F. Ghioldi, F. Piscaglia
A Hybrid CPU-GPU Paradigm to Accelerate Reactive CFD Simulations
International Journal for Numerical Methods in Fluids, published online 03/05/2024
doi:10.1002/fld.5297

DOI: xxx/xxxx

ARTICLE TYPE

A hybrid CPU-GPU Paradigm to Accelerate Reactive CFD Simulations

F. Ghioldi | F. Piscaglia

[1]Dept. of Aerospace Science and Technology (DAER), Politecnico di Milano, via La Masa 34, I-20156 Milan, Italy

Correspondence
*federico.ghioldi@polimi.it

Present Address
Dept. of Aerospace Science and Technology (DAER), Politecnico di Milano, via La Masa 34, I-20156 Milan, Italy

Summary

The solution of reactive CFD simulations is accelerated by the implementation of a hybrid CPU/GPU Finite Volume solver based on the operator-splitting strategy, where the chemistry integration is treated independently of the flow solution. The integration of ordinary differential equations (ODEs) describing the finite-rate chemical kinetics is solved by an adaptive multi-block explicit solver on Graphics Processing Units (GPU), while the load of the fluid solution is distributed on a multi-core CPU algorithm. The resulting speed-up for reactive CFD simulations is up to 10x; the performance gain increases with the size of the mechanism. The proposed implementation is general and can be applied to any CFD problem where the governing equations for the fluid transport are coupled with an ODE system. Code validation is performed against reference solutions on a selection of test cases involving reacting flows.

KEYWORDS:
combustion modeling, hybrid CPU/GPU solver, GPGPU chemistry, HPC, computational engineering, OpenFOAM

1 | INTRODUCTION

The solution of systems of stiff ordinary differential equations (ODEs) is of key importance in advanced multiphysics CFD simulations, for example where the fluid transport is coupled to Lagrangian Particle Tracking (LPT), to the deformation of solids in aeroelastic calculations or to the solution of finite-rate chemistry problems in reactive flow simulations. The large and ever-increasing size of detailed reaction mechanisms makes reactive flow simulations with finite-rate chemistry computationally very expensive. Reaction mechanisms for fuels relevant to aeronautical and aerospace propulsion systems, such as hydrogen, liquid methane, or ethylene, may involve a large number of species [1, 2, 3]; a surrogate mechanism for liquid fuels may include hundreds of chemical species and thousands of elementary reactions [4]. The different characteristics and properties of the sub-problems to be solved in reactive flows with finite-rate chemistry pose great challenges to the efficient treatment of the highly non-linear partial differential equations (PDEs) used to describe the evolution of reacting flows [5, 6, 7, 8, 9, 10]. In solvers using the operator-splitting technique, the solution process is divided into separate problems, namely the fluid transport and the chemical mass action. For each Control Volume (CV), the chemical kinetics is integrated over the specified time step Δt as a system of N_s+1 stiff ordinary differential equations (ODEs) to compute the reaction rates and the heat released by the reactions. The results from the fluid transport and the chemical mass action are finally combined to provide the overall solution. The technique of the operator splitting, also selected in this work, is widely applied in reactive flow simulations with finite-rate chemistry because it is very stable and second-order accurate [11]. An alternative is the Method of Lines (MOL), where the

underlying system of PDEs is discretized in space to give a system of ODEs in time [12, 13]. A discussion about the accuracy of the two methods is reported in [14].

In reactive flow simulations with finite-rate chemistry, massive parallel computing can significantly reduce the computational cost and is becoming a very attractive solution, but: a) high-fidelity simulations use very fine grids and a large number of cells, that pose more stringent constraints on the time-step advancement through the CFL criterion; b) the finite-rate chemical kinetics forces additional limitations to the time steps for stable solution: when the chemical time scales (that are proportional to the inverse of the eigenvalues of the chemical Jacobian) become too large, it is very difficult to maintain a well-conditioned matrix and the local CFL number time step of integration must be limited; c) chemistry integration causes strong load unbalancing in the regions where integration of the chemistry problem is performed. The computational time required by such kind of simulations is therefore the primary bottleneck in industrial and academic studies today and in the foreseeable future even on large computers unless CFD algorithm developments are made which allow significant time step acceleration. ODE systems connected to finite-rate chemistry can be stiff for different reasons, namely: rapid decay of some species, strong coupling among different concentration rates, or for large time steps. Small mechanisms can present large stiffness as well. Algorithms to remove stiffness via reduced chemistry have been developed in [15, 16]. Solvers that are designed for stiff ODEs, known as stiff solvers, often exploit high-order implicit functions based on backward differentiation formulas (BDFs) and they involve expensive linear algebra operations. The pay-off is that their integration time step and their numerical stability are larger compared to the nonstiff (often explicit) solvers. For a given mechanism, non-stiff solvers imply the use of extremely small time steps. New algorithmic and computer science techniques have been implemented through the years in the form of chemistry acceleration tools: In-situ Adaptive Tabulation (ISAT) [17], Chemistry Agglomeration (CA) [18], Dimension Reduction and Dynamic Cell Clustering (DCC) [19], Dynamic Adaptive Chemistry (DAC) for on-the-fly mechanism reduction during reactive flow calculations [20], chemistry tabulation [21, 22]; recently, the training and the application of Artificial Neural Networks to estimate the solution of the chemistry problem is under investigation [23, 24, 25]. An overview of the available methods for realistic chemistry in large-scale computations is provided in [15]. In modern days, the research for methodologies to reduce the cost of integration of the finite-rate chemical kinetic ODEs, as well as to speed up the overall reactive-flow computation, has increased the awareness in the combustion and software developer community to exploit the available fine-grain parallelism provided by Graphics Processing Units (GPUs). In GPUs, a large number of cores is available at low cost and with sufficiently large bandwidth; that allows large user-base access to high-power parallel applications. Calculations of the reaction rates in direct numerical simulation (DNS) of reactive flows on GPU have been presented in [26]. A speedup of 22x in single precision calculations and of 9x in double precision was obtained for a 22 species mechanism by a Runge-Kutta (RK) solver running on GPU. An explicit 4^{th} order RK solver to calculate detailed hydrogen/air mechanisms in hypersonic conditions was applied in [27, 28] with a speedup of 75x. Similarly, two methods based on explicit Runge-Kutta algorithms for the treatment of chemical kinetic ODEs to be used with moderately stiff problems have been discussed in [29]: the Runge-Kutta Cash-Karp (RKCK) [30] and the Runge-Kutta Chebyshev (RKC) [31, 32]. Both solvers showed a substantial speedup (up to 126x) for a large number of reactions against CPU calculations. For a stiff problem and with small time steps, the RKC was able to fasten the simulation up to 17x. Moreover, a 4^{th} order Runge-Kutta-Fehlberg (RKF) with adaptive time step was presented in [33], where a 25x acceleration on a 19 species/15 non-stiff reaction mechanism was obtained, under the application of a quasi-steady approximation of some radicals and reduction for fast elementary reactions in the chain. In [34], domain regions are a priori decomposed according to the chemical kinetic time scales of their reactions; then, a hybrid ODE solver simultaneously runs on the GPU (explicitly) and the CPU (implicitly). All the mentioned works make use of explicit high-order Runge-Kutta solvers [35] that allow for good accuracy even for moderately stiff initial value chemical problems.

1.1 | Motivation of this research

There is renewed interest in aerospace towards green propulsion systems, with engineers working to bring new designs to market in the upcoming years. In order to meet regulatory requirements and market expectations, aircraft engines will need to operate within noise and climate constraints and to be able to operate on synthetic fuels generated from renewable energy. 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. Heterogeneous supercomputing [29, 36] potentially represents a very effective solution to achieve this task: the chemistry problem can be computed on the GPU, while fluid transport and turbulence are computed on conventional CPU-based hardware technologies.

1.2 | Goals and highlights

We describe the development of a GPGPU solver for reactive flow simulations with finite-rate chemistry to run on heterogeneous supercomputers. Direct Integration (DI) of finite-rate chemistry problems on GPUs is linked to an implicit reactive flow solver for all flow speeds [37] based on the operator splitting technique. Main features are:

- Optimal co-design of novel algorithms for direct integration of the finite-rate chemistry problem on GPUs by algebraically stabilized explicit approach based on Runge-Kutta method with adaptive time-stepping [38], to handle sensible levels of chemical stiffness, is selected. While implicit schemes are characterized by unconditional stability, they are not the best solution for optimal GPU performance, unless large and dense matrix systems are solved [39]. This is due to the required divergence and higher memory traffic typical of implicit ODE integrators [40].
- Parallelization is utilized for both chemical kinetics and fluid mechanics. The parallelization of the chemistry problem occurs across both the mesh size and mechanism dimensions.
- The approach does not necessitate any pre-processing nor tuning, maintaining its generality across various geometries, chemical mechanisms, and thermodynamic conditions.

Algorithmic developments have been implemented in the form of a dynamic compilation unit written in the form of C++ and CUDA libraries that can be linked to any reactive flow solver available within the open-source C++ software OpenFOAM [41]. The NVIDIA Tesla V100 GPU card has been the selected hardware for the tests because of its capability to perform fast double-precision calculations. The employed CPU cores are Intel® Xeon® Gold 6242R. Each node is composed of 40 cores. Input data for chemical kinetics and species thermodynamic properties of the mechanism are converted into the OpenFOAM format from Cantera [42] by the in-house developed `canteraToFoam` converter tool. Effectiveness of the method is evaluated against implicit and explicit Direct Integration of the finite-rate chemistry problem on conventional hardware architectures. The proposed methodology can be applied to any flow solver (either implicit or explicit) based on the operator-splitting technique and can be combined to any combustion model.

1.3 | Paper structure

The paper is organized as follows: the governing equation of the reactive flow problem are described in Sec. 2; the methodology for the implicit solution of the flow transport at all speed, its implementation and the description of the discretization techniques are discussed in Sec. 3 and 4 with details in Appendix A. The solution method of the chemistry problem in a hybrid CPU-GPU system and its implementation are described in Sec. 5. The strategy followed for domain decomposition in CPU/GPU heterogeneous systems is the topic of Sec. 6. Code validation against reference solutions is discussed in Sec. 7 for auto-ignition in single-cell batch reactors and multi-cell domains. The advantages of the implemented methodology are demonstrated, in particular as the size of the chemical mechanism becomes larger. Conclusions are drawn in Sec. 8.

2 | GOVERNING EQUATIONS FOR REACTIVE FLOW PROBLEMS

The sequential solution of the governing equations for the flow transport is carried out by the Finite Volume (FV) discretization using semi-discrete, non-staggered central schemes for colocated variables prescribed on a mesh of polyhedral cells. The standard governing fluid equations in an Eulerian frame of reference read:

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

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

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

being ρ, $\boldsymbol{U}$, p and T the fluid density, velocity, pressure and temperature respectively. $\boldsymbol{R}$ is the viscous part of the stress tensor. The total energy density is $E = e + |\boldsymbol{U}|^2/2$, being e the specific internal energy. The diffusive heat flux $\boldsymbol{q} := \lambda \nabla T$ (where λ is the thermal conductivity of the flow) is defined as positive for cooling. The terms $\boldsymbol{S}_U$ and S_e are sources and sinks for momentum and energy, respectively; $\dot{Q}$ is the heat released by the combustion. In addition to the governing equations for the fluid flow, a

set of $N_s - 1$ convection-diffusion partial differential equations (PDEs) is solved to determine the local mass fraction Y_i of each of the N_s chemical species transported by the reactive fluid mixture in each Control Volume (CV):

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

The mass fraction of the inert species N_s is determined as:

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

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 i-th 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 i-th species; $\nu_{i,j}$ is the i-th species stoichiometric coefficient, Q_j is the non-equilibrium reaction rate of the j-th reaction:

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

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 [43], while the ratio $\dfrac{\rho Y_i}{W_i}$ is the molar concentration of the i-th 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 finally obtained as:

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

where 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 i-th gaseous species, as done in the NASA chemical equilibrium code [44], 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 i-th 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 in-house developed `canteraToFoam` utility.

3 | IMPLICIT SOLUTION METHOD OF THE FLOW TRANSPORT FOR ALL FLOW SPEEDS

The flow transport problem (Eqs. 1 to 4) is solved by a SIMPLE-type pressure-based compressible unsteady segregated flow solver [45] for unstructured polyhedral grids for all flow speeds [37]. The solution method is shown in Fig. 1: the linearized equations for velocity components, specie transport, energy and pressure correction are solved in turn, until the coupling of the primary variables is reached. Being all terms discretized by a fully-implicit scheme, non-linear terms (fluxes and source terms) are linearized and computed at the new time level, while equations are solved iteratively. The governing equations implemented in the solution method are reported in detail in Appendix A.

