ICCFD13-2026-072

Towards robust filter–width formulations for architecture independent Large–Eddy Simulation: flow past a circular cylinder at Re = 3900

Alessio Piccolo*, Simone Bnà**, Valerio D'Alessandro***
Corresponding author: a.piccolo@cineca.it
* CINECA, Direzione HPC, Piazza dell'Indipendenza 11/B, 00185, Rome (Italy)
** CINECA, Direzione HPC, Via Magnanelli 2, 40033, Casalecchio di Reno (Italy)
*** Dipartimento di Ingegneria Industriale e Scienze Matematiche, Università Politecnica delle Marche, Via Brecce Bianche 12, 60131, Ancona (Italy)

Abstract: Large–Eddy Simulation (LES) is increasingly performed on GPU–based supercomputers, where massively parallel execution introduces numerical effects that are negligible on traditional CPU architectures. In particular, non–deterministic floating–point perturbations generated by parallel accumulation operations may interact with subgrid–scale (SGS) models and affect the predicted flow field. Specifically, certain filter-width definitions, although mathematically equivalent in exact arithmetic, exhibit a pronounced sensitivity to parallel reductions on GPUs, leading to non-physical behaviors that are not observed on CPUs.
In this work, we investigate the role of flow-dependent filter–width formulations and implementation strategies in controlling the effects of floating–point non–associativity on massively parallel architectures. The analysis is carried out for the canonical flow past a circular cylinder at Re = 3900, a benchmark particularly sensitive to the aforementioned effects. The results demonstrate that the discrepancies observed across heterogeneous computing architectures originate from the interplay between implementation strategy, floating-point accumulation mechanisms, and the mathematical structure of LES filter-width operators. While implementation strategies determine the magnitude of the perturbations entering the SGS model, the resulting flow response is governed by the sensitivity of the filter-width operator to small perturbations in the underlying spatial metrics. Numerical robustness, implementation strategy, and SGS modeling should therefore be considered together when developing LES methodologies for next-generation HPC systems.

Keywords: Large–Eddy Simulation, Circular cylinder, Dynamic Smagorinsky, GPU, Hardware portability.

1. Introduction

The increasing adoption of GPU architectures for Large-Eddy Simulation (LES) is exposing new challenges associated with floating-point arithmetic. In massively parallel environments, the inherently non–associative nature of reduction operations can generate accumulation errors that, while negligible in sequential or weakly parallel CPU implementations, may become significant at scale. While the CFD community has extensively investigated discretization and modeling errors [1, 2], considerably less attention has been devoted to understanding how floating–point perturbations interact with Subgrid–Scale (SGS) closures. Existing studies highlight the importance of the interplay between discretization and turbulence modeling, but generally assume deterministic numerical operators. As a consequence, the impact of floating–point accumulation errors on SGS modeling and flow predictions remains an open research question.
Flow–dependent filter–width definitions constitute a particularly relevant example of the interaction between floating-point perturbations and SGS modeling. These formulations rely on local flow and grid metrics and are therefore potentially sensitive to perturbations arising during their evaluation. Among the

available approaches, the least-squares-based (LSQ) characteristic length proposed by Trias et al. [3] has attracted considerable attention, as it accounts for both grid anisotropy and local flow topology through the velocity-gradient tensor. The LSQ formulation was originally developed and assessed for incompressible flows on anisotropic grids, where it was shown to significantly improve LES predictions. More recently, D'Alessandro et al. [4] successfully extended the approach to compressible LES, demonstrating that the additional computational cost associated with the evaluation of the LSQ filter width can be largely compensated by the reduced grid resolution requirements. However, the behavior of flow-dependent filter-width formulations in the presence of non-deterministic floating-point accumulations generated by massively parallel architectures remains largely unexplored. In particular, flow-dependent filter-width formulations may be intrinsically sensitive to small perturbations arising during the evaluation of the underlying spatial metrics, allowing floating-point errors to propagate through the SGS closure and ultimately affect the resolved flow field.

We have observed that the standard LSQ filter-width formulation, although mathematically equivalent in exact arithmetic, may exhibit a significant sensitivity to perturbations affecting the evaluation of the underlying spatial metrics. On massively parallel GPU architectures, such perturbations can originate from non-deterministic floating-point accumulations and may propagate through the SGS closure, ultimately influencing the resolved flow fields. The numerical analysis community has extensively studied these effects, particularly in the context of reproducibility and accuracy of floating-point computations [5, 6]. However, their impact on turbulence modeling and LES closures has received limited attention. In this work, we investigate mathematically reformulated filter-width definitions aimed at improving the robustness of LES closures in massively parallel environments. Particular attention is devoted to both filter-width formulations and implementation strategies. The latter control the level of numerical perturbations introduced during the evaluation of the underlying spatial metrics, whereas the former determine the sensitivity of the SGS closure to such perturbations and ultimately their influence on the resolved flow field.

The proposed developments are implemented within SPUMA [7], a minimally invasive GPU-oriented framework for OpenFOAM. The framework enables the execution of LES solvers on modern accelerator-based HPC systems while preserving the original algorithmic structure of the underlying OpenFOAM implementation. The analysis is carried out on the canonical flow past a circular cylinder at $\mathrm{Re} = 3900$, [8, 9], a benchmark particularly sensitive to numerical and modeling errors.

The paper is organized as follows. Sec. 2 presents the governing equations, SGS modeling, filter–width formulations, and numerical approximation. Sec. 3 is devoted to the implementation strategies adopted for the evaluation of filter–widths. Sec. 4 discusses the numerical results, while Sec. 5 summarizes the main conclusions.

