On the feasibility of overnight industrial high-fidelity simulations of CSP technologies on modern HPC systems

A Alsalti-Baldellou[1], G Colomer[1] , J A Hopman[1], X Álvarez-Farré[3], A Gorobets[4] , F X Trias[1], C D Pérez-Segarra[1] and A Oliva[1]

[1]Heat and Mass Transfer Technological Center, Universitat Politècnica de Catalunya – BarcelonaTech, Carrer de Colom 11, 08222 Terrassa (Barcelona), Spain
[2]High-Performance Computing Team, SURF, Science Park 140, 1098 XG Amsterdam, The Netherlands
[3]Keldysh Institute of Applied Mathematics, 4A, Miusskaya Sq., Moscow 125047, Russia

E-mail: adel@termofluids.com

Abstract. In the last decades, computational fluid dynamics (CFD) has become a standard design tool in many fields, such as the automotive, aeronautical, and renewable energy industries. The driving force behind this is the development of numerical techniques in conjunction with the progress of high-performance computing (HPC) systems. However, simulation time remains the most limiting factor for large-eddy simulations (LES) to be adopted in the industry. A consensus exists that, to be feasible, LES simulations should be completed overnight In this context, this work assesses the feasibility of overnight LES simulations on GPU-accelerated supercomputers with TFA, our novel in-house code, which relies on a symmetry-preserving discretisation for unstructured collocated grids that, apart from being virtually free of artificial dissipation, is shown to be unconditionally stable. The study cases will be taken from central receivers used in concentrated solar power (CSP) plants, and a comparison with open-source CFD codes will be made.

1. Introduction

In the last decades, CFD has become a standard design tool in many fields, such as the automotive, aeronautical, and renewable energy industries. The driving force behind this is the development of numerical techniques in conjunction with the progress of high-performance computing (HPC) systems. However, progress is nowadays hindered by its legacy from the 90-2000s. The reasons are two-fold. Firstly, the design of digital processors constantly evolves to overcome limitations and bottlenecks. The formerly compute-bound nature of processors led to compute-centric programming languages and simulation codes. However, raw computing power grows faster than the memory bandwidth, turning around the problem and leading to increasingly complex memory hierarchies that make optimising traditional applications a cumbersome task. Moreover, new parallel programming languages emerged to target modern hardware, *e.g.*, CUDA, HIP and oneAPI, and porting algorithms and applications has become restrictive. For this reason, designing more abstract modular codes kept gaining interest. For instance, the PyFR framework (see [1]) is mostly based on matrix multiplications and

point-wise operations. Other examples are Kokkos and RAJA libraries (see [2, 3]), which provide an abstraction layer aiming at providing performance portability. Remarkably enough, implementing modular codes allow linking to standard libraries optimized for particular architectures, in addition to specialized in-house implementations. Examples of this include cuSPARSE and clSPARSE (see [4, 5]).

Secondly, legacy numerical methods chosen to solve (quasi)steady problems using RANS models are inappropriate for more accurate (and expensive) techniques such as large-eddy simulation (LES) or direct numerical simulation (DNS). We aim to interlace these two pillars with the final goal of enabling LES and DNS of industrial applications to be efficiently carried out on modern HPC systems while keeping codes easy to port, optimise, and maintain. In this regard, TermoFluids Algebraic (TFA), our novel in-house code, adopts the fully-conservative discretisation for collocated unstructured grids proposed in [6]: it constitutes a very robust approach that can be easily implemented in existing codes such as OpenFOAM in [7].

The main recognised limitations of LES in the industry are their computational cost and the wall-clock simulation time. Thanks to the above-explained advent of new computational architectures, the former is becoming less and less critical, whereas the latter is still the most limiting factor precluding LES from being routinely used in the industry. For that to be possible, the consensus is that widespread adoption in the industry begins when a run can be carried out overnight, see, for instance, [8]. Namely, the industry is governed by shortening design cycles, faster time-to-market, and increased expectations of operability and reliability for established product lines. Therefore, it is willing to spend on hardware and software as long as analysts can obtain meaningful insights in a time commensurate with design cycles. Overnight runs fit this timescale, and in this context, we aim to answer the following question: *can we use LES modelling to simulate complex industrial flows with overnight simulations accurately?*