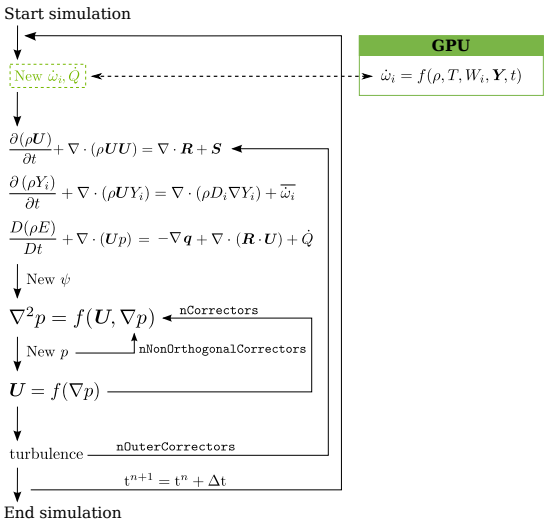


FIGURE 1 solution method employed in the SIMPLE-type GPGPU pressure-based compressible unsteady segregated flow solver. In this work, the calculation of the chemical mass action $\dot{\omega}_i$ at each time step is performed on the GPUs.

The solution of the chemical mass action (finite-rate chemistry problem) is solved separately for each CV before the flow transport. Reaction rates from calculations of the finite-rate chemistry are passed to the fluid transport problem (Fig. 1) as a source term in the species transport equations (Eq. 4). The reaction rate is also used by the combustion model to compute the source term $\dot{Q}$ (Eq. 10), which in turn is added to the energy equation (Eq. 3). In this work, finite-rate chemistry computations are performed on Graphical Processing Units (GPUs), as it will be discussed in Sec 5.

4 | VARIABLES POSITIONING AND DISCRETIZATION OF THE OPERATORS

The Finite Volume Method on a collocated grid arrangement is used for spatial discretization (Fig. 2a). Primary flow variables are defined at the cell centers and their derivatives are computed as follows:

- **Temporal derivatives** of a quantity Ψ, e.g. $\frac{\partial \Psi}{\partial t}$,

$$\int_V \left(\frac{\partial \Psi}{\partial t}\right) dV = \left(\frac{3}{2}\Psi^{n+1} - 2\Psi^n + \frac{1}{2}\Psi^{n-1}\right)\frac{V}{\Delta t} \tag{13}$$

are discretized by the Second-Order Backward Euler, a two-step Adams-Moulton method [46] where a linear combination of the current-time and the old-time derivatives is used. In Eq. 13, $n+1$, n and $n-1$ are the times of the updated, of the current and of the old solution respectively. First order implicit schemes are also eventually supported.

- **Diffusive term (Laplacian)** of a quantity Ψ, e.g. $\nabla \cdot (\Gamma \nabla \Psi)$:

$$\int_V \nabla \cdot (\Gamma \nabla \Psi)\, dV = \int_S (\Gamma \nabla \Psi)_f \cdot \boldsymbol{n} dS \simeq \sum_f \Gamma_f \boldsymbol{S}_f \cdot (\nabla \Psi)_f = \sum_f \Gamma_f |\boldsymbol{S}_f| \nabla_n \Psi_f \tag{14}$$

where $\nabla_n \Psi_f$ is the surface normal gradient of Ψ. The subscript f in Eq. 14 indicates the cell-to-face interpolated quantities. In the present work, linear cell-to-face interpolation has been applied; for irregular polyhedral meshes, interpolation is generalized by defining a weight w for each face:

$$\boldsymbol{\Psi}_f = w \boldsymbol{\Psi}_P + (1-w) \boldsymbol{\Psi}_N \tag{15}$$

where Ψ_f is the face-interpolated quantity. Subscripts P and N indicate values at the centers of two neighboring cells.

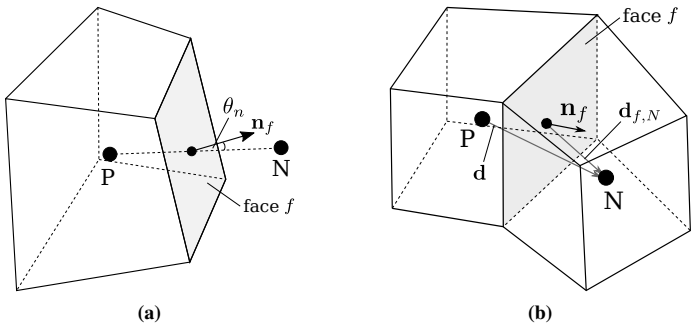


FIGURE 2 a) collocated grid arrangement of primary variables: velocity fluxes are interpolated at the face center; b) computation of the surface gradient: non-orthogonal correction [47].

For the non-orthogonal grid of Fig. 2a in a collocated variable arrangement, the surface gradient of a quantity Ψ is decomposed onto an orthogonal part and a non-orthogonal correction (see Fig. 2b):

$$\nabla_n \Psi_f^{n+1} = \underbrace{\alpha(\Psi_P^{n+1} - \Psi_N^{n+1})}_{\text{implicit}} + \underbrace{(\mathbf{n}_f - \alpha \mathbf{d}) \cdot (\overline{\nabla_n \Psi})_f^n}_{\text{explicit}} \tag{16}$$

where $\alpha = \frac{1}{\boldsymbol{n}_f \cdot \boldsymbol{d}}$ and $\overline{\nabla_n \Psi_f}$ is the *uncorrected* normal gradient from the two values of the two cells sharing the face. The explicit part is computed from Eq. 15 as:

$$\nabla \Psi_f^n = w \nabla \Psi_P^n + (1-w) \Psi_N^n \tag{17}$$

being the interpolation weight $w = 0.5$ in this work; the normal gradient is computed as:

$$(\overline{\nabla_n \Psi})_f^n = \nabla \Psi_f^n \cdot \boldsymbol{n}_f^n \tag{18}$$

- **Gradient terms**: the Green-Gauss theorem yields:

$$\nabla \Psi_P = \frac{1}{V_P} \sum_f \Psi_f \boldsymbol{S}_f \tag{19}$$

being V_P the volume of the polyhedral cell P, and $\boldsymbol{S}_f$ the surface vector of the f-th face of the cell.

- **Non-linear terms (convective terms)**: the convective term in the momentum balance equation is linearized using the Picard approach: the mass flux is treated explicitly and the non-linear term is approximated by:

$$(\rho \boldsymbol{U})\, \Psi \simeq (\rho \boldsymbol{U})^n\, \Psi^{n+1} \tag{20}$$

The index n in Eq. 20 denotes that the values are taken from the result of the old time-step. A technique for momentum-based interpolation of mass fluxes on cell faces [48, 49] is used to mimic the staggered-grid discretization to prevent checkerboard effects. By using the divergence theorem, the convective terms are rewritten as:

$$\int_V \nabla \cdot (\rho \boldsymbol{U} \Psi) \simeq \sum_f (\rho_f^{m-1} \boldsymbol{U}_f^n)\, \Psi_f^{n+1} \tag{21}$$

In Eq. 21, the density ρ^{m-1} comes from the previous outer iteration $m-1$ (Fig. 1); the velocity $\boldsymbol{U}_f$ is interpolated via the same approach of Eq. 15; a second-order central differencing scheme is used for the fluxes.

5 | MULTI-CELL APPROACH TO ACCELERATE THE CHEMICAL SOLUTION ON HYBRID CPU-GPU SYSTEMS

Combustion simulations with finite-rate chemistry involve the solution of a chemical kinetics system of ordinary differential equations (ODEs). The cost of the computation is proportional to the number of species, the number of reactions, and that of grid points in the domain. Non-linearity in the formulation of reaction rate variables and the contemporary presence of species with very different characteristic time scales lead to very stiff ODE systems: the smallest scales control the size of the integration time step to ensure convergence of the solution, which can impact the computational cost significantly. The time step of integration Δt_{fluid} of the Partial Differential Equations (PDEs) describing the fluid transport problem must comply with the CFL condition, while the integration of the ODE system of the kinetic problem is performed over a time interval Δt_{chem} that must ensure computational stability [31]. 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:

$$\dot{\omega}_i = \left(c_i^{n+1} - c_i^n\right) \frac{W_i}{\Delta t_{\text{fluid}}} = \rho^n \frac{(Y_i^{n+1} - Y_i^n)}{\Delta t_{\text{fluid}}} \tag{22}$$

being c_i^n the molar concentration of the i-th 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. It is important to note that chemistry integration for each CV is assumed to occur for a fixed mass of fluid at a constant pressure; therefore, both the density and volume change during integration. When species source terms are computed, they must relate exactly to the mass of fluid considered during integration in order for mass to be conserved; this is achieved by multiplying the mass fractions that results from integration by the old-time density, as that relates to the mass in the cell volume. As a consequence, chemistry integration over a time-step has no dependence on new-time properties of the fluid.

Implicit ODE solvers relying on variable-coefficient methods usually have good performance for the integration of the finite-rate chemistry problem, as they are capable of extending the time-step if the fast modes of the ODE system have already reached their asymptotic values. Conversely, explicit solvers are characterized by smaller integration time-steps but avoid the iterative solution and associated matrix inversions required for implicit integration, so they have an intrinsically parallel nature. They are therefore well-suited to the GPU massive parallel architecture that is optimized to perform a large number of independent operations repeated multiple times [26, 36, 40, 50]. Additionally, their locality reflects on memory usage [51], which is very low if compared to implicit solvers. In this work, explicit RK methods with adaptive time-stepping [38] have been selected to solve the ODE system. Embedded Runge-Kutta formulas [52, 53] have several advantages: a) they can be used to construct high order accurate numerical methods by few evaluation functions; b) their truncation error can be easily estimated and used to compute the next step size. The explicit Runge-Kutta Cash-Karp method (Eq. 24) linked to the butcher tableau [30] of Tab. 1 has been implemented in CUDA. Inputs of the integration algorithm are flow conditions and the global time step of integration. By considering

$$\boldsymbol{y} = [\boldsymbol{Y}, T] \tag{23}$$

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:

$$
\begin{aligned}
\boldsymbol{y}_1 &= \boldsymbol{y}^n + \Delta t_{\text{chem}} \left(a_{2,1} \frac{d\boldsymbol{y}^n}{dt} \right) \\
\boldsymbol{y}_j &= \boldsymbol{y}^n + \Delta t_{\text{chem}} \left(a_{j+1,1} \frac{d\boldsymbol{y}^n}{dt} + \sum_{i=2}^{j} \left(a_{j+1,i} \frac{d\boldsymbol{y}_{i-1}}{dt} \right) \right) \qquad j \in [2,5] \\
\boldsymbol{y}^{n+1} &= \boldsymbol{y}^n + \Delta t_{\text{chem}} \left(b_1 \frac{d\boldsymbol{y}^n}{dt} + \sum_{i=2}^{6} \left(b_i \frac{d\boldsymbol{y}_{i-1}}{dt} \right) \right)
\end{aligned}
\tag{24}
$$

0	0					
c_2	a_{21}	0				
c_3	a_{31}	a_{32}	0			
c_4	a_{41}	a_{42}	a_{43}	0		
c_5	a_{51}	a_{52}	a_{53}	a_{54}	0	
c_6	a_{61}	a_{62}	a_{63}	a_{64}	a_{65}	0
	b_1	b_2	b_3	b_4	b_5	b_6
	$\tilde{b}_1$	$\tilde{b}_2$	$\tilde{b}_3$	$\tilde{b}_4$	$\tilde{b}_5$	$\tilde{b}_6$

0	0						
$\frac{1}{5}$	$\frac{1}{5}$	0					
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$	0				
$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{40}$	$\frac{6}{5}$	0			
1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$	0		
$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	0	
	$\frac{37}{378}$	0	$\frac{250}{621}$	$\frac{125}{594}$	0	$\frac{512}{1771}$	(5^{th} order)
	$\frac{2825}{27648}$	0	$\frac{18575}{48384}$	$\frac{13525}{55296}$	$\frac{277}{14336}$	$\frac{1}{4}$	(4^{th} order)

TABLE 1 Runge-Kutta Cash-Karp extended Butcher tableau (right) with its reference notation (left).

For each step in Eq. 24, the a_i and b_i constants are specific for the chosen Runge-Kutta method; in this context, the values come from Tab. 1 (left-to-right comparison). Also, for each step, an update

$$
\frac{d\mathbf{y}_i}{dt} = f\left(t^n + c_i \Delta t_{\text{chem}} \, , \, \mathbf{y}^n + \Delta t_{\text{chem}} \sum_{j}^{i-1} a_{i,j} \frac{d\mathbf{y}_j}{dt} \right)
$$

is needed; for the Runge-Kutta Cash-Karp, the c_i constants are reported in Tab. 1. An error $\text{err}_{\boldsymbol{y}}$ for each variable in $\boldsymbol{y}$ (chemical mass fractions and temperature) is computed at the end of the procedure shown in Eq. 24:

$$
\mathbf{err}_{\boldsymbol{y}} = \Delta t_{\text{chem}} \left(e_1 \frac{d\boldsymbol{y}^n}{dt} + \sum_{i=2}^{6} \left(e_i \frac{d\boldsymbol{y}_{i-1}}{dt} \right) \right) \tag{25}
$$

With reference to Tab. 1, $e_i = b_i - \bar{b}_i$; in particular, b_i and $\bar{b}_i$ are the weights associated with the 5^{th} and 4^{th} order respectively. Finally, the maximum error

$$
\text{err}_{\text{max}} = \max(\mathbf{err}_{\boldsymbol{y}}) \tag{26}
$$

is determined. Eq. 25 is calculated locally in the computational cell, thus convergence is local to each GPU block and the GPU calculation is asynchronous. This limits the synchronization overhead and promotes the maximization of hardware performance.

