p/c_v)(\partial p/\partial \rho)}$. Both the PR and SW EoSs, use the generalised multiparameter correlations from Chung et al. [20] for the calculation of the laminar transport properties $\mu(\rho,T)$ and $\lambda(\rho,T)$.

2.3. dG framework

The spatial discretization of the governing equations is performed through a high-order implicit dG solver. The continuous variational weak formulation of the problem with m unknowns is obtained in two basic steps, which are: (i) the multiplication of Eq. (6) for an arbitrary smooth test function $\mathbf{v} = \{v_1, \dots, v_m\}$ and (ii) the integration by parts over the physical domain Ω using the divergence theorem. A geometrical approximation Ω_h of Ω is then triangulated in a set of n_e arbitrary shaped non-overlapping elements $\mathcal{T}_h$ with faces $\mathcal{F}_h$. For each element $K \in \mathcal{T}_h$, a set $\{\phi\}$ of N_{dof} orthonormal polynomial basis functions is defined hierarchically, and for $j = 1, \dots, m$ and $l = 1, \dots, N_{dof}$ their convex combination with unknown degrees of freedom $w_{h,j} = \phi_l W_{j,l}$ provides the finite element approximation of $\mathbf{w}$ in every K. The set $\{\phi\}$ is such that $\{\phi\} \subseteq \mathbf{V}_h = [\mathbb{P}_d^q(\mathcal{T}_h)]^m$, and it also represents a finite dimension space of test functions $\mathbf{v}_h \in \mathbf{V}_h$, which are needed to perform a discrete variational weak formulation of the problem in every K. By summing over all the elements in $\mathcal{T}_h$, for $i = 1, \dots, N_{dof}$ and $k = 1, \dots, n_e$, the dG spatial discretization of Eq. (6) consists therefore in seeking at every time instant the elements of $\mathbf{W} \in \mathbb{R}^{n_e \times m \times N_{dof}^K}$ such that

$$\begin{aligned}
&\sum_{K\in\mathcal{T}_h}\int_K \phi_i P_{j,k}\left(\mathbf{w}_h\right)\phi_l \frac{dW_{k,l}}{dt} d\mathbf{x} \\
&- \sum_{K\in\mathcal{T}_h}\int_K \frac{\partial \phi_i}{\partial x_n} F_{j,n}\left(\mathbf{w}_h, \nabla_h \mathbf{w}_h + \mathbf{r}\left(\left[\!\left[\mathbf{w}_h\right]\!\right]\right)\right) d\mathbf{x} \\
&+ \sum_{F\in\mathcal{F}_h}\int_F \left[\!\left[\phi_i\right]\!\right]_n \widehat{F}_{j,n}\left(\mathbf{w}_h^{\pm}, \left(\nabla_h \mathbf{w}_h + \eta_F \mathbf{r}_F\left(\left[\!\left[\mathbf{w}_h\right]\!\right]\right)\right)^{\pm}\right) d\sigma \\
&+ \sum_{K\in\mathcal{T}_h}\int_K \phi_i s_j\left(\mathbf{w}_h, \nabla_h \mathbf{w}_h + \mathbf{r}\left(\left[\!\left[\mathbf{w}_h\right]\!\right]\right)\right) d\mathbf{x} = 0.
\end{aligned} \tag{15}$$

The functional approximation is however discontinuous at every $F \in \mathcal{F}_h$, hence the sum of the convective and viscous flux functions $\mathbf{F}$ is substituted by the numerical flux $\widehat{\mathbf{F}}$ in the contour integrals, to guarantee the discretization consistency. The convective part of $\widehat{\mathbf{F}}$ is calculated with the Vinokur-Montagné approximate Riemann solver [2], generalized to non-ideal gases [21], while the viscous part is discretized with the BR2 scheme [6]. A shock-capturing term is also added to the left-hand side of Eq. (15) to suppress spurious oscillations of the solution at high polynomial degrees. The term acts by introducing an artificial viscosity along the direction of the pressure gradient, but only where and when a shock sensor is triggered [7,18].

After a computation of the integrals in Eq. (15) with exact Gauss quadrature rules, a system of ordinary differential equations (ODEs) in time is obtained in the form

$$\mathbf{M_P}(\mathbf{W})\frac{d\mathbf{W}}{dt} + \mathbf{R}(\mathbf{W}) = \mathbf{0}, \tag{16}$$

where $\mathbf{R}(\mathbf{W})$ is the global vector of the residuals and $\mathbf{M_P}(\mathbf{W})$ is the global mass matrix coming from the first integral. For steady simulations, Eq. (16) is solved by the Linearized Backward Euler (LBE) scheme as

$$\left[\frac{\mathbf{M_P}(\mathbf{W}^n)}{\Delta t} + \frac{\partial \mathbf{R}(\mathbf{W}^n)}{\partial \mathbf{W}}\right]\left(\mathbf{W}^{n+1} - \mathbf{W}^n\right) = -\mathbf{R}(\mathbf{W}^n), \tag{17}$$

where $\partial \mathbf{R}(\mathbf{W}^n)/\partial \mathbf{W}$ is the jacobian matrix of the residual at the time instant n and a pseudo-transient continuation strategy is adopted [8]. In this case, an exponential CFL law, which is a function of the residuals norms, makes possible to use progressively higher values of Δt, and Eq. (17) is reduced to a Newton-Rhapson method. The resulting nonlinear systems are solved at every time-step iteratively, with a restarted version of the Generalized Minimal RESidual (GMRES) Krylov's subspace-type method, as available in the linear algebra library PETSc [22]. For unsteady simulations, the time integration of Eq. (16) is efficiently carried out by means of linearly implicit Rosenbrock-type Runge-Kutta schemes [9]. In particular, the ROS3PL scheme, i.e., a third order L-stable one-step method with four stages is employed in this work [23–25]. Rosenbrock-type schemes require only the solution of linear systems, and the evaluation of the jacobian matrix only once per time-step. These features, coupled with their high accuracy, are remarkably favourable for an efficient integration in time.

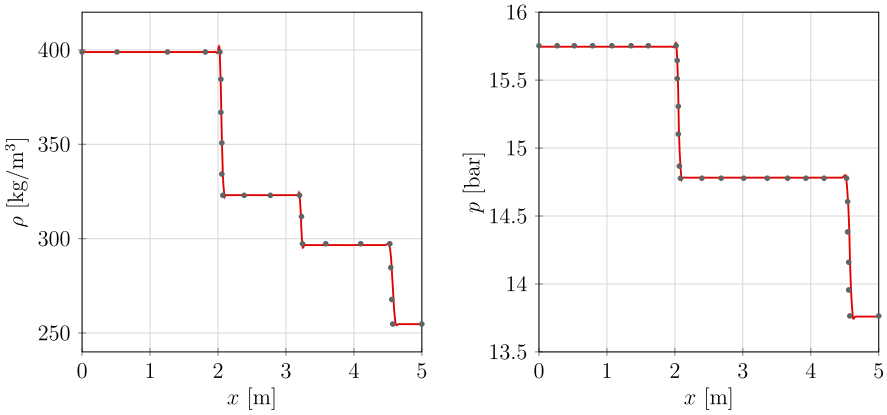

Fig. 2. Shock tube. Density (left) and pressure (right) profiles at time $t = 29.46 \times 10^{-3}$ s. EoS(PR) $\mathbb{P}^2$ ——, Guardone et al. [26] ●. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

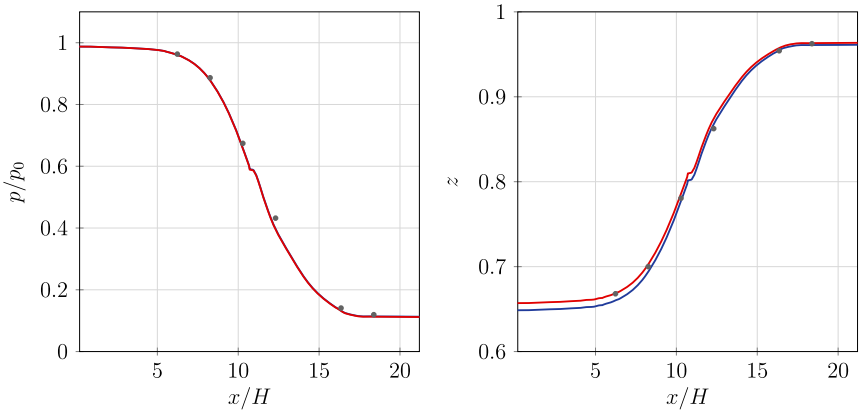

Fig. 3. Supersonic nozzle. Static-to-total pressure ratio (left) and compressibility factor (right) profiles on the nozzle axis. EoS(PR) $\mathbb{P}^4$ ——, EoS(SW) $\mathbb{P}^4$ ——, Spinelli et al. exp. [27] ●. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

2.4. Validation of the implemented EoSs for inviscid flows

The implementation of the PR EoS is validated computing the one dimensional Riemann problem proposed by Guardone et al. [26] for the PP10 fluid ($C_{13}F_{22}$). The domain is a tube with length 5 m, which is discretized with 400 uniform elements. Initially, the tube is separated in two regions by a diaphragm located at $x = 3$ m, characterized by the following initial conditions: $\{p, \rho, T_{\text{EoS(PR)}}\}_{left} = \{15.746\ bar, 398.883\ kg/m^3, 632.01\ K\}$ and $\{p, \rho, T_{\text{EoS(PR)}}\}_{right} = \{13.760\ bar, 254.712\ kg/m^3, 634.98\ K\}$ (the temperature is calculated with the direct evaluation of the PR EoS). At time $t = 0$ s the diaphragm is removed and a right-running compression shock wave, a contact discontinuity and a left-running rarefaction shock wave are generated. The unsteady simulation is carried out until $t = 29.46 \times 10^{-3}$ s with 400 time-steps, i.e., $\Delta t = 7.37 \times 10^{-5}$. Fig. 2 compares the density and pressure profiles along the tube with EoS(PR) and $\mathbb{P}^2$ solution approximation with a reference solution, showing a satisfactory agreement. To validate SW EoS implementation, the supersonic non-ideal nozzle of Spinelli et al. [27] is considered. The working fluid is the linear siloxane MDM ($C_8H_{24}O_2Si_3$) and the operating conditions are: inlet total pressure $p_0 = 9.02\,bar$, inlet total temperature $T_0 = 269\,°C$ and total-to-static expansion ratio $\beta = 9.16$. The employed grid is structured and consists of 100 quadrilateral elements with quadratic edges, and a $\mathbb{P}^4$ solution approximation is used. Fig. 3 compares the static-to-total pressure ratio (left) and the compressibility factor $z = p/(\rho R^* T)$ (right) profiles on the nozzle axis with respect to the experimental measurements reported in [27] for the M2.0H test case. A satisfactory agreement is obtained, thus demonstrating the correct implementation of the EoSs.

3. Efficient implementation of the thermodynamics models

For an efficient calculation of the thermodynamic properties, the use of LuTs is discussed in Sec. 3.1, while for an efficient calculation of the jacobian, the use of automatic differentiation (AD) is discussed in Sec. 3.2. The performance of LuT and AD are compared with the direct evaluation of the EoS and the finite difference approach, respectively.

