Computers and Fluids 266 (2023) 106041

Contents lists available at ScienceDirect

Computers and Fluids

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

Acceleration of supersonic/hypersonic reactive CFD simulations via heterogeneous CPU-GPU supercomputing

F. Ghioldi *, F. Piscaglia

Department of Aerospace Science and Technology (DAER), Politecnico di Milano, Italy

ARTICLE INFO

Keywords:
Supersonic combustion
Reactive flows
GPU chemistry
Heterogeneous super-computing
CUDA
GPGPU solver
`hyperFoam`
OpenFOAM

ABSTRACT

The numerical study of reactive flows subjected to supersonic conditions is accelerated by the co-design of a novel strategy to integrate finite-rate chemistry by an adaptive multi-block ODE algebra solver for Graphical Processing Units (GPU), that is coupled to a parallel, shock-capturing Finite-Volume reactive flow solver running on CPUs. The resulting GPGPU solver is validated on Large Eddy Simulations (LES) of a scramjet configuration, whose experimental measurements are available from the literature. It is demonstrated that the proposed method significantly accelerates the solution of reactive CFD computations with Direct Integration of the finite-rate chemistry.

1. Introduction

Supersonic combustion ramjet (scramjet) engines embody the core technology required to develop the future generation of supersonic/hypersonic transportation and high-altitude cruisers [1]. Furthermore, these air-breathing propulsion systems have been recognized as the most promising design to enhance the mission potential of the next-generation military tactical missiles as they lack moving parts or oxygen tanks [2]. The flow structure in the scramjet engine is very complex and dominated by multiple shocks reflecting on the chamber walls as they travel downstream in the combustor. Large velocity and temperature gradients caused by the shocks trigger combustion. Among the several technical challenges that limit the employment of scramjet engines in commercial applications, it is worth mentioning: the influence of the engine geometry and that of the flow regime on the shock pattern; the losses of total temperature and pressure that influence the combustion efficiency [3]; the study of the mass capture ratio at different operating conditions [4,5]; the presence of flow-choking at the inlet section [6]; the interaction between the shock wave systems and the boundary layer [7,8]; the effects produced by a variation of the cowl deflection angle [9] and other geometrical quantities [10]. Ignition and flame holding in scramjet engines are critical aspects to be addressed [3,11]. Fuel injection in scramjet engines [12] usually occurs by means of: (a) wall injectors, that inject the fuel from a side wall; (b) strut injectors, that are located in proximity of the channel mid-axis, hence injecting the fuel directly in the core of the surrounding air stream. Adequate air-fuel mixing must be achieved to facilitate combustion in each case [13,14]. Studies of actual operational flight scenarios of scramjet engines are very limited because the harsh operating conditions make experiments very difficult to perform. Additionally, the lack of appropriate experimental techniques or instrumentation to measure reacting flow quantities in sophisticated ground facilities contributes to affect negatively the experimental characterization of the configurations [15]. In this scenario, Computational Fluid Dynamics (CFD) can help to study hypersonic vehicles at different flow conditions and with strong flame unsteadiness [16,17]. When modeling flame dynamics, an accurate prediction of the chemical evolution, including the interaction with the different turbulence scales, must be accounted; the numerical solver must therefore be able to account for: (a) the turbulent flow dynamics under supersonic conditions; (b) the transport of N_s chemical (reacting) species, that are transported in the computational domain by $N_s - 1$ convection–diffusion equations; reactions of the species are accounted by a system of chemical Ordinary Differential Equations (ODEs), whose size is influenced by the grid resolution, the number of chemical species involved, and the number of elementary reactions of the mechanism [18]. Integration of finite-rate chemistry also causes strong load unbalancing among the processors and limits the time-step advancement required to preserve a stable solution.

1.1. Motivation of this research

The renewed interest in supersonic aircraft travel promotes the rise of studies whose goal is to propose novel and efficient engine designs,

* Corresponding author.
E-mail address: federico.ghioldi@polimi.it (F. Ghioldi).

https://doi.org/10.1016/j.compfluid.2023.106041
Received 31 December 2022; Received in revised form 27 May 2023; Accepted 20 August 2023
Available online 24 August 2023

that should operate under strict environmental constraints. In this regard, CFD simulations can help engine designers to predict performance and emissions of the engines with different Sustainable Aviation Fuel (SAFs). Reactive CFD simulations of supersonic/hypersonic combustion are computationally demanding, because the flow transport is solved in combination to the ODE system describing the finite-rate chemistry problem. The large resolution needed by reactive supersonic tests makes the computational demand significant. The research described in this paper is motivated by the need to speed up combustion simulations with finite-rate chemistry, where chemical non-equilibrium effects are imperative, in order to predict the pollutant emissions with different fuels and combustion modes. In this sense, algorithmic developments to speed up the solution of ODEs in reactive CFD solvers have been already presented through the years, in the form of reduction [19,20], tabulation [21,22], and Artificial Neural Networks strategies [23–25]. Their success is due to their speed if they are compared to Direct-Integration (DI) of finite-rate chemistry, in particular with detailed mechanisms. On the other hand, they usually require extensive pre-processing operations [26] or case-dependent tuning of some user-input parameters, that is not required with DI. Heterogeneous GPGPU computing potentially represents a very effective solution to accelerate DI of the chemistry problem, that can be solved on accelerators (GPU) while the fluid transport and turbulence are computed on conventional CPU-based hardware technologies. Such a strategy requires significant re-design of the methods (software co-design) to work [27,28].

1.2. Goals and highlights

This paper discusses the extension of a density-based shock-capturing flow solver for the treatment of supersonic combustion. The solution of flow and species transport is done on the CPU via an operator-splitting approach; the application of the Kurganov-Noel-Petrova central upwind scheme ensures to capture the shocks with a limited amount of numerical dissipation [29,30]. Chemical source terms are computed by the solution of the finite-rate chemistry problem: the reaction mechanism [31,32] is solved in the form of Ordinary Differential Equations (ODEs) by a novel GPU-ODE integration method that is structured on the explicit time-step-adaptive Runge–Kutta Cash–Karp approach and is linked to the flow transport computed on the CPU. To improve the efficiency and the performance of the implementation, the resulting GPGPU strategy includes: (a) automatic load balance between different hardware architectures (CPUs and GPUs); (b) minimization of the data transfer overheads; (c) coalesced run-time access of the GPU dynamic data. The resulting hybrid CPU/GPU solver is built as a separate compilation unit (C++/CUDA dynamic library) linked to the open-source C++ software OpenFOAM® [33,34]. The GPU-ODE integrator can be linked to any reactive flow solver available in the selected framework. It is demonstrated that the proposed method significantly accelerates the Direct Integration (DI) of the chemistry problem in reactive CFD computations.

1.3. Paper structure

The paper is structured as follows. The governing equations for compressible reactive supersonic flow problems are presented in Section 2. Details of the applied discretization methods are presented in Section 3. The design of the GPU-ODE integrator for the solution of finite-rate chemistry is discussed in Section 4. Code verification is reported in Section 5 via GPGPU simulations on single-cell batch reactors. The tested chemical mechanism is subsequently used in Section 6 to simulate via the GPGPU method the combustion phenomena in a scramjet configuration from the DLR combustor facility [35–37]. Observations on the computational speedup are reported in Section 7. Conclusions are drawn in Section 8.

2. Governing equations for supersonic/hypersonic reactive flows

The compressible fluid-dynamic Navier–Stokes equations include mass conservation, momentum and energy balances. They are written as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}) = 0 \tag{1}$$

$$\frac{\partial (\rho \boldsymbol{U})}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}\boldsymbol{U}) = -\nabla p + \nabla \cdot \boldsymbol{R} + \boldsymbol{S}_U \tag{2}$$

$$\frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho \boldsymbol{U} E) + \nabla \cdot (\boldsymbol{U} p) = -\nabla \cdot \boldsymbol{q} + \nabla \cdot (\boldsymbol{R} \cdot \boldsymbol{U}) + \dot{Q} + S_e \tag{3}$$

$\boldsymbol{R}$ is the viscous part of the stress tensor. In Eq. (3), the density $\rho(p,T)$ multiplies $E := e(T) + |\boldsymbol{U}|^2/2$, i.e. the sum of the internal and kinetic energy per unit mass; $\boldsymbol{q} := \lambda \nabla T$ represents the heat flux vector; λ is the thermal conductivity; the terms $\boldsymbol{S}_U$ and S_e include sources and sinks for momentum (Eq. (2)) and energy (Eq. (3)), respectively. $\dot{Q}$ is the heat released by the combustion. Due to the presence in the domain of N_s chemical species, the system of equations is augmented. The addition of $N_s - 1$ convection–diffusion functions investigates the evolution of each mass fraction Y_i through space and time:

$$\frac{\partial (\rho Y_i)}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}) Y_i = \nabla \cdot (\rho D_i \nabla Y_i) + \overline{\dot{\omega}_i} \qquad \text{for} \quad i \in [1, N_s - 1] \tag{4}$$

The mass fraction of the inert species Y_{N_s} is obtained by considering a unitary sum:

$$Y_{N_s} = 1 - \sum_{i=1}^{N_s - 1} Y_i \tag{5}$$

In Eq. (4), D_i is the mass diffusion coefficient; in reactive simulations, $\overline{\dot{\omega}_i}$ is defined as:

$$\overline{\dot{\omega}_i} = K_i \, \dot{\omega}_i \tag{6}$$

where the reaction rate of the ith specie $\dot{\omega}_i$

$$\dot{\omega}_i = W_i \sum_{j=1}^{N_R} \nu_{i,j} Q_j \tag{7}$$

is scaled by a specific set of coefficients K_i depending on the selected combustion model, to account eventually for the interaction between turbulent mixing and chemistry in the CV. With laminar combustion, the laminar finite-rate model is used and $K_i = 1$ in Eq. (6). In Eq. (7), W_i is the molecular weight of the ith species; $\nu_{i,j}$ is the ith species stoichiometric coefficient, and Q_j is the non-equilibrium reaction rate of the jth reaction defined as:

$$Q_j = \kappa_{f,j}(T,p) \prod_{i \in P} \left(\frac{\rho Y_i}{W_i} \right)^{\nu'_{i,j}} - \kappa_{r,j}(T,p) \prod_{i \in R} \left(\frac{\rho Y_i}{W_i} \right)^{\nu''_{i,j}} \tag{8}$$

In Eq. (8), $\kappa_{f,j}(T,p)$ and $\kappa_{r,j}(T,p)$ are the forward and reverse rate constants at the local fluid dynamic conditions [38], R and P denote reactants and products, and the ratio $\rho Y_i / W_i$ is the molar concentration of the ith species, that will be named c_i in the following:

$$c_i = \frac{\rho Y_i}{W_i} \tag{9}$$

The heat released by the combustion $\dot{Q}$ is:

$$\dot{Q} = \sum_{i=1}^{N_s} \left(\overline{\dot{\omega}_i} \, H_{f,i} \right) \tag{10}$$

in which H_f is the enthalpy of formation. To achieve the closure of the system, constitutive relations are needed; their formulation depends on the properties of the continuous medium. The following set of constitutive relations is used:

– the generalized form of the Newton's law of viscosity:

$$\boldsymbol{R} = \mu \left[\nabla \boldsymbol{U} + (\nabla \boldsymbol{U})^T \right] + \left(\frac{2}{3} \mu \nabla \cdot \boldsymbol{U} \right) \boldsymbol{I} \tag{11}$$

in which μ is the dynamic viscosity and $\boldsymbol{I}$ is the identity matrix.