The procedure develops as follows. Data required by the GPU to perform GPU-ODE integration are copied from host to device. The device performs ODE integration and computes: a) the reaction rates; b) a guess of the next time step integration for the chemical ODEs. As soon as the kernel calculation is over, the reaction rates and the time step advancement for each cell are first stored into the global GPU memory and then copied back to the CPU host (see Fig. 1). In reactive flow simulations, the GPU solutions are eventually updated (on CPU) by the flow solver. Since the solution of chemistry is decoupled from the fluid transport, the novel GPU-ODE chemistry integrator can be combined any flow solver (compressible, incompressible, multiphase) where the operator splitting technique is applied.

As shown in Fig. 3, there are three levels of tasks in the CUDA framework, namely the grid, the block and the thread. The grid is made of several blocks that are launched by a GPU kernel. 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. All threads in a block run in parallel, in the Single Instruction Multiple Threads (SIMT) mode [54]. 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 accelerate data access, the use of shared memory is quite critical [55]: 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 [56]. 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 [57] has been applied with some effects on data structure and organization: data access time is minimized by 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, the ODE solver settings, and the parameters of the Butcher tableau 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. 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. block), each species in the reaction mechanism is handled in parallel on multiple active threads (Fig. 3). If the amount of data to be transferred overcomes the maximum memory availability of the GPU(s), the chemical problem gets automatically split into mesh chunks, each containing a cluster of cells. If the mesh dimension overcomes the maximum number of cells that can be concurrently treated by the GPU streaming multiprocessors, a queue is automatically 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.

6 | FURTHER NOTES ABOUT DOMAIN DECOMPOSITION IN CPU/GPU HETEROGENEOUS SYSTEMS

The GPU ODE solver is dynamically linked to the reactive flow solvers available in the open-source framework (Fig. 3). Two independent partitionings have been applied to the chemistry problem and the flow domain respectively. The computational mesh is decomposed into a series of ordered subgroups by a parallel domain decomposition algorithm over the available CPU processor cores. This procedure involves the use of MPI libraries and it is independent of the subsequent treatment of the chemistry ODE system. Memory allocation for the transfer of data to the device is produced automatically: this ensures correct memory allocation for each processor, based on the available dynamic memory at the beginning of the simulation. Each CPU processor solves fluid transport, energy, and species transport equations over the assigned subdomain mesh. In the hybrid GPGPU solver, computation is distributed across many MPI ranks on CPUs. In each computing node, each MPI rank nominates a single master thread which will handle all communication with the GPU. In order to specify how the global system is distributed across the MPI ranks, communication maps are set. This procedure to combine the data into single large datasets in an efficient mode and to transfer data onto GPUs is handled automatically in the presented algorithmic developments. On GPUs, the decomposition of the chemical problem is done independently and relies on a double subdivision over blocks and threads. In particular, each cluster of cells is represented by a series of blocks where the chemistry problem is solved in parallel; for each block, the solution of the reaction mechanism is parallelized over multiple threads. On GPU, synchronization is only required within the block. In this approach, each CUDA block is handled asynchronously, while operations are fully synchronous within the CUDA block. As a result, a double parallelization of the calculation of the chemistry problem on cells/blocks and on species/threads is

achieved. A three-level parallelization is therefore globally employed, as shown in Fig. 3: a) parallelization of the fluid dynamic problem over CPU processor cores/mesh subdomains; b) simultaneous solution of the chemistry problem over clusters of cells; c) simultaneous/parallel solution of the reaction mechanism in the block threads. The limit on the maximum number of threads per block can be a potential bottleneck for parallelism. This is usually not the case in reactive unsteady CFD simulations, where a 1024-species mechanism is considered a very large mechanism to be solved at run-time. For problems involving larger numbers of species (over a thousand), the thin thread approach [58] is preferred.

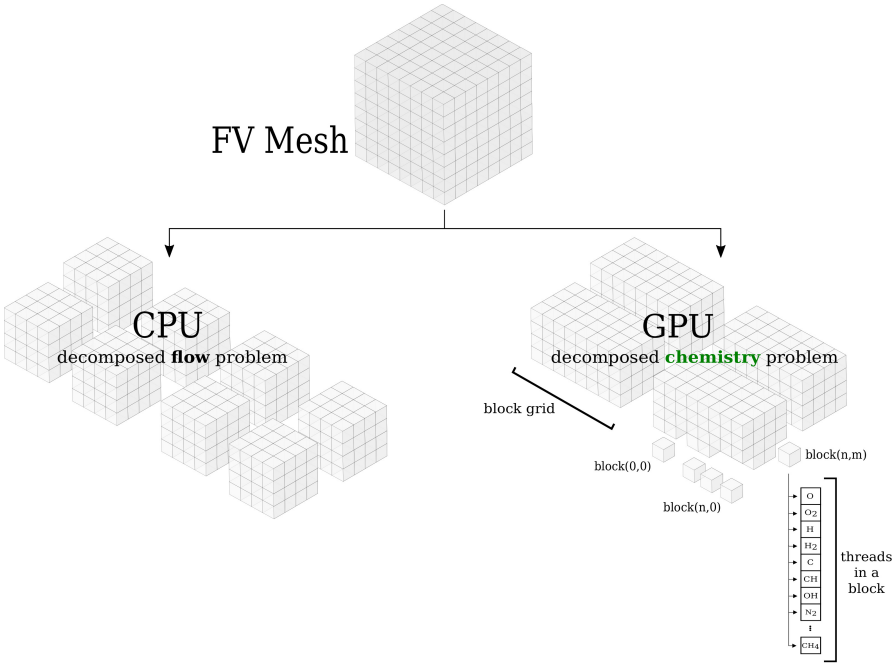


FIGURE 3 decomposition method for hybrid CPU/GPU computations. The heterogeneous solver employs a three-level parallelization on the fluid dynamic problem, the chemistry problem and the reaction mechanism.

7 | VALIDATION

Three mechanisms differing in the number of species, reactions, and stiffness level have been used for validation, namely:

- Hydrogen combustion (H_2/air): the use of Hydrogen fuel is widespread in space industry for rocket propulsion applications because of its high specific impulse. Its oxidation kinetics involves very fast mass diffusivity and low molecular weight, and it represents a valid option to decarbonize high-emission industries, aviation, and other hard-to-abate sectors [59]. Hydrogen combustion is an example of a non-stiff mechanism [1, 60, 61];

- Methane combustion (CH_4/air): the GRI-Mech 3.0 by the Gas Research Institute (GRI) [62] is a very-stiff mechanism, with its 53 species and 325 reactions. It is used to predict NOx emissions in aeronautical combustors operating in diffusion mode;

- Ethylene combustion (C_2H_4/air): a series of mildly-stiff mechanisms [63, 64, 65] used for scramjet engines; ethylene glycol is also used as antifreeze in de-icing processes in aeronautics; different reaction mechanisms of different sizes are available in the literature for Ethylene combustion.

Reaction mechanisms were tested on multiple test geometries that were selected for the computations, namely:

- auto-ignition in a single-cell batch reactor, to check the solution of Direct integration (DI) of kinetic mechanisms without the computation of the fluid transport. The results and performance of the GPGPU algorithm are compared against: a) an equivalent CPU version of the ODE solver [41]; b) an implicit ODEs integrator contained in OpenFOAM; c) the solution

from the Cantera software [42]. This set of simulations is used to evaluate the impact of latency from data transfer on the overall time of the computation and to also show that all the used ODE solvers provide similar results, for a fair comparison.

- Reactive flow simulations on multi-cell domains, to test the overall performance of the GPGPU solver with fluid transport in presence of multi-domain parallelization. A one-dimensional laminar flame speed propagation and a two-dimensional low-Mach-Number unsteady laminar counterflow diffusion flame are selected as validation test cases. Calculations from the GPGPU solver are compared against results from the CPU-ODE solver.

All calculations on the GPUs were performed in double precision. The following quantities have been monitored through the simulations:

- the spatial distribution of temperature and species mass fraction;
- the computational time required by the calculation;
- the cumulative mass fraction of the chemical species:

$$\bar{Y}_i = \frac{1}{\Delta t} \int_0^t Y_i \, dt \tag{27}$$

Time integral values of $\bar{Y}_i$ are used to calculate the error of the solution against Cantera [42], that is assumed as reference:

$$\text{err}_{i,CPU} = \frac{\left|\bar{Y}_{i,CAN} - \bar{Y}_{i,CPU}\right|}{\bar{Y}_{i,CAN}} \cdot 100 \tag{28}$$

$$\text{err}_{i,GPU} = \frac{\left|\bar{Y}_{i,CAN} - \bar{Y}_{i,GPU}\right|}{\bar{Y}_{i,CAN}} \cdot 100 \tag{29}$$

Finally, the difference of the results between GPU and CPU is calculated as:

$$\text{diff}_i = \frac{\left|\bar{Y}_{i,CPU} - \bar{Y}_{i,GPU}\right|}{\bar{Y}_{i,CPU}} \cdot 100 \tag{30}$$

7.1 | Auto-Ignition in Single-Cell Batch Reactors

Direct integration of kinetic mechanisms in single-cell batch reactors is used to verify the accuracy of the CPU/GPU method in solving the chemical ODE systems. For each studied mechanism, the solutions produced by the hybrid CPU/GPU code are compared against Cantera [42]. Starting from a set of initial conditions, data are synchronized between the CPU and the GPU at each time step of integration (see Fig. 4). Synchronization between CPU and GPU is the main performance bottleneck and it is in principle unnecessary for single-cell batch reactor cases, where fluid transport equations are absent. This set of simulations is meant to be used to estimate the time required by data transfer and to compare it to the overall time required for the integration.

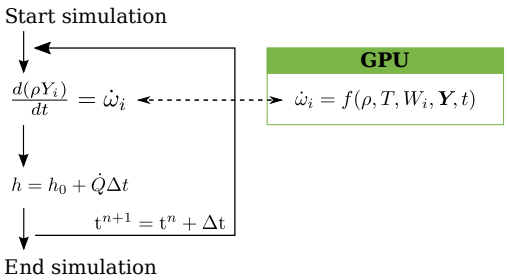
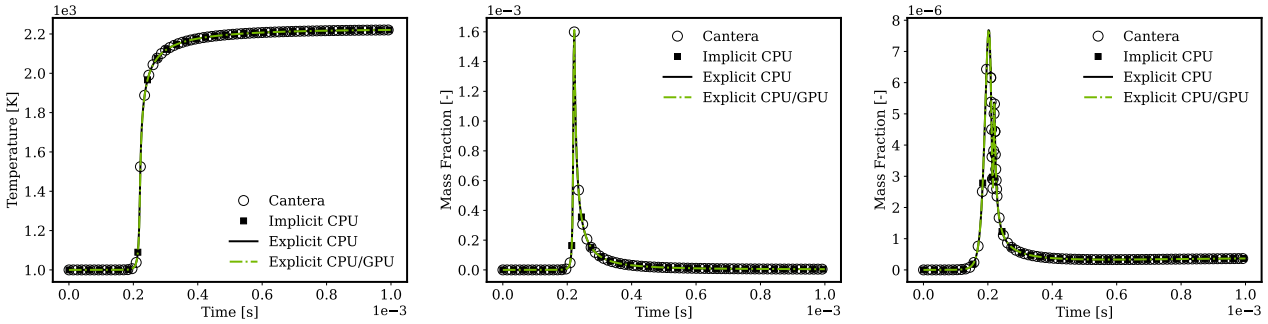


FIGURE 4 solution of the autoignition in single cell batch reactors: structure of the hybrid solver.

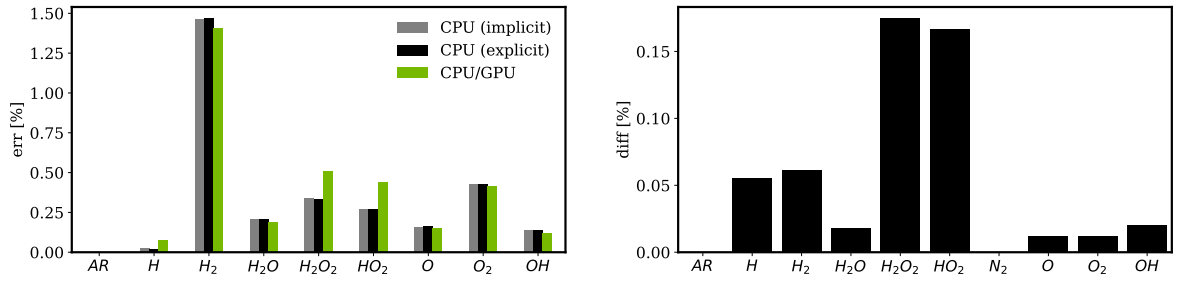
Hydrogen combustion. The first set of calculations involves a stoichiometric Hydrogen-air mixture reacting at a constant pressure of 2.0 bar and an initial temperature of 1000 K. The mechanism is made of 10 species and 27 reactions [60]. The lack

of stiffness is due to both the particular chemistry considered and the small global time step used. Fig. 5 shows the evolution over time of the temperature and the mass fraction of selected chemical species. A good agreement is noted between the results produced by CPU solvers and the hybrid CPU/GPU integrator. Limited differences with respect to the reference solution from Cantera (Eq. 28 and 29) are reported for each of the tracked species in Fig. 6a.