2. Computational framework

2.1 Governing equations

Within the LES framework, the governing equations for low-Mach-number compressible flows are obtained by Favre filtering the Navier–Stokes equations and can be written in compact vector form as

$$\frac{\partial \bar{\mathbf{u}}}{\partial t} + \boldsymbol{\nabla} \cdot \mathbf{F}_c(\bar{\mathbf{u}}) - \boldsymbol{\nabla} \cdot \mathbf{F}_v(\bar{\mathbf{u}}, \boldsymbol{\nabla}\bar{\mathbf{u}}) + \boldsymbol{\nabla} \cdot \mathbf{F}_{SGS}(\bar{\mathbf{u}}, \boldsymbol{\nabla}\bar{\mathbf{u}}) = \mathbf{0} \tag{1}$$

$$\mathbf{F}_{\mathrm{c,j}} = \begin{pmatrix} \bar{\rho}\tilde{u}_j \\ \bar{\rho}\tilde{u}_1\tilde{u}_j + \bar{p}\delta_{1j} \\ \bar{\rho}\tilde{u}_2\tilde{u}_j + \bar{p}\delta_{2j} \\ \bar{\rho}\tilde{u}_3\tilde{u}_j + \bar{p}\delta_{3j} \\ \bar{\rho}\tilde{u}_j\tilde{H} \end{pmatrix}, \qquad \mathbf{F}_{\mathrm{v,j}} = \begin{pmatrix} 0 \\ \bar{\sigma}_{1j} \\ \bar{\sigma}_{2j} \\ \bar{\sigma}_{3j} \\ \bar{\sigma}_{ji}\tilde{u}_i - \tilde{q}_j \end{pmatrix}, \qquad \mathbf{F}_{\mathrm{SGS,j}} = \begin{pmatrix} 0 \\ \tau_{1j} \\ \tau_{2j} \\ \tau_{3j} \\ \phi \end{pmatrix}. \tag{2}$$

where $\bar{\mathbf{u}} = \left(\bar{\rho}, \bar{\rho}\tilde{u}_i, \bar{\rho}\tilde{E}\right)^T$ is the vector of Favre-filtered conservative variables, while $\mathbf{F}_c\left(\bar{\mathbf{u}}\right)$, $\mathbf{F}_v\left(\bar{\mathbf{u}}, \boldsymbol{\nabla}\bar{\mathbf{u}}\right)$ and $\mathbf{F}_{SGS}\left(\bar{\mathbf{u}}, \boldsymbol{\nabla}\bar{\mathbf{u}}\right)$ denote the convective, viscous and subgrid–scale flux tensors, respectively, whose j–th components are given by eq. 2. The viscous stress tensor and heat flux vector are defined according to the constitutive relation for Newtonian fluids and Fourier's law:

$$\tilde{\sigma}_{ij} = 2\mu\left(\tilde{S}_{ij} - \frac{1}{3}\frac{\partial\tilde{u}_k}{\partial x_k}\delta_{ij}\right), \quad \tilde{q}_j = -\lambda\frac{\partial\tilde{T}}{\partial x_j}. \tag{3}$$

In the above equations, μ is the molecular viscosity and $\tilde{S}_{ij} = \frac{1}{2}\left(\frac{\partial\tilde{u}_i}{\partial x_j} + \frac{\partial\tilde{u}_j}{\partial x_i}\right)$ is the Favre–filtered strain-rate tensor. The thermal conductivity, λ, is obtained from the Prandtl number, $\mathrm{Pr} = \mu c_p/\lambda$, where c_p denotes the specific heat at constant pressure. The pressure is computed from the ideal–gas equation of state, while the temperature is evaluated from the total internal energy according to:

$$\bar{p} = \bar{\rho}\left(\gamma - 1\right)\left(\tilde{E} - \frac{1}{2}\tilde{u}_k\tilde{u}_k\right), \quad c_v\tilde{T} = \tilde{E} - \frac{1}{2}\tilde{u}_k\tilde{u}_k\ , \tag{4}$$

where γ is the heat capacity ratio, and c_v is the specific heat at constant volume. The anisotropic part of the SGS stresses is parametrized using the eddy-viscosity hypothesis:

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = -2\mu_{SGS}\left(\tilde{S}_{ij} - \frac{1}{3}\frac{\partial\tilde{u}_k}{\partial x_k}\delta_{ij}\right) \tag{5}$$

where SGS viscosity is evaluated as

$$\mu_{SGS} = \bar{\rho}C_s^2\Delta^2\left|\tilde{S}\right|\ . \tag{6}$$

Here, $\left|\tilde{S}\right|$ denotes the magnitude of the Favre-filtered strain-rate tensor and Δ is the spatial filter width. The Smagorinsky coefficient, C_s^2 is dynamically evaluated using the Germano–Lilly procedure, [10]. The SGS contribution to the energy equation, ϕ, is treated following Pino Martín et al. [11] approach. On the other hand, the terms involving SGS turbulent diffusion and SGS viscous diffusion are neglected, as they produce a negligible effect at low Mach numbers [11, 12].
To prevent spurious acoustic wave reflections at the far–field boundaries, a sponge–layer technique is employed. Hence, eq. 1 can be rewritten as:

$$\frac{\partial\bar{\mathbf{u}}}{\partial t} + \boldsymbol{\nabla}\cdot\mathbf{F}_c(\bar{\mathbf{u}}) - \boldsymbol{\nabla}\cdot\mathbf{F}_v(\bar{\mathbf{u}}, \boldsymbol{\nabla}\bar{\mathbf{u}}) + \boldsymbol{\nabla}\cdot\mathbf{F}_{SGS}(\bar{\mathbf{u}}, \boldsymbol{\nabla}\bar{\mathbf{u}}) = \sigma\left(\bar{\mathbf{u}}_{ref} - \bar{\mathbf{u}}\right) \tag{7}$$