– a nine-coefficient polynomial computes the thermodynamic properties in standard-state for the ith gaseous species, as done in the NASA chemical equilibrium code [39], to define the internal energy as function of the pressure and temperature.
– the Equation of State (EoS) for the gas, that is assumed as a mixture of N_s species:

$$p = \rho R_0 T \sum_{i=1}^{N_s} \frac{Y_i}{W_i} = \rho \frac{R_0}{W} T \quad \text{with} \quad \frac{1}{W} = \sum_{i=1}^{N_s} \frac{Y_i}{W_i} \tag{12}$$

where W is the mean molecular weight of the mixture. All the species of the mixture are treated as perfect gases with common temperature T; each species is described by Mendeleev–Clapeyron EoS $p_k = \rho_k \frac{R_0}{W_k} T$, being $R_0 = 8.314$ J/(mol K) the perfect gas constant, p_i and ρ_i partial pressure and density of the ith species, with $p = \sum_{i=1}^{N_s} p_i$ (Dalton law). Despite the assumption of mixture of perfect gases is applied in this work, the solver natively supports also the real-gas formulation.
– Eddy-viscosity based models for turbulence closure.

Transport properties, (mass diffusion coefficients, thermal conductivity and viscosity of the species) and thermochemical data for the gas phase are imported from the Cantera transport database through the developed `canteraToFoam` utility. The time-step of integration Δt_{fluid} of the Partial Differential Equations (PDEs) describing the fluid transport problem must comply with the CFL condition. The progressive production/consumption of Y_i depends on the occurring combustion phenomena and it is regulated by the prescribed chemical chain and the thermo-fluid dynamic conditions in the system. In finite-rate chemistry, this implies the solution of a set of Ordinary Differential Equations (ODEs) structured on the overall combustion chain. The integration of the ODE system of the kinetic problem is performed over a time interval Δt_{chem} that must ensure computational stability [40]. If $\Delta t_{\text{chem}} < \Delta t_{\text{fluid}}$, a time step sub-cycling strategy is used and the reaction rate $\dot{\omega}_i$ is computed over $\Delta t_{\text{fluid}} = t^{n+1} - t^n$, as:

$$\dot{\omega}_i = \left(c_i^{n+1} - c_i^n\right) \frac{W_i}{\Delta t_{\text{fluid}}} = \rho^n \frac{(Y_i^{n+1} - Y_i^n)}{\Delta t_{\text{fluid}}} \tag{13}$$

being c_i^n the molar concentration of the ith species at t^n, and c_i^{n+1} that at t^{n+1}. The update of the species concentration in the CV from time n to $n+1$ is computed by the selected ODE integrator. Implicit ODE solvers may guarantee the best performance when the integration time step is extended, i.e when chemical stiffness is low or the fast modes of the ODE system have reached an asymptotic value. Nonetheless, the explicit integration over a time is computationally cheaper, as it avoids iterative solutions and associated matrix inversions. In addition, explicit methods are well-suited to GPGPU computing because of their intrinsic parallel nature [41]. For this reason, the explicit Runge–Kutta Cash–Karp (`RKCK45`) method with adaptive time stepping is selected in this work [32]. In `RKCK45`, the presence of embedded formulas produces several advantages [40]: a high-order accuracy can be reached with few evaluation functions; simple estimations about the size of subsequent integration steps can be done via calculation of the truncation error. By considering

$$\boldsymbol{y} = [\boldsymbol{Y}, T] \tag{14}$$

a vector of N_s+1 variables, under the assumption of isobaric integration ($dp/dt = 0$) within the time interval $\Delta t_{\text{fluid}}^n \in [t^n; t^{n+1}]$, one has:

$$\begin{aligned}
\boldsymbol{y}_1 &= \boldsymbol{y}^n + \Delta t_{\text{chem}} \left(a_{2,1} \frac{d\boldsymbol{y}^n}{dt} \right) \\
\boldsymbol{y}_j &= \boldsymbol{y}^n + \Delta t_{\text{chem}} \left(a_{j+1,1} \frac{d\boldsymbol{y}^n}{dt} + \sum_{i=2}^{j} \left(a_{j+1,i} \frac{d\boldsymbol{y}_{i-1}}{dt} \right) \right) \qquad j \in [2,5] \\
\boldsymbol{y}^{n+1} &= \boldsymbol{y}^n + \Delta t_{\text{chem}} \left(b_1 \frac{d\boldsymbol{y}^n}{dt} + \sum_{i=2}^{6} \left(b_i \frac{d\boldsymbol{y}_{i-1}}{dt} \right) \right)
\end{aligned} \tag{15}$$

The constants in Eq. (15) come from the corresponding `RKCK45` butcher tableau [32].

3. Variable positioning and spatial discretization

The conservation laws presented in Section 2 are written as:

$$\frac{\partial \boldsymbol{\Psi}}{\partial t} + \nabla \cdot F(\boldsymbol{\Psi}) = \mathbf{S} \tag{16}$$

Since $F(\boldsymbol{\Psi})$ represents the flux function of the conserved set of variables $\boldsymbol{\Psi}$, its determination is fundamental to solve the investigated problem. Finite volume schemes based on collocated grid arrangements are used for the spatial discretization. Variables are defined at the cell centers and their derivatives are computed as follows [30]:

– **Non-linear terms (convective terms)**: the linearization in the control volume is:

$$\nabla \cdot (\boldsymbol{U}\boldsymbol{\Psi}) \simeq \frac{1}{V_P} \sum_f \boldsymbol{S}_f \cdot \boldsymbol{U}_f \boldsymbol{\Psi}_f = \frac{1}{V_P} \sum_f \boldsymbol{\Phi}_f \boldsymbol{\Psi}_f \tag{17}$$

being V_P the volume of the polyhedral cell P, and $\boldsymbol{S}_f$ the surface of the fth face of the cell. $\sum_f$ defines a sum over the cell faces, and $\boldsymbol{S}_f \cdot \boldsymbol{U}_f$ is the volume of fluid flowing through the face per second. To accurately catch discontinuities such as slip lines and shock waves in supersonic cases, the physical flux is replaced by a numerical one $F(\boldsymbol{\Psi}_f^+, \boldsymbol{\Psi}_f^-)$. $\boldsymbol{\Psi}_f^+$ and $\boldsymbol{\Psi}_f^-$ respectively denote the positive- and negative-sided interpolations of $\boldsymbol{\Psi}$ at the face f based on the direction of the face normal vector (see Fig. 1(a)). In the current approach, the Kurganov-Noelle-Petrova (KNP) central upwind method is used. It follows:

$$\sum_f \boldsymbol{\Phi}_f \boldsymbol{\Psi}_f = \sum_f \left[\alpha \boldsymbol{\Phi}_f^+ \boldsymbol{\Psi}_f^+ + (1-\alpha) \boldsymbol{\Phi}_f^- \boldsymbol{\Psi}_f^- + \omega_f \left(\boldsymbol{\Psi}_f^- - \boldsymbol{\Psi}_f^+ \right) \right] \tag{18}$$

where α is determined as:

$$\alpha = \frac{\psi_f^+}{\psi_f^+ + \psi_f^-} \tag{19}$$

based on one-sided local speeds of propagation, with a bias in the upwind direction. The volumetric fluxes associated with the local speeds of propagation are calculated as follows:

$$\psi_f^+ = \max \left(c_f^+ \left| \boldsymbol{S}_f \right| + \boldsymbol{U}^+ \cdot \boldsymbol{S}_f, c_f^- \left| \boldsymbol{S}_f \right| + \boldsymbol{U}^- \cdot \boldsymbol{S}_f, 0 \right) \tag{20}$$

$$\psi_f^- = \max \left(c_f^+ \left| \boldsymbol{S}_f \right| - \boldsymbol{U}^+ \cdot \boldsymbol{S}_f, c_f^- \left| \boldsymbol{S}_f \right| - \boldsymbol{U}^- \cdot \boldsymbol{S}_f, 0 \right) \tag{21}$$