3.1. Approximated thermodynamic modelling

In order to decrease the computational cost for the calculation of the thermodynamic properties without spoiling the level of accuracy, LuTs are widely used in literature [12–15,28,29,16]. The generation and use of a LuT consists of three basic steps: (*i*) the calculation of a discrete set of values for the thermodynamic properties using an EoS on the nodes of the mesh and the storing in a data structure, (*ii*) a search algorithm to retrieve neighbours of a specific set of thermodynamic coordinates, and (*iii*) an interpolation formula that uses a combination of these retrieved values to calculate the properties in the point of interest.

The generation of the table is carried out only once, in the pre-processing phase, and this greatly reduces the cost of the simulation with respect to the solution of EoSs. In particular, the computational cost for the evaluation of the properties is independent from the EoS, providing increased computational savings with the complexity of the thermodynamic model. Instead, the cost of the LuT is mainly affected by the efficiency of the search algorithm. For this reason, the interpolation formula must be carefully chosen in order to guarantee a sufficient accuracy for the numerical simulation without being too costly.

Many authors in literature proposed different LuT approaches, both based on structured and unstructured meshes. In particular, Boncinelli et al. [12], Dumbser et al. [13], Pini et al. [14], and De Lorenzo et al. [15] based the LuT on structured meshes, while Rubino et al. [16] on unstructured meshes. Wilhelmsen et al. [29] and Xia et al. [28] proposed LuTs based on an adaptive Cartesian mesh in order to increase the accuracy and decrease the number of discretization points for the thermodynamic region of interest. Rubino et al. [16] spotlighted that the use of quadrilateral meshes, in combination with local refinement, can lead to local discontinuities and poor interpolation accuracy of properties in the proximity of smooth boundaries [30,28]. This can especially occur for properties reconstruction or for interpolation close to the vapour–liquid saturation line. For this reason other authors based LuTs on adaptive unstructured triangular meshes, which allow the local mesh refinements to increase the accuracy in case of strong property variations, such as in proximity of the vapour–liquid critical point and/or of the vapour–liquid saturation line. The use of adaptive unstructured meshes can be time consuming for the high computational cost of the search and interpolation algorithms. Rubino et al. [16] proposed a search algorithm based on a trapezoidal map of the tabulated region to reduce the computational cost.

In summary, the great advantage of the unstructured meshing approach resides in the possibility to obtain local refinements of any grade and everywhere in the thermodynamic region of interest, with a potentially reduction of memory requirements. On the other hand, as it emerges from the literature, the nearest neighbour point search (NNPS) problem on an unstructured dataset has a cost that is always proportional to the dimension of the dataset itself. Nowadays some of the most efficient searching algorithms for unstructured grids are still the octree and the kd-tree ones [31,32], which are characterized by a complexity cost $O(\log N)$, where N is the total number of points in the datasets. Moreover, some algorithms may experience a performance degradation when points are too close to each other [33]. This behaviour is called curse of dimensionality, and in the worst cases the complexity cost is comparable to a linear search, i.e., $O(N)$.

Due to these limits, a structured approach is adopted in this work, since the main objective of the LuT implementation is to guarantee an almost constant computational saving. In this context the NNPS problem can in fact be easily solved with a complexity cost of $O(1)$, and without any performance degradation in the limit of extremely close data points. Furthermore, without an increment of the computational cost with the number of nodes, LuTs could be generated with very high accuracy to deal with regions of the thermodynamic plane with strong properties variations.

Fig. 4 shows the flow chart that summaries the generation of a LuT. The first step for the generation is the choice of the input parameters, i.e., the working fluid, the EoS and the upper and lower limits $\{x_{min}, x_{max}\}$ and $\{y_{min}, y_{max}\}$ for a x-y thermodynamic plane. In addition, the number of points on each axis n_{p_x} and n_{p_y} must be specified, together with a number of additional nodes at their ends n_{o_x} and n_{o_y} which work as ghost points to preserve the interpolation formula accuracy also at LuT boundaries. At the coordinates of the nodes the generic thermodynamic property $z_{ij}(x_i, y_j)$ is calculated with an EoS, and the accuracy of each new table is evaluated also with a Root-Mean-Square (RMS) error estimator [16]. This estimator compares the interpolated values of the properties with the exact solution of the EoS, avoiding to query the nodes where the error is zero by definition. A comparison between the retrieved RMS error and an user-defined tolerance allows to decrease or increase automatically the number of nodes in each direction and for each LuT before starting the simulations.

Three different thermodynamic properties are used in pairs as independent variables, to generate a total of 18 LuT. In particular, the independent variables are the density ρ, temperature T and pressure p, while the output properties are: the internal energy $e(p,T)$ or $e(\rho,T)$, the enthalpy $h(p,T)$ or $h(\rho,p)$, the speed of sound $c(p,T)$, the entropy $s(p,T)$ or $s(\rho,p)$, the compressibility factor $z(p,T)$, the fundamental derivative of gas dynamics $\Gamma(\rho,T)$, the isobaric and isochoric specific heat capacity $c_p(\rho,T)$ and $c_v(\rho,T)$, the temperature $T(\rho,p)$ or $T(\rho,e)$, the pressure $p(\rho,T)$ or $p(\rho,e)$, the density $\rho(p,T)$, the dynamic viscosity $\mu(\rho,T)$ and the thermal conductivity $\lambda(\rho,T)$. The values of the upper and lower limits ρ_{min}, ρ_{max}, T_{min}, T_{max} and p_{min}, p_{max} are defined by the user, while the limits of e are calculated as $e_{min} = e(p_{max}, T_{min})$ and $e_{max} = e(p_{min}, T_{max})$. Even if in principle also the density bounds could be computed as $\rho_{min} = e(p_{min}, T_{max})$ and $\rho_{max} = e(p_{max}, T_{min})$, this choice is not here used to avoid an excessive spread of the density axis that would require to use a high number of nodes. This problem is not so frequent with the internal energy, whose bounding values would be also trickier to estimate manually.

For each table the spacing between neighbouring nodes in each direction is given by

$$\Delta x = \frac{x_{max} - x_{min}}{n_{p_x} + 2n_{o_x} - 1}, \quad \Delta y = \frac{y_{max} - y_{min}}{n_{p_y} + 2n_{o_y} - 1}, \tag{18}$$

where n_{p_x} and n_{p_y} can be different between the various tables, while n_{o_x} and n_{o_y} are the same. The coordinates of each node of the table (x_i, y_j), with $1 \le i \le n_x$ and $1 \le j \le n_y$, are given by

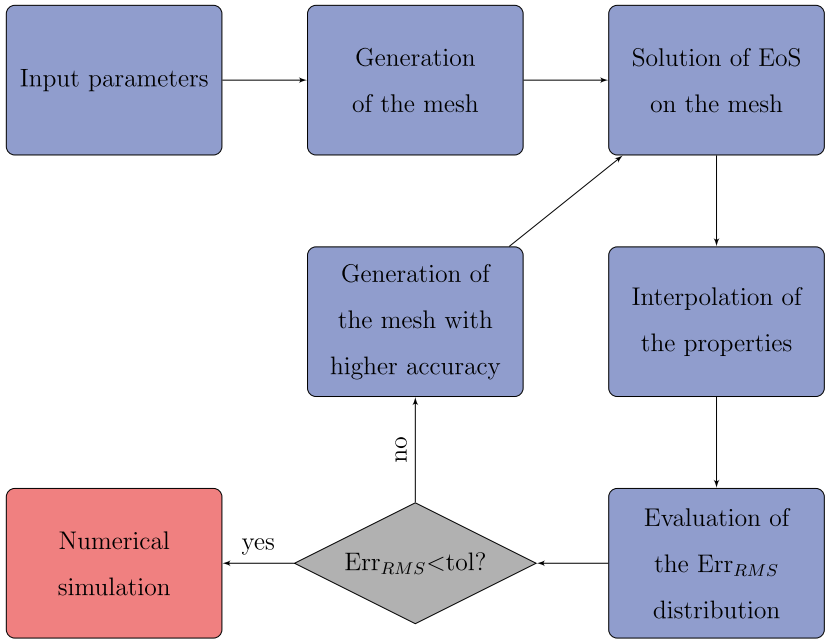

Fig. 4. Flow chart that summaries the generation of a LuT. The input parameters are the fluid, the EoS, the number of nodes of the structured mesh n_x, n_y, and the limits of the temperature T, the pressure p, and the density ρ in terms of the minimum and maximum values.

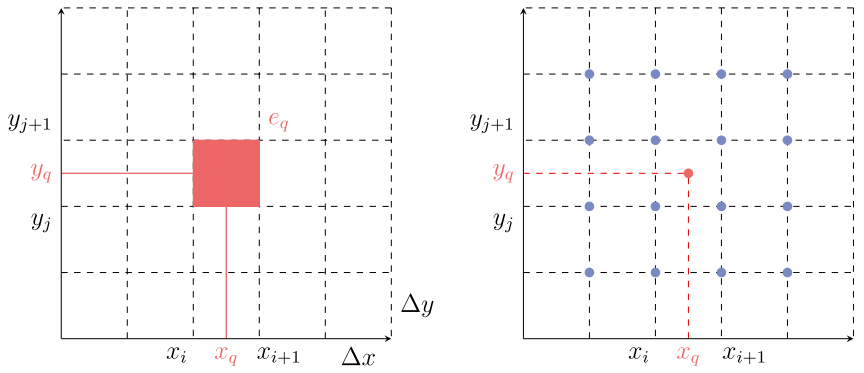

Fig. 5. Schematic representation of an element containing a query point (left) and of the nodal values needed to perform a bicubic interpolation for a query point (right).

$$x_i = x_{min} + (i-1)\Delta x, \quad y_j = y_{min} + (j-1)\Delta y, \tag{19}$$

where $n_x = n_{p_x} + 2n_{o_x}$ and $n_y = n_{p_y} + 2n_{o_y}$. The search and interpolation algorithms are based on the structured nature of LuTs. In particular, given a point to query with coordinates (x_q, y_q), inside the element of interest e_q, with nodes (x_i, y_j), (x_{i+1}, y_j), (x_{i+1}, y_{j+1}) and (x_i, y_{j+1}), the search algorithm allows to find these nodes and the neighbouring nodes for the interpolation formula (Fig. 5). In fact, thanks to the structured mesh, the coordinates of the points satisfy the relations $x_i \leqslant x_q \leqslant x_{i+1}$ and $y_j \leqslant y_q \leqslant y_{j+1}$, and the indices i and j can be calculated as

$$i = int\left(\frac{x_q - x_{min}}{x_{max} - x_{min}}\right) + 1, \quad j = int\left(\frac{y_q - y_{min}}{y_{max} - y_{min}}\right) + 1. \tag{20}$$

As expected, the searching cost does not depend on the number of points.