where the non–physical source term added in right–hand side is aimed in damping the resolved quantities to a user defined reference value, $\bar{\mathbf{u}}_{ref}$, typically selected equal to the undisturbed free-stream conditions. The damping coefficient σ is defined as

$$\sigma = \sigma_0\left(\frac{L_{sp} - d}{L_{sp}}\right)^n \tag{8}$$

where L_{sp} is the sponge-layer width, d denotes the minimum distance from the outer boundary, σ_0 controls the damping strength, and n determines the shape of the sponge profile. In the present work, n is fixed to 2, while the sponge strength and layer width are selected according to the procedure described by D'Alessandro et al. [13].

2.2 Filter widths

As regards the spatial filter width, our baseline approach relies on least-square (LSQ) length scale (Δ_{lsq}) proposed by Trias et al. [3]:

$$\Delta_{\mathrm{lsq}} = \sqrt{\frac{\boldsymbol{\Delta}_D\mathbf{G}^T\mathbf{G} : \boldsymbol{\Delta}_D\mathbf{G}^T\mathbf{G}}{\mathbf{G}^T\mathbf{G} : \mathbf{G}^T\mathbf{G}}} \tag{9}$$

where $\mathbf{G}$ is the velocity gradient tensor and $\mathbf{\Delta}_D = \mathrm{diag}(\Delta_x, \Delta_y, \Delta_z)$ is the length scales tensor. Using Green's theorem, the generic cell P velocity gradient components are calculated as follows:

$$\left(\frac{\partial u_i}{\partial x_j}\right)_P = \frac{1}{2V_P} \sum_{N \in N(P)} \left[(u_i)_P + (u_i)_N\right] S_{PN} \left(n_{x_j}\right)_{PN} \tag{10}$$

with $\left(n_{x_j}\right)_{PN} = \left[n_x, n_y, n_z\right]^T_{PN}$ denoting the outward–pointing unit normal vector to the interface PN which is shared by the cell P and its neighbour cell N. The term S_{PN} denotes the face area of the interface PN and $N\,(P)$ is the set of cell P's direct neighbours. Differently, the directional length scales are computed consistently with the discrete gradient operator:

$$(\Delta_{x_j})_P = \frac{2V_P}{\sum_{N \in \mathcal{N}(P)} S_{PN} |n_{x_j}|_{PN}} \,. \tag{11}$$

The Δ_{lsq} filter provides a flow–dependent length scale that accounts for both cell anisotropy and the local velocity gradient direction. This approach has been demonstrated to significantly reduce the sensitivity of LES results to spanwise resolution on anisotropic grids, [4].
In our computational experience, we have observed that the above-mentioned formulation for Δ_{lsq} is sensitive to the adopted computing hardware due to non-deterministic floating–point accumulation on GPUs. To mitigate this sensitivity, we propose a new bounded definition of the filter width that combines a metric–based estimate with a lower bound proportional to the smallest local grid spacing. Specifically, we define the spatial filter width as $\mathbf{\Delta}_D = \Delta\,\mathbf{I}$

$$\Delta = \max\left(\Delta_g,\ C_\Delta \min(\Delta_x, \Delta_y, \Delta_z)\right) \tag{12}$$

and $\Delta_g = \left(\Delta_x \Delta_y \Delta_z\right)^{1/3}$. Moreover, the directional filter width used to build the two Δ branches is defined as

$$\left(\Delta x_j\right)_P = \frac{2V_P}{\max\left(\sum_{N \in \mathcal{N}(P)} S_{PN} \left|n_{x_j}\right|_{PN},\ \epsilon V_P^{2/3}\right)} \,. \tag{13}$$

In the following, the constant C_Δ is set equal to 1.3, while ϵ is a small positive parameter chosen as 10^{-6}. The cube-root operation introduced in the definition of Δ is deliberately employed to mitigate the amplification of floating–point accumulation errors, effectively reducing their impact by one third. However, this formulation may lead to excessively small values of the diagonal entries of the tensor $\mathbf{\Delta}_D$, which can result in an overproduction of SGS turbulent kinetic energy. To prevent this unphysical behavior, a realizability constraint based on the minimum grid spacing, $\min\left(\Delta_x, \Delta_y, \Delta_z\right)$ is introduced. This constraint ensures that the resulting filter width does not fall below the smallest physically meaningful length scale imposed by the grid resolution.

2.3 Numerical approximation

The governing equations described above are discretized in space using a standard cell-centered unstructured finite-volume method. Convective fluxes are evaluated using Pirozzoli's second-order accurate, energy-conserving scheme [14], which preserves kinetic energy in the inviscid incompressible limit. This property is particularly important in LES, where excessive numerical dissipation may overwhelm the physical subgrid-scale dissipation. Time integration is performed using a five-stage, fourth-order accurate explicit Runge–Kutta scheme with 2N compact storage, [15]. This scheme supports Courant numbers up to unity, [13], thereby exploiting the excellent parallel scalability of explicit time integration without imposing excessively restrictive time-step limitations.