In the two definitions, $c = \sqrt{\gamma p/\rho}$ is the local speed of sound. Also, the third contribution of Eq. (18) is needed only when the convective term comes from a substantive derivative; it includes a volumetric flux ω_f which depends on the maximum propagation speed of any discontinuity between the positive- and negative-sided interpolated values measured at the face f. For the KNP method:

$$\omega_f = \alpha(1-\alpha)(\psi_f^+ + \psi_f^-) \tag{22}$$

Simultaneously, the interpolation on face f of the $\boldsymbol{\Psi}$ variables is carried out using a Total Variation Diminishing (TVD) scheme. For a generic scalar quantity, at the fth face of a cell, the linear interpolation between the *neighbor* ($\bar{\boldsymbol{\Psi}}_N$) and *owner* ($\bar{\boldsymbol{\Psi}}_P$) cell-averaged values produces:

$$\boldsymbol{\Psi}_f = w_f \bar{\boldsymbol{\Psi}}_P + (1-w_f) \bar{\boldsymbol{\Psi}}_N \tag{23}$$

in which w_f and $(1-w_f)$ represent the central-difference weights. From this, the TVD interpolation yields:

$$\boldsymbol{\Psi}_f^+ = \bar{\boldsymbol{\Psi}}_P + (1-w_f)\, \beta(r_{f,P}) \left(\bar{\boldsymbol{\Psi}}_N - \bar{\boldsymbol{\Psi}}_P \right) \tag{24}$$

$$\boldsymbol{\Psi}_f^- = \bar{\boldsymbol{\Psi}}_N + w_f\, \beta(r_{f,N}) \left(\bar{\boldsymbol{\Psi}}_P - \bar{\boldsymbol{\Psi}}_N \right) \tag{25}$$

in which the Van Leer limiter

$$\beta(r) = \frac{|r| + r}{1 + r} \tag{26}$$

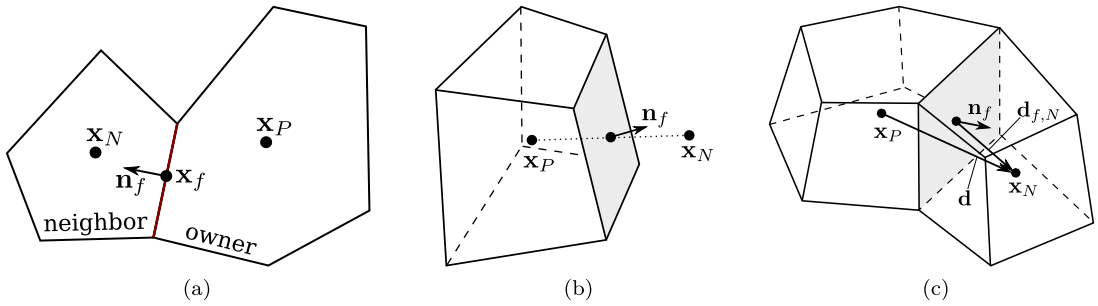


Fig. 1. (a) definition of the positive and negative sides of an arbitrary cell; (b) collocated grid arrangement of primary variables: velocity fluxes are interpolated at the face center; (c) computation of the surface gradient: non-orthogonal correction [30].

depends on the factors $r_{f,P}$ and $r_{f,N}$ that are measures of the relative smoothness of the solution. For TVD, being $\boldsymbol{d} := \boldsymbol{x}_N - \boldsymbol{x}_P$ the vector connecting the *owner* and the *neighbor* cell center, one has:

$$r_{f,i} = 2\left(\frac{\mathbf{d}\cdot(\nabla\boldsymbol{\Psi})_i}{\bar{\boldsymbol{\Psi}}_N - \bar{\boldsymbol{\Psi}}_P}\right) - 1 \quad \text{where} \quad i = P, N \tag{27}$$

– **Gradient terms**. By means of the Green-Gauss theorem, one has:

$$\nabla\boldsymbol{\Psi}_P \simeq \frac{1}{V_P}\sum_f \boldsymbol{\Psi}_f \boldsymbol{S}_f \tag{28}$$

By using the KNP scheme, the interpolation procedure is split into the positive- and negative-sided terms so that:

$$\sum_f \boldsymbol{\Psi}_f \boldsymbol{S}_f = \sum_f \left(\alpha\boldsymbol{\Psi}_f^+ \boldsymbol{S}_f + (1-\alpha)\boldsymbol{\Psi}_f^- \boldsymbol{S}_f\right) \tag{29}$$

The same aforementioned limiter (Eq. (26)) is used for the interpolation.

– **Diffusive term (Laplacian)**. By considering the diffusion coefficient Γ, the diffusive term $\nabla\cdot(\Gamma\nabla\boldsymbol{\Psi})$ can be manipulated into:

$$\begin{aligned}\int_V \nabla\cdot(\Gamma\nabla\boldsymbol{\Psi})\,dV &= \int_S (\Gamma\nabla\Psi)_f \cdot \boldsymbol{n}dS \simeq \sum_f \Gamma_f \boldsymbol{S}_f\cdot(\nabla\boldsymbol{\Psi})_f \\ &= \sum_f \Gamma_f |\boldsymbol{S}_f| \nabla_n \boldsymbol{\Psi}_f \end{aligned} \tag{30}$$

where $\nabla_n\boldsymbol{\Psi}_f$ is the surface normal gradient of $\boldsymbol{\Psi}$. The subscript f in Eq. (30) indicates the cell-to-face interpolated quantities. For the non-orthogonal grid of Fig. 1(b) in a collocated variable arrangement, the surface gradient $\boldsymbol{S}_f\cdot(\nabla\boldsymbol{\Psi})_f$ is decomposed into an orthogonal part (function of the owner and neighbor cell values) and a non-orthogonal correction (full gradient, see Fig. 1(c)):

$$\boldsymbol{S}_f\cdot(\nabla\boldsymbol{\Psi})_f = \underbrace{A(\bar{\boldsymbol{\Psi}}_N - \bar{\boldsymbol{\Psi}}_P)}_{\text{orthogonal}} + \underbrace{\boldsymbol{B}(\nabla\boldsymbol{\Psi})_f}_{\text{non-orthogonal}} \tag{31}$$

being $A := |\boldsymbol{S}_f|^2/(\boldsymbol{S}_f\cdot\boldsymbol{d})$, and $\boldsymbol{B} := \boldsymbol{S}_f - A\boldsymbol{d}$.

4. Solution method

The GPGPU solver employs the solution of the flow transport on the CPU (host), while the ODEs describing the finite-rate chemistry problem are solved on the GPU (device), see Fig. 2. In the CUDA framework there are three levels of tasks, namely the grid, the block and the thread (Fig. 3(b)). Blocks can be handled asynchronously by the same Streaming Multiprocessor (SM); being all the resources between blocks shared, the communication among blocks is expensive. Each block can execute a certain number of threads. There is only a lightweight synchronization overhead between the threads in a block.

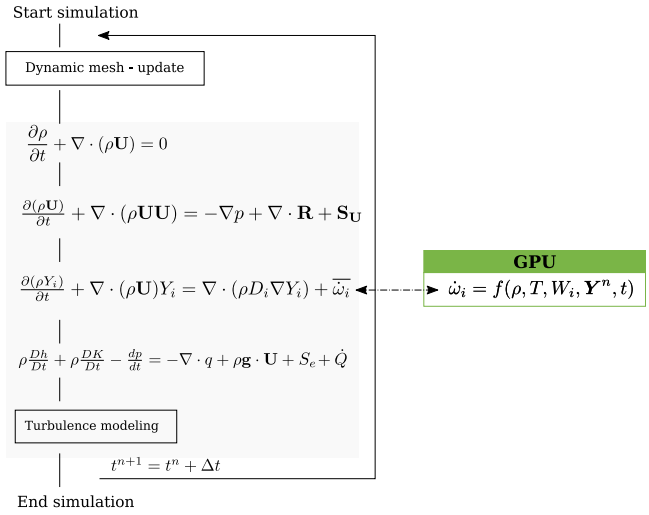
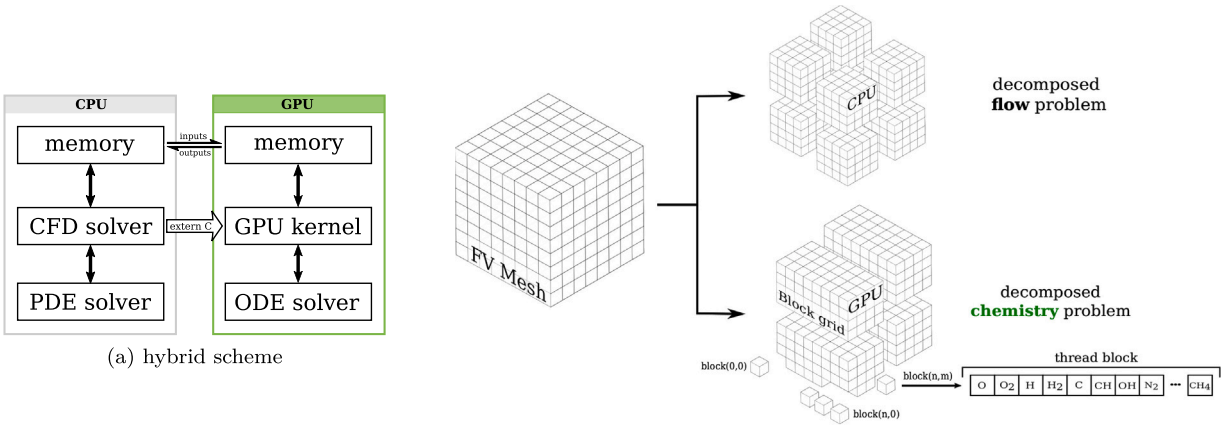


Fig. 2. Structure of the shock-capturing reactive flow solver `hyperFoam` used for the simulations.