2. Rethinking CFD for present and future portability

Building codes on top of a minimal set of kernels is the cornerstone for portability and optimisation, which became crucial after the increasing variety of computational architectures and vendors competing in the exascale race. Moreover, the hybridisation of HPC systems imposes additional constraints since heterogeneous computations are usually needed to engage processors and massively parallel accelerators efficiently. This involves different parallel paradigms and computing frameworks and requires complex data exchanges between computing units. However, legacy CFD codes usually rely on sophisticated data structures and computing subroutines, making portability extremely complex. In this context, we proposed a completely different approach [9]. That is, making CFD algorithms rely on a very reduced set of algebraic kernels, *e.g.*, the sparse matrix-vector product (`SpMV`), the dot product (`dot`) and the linear combination of vectors (`axpy`).

Relying on a minimal set of algebraic kernels enables code portability and facilitates its maintenance and optimisation. However, it comes together with two types of challenges and restrictions. Firstly, *computational challenges* like the low arithmetic intensity of the `SpMV`, which can be alleviated by using the more compute-intensive sparse matrix-matrix product (`SpMM`). This is possible in a great variety of situations, such as with multiple transport equations, in cases with spatial reflection symmetries, parallel-in-time simulations and, in general, whenever dealing with matrices, $\hat{\mathsf{A}} \in \mathbb{R}^{N\times N}$, decomposable as the Kronecker product of a diagonal matrix, $\mathsf{C} \equiv diag(\boldsymbol{c}) \in \mathbb{R}^{K\times K}$, and a sparse matrix, $\mathsf{A} \in \mathbb{R}^{N/K\times N/K}$, *i.e.*, $\hat{\mathsf{A}} = \mathsf{C} \otimes \mathsf{A}$. Indeed, under such circumstances, the standard `SpMV` can be replaced with the `SpMM`:

$$\boldsymbol{y} = \hat{\mathsf{A}}\boldsymbol{x} \quad \Longrightarrow \quad (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K) = \mathsf{A}\,(c_1\boldsymbol{x}_1, \ldots, c_K\boldsymbol{x}_K)\,, \tag{1}$$

where $\boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^{N/K}$. By doing so, matrix coefficients are recycled, thus significantly reducing the memory accesses and the memory footprint of the operators.

Secondly, *algorithmic challenges* such as reformulating classical flux limiters [10] or the boundary conditions must also be addressed. The latter can be naturally solved by casting boundary conditions into an affine transformation:

$$\boldsymbol{\varphi}_h \rightarrow \mathsf{A}\boldsymbol{\varphi}_h + \boldsymbol{b}_h, \tag{2}$$

allowing a purely algebraic treatment of virtually all existing boundary conditions both for explicit and implicit time-integration methods. Furthermore, an accurate and portable approach solely relying on the above-mentioned algebraic kernels for bounding the eigenvalues of the convective and diffusive operators has also been proposed.

3. Performance analysis and application to CSP technologies

The performance analysis of the code will be done in two stages. Firstly, we will focus on the performance analysis for a given mesh without considering the results' accuracy. The comparison will be made with the open-source CFD code OpenFOAM. Several factors will be analysed separately and compared, such as exploiting the shared-memory paradigm with OpenMP, increasing the arithmetic intensity through the strategies mentioned above, and the effect of GPUs. In the second stage, a relevant case from the CSP industry (see figure 1) will be studied as a demonstrative test case to assess the feasibility of overnight industrial simulations.

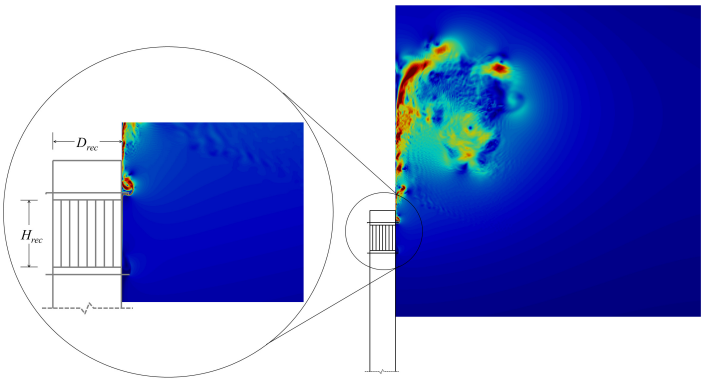


Figure 1. CSP simulation of a central tower receiver.