3. Filter–width implementation strategies

The filter–width formulations discussed in Sec. 2 were implemented in `caaspuma`, a compressible density–based solver developed within the SPUMA framework, [7]. SPUMA preserves the OpenFOAM finite–volume data layout and exposes parallel execution through portable backend kernels. This choice is deliberately minimally invasive, but it also means that several geometrical operators retain the face–based structure of the original OpenFOAM implementation. The evaluation of the directional length scales in eq. 11 is therefore a useful test case, since each face contributes to the two adjacent control volumes and the final cell–centred quantity is an accumulation of projected face areas.

In the following, the standard face–based GPU implementation is denoted as *LSQ–GPU (face–based)*. The deterministic cell–based variant is denoted as *LSQ–GPU (cell–based)*. Finally, the bounded formulation is denoted as *LSQ–maxmin–GPU*. This last implementation uses the same face–based accumulation pattern as *LSQ–GPU (face–based)*, but replaces the unbounded directional tensor by the scalar maxmin filter defined in eq. 12. When the same bounded formulation is executed on the CPU reference backend, it is labelled *LSQ–maxmin–CPU* in the following.

3.1 Face–based accumulation

The face–based algorithm follows the natural OpenFOAM ownership addressing. The loop is parallelized over mesh faces. For each face, the absolute Cartesian components of the face area vector are added to the directional sums of the owner cell and, for internal faces, to those of the neighbour cell. On GPUs, different threads can update the same cell simultaneously; the race condition is removed by using `foamAtomic::AtomicAdd`. The relevant part of the implementation is reported in Listing 1.

Listing 1: Face–based accumulation used by LSQ–GPU (face–based).

```
auto faceLambda = [=](label faceI)
{
  const vector& fa = faceAreas_p[faceI];
  const scalar  ax = Foam::mag(fa.component(0));
  const scalar  ay = Foam::mag(fa.component(1));
  const scalar  az = Foam::mag(fa.component(2));

  const label ownerCell = owner[faceI];
  foamAtomic::AtomicAdd(dxyz_p[ownerCell].component(0), ax);
  foamAtomic::AtomicAdd(dxyz_p[ownerCell].component(1), ay);
  foamAtomic::AtomicAdd(dxyz_p[ownerCell].component(2), az);

  if (faceI < nInternalFaces)
  {
    const label neighbourCell = neighbour[faceI];
    foamAtomic::AtomicAdd(dxyz_p[neighbourCell].component(0), ax);
    foamAtomic::AtomicAdd(dxyz_p[neighbourCell].component(1), ay);
    foamAtomic::AtomicAdd(dxyz_p[neighbourCell].component(2), az);
  }
};
foamExecutor exec;
exec.parallelFor(faceLambda, mesh.nFaces());
```

This implementation is compact and directly mirrors the original finite–volume face loop. Its drawback is that the order of the partial sums is determined by GPU thread scheduling. Since floating–point addition is not associative, the final value of the denominator in eq. 11 may depend slightly on the backend and hardware. These differences are very small at the level of the geometrical sums, but they enter the LSQ filter through $\mathbf{\Delta}_D = \mathrm{diag}(\Delta_x, \Delta_y, \Delta_z)$ and are then propagated to the SGS viscosity through eq. 6. The *LSQ–maxmin–GPU* implementation keeps this same face–based accumulation but computes a bounded scalar filter width after the projected face–area sums have been accumulated, as shown in Listing 2.

Listing 2: Bounded maxmin filter used by LSQ–maxmin–GPU after face–based accumulation.

```
const scalar V   = V_p[cellI];
const scalar V23 = Foam::pow(V, scalar(2.0/3.0));
```

```
const scalar eps = 1e-6;

const scalar sx = max(x, eps*V23);
const scalar sy = max(y, eps*V23);
const scalar sz = max(z, eps*V23);

const scalar dx = 2*V/sx;
const scalar dy = 2*V/sy;
const scalar dz = 2*V/sz;
const scalar delta_g   = Foam::cbrt(dx*dy*dz);
const scalar delta_min = Foam::min(dx, Foam::min(dy, dz));
const scalar delta     = max(delta_g, 1.3*delta_min);

dist_p[cellI].xx() = delta;
dist_p[cellI].yy() = delta;
dist_p[cellI].zz() = delta;
```

3.2 Cell–based CSR accumulation

To separate the effect of the filter formulation from the effect of the accumulation strategy, a cell–based implementation was introduced. This version uses the same non–limited LSQ definition as *LSQ–GPU (face–based)*, but replaces the atomic face loop by a deterministic summation over the faces of each cell. It relies on a Compressed Sparse Row (CSR) representation of the mesh cell–to–face connectivity, implemented in SPUMA in the `fvMeshCsrAddressing` class.

The class `fvMeshCsrAddressing` is built lazily from OpenFOAM's `lduAddressing`. Two CSR views are available. The arrays `offsets`, `indices`, and `signs` contain internal faces only, while `offsetsAll`, `indicesAll`, and `signsAll` also include boundary faces. The filter–width computation uses the latter view, because boundary faces contribute to the projected area sums in eq. 11. For a cell c, its face entries are stored in the contiguous range `indicesAll[offsetsAll[c]:offsetsAll[c+1]]`. The companion array `signsAll` stores $+1$ for owner orientation and -1 for neighbour orientation; this information is useful for flux operators, although it is not needed here because only absolute face–area components are accumulated, but it can be useful in case we want to compute the gradient with a cell-based approach.

The construction follows the usual CSR pattern. First, the number of entries per cell is counted from `ownerStartAddr` for owner appearances, from `upperAddr` for neighbour appearances, and from the boundary patch `faceCells()` lists for boundary faces. Second, a prefix sum converts these counts into `offsetsAll`. Finally, a write–head array fills the flat `indicesAll` and `signsAll` arrays. The persistent arrays are allocated with `poolSwitch(1)`, so the storage is compatible with SPUMA device kernels and can be accessed inside `foamExecutor` loops without an additional host–device copy.