All threads in a block run in parallel, in the Single Instruction Multiple Threads (SIMT) mode [42]. More precisely, each block contains multiples of 32 threads called warps. Threads in a warp are executed concurrently on a multiprocessor. Modern general-purpose GPUs have a large amount of (slow) global memory and a small amount of (fast) shared memory. Best practice guidelines to improve the performance of a GPU solver suggest: (a) to saturate the GPU with computational work and to balance the load among all the threads; (b) to reduce data transfer/communication as much as possible between CPU and GPU; (c) to limit the access of threads to the global memory, if possible. The chemistry solver presented in this work addresses some of these issues based on profiling outputs. Load balancing, communication overhead, latency, synchronization overhead, and data locality are important factors that may affect performance. To hide latency, asynchronous GPU/CPU data transfer is adopted. To reduce the synchronization overhead, the number of tasks running asynchronously should be maximized. To reduce data transfer, the use of shared memory is quite critical [43]: it limits the threads' access to the global memory and favors an increase in the efficiency of the algorithm, but it might lead to possible threads' divergence [44]. To avoid threads' divergence, the code has been written in branchless form. Also, because of its limited size, the chunk of GPU shared memory is dynamically allocated at the beginning of the simulation and it is used by the GPU for: (a) the progressive explicit update of chemical concentrations and temperature; (b) the calculation of production/consumption rate and (c) the determination of the maximum error. For the methodology proposed in this work, the fat thread approach [45] has been applied with some effects on data structure and organization: data access time is minimized by



(a) hybrid scheme

(b) heterogeneous decomposition

Fig. 3. Heterogeneous ODE integrator for the treatment of chemistry in reactive flow simulations.

relying on shared memory and registers, where data required by the threads are stored. To reduce the communication overhead of data transfers between the CPU and the GPU, time-dependent quantities are stored in the dynamic global memory, they are accessed in a coalesced manner and progressively stored in chunks of shared memory for the amount of time needed for their use: these are defined as dynamic data. Conversely, constant data are stored when the constant memory is allocated, i.e. at the beginning of the simulation only. Molecular weight of the species, stoichiometric coefficients and exponents of the reaction mechanism, and the ODE solver settings are initialized on the host and copied and stored in the GPU's constant memory. Settings of the ODE solver include solution controls (tolerances and maximum number of iterations), scaling factors and time scaling controls. Time-varying quantities are copied in the GPU's cached global memory. Thanks to the optimization of the memory access time and latency from the threads, the fat thread approach results to be very fast because it allows to achieve a double parallelization of the chemistry problem: (a) the ODE system is solved in parallel for the computational cells (on CPUs, this operation is done in serial); (b) for each computational cell (i.e. CUDA block), each species in the reaction mechanism is handled in parallel on multiple active threads (Fig. 3(b)). If the amount of data to be transferred exceeds the maximum memory availability of the GPU(s), the chemical problem is split into mesh chunks, each of them containing a cluster of cells. If the mesh dimension exceeds the maximum number of cells that can be concurrently treated by the GPU streaming multiprocessors, a queue is generated and the overall computational lag and latency is limited thanks to asynchrony. Besides, a GPU block-level control over the cells is done to avoid unnecessary operations. Chemically reactive cells are identified through their local temperature that must be higher than a given threshold. No ODE integration is performed on the other cells, that are cast-off. Finally, the chunk of memory allocated for each GPU block is freed as soon as the relative ODE system is solved to allow the handling of another cell.

5. Validation

A 10-species, 27-reaction H_2–O_2 reaction mechanism has been selected for validation. Hydrogen chemical mechanisms are commonly employed in rocket propulsion applications because they generate a large specific impulse; Hydrogen is also a very good candidate to promote decarbonization and supersonic commercial air transportation. Single-cell batch reactor tests, without flow transport, have been initially employed to test the accuracy of the GPU-ODE chemistry solver against the reference solution from the same solver on the CPU and from Cantera.

Four combustion modes, reported in Table 1, have been selected for verification. For all the tested operating conditions, the calculations

Table 1
Operating conditions (modes) tested for the H_2–O_2 mechanism.

Mode	p (bar)	T (K)	Y_{H_2} (–)	Y_{O_2} (–)	Y_{N_2} (–)
1 [46]	2	1000	0.0145	0.2296	0.7559
2	1	1000	0.0145	0.2296	0.7559
3	1.5	2000	0.0145	0.2296	0.7559
4	0.5	2000	0.0284	0.2296	0.7420

are conducted considering a constant-pressure vessel. Results produced by Cantera [47] are deemed as reference. The following quantities are determined to produce metrics about the accuracy of the novel GPU-ODE integrator:

(a) relative error between the legacy CPU solver and Cantera:

$$\mathrm{err}_{i,CPU} = \frac{\left|\bar{Y}_{i,CAN} - \bar{Y}_{i,CPU}\right|}{\bar{Y}_{i,CAN}} \cdot 100 \tag{32}$$

(b) relative error between the GPGPU solver and Cantera:

$$\mathrm{err}_{i,GPU} = \frac{\left|\bar{Y}_{i,CAN} - \bar{Y}_{i,GPU}\right|}{\bar{Y}_{i,CAN}} \cdot 100 \tag{33}$$

(c) relative error between the GPGPU solver and the CPU counterpart:

$$\mathrm{diff}_i = \frac{\left|\bar{Y}_{i,CPU} - \bar{Y}_{i,GPU}\right|}{\bar{Y}_{i,CPU}} \cdot 100 \tag{34}$$

being:

$$\bar{Y}_i = \frac{1}{\Delta t}\int_0^t Y_i \, dt \tag{35}$$

the cumulative chemical mass fraction of a specie. Comparisons of the predicted flow temperature and the evolution in time of the mass fractions of H_2O_2 and H_2O (intermediate and product species respectively) are reported in Fig. 4. The temperature and mass fraction reference solutions are well matched by the implicit and explicit solutions of the CPU ODE integrator and by the GPU-ODE approach. Relative errors for the operating conditions simulated (modes of Table 1) are reported in Fig. 5. Discrepancies between the GPU-ODE integrator and the explicit CPU solutions are always lower than 0.2%, see Fig. 5(b), 5(d), 5(f), 5(h), and clearly are not affecting the accuracy of the results; their presence can be attributed to the round-off error propagation linked to the different parallelization employed on the different architectures. As discussed in Section 4, the GPU-ODE solver must be explicit to take advantage of the strong vectorization promoted by the architecture of the accelerators. A fair analysis must include the best of both

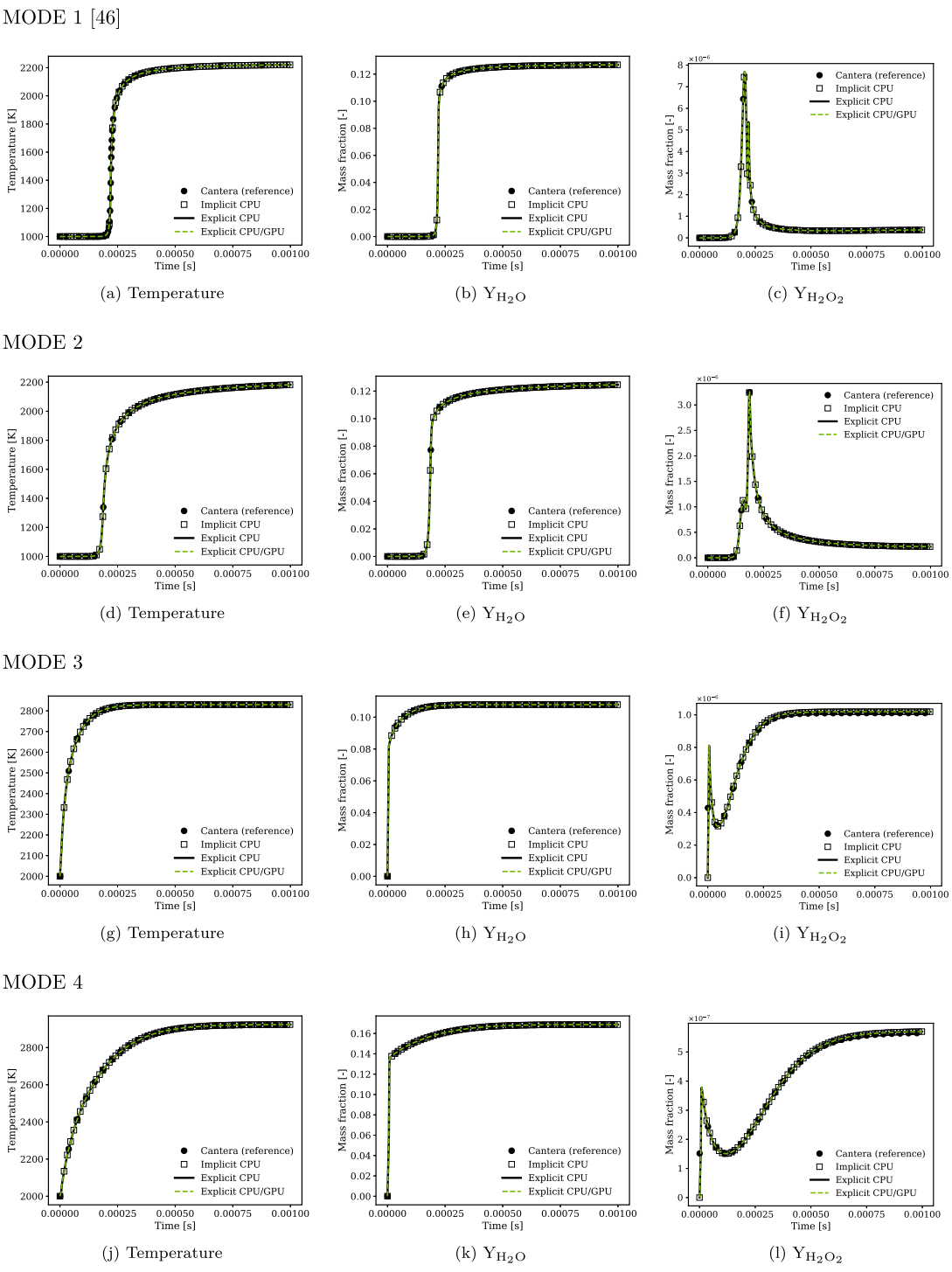


Fig. 4. Simulations of the auto-ignition of $H_2 - O_2$ in a batch reactor at the four operating modes reported in Table 1. ODEs integrations conducted at constant pressure.

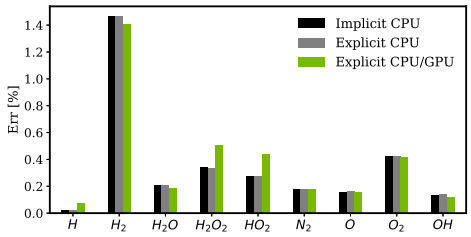
technologies: the speed of computation of the implicit ODE integrators running on CPUs will be therefore compared in the final section against the explicit GPGPU solver.