(a) temperature **(b)** H **(c)** H_2O_2

FIGURE 5 Hydrogen/air auto-ignition predicted over time by the direct integration of the chemistry problem via the hybrid CPU/GPU code, explicit and implicit CPU integrators and by Cantera [42].



(a) **(b)**

FIGURE 6 Hydrogen/air batch reactor: (a) cumulative mass fraction discrepancy based on Eq. 28 and 29; (b) difference between CPU and CPU/GPU based on Eq. 30.

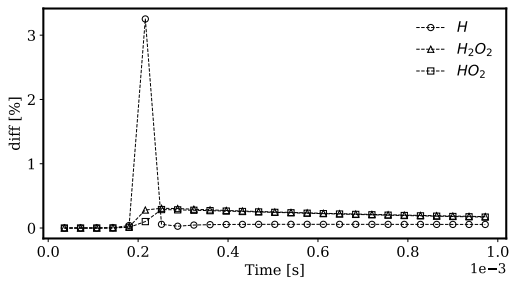
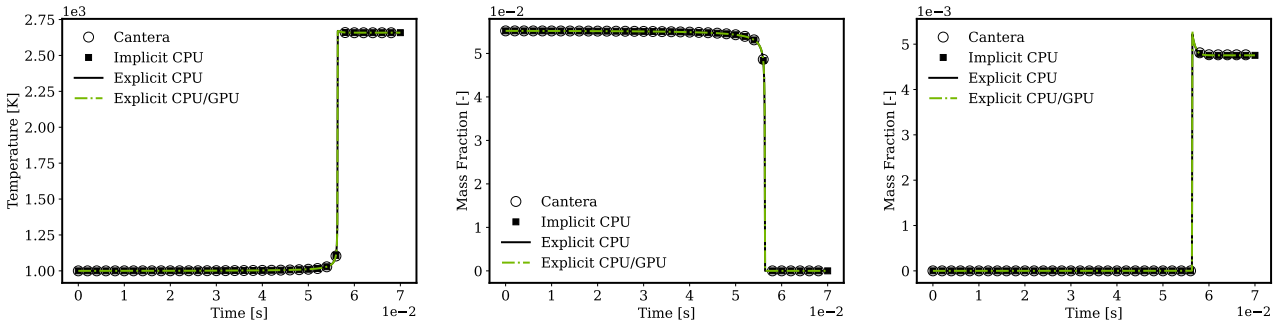


FIGURE 7 Hydrogen/air batch reactor: evolution over time of the three largest discrepancies reported in Fig. 6b.

Fig. 6a shows that the maximum discrepancy for this test is contained below 1.5%. The result includes the new hybrid solver. Also, the largest reported discrepancies between explicit CPU and GPU computing are linked to intermediate species whose mass fraction is very small throughout the development of the reaction. In this regard, Fig. 6b shows discrepancies always below 0.2%. They are present because: a) the developed GPGPU approach does not consist in a one-to-one porting of the CPU algorithm onto GPU; b) the order of the operations is rearranged to better suit GPGPU computing; c) GPU computing implies a different parallelization strategy compared to the serial CPU treatment; d) algebraic libraries on the two architectures (CPU and GPU) are not the same [27]; e) rounding and truncation errors may produce small yet non negligible effects on the prediction of the subsequent time advancement returned back from the GPU to the CPU.

For completeness, Fig. 7 illustrates the evolution of the three largest differences between explicit CPU and CPU/GPU integrations over time. The peak around t $\approx 2\times10^{-4}$ s stems from a marginal, albeit non-zero, discrepancy in time marching between CPU and GPGPU ODE integration. This difference diminishes rapidly due to the integral nature of $\bar{Y}$, swift consumption of intermediate species, and the overall similarity in areas under the explicit CPU and CPU/GPU curve (Fig. 5b).

Methane combustion. The 53-species, 325-reaction mechanism GRI-Mech 3.0 which models natural gas combustion is selected as stiff mechanism. In the proposed test, a stoichiometric Methane-air mixture reacts in a batch reactor at a constant pressure of 13.5 bar and with an initial temperature of 1000 K. To guarantee good levels of accuracy, the tolerance criteria for explicit integration are now stricter compared to those used for the Hydrogen test case. Explicit CPU and CPU/GPU integrations are performed using the same settings.



(a) temperature **(b)** CH_4 **(c)** OH

FIGURE 8 Methane/air (GRI-Mech 3.0) auto-ignition predicted over time by the direct integration of the chemistry problem via the hybrid CPU/GPU code, explicit and implicit CPU integrators and by Cantera [42].

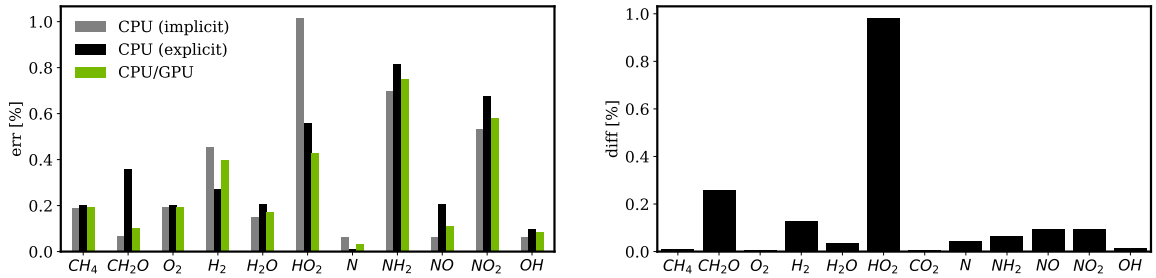
From Fig. 8, the evolution of temperature in time predicted by the different methods is similar. Cumulative relative discrepancies against the computation from Cantera for GRI-Mech are reported in Fig. 9a and 9b for main species.

Cumulative relative discrepancies on mass by the CPU code (Eq. 28) and the hybrid CPU/GPU code (Eq. 29) are limited below 1%. The explicit CPU/GPU solutions (green bars in Fig. 9a) exhibit strong agreement with explicit CPU results for most species. Notably, even in this context, differences between hybrid CPU/GPU integration and explicit CPU integration remain modest. Fig. 9b reports a peak of approximately 1% for an intermediate species characterized by rapid creation and depletion, with limited mass fraction throughout the simulation. For all other species, the cumulative difference remains below 0.3%.

Fig. 10 shows the evolution of the three main differences over time for the GRI-Mech 3.0 mechanism. Peaks of differences are located in correspondence of the sudden rise of the mass fractions of the intermediate species; the differences quickly reduce even for the GRI-Mech 3.0 test case.

Ethylene combustion. The third auto-ignition test is performed on an Ethylene chemical chain composed by 57 species and 268 reactions [63]. This mechanism is categorized as mildly-to-very stiff because of its large number of species and reactions. In this specific test, a mixture of Ethylene and air reacts in a batch reactor at a constant atmospheric pressure and an initial temperature of 1000 K.

In Fig. 11, the temperature profile from direct integration by the CPU/GPU code closely follows the calculation performed on the CPU. Mass fractions confirm that the hybrid CPU/GPU code performs similarly to CPU explicit and implicit direct



(a) **(b)**

FIGURE 9 Methane/air (GRI-Mech 3.0) batch reactor: (a) cumulative mass fraction discrepancy based on Eq. 28 and 29; (b) difference between CPU and CPU/GPU based on Eq. 30.

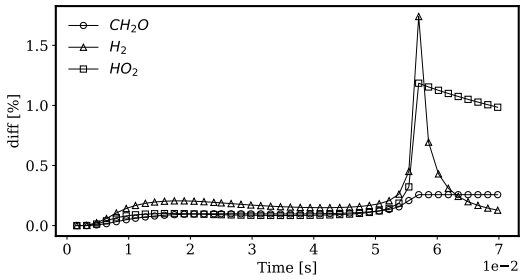
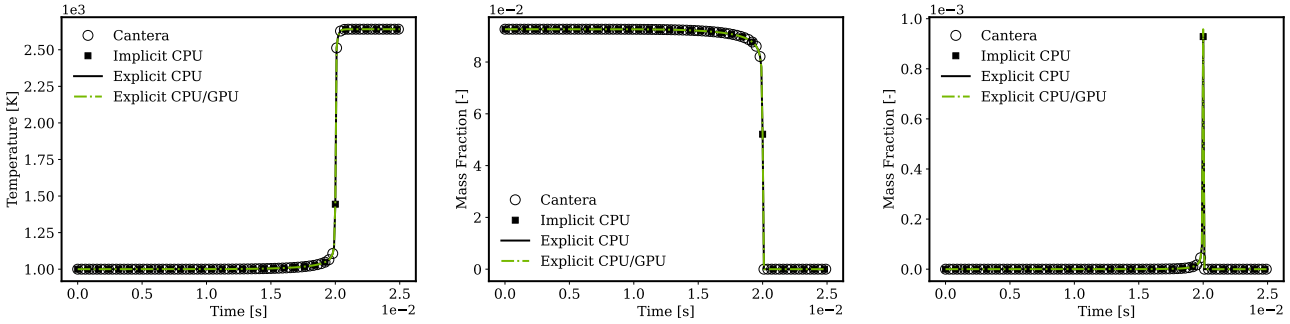


FIGURE 10 Methane/air (GRI-Mech 3.0) batch reactor: evolution over time of the three largest discrepancies reported in Fig. 9b.



(a) temperature **(b)** C_2H_4 **(c)** CH_4

FIGURE 11 Ethylene/air auto-ignition predicted over time by the direct integration of the chemistry problem via the hybrid CPU/GPU code, explicit and implicit CPU integrators and by Cantera [42].

integration. As for the other mechanisms tested, cumulative and comparative discrepancies (see Fig. 12a and 12b) are reported. Cumulative discrepancies of the solution performed by the different methods are limited and comparable.

Once again, the largest reported errors remain below 1%. They are primarily associated with intermediate species (CH_3, CH_4, C_3H_8, HCO) exhibiting small mass fractions, active only briefly around t ≈0.02 s, whose generation and depletion is rapid. The reported errors in Fig. 12a correspond to differences smaller than 0.5% in Fig. 12b. For other reported species, the discrepancies between CPU and hybrid CPU/GPU computing are marginal (Fig. 13). Despite employing identical tolerances and integration settings for both explicit CPU and CPU/GPU ODE integrators, discrepancies persist due to previously mentioned factors.

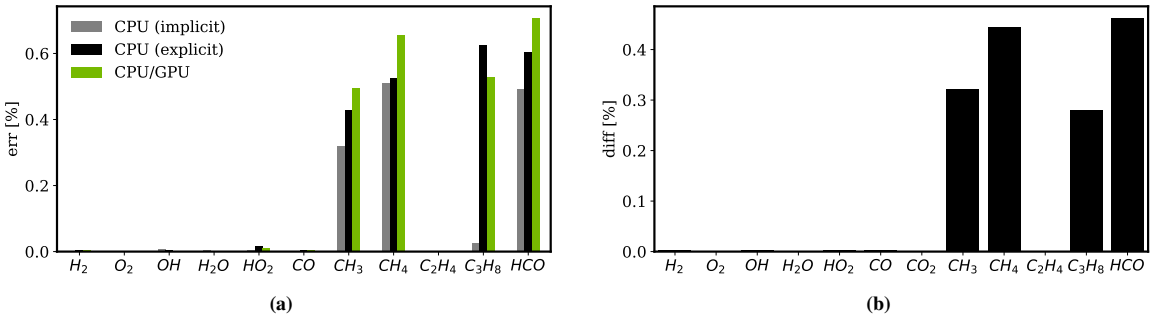


FIGURE 12 Ethylene/air batch reactor: (a) cumulative mass fraction discrepancy based on Eq. 28 and 29; (b) difference between CPU and CPU/GPU based on Eq. 30.