With this addressing, the GPU kernel is launched over cells rather than faces. Each thread owns one cell, loops over its local CSR face list, and accumulates the three projected face–area sums in private registers before writing the final values to memory. No two threads write to the same cell entry, and no atomic operation is required, as shown in Listing 3.

Listing 3: Cell–based deterministic accumulation used by LSQ–GPU (cell–based).

```
const auto& csrAddr = mesh.csrAddr();
const auto offsets  = csrAddr.offsetsAll().cbegin();
const auto indices  = csrAddr.indicesAll().cbegin();

auto cellLambda = [=](label cellI)
{
  scalar ax = 0.0;
  scalar ay = 0.0;
  scalar az = 0.0;

  for (label k = offsets[cellI]; k < offsets[cellI + 1]; ++k)
  {
    const label faceI = indices[k];
    const vector& fa = faceAreas_p[faceI];
    ax += Foam::mag(fa.component(0));
    ay += Foam::mag(fa.component(1));
```

```
    az += Foam::mag(fa.component(2));
  }

  dxyz_p[cellI].component(0) = ax;
  dxyz_p[cellI].component(1) = ay;
  dxyz_p[cellI].component(2) = az;
};
foamExecutor exec;
exec.parallelFor(cellLambda, mesh.nCells());
```

The comparison between the two strategies is summarized in Table 1. The face–based algorithm has the advantage of using the native OpenFOAM ownership addressing and of being directly applicable to many existing operators in SPUMA. Its drawback is the non–deterministic accumulation required by shared cell updates on GPUs. The cell–based CSR algorithm removes this source of perturbation for the directional length scales, at the cost of an additional mesh addressing structure and a slightly less direct mapping to the original face–loop formulation. In the present work, the cell–based implementation of the standard LSQ filter is used to verify that the observed GPU sensitivity originates from the atomic accumulation path rather than from a change in the physical model.

Table 1: Filter–width implementation strategies compared in `caaspuma`.

Implementation	Loop entity	Role in this work
LSQ–GPU (face–based)	Face with `atomicAdd`	Non–limited GPU baseline
LSQ–GPU (cell–based)	Cell with CSR	Non–limited deterministic reference
LSQ–maxmin–GPU	Face with `atomicAdd`	Bounded robust formulation

4. Results

The present analysis focuses on the flow past a circular cylinder at a Reynolds number based on the cylinder diameter, Re = 3900, [8, 9], and a free-stream Mach number of 0.2. This configuration represents a canonical benchmark for the assessment of LES methodologies, [16, 17, 18, 19], owing to its pronounced sensitivity to both numerical and modeling errors. The flow is characterized by the laminar separation of the shear layers from the cylinder surface, followed by transition to turbulence in the near wake, see Fig. 1 for a representation.

The computational domain extends $38D$ in the radial direction from the cylinder centre and πD in

Figure 1: Vortical structures visualized by the Q-criterion, with the numerical Schlieren field displayed on the vertical plane.

the spanwise direction. The same fully structured O–type grid employed by D'Alessandro et al. [4] was

adopted, with $N_\theta = 300$, $N_r = 330$, and $N_z = 48$ cells in the circumferential, radial, and spanwise directions, respectively. Computational cells were clustered near the cylinder surface, and the dimensionless height of the first wall-adjacent cell was set to 10^{-3}. A sponge–layer technique was adopted as a non–reflecting boundary treatment. In the present configuration, the sponge layer extends $13D$ from the outer boundary, corresponding to a dimensionless thickness $L_{sp}f/c_\infty \simeq 0.5$, where f is the vortex-shedding frequency and c_∞ is the free-stream speed of sound. Following the recommendations of Mani et al. [20], the target damping level was set to $\eta_{target} = 40$ dB.

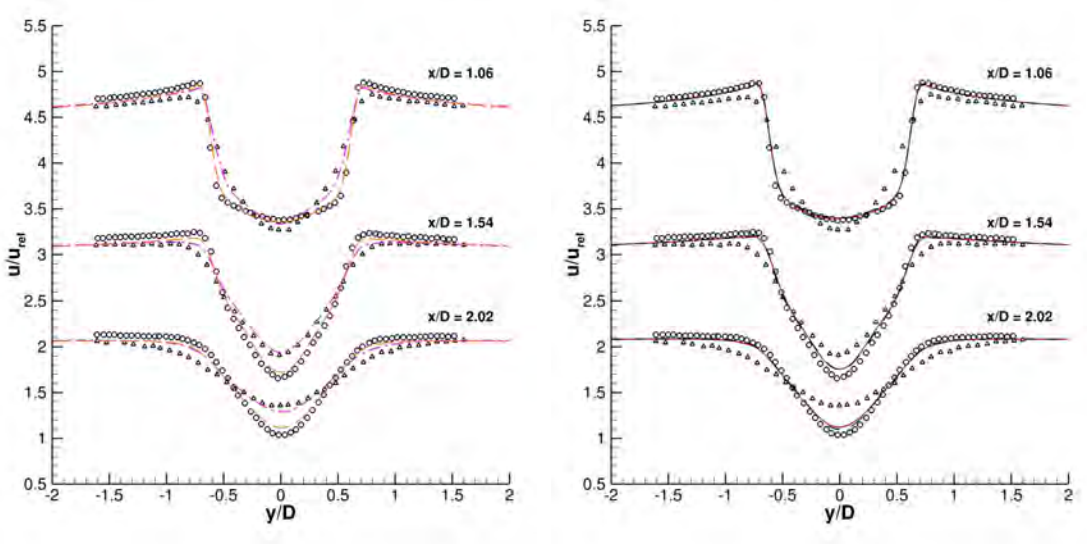
A well-documented feature of this benchmark is the sensitivity of the near-wake topology to numerical and modeling parameters, [21]. The mean streamwise velocity profile in the very near wake (at $x/D = 1.06$) can exhibit either a U–shaped or V–shaped character. The U–shaped profile has been confirmed as the correct physical behavior, while the V–shaped profile has been associated with premature laminar-to-turbulent transition in the separated shear layers, either due to upstream disturbances in experiments or excessive numerical dissipation in simulations, [9].