6. Simulation of the supersonic combustion in a scramjet engine

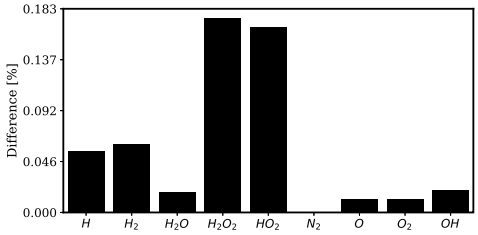
The three-dimensional simulation of supersonic combustion in a scramjet engine by the DLR combustor facility [35–37] is used to validate the proposed GPGPU solver. Geometrical features of the scramjet engine geometry are reported in [11,16,48] and summarized in Fig. 6. The configuration is made of a one-sided divergence channel that confines preheated air and a wedge-shaped flame stabilizer. The upper wall diverges to compensate the expansion of the boundary layer.

The height of the combustor at the entrance is 50 mm and the length of the rectangular region is 100 mm. The overall length of the combustor is 340 mm. The divergence angle is 3°. Vitiated air enters the combustor and mixes with the fuel injected from the holes cut in the wedge. The strut injector is located at the centerline of the inlet section at 25 mm from the bottom wall, and it starts at 77 mm downstream of the entrance. The wedge structure measures 32 mm in length and has a semi-open angle of 6°. Thus, fuel injection is at 109 mm from the reference origin along the x direction. The width of the overall

MODE 1 [46]

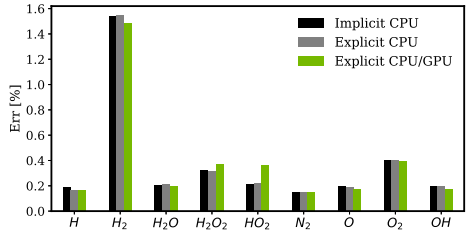


(a) CPU and hybrid integration vs. Cantera

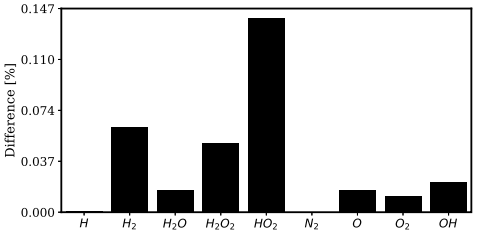


(b) explicit CPU vs. hybrid integration

MODE 2

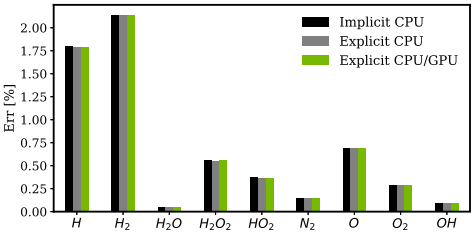


(c) CPU and hybrid integration vs. Cantera

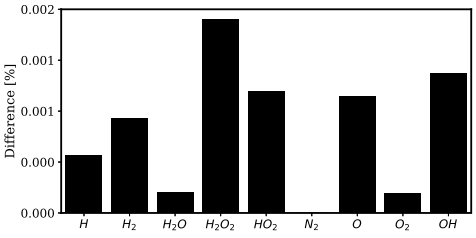


(d) explicit CPU vs. hybrid integration

MODE 3

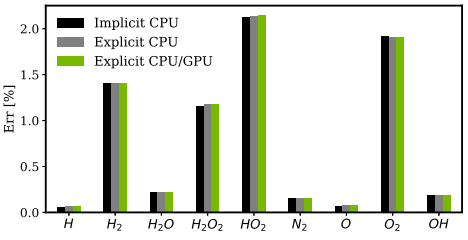


(e) CPU and hybrid integration vs. Cantera

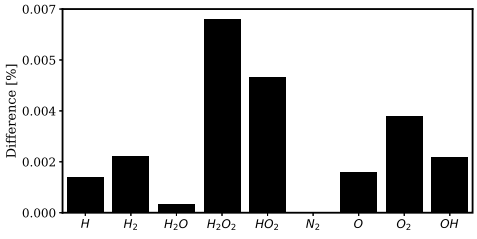


(f) explicit CPU vs. hybrid integration

MODE 4



(g) CPU and hybrid integration vs. Cantera



(h) explicit CPU vs. GPGPU

Fig. 5. Relative errors (see Eqs. (32)–(34)) of different ODE integrators for the operating modes described in Table 1. ODEs integrations are at constant pressure.

configuration is constant and measures 51 mm. Each fuel injection hole has a diameter of 1 mm; each of the 15 circular holes is separated by 2.4 mm.

Boundary conditions of the problem are summarized in Table 2. Fixed values of pressure, velocity and temperature are set at the air and fuel inlets. The inlet air enters into the domain with Ma = 2; it is preheated and includes a fraction of water in gaseous form. Hydrogen is injected by the circular fuel inlets at Ma = 1. The chamber and wedge walls are adiabatic. In the literature, the combustor has been analyzed considering one [11,48], three [16], and five [16] of the fifteen injectors, neglecting the effects of the side walls. In the current study, the three-nozzle configuration [16] has been considered. The

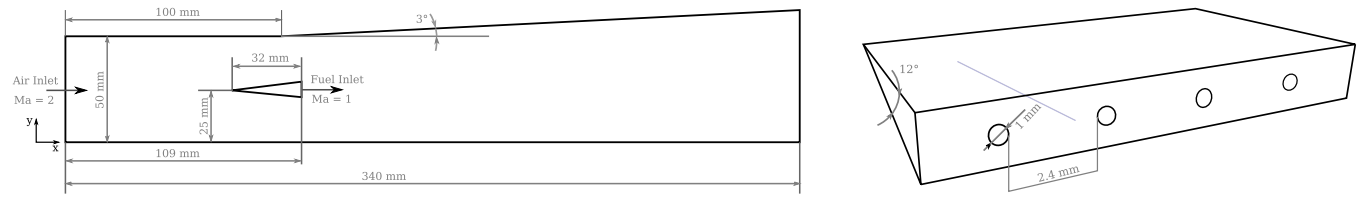


Fig. 6. Geometry of the three-dimensional scramjet test case.

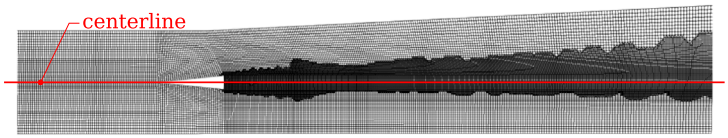


Fig. 7. Body-fitted hexahedral Finite-Volume (FV) mesh of the scramjet engine. Adaptive Mesh Refinement (AMR) is dynamically applied at run-time in proximity of large temperature gradients between neighboring cells. The number of cell elements ranges between 1.5 M (initial mesh) and 15 M.

(a) experiments (b) simulations

Fig. 8. Shock wave pattern in the scramjet engine geometry (cold-flow case, no fuel injection). Comparison between simulations and experiments [11,17].

Table 2
Case setup: boundary conditions applied for the simulation of the scramjet engine.

	U [m/s]	T_0 [K]	p [Pa]	Y_{N_2} [–]	Y_{H_2} [–]	Y_{H_2O} [–]	Y_{H_2} [–]
Air	730	600	10^5	0.736	0.232	0.032	0
Fuel	1200	300	10^5	0	0	0	1

boundary layers of the upper and lower walls are not resolved [11]; this aspect was out of scope for the present work and requires further studies.

The computational domain is reported in Fig. 7. In regions with large temperature gradients, Adaptive Mesh Refinement (AMR) is dynamically applied at run-time to the initial body-fitted hexahedral mesh of 1.5 M cell elements. The flow field is initialized by a precursor cold-flow simulation to reproduce the complex shock wave pattern (duration: $1.5 \cdot 10^{-3}$ s); the duration of the reactive simulation is $5 \cdot 10^{-3}$ s. A hot spot in the recirculating region is set to ignite the mixture.

6.1. Shock wave pattern in the scramjet engine

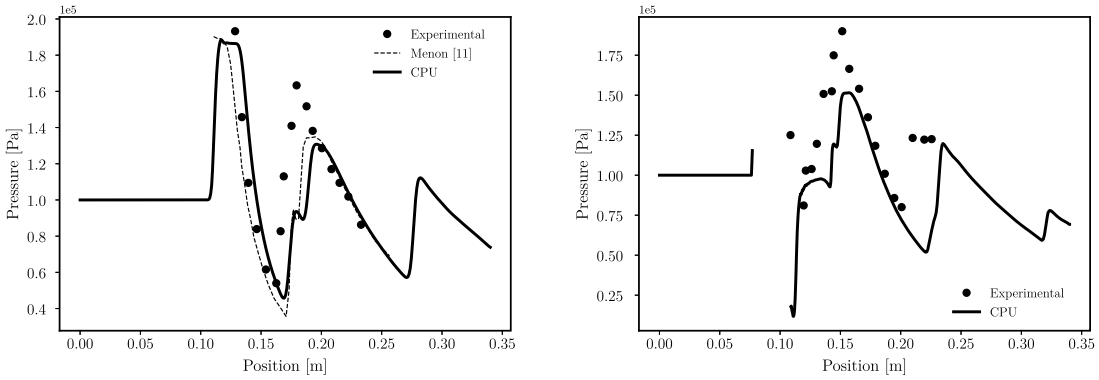
Predictions from non-reactive simulations to determine the shock wave pattern, with and without fuel injection, are compared against experimental Schlieren images [11,17]. The supersonic air from the inlet is deflected at the tip of the wedge; two shocks are formed defining an initial symmetrical behavior on each side; after impacting the upper and lower walls, they are reflected towards the center. Because of the divergence at the upper wall, the trajectories of the shocks become asymmetric. In proximity of the corners, two expansion fans are generated as a result of flow divergence. The flow at the walls of the wedge also separates. In absence of fuel injection (Fig. 8), a subsonic triangular recirculation zone forms behind the wedge structure; in this region, the mixing of the fuel with the surrounding vitiated air is favored also when the fuel injection is activated. Due to the low pressure in the recirculation region, the two originated shear layers tend to converge towards the centerline (see Fig. 8). For the same reason, the jet expands into a diamond-shaped structure as the fuel is injected (Fig. 9).