All the numerical experiments were conducted on the Snellius supercomputer at SURF. The CPU executions were run on nodes equipped with two AMD Rome 7H12 (64 cores, 2.6 GHz, 256 MB L3 cache and 204.8 GB/s memory bandwidth) linked to 256 GB of RAM and interconnected through HDR100 ConnectX-6. On the other hand, GPU executions were run on nodes equipped with two Intel Xeon Platinum 8360Y (36 cores, 2.4 GHz, 54 MB L3 cache and 204.8 GB/s memory bandwidth), accelerated with four NVIDIA A100 (9.7 TFLOPS with FP64, 40GB HBM2e and 1.935 GB/s memory bandwidth), linked to 512 GB of RAM, and interconnected through two HDR100 ConnectX-6, two 25GbE SFP28 LOM and one 1GbE RJ45 LOM, adding up to 251Gb/s.

The domain considered is analogous to figure 2. As discussed in [11], we only discretised a fraction of it in accordance with the number of symmetries being exploited. Namely, a half, a quarter and an eighth of the domain when exploiting $s = 1, 2, 3$ symmetries, respectively. Then, given the mesh resolution demands of high-fidelity simulations of the CSP case studied, the discretisation considered had 512 million grid points (see [12, 13]).

To evaluate the parallel efficiency of TFA and compare it to that of OpenFOAM, we benchmarked OpenFOAM's PISO against TFA's fully-explicit fractional step method. Given

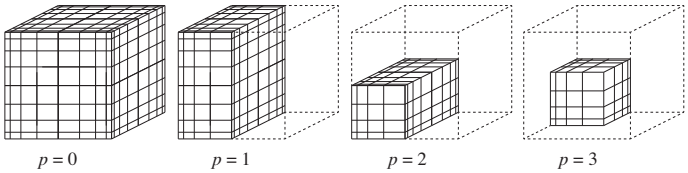
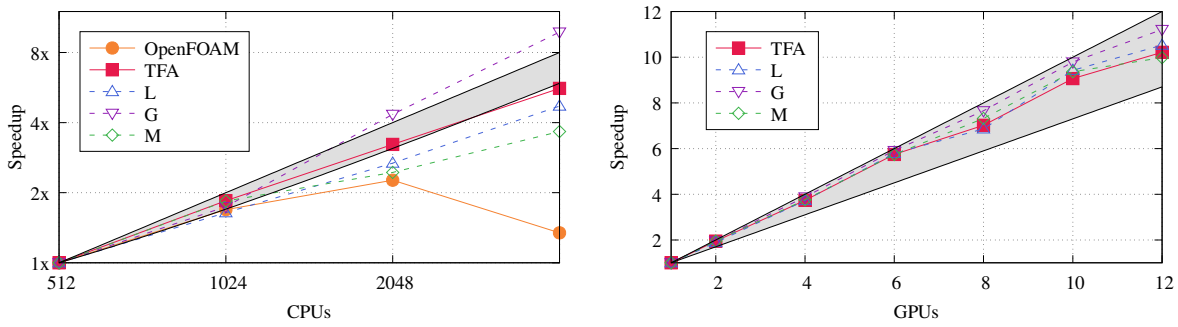


Figure 2. Example of portions meshed when exploiting s reflection symmetries.

that both algorithms cannot be directly compared, we limited these preliminary results to analysing how both codes scale. Figure 3a presents their strong scalability on CPUs, including, for TFA, the scalability of applying the discrete Laplacian, gradient and divergence operators. As expected, OpenFOAM's MPI-only parallelisation (which assigns an MPI process per CPU core) resulted in larger communication overheads and, consequently, poorer parallel efficiencies. Conversely, TFA's multithreaded parallelism resulted in significantly higher efficiencies as the number of processors increased. As explained earlier, TFA's adoption of an "algebraic approach" grants easy portability to massively parallel accelerators. Figure 3b displays its scalability on GPU-accelerated nodes.