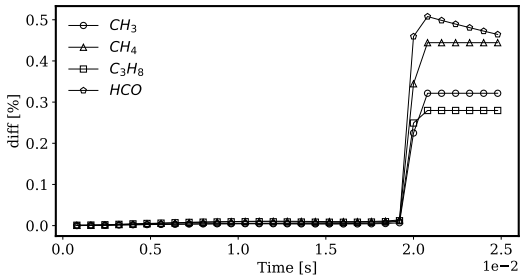
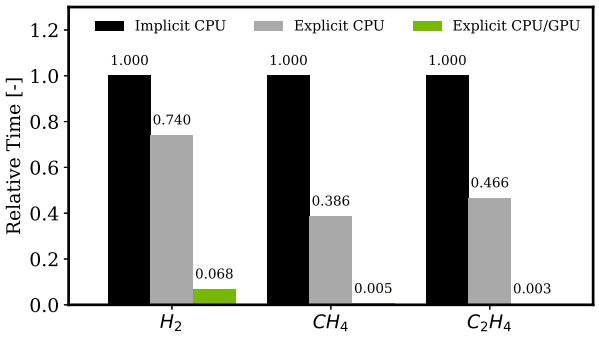


FIGURE 13 Ethylene/air batch reactor: evolution over time of the four largest discrepancies reported in Fig. 12b.

Performance metrics. Tests on single-cell batch reactors have proved that Direct Integration on the GPU provides a reliable and accurate solution for both non-stiff and stiff reaction mechanisms. Code profiling shows a very low device occupancy for auto-ignition calculations. With mechanisms of less than 32 species, only one warp and a single GPU block are used, because of the fat-thread approach. Therefore, the potential of the GPU card is not fully exploited. Nevertheless, a significant speed-up is achieved (Fig. 14a): explicit direct integration on GPU is 10x faster for the Hydrogen mechanism, 83x faster for the Methane (GRI-Mech 3.0) chemical chain, and 148x faster for Ethylene, compared to the same process done via CPU. The speed-up increases with the computational load on the GPU, and latency in data access on GPU becomes proportionally less important. Code profiling shows that a very small percentage of the available bandwidth is used for these three simulations: the small size of the vectors transferred and the use of the fat thread approach reduces the bandwidth usage. As expected, repeated data transfer between CPU and GPU represents a serious bottleneck (see Fig. 15). For this reason, the optimal configuration for the hybrid CPU/GPU code for single-cell batch reactor simulation would employ the following procedure: a) load the GPU kernel APIs as the simulation starts; b) transfer data between CPU and GPU for direct integration at the beginning and at the end of the computation; c) unused GPU memory must store intermediate solution for subsequent post-processing. Clearly, this is out of scope for the present investigation.

7.2 | Multi-cell reactive flow simulations

Reactive flow computations on one- and two-dimensional domains are presented. A one-dimensional laminar flame propagation and a two-dimensional low-Mach-Number unsteady laminar counterflow diffusion flame are simulated. The laminar finite-rate model is applied in the simulations to compute the chemical source terms. The presented algorithmic developments are natively compatible with the turbulence-chemistry interaction models already available in OpenFOAM for the simulation of turbulent combustion [41]. The choice of laminar combustion test-cases does not have any impact on the generality of the presented results, being the aim of this section the testing of the performance of the GPU solver in reactive flow computations and the comparison of the results against a reference solution from CPU calculations.



(a)

Fuel	Mechanism	Species	Reactions	Speed-up factor
H_2	H_2-O_2	10	27	10
CH_4	GRI-Mech 3.0	53	325	83
C_2H_4	UCSD [63]	57	268	148

(b)

FIGURE 14 average time to perform ODE integration compared to implicit (a); speed-up factor related to sole ODE integration (b).

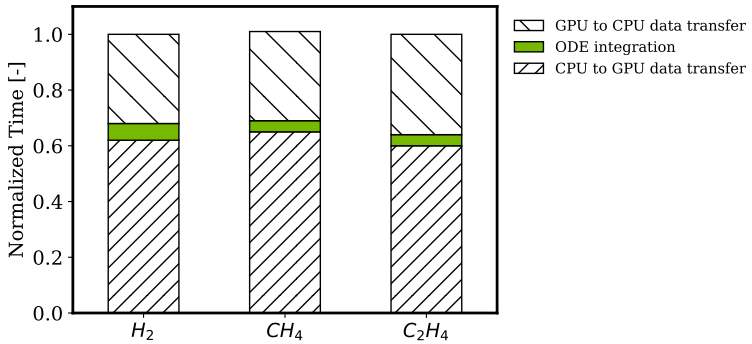


FIGURE 15 normalized clock time of the tasks of the hybrid ODE integration, single cell batch reactor cases. Time for data transfer from CPU to GPU is larger as the number of inputs passed from the CPU is larger than the number of the outputs from the GPU.

1D laminar flame speed propagation. One-dimensional laminar flame propagation is selected as a validation test case to account for the fluid dynamic and the chemical kinetic problem simultaneously. The scheme of the one-dimensional shock tube used for this investigation is proposed in Fig. 16. The computational domain is uniformly discretized along the elongation axis, and specific cell refinements are chosen to emulate problems of increasing computational demand. Simulations are performed at different grid sizes, ranging from 800 to 20000 cells. Multiple chemical mechanisms for Hydrogen and Ethylene are used. In the simulations, the fuel inlet is placed at one end of the domain; the fuel has a prescribed velocity and an inlet temperature $T_{in} = 1350$ K (Tab. 2 and 3).

Results from the hybrid CPU/GPU code are validated against the direct integration achieved on the CPU by an explicit and an implicit method, the latter of which is used as reference. The analysis proposed in Fig. 17 highlights the evolution of relevant quantities in space and time for the Hydrogen mechanism proposed in [1]. Time evolution of temperature and mass fraction of the chemical species are shown in Fig. 17. Flame front and temperature evolution in the domain are computed on the axis centerline.

The hybrid and the reference solutions show a good agreement for all the simulations. Minor discrepancies are visible in Fig. 17a and 17b. The propagation of the laminar flame is presented in Fig. 17c.

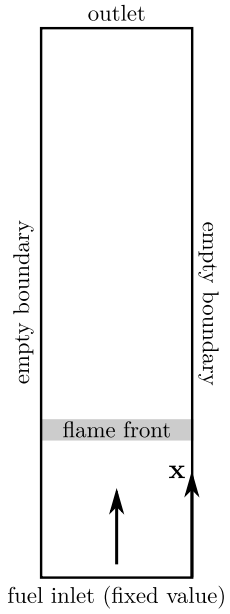


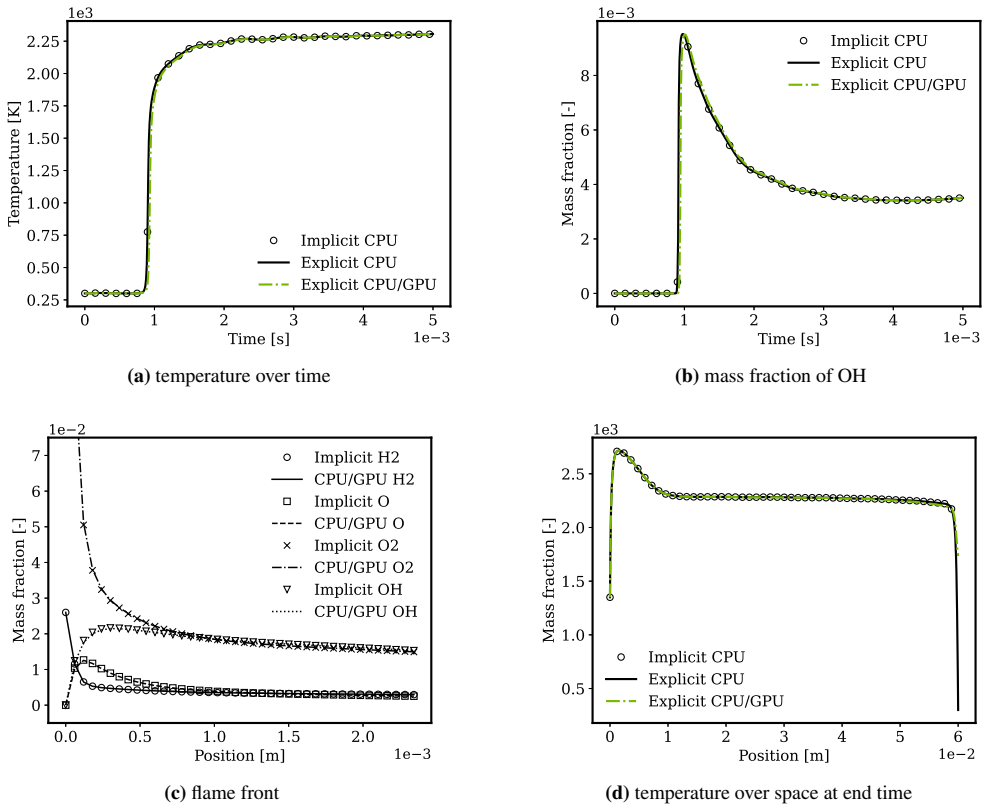
FIGURE 16 Flame propagation 1D test. Domain length L = 0.06 m. Mesh resolutions: 800, 3200, 9600, 15000, 20000 cells over the axial x-direction.

Parameter	Hydrogen case value	Ethylene case value	Unit
$\lvert \boldsymbol{U}_{in} \rvert$	0.4	0.4	[m/s]
T_{in}	1350	1350	[K]
Y_{fuel}	0.026	0.058	[-]
Y_{O_2}	0.204	0.198	[-]
Y_{N_2}	0.77	0.744	[-]

TABLE 2 Inlet BC for the one-dimensional flame propagation test.

Parameter	Hydrogen case value	Ethylene case value	Unit
T_{if}	300	600	[K]
p_{if}	101325	101325	[Pa]

TABLE 3 Internal field conditions for the one-dimensional flame propagation test.



(a) temperature over time

(b) mass fraction of OH

(c) flame front

(d) temperature over space at end time

FIGURE 17 Hydrogen/air reaction [1] in the 1D shock tube of Fig. 16; the probe to collect time-varying quantities is located at 15% of the domain elongation.

There is no sensible difference between the flame front position provided by the hybrid methodology and that of the reference. To conclude the analysis, the speed-up factors for the studied 1D shock tube configurations are proposed in Tab. 4.

Mechanism	Authors	Species no.	Reactions no.	Speed-up at number of cells				
				800	3200	9600	15000	20000
Hydrogen	Hong et al. [60]	10	31	1.84	2.02	2.06	2.01	2.01
	Sánchez [61]	11	23	2.06	2.30	2.31	2.27	2.27
	Burke et al. [1]	13	27	2.10	2.27	2.30	2.29	2.28
Ethylene	UCSD [63]	57	268	2.42	2.50	2.50	2.48	2.56
	Laskin [64]	75	529	2.56	2.59	2.65	2.59	2.60

TABLE 4 speed-up factor for the 1D shock tube test; mixture of Hydrogen/air or Ethylene/air at atmospheric pressure and inflow temperature $T_{in} = 1350$ K.

For all configurations, the use of the newly presented hybrid CPU/GPU methodology is convenient compared to its CPU counterpart. The computational time is more than halved. As the chemical reaction chain becomes more complex, the computational advantage increases. As the number of cells grows, the computational advantage enhances, up to a maximum which is case/configuration dependent.

Two-dimensional counter flow flame. To further extend the analysis carried out on multi-cell cases, reactive flow simulations are now developed on the two-dimensional counter flow flame configuration of Fig. 18. Boundary conditions are in Tab. 5. Simulations are initially performed on grids of different resolutions, namely of 800, 3200, 9600, 15000, and 20000 cells.

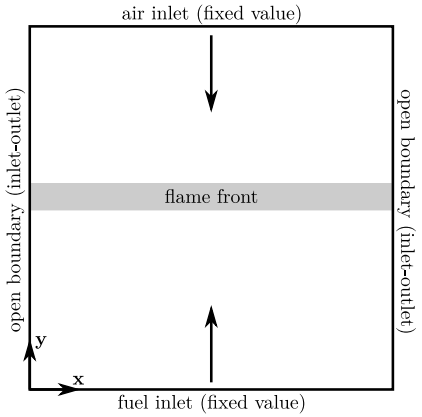


FIGURE 18 Two-dimensional domain for laminar flame test: length L = 0.06 m; mesh resolutions are 800, 3200, 9600, 15000, 20000.