The results presented in the following section were obtained on two different computing architectures: NVIDIA A100 GPUs available on the LEONARDO supercomputer at CINECA (Italy) and Intel Xeon Platinum 8952+ CPUs available on the CRESCO8 supercomputer at ENEA (Italy).

As shown in Fig. 2, the *LSQ–GPU (face–based)* strategy yields velocity profiles that exhibit a pronounced V-shaped behavior at $x/D = 1.06$, rather than the expected U-shaped profile observed in reference data and CPU-based simulations, [4]. This discrepancy is not directly attributable to the hardware itself, but rather to the sensitivity of the filter–width formulation to small numerical perturbations arising during the evaluation of the spatial metrics. In the face-based implementation, the directional length scales defined by eq. 11 are evaluated through `atomicAdd` operations, which introduce a non–deterministic ordering of floating–point updates and consequently small round–off perturbations. Although these perturbations are negligible in absolute terms, the standard LSQ mapping can significantly amplify them. The amplification becomes particularly severe when the velocity-gradient tensor, $\mathbf{G}$, approaches a rank–deficient state, as typically occurs within separated shear layers. Under these conditions, small perturbations affecting the spatial metrics can propagate through the SGS closure and ultimately influence the resolved flow field. As a consequence, the *LSQ–GPU (face–based)* formulation yields a shortened shear layer and a reduced separation bubble, consistent with an artificially early laminar–turbulent transition, as shown in Fig. 5. This behavior is further confirmed by Fig. 4, which reveals a clear overestimation of the mean resolved streamwise Reynolds stresses with respect to the reference data. In contrast, Fig. 2 also demonstrates that the spatial filter-width formulation proposed in eq. 12 yields velocity profiles in excellent agreement with the reference solution on both CPU and GPU architectures. The effectiveness of the proposed approach is also confirmed by observing the mean transverse velocity (see Fig. 3). The proposed formulation introduces a bounded dependence of the filter width on the spatial metrics, thereby reducing the sensitivity of the SGS closure to floating-point perturbations. In particular, the combination of cube-root scaling and the max-min constraint prevents small perturbations arising during the evaluation of the directional length scales from being significantly amplified by the filter-width operator. As a consequence, the resulting SGS viscosity exhibits a substantially reduced sensitivity to non-deterministic floating-point accumulations, leading to improved consistency across different computing architectures.

To distinguish between perturbations generated by the accumulation strategy and those amplified by the filter-width formulation, an additional set of simulations was performed using the *LSQ–GPU (cell–based)* implementation strategy described in Sec. 3. By eliminating the non–deterministic atomic accumulations associated with the face–based approach, the cell–based implementation isolates the source of the perturbations entering the SGS model while preserving the original LSQ definition. The resulting solution closely approaches the CPU reference, demonstrating that a significant fraction of the observed architecture dependence originates from the `atomicAdd` accumulation path. On the other hand, *LSQ–maxmin–GPU* formulation instead targets the modelling side of the problem, reducing the sensitivity of the SGS closure to the residual floating–point perturbations that may still arise in other operators, such as velocity–gradient evaluation. This combined analysis supports the central point of the paper: in LES on GPU–accelerated architectures, filter–width robustness depends on both the mathematical definition

of the filter and the implementation strategy used to evaluate its geometrical ingredients.



(a) LSQ–GPU (face–based), LSQ–GPU (cell–based) (b) LSQ–maxmin–GPU, LSQ–maxmin–CPU

Figure 2: Mean stream–wise velocity in the wake region.

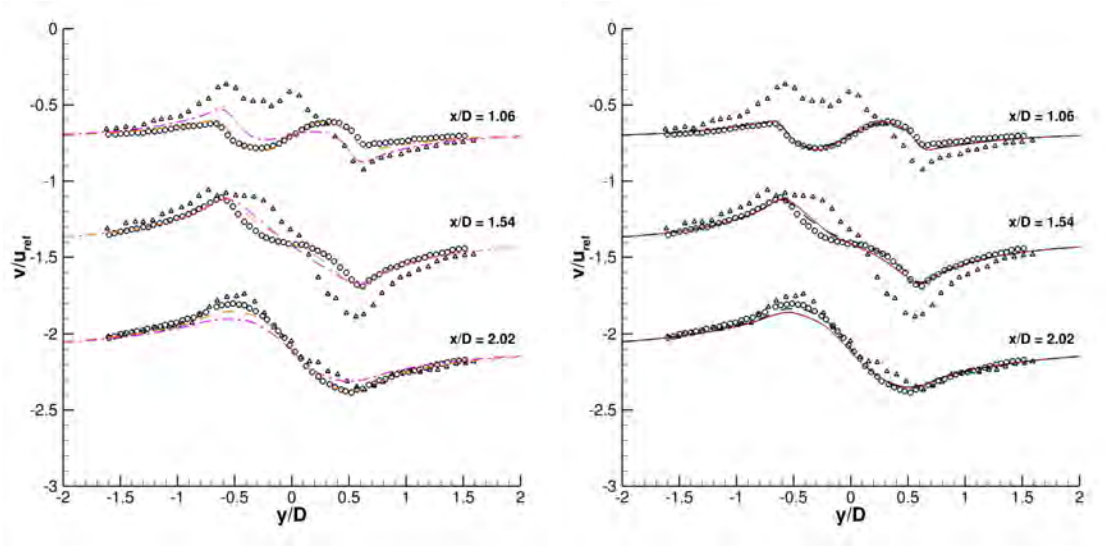
Legend: -·-·- LSQ–GPU (face–based), - - - LSQ–GPU (cell–based), ——— LSQ–maxmin–GPU, - - - LSQ–maxmin–CPU, ○ Exp. Parnaudeau et al., △ Exp. Lourenco and Shih.