(a) 512M mesh on AMD EPYC 7H12 **(b)** 64M mesh on NVIDIA A100

Figure 3. Strong scalability analysis with 70% to 100% parallel efficiency range in light grey.

In order to assess the feasibility of overnight LES simulations, the CSP case of figure 1 has been studied. More concretely, we recalled the excellent weak scaling of TFA, as demonstrated in [9] by the almost perfect weak scalability of `SpMV`. Then, by picking from figure 3 the CPU work loads of 1024 and 4096 CPUs, which correspond to 95% and 65% parallel efficiencies, we could approximate the maximum simulation sizes that TFA could afford in overnight simulations, *i.e.*, taking at most 16 wall-clock hours. This was very convenient given the limited availability of computing resources, as it allowed us to obtain low-cost approximations of the positive impact of exploiting symmetries and leveraging GPUs.

On the one hand, we can infer from figure 3 the time a time-step takes, $T_{\Delta t}^{\text{eff}}$, at a given parallel efficiency and on a mesh of size N_{ref}. On the other hand, the number of time-steps required to simulate τ time units can be approximated as follows:

$$n_{\Delta t} = \frac{\tau}{\Delta t}. \tag{3}$$

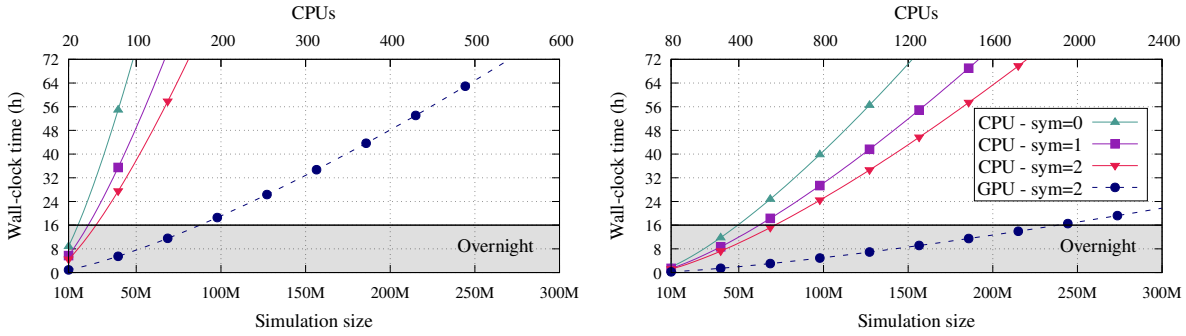
Then, recalling that LES are generally convection-dominated, its time-step can be approximated as follows:

$$\Delta t = \min\left\{\frac{\Delta x_i}{|u_i|}\right\} \simeq \frac{c}{\sqrt[3]{N}}, \tag{4}$$

where Δx, u and N stand for the cell length, local velocity and mesh size, respectively. Then, from equation (4), Δt is (approximately) inversely proportional to $\sqrt[3]{N}$ and, as shown in [14], the correction constant c typically takes values around 0.3. Therefore, the wall-clock time of a simulation of size N can be approximated as:

$$T_{\text{LES}}(N) \simeq n_{\Delta t}\frac{T^{\text{eff}}_{\Delta t}N}{N_{\text{ref}}} = \frac{\tau T^{\text{eff}}_{\Delta t}}{cN_{\text{ref}}}\sqrt[3]{N^4}. \tag{5}$$

When it comes to τ, it is also shown in [14] that after around 100 time-units the flow starts to become statistically stationary. Therefore, figure 4 considers the wall-clock time for simulating $\tau = 150$ time units.



(a) 95% efficiency **(b)** 65% efficiency

Figure 4. Estimation of largest affordable overnight simulations in up to 16 wall-clock time hours. Results for several efficiencies according to figure 3.

As expected, increasing the number of CPUs allows for larger overnight simulations, but this comes at the cost of lower efficiencies. According to figure 4b, restricting ourselves to 65% efficiencies allows for around 50M overnight LES simulations. Then, thanks to exploiting up to two symmetries, 75M simulations become affordable. Nonetheless, as shown in [12, 13], high-fidelity simulations of CSP applications would easily require 300 to 500 million-sized grids. As a result, leveraging massively-parallel accelerators is imperative. In this sense, one of the critical advantages of TFA is its modular design, which provides support for both CUDA and OpenCL, therefore covering virtually all GPU vendors. Given our limited access to the GPU nodes, the CPU tests could not be extended to GPUs. As a result, we were forced to approximate its performance by applying a conservative speed-up factor based on past results, see [9]. More concretely, by assuming a 5x GPU speed-up, figure 4b illustrates how up to 250M overnight simulations can be tackled.