The interpolation formula allows then to calculate the value of a generic thermodynamic property in the point of interest $z(x_q, y_q)$, by combining values from the neighbouring nodes. The interpolation formula here used is a bicubic polynomial in x and y and is based on the local coordinates system (ξ, η) inside the element e_q, given by

$$\xi(x) = \frac{x - x_i}{\Delta x}, \quad \eta(y) = \frac{y - y_j}{\Delta y}, \tag{21}$$

where $0 \leqslant \xi \leqslant 1$ and $0 \leqslant \eta \leqslant 1$, while $r(\xi(x_q), \eta(y_q)) = z(x_q, y_q)$ is the polynomial approximation of the thermodynamic property in the point of interest. The bicubic interpolation function $r(\xi, \eta)$ is given by

$$r(\xi, \eta) = A(\xi) S B^T(\eta), \tag{22}$$

where

$$A(\xi) = [f_1(\xi), f_2(\xi), f_3(\xi), f_4(\xi)], \quad B(\eta) = [g_1(\eta), g_2(\eta), g_3(\eta), g_4(\eta)],$$
$$f_1(\xi) = 2\xi^3 - 3\xi^2 + 1, \quad f_2(\xi) = -2\xi^3 + 3\xi^2,$$
$$f_3(\xi) = \xi^3 - 2\xi^2 + \xi, \quad f_4(\xi) = \xi^3 - \xi^2,$$
$$g_i(\eta) = f_i(\eta),$$

and S is the matrix

$$S \equiv \begin{pmatrix} r(0,0) & r(0,1) & \tilde{r}_\eta(0,0) & \tilde{r}_\eta(0,1) \\ r(1,0) & r(1,1) & \tilde{r}_\eta(1,0) & \tilde{r}_\eta(1,1) \\ \tilde{r}_\xi(0,0) & \tilde{r}_\xi(0,1) & \tilde{r}_{\xi\eta}(0,0) & \tilde{r}_{\xi\eta}(0,1) \\ \tilde{r}_\xi(1,0) & \tilde{r}_\xi(1,1) & \tilde{r}_{\xi\eta}(1,0) & \tilde{r}_{\xi\eta}(1,1) \end{pmatrix}, \tag{23}$$

which includes the interpolation function and its derivatives for the generic element e (Fig. 5). The derivatives $\tilde{r}_\xi \simeq \partial r/\partial\xi$, $\tilde{r}_\eta \simeq \partial r/\partial\eta$ and $\tilde{r}_{\xi\eta} \simeq \partial^2 r/\partial\xi\partial\eta \simeq \tilde{r}_{\eta\xi}$ are calculated with centred FD approximations. As an example, $\tilde{r}_\xi(0,0)$ is computed as $\tilde{r}_\xi(0,0) = [z(x_{i+1}, y_j) - z(x_{i-1}, y_j)]/2$ since $r(0,0)$ is $z(x_i, y_j)$.

During a simulation, many numerical procedures require the values of some derivative of the tabulated thermodynamic properties, which in turn should be tabulated or approximated with a sufficient accuracy. Since the implicit solver uses potentially all the pure and mixed derivatives of the thermodynamic properties up to the second order, the generation of a LuT for every derivative would result in unbearable memory requirements. A possible solution is the use of a FD approximation for each derivative based on the LuT of the corresponding properties. However, this approach needs at least two evaluations of the interpolation function, which increases the computational cost (two times the cost of a property evaluation) and the truncation error. For these reasons, a different approach is used in this work, which is based on the exact derivatives of the interpolation function $r[\xi(x), \eta(y)]$. With a direct application of the differentiation chain rule on Eq. (22), the first and second order derivatives are

$$\frac{\partial r}{\partial x}[\xi(x), \eta(y)] = \frac{\partial r}{\partial \xi}\frac{d\xi}{dx}, \tag{24}$$
$$\frac{\partial r}{\partial y}[\xi(x), \eta(y)] = \frac{\partial r}{\partial \eta}\frac{d\eta}{dy}, \tag{25}$$
$$\frac{\partial^2 r}{\partial x^2}[\xi(x), \eta(y)] = \frac{\partial^2 r}{\partial \xi^2}\left(\frac{d\xi}{dx}\right)^2 + \frac{\partial r}{\partial \xi}\frac{d^2\xi}{dx^2}, \tag{26}$$
$$\frac{\partial^2 r}{\partial y^2}[\xi(x), \eta(y)] = \frac{\partial^2 r}{\partial \eta^2}\left(\frac{d\eta}{dy}\right)^2 + \frac{\partial r}{\partial \eta}\frac{d^2\eta}{dy^2}, \tag{27}$$
$$\frac{\partial^2 r}{\partial x \partial y}[\xi(x), \eta(y)] = \frac{\partial}{\partial y}\left(\frac{\partial r}{\partial \xi}\frac{d\xi}{dx}\right) = \frac{\partial^2 r}{\partial \xi \partial \eta}\frac{d\xi}{dx}\frac{d\eta}{dy} + \frac{\partial r}{\partial \xi}\frac{d^2\xi}{dxdy}. \tag{28}$$

In Eqs. (24)-(28), the derivatives of the local coordinates ξ and η are trivial, since $d\xi/dx = 1/\Delta x$, $d\eta/dy = 1/\Delta y$, $d^2\xi/dx^2 = d^2\eta/dy^2 = d^2\xi/dxdy = 0$. For this reason, the second addend on the right hand side of Eqs. (26)-(28) is always zero. The derivatives of the interpolation function with respect to the local coordinates are obtained by differentiating Eq. (22) with respect to ξ and η as

$$\frac{\partial r}{\partial \xi} = \frac{dA}{d\xi} S B^T, \quad \frac{\partial r}{\partial \eta} = A S \left[\frac{dB}{d\eta}\right]^T, \tag{29}$$
$$\frac{\partial^2 r}{\partial \xi^2} = \frac{d^2 A}{d\xi^2} S B^T, \quad \frac{\partial^2 r}{\partial \eta^2} = A S \left[\frac{d^2 B}{d\eta^2}\right]^T, \quad \frac{\partial^2 r}{\partial \xi \partial \eta} = \frac{dA}{d\xi} S \left[\frac{dB}{d\eta}\right]^T, \tag{30}$$

where the first and second order derivatives of the vectors $A(\xi)$ and $B(\eta)$ contain just the first and the second order derivatives of the functions $f_i(\xi)$ and $g_i(\eta)$:

$$\frac{df_1}{d\xi} = 6\xi^2 - 6\xi, \quad \frac{d^2 f_1}{d\xi^2} = 12\xi - 6, \quad \frac{df_2}{d\xi} = -6\xi^2 + 6\xi, \quad \frac{d^2 f_2}{d\xi^2} = -12\xi + 6,$$
$$\frac{df_3}{d\xi} = 3\xi^2 - 4\xi + 1, \quad \frac{d^2 f_3}{d\xi^2} = 6\xi - 4, \quad \frac{df_4}{d\xi} = 3\xi^2 - 2\xi, \quad \frac{d^2 f_4}{d\xi^2} = 6\xi - 2,$$
$$\frac{dg_i}{d\eta}(\eta) = \frac{df_i}{d\xi}(\eta), \quad \frac{d^2 g_i}{d\eta^2}(\eta) = \frac{d^2 f_i}{d\xi^2}(\eta).$$

Table 1
Relative memory storage, maximum error Err_{MAX}, RMS error Err_{RMS} and percentage difference of the computational cost with respect to EoS(IG) for the calculation of the thermodynamics properties of $n_q = 10000$ random points using different LuTs(SW).

LuT nodes	Relative memory storage	Err_{MAX}	Err_{RMS}	$\Delta t_{LuT(SW)-EoS(IG)}$
15 625	1.000	5.200E-02	1.784E-03	37.895%
62 500	3.957	2.717E-02	4.745E-04	35.417%
250 000	15.359	4.750E-03	7.468E-05	36.458%
1 000 000	60.500	6.247E-04	3.453E-06	38.636%

Table 2
Maximum Err_{MAX} and RMS Err_{RMS} errors for different LuTs(IG) sizes, Ringleb flow case.

LuT nodes	10^4	10^6	10^8
Err_{MAX}	3.28E-03	4.59E-06	3.91E-09
Err_{RMS}	3.01E-04	4.27E-07	3.85E-10

With this implementation, the derivatives of any tabulated thermodynamic property generated by the bicubic interpolation function have an exact and consistent approximation. Moreover, the cost for the calculation of a thermodynamic property and its derivative is comparable. Notice that, bicubic polynomial interpolants are necessary to guarantee the existence of derivatives up to the second order, which are represented by bilinear maps.

In order to assess the accuracy and memory storage of LuTs a test is performed, where $n_q = 10000$ thermodynamic points are evaluated for a heavy cyclic siloxane, i.e., D5. The following ranges are considered for density, temperature and pressure: $0.034 < \rho/\rho_{cr} < 0.889$, $1.009 < T/T_{cr} < 1.090$, and $0.087 < p/p_{cr} < 1.748$. The choice of slightly supercritical temperatures guarantees that the considered zone of the thermodynamic plane lies in a single-phase region. A compressibility factor range of $0.229 < z < 0.981$ is guaranteed, which is representative of strong continuous variations in the properties. Temperature variations are kept smaller than the ones of density and pressure since this also happens in many applications. The exact and approximated values of the thermodynamics properties are compared by calculating the maximum and RMS error, Err_{MAX} and Err_{RMS}, on a number n_q of random points (x_k, y_k) as

$$Err_{RMS} = \max_{k=1,\dots,n_q} \left\{ z(x_k, y_k) - r[\xi(x_k), \eta(y_k)] \right\} \tag{31}$$

$$Err_{MAX} = \sum_{k=1}^{n_q} \sqrt{\frac{\left\{ z(x_k, y_k) - r[\xi(x_k), \eta(y_k)] \right\}^2}{n_q}}. \tag{32}$$

Table 1 reports the interpolation errors, Err_{MAX} and Err_{RMS}, the memory storage, and the computational cost for different LuTs(SW) nodes. The memory storage is normalized with respect to the lowest one, which is $440kb$ (ASCII format), while the computational cost is shown as the difference with respect to the ideal gas (IG) EoS, because IG computational time is independent from the number of evaluations. The LuT relative memory storage increases almost linearly with the number of nodes and, as expected, the interpolation cost does not depend appreciably on the LuT size.

To highlight the effects of the LuTs interpolation error on the solution of a simulation, a spatial convergence analysis is here carried out using the IG model on the inviscid isentropic test case of Ringleb [34]. The analytical solution for this problem is well known, only for the IG model, and can be derived using the hodograph method [35]. Table 2 shows the interpolation errors, Err_{MAX} and Err_{RMS} for different LuTs(IG) with different number of nodes. Fig. 6 shows the L^2 norm of the pressure errors for simulations with different (i) sizes of the LuT, (ii) number of elements n_e for the mesh, and (iii) solution approximations. The number of nodes of the LuT, if enough, does not affect the convergence order of the spatial discretization. However, if coarser LuTs are used, e.g., with 10^4 and 10^6, a lower limit appears on the error of the solution. This limit is lower, around two order of magnitude, than Err_{MAX} and Err_{RMS} interpolation errors of each LuT. From a practical point of view, LuTs with a number of nodes from 10^4 to 10^6 guarantee an accuracy comparable to the EoS up to $\mathbb{P}^4$. To fully exploit higher degrees of discretization, a refinement of the LuTs is mandatory. As a reference for the reader, the pressure, temperature and density spacing for the medium size LuT (10^6 nodes) is $\{dp, dT, d\rho\} = \{2.50\ kPa, 0.65\,°C, 1.50\ kg/m^3\}$. The error plots for temperature, x-component and y-component of the velocity show identical trends and are not reported here for sake of briefness.

