Ensemble averaging parallel-in-time approach for industrial LES

Josep Plana-Riu [1], F Xavier Trias [1], Àdel Alsalti-Baldellou [1,2], Guillem Colomer [1] and Asensio Oliva [1]

[1]Heat and Mass Transfer Technological Centre
Technical University of Catalonia, ESEIAAT
Carrer de Colom 11, 08222 Terrassa (Barcelona), Spain
[2] Termo Fluids S.L.
Carrer de Magí Colet 8, 08204 Sabadell (Barcelona), Spain

E-mail: josep.plana.riu@upc.edu

Abstract. Computational Fluid Dynamics for industrial applications usually relies on RANS modeling instead of LES, as the latter is more expensive and requires much longer simulation times. In order to reduce the computational cost of the simulation, an ensemble averaging parallel-in-time approach is presented so that the simulation time of LES simulations can be reduced up to overnight lengths by exploting the benefits of using sparse matrix-matrix products, `SpMM`, instead of the classic sparse matrix-vector products, `SpMV`, or `SpMV`-like stencil-based kernels, as it might lead to a speed-up given the proper conditions. These conditions are tested for a classical differentially heated cavity benchmark, to apply later on to the simulation of a CSP collector, as it is a well-known academic case that resembles the final geometry.

1. Introduction

When Computational Fluid Dynamics (CFD) is applied in real-world situations, the user expects robust and stable methods, as failure would lead to increased costs, and a rather short completion time. Combining these parameters is why RANS modeling is the most common technique for industrial applications. However, RANS has some remarkable limitations which could be solved by using Large Eddy Simulations (LES), even though the latter has much higher computational costs.

Current CFD simulations are usually memory-bound [1]. This means that the performance of the simulations is bounded by the maximum data transfer rate, instead of the number of flops. Hence, one way of improving the performance would be reducing the amount of data to be transferred. This concept is measured with the so-called arithmetic intensity (AI), which is defined as the ratio between the number of floating point operations and the data to be transferred.

In order to improve the AI, Krasnopolsky [2] generalized the computation of sparse matrix-vector products (`SpMV`) within the solution to the Poisson equation, to consider a greater number of right-hand-side (RHS) vectors, leading to what was called generalized sparse matrix-vector product (`GSpMV`), as seen in figure 1, for the case of two RHS.

This generalization implied that, if all RHS vectors were sent together as a dense matrix, the number of communications required was reduced, leading to an increase of the AI. With this

$$\begin{pmatrix} A & \\ & A \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} \qquad\qquad A(\mathbf{u}_1 \;\; \mathbf{u}_2)$$

Figure 1. Left: structure in which two `SpMV` are done "simultaneouslly" as two different products. Right: structure in which these two `SpMV` are performed with only one call of matrix A, in a `GSpMV` or `SpMM`. In this way, the left setup performs two products with 2 calls of A, while for the right setup, even though two operations are performed, A is only called once, which increases the arithmetic intensity of the operation.

concept, an ensemble averaging approach was considered so that, instead of running a single long simulation for e.g. a channel flow, multiple shorter simulations were ran simultaneously, to ensemble average the results of those different runs. In this study, Krasnopolsky found that given the proper conditions of weight of the Poisson equation solution in the total iteration time, as well as the ratio between the averaging and transition times, there was some significant speed-up compared to the single simulation case.

Very recently, Alsalti-Baldellou et al. [3] introduced a similar concept in which symmetries in the domain were exploited so that the `SpMV` could be transformed into, as it was called, sparse matrix-matrix products, `SpMM`. By doing so, as Krasnopolsky, the AI was increased notably, leading to speed-up even with the same number of compute units. Opposite to Krasnopolsky, Alsalti-Baldellou et al. [3] applied this technique to all `SpMV`-like operations, thus leading to bigger speed-ups than the original work. Moreover, this method was developed within the HPC^2 framework [4], in which its algebra-based approach allows the code to be portable to both CPU- and GPU-accelerated nodes, making it flexible for modern supercomputers. Thus, it could be applied to speed up CFD simulations to fulfill the goal of overnight LES, defined as a fully-resolved LES simulation with a maximum wall clock time of 16 hours, given several symmetries exploited.