4. Conclusions

This conference paper presents a symmetry-preserving discretisation for unstructured collocated grids that is virtually free of artificial dissipation and unconditionally stable. The proposed discretisation relies on a minimal set of algebraic kernels to ensure cross-platform portability, making it suitable for complex geometries and modern computational architectures.

The paper addresses several challenges related to the algebraic approach, such as the low arithmetic intensity of the sparse matrix-vector product, reformulating boundary conditions and flux limiters, and efficiently computing eigenbounds to determine the time-step.

The advantages and disadvantages of the proposed approach are analysed by comparing our in-house code, TFA, with OpenFOAM. The performance analysis demonstrates that the algebraic approach in TFA scales better thanks to its hybrid MPI+OpenMP parallelisation, leading to higher efficiencies as the number of processors increases.

Furthermore, this paper evaluates the feasibility of overnight industrial LES simulations. With this aim, we recall a relevant case from the CSP industry to estimate the largest affordable overnight simulations. As it is shown, leveraging massively-parallel accelerators such as GPUs becomes crucial for meeting the industry demands, and we estimate that TFA could tackle up to 250M overnight simulations.

Overall, this work shows promising results for enabling reliable DNS and LES simulations of turbulent flows in industrial applications with robust and stable numerical methods on modern high-performance computing systems. It demonstrates the potential of the algebraic approach to achieve performance portability while proposing strategies to increase the arithmetic intensity of the simulations.

Regarding future work, we aim to extend the scalability tests both on CPUs and GPUs. Additionally, we plan to enrich the comparison between TFA and OpenFOAM by studying their cost vs accuracy.

References

[1] Witherden F D, Farrington A M and Vincent P E 2014 *Comput. Phys. Commun.* **185** 3028–3040
[2] Carter Edwards H, Trott C R and Sunderland D 2014 *J. Parallel Distrib. Comput.* **74** 3202–3216
[3] Beckingsale D A, Burmark J, Hornung R, Jones H, Killian W, Kunen A J, Pearce O, Robinson P, Ryujin B S and Scogland T R 2019 *IEEE/ACM Int. Workshop on Performance, Portability and Productivity in HPC* (Denver: IEEE) pp 71–81
[4] Bell N and Garland M 2009 *Proc. of the Conf. on High Performance Computing Networking, Storage and Analysis* (Portland: ACM) pp 1–11
[5] Greathouse J L, Knox K, Poła J, Varaganti K and Daga M 2016 *Proc. of the 4th Int. Workshop on OpenCL* (New York: ACM)
[6] Trias F X, O Lehmkuhl, A Oliva, CD Pérez-Segarra and R W C P Verstappen 2014 *J. Comput. Phys.* **258**(1) 246–267
[7] Komen E, Hopman J A, Frederix E M A, Trias F X and Verstappen R W C P 2021 *Comput. Fluids* **225** 104979
[8] Löhner R, Othmer C, Mrosek M, Figueroa A and Degro A 2021 *Comput. Fluids* **214** 104771
[9] Álvarez Farré X, Gorobets A, Trias F X, Borrell R and Oyarzun G 2018 *Comput. Fluids* **173** 285–292
[10] Valle N, Álvarez-Farré X, Gorobets A, Castro J, Oliva A and Trias F X 2022 *Comput. Phys. Commun.* **271** 108230
[11] Alsalti-Baldellou A, Álvarez-Farré X, Trias F X and Oliva A 2023 *J. Comput. Phys.* **486** 112133
[12] David M, Toutant A and Bataille F 2021 *Phys. Fluids* **33** 045104
[13] David M, Toutant A and Bataille F 2023 *Phys. Fluids* **35** 035106
[14] Trias F, Gorobets A, Soria M and Oliva A 2010 *Int. J. Heat Mass Transf.* **53** 665–673