3.2. Efficient calculation of jacobian matrices

As shown in Sec. 2.2, the implementation of a non ideal gas model is usually a task that requires the computation of a great number of derivatives. In fact, if an implicit time integration is used, the jacobian matrix of the residual vector with respect to the solution must be computed at each time-step. In particular, given a generic function $\mathbf{f}(\mathbf{x}) = [f_1, \dots, f_m]^T : \mathbb{R}^n \to \mathbb{R}^m$ with $\mathbf{x} = [x_1, \dots, x_n]^T$, the jacobian matrix of $\mathbf{f}(\mathbf{x})$ can be written as

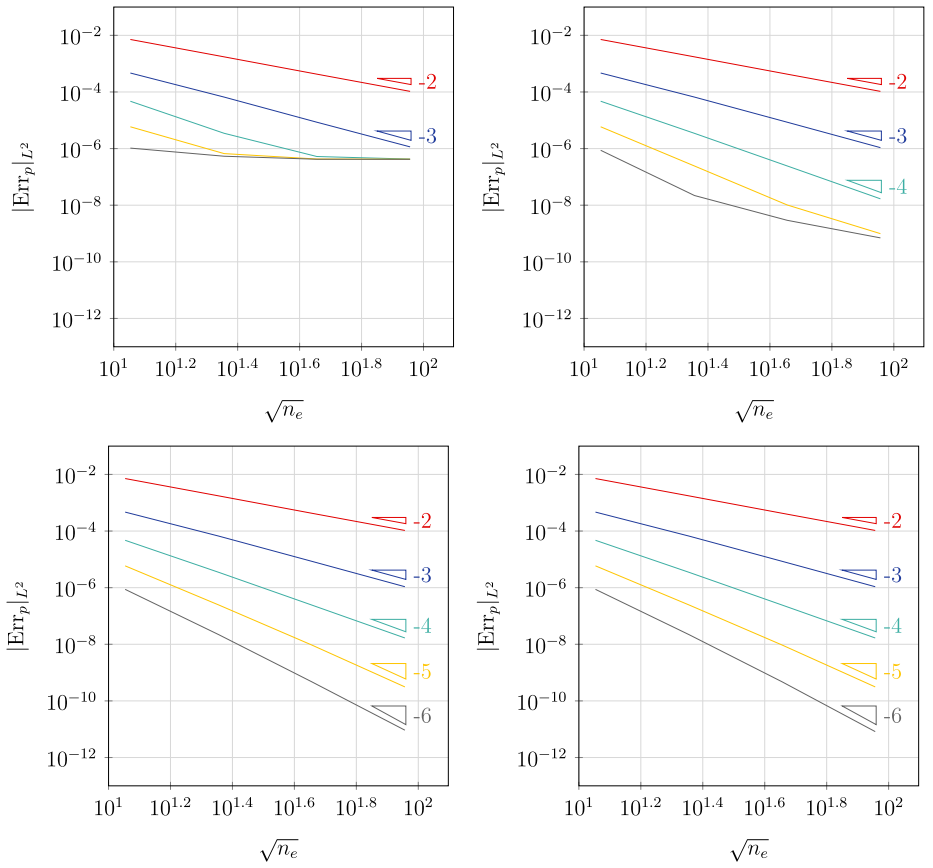

Fig. 6. L^2 norm of the pressure error with respect to the analytical solution as a function of the number of elements of the mesh n_e in the Ringleb flow case: LuT(IG) with 10^4 nodes (top left), LuT(IG) with 10^6 nodes (top right), LuT(IG) with 10^8 nodes (bottom left) and EoS(IG) (bottom right). $\mathbb{P}^1$ ——, $\mathbb{P}^2$ ——, $\mathbb{P}^3$ ——, $\mathbb{P}^4$ ——, $\mathbb{P}^5$ —— solution approximation. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \partial f_1/\partial x_1 & \dots & \partial f_1/\partial x_n \\ \vdots & \ddots & \vdots \\ \partial f_m/\partial x_1 & \dots & \partial f_m/\partial x_n \end{bmatrix}. \tag{33}$$

A possible approach to compute the jacobian matrix is the numerical differentiation through the FD approximation, where $\mathbf{J}$ is generated column by column. In this case, the function $\mathbf{f}$ is numerically differentiated n times with a formula of variable order of accuracy, which requires some evaluations of $\mathbf{f}$. The first order forward FD approximation is the simplest and uses the lowest number of evaluations:

$$\begin{aligned} \frac{\partial \mathbf{f}}{\partial x_1} &= \frac{\mathbf{f}(x_1 + h_1, \dots, x_n) - \mathbf{f}(x_1, \dots, x_n)}{h_1} + o(h_1), \\ &\vdots \\ \frac{\partial \mathbf{f}}{\partial x_n} &= \frac{\mathbf{f}(x_1, \dots, x_n + h_n) - \mathbf{f}(x_1, \dots, x_n)}{h_n} + o(h_n), \end{aligned} \tag{34}$$

where h_i for $i = i, \dots, n$ are small positive increments. The main drawback of the FD approach is the computational cost, since every approximation requires more than one evaluation of the function to derive. Equation (34) requires $n + 1$ evaluations of $\mathbf{f}$, second order formulae may require up to $2n + 1$ evaluations and so on for higher orders. Moreover, the derivatives are not exact, and their accuracy may deteriorate in presence of strong gradients or when the step size h_i is too small or too large.

A second approach to calculate $\partial \mathbf{f}/\partial \mathbf{x}$ is the manual differentiation (MD). This procedure is the most rigorous one, since it requires to derive and implement manually the mathematical expressions of the derivatives of every procedure involved in the calculation of the jacobian, by accounting explicitly for any dependence of the residual from the solution values. This approach guarantees the exactness of the computed derivatives and the possibility to minimise the computational cost by omitting any unnecessary or repeated calculation. On the other hand, the time required for such derivation may be too long, because very complex functions may arise when real gas models are adopted. Also, even if correct expressions are obtained, their implementation is prone to errors.

Another approach, that can be viewed as a combination of the previous ones, is AD. AD algorithms interpret some selected outputs of a main program, i.e., the components of the function $\mathbf{f}$, as functions that depends on all code lines which come before them. In

Table 3
Critical properties, molecular weight and acentric factor of D5 fluid [37].

p_{cr} [MPa]	T_{cr} [K]	ρ_{cr} [kg/m^3]	M_m [g/mol]	θ [adim]
1.161	619.235	292.571	370.770	0.6658

Table 4
Coefficients for the ideal gas isobaric specific heat of D5 fluid [37].

c_0 [J/(kgK^1)]	c_1 [J/(kgK^2)]	c_2 [J/(kgK^3)]	c_3 [J/(kgK^4)]
-9.412E+01	+5.021E+00	-3.785E-03	+1.349E-06

this way, the chain rule of differentiation can be applied from the first to the last line, generating the derivatives of all the outputs in the end. The chain can be travelled either from top to bottom or from bottom to top, generating two different approaches called direct or reverse differentiation. The main advantage of AD is the exactness of the computed derivatives, since they are generated with repeated basic analytical differentiation rules. The implementation impact is therefore dramatically lower with respect to MD, but the cost of a derivative generated with a standard AD approach can be comparable with a first order FD. Appendix A shows examples of AD codes generated by TAPENADE for the calculation of a generic function $\mathbf{f}(\mathbf{x})$ through the subroutine `fun`, using also an external procedure `g`, to better understand how an AD-generated derivative is and how it can be made more efficient.

4. Results

In this section, the implementation of LuTs and AD-derived jacobian matrices of the residual are discussed, and their computational efficiency is compared with the direct evaluation of the EoS and FD-derived jacobian matrices. In particular, the LuTs accuracy is investigated in Sec. 4.1, while the efficiency of the approach LuT+AD is compared with EoS+FD in Sec. 4.2. In the following, this nomenclature is adopted: *i*) EoS(name of the model) is the direct evaluation of the EoS "name of the model", *ii*) LuT(name of the model) is the LuT built with the EoS "name of the model".

The assessment of the proposed implementations is performed by computing a test case from [36], the subsonic flow around a NACA0012 airfoil for the fluid D5, i.e., decamethylcyclopentasiloxane ($C_{10}H_{30}O_5Si_5$). The fluid critical pressure, temperature and density, as well as its molecular weight and acentric factor, are reported in Table 3, while Table 4 reports the coefficients used for the polynomial approximation of its ideal gas contribution to the isobaric specific heat as a function of the temperature. The set of coefficients for the residual part of the free Helmholtz energy function used by the SW model are given in [37].

A freestream pressure $p_\infty = 0.985 p_{cr}$, density $\rho_\infty = 0.622 \rho_{cr}$ and velocity $U_\infty = 30.151\ m/s$ are used, with a reference axial chord length $c_{ax} = 2.346 \times 10^{-2}\ m$. In these conditions, EoS(SW) provides a freestream Reynolds number $Re_\infty = (\rho_\infty U_\infty c_{ax})/\mu_\infty = 6 \times 10^6$ and Mach number $Ma_\infty = U_\infty / c_\infty = 0.630$, with a compressibility factor $z_\infty = 0.451$. The angle of attack is $\alpha = 2°$. Both the Euler and RANS equations are used to test non-ideal gas approaches.

4.1. LuTs accuracy

Three different meshes are used with $n_e = 5\,288$, $8\,087$, and $15\,484$ quadrilateral elements with quadratic edges with $\mathbb{P}^5$ and $\mathbb{P}^3$ maximum solution approximation for the Euler and RANS equations, respectively (see Fig. 7). Three different LuTs are used, characterized by 10^4 (coarse), 0.25×10^6 (medium), and 6.25×10^6 (fine) nodes, and the resulting spacings of pressure, temperature and density for each table are:

- $\{dp, dT, d\rho\}_{coarse} = \{5.00\ kPa, 0.50\,°C, 1.750\ kg/m^3\}$,
- $\{dp, dT, d\rho\}_{medium} = \{1.00\ kPa, 0.10\,°C, 0.35\ kg/m^3\}$,
- $\{dp, dT, d\rho\}_{fine} = \{0.20\ kPa, 0.02\,°C, 0.07\ kg/m^3\}$.

The following bounds are used to generate the LuTs: $0.410 < \rho/\rho_{cr} < 0.990$, $0.950 < T/T_{cr} < 1.050$ and $0.785 < p/p_{cr} < 1.250$. From hereafter, unless otherwise specified, the use of the finest LuT is always assumed. Fig. 8 shows contours of the Mach number for the inviscid (left) and turbulent (right) case with EoS(SW) on the coarse mesh and with the maximum solution approximation.