The interaction between the fuel and the shear layers produces a series of compression waves. Further downstream, the upper oblique wave merges with the transmitted bottom one. A slip line is formed and it propagates towards the upper wall. At the bottom, the compression waves originated from the shear layer interact with the expansion fan; reflected shocks intersect once more further downstream. Fuel injection influences the flow behavior at the centerline and the reflection of the shocks downstream of the wedge: the averaged flow behavior shows that the wake slightly deviates from the centerline and moves upwards, with a thickness that progressively enlarges towards the end of the considered domain (Fig. 9).

Pressure, temperature, and velocity fields were also available from the experiments [35–37]. Pressure values were collected near the bottom wall and over the centerline. At the wall, an initial decrease of pressure is linked to the presence of the expansion fan from the lower corner of the wedge. Then, the pressure increases where the lower wall is impacted by: (a) the primary shock and (b) the shock transmitted from the top. Good agreement is found against experimental solutions available in [11,16,49]. A small drop in the computed pressure for $x \simeq 0.19$ m is observed, and the pressure peak further downstream is underestimated (Fig. 10(a)). The experimental results show only one peak (upwards) located at $x \simeq 0.19$ m. The observed differences are linked to: (a) the recompression of the shear layer which occurs slightly more downstream in the computational simulation; (b) the separation of the boundary layer at the bottom wall as a consequence of the shock/boundary layer interaction, which is unresolved in the computational analysis. In the simulations, the two collapsed wake systems impinge the bottom wall at two slightly different positions, as shown in Fig. 8. That affects the location and the magnitude of the peak. Results of Fig. 10(a) are also in good agreement with [11]. At the centerline, the distribution of pressure has marginal discrepancies in correspondence of the two peak values ($x \simeq 0.16$ m and $x \simeq 0.24$ m): this can be explained as the consequence of a small difference in the estimation of the vertical position of the intersection of the two major shock systems. Such variation marginally influences the first peak value at the centerline and delays the second one. These results agree with those of [11,13,49].

(a) experiments (b) simulations

Fig. 9. Shock wave pattern in the scramjet engine geometry (with fuel injection). Comparison between cold-flow simulations and experiments [11,17]. Time-averaged gradients of density are marked by dark lines.



(a) bottom wall (b) centerline

Fig. 10. Scramjet engine, absolute pressure of the flow at: (a) bottom wall (b) centerline. Simulations are non-reactive.

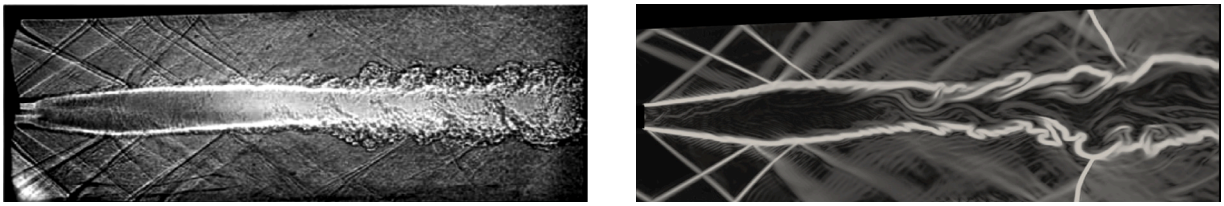


Fig. 11. Comparison of density gradients for the hot simulation: Schlieren image [11] (left) vs. numerical solution (right).

6.2. Simulation of the supersonic combustion in the scramjet engine

Starting from the flow field of the precursor simulation, reactive flow computations have been carried out. The ignition of the reactive mixture is very challenging in supersonic flow conditions. In this work, an ignition point (hot spot) in the recirculating region has been set to trigger the combustion. The evolution of the reactive region modifies the shock pattern behind the struct injector, while the pattern of the waves upstream is unchanged (see Fig. 15). The formation of recirculation regions favors flame stabilization. The supersonic flow impinging the struct continues to produce two oblique shocks that, in turn, impinge the bottom and the upper walls and bounce back towards the central region. The core flow at the center is subjected to augmented turbulence because of the occurring combustion. For this reason, the regularity of the shock wave pattern in the far region of the combustor is disrupted, as the core region enlarges. A comparison between an experimental Schlieren image and the correspondent numerical density gradient plot is reported in Fig. 11 for the reactive simulation. The rectangular region of Fig. 11 comes from the area marked by a dotted line in Fig. 9(a). The representation shows a good agreement for both the core development and the core enlargement.

The behavior of the pressure at the base largely varies when compared to the non reactive case. The pressure slightly increases (see Fig. 12); the core wake becomes quasi parallel to the freestream flow as observed in [11], thus only small waves are observed further downstream. From this, it is derived that only small variations in pressure can be experienced towards the outlet of the combustor.

The flowfield has been quantitatively measured by DLR using particle image displacement velocimetry and Laser Doppler Velocimetry (LDV). Experimental results of the velocity available at x = 11 mm, 58 mm, and 99 mm from the fuel inlet section [11,16,17] were used for code validation (Fig. 13). Computed mean flow quantities were time-averaged over 10 flow-through times (the first flow-though time was discarded). The velocity profile at the first location slightly deviates from the experiments in the core region (Fig. 13(a)), but is in accordance with other numerical results from different approaches. The extent of the recirculation region seems to be marginally overpredicted in the numerical simulation. The minimum velocity in the core region is lower than its measured counterpart. Nonetheless, similar observations have been derived in the LES studies that have been published in the literature [11,13,48]. Further downstream, the agreement improves (Figs. 13(b) and 13(c)). The different location of the predicted velocity drop (and peak) along the y direction (Fig. 13(c)) is still attributed to the poor modeling of the boundary layer.

The temperature is investigated quantitatively by comparing Coherent anti-Stokes Raman spectroscopy solutions at x = 11 mm, 58 mm, and 166 mm (Fig. 14). The temperature peaks predicted at the first location are slightly higher than the measured counterparts; the symmetric profile of the experiments is correctly captured. A better agreement is appreciable downstream (Figs. 14(b) and 14(c)), even though the region where combustion takes place is slightly larger than the experimental one (Fig. 14(b)). The reactive simulation is run once more by using the GPU-ODE integrator, thus exploiting an heterogeneous application. Results in green contained in Fig. 14 confirm the good agreement against full-CPU solutions used as reference.

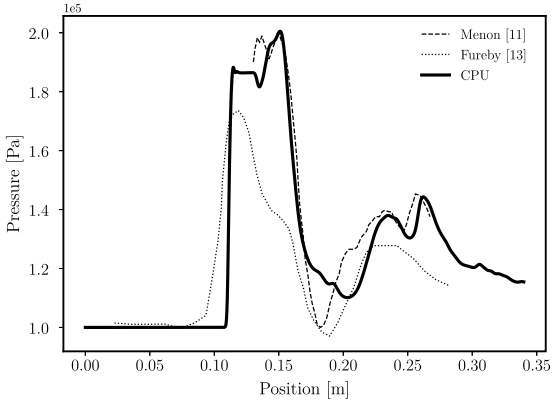
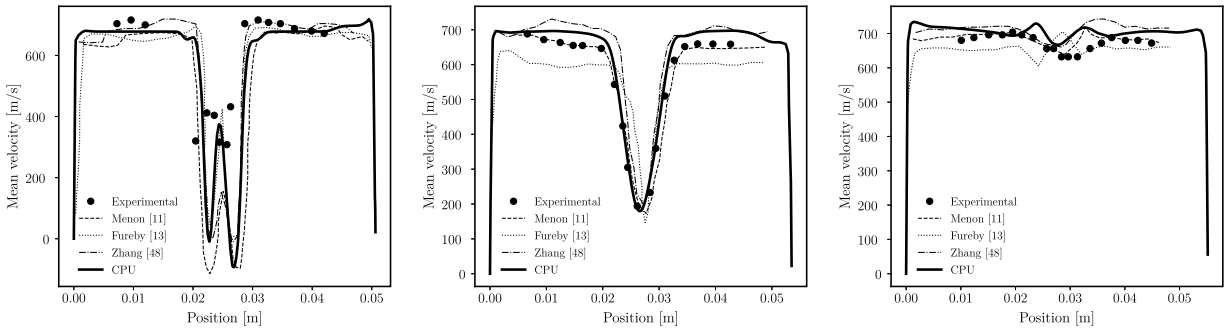
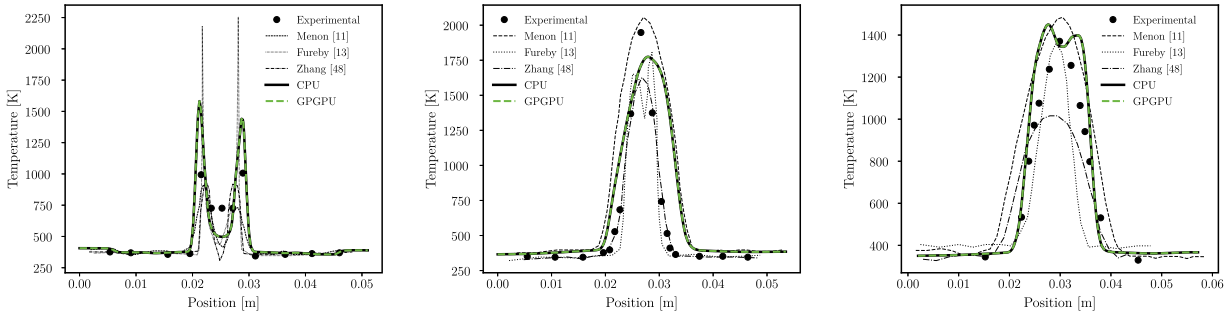


Fig. 12. Pressure distribution at the bottom wall for the reactive flow simulation. Comparison between simulations and experiments [11].



(a) x = 11 mm (b) x = 58 mm (c) x = 90 mm

Fig. 13. Time-averaged mean axial velocity at different positions along the flowpath with supersonic combustion.



(a) x = 11 mm (b) x = 58 mm (c) x = 106 mm

Fig. 14. Mean temperature [K] at different positions along the flowpath.