In this conference paper, the latter approach from Alsalti-Baldellou et al. [3] will be applied in the ensemble averaging parallel-in-time method from Krasnopolsky in a differentially heated cavity as well as in a model of a CSP tower to test the possibility of performing industrial LES simulations.

2. Methodology

The numerical solution to the incompressible Navier-Stokes equations is done with the following semi-discrete set of equations, using the notation from Trias et al. [5],

$$M\mathbf{u}_s = \mathbf{0}_c, \tag{1a}$$

$$\Omega\frac{d\mathbf{u}_c}{dt} + C(\mathbf{u}_s)\mathbf{u}_c - \frac{1}{\mathrm{Re}}D\mathbf{u}_c + \Omega G_c\mathbf{p}_c = \mathbf{0}_c, \tag{1b}$$

where M is the face-to-cell divergence operator, Ω_c is a diagonal matrix containing the cell volumes so that $\Omega = I_3 \otimes \Omega_c$; C_c is the cell-to-cell convective operator, so that $C = I_3 \otimes C_c$; D_c is the cell-to-cell diffusive operator, so that $D = I_3 \otimes D_c$, G_c is the cell-to-cell gradient operator, $\mathbf{u}_s$ is the velocity field defined at the faces, and I_3 is the identity matrix of size 3. All the operators are built to preserve the symmetries from the continuous operators, according to Trias et al. [5]. In order to integrate the set of equations in time, the standard fourth-order Runge-Kutta scheme (RK4) will be used within the framework of the projection method of Sanderse and Koren [6], given its improved stability properties compared to his lower-order counterparts.

This set of equations will be solved, thus, simultaneouslly for m simulations of identical geometry and conditions in a parallel-in-time approach. By doing this, all simulations will be

run on the same device, which will replace automatically the SpMV operations by their SpMM counterparts for all operators as well as products in the Poisson equation solution. This will be implemented within the framework of the in-house code TermoFluids Algebraic (TFA), which allows performing these SpMM operations, as it is built around HPC^2.

As detailed by Krasnopolsky [2], starting from a simulation with $m = 1$, i.e. the usual way of running a simulation, the time evolution of the run can be split into a transition time T_T and an averaging time T_A, so that the total simulation time T_1, with 1 referring to $m = 1$, can be defined as,

$$T_1 = T_T + T_A. \tag{2}$$

From this point, a parallel-in-time simulation consisting of m simulations will have a total simulation time of,

$$T_m = T_T + \frac{T_A}{m}, \tag{3}$$

so that the transition time is preserved, as it is required for the development of the turbulence and to obtain statistically independent seeds, yet the averaging time will be evenly split among the different cases. From the definition of these magnitudes, the ratio between the averaging and transition time can be defined as $\beta = T_A/T_T$.

In single-simulation setups, T_T would correspond to the time that the flow, from the initial state takes to fully develop into a turbulent flow. However, in this parallel-in-time setup, the transition time will need to account for the time that the different simultaneous simulations take to be statistically independent.

3. Performance improvements with SpMM

According to Alsalti-Baldellou et al. [3], the AI of a SpMM with d RHS, with the sparse matrix $A \in \mathbb{R}^{m\times n}$, consisting of $nnz(A)$ non-zero entries, can be expressed as

$$\mathrm{AI}_{\texttt{SpMM}}(A, d) = \frac{(2nnz(A) + 1)d}{12nnz(A) + 4(m + 1) + 8(m + n + 1)d}, \tag{4}$$

so that, for instance, the arithmetic intensity of a SpMV can be stated as $\mathrm{AI}_{\texttt{SpMV}}(A) = \mathrm{AI}_{\texttt{SpMM}}(A, 1)$. Hence, the speed-up due to using a single SpMM instead of multiple SpMV finds its upper bound as the ratio of both variables,

$$P_{\texttt{SpMM},ub} = \frac{\mathrm{AI}_{\texttt{SpMM}}(d)}{\mathrm{AI}_{\texttt{SpMV}}}, \tag{5}$$

while the lower bound, $P_{\texttt{SpMM},lb}$, is obtained with the same ratio, yet computing the AI by replacing $nnz(A)$ by n in Eq.(4). For a rather dense case, where $nnz(A)/m = 17$, and $n = d$, the representation of the bounds can be found in figure 2.