Parameter	Patch	Value	Unit
$\lvert U_{in} \rvert$	fuel and air inlet	1.0	[m/s]
T_{in}	fuel and air inlet	293	[K]
Y_{CH_4}	fuel inlet	1	[-]
Y_{O_2}	air inlet	0.23	[-]
Y_{N_2}	air inlet	0.77	[-]
T_{if}	internal field	2000	[K]
p_{if}	internal field	1e5	[Pa]

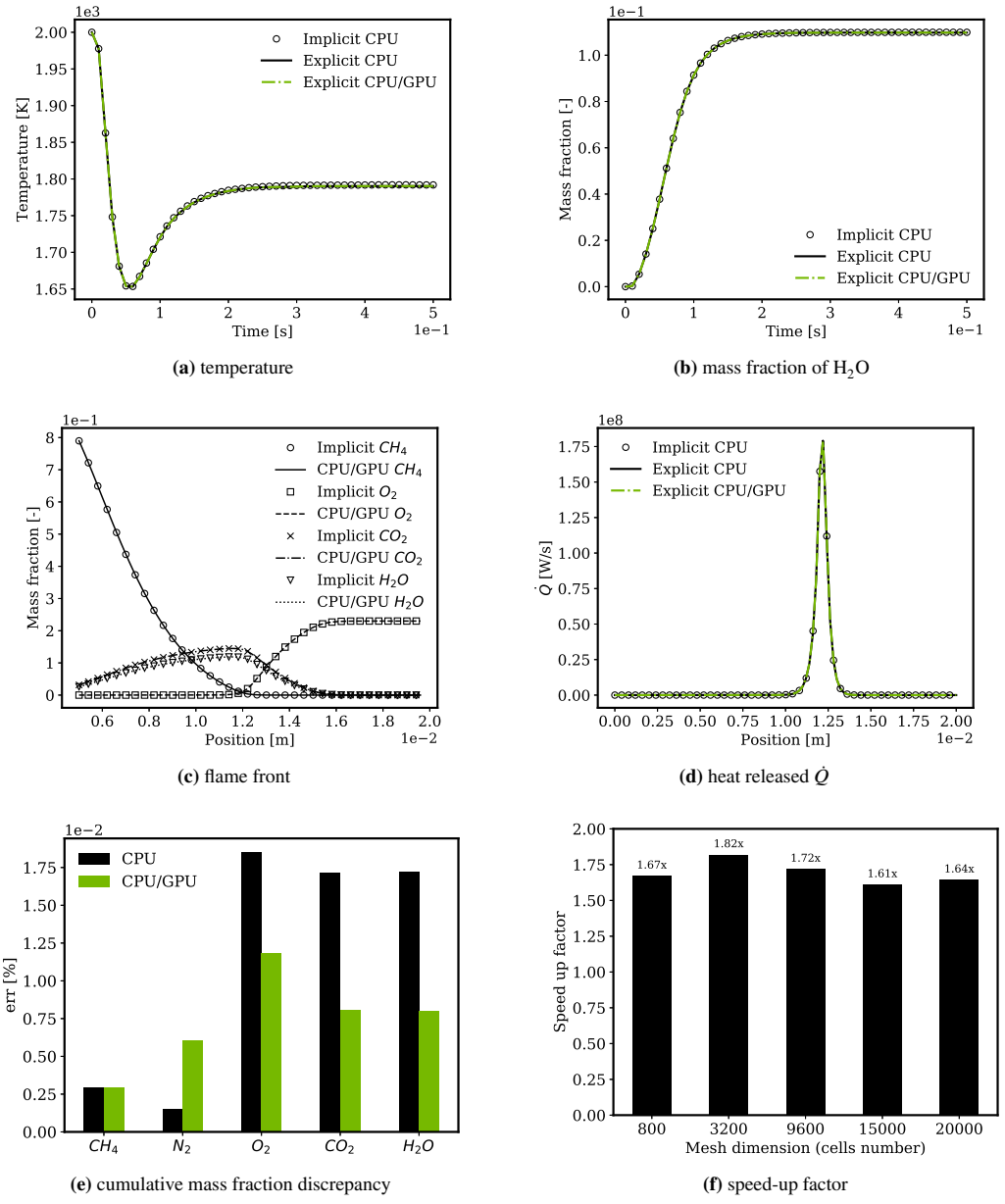
TABLE 5 Initial and boundary conditions for the two-dimensional counter flow flame test.

Initially, a non-stiff single reaction composed of 5 species is investigated. The mechanism corresponds to the global methane reaction:

$$CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O \tag{31}$$

Quantities in Figs. 19a-19d are defined at different positions over the axis centerline of the domain as function of time. The flame front is defined from the evolution of the mass fraction of the chemical species and of the heat released $\dot{Q}$: at the flame front, the concentration of reactants decreases, while new products are formed (Fig. 19c) and the heat released increases (Fig. 19d). For further clarification, a two-dimensional view of the temperature distribution at two different timesteps is reported in Fig. 20. In there, the results obtained via the GPU-ODE integrator are in good agreement with those produced by the legacy CPU code. Also, Fig. 19e and 19f show a speed-up greater than 1.5X: that already accounts for memory allocation, data copy and two-way data transfer for each fluid dynamic iteration. The apparently limited gain in performance is clearly linked to the small mechanism and the reduced grid sizes of this test. The adopted GPU (NVIDIA Tesla V100) can concurrently handle up

to 16 cells/blocks per streaming multiprocessor; only 5 threads per block are used in this situation, corresponding to a load of about 50%.



(a) temperature

(b) mass fraction of H_2O

(c) flame front

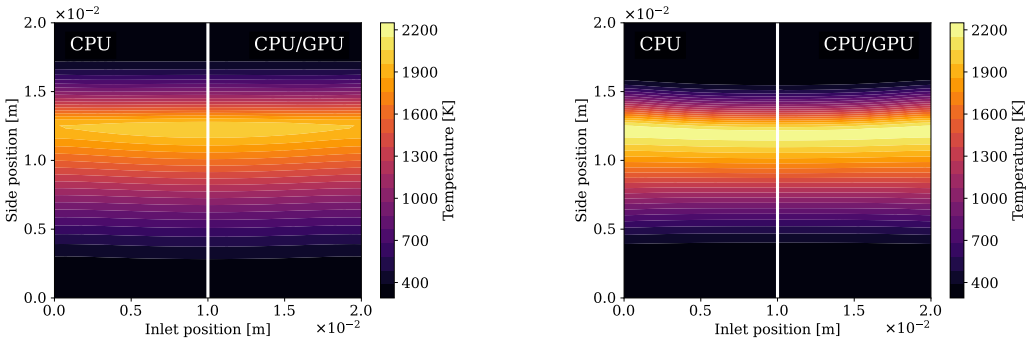
(d) heat released $\dot{Q}$

(e) cumulative mass fraction discrepancy

(f) speed-up factor

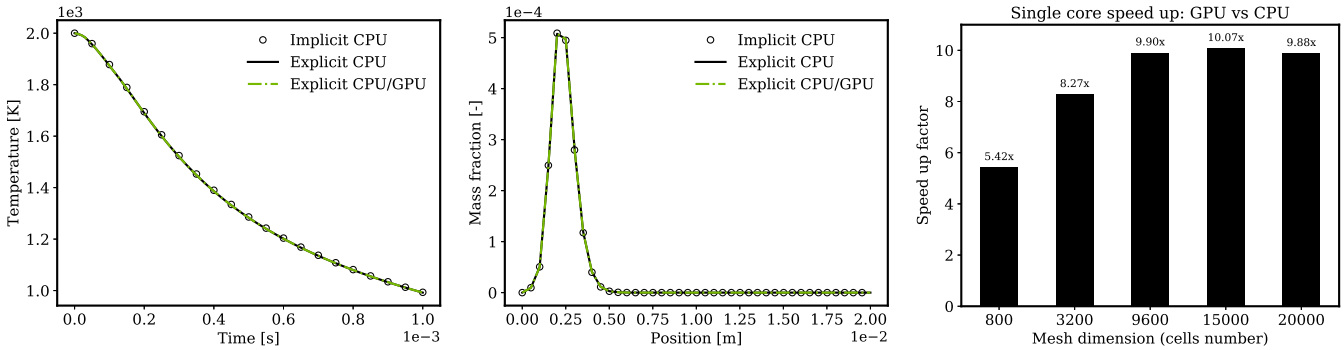
FIGURE 19 Methane/air single global reaction in the two-dimensional counter flow flame domain of Fig. 18: (a-b) time varying quantities collected by a probe located at the center of the domain; (c-d) data at end time $t = 0.5$ s over a center line spacing through the domain from inlet to outlet; (e) cumulative mass fraction discrepancy based on Eq. 28 and 29.

Performance is expected to improve for larger problems in which the two-level parallelization and the fat-thread approach play an important role. The same setup is tested with the GRI-Mech 3.0 mechanism. Since the correct handling of the GRI chain is validated for single-cell batch reactor tests, a simulation of 10^{-3} s is considered here to ease the profiling of the data. Initial conditions ensure that chemical reaction occurs in this time interval (see Fig. 21b). Fig. 21 shows the evolution of temperature over time by an internal probe located at the center of the domain, the distribution of the mass fraction of an intermediate product (CH_3) along an axial line, and the obtained speed-up.



(a) t = 0.05 s **(b)** t = 0.5 s

FIGURE 20 Methane/air single global reaction in the two-dimensional counter flow flame domain of Fig. 18: contour plot of temperature distribution.



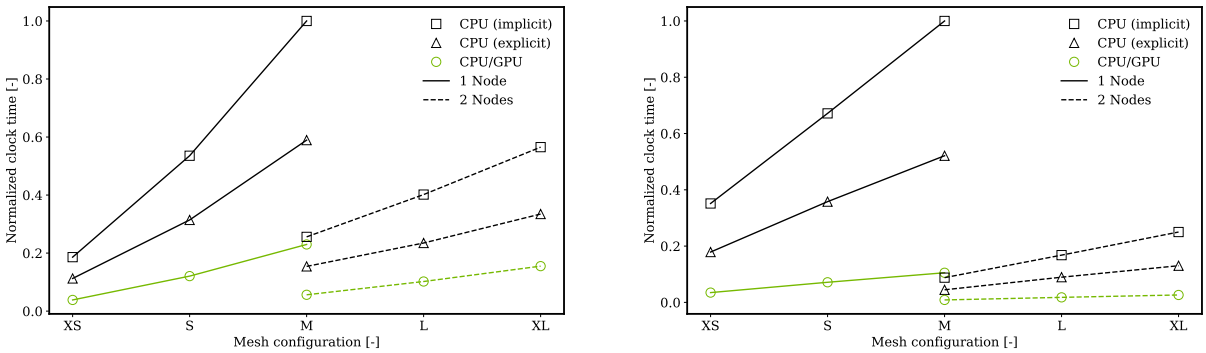
(a) temperature over time **(b)** mass fraction of CH_3 **(c)** speed-up factor

FIGURE 21 Methane/air (GRI-Mech 3.0 chain) reaction in the two-dimensional counter flow flame domain of Fig. 18; time varying quantities (a) collected by a probe located at the center of the domain.

Results from hybrid CPU/GPU computations are reported in Fig. 21a and 21b. As expected, the speed-up factor with large mechanisms is significant (Fig. 21c). The hybrid CPU/GPU solver is more than 5 times faster than its CPU counterpart for the very coarse mesh. For the domain composed of 3200 cells, the speed-up factor is 8.27x, and for finer meshes it reaches a value greater than 9.5x, with a peak of 10x for the domain with 15 thousand cells. The speed-up factor is therefore dependent on the number of reacting cells, the number of involved chemical species and the complexity of the chemical chain. This also indicates that the speed-up grows as the number of cells increases, thanks particularly to those contained in the thin flame front region which spaces along the spanwise direction.

Further details about performance. Reactive flow simulations are now conducted on refined versions of the counter flow flame mesh. Cases are denoted XS, S, M, L, XL. The coarsest and finest meshes comprise 1 and 3 million cells, respectively, with intermediate meshes progressively refined by adding 500 thousand more cells. Meshes XS to M are studied using 1 node (CPU only) or 1 node + 1 GPU (CPU/GPU configuration); meshes M to XL are investigated using 2 nodes (CPU only) or 2 nodes + 2 GPUs (CPU/GPU configuration). CPU-only results employ explicit or implicit ODE integration. For an adequate comparison, the explicit ODEs integrator is the Runge-Kutta Cash-Karp; the implicit integrator is an extrapolation-algorithm based on the linearly implicit Euler method with step size control and order selection. The selected chemical mechanisms are the Hydrogen chain of Burke et al. [1] and the Methane mechanism GRI-Mech 3.0 [62]. The maximum Courant number is set to 0.3. Results of the performance tests are reported in Fig. 22.

All data required for solving the chemical ODE system can reside concurrently on the GPUs and be managed by the devices. Fig. 22a demonstrates that, across all tested conditions, the hybrid CPU/GPU approach outperforms CPU-only methods. For both



(a) Hydrogen reaction [1]

(b) GRI-Mech 3.0 reaction [62]

FIGURE 22 performance of the hybrid approach on refined meshes of the counter flow flame test case; normalization with respect to slowest run; for each mesh configuration, lower is better.

single-node and multi-node hybrid configurations, the speedup consistently ranges around 3x compared to explicit CPU methods and 4x to 4.5x against implicit ODE integration. On average, the time advancement for the Hydrogen chain is approximately 5×10^{-6} s (for mesh XS) and 10^{-6} s (for mesh XL); for the Methane chain, these values are about 4×10^{-7} s (for mesh XS) and 2×10^{-7} s (for mesh XL). More lenient tolerances are used for the Hydrogen chain integration compared to the stiffer GRI-Mech 3.0 chain. With less restrictive tolerances, one reduces the number of rejected and repeated integrations (err > tol) in the considered timestep. That positively affects the speedup, but potentially impacts the stability of explicit integration (both on CPU and GPU). Lenient tolerances pose almost no threat to non-stiff chains; more stringent tolerances are required for stiff ODEs integration. At the same time, GPGPU computing benefits from increased GPU load (more species, reactions) because of the double parallelization. The number of cells that can be concurrently handled by the GPU is hardware dependent. For increasing number of cells in the computational domain, the queue on the GPU lengthens; that contributes to reduce the achievable performance gain. For the considered GRI-Mech 3.0 cases, GPGPU computing produces a performance gain of about 5x and 9x against explicit and implicit ODEs integration respectively (Fig. 22b).