Finally, Fig. 15 (upper line) reports the temperature flow distribution within a threshold and is used to highlight the shape of the flame at two different timesteps. The shape of the flame clearly highlights the presence of flow instabilities and small recirculating regions at the interface. Large recirculation vortices are present within the core region: they are generated behind the struct and carried towards the end of the domain as they evolve. The penetration of the fuel jet is lower than in the non-reacting case. Again, it is clear how the shear layer instabilities generated at the corners of the flame stabilizer do not converge towards the centerline, but they interact with the recirculation bubble behind the struct.

7. Performance

The scalability of the GPGPU solver `hyperFoam` is investigated considering the initial coarse mesh used for the supersonic test case over a span of the first 100 time-steps (the time to read the mesh from disk is included). The AMR is not active in this test, so the grid counts a limited number of cells (1.5 M). The scalability for the cold-flow simulation is linear only within a limited range (up to about 24 cores, Fig. 16(a)). The same holds if chemical species are tracked without reactions (Fig. 16(b)). Finally, if combustion with finite-rate chemistry is triggered, the computational load increases because: (a) the number of convection–diffusion equations is larger due to the tracking of the intermediate species; (b) the solution of finite-rate chemistry ODEs is now active. In this case, linear scalability is preserved for a higher number of cores (Fig. 16(c)).

The use of GPGPU solver is advantageous if:

$$t_{CPU} > t_{GPU} = t_{f,GPU} + t_{k,GPU} + t_{r,GPU} \tag{36}$$

being t_{CPU} and t_{GPU} the times to solve the finite-rate chemistry problem on the CPU and the GPU respectively; $t_{f,GPU}$ is the time taken to allocate the GPU memory, collect the CPU data and perform the

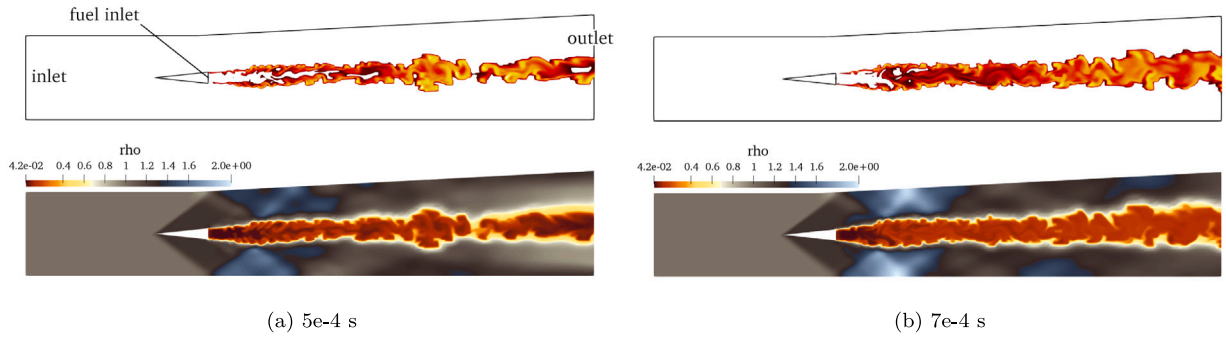


Fig. 15. Combustion simulation at two different time steps; representation of the flame via threshold (top); density field (bottom).

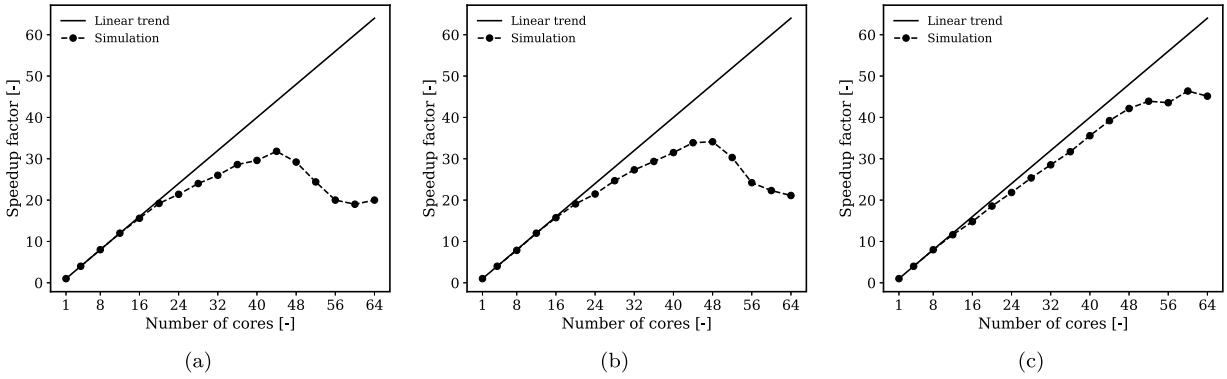


Fig. 16. Scalability of the reactive flow solver on the initial coarse mesh (1.5 M cells, AMR deactivated): (a) cold-flow simulation; (b) cold-flow simulation with specie-transport; (c) reactive flow simulation with finite-rate chemistry.

forward data transfer (CPU-to-GPU); $t_{k,GPU}$ is the time required for the kernel call and the actual ODE integration via GPU; $t_{r,GPU}$ is the time to complete the backward data transfer (GPU-to-CPU). From Eq. (36), it is apparent that the speedup due to the vectorization employed by the GPU during the calculations of the chemistry problem becomes more favorable as the size of the problem increases. For the reactive flow simulation, the GPGPU solver using 128 cores and a Nvidia V100 GPU card proved to be 9.3X faster with respect to the same solver fully running on the same 128 CPU cores.

8. Conclusions

An explicit time-step-adaptive ODE solver has been re-designed to work on accelerators (GPU) and to be efficiently coupled to a shock-capturing density-based solver for the solution of supersonic flows. The co-design of the GPGPU strategy allows to improve the efficiency and performance of combustion calculations, thanks to a three-level parallelization where the fluid dynamic problem is split over multiple CPU processor cores, while the chemistry problem is solved on the hardware accelerators in clusters of cells. On the GPU, blocks and threads execute the simultaneous/parallel solution of the reaction mechanism. As a result, the explicit 5th order Runge–Kutta method employs the parallel integration of the chemistry ODEs on GPUs with significant speedups if compared to the corresponding CPU-based version. Main features of the proposed methodology are: (a) it is fully automatic, it does not require any manual specific operation for pre-processing; (b) its efficiency increases as the size and the stiffness of the kinetic mechanism becomes large; (c) it can work on multiple CPUs/GPUs for high-fidelity simulations, but it also results advantageous when applied to small/medium size problems, since it makes use of the full potential of the hardware of modern workstations. The strategy has been applied to the simulation of supersonic combustion of Hydrogen. Validation tests against solutions from established CPU ODE integrators show a very good accordance of the results, with speedups that are proportional to the size of the mechanism. Finally, the CFD simulation of a supersonic combustion scramjet engine is presented. The GPGPU approach is able to well reproduce the experimental trends and provide results comparable to traditional solvers. The speedup achieved for this set of tests is 9.3X. The performance gain is limited by: (a) the slow data transfer between the CPU and the GPUs, occurring at each time step; (b) the stiffness of the ODE system: to take advantage of the parallel architecture of the GPU, the use of explicit integrators is favored. The GPU ODE solver for the finite-chemistry problem is developed in the form of a dynamic object-oriented C++/CUDA library. The proposed GPU-ODE chemistry integrator can be combined with any flow solver (compressible, multiphase, ecc.) based on the operator-splitting technique, where the solution of chemistry is decoupled from the fluid transport.

Current work is about the implementation of the `amgx4Foam` library. Structured on `AmgX` [50], it solves the linear algebra of the Partial Differential Equations via GPU.

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.

Acknowledgments

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956416. The JU receives support from the European Union's

Horizon 2020 research and innovation programme and France, United Kingdom, Germany, Italy, Croatia, Spain, Greece, Portugal. Authors also gratefully acknowledge the Laboratory Computing Resource Center (LCRC) at the Argonne National Lab. (Lemont, US) for the computing resources provided. The software presented in this paper has been developed by the authors at the Dept. of Aerospace Science and Technology (DAER) at Politecnico di Milano (PoliMi), in the form of open-source C++ dynamic libraries to be linked to the most recent official versions of the software OpenFOAM released by the OpenFOAM Foundation (OF-8, OF9, OF-10, OF-dev) [34] and ESI-OpenCFD (OF-v2206, OF-v2212) [51].

References

[1] Viola N, Fusaro R, Vercella V. Technology roadmapping methodology for future hypersonic transportation systems. Acta Astronaut 2022;195:430–44. http://dx.doi.org/10.1016/j.actaastro.2022.03.038.

[2] Smart M. Scramjets. Aeronaut J 2007;111(1124):605–19. http://dx.doi.org/10.1017/S0001924000004796.

[3] Ferguson F, Dasque N, Dhanasar M, Blankson IM. The design of scramjet engine configurations for optimal operational temperature and overall engine efficiency. In: 54th AIAA aerospace sciences meeting. 2016, http://dx.doi.org/10.2514/6.2016-0913.

[4] Kanda T, Hiraiwa T, Izumikawa M, Mitani T. Measurement of mass capture ratio of scramjet inlet models. In: 41st aerospace sciences meeting and exhibit. 2003, http://dx.doi.org/10.2514/6.2003-11.

[5] Curran E, Heiser W, Pratt D. Fluid phenomena in scramjet combustion systems. Annu Rev Fluid Mech 1996;28(1):323–60.

[6] kyun Im S, Do H. Unstart phenomena induced by flow choking in scramjet inlet-isolators. Prog Aerosp Sci 2018;97:1–21. http://dx.doi.org/10.1016/j.paerosci.2017.12.001.

[7] Krishnan L, Sandham N, Steelant J. Shock-wave/boundary-layer interactions in a model scramjet intake. AIAA J 2009;47(7):1680–91. http://dx.doi.org/10.2514/1.41107.

[8] Sandham ND, Schülein E, Wagner A, Willems S, Steelant J. Transitional shock-wave/boundary-layer interactions in hypersonic flow. J Fluid Mech 2014;752:349–82. http://dx.doi.org/10.1017/jfm.2014.333.