According to Krasnopolsky, the speed-up of m simultaneous runs in a parallel-in-time simulation, P_m can be estimated with

$$P_m = m\frac{1 + \beta}{m + \beta}\frac{t_1}{t_m}, \tag{6}$$

where t_m is the wall clock time per iteration in a simulation with m RHS. From Eq. (6), it can be split into two different parts. First of all, the term $\dfrac{1 + \beta}{m + \beta}$ only depends on parameters that are user inputs and thus can be stated to be case-dependent, and does not depend at all

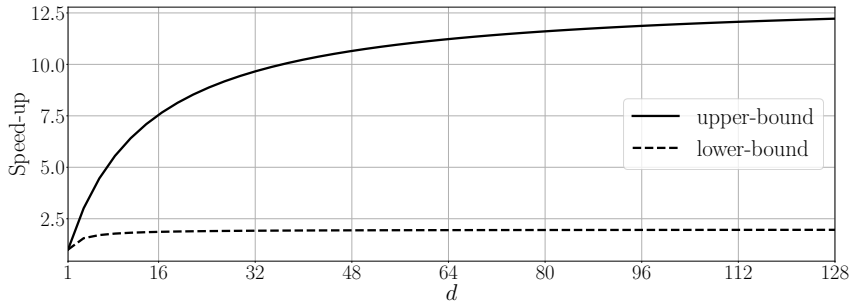

Figure 2. Theoretical speed-up bounds for a sparse matrix A with $nnz(A)/m = 17$.

on the architecture or code used. On the contrary, the second term $\frac{mt_1}{t_m}$ is implementation dependent, as it will depend both on the code as well as on m.

According to the implementation for a Runge-Kutta integration with s stages, the wall clock time per iteration, with m RHS can be stated as

$$t_m(s) = s\left[t_{\text{Poisson}}(m) + 33t_{\texttt{SpMM}}(m) + m\left[10t_{\texttt{axty}} + \left(24 + 3\frac{s-1}{2}\right)t_{\texttt{axpy}}\right]\right], \tag{7}$$

so that, if γ is the ratio between t_m and $t_{\texttt{SpMM}}$, and θ is the SpMM fraction within the Poisson solver, the ratio $\frac{mt_1}{t_m}$ can be rewritten as

$$\frac{s(\theta+33)}{\gamma}(P_{\texttt{SpMM}} - 1) + 1. \tag{8}$$

Introducing Eq. (8) into Eq. (6), the expected speed-up for the parallel-in-time simulation will be

$$P_m = \frac{1+\beta}{m+\beta}\left[\frac{s(\theta+33)}{\gamma}(P_{\texttt{SpMM}} - 1) + 1\right]. \tag{9}$$

4. Numerical results

In order to test the methodology, the case has been run in a differentially heated cavity with an aspect ratio of 4, following the work from Trias et al. [7] for both laminar and turbulent cases. In the case of this heat transfer problem, the governing equations have to be rewritten as

$$M\mathbf{u}_s = \mathbf{0}_c, \tag{10a}$$

$$\Omega\frac{d\mathbf{u}_c}{dt} + C(\mathbf{u}_s)\mathbf{u}_c - \frac{\text{Pr}}{\text{Ra}^{1/2}}D\mathbf{u}_c + \Omega G_c\mathbf{p_c} + \Omega\mathbf{f}_c = \mathbf{0}_c, \tag{10b}$$