Initially, the effect of the LuT discretization error on the spatial accuracy is investigated for inviscid and turbulent simulations. Fig. 9 shows the RMS of the entropy error $s - s_\infty$ for inviscid simulations with different (*i*) size of the LuT, (*ii*) number of mesh elements n_e, and (*iii*) solution approximation. As expected, a satisfactory spatial convergence is achieved using EoS(PR). For example, the last measured convergence rate for $\mathbb{P}^5$ is $O(h^{6.047})$, where h is the mesh size and 6 the expected theoretical order, while it deteriorates rapidly when the coarse LuT is employed. From a practical point of view, the medium size LuT is sufficient to preserve spatial convergence up to $\mathbb{P}^3$, which is a reasonable target for more complex flow configurations. Respect to the Ringleb test case, presented in Sec. 3.1, the use of a non-ideal thermodynamic model in a dense gas state requires LuTs with smaller spacings for pressure, temperature and density. The increase of the LuT storage size for dense states can be mitigated with a proper choice of the upper and lower bounds of pressure, temperature and density. Same trends are verified also for the turbulent test case.

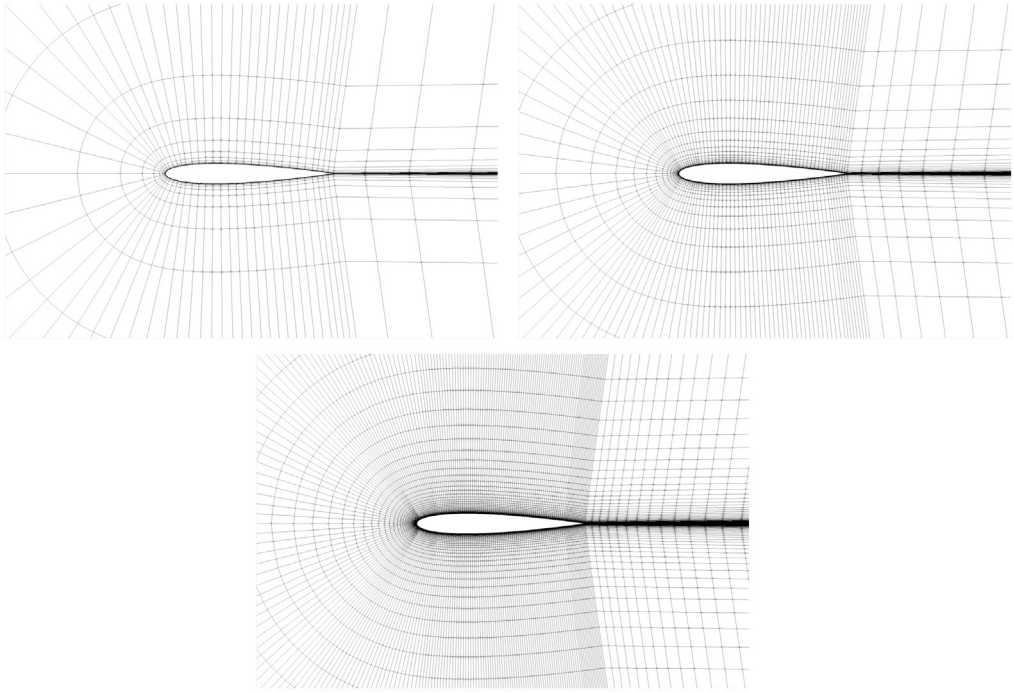

Fig. 7. Detail of the computational meshes, coarse (top left), medium (top right), and fine (bottom), around the airfoil with $n_e = 5\,288$, $8\,087$, and $15\,484$ quadrilateral elements with quadratic edges.

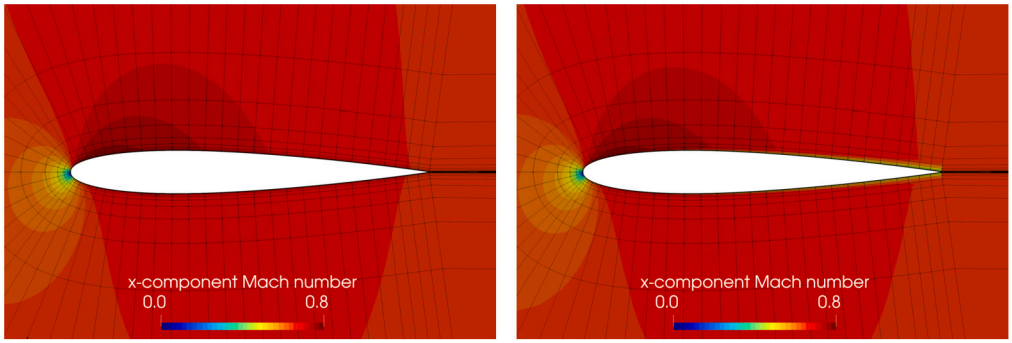

Fig. 8. Mach number contours with EoS(SW) for the Euler (left) and RANS (right) equations around the NACA0012 airfoil, on the coarse mesh with $\mathbb{P}^5$ (Euler) and $\mathbb{P}^3$ (RANS) solution approximation.

Finally, a convergence study of the EoS and LuT accuracy is performed for Euler and RANS equations, exploiting the three meshes described above and $\mathbb{P}^{1\to3}$ solution approximations. Fig. 10 compares the predicted profile of pressure around the airfoil for EoS(PR) in the turbulent case, that show the expected spatial convergence both in h and p. Also EoS(SW), LuT(PR) and LuT(SW) show the same behaviour for the predicted profiles, and for brevity they are omitted.

A qualitative investigation of the LuTs accuracy is also performed, and Fig. 11 shows the profiles of pressure and temperature around the airfoil with EoS(PR), EoS(SW), LuT(PR), and LuT(SW). The predicted curves with EoS and LuTs are coincident.

4.2. Comparison of the computational efficiency for EoS/LuT and AD/FD

Tables 5 and 6 show the cost per iteration $[s/iter]$ to solve the Euler and RANS equations, with EoS(PR), EoS(SW), LuT(PR), and LuT(SW). Jacobian matrices are computed with both AD and first order FD. As expected, EoS(SW), with both AD and FD, has a higher cost than EoS(PR), for the huge complexity of the SW thermodynamic model, while the two models have similar costs using LuTs. The difference between EoS(SW) and EoS(PR) is maximum at $\mathbb{P}^0$ (simulation with EoS(SW) is ≈ 65 times slower for the Euler equations), and decreases with the polynomial order (≈ 7 times slower at $\mathbb{P}^5$ for the Euler equations). The reduction of the thermodynamic evaluations cost with the solution approximation is motivated by the increase of the computational time needed to assembly the dG spatial discretization. The trend is confirmed also for the RANS equations, even if the overhead of EoS(SW) is

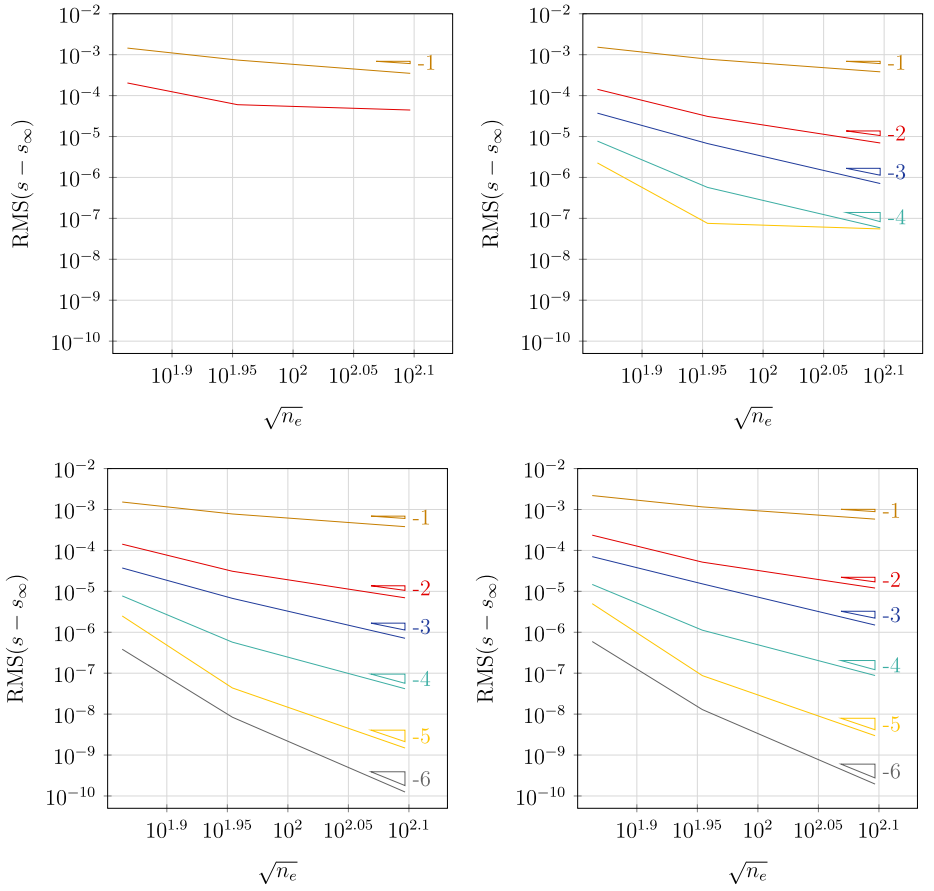

Fig. 9. RMS of the entropy error as a function of the number of elements in the mesh n_e for the NACA0012 flow case with Euler equations: LuT(PR) with 1.00×10^4 nodes (top left), LuT(PR) with 2.50×10^5 nodes (top right), LuT(PR) with 6.25×10^6 nodes (bottom left) and EoS(PR) (bottom right). $\mathbb{P}^0$ ——, $\mathbb{P}^1$ ——, $\mathbb{P}^2$ ——, $\mathbb{P}^3$ ——, $\mathbb{P}^4$ ——, $\mathbb{P}^5$ —— solution approximation. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

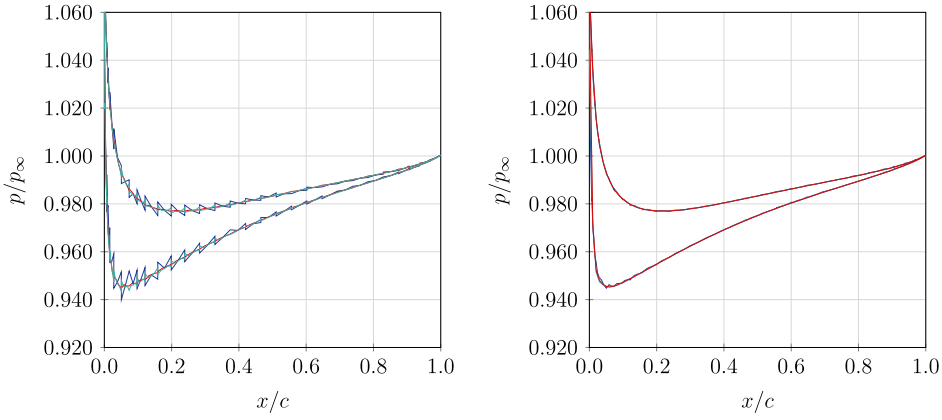

Fig. 10. Distribution around NACA0012 airfoil of pressure with EoS(PR) using different solution approximations on the coarse mesh (left) and different meshes with a $\mathbb{P}^3$ solution approximation (right). RANS equations. Left: $\mathbb{P}^1$ ——, $\mathbb{P}^2$ ——, $\mathbb{P}^3$ ——; Right: coarse mesh ——, medium mesh ——, fine mesh ——. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

lower, since the thermodynamic model decreases its computational impact with respect to the assembly of the spatial discretization (diffusive flux and turbulent source term are added in the spatial discretization).