8 | CONCLUSIONS

The algorithmic co-design of a GPGPU solver for heterogeneous system architectures for the rapid solution of reactive flow problems is presented. In the computations, a three-level parallelization is employed: 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, where blocks and threads execute the simultaneous/parallel solution of the reaction mechanism. As a result, the explicit 5^{th} 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. For the mechanisms tested, the solution of the chemistry problem on the GPU would be about two orders of magnitude faster than on the CPU. On the other hand, the performance gain is limited to about 10x, because of some bottlenecks: a) the slow transfer of data between CPU and GPUs, occurring at each time step of integration of the fluid solver in reactive flow computation; b) the stiffness of the ODE system: to take advantage of the parallel architecture of the GPU, the use of explicit integrators is favored. Future improvements include the coupling of the proposed approach with a stiffness reduction method operating on the GPU and the design of implicit GPU-ODE integrators, to take advantage of the upcoming unified memory capabilities of next generation GPUs.

ACKNOWLEDGEMENTS

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 Argonne National Laboratory (Lemont, US) for the computing resources provided. The method proposed in this manuscript has been developed by the authors at the Dept. of Aerospace Science and Technology (DAER) at Politecnico di Milano (PoliMi). The resulting C++ dynamic libraries were linked to the most recent official versions of the software OpenFOAM released by the OpenFOAM Foundation [41] and ESI-OpenCFD [66].

CONFLICT OF INTEREST STATEMENT

The authors declare no potential conflict of interests.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

[1] Burke MP, Chaos M, Ju Y, Dryer FL, Klippenstein SJ. Comprehensive H2/O2 kinetic model for high-pressure combustion. *International Journal of Chemical Kinetics* 2012; 44(7): 444-474. doi: 10.1002/kin.20603

[2] Qin Z, Lissianski VV, Yang H, Gardiner WC, Davis SG, Wang H. Combustion chemistry of propane: A case study of detailed reaction mechanism optimization. *Proceedings of the Combustion Institute* 2000; 28(2): 1663 - 1669. doi: 10.1016/S0082-0784(00)80565-2

[3] Sheffer S, Jameson A, Martinelli L. *Parallel computation of supersonic reactive flows with detailed chemistry*; 1997

[4] Herbinet O, Pitz W, Westbrook C. Detailed chemical kinetic mechanism for the oxidation of biodiesel fuels blend surrogate. *Combustion and Flame* 2010; Vol. 157(Iss. 5): pp. 893-908. doi: 10.1016/j.combustflame.2009.10.013

[5] Wichman IS. On the use of operator-splitting methods for the equations of combustion. *Combustion and Flame* 1991; 83(3): 240-252. doi: https://doi.org/10.1016/0010-2180(91)90072-J

[6] Descombes S, Duarte M, Massot M. *Operator Splitting Methods with Error Estimator and Adaptive Time-Stepping. Application to the Simulation of Combustion Phenomena*: 627–641; Cham: Springer International Publishing . 2016

[7] Yang B, Pope S. An investigation of the accuracy of manifold methods and splitting schemes in the computational implementation of combustion chemistry. *Combustion and Flame* 1998; 112(1): 16-32. doi: 10.1016/S0010-2180(97)81754-3

[8] Singer M, Pope S, Najm H. Modeling unsteady reacting flow with operator splitting and ISAT. *Combustion and Flame - COMBUST FLAME* 2006; 147: 150-162. doi: 10.1016/j.combustflame.2006.06.007

[9] Ren Z, Xu C, Lu T, Singer MA. Dynamic adaptive chemistry with operator splitting schemes for reactive flow simulations. *Journal of Computational Physics* 2014; 263: 19-36. doi: 10.1016/j.jcp.2014.01.016

[10] Lu Z, Zhou H, Li S, Ren Z, Lu T, Law CK. Analysis of operator splitting errors for near-limit flame simulations. *Journal of Computational Physics* 2017; 335: 578-591. doi: https://doi.org/10.1016/j.jcp.2017.01.044

[11] Wu H, Ma PC, Ihme M. Efficient time-stepping techniques for simulating turbulent reactive flows with stiff chemistry. *Computer Physics Communications* 2019; 243: 81-96. doi: https://doi.org/10.1016/j.cpc.2019.04.016

[12] Fahs M, Younes A, Ackerer P. An Efficient Implementation of the Method of Lines for Multicomponent Reactive Transport Equations. *Water Air Soil Pollut* 2011; 215: 273-283. doi: 10.1007/s11270-010-0477-y

[13] Lukassen AA, Kiehl M. Operator splitting for chemical reaction systems with fast chemistry. *Journal of Computational and Applied Mathematics* 2018; 344: 495-511. doi: doi.org/10.1016/j.cam.2018.06.001

[14] Sportisse B, Bencteux G, Plion P. Method of Lines versus Operator Splitting for reaction–diffusiuon systems with fast chemistry. *Environmental Modelling & Software* 2000; 15(6): 673-679. Air pollution modelling and simulationdoi: https://doi.org/10.1016/S1364-8152(00)00036-0

[15] Lu T, Law C. Toward accommodating realistic fuel chemistry in large-scale computations. *Progress in Energy and Combustion Science* 2009; Vol. 135(Iss. 2): pp. 192-215. doi: 10.1016/j.pecs.2008.10.002

[16] Bhattacharjee B, Schwer D, Barton P, Green W. Optimally-reduced kinetic models: reaction elimination in large-scale kinetic mechanisms. *Combustion and Flame* 2003; 135: pp. 191–208. doi: 10.1016/S0010-2180(03)00159-7

[17] Singer MA, Pope SB. Exploiting ISAT to solve the reaction–diffusion equation. *Combustion Theory and Modelling* 2004; 8(2): 361-383. doi: 10.1088/1364-7830/8/2/009

[18] Goldin GM, Ren Z, Zahirovic S. A cell agglomeration algorithm for accelerating detailed chemistry in CFD. *Combustion Theory and Modelling* 2009; 13(4): 721-739. doi: 10.1080/13647830903154542

[19] Kee RJ, Coltrin ME, Glarborg P. *Chemically Reacting Flow. Theory and Practice*. John Wiley, New York . 2003.

[20] Liang L, Stevens JG, Raman S, Farrell JT. The use of dynamic adaptive chemistry in combustion simulation of gasoline surrogate fuels. *Combustion and Flame* 2009; 156(7): 1493 - 1502. doi: 10.1016/j.combustflame.2009.02.008

[21] 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 International; 2012

[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): 10121-10131. doi: 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. *Combustion and Flame* 1998; 113(1): 38 - 52. doi: 10.1016/S0010-2180(97)00211-3

[24] Nikitin V, Karandashev I, Malsagov MY, Mikhalchenko E. Approach to combustion calculation using neural network. *Acta Astronautica* 2022; 194: 376-382. doi: 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. *The Journal of Physical Chemistry A* 2021; 125(36): 8098-8106. doi: 10.1021/acs.jpca.1c05102

[26] Hawkes ER, Sankaran R, Sutherland JC, Chen JH. Direct numerical simulation of turbulent combustion: fundamental insights towards predictive models. *Journal of Physics: Conference Series* 2005; 16: 65–79. doi: 10.1088/1742-6596/16/1/009

[27] Spafford K, Meredith J, Vetter J, Chen J, Grout R, Sankaran R. Accelerating S3D: A GPGPU Case Study. In: ; 2009: pp. 122-131

[28] Niemeyer K, Sung CJ, Fotache C, J.C. L. Turbulence-chemistry closure method using graphics processing units: a preliminary test. *7th Fall Technical Meeting of the Eastern States Section of the Combustion Institute* 2011. doi: 10.6084/m9.figshare.3384964.v1

[29] Niemeyer KE, Sung CJ. Accelerating moderately stiff chemical kinetics in reactive-flow simulations using GPUs. *Journal of Computational Physics* 2014; 256: 854 - 871. doi: 10.1016/j.jcp.2013.09.025

[30] Cash J, Karp A. A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-Hand Sides. *ACM Transactions on Mathematical Software* 1990; Vol. 16(No. 3): pp. 201-222. doi: 10.1145/79505.79507

[31] Van Der Houwen P, B.P. S. On the Internal Stability of Explicit, m-Stage Runge-Kutta Methods for Large m-Values. *Journal of Applied Mathematics and Mechanics* 1980; Vol. 60(Iss. 10). doi: 10.1002/zamm.19800601005

[32] Verwer J. Explicit Runge-Kutta methods for parabolic partial differential equations. *Applied Numerical Mathematics* 1996; Vol. 22(Iss. 1-3): pp. 359-379. doi: 10.1016/S0168-9274(96)00022-0

[33] Stone C, Davis R. Techniques for Solving Stiff Chemical Kinetics on Graphical Processing Units. *Journal of Propulsion and Power* 2013; Vol. 29(No. 4). doi: 10.2514/1.B34874

[34] Shi Y, Green WH, Wong HW, Oluwole OO. Accelerating multi-dimensional combustion simulations using GPU and hybrid explicit/implicit ODE integration. *Combustion and Flame* 2012; 159(7): 2388 - 2397. doi: 10.1016/j.combustflame.2012.02.016

[35] Shampine L. Some Practical Runge-Kutta Formulas. *Mathematics of Computation* 1986; Vol. 46(No. 173): pp. 135-150.

[36] Brock B, Belt A, Billings JJ, Guidry M. Explicit integration with GPU acceleration for large kinetic networks. *Journal of Computational Physics* 2015; 302: 591 - 602. doi: 10.1016/j.jcp.2015.09.013

[37] Ferziger JH, Perić M, Street RL. *Computational Methods for Fluid Dynamics*. Springer. 4th edition ed. 2020.

[38] Press W, Teukolsky S, Vetterling W, Flannery B. *Numerical Recipes in C*. Cambridge University Press. 2nd ed. ed. 1997.

[39] Abdelfattah A, Keyes D, Ltaief H. KBLAS: An Optimized Library for Dense Matrix-Vector Multiplication on GPU Accelerators. *ACM Trans. Math. Softw.* 2016; 42(3). doi: 10.1145/2818311

[40] Curtis NJ, Niemeyer KE, Sung CJ. An investigation of GPU-based stiff chemical kinetics integration methods. *Combustion and Flame* 2017; 179: 312 - 324. doi: 10.1016/j.combustflame.2017.02.005

[41] The OpenFOAM Foundation . OpenFOAM: a free, open source software for Computational Fluid Dynamics. http://www.openfoam.org/dev.php; 2023.

[42] Goodwin DG, Speth RL, Moffat HK, Weber BW. Cantera: An Object-oriented Software Toolkit for Chemical Kinetics, Thermodynamics, and Transport Processes. https://www.cantera.org; 2018. Version 2.4.0

[43] Poinsot T, Veynante D. *Theoretical and Numerical Combustion*. CNRS. 3 ed. 2012.

[44] Gordon S, McBride B. Computer program for calculation of complex equi- librium compositions, rocket performance, incident and reflected shocks, and Chapman-Jouguet detonations. tech. rep., NASA SP-273; : 1971.

[45] Caretto LS, Gosman AD, Patankar SV, Spalding DB. Two calculation procedures for steady, three-dimensional flows with recirculation. In: Cabannes H, Temam R., eds. *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*Springer Berlin Heidelberg; 1973; Berlin, Heidelberg: 60–68.

[46] Peinado J, Ibáñez J, Arias E, Hernández V. Adams–Bashforth and Adams–Moulton methods for solving differential Riccati equations. *Computers & Mathematics with Applications* 2010; 60(11): 3032 - 3045. doi: 10.1016/j.camwa.2010.10.002

[47] Jasak H. *Error analysis and estimation in the Finite Volume method with applications to fluid flows*. PhD thesis. Imperial College, University of London, ; 1996.

[48] Rhie C, Chow WL. A numerical study of the turbulent flow past an isolated airfoil with trailing edge separation. *AIAA J.* 1983; 21: 1525-1532.

[49] Martínez J, Piscaglia F, Montorfano A, Onorati A, Aithal S. Influence of momentum interpolation methods on the accuracy and convergence of pressure–velocity coupling algorithms in OpenFOAM. *Journal of Computational and Applied Mathematics* 2017; 309: 654-673. doi: 10.1016/j.cam.2016.03.037

[50] NVIDIA Corporation & affiliates . CUDA C++ Programming Guide - Release 12.3. https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf; 2024.