$$\Omega\frac{d\boldsymbol{\theta}_c}{dt} + C(\mathbf{u}_s)\boldsymbol{\theta}_c - \frac{1}{\text{Ra}^{1/2}}D\boldsymbol{\theta}_c = \mathbf{0}, \tag{10c}$$

where Pr is the Prandtl number, Ra is the Rayleigh number, θ is the dimensionless temperature, $\theta = (T - (T_H + T_C)/2)(T_H - T_C)$, being T_H the temperature of the hot wall, and T_C the temperature of the cold wall, and $\mathbf{f}_c$ the Boussinesq approximation, which is non-zero in the direction of gravity and takes a value of $\mathbf{f_c} \cdot \mathbf{g}/g = \text{Pr}\theta$.

Hence, a domain of size $1.0 \times 4.0 \times 1.0$ has been discretized using a $256 \times 256 \times 64$ mesh, with Ra=2×10^9 and Pr=0.71, being y the vertical direction. The grid is periodic in the z direction to

model an infinite domain so that the flow becomes statistically homogeneous in this direction. The cavity will be then subjected to a hot wall, at temperature T_H placed at $x = 0$; and a cold wall, at T_C placed at $x = 1$. Hence, applying the dimensionless temperature definition, $\theta(0, y, z) = 0.5$, and $\theta(1, y, z) = -0.5$. All the walls will be considered as stationary and non-slip. With regards to the timestep, it will be adapted with the maximum stable timestep from its stability region, adapting the works from Trias and Lehmkuhl [8] to a standard fourth-order explicit Runge-Kutta integration.

For the baseline case of 1 RHS, the simulation is run for 100000 iterations, as the time averaging starts after 10000, which leads to $\beta = 9$. By doing so, the obtained fields (u, v, θ) are presented in figure 3. The simulation was run in 1 JFF fourth-generation compute node (2x Intel Xeon 6230, 38 computing CPUs + 2 halo update CPUs), in a time per iteration of 2.539 s.

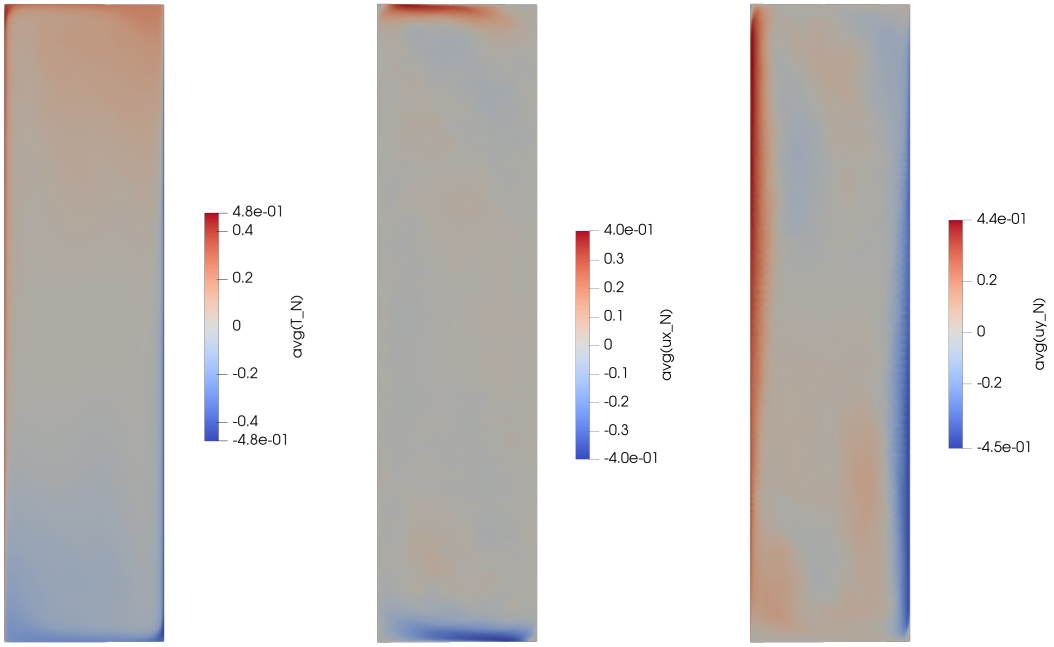

Figure 3. Average fields after 100000 iterations of dimensionless temperature (left), horizontal velocity (centre), and vertical velocity (right), for a differentially heated cavity of aspect ratio 4, full of air, with $\mathrm{Ra} = 2 \times 10^9$, simulated with 1 RHS.