The use of AD allows a computational saving with respect to FD, which increases with the thermodynamic and flow models complexity, and decreases with the solution approximation. For the Euler equations at $\mathbb{P}^0$ EoS(PR)+AD is 10% faster than EoS(PR)+FD, while EoS(SW)+AD is 32% faster than EoS(SW)+FD. When RANS equations are solved, AD allows reductions $\approx 37\%$

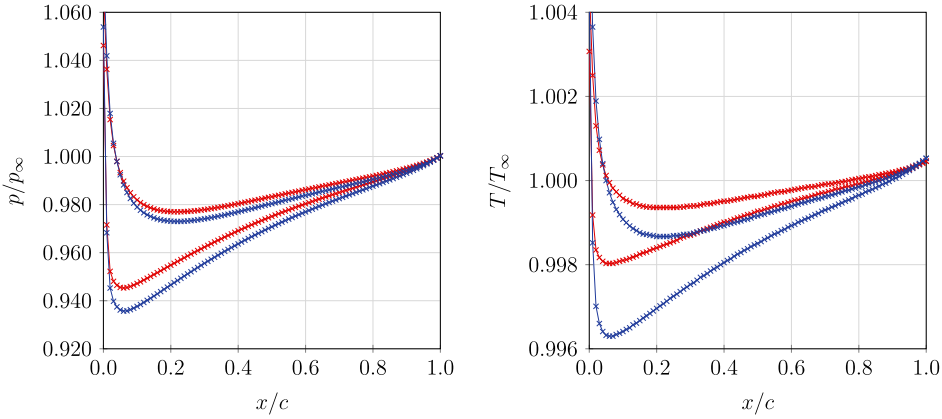

Fig. 11. Distribution around NACA0012 airfoil of pressure (left) and temperature (right) with RANS, fine mesh and $\mathbb{P}^3$ solution approximation. The profiles of LuT(PR) and LuT(SW) are here discretized with one mark each $x/c = 0.01$ to prove the overlap with respect to EoS(PR) and EoS(SW) respectively. EoS(PR) ——, LuT(PR) ×, EoS(SW) ——, LuT(SW) ×. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

Table 5
Comparison of the mean computational cost per iteration ($[s/iter]$) required for the solution of the Euler equations around the NACA0012 airfoil with EoS+AD, EoS+FD, LuT+AD and LuT+FD.

Approach	$\bar{t}(\mathbb{P}^0)$	$\bar{t}(\mathbb{P}^1)$	$\bar{t}(\mathbb{P}^2)$	$\bar{t}(\mathbb{P}^3)$	$\bar{t}(\mathbb{P}^4)$	$\bar{t}(\mathbb{P}^5)$
EoS(PR) + AD	0.161	1.781	3.684	8.837	18.208	35.562
EoS(PR) + FD	0.180	1.845	3.737	8.918	17.436	36.543
LuT(PR) + AD	0.144	1.536	3.279	8.193	15.501	33.785
LuT(PR) + FD	0.153	1.641	3.509	7.679	15.786	32.469
EoS(SW) + AD	8.824	28.278	60.070	109.709	166.460	249.606
EoS(SW) + FD	13.101	39.924	80.180	135.857	205.350	289.910
LuT(SW) + AD	0.153	1.511	3.430	7.686	15.650	31.931
LuT(SW) + FD	0.160	1.656	3.521	8.433	16.931	34.605

Table 6
Comparison of the mean computational cost per iteration ($[s/iter]$) required for the solution of the RANS equations around the NACA0012 airfoil with EoS+AD, EoS+FD, LuT+AD and LuT+FD.

Approach	$\bar{t}(\mathbb{P}^0)$	$\bar{t}(\mathbb{P}^1)$	$\bar{t}(\mathbb{P}^2)$
EoS(PR) + AD	0.148	0.660	3.553
EoS(PR) + FD	0.236	1.260	5.224
LuT(PR) + AD	0.145	0.649	3.527
LuT(PR) + FD	0.210	1.076	4.875
EoS(SW) + AD	1.899	8.910	24.813
EoS(SW) + FD	14.085	74.549	326.364
LuT(SW) + AD	0.151	0.644	3.552
LuT(SW) + FD	0.225	1.163	5.175

for the EoS(PR)+AD, and $\approx 86\%$ for the EoS(SW)+AD with respect to first order FD. The higher savings provided by AD with RANS equations is motivated by the higher number of procedures which need a jacobian counterpart for implicit time integration.

When LuTs are used for the solution of the Euler equations, AD and FD show very similar costs for both PR and SW models, while for RANS equations AD allows a reduction of the computational cost with respect to FD around 30% for both LuTs(PR)+AD and LuT(SW)+AD.

As expected, the most efficient approach to perform non-ideal gas simulation is provided by AD and LuTs, while FD and EoSs show worst performance.

Finally, the overhead of AD with respect to the best scenario, i.e., EoS(IG) with MD-derived jacobian, is investigated. In fact, Table 1 in Sec. 3.1 compares only the computing times to evaluate thermodynamic properties between LuT(SW) and EoS(IG), showing a LuT overhead around 36%. However, the real impact on the simulation time is quite different increasing the solution approximation. In particular, the difference decreases for higher polynomial orders, as the assembly of the spatial discretization and the linear system solution are predominant on the computational cost. At $\mathbb{P}^0$ the mean time increment per iteration ([%])

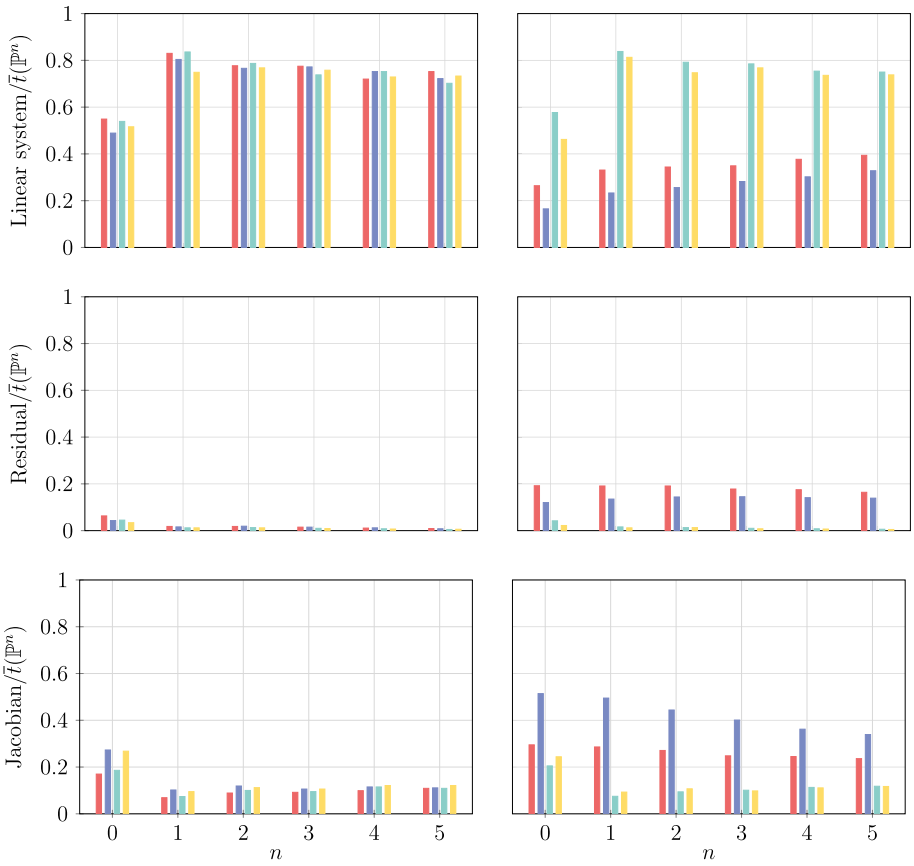

Fig. 12. Time per iteration spent for the solution of the linear systems, the assembly of the residual and of the jacobian matrix for the Euler equations with PR (left) and SW (right) model around the NACA0012 airfoil. The time of each contribution is normalized with respect to the corresponding mean computational cost per iteration $\bar{t}(\mathbb{P}^n)$ (n is the polynomial degree), reported in Table 5. EoS + AD, EoS + FD, LuT + AD, LuT + FD. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

of the LuT(SW)+AD approach with respect to the EoS(IG)+MD for the solution of the RANS equation around NACA0012 airfoil is $\overline{\Delta t}(\mathbb{P}^0) \approx 35\%$, which is comparable with results reported in Table 1, but decreases for higher solution approximations ($\overline{\Delta t}(\mathbb{P}^2) \approx 17\%$).

4.3. Cost of the residual and jacobian assembly, and linear system solution for a non-ideal gas simulation

In this paragraph the effect of a non-ideal gas code extension is investigated in depth, and the mean computational cost per iteration to perform Euler and RANS simulations reported in Tables 5 and 6 are exploded, to compare the differences in terms of residual and jacobian assembly, and linear system solution.

Figs. 12 and 13 show at each iteration the time spent for the solution of the linear systems, the assembly of the residual and of the jacobian matrix of the residual for the Euler and RANS equations and different solution approximations, with PR and SW models. The time of each contribution is normalized with respect to the corresponding mean computational cost per iteration $\bar{t}(\mathbb{P}^n)$ (n is the polynomial degree), reported in Tables 5 and 6.

Euler simulations, see Fig. 12, show that the main cost of an iteration for simple EoS, e.g., the PR model, can be ascribed to the linear system solution ($\approx 80\%$), $\approx 10\%$ is spent for the assembly of the jacobian matrix, while the cost of the residual assembly is negligible. When complex thermodynamic models are adopted, e.g., SW model, the use of the LuT is mandatory to recover the time distribution obtained with PR model. In fact, the direct evaluation of the EoS entails a huge amount of time spent for the residual ($\approx 20\%$) and the jacobian ($\approx 30\%$ with AD and $\approx 40\%$ with FD) assembly. As a consequence, the relative time spent for the linear system solution decreases and it is around 40%. These results suggest that the non-ideal gas extension (LuT+AD or LuT+FD) for the Euler equation has a small impact on the iteration time, as most of the time is spent to solve the linear system.

RANS simulations, see Fig. 13, show a different behaviour in comparison to Euler results. In fact, the time spent for the assembly of the jacobian matrix is predominant ($50\% - 60\%$), while the linear system solution requires $\approx 30\%$ of the iteration time. The direct evaluation of the EoS for the SW model shows the higher time for the assembly, and as a consequence, lower values for the solution of the linear system. In general, turbulent simulations show a stronger dependence on the thermodynamic model.