[9] Yue L, Jia Y, Xu X, Zhang X, Zhang P. Effect of cowl shock on restart characteristics of simple ramp type hypersonic inlets with thin boundary layers. Aerosp Sci Technol 2018;74:72–80. http://dx.doi.org/10.1016/j.ast.2017.12.018.

[10] de Siqueira J, Rosa M, Ribeiro G. Numerical evaluation of effects of the isolator height and the leading-edge bluntness on a scramjet inlet. J Braz Soc Mech Sci Eng 2020;42. http://dx.doi.org/10.1007/s40430-020-02352-z.

[11] Génin F, Menon S. Simulation of turbulent mixing behind a strut injector in supersonic flow. Aiaa J - AIAA J 2010;48:526–39. http://dx.doi.org/10.2514/1.43647.

[12] Cao D, Brod HE, Yokev N, Michaels D. Flame stabilization and local combustion modes in a cavity-based scramjet using different fuel injection schemes. Combust Flame 2021;233:111562. http://dx.doi.org/10.1016/j.combustflame.2021.111562.

[13] Berglund M, Fureby C. LES of supersonic combustion in a scramjet engine model. Proc Combust Inst 2007;31(2):2497–504. http://dx.doi.org/10.1016/j.proci.2006.07.074.

[14] Hawkes ER, Sankaran R, Sutherland JC, Chen JH. Direct numerical simulation of turbulent combustion: fundamental insights towards predictive models. J Phys Conf Ser 2005;16:65–79. http://dx.doi.org/10.1088/1742-6596/16/1/009.

[15] Seleznev R, Surzhikov S, Shang J. A review of the scramjet experimental data base. Prog Aerosp Sci 2019;106:43–70. http://dx.doi.org/10.1016/j.paerosci.2019.02.001.

[16] Potturi A, Edwards J. Investigation of subgrid closure models for finite-rate scramjet combustion. In: 43rd Fluid dynamics conference. 2013, p. 1–10. http://dx.doi.org/10.2514/6.2013-2461.

[17] Cao C, Ye T, Zhao M. Large eddy simulation of hydrogen/air scramjet combustion using tabulated thermo-chemistry approach. Chin J Aeronaut 2015;28(5):1316–27. http://dx.doi.org/10.1016/j.cja.2015.08.008.

[18] Kee RJ, Coltrin ME, Glarborg P. Chemically reacting flow. theory and practice. John Wiley, New York; 2003.

[19] Liang L, Stevens JG, Raman S, Farrell JT. The use of dynamic adaptive chemistry in combustion simulation of gasoline surrogate fuels. Combust Flame 2009;156(7):1493–502. http://dx.doi.org/10.1016/j.combustflame.2009.02.008.

[20] Goldin GM, Ren Z, Zahirovic S. A cell agglomeration algorithm for accelerating detailed chemistry in CFD. Combust Theory Model 2009;13(4):721–39. http://dx.doi.org/10.1080/13647830903154542.

[21] Singer MA, Pope SB. Exploiting ISAT to solve the reaction–diffusion equation. Combust Theory Model 2004;8(2):361–83. http://dx.doi.org/10.1088/1364-7830/8/2/009.

[22] Li Z, Lewandowski MT, Contino F, Parente A. Assessment of on-the-fly chemistry reduction and tabulation approaches for the simulation of moderate or intense low-oxygen dilution combustion. Energy Fuels 2018;32(10):10121–31. http://dx.doi.org/10.1021/acs.energyfuels.8b01001.

[23] Blasco J, Fueyo N, Dopazo C, Ballester J. Modelling the temporal evolution of a reduced combustion chemical system with an artificial neural network. Combust Flame 1998;113(1):38–52. http://dx.doi.org/10.1016/S0010-2180(97)00211-3.

[24] Nikitin V, Karandashev I, Malsagov MY, Mikhalchenko E. Approach to combustion calculation using neural network. Acta Astronaut 2022;194:376–82. http://dx.doi.org/10.1016/j.actaastro.2021.10.034.

[25] Ji W, Qiu W, Shi Z, Pan S, Deng S. Stiff-PINN: Physics-informed neural network for stiff chemical kinetics. J Phys Chem A 2021;125(36):8098–106. http://dx.doi.org/10.1021/acs.jpca.1c05102.

[26] Tap F, Schapotschnikow P. Efficient combustion modeling based on tabkin® CFD look-up tables: A case study of a lifted diesel spray flame. In: SAE technical paper. SAE International; 2012, http://dx.doi.org/10.4271/2012-01-0152.

[27] Haidar A, Brock B, Tomov S, Guidry M, Billings JJ, Shyles D, Dongarra JJ. Performance analysis and acceleration of explicit integration for large kinetic networks using batched GPU computations. In: 2016 IEEE High performance extreme computing conference (HPEC). 2016, p. 1–7.

[28] Wilt N. the CUDA handbook: A comprehensive guide to GPU programming. Addison-Wesley; 2013, URL https://books.google.it/books?id=KUxsAQAAQBAJ.

[29] Kurganov A, Guergana A, Petrova S. Semidiscrete central-upwind schemes for hyperbolic conservation laws and hamilton–Jacobi equations. J Comput Phys SIAM J Sci Comput 2000;23:707–40. http://dx.doi.org/10.1137/S1064827500373413.

[30] Greenshields CJ, Weller HG, Gasparini L, Reese JM. Implementation of semi-discrete, non-staggered central schemes in a colocated, polyhedral, finite volume framework, for high-speed viscous flows. Internat J Numer Methods Fluids 2010;63(1):1–21. http://dx.doi.org/10.1002/fld.2069.

[31] Warner DD. The numerical solution of the equations of chemical kinetics. J Phys Chem 1977;81(25):2329–34. http://dx.doi.org/10.1021/j100540a006.

[32] Cash J, Karp A. A variable order runge-kutta method for initial value problems with rapidly varying right-hand sides. ACM Trans Math Software 1990;16(3):pp. 201–222. http://dx.doi.org/10.1145/79505.79507.

[33] Ferziger JH, Perić M, Street RL. Computational methods for fluid dynamics. 4th ed.. Springer; 2020.

[34] The OpenFOAM Foundation, URL http://www.openfoam.org/dev.php.

[35] Guerra R, Waidmann W, Laible C. An experimental investigation of the combustion of a hydrogen jet injected parallel in a supersonic air stream. In: AIAA, editor. AIAA 3rd international aerospace conference, Washington, D. C.. 1991, URL https://elib.dlr.de/24815/.

[36] Waidmann W, Alff F, Bohm M, Claus W, Oschwald M. Experimental investigation of the combustion process in a supersonic combustion ramjet (SCRAMJET). Tech. rep., Erlangen, Germany: DGLR Jahrestagung; 1994.

[37] Waidmann W, Alff F, Böhm M, Brummund U, Clauss W, Oschwald M. Supersonic combustion of hydrogen/air in a scramjet combustion chamber. Space Technol 1994;6:421–9.

[38] Poinsot T, Veynante D. Theoretical and numerical combustion. 3rd ed.. CNRS; 2011, URL https://books.google.it/books?id=6u5rNAEACAAJ&hl.

[39] Gordon S, McBride B. Computer program for calculation of complex equilibrium compositions, rocket performance, incident and reflected shocks, and Chapman-Jouguet detonations. Tech. rep. NASA SP-273, 1971.

[40] Van Der Houwen P, B.P. S. On the internal stability of explicit, m-stage runge-kutta methods for large m-values. J Appl Math Mech 1980;60(10). http://dx.doi.org/10.1002/zamm.19800601005.

[41] Volkov V. Better performance at lower occupancy. In: Proceedings of the 2008 ACM/IEEE conference on Supercomputing. 2008, URL https://people.sc.fsu.edu/~gerlebacher/gpus/better_performance_at_lower_occupancy_gtc2010_volkov.pdf.

[42] Nickolls J, Dally WJ. The GPU computing era. IEEE Micro 2010;30(2):56–69. http://dx.doi.org/10.1109/MM.2010.41.

[43] Lindholm E, Nickolls J, Oberman S, Montrym J. NVIDIA tesla: A unified graphics and computing architecture. IEEE Micro 2008;28(2):39–55. http://dx.doi.org/10.1109/MM.2008.31.

[44] Branch statistics. 2021, Accessed: 2021/02/03, https://docs.nvidia.com/gameworks/content/developertools/desktop/analysis/report/cudaexperiments/kernellevel/branchstatistics.htm.

[45] NVIDIA, Vingelmann P, Fitzek FH. CUDA, release: 10.2.89. 2020, URL https://developer.nvidia.com/cuda-toolkit.

[46] Burke MP, Chaos M, Ju Y, Dryer FL, Klippenstein SJ. Comprehensive H2/O2 kinetic model for high-pressure combustion. Int J Chem Kinet 2012;44(7):444–74. http://dx.doi.org/10.1002/kin.20603.

[47] Goodwin DG, Speth RL, Moffat HK, Weber BW. Cantera: An object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes. 2018, http://dx.doi.org/10.5281/zenodo.1174508, Version 2.4.0, https://www.cantera.org.

[48] Zhang H, Zhao M, Huang Z. Large eddy simulation of turbulent supersonic hydrogen flames with openfoam. Fuel 2020;282:118812. http://dx.doi.org/10.1016/j.fuel.2020.118812.

[49] Oevermann M. Numerical investigation of turbulent hydrogen combustion in a scramjet using flamelet modeling. Aerosp Sci Technol 2000;4(7):463–80. http://dx.doi.org/10.1016/S1270-9638(00)01070-1.

[50] Naumov M, Arsaev M, Castonguay P, Cohen J, Demouth J, Eaton J, Layton S, Markovskiy N, Reguly I, Sakharnykh N, Sellappan V, Strzodka R. Amgx: A library for GPU accelerated algebraic multigrid and preconditioned iterative methods. SIAM J Sci Comput 2015;37(5):S602–26, URL https://doi.org/10.1137/140980260.

[51] ESI OpenCFD OpenFOAM, URL http://www.openfoam.com/.