On the other hand, the same case has been run with 2 RHS, for a total of 55000 iterations. In this case, the time per iteration is 5.807 s, which does not lead to eventual speed-up at the end of the simulation. For 4 RHS, and a total of 33500 iterations, the time per iteration obtained was of 14.840 s, which is still worse than the 2 RHS case. The results are summarized in Tab. 1. This lack of speed-up could be explained by the fact that the original 1 RHS case fits in the cache memory of the system used to run the case and, thus, it is not more efficient to run a shorter, and more compute-intense case, than a longer, less compute-intense case.

5. Conclusion

In this conference paper, the methodology used to extend the work from Krasnopolsky [2], in which he generalized the `SpMV` kernel to a `SpMM` for the solution of the Poisson equation; to the rest of the operators (gradient, divergence) of the whole simulation so that the improved AI of the operation can be completely exploited, as in the work from Alsalti-Baldellou et al. [3]. This

Table 1. Performance figures obtained after the test runs for 1, 2, and 4 RHS in a differentially heated cavity filled with air, with $\mathrm{Ra} = 2 \times 10^9$.

Number of RHS	Time per iteration [s]	Total wall clock time [s]	Speed-up
1	2.539	253900	1.0
2	5.807	319385	0.795
4	14.840	497130	0.511

inherent speed-up to the simulations just by simulating multiple cases simultaneously might as well be exploited to reduce the wall clock time required in LES simulations so that they run in 16 hours. This would thus increase the accuracy of the industrial simulations, which could benefit the design process while keeping a reasonable wall clock time.

The method has been tested for a differentially heated cavity filled with air with $\mathrm{Ra} = 2 \times 10^9$, and $\mathrm{Pr} = 0.71$, as it requires using the methodology not only for the classical mass and momentum equations but also extending it to the energy equation. This is relevant as the case will be applied to a CSP collector characterized by a similar geometry.

These numerical experiments show that, for meshes that are not big enough, i.e. $256 \times 256 \times 64$ for one JFF fourth-generation compute node, there is no speed-up as the operators fit in cache memory, which has a faster access. Thus, it is more efficient to run a longer and lighter simulation instead of a short, more compute-intense simulation. This will also be applied for a differentially heated cavity case with a larger mesh, more suitable for DNS or LES simulations, in order to check the existence of speed-up or not. Hence, the load per CPU should be increased from the current 110k control volumes per CPU, since this mesh will most likely fit in the cache memory.

Acknowledgements

The investigations presented in this abstract are supported by the *Ministerio de Economía y Competitividad*, Spain, SIMEX project (PID2022-142174OB-I00). J. P-R. is also supported by the Catalan Agency for Management of University and Research Grants (AGAUR). The authors thankfully acknowledge these institutions.

References

[1] Williams S *et al* 2009 *Commun. ACM* **52** 65-76
[2] Krasnopolsky B 2018 *Comp. Phys. Comm.* **229** 8-19
[3] Alsalti-Baldellou A *et al* 2023 *J. Comp. Phys.* **486** 112133
[4] Álvarez X *et al* 2018 *Comp. and Fl.* **173** 285-92
[5] Trias F X *et al* 2014 *J. Comp. Phys.* **258** 246-67
[6] Sanderse B and Koren B 2012 *J. Comp. Phys.* **231** 3041-63
[7] Trias F X *et al* 2010 *Int. J. Heat Mass Transfer* **53** 665-73
[8] Trias F X and Lehmkuhl O 2011 *Numer. Heat Transfer* B **60** 116-34