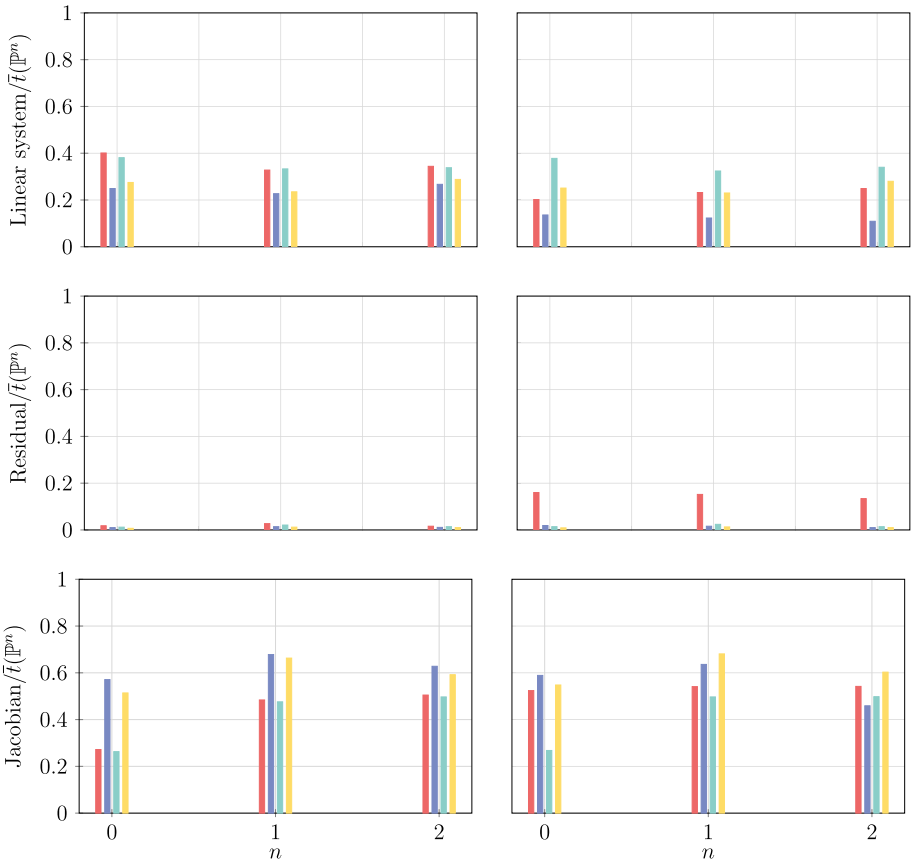

Fig. 13. Time per iteration spent for the solution of the linear systems, the assembly of the residual and of the jacobian matrix for the RANS equations with PR (left) and SW (right) model around the NACA0012 airfoil. The time of each contribution is normalized with respect to the corresponding mean computational cost per iteration $\overline{t}(\mathbb{P}^n)$ (n is the polynomial degree), reported in Table 6. EoS + AD, EoS + FD, LuT + AD, LuT + FD. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

5. Conclusions

In this work, two candidate ways to perform efficient non-ideal gas simulation are implemented and assessed in an implicit dG solver. In particular, for the calculation of thermodynamic properties and derivatives, a structured LuT approach based on bicubic interpolation is used rather than a direct resolution of the EoSs. For the calculation of the thermodynamic dependent contributions to the jacobian matrix of the residual, the open source AD tool TAPENADE [17] is used through an automated Python program developed to obtain non-redundant and mathematically exact procedures, as an alternative to first order FD jacobians. The performance of the new the solver is assessed in the computation of the inviscid and turbulent flow around a NACA0012 airfoil, using the PR and SW gas models in dense free stream conditions.

Results show that medium size LuTs (nodes in the range $10^4 - 10^6$) guarantee *i*) a correct approximation of the thermodynamic plane that does not deteriorate the accuracy of the final solution, and *ii*) a reduction of the time required to perform non-ideal gas simulations. LuTs allow great benefits for both inviscid and turbulent flows, especially when complex thermodynamic models are used, e.g., the SW model.

Moreover, the use of AD to build the jacobian matrix guarantees a speed-up with respect to first order FD, which is more evident in turbulent simulations, where more dependencies from the thermodynamic model are present in the jacobian matrix. As expected, the LuT + AD approach is the most efficient, while the EoS + FD is the worst. The LuT + AD approach guarantees a computational saving up to 99% with respect to EoS + FD, when RANS equations are solved with SW model. The computational overhead of the LuT + AD approach in the worst condition, i.e., against an ideal gas solver with MD-derived jacobian, shows a decreasing behaviour with the polynomial order of the solution approximation, and is around 17% for a $\mathbb{P}^2$ solution approximation.

Future works will investigate *i*) the use of AD and LuTs for non-ideal gas simulations of complex geometries characterised by complex flow features, e.g., ORC's turbomachinery, both with RANS and implicit LES models, and *ii*) new strategies to create structured LuT that are able to capture correctly the saturation curves.

CRediT authorship contribution statement

E. Mantecca: Data curation, Formal analysis, Investigation, Methodology, Validation, Visualization, Writing – original draft, Writing – review & editing. **A. Colombo:** Conceptualization, Project administration, Software, Supervision, Writing – review & editing. **A. Ghidoni:** Conceptualization, Project administration, Resources, Software, Supervision, Writing – review & editing. **G. Noventa:** Data curation, Formal analysis, Investigation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgements

The Authors sincerely thank Prof. Andrea Crivellini of Università Politecnica delle Marche, who provided the structured grids used for the computations around the NACA0012 airfoil.

Appendix A. Examples of AD

To better understand how an AD-generated derivative is and how it can be made more efficient, an example is shown in Fig. A.14, where a generic function $\mathbf{f}(\mathbf{x})$ is computed through the subroutine `fun`, using also an external procedure `g`. To generate the jacobian matrix $\partial\mathbf{f}/\partial\mathbf{x}$, TAPENADE needs both `fun` and `g` as source programs, and differentiates the target `fun` with respect to the n components of $\mathbf{x} = [x_1, ..., x_n]^T$. The result of the differentiation with respect to x_1, after a split of the input vector $\mathbf{x}$ into its single components, is shown in Fig. A.15. The structure of the differentiated code resembles the typical output of TAPENADE.

The new routine `fun_d` generated by TAPENADE returns `fd` = `f` for `x1d` = 0, or `fd` = ∂`f`/∂`x1` for `x1d` = 1, which is the exact first column of $\partial\mathbf{f}/\partial\mathbf{x}$, as reported in Eq. (33). The new subroutine `g_d` computes the derivative of the output of `g`, that is called by `fun` and does not depend on `x1`, ..., `xn`. Notice that both `fun_d` and `g_d` contain also the code lines of `fun` and `g`. Therefore, during the calculation of every column of $\partial\mathbf{f}/\partial\mathbf{x}$, the lines of `fun` and `g` are repeated uselessly. This roughly equals the cost of evaluating $\mathbf{f}(\mathbf{x})$ twice for every jacobian column, leading to a computational cost comparable with a first order FD approximation.

However, AD allows to extract these lines and calculate them just once for every column. The benefit provided by this operation increases with the number of extracted lines. Moreover, AD detects automatically if `f` has no dependencies from the input variables `x1`, ..., `xn`, and does not create the corresponding routine for the relative jacobian column, which can be set to zero avoiding some calculations. By applying these considerations, modified routines `fun_d1m` and `g_dm` can be created, where only the differential contribution is computed and the results from `fun` and `g` are given to them as new inputs (see Fig. A.16).

Finally, a wrapping routine `funj` is created for the assembly of the whole jacobian, as shown in Fig. A.17. Notice that `fun_dim` is not a null vector just for `i` = 1, 2, `n`, and that `a`, `b`, `c`, `y` and `temp` are calculated just once in the routine `funj_p`, which contains all the non differentiated code lines. In this work, the AD-workflow has been fully automated through a Python code that was designed to generate efficient jacobian matrices for convective and diffusive fluxes, turbulence model source terms and boundary conditions.

```
SUBROUTINE fun(n,m,x,f)

   integer :: n, m, i
   real    :: x(n), f(m), y(m)
   real    :: a, b, c

   a = x(1)+x(2)+x(n)
   b = x(1)*x(n)
   c = b/a

   CALL g(m,a,b,c,y)

   DO i = 1, m
      f(i) = 2*a*i*y(i)
   END DO

END SUBROUTINE
```

```
SUBROUTINE g(m,a,b,c,y)

   integer :: m, i
   real    :: a, b, c, y(m)

   y(1) = a**3/b

   DO i = 2, m-1
      y(i) = a
   END DO

   y(m) = y(1)*exp(c/a)

END SUBROUTINE
```

Fig. A.14. Example of source code that is automatically differentiated for the computation of its jacobian matrix.

```
SUBROUTINE fun_d(m,x1,x1d,x2, &
   ...,xn,f,fd)

   integer :: m, i
   real    :: x1, x1d, x2, ..., xn
   real    :: f(m), y(m)
   real    :: fd(m), yd(m)
   real    :: a, b, c
   real    :: ad, bd, cd

   a  = x1+x2+xn
   ad = x1d
   b  = x1*xn
   bd = x1d*xn
   c  = b/a
   cd = (bd*a-b*ad)/a**2

   CALL g_d(m,a,ad,b,bd,c,cd,y,yd)

   DO i = 1, m
      f(i) = 2*a*i*y(i)
      fd(i) = 2*i*(ad*y(i) &
         +a*yd(i))
   END DO

END SUBROUTINE
```

```
SUBROUTINE g_d(m,a,ad,b,bd,c,cd,y,yd)

   integer :: m, i
   real    :: a, b, c, y(m), temp
   real    :: ad, bd, cd, yd(m), tempd

   y(1) = a**3/b
   yd(1) = (b*3*ad*a**2 &
      -bd*a**3)/b**2

   DO i = 2, m-1
      y(i) = a
      yd(i) = ad
   END DO

   temp = c/a
   tempd = (cd*a-c*ad)/a**2
   y(m) = y(1)*exp(temp)
   yd(m) = yd(1)*exp(temp) &
      +y(1)*exp(temp)*tempd

END SUBROUTINE
```

Fig. A.15. AD code generated by TAPENADE with respect to the input variable `x1`.

```
SUBROUTINE fun_d1m(m,x1,x1d,x2, &
   ...,xn,fd,a,b,c,y,temp)

   integer :: m, i
   real    :: x1, x1d, x2, ..., xn
   real    :: a, b, c, y(m), temp
   real    :: ad, bd, cd
   real    :: yd(m), fd(m)

   ad    = x1d
   bd    = x1d*xn
   cd    = (bd*a-b*ad)/a**2

   CALL g_dm (m,a,ad,b,bd,c,cd, &
      y,yd,temp)

   DO i = 1, m
      fd(i) = 2*i*(ad*y(i) &
         +a*yd(i))
   END DO

END FUNCTION
```

```
SUBROUTINE g_dm(m,a,ad,b,bd,c,cd, &
    y,yd,temp)

   integer :: m, i
   real    :: a, b, c, y(m), temp
   real    :: ad, bd, cd, yd(m), tempd

   yd(1) = (b*3*ad*a**2 &
      -bd*a**3)/b**2

   DO i = 2, m-1
      yd(i) = ad
   END DO

   tempd = (cd*a-c*ad)/a**2
   yd(m) = yd(1)*exp(temp) &
      +y(1)*exp(temp)*tempd

END SUBROUTINE
```

Fig. A.16. Efficient version (`fun_d1m` and `g_dm`) of TAPENADE routines `fun_d` and `g_d`.

```
SUBROUTINE funj(n,m,x,fj)

   integer :: n, m, i
   real    :: x1, x2, ..., xn
   real    :: x1d, x2d, ..., xnd
   real    :: x(n), fd(m), fj(m,n)
   real    :: a, b, c, y(m), temp

   x1 = x(1)
   x1d = 1d0
   ...
   xn = x(n)
   xnd = 1d0

   CALL funj_p(m,x1,...,xn, &
      a,b,c,y,temp)

   CALL fun_d1m(m,x1,x1d,x2, &
      ...,xn,fd,a,b,c,y,temp)
   DO i = 1, m
      fj(i,1) = fd(i)
   END DO
   ...
   CALL fun_dnm(m,x1,x2, &
      ...,xn,xnd,fd,a,b,c,y,temp)
   DO i = 1, m
      fj(i,n) = fd(i)
   END DO

END SUBROUTINE
```

```
SUBROUTINE funj_p(m,x1,...,xn, &
                  a,b,c,y,temp)

   integer :: m, i
   real    :: x1, x2, ..., xn
   real    :: a, b, c, y(m), temp

   a    = x1+x2+xn
   b    = x1*xn
   c    = b/a

   y(1) = a**3/b

   DO i = 2, m-1
      y(i) = a
   END DO

   temp  = c/a
   y(m) = y(1)*exp(temp)

END SUBROUTINE
```

Fig. A.17. Wrapping routine to assembly the whole jacobian matrix.