5. Conclusions

This work investigates the impact of the mathematical formulation of the LES spatial filter–width on the robustness and portability of LES across different hardware architectures. In particular, relevant attention has been focused on the behavior of the LSQ filter-width formulation when deployed on massively parallel GPU platforms, where non-associative floating-point reductions are unavoidable. The numerical results obtained for the flow past a circular cylinder at Re = 3900 demonstrate that the original LSQ formulation is highly sensitive to floating–point accumulation errors when implemented in a face-centric GPU framework. To address this issue, a robust reformulation of the spatial filter–width has been proposed. The new definition combines a metric-based estimate with a realizability constraint tied to the minimum grid spacing. The proposed formulation successfully restores consistency between CPU and GPU-based LES results, yielding mean velocity profiles and wake topologies in excellent agreement with reference experimental data and CPU simulations.
The present analysis further reveals that implementation strategy and filter-width formulation address different aspects of the problem. While deterministic cell-based accumulations reduce the level of perturbations entering the SGS model, the bounded max–min formulation controls their amplification through the filter-width operator itself. Robust LES methodologies for heterogeneous HPC systems therefore require both perturbation-aware implementation strategies and filter-width formulations with bounded sensitivity to the spatial metrics. Overall, the present results suggest that numerical robustness and hardware portability should be regarded as intrinsic requirements of LES modeling, on par with physical accuracy.

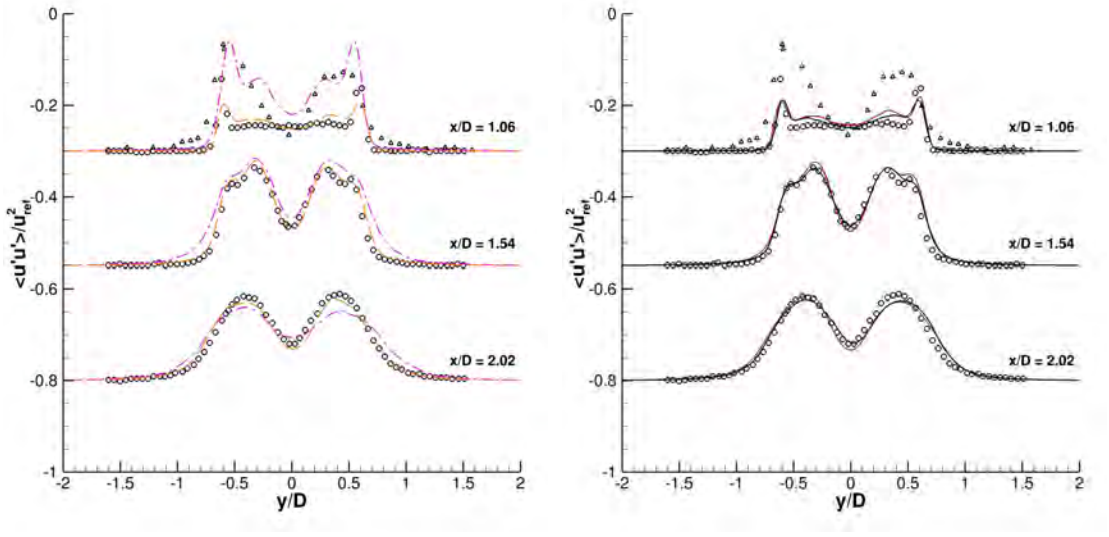
Acknowledgements

We acknowledge the EuroHPC Joint Undertaking for awarding this project access to the EuroHPC supercomputer LEONARDO, hosted by CINECA (Italy) and the LEONARDO consortium through



(a) LSQ–GPU (face–based), LSQ–GPU (cell–based) (b) LSQ–maxmin–GPU, LSQ–maxmin–CPU

Figure 3: Mean transverse velocity in the wake region. For the legend details, see the caption for Fig. 2



(a) LSQ–GPU (face–based), LSQ–GPU (cell–based) (b) LSQ–maxmin–GPU, LSQ–maxmin–CPU

Figure 4: Mean resolved stream–wise Reynolds stresses in the wake region. For the legend details, see the caption for Fig. 2