[51] Volkov V. Better performance at lower occupancy. In: Proceedings of the 2008 ACM/IEEE conference on Supercomputing; 2008.

[52] Fehlberg E. *Some Experimental Results Concerning the Error Propagation in Runge-Kutta Type Integration Formulas*. NASA techical reportNational Aeronautics and Space Administration . 1970.

[53] Fehlberg E. *Low-order Classical Runge-Kutta Formulas with Stepsize Control and Their Application to Some Heat Transfer Problems*. NASA technical reportNational Aeronautics and Space Administration . 1969.

[54] Nickolls J, Dally WJ. The GPU Computing Era. *IEEE Micro* 2010; 30(2): 56-69. doi: 10.1109/MM.2010.41

[55] Lindholm E, Nickolls J, Oberman S, Montrym J. NVIDIA Tesla: A Unified Graphics and Computing Architecture. *IEEE Micro* 2008; 28(2): 39-55. doi: 10.1109/MM.2008.31

[56] Branch Statistics. https://docs.nvidia.com/gameworks/content/developertools/desktop/analysis/report/cudaexperiments/kernellevel/branchstatistics.htm; . Accessed: 2023/02/03.

[57] NVIDIA , Vingelmann P, Fitzek FH. CUDA, release: 10.2.89.; 2020.

[58] Klingbeil G, Erban R, Giles M, Maini PK. Fat versus Thin Threading Approach on GPUs: Application to Stochastic Simulation of Chemical Reactions. *IEEE Transactions on Parallel and Distributed Systems* 2012; 23(2): 280-287. doi: 10.1109/TPDS.2011.157

[59] Kuo K. *Principles of Combustion*. Wiley. 2 ed. 2005.

[60] Hong Z, Davidson DF, Hanson RK. An improved H2/O2 mechanism based on recent shock tube/laser absorption measurements. *Combustion and Flame* 2011; 158(4): 633-644. Special Issue on Kineticsdoi: 10.1016/j.combustflame.2010.10.002

[61] Sánchez AL, Williams FA. Recent advances in understanding of flammability characteristics of Hydrogen. *Progress in Energy and Combustion Science* 2014; 41: 1-55. doi: 10.1016/j.pecs.2013.10.002

[62] Frenklach M, Wang H, Goldenberg M, et al. GRI-Mech - An Optimized Detailed Chemical Reaction Mechanism for Methane Combustion. 2000.

[63] Chemical-kinetic mechanisms for combustion applications. https://www.cerfacs.fr/cantera/mechanisms/eth.php#ucsd; .

[64] Wang H, Laskin A. A comprehensive kinetic model of ethylene and acetylene oxidation at high temperatures.; 1998. Progress Report for an AFOSR New World Vista Program. Available at [link] .

[65] Explosion Dynamics Laboratory, California Institute of Technology. https://shepherd.caltech.edu/EDL/PublicResources/sdt/cti_mech.html; .

[66] ESI-OpenCFD . OpenFOAM: a free, open source software for Computational Fluid Dynamics. http://www.openfoam.com; 2023.

□

APPENDIX

A LINK BETWEEN DENSITY AND PRESSURE CORRECTION

In the fully implicit unsteady solver, *outer* iterations (denoted with m in the following) are repeated multiple times while solving across the time interval $[n; n+1]$; the *outer*-loop ends when the entire set of non-linear equations satisfies the solver's stopping criteria. These iterations are different from the *inner* ones which are performed on linear systems with fixed coefficients.

Similarly to the procedure employed by SIMPLE-type methods for incompressible flows [45], the momentum equations are used to calculate an intermediate velocity field $\boldsymbol{U}^*$:

$$\left.\frac{\partial(\rho \boldsymbol{U})}{\partial t}\right|_n^* + \nabla \cdot (\rho^{m-1}\boldsymbol{U}^n \cdot \boldsymbol{U}^*) = -\nabla p^{m-1} + \nabla \cdot \boldsymbol{R} + \boldsymbol{S}_U \tag{A1}$$

In Eq. A1 the density ρ^{m-1} comes from previous outer iteration $m-1$. If the flow viscosity depends on temperature or other variables, the viscous terms are also computed using quantities from the previous iteration. The velocity field $\boldsymbol{U}^*$ obtained via the linearized momentum equations structured on the old pressure and density values does not satisfy the mass conservation equation. In fact, when the mass fluxes computed using these velocities and the old density are inserted into the continuity equation, a mass imbalance is produced in the volume V and must be eliminated by a correction method:

$$\left.\frac{\partial \rho}{\partial t}\right|_n^m V + \sum_f (\dot{m}_f^* + \dot{m}_f') = 0 \tag{A2}$$

where:

$$\dot{m}_f = \dot{m}_f^* + \dot{m}_f' = (\rho^* + \rho')_f (\boldsymbol{U}^* + \boldsymbol{U}')_f \boldsymbol{S}_f = \underbrace{(\rho^* \boldsymbol{U}^*)_f \cdot \boldsymbol{S}_f}_{\dot{m}_f^*} + \underbrace{(\rho^* \boldsymbol{U}' + \rho' \boldsymbol{U}^* + \cancel{\rho' \boldsymbol{U}'})_f \cdot \boldsymbol{S}_f}_{\text{pressure correction } \dot{m}_f'} \tag{A3}$$

The second-order term in Eq. A3 is neglected because is assumed to go to zero more rapidly than the other terms. Such an approximation doesn't influence the convergence rate and doesn't impact the final solution because it tends to zero at convergence. It follows that:

$$\dot{m}_f' = (\rho^* \boldsymbol{U}' + \rho' \boldsymbol{U}^*) \cdot \boldsymbol{S}_f \tag{A4}$$

in which $\boldsymbol{S}_f$ is the surface normal vector; $\boldsymbol{U}'$ and $\boldsymbol{p}'$ are defined from:

$$\boldsymbol{U}^m = \boldsymbol{U}^* + \boldsymbol{U}' \tag{A5}$$

$$p^m = p^{m-1} + p' \tag{A6}$$

and for the density:

$$\rho^m = \rho^{m-1} + \rho' \tag{A7}$$

While the pressure gradient only corrects the flow velocity in low-Mach implementations, in pressure-based compressible solvers the pressure correction acts on the density. The first term of Eq. A4 is similar to the term for incompressible flows, while the second term includes a density correction, that goes to zero in low-Mach number cases.

The intermediate velocity field $\boldsymbol{U}^*$ calculated from the predictor step (Eq. A1) must be corrected by a pressure gradient to enforce mass conservation in the domain. The combination of Eq. A2 and A3 written in differential form reads:

$$\left.\frac{\partial \rho}{\partial t}\right|_n^m + \nabla \cdot (\rho' \boldsymbol{U}^*) + \nabla \cdot (\rho^* \boldsymbol{U}') = -\nabla \cdot \rho^* \boldsymbol{U}^* \tag{A8}$$

Eq. A8 is called pressure correction equation; its solution determines the correction to be applied to the mass fluxes computed via the use of the intermediate velocity field $\boldsymbol{U}^*$. Being the correction done through pressure, the thermodynamic state depends on the correction procedure as well. The time derivative in Eq. A8 can be decomposed as follows:

$$\left.\frac{\partial \rho}{\partial t}\right|_n^m = \left.\frac{\partial \rho}{\partial t}\right|_n^* + A_P \frac{\partial \rho'}{\partial t} = \left.\frac{\partial \rho}{\partial t}\right|_n^* + \left(\frac{A_P \psi}{\Delta t}\right) p' \tag{A9}$$

In Eq. A9, ψ is the fluid compressibility (see Appendix B), while the A_P coefficient depends on the adopted time differencing scheme. For the three-time-level scheme (backward Euler method) $A_P = \frac{3}{2}$, while for the two-level first order implicit method $A_P = 1$. Also, being (Appendix B):

$$\nabla \cdot (\rho' \boldsymbol{U}^*) = \nabla \cdot \left(\boldsymbol{U}^* \psi\, p'\right) \tag{A10}$$

and (Appendix C):

$$\nabla \cdot (\rho^* \boldsymbol{U}') = -\rho^* \left(\frac{\Delta t}{A_P}\right) \nabla^2 p' \tag{A11}$$

the final form of the pressure correction equation is written as follows:

$$\left(\frac{A_P\psi}{\Delta t}\right)p' + \nabla\cdot\left(\boldsymbol{U}^*\psi p'\right) - \rho^*\left(\frac{\Delta t}{A_P}\right)\nabla^2 p' = -\left[\left.\frac{\partial\rho}{\partial t}\right|_n^* + \nabla\cdot(\rho^*\boldsymbol{U}^*)\right] \tag{A12}$$

Eq. A12 includes an incompressible divergence term to correct the mass fluxes and a compressible convective term to correct the density. They alternatively become dominant when the flow is largely incompressible or compressible, respectively, making the pressure-correction strategy applicable over a wide range of applications at all flow speeds [37]. At low Mach number values, the $\nabla p'$ correction term dominates, and Eq. A12 assumes an elliptic form for incompressible flow cases; at high Mach numbers, the contribution of the term containing p' enlarges and Eq. A12 assumes an hyperbolic form. Upon solving the pressure-correction equation, velocity and density are updated to obtain $\boldsymbol{U}^m$ and ρ^*; those values are used to solve the energy equation at the next step, from which the updated internal energy e^m is obtained. In turn, the temperature T^m is determined via thermodynamics. The new density is computed from the EoS (see Appendix B):

$$\rho = \psi T \tag{A13}$$

and the velocity is updated:

$$\boldsymbol{U}^{n+1} = \boldsymbol{U}^* + \nabla p' \tag{A14}$$

Finally, if turbulence modeling is active, turbulence is solved and turbulent viscosity is updated. After a sufficient number of iterations, the corrections become negligible and the state $m \rightarrow (n+1)$; the solver can then proceed to the calculation of the next time step.

B LINK BETWEEN DENSITY AND PRESSURE CORRECTION

The link between density correction and pressure correction is given by:

$$\rho = \left.\frac{\partial\rho}{\partial p}\right|_T p = \psi p \tag{B15}$$

where

$$\psi = \left.\frac{\partial\rho}{\partial p}\right|_T \tag{B16}$$

is the compressibility of the fluid. Being:

$$\rho' = \rho^* - \rho^{m-1} = \psi p^m - \psi p^{m-1} = \psi(p^m - p^{m-1}) = \psi p' \tag{B17}$$

if follows:

$$\rho' = \psi p' \tag{B18}$$

C LINK BETWEEN VEOCITY AND PRESSURE CORRECTION

At the m-th outer iteration within the time step integration from time n to $n+1$, the momentum equation can be written as:

$$\left.\frac{\partial(\rho^{m-1}\boldsymbol{U})}{\partial t}\right|_n^* + \nabla\cdot(\rho^{m-1}\boldsymbol{U}^n\cdot\boldsymbol{U}^*) = -\nabla p^{m-1} + \nabla\cdot\boldsymbol{R}(\boldsymbol{U}^*) + \boldsymbol{S}_U(\boldsymbol{U}^*) \tag{C19}$$

The corrected velocity and pressure must also satisfy:

$$\left.\frac{\partial(\rho^{m-1}\boldsymbol{U})}{\partial t}\right|_n^m + \nabla\cdot(\rho^{m-1}\boldsymbol{U}^n\cdot\boldsymbol{U}^*) = -\nabla p^m + \nabla\cdot\boldsymbol{R}(\boldsymbol{U}^*) + \boldsymbol{S}_U(\boldsymbol{U}^*) \tag{C20}$$

By subtracting Eq. C19 from C20, it follows:

$$\frac{1}{\Delta t}A_P\boldsymbol{U}' = -\nabla p' \tag{C21}$$

being:

$$\left.\frac{\partial(\rho^{m-1}\boldsymbol{U})}{\partial t}\right|_n^m - \left.\frac{\partial(\rho^{m-1}\boldsymbol{U})}{\partial t}\right|_n^* = \frac{\partial(\rho^{m-1}\boldsymbol{U}')}{\partial t} = \frac{1}{\Delta t}A_P\boldsymbol{U}' \tag{C22}$$

where

$$A_P = \begin{cases} 1 & \text{with a } 1^{\text{st}} \text{ order two-time-level time differencing scheme} \\ \frac{3}{2} & \text{with a } 2^{\text{nd}} \text{ order three-time-level time differencing scheme} \end{cases}$$

From Eq. C22, the time differencing scheme applied for temporal discretization appears in the coefficient A_P only. Eq. C21 can be manipulated and written as:

$$\rho^*\boldsymbol{U}' = -\rho^*\left(\frac{\Delta t}{A_P}\right)\nabla p' \tag{C23}$$

being ρ^* the density updated at the current outer iteration m (e.g. $\rho^* \equiv \rho^m$). Finally:

$$\nabla\cdot(\rho^*\boldsymbol{U}') = -\rho^*\left(\frac{\Delta t}{A_P}\right)\nabla^2 p' \tag{C24}$$