References

[1] P. Colonna, S. Rebay, Numerical simulation of dense gas flows on unstructured grids with an implicit high resolution upwind Euler solver, Int. J. Numer. Methods Fluids 46 (7) (2004) 735–765, https://doi.org/10.1002/fld.762.

[2] M. Vinokur, J. Montagné, Generalized flux-vector splitting and Roe average for an equilibrium real gas, J. Comput. Phys. 89 (2) (1990) 276–300, https://doi.org/10.1016/0021-9991(90)90145-Q.

[3] D. Peng, D. Robinson, A new two-constant equation of state, Ind. Eng. Chem. Fundam. 15 (1) (1976) 59–64, https://doi.org/10.1021/i160057a011.

[4] R. Span, W. Wagner, A new equation of state for carbon dioxide covering the fluid region from the triple-point temperature to 1100 K at pressures up to 800 MPa, J. Phys. Chem. Ref. Data 25 (6) (1996) 1509–1596, https://doi.org/10.1063/1.555991.

[5] M. Pini, S. Vitale, P. Colonna, G. Gori, A. Guardone, T. Economon, J. Alonso, F. Palacios, SU2: the open-source software for non-ideal compressible flows, J. Phys. Conf. Ser. 821 (1) (2017) 012013, https://doi.org/10.1088/1742-6596/821/1/012013.

[6] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, M. Savini, A high-order accurate discontinuous finite element method for inviscid and viscous turbomachinery flows, in: Proceedings of the 2nd European Conference on Turbomachinery Fluid Dynamics and Thermodynamics, Antwerpen, Belgium, 1997, pp. 99–109.

[7] F. Bassi, L. Botti, A. Colombo, A. Crivellini, N. Franchina, A. Ghidoni, S. Rebay, Very high-order accurate discontinuous Galerkin computation of transonic turbulent flows on aeronautical configurations, in: ADIGMA - a European Initiative on the Development of Adaptive Higher-Order Variational Methods for Aerospace Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 25–38.

[8] F. Bassi, L. Botti, A. Colombo, A. Crivellini, C. De Bartolo, N. Franchina, A. Ghidoni, S. Rebay, Time Integration in the Discontinuous Galerkin Code MIGALE - Steady Problems, Springer International Publishing, Cham, 2015, pp. 179–204.

[9] F. Bassi, L. Botti, A. Colombo, A. Crivellini, A. Ghidoni, A. Nigro, S. Rebay, Time Integration in the Discontinuous Galerkin Code MIGALE - Unsteady Problems, Springer International Publishing, 2015, pp. 205–230.

[10] E. Mantecca, A. Colombo, A. Ghidoni, G. Noventa, D. Pasquale, S. Rebay, On the development of an implicit discontinuous Galerkin solver for turbulent real gas flows, Fluids 8 (2023), https://doi.org/10.3390/fluids8040117.

[11] A. Colombo, A. Ghidoni, E. Mantecca, G. Noventa, S. Rebay, D. Pasquale, Development of a discontinuous Galerkin solver for the simulation of turbine stages, https://doi.org/10.23967/eccomas.2022.087, 2022.

[12] P. Boncinelli, F. Rubechini, A. Arnone, M. Cecconi, C. Cortese, Real gas effects in turbomachinery flows: a computational fluid dynamics model for fast computations, J. Turbomach. 126 (2) (2004) 268–276, https://doi.org/10.1115/1.1738121.

[13] M. Dumbser, U. Iben, C.-D. Munz, Efficient implementation of high order unstructured WENO schemes for cavitating flows, Comput. Fluids 86 (2013) 141–168, https://doi.org/10.1016/j.compfluid.2013.07.011.

[14] M. Pini, A. Spinelli, G. Persico, S. Rebay, Consistent look-up table interpolation method for real-gas flow simulations, Comput. Fluids 107 (2015) 178–188, https://doi.org/10.1016/j.compfluid.2014.11.001.

[15] M. De Lorenzo, P. Lafon, M. Di Matteo, M. Pelanti, J.-M. Seynhaeve, Y. Bartosiewicz, Homogeneous two-phase flow models and accurate steam-water table look-up method for fast transient simulations, Int. J. Multiph. Flow 95 (2017) 199–219, https://doi.org/10.1016/j.ijmultiphaseflow.2017.06.001.

[16] A. Rubino, M. Pini, M. Kosec, S. Vitale, P. Colonna, A look-up table method based on unstructured grids and its application to non-ideal compressible fluid dynamic simulations, J. Comput. Sci. 28 (2018) 70–77, https://doi.org/10.1016/j.jocs.2018.08.001.

[17] L. Hascoet, V. Pascual, The Tapenade automatic differentiation tool: principles, model, and specification, ACM Trans. Math. Softw. 39 (3) (2013), https://doi.org/10.1145/2450153.2450158.

[18] F. Bassi, L. Botti, A. Colombo, A. Crivellini, N. Franchina, A. Ghidoni, Assessment of a high-order accurate discontinuous Galerkin method for turbomachinery flows, Int. J. Comput. Fluid Dyn. 30 (4) (2016) 307–328, https://doi.org/10.1080/10618562.2016.1198783.

[19] D.C. Wilcox, Turbulence Modelling for CFD, DCW Industries Inc., La Cañada, CA 91011, USA, 2006.

[20] T.H. Chung, M. Ajlan, L.L. Lee, K.E. Starling, Generalized multiparameter correlation for nonpolar and polar fluid transport properties, Ind. Eng. Chem. Res. 27 (1988) 671–679.

[21] P. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (2) (1981) 357–372, https://doi.org/10.1016/0021-9991(81)90128-5.
[22] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W.D. Gropp, D. Karpeyev, D. Kaushik, M.G. Knepley, D.A. May, L.C. McInnes, R.T. Mills, T. Munson, K. Rupp, P. Sanan, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 - Revision 3.11, Argonne National Laboratory, 2019.
[23] G. Noventa, F. Massa, F. Bassi, A. Colombo, N. Franchina, A. Ghidoni, A high-order discontinuous Galerkin solver for unsteady incompressible turbulent flows, Comput. Fluids 139 (2016) 248–260, https://doi.org/10.1016/j.compfluid.2016.03.007.
[24] G. Noventa, F. Massa, S. Rebay, F. Bassi, A. Ghidoni, Robustness and efficiency of an implicit time-adaptive discontinuous Galerkin solver for unsteady flows, Comput. Fluids 204 (2020), https://doi.org/10.1016/j.compfluid.2020.104529.
[25] F. Massa, G. Noventa, F. Bassi, A. Colombo, A. Ghidoni, M. Lorini, High-order linearly implicit two-step peer methods for the discontinuous Galerkin solution of the incompressible RANS equations, in: ECCOMAS Congress 2016, JUNE 2016 Crete Island, Greece, VII European Congress on Computational Methods in Applied Sciences and Engineering, 2016, pp. 2664–2683.
[26] A. Guardone, L. Vigevano, B.M. Argrow, Assessment of thermodynamic models for dense gas dynamics, Phys. Fluids 16 (2004) 3878–3887, https://doi.org/10.1063/1.1786791.
[27] A. Spinelli, G. Cammi, S. Gallarini, M. Zocca, F. Cozzi, P. Gaetani, V. Dossena, A. Guardone, Experimental evidence of non-ideal compressible effects in expanding flow of a high molecular complexity vapor, Exp. Fluids 59 (2018), https://doi.org/10.1007/s00348-018-2578-0.
[28] G. Xia, D. Li, C.L. Merkle, Consistent properties reconstruction on adaptive Cartesian meshes for complex fluids computations, J. Comput. Phys. 225 (1) (2007) 1175–1197, https://doi.org/10.1016/j.jcp.2007.01.034.
[29] O. Wilhelmsen, A. Aasen, G. Skaugen, P. Aursand, A. Austegard, E. Aursand, M.A. Gjennestad, H. Lund, G. Linga, M. Hammer, Thermodynamic modeling with equations of state: present challenges with established methods, Ind. Eng. Chem. Res. 56 (13) (2017) 3503–3515, https://doi.org/10.1021/acs.iecr.7b00317.
[30] K. Yiu, D. Greaves, S. Cruz, A. Saalehi, A. Borthwick, Quadtree grid generation: information handling, boundary fitting and cfd applications, Comput. Fluids 25 (8) (1996) 759–769, https://doi.org/10.1016/S0045-7930(96)00029-1.
[31] C.A. Duncan, M.T. Goodrich, S. Kobourov, Balanced aspect ratio trees: combining the advantages of k-d trees and octrees, J. Algorithms 38 (2001), https://doi.org/10.1006/jagm.2000.1135.
[32] I. Wald, V. Havran, On building fast kd-trees for ray tracing, and on doing that in $o(NlogN)$, https://doi.org/10.1109/RT.2006.280216, 2006.
[33] N. Kouiroukidis, G. Evangelidis, The effects of dimensionality curse in high dimensional kNN search, https://doi.org/10.1109/PCI.2011.45, 2011.
[34] F. Ringleb, Exakte lösungen der differentialgleichungen einer adiabatischen gasströmung, J. Appl. Math. Mech. (Zeitschrift für Angewandte Mathematik und Mechanik) 20 (1940), https://doi.org/10.1002/zamm.19400200402.
[35] A.H. Shapiro, The Dynamics and Thermodynamics of Compressible Fluid Flow, The Ronald Press Company, 1955.
[36] R. Abgrall, P. Congedo, D. De Santis, N. Razaaly, A non-linear residual distribution scheme for real-gas computations, Comput. Fluids 102 (2014), https://doi.org/10.1016/j.compfluid.2014.06.031.
[37] P. Colonna, N. Nannan, A. Guardone, E. Lemmon, Multiparameter equations of state for selected siloxanes, Fluid Phase Equilib. 244 (2006), https://doi.org/10.1016/j.fluid.2006.04.015.