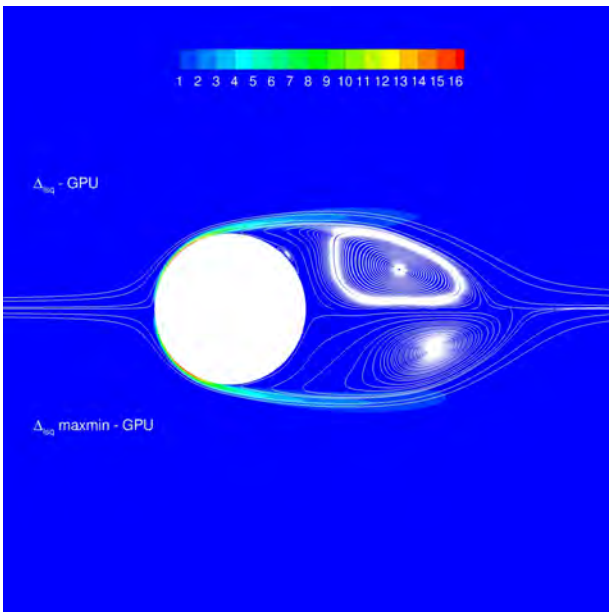


Figure 5: Mean velocity streamlines superimposed to mean span–wise vorticity.

an EuroHPC Development Access call. Part of the computing resources and the related technical support used for this work have been provided by CRESCO/ENEAGRID High Performance Computing infrastructure and its staff, [22]. CRESCO/ENEAGRID High Performance Computing infrastructure is funded by ENEA, the Italian National Agency for New Technologies, Energy and Sustainable Economic Development and by Italian and European research programmes, see http://www.cresco.enea.it/english for information.

References

[1] S. Ghosal. An analysis of numerical errors in large-eddy simulations of turbulence. *Journal of Computational Physics*, 125:187–206, 1996.

[2] P. Sagaut. *Large Eddy Simulation for Incompressible Flows*. Springer, 2006.

[3] F. X. Trias, A. Gorobets, M. H. Silvis, R. W. C. P. Verstappen, and A. Oliva. A new subgrid characteristic length for turbulence simulations on anisotropic grids. *Physics of Fluids*, 29(11):115109, 2017.

[4] V. D'Alessandro, Y. Delorme, M. Falone, M. Wasserman, and R. Ricci. Assessment of a Flow-dependent Subgrid Characteristic Length for Large-Eddy Simulation on Anisotropic Grids. *Journal of Physics: Conference Series*, 2685:012008, 2024.

[5] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.

[6] J. Demmel and H. D. Nguyen. Fast reproducible floating-point summation. *IEEE Transactions on Computers*, 64(7):2061–2074, 2015.

[7] S. Bnà, G. Giaquinto, E. Fadiga, T. Zanelli, and F. Bottau. SPUMA: A minimally invasive approach to the GPU porting of OPENFOAM. *Computer Physics Communications*, 321:110009, 2026.

[8] L.M. Lourenco and C Shih. Characteristics of the plane turbulent near wake of a circular cylinder, a particle image velocimetry study, 1993. Published in: Beaudan, P. and Moin, P., Report No. TF62, Thermosciences Division, Department of Mechanical Engineering, Stanford University.

[9] P. Parnaudeau, J. Carlier, D. Heitz, and E. Lamballais. Experimental and numerical studies of the flow over a circular cylinder at Reynolds number 3900. *Physics of Fluids*, 20(8):085101, 2008.

[10] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics*, 3(7):1760–1765, 07 1991.

[11] M. Pino Martín, U. Piomelli, and G. V. Candler. Subgrid-scale models for compressible Large-Eddy

Simulations. *Theoretical and Computational Fluid Dynamics*, 13:361–376, 2000.

[12] E. Garnier, N. Adams, and P. Sagaut. *Large Eddy Simulation for Compressible Flows*. Springer, Dordrecht, 2009.

[13] V. D'Alessandro, M. Falone, and R. Ricci. Direct computation of aeroacoustic fields in laminar flows: Solver development and assessment of wall temperature effects on radiated sound around bluff bodies. *Computers & Fluids*, 203:104517, 2020.

[14] S. Pirozzoli. Numerical methods for high-speed flows. *Annual Review of Fluid Mechanics*, 43:163–194, 2011.

[15] C. A. Kennedy, M. H. Carpenter, and R. M. Lewis. Low-storage, explicit Runge-Kutta schemes for the compressible Navier-Stokes equations. *Applied Numerical Mathematics*, 35(3):177–219, 2000.

[16] Breuer, M. Large eddy simulation of the subcritical flow past a circular cylinder: Numerical and modeling aspects. *International Journal for Numerical Methods in Fluids*, 28(9):1281 – 1302, 1998.

[17] A. G. Kravchenko and P. Moin. Numerical studies of flow over a circular cylinder at Re=3900. *Physics of Fluids*, 12(2):403–417, 2000.

[18] J.G. Wissink and W. Rodi. Numerical study of the near wake of a circular cylinder. *International Journal of Heat and Fluid Flow*, 29(4):1060 – 1070, 2008.

[19] D. A. Lysenko, I. S. Ertesvåg, and K. E. Rian. Large-Eddy Simulation of the Flow Over a Circular Cylinder at Reynolds Number 3900 Using the OpenFOAM Toolbox. *Flow, Turbulence and Combustion*, 89:491–518, 2012.

[20] A. Mani. Analysis and optimization of numerical sponge layers as a nonreflective boundary treatment. *Journal of Computational Physics*, 231(2):704–716, 2012.

[21] P. Beaudan and P. Moin. Numerical experiments on the flow past a circular cylinder at sub-critical Reynolds number. Technical Report TF-62, Center for Turbulence Research, Stanford University, 1994.

[22] F. Iannone et al. CRESCO ENEA HPC clusters: a working example of a multifabric GPFS Spectrum Scale layout. In *Proceedings of the 2019 International Conference on High Performance Computing & Simulation (HPCS)*, pages 1051–1052, Dublin, Ireland, 2019.